UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA INDUSTRIAL

SELECCIÓN DE LAS VARIABLES PSICOSOCIALES QUE CARACTERIZAN EL DESEMPEÑO DE FUNCIONES EJECUTIVAS EN SITUACIONES COTIDIANAS, A TRAVÉS DE UN MODELO DE PREDICCIÓN ETARIA

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL INDUSTRIAL

JOSÉ IGNACIO CANTO GAJARDO

PROFESOR GUÍA:
JUAN DOMINGO VELÁSQUEZ SILVA

MIEMBROS DE LA COMISIÓN:
ROCÍO BELÉN RUIZ MORENO
FERNANDO ANTONIO HENRIQUEZ CHAPARRO

SANTIAGO DE CHILE
2021

RESUMEN DE LA MEMORIA PARA OPTAR
AL TÍTULO DE INGENIERO CIVIL INDUSTRIAL
POR: **JOSÉ IGNACIO CANTO GAJARDO**
FECHA: 2021
PROF. GUÍA: JUAN DOMINGO VELÁSQUEZ SILVA

SELECCIÓN DE LAS VARIABLES PSICOSOCIALES QUE CARACTERIZAN EL DESEMPEÑO DE FUNCIONES EJECUTIVAS EN SITUACIONES COTIDIANAS, A TRAVÉS DE UN MODELO DE PREDICCIÓN ETARIA

Una Disfunción Ejecutiva (DE) es una alteración en alguna habilidad cognitiva, y puede ser progresiva en el tiempo, lo que significa que a mayor demora en el tratamiento de esta condición, más se acentúan los daños y se hacen más difícil de revertir. Una DE podría implicar déficits a la hora de consentir informadamente, en el juicio, y en la capacidad que tienen las personas de planificar y cumplir ciertos objetivos. Es por esto que es tan importante el correcto y rápido diagnóstico de Disfunción Ejecutiva. Sin embargo, los medios convencionales de diagnóstico tienen la gran limitación de poseer un bajo nivel ecológico, es decir, no son extrapolables a casos o situaciones cotidianas. El proyecto Neuronat desarrollado por la Facultad de Medicina de la Universidad de Chile y el Web Intelligence Centre propone una nueva herramienta de evaluación y diagnóstico de DE. Sin embargo, no se conoce el conjunto de variables psicosociales que dado un contexto cotidiano, representan de mejor manera una Disfunción ejecutiva.

Es por esto que en el presente trabajo de título se realizó un estudio de las variables obtenidas desde una plataforma llamada Neuronat, que sitúa a sus usuarios en un contexto cotidiano virtual, quienes deben resolver una serie de tareas en específico. Se sigue la metodología KDD (Knowledge Discovery in Databases), y utilizando técnicas de Feature Engineering, Feature Selection y Machine Learning, se seleccionó distintos conjuntos de variables que se utilizaron para entrenar varios modelos de clasificación.

El mejor modelo obtenido tiene un 100 % de Accuracy y Recall para la muestra de entrenamiento y un 67 % en la muestra de test, logrando resultados mucho mejores que el caso base de selección aleatoria de clases (25 %). A pesar de lo anterior, debido a las condiciones poco controladas en la cual se tomaron los datos, se cree que estos no son de calidad, por lo tanto los resultados obtenidos no son de fiar en su totalidad, aunque si se tienen ciertos indicios de que con los datos obtenidos se puede detectar en cierta medida el desempeño de funciones ejecutivas.

A mi abuela que ahora nos cuida desde el cielo

Gracias por todo

Agradecimientos

Recuerdo cuando empecé con la memoria estaba emocionado por escribir los agradecimientos ya que representaban lo último que quedaba de unos largos 7 años de estudio en la U, pero decidí dejarlo para el final, y bueno, aquí estamos, casi un año después, un año difícil en donde se fue uno de los pilares de mi familia, mi abuela, mi madrina, mi segunda madre, pero también un año donde llegaron nuevos integrantes a mi familia, me enteré de que voy a ser tío, y estoy más feliz que nunca con mi polola, mis amigos y mi familia.

Primero me gustaría agradecer a mis compañeros del WIC, uno de los mejores lugares en los que he trabajado con gente increíble como Don Felipes, la Fran, Pangui, la Kathy, Los 2 Cris, Don Albert, Carlitos, el Mauri, el Dani, la Gema y el Seba. Me gustaría agradecer especialmente al Vixon por reclutarme como ayudante y auxiliar en tics, e introducirme al mundo del WIC, además de ayudarme en varios aspectos técnicos con mi trabajo de título. También agradecer al profe Juan Velásquez por darme la oportunidad de ser parte de ese grupo, por confiar en mí y hacerme auxiliar de Web Intelligence y por guiarme como profe guía. Finalmente, especial gracias a Rocío que aparte de ser la mejor jefa en el WIC me acompañó durante todo mi trabajo de título como profesora coguía, me aconsejó, me ayudó y me hizo dar lo mejor de mí.

Luego, me gustaría agradecer a mis amigos que me acompañaron durante la carrera e hicieron de la Universidad un lugar al que deseaba ir, gracias al Nico que desde Bachi hasta el último día de industrias (y más) estuvo ahí, igual que la Flo y la Mabel, y a mis amigos que conocí después en industrias como el Agus, la Ani, la AnLLi, Mellado, la Carito, el Mota, la AB, el Fipe, Luquitas, la Nicole, Pablito, la Tere, el Yerko y Marquito. Los quiero mucho a todos y espero seamos amigos para siempre <3.

Por último pero no menos importante, me gustaría agradecer a mi familia, a todos mis tíos, mis tías, mis primos y primas que más que primos somos como hermanos. Gracias a mi abuela que ahora descansa en paz, por ser la raíz de ese gran árbol familiar. Gracias a mi papá, por darme la mejor educación que puede haber, porque nunca me faltó nada y por sus sabios consejos. Gracias a la Glo también por ser como una hermana mayor para mí, preocuparse por mí y cuidarme cuando era más chico. Gracias a la Nati, mi mejor amiga y polola, que me hace el más feliz, y me ha ayudado en cada aspecto de mi vida con todo su amor y su cariño, gracias por todo mi amor <3. Gracias a mi hermano por acompañarme durante toda mi vida, por darme consejos, jugar conmigo (como darme el control del super Nintendo desenchufado) y por estar siempre ahí. Finalmente, me gustaría agradecer a la mejor madre que alguien podría desear, la mujer más fuerte que conozco y la persona que más amo en este mundo, gracias por todo lo que ha hecho por mí y por hacer de mi la persona que soy.

Tabla de Contenido

Índice de Tablas

Índice de Ilustraciones

Capítulo 1

Introducción

En el presente capítulo se introducirá el contexto bajo el cual se desarrolla el trabajo de título presentando tanto la situación actual en el mundo, el proyecto bajo el cual se enmarca el trabajo de título y las instituciones participantes de dicho proyecto. Posteriormente se describirá el problema a detalle, indicando las causas y consecuencias de este, además del impacto correspondiente desde una perspectiva económica social y privada. Finalmente se presenta la hipótesis de solución, el objetivo general y los objetivos específicos del trabajo de título, para terminar con la metodología de trabajo que se seguirá y los resultados esperados.

1.1. Antecedentes generales

Para el año 2018, se estimó que más de 60 millones de personas sufrieron de un traumatismo craneoencefálico (TEC) en el mundo [1]. Dentro de las posibles consecuencias que conlleva sufrir un TEC está el hecho de generar una Disfunción Ejecutiva (DE), es decir, puede significar la pérdida o disminución en el desempeño de distintas habilidades cognitivas tales como la habilidad de consentir informadamente, el juicio y la voluntad, entre otras [2], llevando finalmente a una pérdida de la independencia personal. Además, la Disfunción Ejecutiva puede ser progresiva e incapacitante con el tiempo, razón por la cual se hace fundamental el pronto diagnóstico de una DE, para así comenzar con un tratamiento adecuado para el paciente lo antes posible, que permita disminuir el efecto de la Disfunción Ejecutiva o en el mejor de los casos sanarla [2].

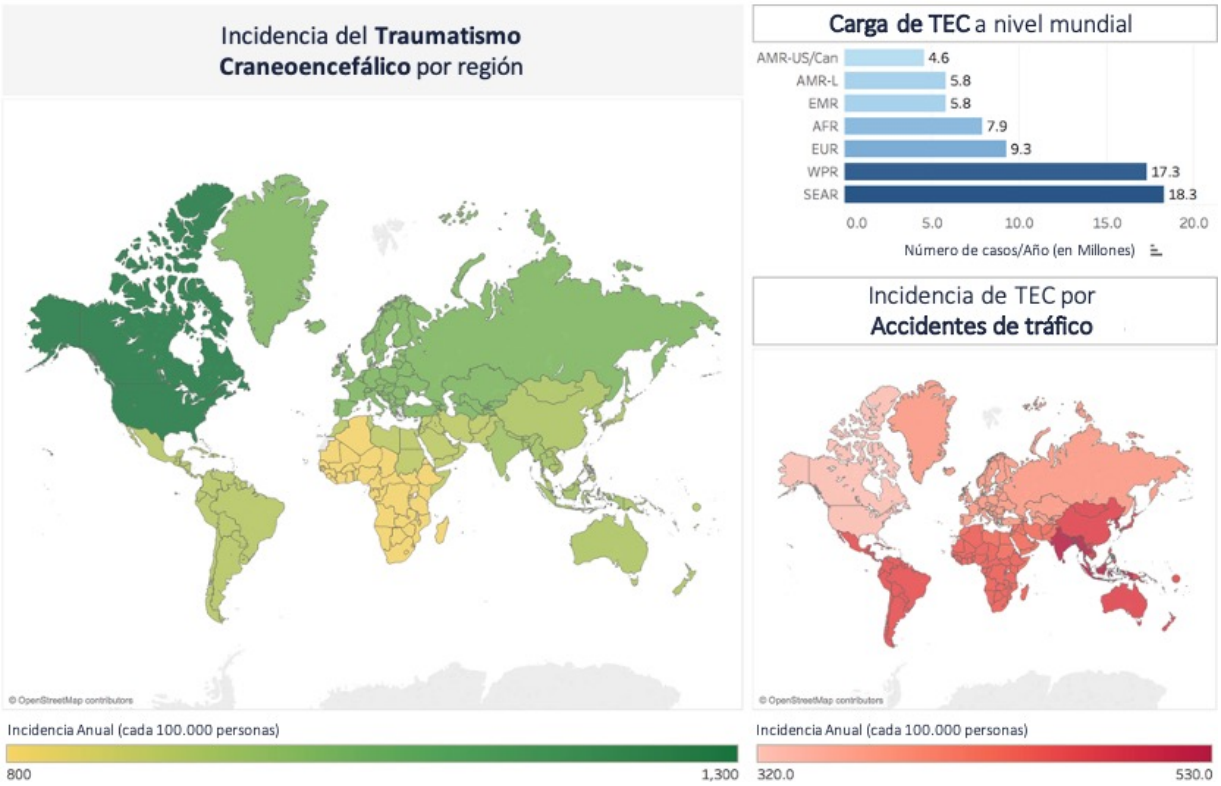


Figura 1.1: Incidencia de traumatismo craneoencefálico en el mundo.
Fuente: Dewan & Rattani (2018) [1]

Una DE se produce por un trastorno o alteración en una función ejecutiva (capacidad que tienen las personas para planificar y cumplir objetivos) [2] [3]. Estas funciones ejecutivas se desarrollan durante la etapa de crecimiento de las personas, y a medida que se envejece van disminuyendo su desempeño, es decir, existe una relación directa entre la edad y el desempeño de las funciones ejecutivas [4]. Sin embargo, existen varias causas que pueden producir una DE, como por ejemplo Parkinson, depresión, abuso de sustancias, enfermedad de Lyme, autismo, entre otras, lo que aumenta el universo total de pacientes que sufren de una DE [4].

Es importante definir brevemente el concepto de validez ecológica. Este corresponde a la capacidad que se tiene de extrapolar o generalizar desde los resultados o el comportamiento observado en un laboratorio (o entorno controlado) hacia el comportamiento natural en el mundo real [5]. De acuerdo con un equipo clínico de investigadores de la Universidad de Chile, una de las principales falencias en los métodos actuales de diagnóstico de DE es que no se miden los aspectos de la vida cotidiana de los pacientes, por lo tanto se tiene poca ecología. La cotidianeidad involucra varias habilidades cognitivas que pueden verse dañadas con la presencia de una Disfunción Ejecutiva. Algunos de los test utilizados en la evaluación y diagnóstico de DE son el "test del reloj", en el cual los pacientes dibujan un reloj con una hora en específico, que evalúa principalmente habilidades visoconstructivas (capacidad de realizar construcciones bidimensionales o tridimensionales), pero de la cual se puede obtener información sobre la capacidad de planificación ejecutiva; el "trail making test parte B" en donde los pacientes deben unir secuencialmente pares de números y letras en orden creciente,

y evalúa la flexibilidad cognitiva; o el "test de fluencia verbal", el cual consiste en que el paciente diga todas las palabras que se le ocurran que partan con una letra en específico [2], que evalúa flexibilidad cognitiva espontánea, que junto a las demás habilidades cognitivas presentadas forman parte de las funciones ejecutivas. Lo anterior lo reafirma la Dra. Brenna McDonald, quien dice que lamentablemente, el ambiente altamente estructurado en el que se hace la evaluación de los pacientes puede minimizar el efecto de algunos déficits ejecutivos [6].

Por lo tanto, un paciente puede tener un gran rendimiento en la entrevista y en los tests que se le realizan, pero en otros contextos cotidianos que quedan fuera de la consulta médica, tienen un pobre desempeño. Lo anterior demuestra la baja validez ecológica que tienen los actuales métodos de evaluación y diagnóstico de Disfunciones Ejecutivas. Es por esto que desde la Facultad de Medicina de la Universidad de Chile en conjunto con el Web Intelligence Centre del Departamento de Ingeniería Industrial de la Universidad de Chile (DII) se está desarrollando un nuevo método de evaluación para apoyar en el diagnóstico de Disfunción Ejecutiva, que pone al paciente en situaciones cotidianas, para así evaluar el desempeño de las funciones ejecutivas de manera ecológica que no se pueden evaluar en una consulta médica tradicional. Este método de evaluación consiste en un videojuego serio (posteriormente se explica en detalle en que consiste un videojuego serio, pero a grandes rasgos es un videojuego que se usa con fines distintos al ocio), en donde el paciente tendrá que tomar el rol de un mozo y atender distintos clientes en un restorán.

Hoy en día, existen varios estudios sobre la incorporación de modelos computaciones de Machine Learning [1] (ML) en la evaluación y en el diagnóstico de distintas patologías médicas, en varias áreas de la medicina, como por ejemplo, el diagnóstico de etapas tempranas de la enfermedad de Alzheimer utilizando un modelo de ML que en base a imágenes de resonancias magnéticas permite discriminar entre Alzheimer, discapacidad cognitiva leve y deterioro natural por envejecimiento [8]. Otro ejemplo es el diagnóstico de ansiedad y depresión en niños utilizando herramientas de Machine Learning y datos generados de un test que se le realiza a este tipo de pacientes [9]. Existe bastante documentación sobre este tipo de facultades que permite el ML en el área de la medicina, sin embargo, no se puede encontrar mucho sobre el uso de técnicas y algoritmos de ML en el diagnóstico de Disfunción Ejecutiva.

Es por esto, que dados los antecedes que existen, es que se cree posible desarrollar un modelo computacional de Machine Learning, que permita apoyar en el diagnóstico de Disfunciones Ejecutivas en personas que hayan sufrido un traumatismo craneoencefálico, ya que existe documentación sobre problemas parecidos con resultados positivos. Sin embargo, dado lo nuevo e innovador de la investigación es que no existe mucho conocimiento sobre qué variables médicas y psicosociales de las personas permiten modelar el fenómeno del desempeño de las funciones ejecutivas en una situación cotidiana, para poder crear así un modelo que pueda clasificar si una persona sufre efectivamente de DE o no. Debido a esto, en el presente documento se hace un estudio de un conjunto de variables recopiladas del proyecto Neuronat, para de esta manera poder entender que tipos de comportamientos y variables en específico reflejan un deterioro en las funciones ejecutivas de las personas.

[1] Machine learning es un campo dentro de la inteligencia artificial, que consiste en algoritmos capaces de aprender y adaptar su estructura y "conocimiento" en base a variables que describen un cierto fenómeno [7].

1.2. Características institucionales

El presente trabajo de título se llevó a cabo bajo la tutela de 2 instituciones, el Web Intelligence Centre (WIC) y la Facultad de Medicina de la Universidad de Chile, en particular, el área de neurociencia del campus oriente de la facultad.

El WIC es un centro de investigación del departamento de industrias de la Facultad de Ciencias Físicas y Matemáticas (FCFM), dedicado al estudio, investigación e implementación de tecnologías de información, análisis de datos y Data Science, en particular, para el sector de la salud en Chile. Gran parte de los proyectos que se realizan en el WIC se hacen en conjunto con la Facultad de Medicina de la Universidad de Chile, como los proyectos Akori, Kefuri, Delirium y Neuronat. Complementando lo anterior, se encuentra su misión: "Queremos ser un actor relevante en el área del Data Science y sus aplicaciones en la salud. Para ello, contaremos al 2022 con al menos 3 proyectos transferidos en distintos centros de salud" [10].

El fundador y actual director académico del WIC es el profesor Juan Velásquez, y en conjunto con la directora ejecutiva Rocío Ruiz y el director de tecnología Felipe Vera lideran el WIC. Además, se cuenta con la doctora Flavia Guiñazú que asesora en los proyectos desde el punto de vista médico, y lidera investigaciones dentro del centro. Finalmente, se encuentra una serie de ingenieros de proyectos, diseñadores, data scientists, un jefe comercial, un ingeniero de sistemas, un desarrollador web y una encargada de recursos humanos, llegando así a contar con 19 trabajadores [10].

Por otro lado, la Facultad de Medicina de la Universidad de Chile es una de las 14 facultades que tiene la Universidad. Imparte distintas carreras como medicina, odontología, enfermería entre otras, y consta de 5 campus ubicados en Santiago. En particular, se está trabajando con el área de ciencias neurológicas y el área de psiquiatría y salud mental, que están ubicadas en el campus oriente de la facultad, en el Hospital del Salvador [11]. En las áreas mencionadas hay académicos e investigadores que trabajan en distintos proyectos de investigación, y es con estos investigadores con los que se trabajó. Andrea Slachevsky es la líder del proyecto desde el área de medicina.

1.3. Proyecto Neuronat

El trabajo de memoria se enmarca en el proyecto Neuronat. Este proyecto es un FONDEF (código ID 18I10113), liderado por la Facultad de Medicina de la Universidad de Chile, en conjunto con el WIC y con una empresa de desarrollo y diseño llamada Tinet. Cada institución aporta en un aspecto único al proyecto.

Antes de explicar a detalle en que consiste el proyecto, es importante definir ciertos conceptos médicos. A continuación se presenta una lista de estos conceptos:

- Discapacidad psicosocial: se define como la presencia de limitaciones que sufre una persona para desenvolverse de forma independiente en las actividades de la vida diaria (relacionarse con otras personas, trabajo, tener pareja, etc). [12]
- Función ejecutiva: corresponde a la capacidad cognitiva que tienen las personas para poder declarar un objetivo, y establecer una serie de pasos para lograr ese objetivo.

Algunas actividades cognitivas relacionadas con las funciones ejecutivas pueden ser el hecho de iniciar una actividad, la autorregulación, la monitorización de tareas o la organización en el tiempo y en el espacio [3].

- Disfunción Ejecutiva: Una alteración en las funciones ejecutivas se conoce como Disfunción Ejecutiva (DE). Dicho deterioro de las funciones ejecutivas puede tener consecuencias de distinta magnitud en los pacientes que lo sufren, tales como la incapacidad de dar un consentimiento, perdida o disminución del juicio y la voluntad, que al final se traduce en una pérdida de la independencia del paciente [2].

Con lo anterior ya definido, se procede a explicar en qué consiste el proyecto. Éste, busca desarrollar una nueva forma, más ecológica, eficiente y económica de detectar una Disfunción Ejecutiva (DE), y esto lo hace por medio de un Serious Game, es decir, de un videojuego serio. Esta categoría de videojuegos se utiliza con fines distintos al ocio o entretenimiento de las personas que lo consumen. Puede ser utilizado en el rubro de la educación, de las ciencias, en la atención médica, planificación urbana, política e ingeniería según el observatorio de innovación educativa de la universidad de México, Tecnológico de Monterrey [13]. En el caso del proyecto, el Serious Game se utilizará como instrumento de evaluación de pacientes, aportando así más información a los médicos especialistas en el diagnóstico de Disfunción Ejecutiva.

El contexto del videojuego es un restorán en donde el jugador asume el rol del mozo. Este cuenta con distintas tareas que involucran distintas capacidades cognitivas que permiten evaluar las funciones ejecutivas. Dentro de las tareas que tendrá que realizar el jugador se encuentra: dar la bienvenida a los clientes, tomar la orden de los clientes, ir a buscar los platos a la cocina, servir los platos a los clientes, ofrecer postre, cobrar los pedidos a los clientes, priorizar la atención de una mesa u otra, recordar ciertas actividades como entregar un vaso con agua después de un cierto tiempo y recordar ofrecerles un dulce a los clientes una vez que estos paguen la cuenta. Cada una de las actividades mencionadas está relacionada con dominios cognitivos específicos, por lo tanto, se evaluará el desempeño del jugador en base a las decisiones que tome durante el juego, asignándole una calificación por dominio cognitivo a los pacientes, además de un índice de Disfunción Ejecutiva que represente una probabilidad de padecer alguna Disfunción Ejecutiva.

1.3.1. Facultad de Medicina - Universidad de Chile

Como se mencionó anteriormente, cada institución involucrada en Neuronat tiene una labor en particular en el desarrollo de este. En el caso de la Facultad de Medicina, quienes son los líderes del proyecto, estos definen tanto el contenido de la plataforma, como los dominios cognitivos que se deben medir, toman las decisiones finales sobre los aspectos estéticos del Serious Game, determinan lo que debe dar como resultado la plataforma, además de encargarse de los aspectos administrativos del proyecto, como la redacción de informes para entregarlos a FONDEF, programación de plazos, y la contratación de personal extra en caso de ser necesario.

Es importante mencionar que para crear un videojuego (o Serious Game) se necesita de varias personas que trabajen en conjunto. Por una parte están los encargados de definir el

tema y el contenido tanto en diálogos como en lo que se debe visualizar, y en la historia que se debe contar. Esta labor se podría considerar como director o productor de un videojuego, además de guionista [14]. Como se mencionó anteriormente, la Facultad de Medicina cumple dicho rol. Por otro lado, se debe contar con un desarrollador que programe la plataforma, es decir, que cree el Serious Game desde un programa en el computador. Finalmente, debe haber un diseñador encargado de los aspectos visuales. En síntesis, las labores se pueden resumir en esas 3, aunque se pueden agregar más roles dentro del desarrollo de un videojuego como se muestra en el libro de Rich Newman, *Cinematic Game Secrets for Creative Directors and Producers: Inspired: inspired techniques from industry legends* [14].

El equipo de medicina se hace cargo de las 3 labores mencionadas, realizando el contenido, la programación y el diseño en sí de la plataforma de evaluación. Es importante destacar que el videojuego está en una fase inicial de su implementación, y al momento de realizar el trabajo de título se veía como se muestra en la Figura 1.2.

Figura 1.2: Bosquejo del Serious Game. *Fuente: Proyecto Neuronat*

Finalmente, será el equipo de medicina el que utilice la plataforma para evaluar pacientes en caso de que el proyecto cumpla con los estándares esperados, por lo tanto, además de ser dueños de la plataforma, serán usuarios de ella.

1.3.2. Web Intelligence Centre

Por otro lado, está el Web Intelligence Centre (WIC) dentro del proyecto. La labor que cumple el WIC en Neuronat es la de hacer el análisis de datos de la plataforma, y el desarrollo de los modelos que permitan evaluar al jugador en base a sus variables. Además, el WIC ha apoyado en tanto la formulación del proyecto, como en el desarrollo de este. En primer lugar, se creó una serie de diagramas de flujo que representan los distintos caminos que puede seguir un jugador en el Serious Game. Esto se realizó en conjunto con el equipo clínico, con quienes se definió la historia que se contaría en el videojuego, los distintos escenarios posibles, los actores de cada escenario, y como interactuaban dichos actores. En la Figura 1.3 se puede apreciar uno de los diagramas que se hicieron, siguiendo la notación BPMN.

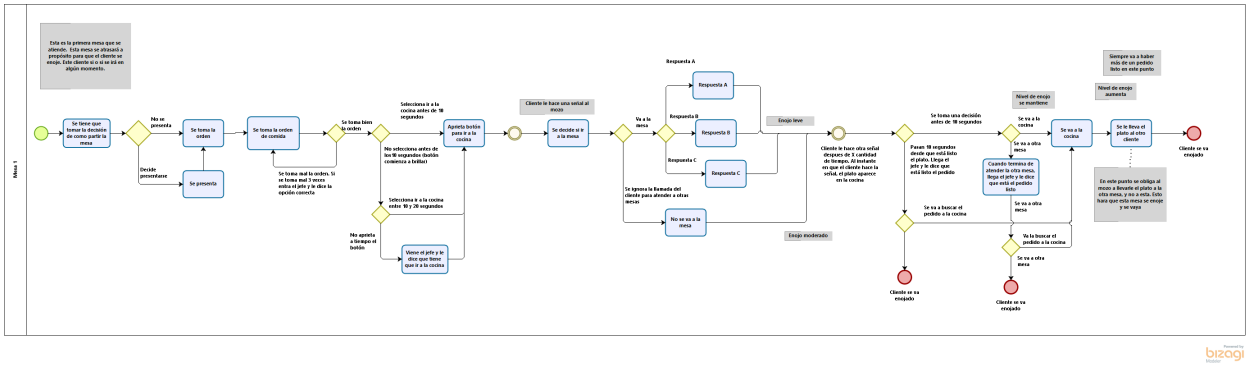


Figura 1.3: Diagrama que muestra las distintas actividades que realiza el mozo en una mesa. *Fuente: Web Intelligence Centre*

Una vez finalizada la creación de flujos se definió la estructura de las variables que tendría la plataforma, lo cual se realizó en conjunto con el equipo clínico para corroborar que era importante medir y cómo. En conclusión, se miden los tiempos de ejecución de tareas, las instancias en las que se resuelve un problema (variables categóricas, si se resuelve en primera instancia se asigna un uno, si se resuelve en segunda instancia un dos, hasta llegar a un máximo de tres intentos por actividad), variables binarias sobre si se hace algo o no, y variables cuantitativas como la cantidad de errores al momento de resolver una actividad.

Es importante mencionar que los flujos realizados para modelar los distintos caminos que puede tomar el videojuego se segmentan tanto por mesa como por hito. Esto quiere decir que existe un flujo por cada una de las 5 mesas que tiene el videojuego, y cada flujo se divide en distintos hitos, en donde un hito representa una etapa de la atención del mozo en dicha mesa. Algunos hitos son la toma de orden, servir los platos en la mesa y realizar la transacción final, por mencionar algunos. Por lo tanto, las variables se pueden ordenar en 3 niveles distintos. A modo general, está el nivel de la mesa, luego se encuentran los hitos de la mesa, y finalmente las variables que corresponden a dicho hito de dicha mesa. En la figura 1.4 se pueden apreciar los 3 niveles que se tienen con unas variables que representan distintos momentos del juego. Esta misma estructura siguen las 306 variables del videojuego.

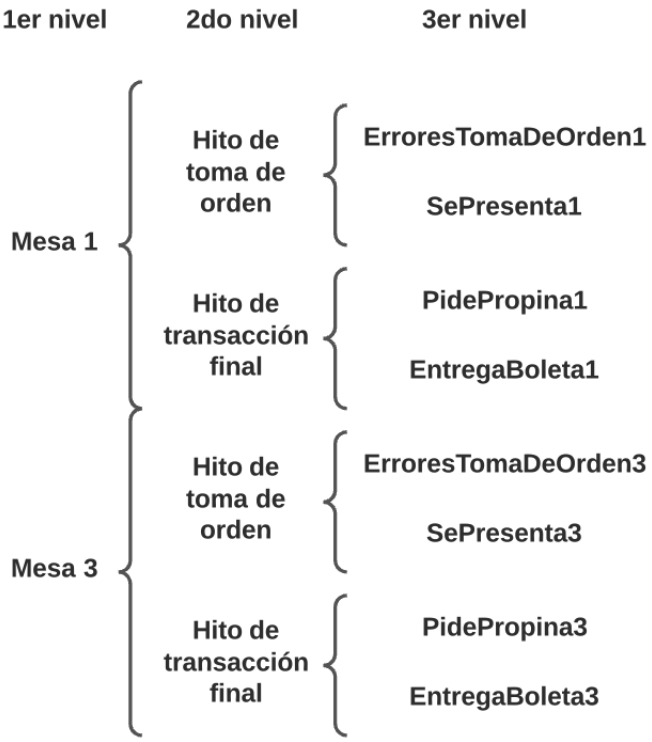


Figura 1.4: Niveles de las variables producidas por la plataforma Neuronat *Fuente: Elaboración propia*

Por lo tanto, el WIC es el encargado de hacer el procesamiento de los datos que se generen con la plataforma Neuronat. Esto implica en primera instancia hacer un análisis exploratorio de los datos. Luego, debido a la poca cantidad de datos que se tendrá (40 registros en total), se debe encontrar la manera de aumentar el volumen de datos, es decir, se tiene que hacer una creación de datos virtuales. Luego, una vez que se tenga un volumen considerable de datos, se procede a entrenar distintos modelos que permitan obtener los resultados que se requieren desde el equipo de medicina, que son una evaluación por cada dominio cognitivo, y una predicción que indique si el paciente posee una Disfunción Ejecutiva con un cierto porcentaje de precisión.

1.3.3. Descripción del problema

Actualmente, los mecanismos que se tienen para evaluar DE son una serie de tests como el test del reloj o el trail making test parte B (los cuales serán explicados en detalle posteriormente), además de una serie de entrevistas tanto a los pacientes como a los acompañantes de los pacientes [2]. Dichos mecanismos de evaluación se realizan en el contexto de una consulta médica, lo cual condiciona a los pacientes a un contexto en específico, lo que puede minimizar el efecto de algunos déficits ejecutivos de acuerdo con la Dra. Brenna McDonald [6].

Como se ha mencionado anteriormente, las funciones ejecutivas son aquellas habilidades cognitivas, o capacidades cognitivas para poder declarar un objetivo y establecer una serie de pasos para lograr dicho objetivo [3]. Las FE se pueden asociar a habilidades en específico,

como el hecho de iniciar una actividad, la autorregulación, organización en el tiempo y en el espacio entre otras [3]. Existen 3 sistemas funcionales ejecutivos, encargados de ciertas funciones ejecutivas en específico: sistema dorsolateral, sistema orbital y sistema medial; y estos son mediados o regulados por la corteza prefrontal del cerebro [2].

En cada sistema funcional ejecutivo se pueden encontrar distintas funciones ejecutivas, por lo tanto, dependiendo de qué sistema esté alterado se manifestarán ciertas DE. Por ejemplo, si el daño se encuentra en el sistema dorsolateral, entonces la persona afectada puede presentar apatía, falta de iniciativa para la acción, o desinterés. En lo práctico, esto se puede apreciar en que alguien que padezca de este tipo de daño no pueda iniciar o terminar una actividad como dibujar un círculo, o también tiene la necesidad de utilizar lo que encuentre como por ejemplo un lápiz. Por otro lado, si el daño se encuentra en el sistema orbital la persona afectada puede presentar comportamiento pueril, egocéntrico, hipersexualidad y desinhibición, lo que se puede reflejar en una dificultad para controlar impulsos. Finalmente, si el daño se encuentra en el sistema medial puede verse dañado el control emocional y/o atencional de las personas, lo que puede llevar a que un paciente que presente dicho daño no sienta el dolor emocional de perder a un ser querido, por ejemplo [2].

Por lo tanto, la DE sufrida por los pacientes depende de la ubicación del daño, y existen distintas funciones ejecutivas que se pueden ver alteradas, mientras que otras se mantienen relativamente sanas. De acuerdo con el equipo de investigadores de la Facultad de Medicina de la Universidad de Chile, encargados del proyecto Neuronat, los métodos actuales de evaluación de Disfunción Ejecutiva no tienen un buen desempeño al evaluar la presencia de disfunciones ejecutivas en actividades que se realizan en la vida cotidiana, es decir, tienen una baja validez ecológica, lo cual se relaciona con lo enunciando por la Dra. Brenna McDonald. Esto puede significar que un paciente que presenta daños puede tener un gran desempeño en los tests que se realizan en la consulta médica, pero no se le va a detectar que presenta problemas en las actividades cotidianas [15], generando así una Discapacidad Psicosocial (limitaciones para desenvolverse de forma independiente en las actividades de la vida diaria) [12]. Por lo tanto, considerando las herramientas que existen en la actualidad para diagnosticar DE (test de lápiz y papel y entrevistas con médicos), se identifica el siguiente problema: *Se tiene una medición deficiente de Disfunciones Ejecutivas en situaciones de la vida cotidiana*, ya que hay personas que no son diagnosticadas con DE, siendo que si la padecen (los instrumentos de evaluación fallaron).

El proyecto Neuronat busca evaluar la presencia de una DE en ámbitos relacionados con la vida cotidiana de manera más eficaz y eficiente que los métodos actuales de evaluación, es decir, buscan hacerse cargo del problema, pero ¿Por qué es importante obtener métodos más eficaces y eficientes?

Es fundamental evaluar el deterioro de alguna función ejecutiva, es decir, una Disfunción Ejecutiva, ya que esto puede implicar la perdida de la independencia personal para la persona que padece de esta DE. Actividades tan importantes como la planificación, organización, realizar actividades en paralelo, disminución del juicio y de la habilidad para tomar decisiones, problemas de atención y concentración, inflexibilidad e impulsividad, por mencionar algunas [4]. Incluso, la presencia de una DE tiene consecuencias sobre el juicio, la voluntad y la planificación, más graves en la independencia de los pacientes que el deterioro

de la memoria. Además, es importante la evaluación pronta de una DE, ya que el trastorno puede ser progresivo e incapacitante a medida que pasa el tiempo, por lo que tomar acción inmediata se hace de vital importancia para lidiar con dicho progreso del trastorno, y lograr revertirlo en la medida de lo posible [2]. Esto último se puede hacer mediante un tratamiento que consiste en la ingesta o supresión de medicamentos según corresponda, alimentación sana y la práctica de una vida social activa, reducción de distractores, ejecución de actividades simples, incorporación de complementos para la vida diaria (como temporizadores, alarmas, etc.) o la realización de tareas repetitivas para reforzar el área dañada [2][4]. Sin embargo, cada paciente requiere de un tratamiento único y diseñado exclusivamente para este, ya que, como se mencionó anteriormente, el problema que presente esa persona depende de donde se encuentra el daño en el cerebro [4].

Por otro lado, existen ciertas consecuencias económicas relacionadas con el mal diagnóstico de DE, o el tardío diagnóstico de este. Por todo lo mencionado anteriormente, se puede dar el caso en que se le diagnostique erróneamente a una persona la presencia de DE. Esto implica al paciente pagar el tratamiento asociado a este, además de la ingesta de medicamentos en caso de ser recetado por el médico tratante. Este tratamiento asociado, tiene un costo de más de 3 millones de pesos, según el equipo de médicos involucrados en el proyecto Neuronat. Por otro lado, como ya se ha mencionado anteriormente, el daño producido por DE puede ser progresivo e incapacitante, por lo tanto un tardío diagnóstico de DE acentuaría los síntomas sufridos por los pacientes y profundizaría el daño. Esto tiene claras consecuencias en la vida profesional de los afectados, ya que al sufrir una DE una persona puede sufrir problemas de planificación, organización, adaptación a cambios ambientales, seguir reglas o direcciones, entre otros [6], todas habilidades fundamentales a la hora de trabajar en cualquier contexto. Debido a lo anterior, es que las personas cuando sufren de una DE recurren a la toma de licencias médicas, ya que no pueden trabajar. Se entiende como licencia médica al derecho que tiene cada trabajador de ausentarse de su trabajo o reducir su carga laboral por motivo de una indicación médica [16]. Además del derecho a una reducción de la carga laboral, al afectado se le otorga una cifra de dinero que cubre las necesidades de este mientras no está trabajando, y funciona como un reemplazo a la remuneración habitual. Dicho pago que se le hace a la persona afectada se conoce como *Subsidio por Incapacidad Laboral (SIL)*, y las entidades encargadas de realizar ese pago son las Comisiones de Medicina Preventiva e Invalidez (COMPIN), las Unidades de Licencias Médicas, las Cajas de Compensación de Asignación Familiar (CCAF) y las Instituciones de Salud Previsional (ISAPRE), según corresponda el caso [17]. Por lo tanto, una DE diagnosticada tardíamente provoca un daño más profundo y complicado de tratar, lo que se traduce en una mayor toma de licencias médicas por los tratamientos más largos y el mayor tiempo de incapacidad para trabajar, lo que significa un mayor gasto en SIL.

Por lo tanto, el problema identificado: Se tiene una medición deficiente de Disfunciones Ejecutivas en situaciones de la vida cotidiana, tiene una serie de consecuencias directas que se podrían resumir en:

- Profundización del daño por parte de los pacientes.
- Tratamientos más largos y menos efectivos.
- Costos de tratamientos para pacientes mal diagnosticados.

- Aumento del uso de licencias médicas.

Al igual que las consecuencias, se puede encontrar una serie de causas directas de por qué existe el problema. Estas causas serían: Instrumentos convencionales de medición, contexto convencional de medición y la poca claridad sobre las variables que representan el fenómeno de la DE en situaciones cotidianas.

La primera causa hace referencia a las herramientas o test de medición que se utilizan, que dada la distinta documentación que existe, se pueden encontrar los siguientes: trail making test, COWAT (test de fluencia), Stroop y test de Wisconsin (WCST) [2], entre otros [4] [6]. Dichos tests son los más utilizados según la distinta documentación, y considerando que [6] tiene fecha de 2002, mientras que [4] tiene fecha de 2015, se tiene que en 15 años no ha habido una mayor actualización en los instrumentos de medición que se utilizan.

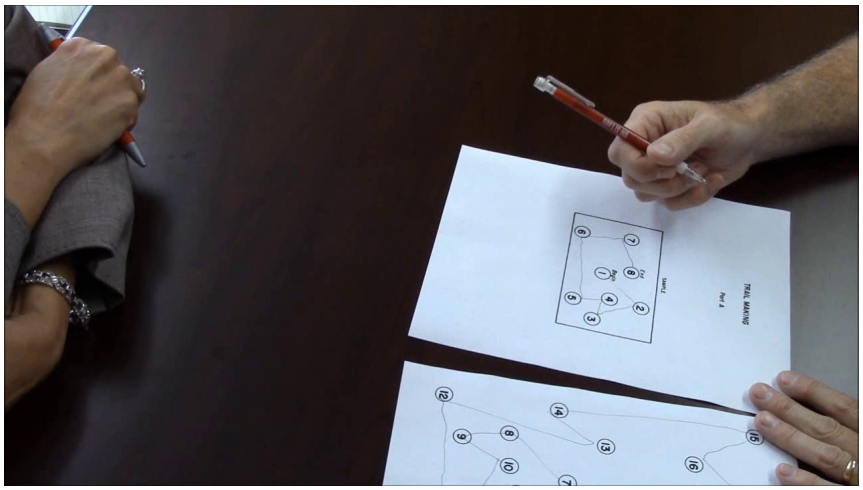


Figura 1.5: Imagen que muestra cómo se realiza el trail making test a un paciente. *Fuente: Thomas Meuser, University of New England, "Administration of the Trail Making Test, Parts A & B":* https://youtu.be/ybnhZCQ_Ji0

La segunda causa, contexto convencional de medición, hace referencia al contexto o entorno bajo el cual se realizan los test a los pacientes. Anteriormente se definió el concepto de validez ecológica como la capacidad que se tiene de extrapolar o generalizar desde los resultados o el comportamiento observado en un laboratorio (o entorno controlado) hacia el comportamiento natural en el mundo real [5]. Es más, Gerard Gioia define la validez ecológica en el contexto psicológico como la habilidad de generalizar resultados de experimentos controlados hacia eventos que ocurran de forma natural en el mundo real [18]. Entonces, bajo esa definición, se tiene que los clásicos tests mencionados anteriormente presentan una baja validez ecológica, ya que en una situación real, el paciente a la hora de resolver un cierto problema no sólo tiene la dificultad cognitiva del problema, sino que también una serie de estímulos que se dan por el contexto en el que está, teniendo así que resolver un problema multidimensional que considera los distintos estímulos presentes. En cambio durante un test clínico, dada la estructura tradicional que siguen estos, el médico tratante es el que le da al paciente la estructura, la organización, la guía y el plan a seguir para el óptimo rendimiento del paciente, quitando así cualquier multidimensionalidad y cotidianeidad que pueda

presentar el problema [18].

Finalmente, la tercera causa que se puede reconocer es la poca claridad que se tiene sobre las variables que representan el fenómeno de la DE en situaciones cotidianas. Como ya se ha planteado anteriormente, distintas funciones ejecutivas están asociadas a sistemas específicos del cerebro [2], sin embargo, identificar que función o habilidad ejecutiva está asociada a algún dominio es uno de los desafíos que tienen los neuropsicólogos (profesionales encargados de las estructuras del cerebro y el comportamiento humano, principalmente encargados de estudiar que conductas cambian al sufrir daños en el cerebro [19]). En parte, esto se debe a que las funciones ejecutivas no funcionan de manera independiente, sino que en conjunto para poder resolver problemas del día a día [18]. Si bien, se pueden asociar ciertos test a algunas funciones ejecutivas como lo hace el documento de Rabinovici [4], la falencia de estos es que evalúan aspectos acotados del paciente, no lo evalúan integralmente, lo que hace que pierda validez ecológica [18]. Por lo tanto, no existe un test, o un conjunto de test lo suficientemente ecológico que permita evaluar integralmente a los pacientes para poder diagnosticar las DE asociadas a aspectos de la vida cotidiana (aunque si se puedan diagnosticar clínicamente). Lo anterior implica que existe un gran número de pacientes que no son diagnosticados.

El principal enfoque que se ha tomado para intentar resolver el problema planteado, es la implementación de entornos virtuales o más conocidos como test de realidad virtual, que permiten simular situaciones reales para los pacientes, sin embargo, el gran problema de este tipo de soluciones es la poca factibilidad de implementar dichas soluciones, ya sea por temas logísticos, económicos o de infraestructura [20]. También se encuentran evaluaciones basadas en observaciones naturalistas, que consiste en observar a los pacientes mientras realizan actividades cotidianas reales. Esto si bien presenta una mayor validez ecológica tiene el mismo tipo de problemas que los entornos virtuales, que es poco factible de implementar dentro de un contexto clínico habitual [21].

En la Figura 1.6 se puede ver el árbol del problema, que resumen las consecuencias y las causas de problema identificado.

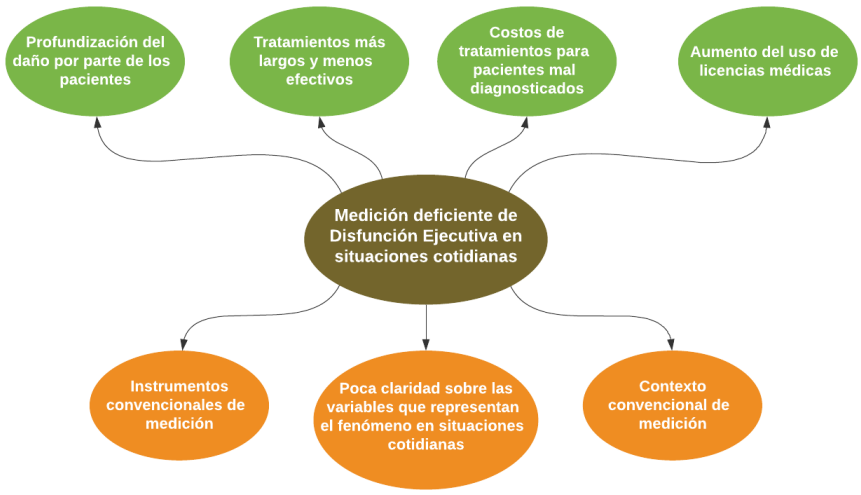


Figura 1.6: Árbol de problemas, que resume las consecuencias y las causas del problema identificado. *Fuente: Elaboración propia*

1.3.4. Impacto estimado

Las funciones ejecutivas se desarrollan durante la infancia y la adolescencia, y a medida que los seres humanos envejecen, estas van disminuyendo su desempeño[4]. Sin embargo, existen otras razones por las cuales una persona puede sufrir un trastorno en una función ejecutiva, tales como: enfermedades neurodegenerativas (Parkinson, Alzheimer), condiciones neurológicas (traumatismo craneoencefálico, tumor), condiciones psiquiátricas (depresión, esquizofrenia), condiciones médicas o intoxicación (abuso de sustancias, trastornos del sueño), infecciones (neurosifilis, enfermedad de Lyme) o condiciones de desarrollo (autismo, déficit atencional) [4]. En la Figura A.1 del anexo se encuentran varias posibles causas de una Disfunción Ejecutiva.

Entonces, una DE puede ser provocada por varias razones. En el caso del proyecto Neuronat, se selecciona la causa del traumatismo craneoencefálico por la relativa facilidad de obtener una muestra de pacientes que hayan sufrido un TEC.

A nivel mundial, se tienen más de 60 millones de casos de TEC al año, y la incidencia por cada 100.000 es de 939 globalmente hablando, es decir, un 1 % de la población aproximadamente sufre de TEC [1]. Además, se estima que en un 15 % de los casos leves de TEC se produce una DE, mientras que en casos moderados o severos este número asciende a un 65 % [22]. Existen variados reportes que indican la proporción de casos severos, moderados y leves de TEC en el mundo, sin embargo no se ha llegado a un consenso al respecto. Estos varían desde un 80 % como casos leves, y un 20 % para casos moderados y severos, hasta un 95 % de casos leves y un 5 % de casos moderados y severos [23].

En Chile por otro lado, las cifras cambian, teniendo alrededor de 200 hospitalizaciones asociadas a un TEC por cada 100.000 habitantes, en donde el 50 % de estas corresponde a TEC leves, un 25 % a moderados y el otro 25 % a casos severos[24]. Por lo tanto, la cantidad de afectados disminuye de un 1 % que se tiene en la población global a un 0,2 % en Chile, con respecto al total de la población. Sin embargo, se tiene una mayor tasa de casos graves y moderados, alcanzando un 50 % del total de los casos. Si se calcula la cantidad de DE que se producen con los datos que se tienen hasta el momento, de los 939 afectados que se tienen a nivel global (proporcionalmente hablando), se tendrían 235 personas con DE provocada por TEC, lo que equivale a un 0,24 % de incidencia a nivel mundial. Por otro lado, si se realiza el mismo cálculo para los datos que se tienen en Chile, se tendría que de las 200 personas que sufren de un TEC en el año, 80 desarrollarían una DE, llegando así a un 0,08 % de incidencia en Chile. Si consideramos que Chile tiene 19.458.310 habitantes según el Instituto Nacional de Estadísticas (INE) [25], con un 0,08 % de personas que sufren de DE se tendrían 15.500 personas aproximadamente que padecerían dicho problema al año.

Si se considera que el 1 % de esas 15.500 personas está erróneamente diagnosticado, haciendo referencia a una de las consecuencias enunciadas en la sección anterior, se tendrían 155 personas que tendrían que someterse al tratamiento, y si se fija el valor de dicho tratamiento en 3 millones de pesos como lo sugiere el equipo de investigadores del proyecto Neuronat, se tendría un gasto de 465 millones de pesos por parte de los pacientes en un año. Es importante calcular la perpetuidad de dicho valor, para así tener un estimado del costo actual de tener dichos costos indefinidamente [26]. La fórmula de perpetuidad es:

$$VP = \frac{FE}{i} \tag{1.1}$$

En la fórmula 1.1 VP corresponde al valor presente, FE al flujo efectivo, o en este caso al pago anual que deben realizar los pacientes, e i corresponde a la tasa de interés o descuento. A modo de cálculo se utiliza la tasa social de descuento que fija el Ministerio de Desarrollo Social que tiene un valor de 6 % [27]. Por lo tanto, utilizando la fórmula 1.1 se obtiene una perpetuidad de 7.750 millones de pesos.

Por otro lado, se tiene el otro aspecto económico mencionado en las consecuencias que se atribuyen al problema identificado, que son las licencias médicas. De acuerdo con los datos disponibles en la Superintendencia de Seguridad Social (SUSESO) [28], el año 2018 se autorizaron 895.548 licencias médicas para pacientes FONASA, bajo el diagnóstico de trastornos mentales, mientras que para pacientes ISAPRE de ese mismo grupo se autorizaron 264.088 licencias, llegando a un total de 1.159.636 licencias autorizadas a pacientes con trastornos mentales. Esa cantidad de licencias corresponden a 17.789.704 días autorizados, 14.815.256 para pacientes FONASA y 2.974.448 para pacientes ISAPRE. Además, se tiene el monto en dinero que conlleva esa cantidad de licencias y días, que es 257.472 millones de pesos para pacientes FONASA y 115.521 millones de pesos pacientes ISAPRE. Por lo tanto, si se calcula el costo promedio por día de licencia, tanto para pacientes FONASA y ISAPRE se tiene que un día de licencia cuesta 17.379 pesos y 38.838 pesos, respectivamente.

Lamentablemente, los reportes y estadísticas que la SUSESO pone a disposición del público no entrega el detalle de las enfermedades o trastornos mentales a los cuales está asociada cada licencia. Por lo tanto, se procede a tomar ciertos supuestos. En primer lugar, se toma el supuesto que el 1 % de las licencias autorizadas corresponde a pacientes que se les diagnosticó DE después de haber sufrido un TEC. Esto corresponde a 8.955 y 2.640, para FONASA e ISAPRE respectivamente. Ahora, se considera que el 10 % de estas licencias corresponde a pacientes a los que se les diagnostica tardíamente una DE. Por lo tanto, se tienen 895 licencias FONASA, y 264 licencias ISAPRE. Finalmente, se toma el supuesto de que el hecho de diagnosticar tardíamente una DE se traduce en 30 días extras de licencia.

Bajo los supuestos tomados, y los cálculos realizados, se tiene un gasto asociado al diagnóstico tardío de 774 millones de pesos al año, lo que llevado a perpetuidad da un valor de 12.903 millones de pesos, considerando el 6 % definido por el Ministerio de Desarrollo Social.

En síntesis, el costo estimado asociado al diagnóstico erróneo de DE es de 7.750 millones a perpetuidad, valor que es pagado por todos los pacientes. Mientras que el otro costo identificado, asociado a las entidades encargadas de pagar los SIL, es de 12.903 millones de pesos a perpetuidad.

1.3.5. Hipótesis de investigación

Como se mencionó en el apartado de Descripción del problema, se identificaron 3 causas directas del problema a abordar. Estas 3 causas son: instrumentos convencionales de medición, contexto convencional de medición y la poca claridad sobre las variables que representan la DE en situaciones cotidianas.

Sobre las primeras 2 causas, el proyecto Neuronat busca hacerse cargo de estas. Esto, proponiendo un nuevo instrumento de evaluación que será la plataforma que se está desarrollando, en la cual el paciente se sitúa dentro de un contexto cotidiano, independiente de que físicamente esté en una consulta clínica. Sin embargo, esto deja de lado la tercera causa, y es por esto que en el presente trabajo de título se aborda dicha causa.

Por lo tanto, en paralelo al trabajo que se desarrolla en el proyecto Neuronat, se investigó sobre las variables, o el conjunto de variables que mejor representa una Disfunción Ejecutiva en situaciones de la vida diaria. Para realizar esto se trabajó con las variables que genera el Serious Game Neuronat. Entonces, bajo la premisa de que Neuronat sitúa a los jugadores dentro de un contexto cotidiano, se espera que las variables describan las decisiones y los comportamientos de estos en dicho contexto. Además, las variables de la plataforma Neuronat fueron construidas por el equipo clínico detrás del proyecto, que conocen los test que se realizan a los pacientes y los distintos tipos de funciones ejecutivas que hay, los síntomas y comportamientos de los pacientes cuando existe cierto tipo de disfunciones ejecutivas, es decir, son expertos en el diagnóstico y tratamiento de DE, y se tomó todo esto en consideración a la hora de construir las variables. Por lo tanto, existen distintos tipos de variables, asociadas a ciertos dominios cognitivos o funciones ejecutivas, por lo que la hipótesis de investigación que se plantea es: *Es posible estimar el desempeño de las funciones ejecutivas y determinar cuáles son las variables que mejor caracterizan este desempeño en una situación cotidiana, con los datos obtenidos de la plataforma Neuronat.*

Para lograr determinar el efecto que tiene un conjunto de variables en específico se desarrolló un modelo que predice el segmento etario, en base a las variables seleccionadas de Neuronat. Pero, ¿Por qué predecir la edad? Varios estudios indican el efecto de la edad en el desempeño que tienen las funciones ejecutivas. Estas se desarrollan durante la infancia y adolescencia de las personas, y luego van disminuyendo su desempeño con la edad, en relación con la perdida de la funcionalidad prefrontal [4]. Por lo tanto, se puede encontrar una relación directa entre la edad y las funciones ejecutivas, ya que a mayor edad, se tiene peor desempeño de estas, lo que podría provocar efectos similares a los de una DE. Es más, McAlister & Schmitter-Edgecombe realizaron un estudio el 2013, en donde evalúan la diferencia en el desempeño de las funciones ejecutivas entre adultos jóvenes sanos y adultos mayores sanos (sin un diagnóstico de DE), y llegaron a la conclusión de que efectivamente los adultos mayores tienen un peor desempeño, resolviendo tareas específicas en más tiempo, secuenciando tareas de manera más pobre, y entregando resultados a problemas de peor calidad y con peor precisión [21]. Por lo tanto, es válido utilizar el segmento etario como indicador de desempeño de las funciones ejecutivas, teniendo que a una mayor edad, se encuentra un peor desempeño de estas, lo que a grandes rasgos puede asemejarse a una DE. Entonces, se entrenan varios modelos que predicen el segmento etario en base a distintos conjuntos de variables, los cuales están conformados según el estudio de las variables que se realizó, y el conjunto de variables que logró entrenar al modelo con mejores resultados según las métricas de *accuracy* (exactitud en español), *AUC* (área bajo la curva), *recall* (exhaustividad en español) y *F1-Score*[2], entrega el conjunto de variables que mejor caracteriza una DE en situaciones cotidianas.

[2] En el apartado de Marco conceptual se explica en qué consiste cada una de estas métricas de evaluación, pero a grandes rasgos estas métricas indican qué tan bien o mal predice un modelo según las veces que predijo correctamente y las veces que se equivocó.

Es importante destacar, que el encontrar un set o conjunto de variables relevantes y significativas que permitan caracterizar una DE en una situación cotidiana, no sólo sirve como conocimiento para el proyecto Neuronat, sino que podría llevar a nuevos estudios y experimentos que busquen nuevas maneras de diagnosticar (o incluso tratar) una DE en base a las variables propuestas, y no sólo para pacientes que hayan sufrido de un TEC, sino que a distintos pacientes que sufran de alguna de las enfermedades o condiciones que se muestran en la figura A.1.

1.4. Objetivos

A continuación se detallan cuáles son los objetivos del trabajo de título.

1.4.1. Objetivo general

Seleccionar el conjunto de variables psicosociales que permite caracterizar una Disfunción Ejecutiva en situaciones cotidianas para hacer más certero el diagnóstico de esta condición, y lograr reducir los tiempos de tratamiento y los costos asociados a estos.

1.4.2. Objetivos específicos

1. Ejecutar análisis del estado del arte, con el fin de contextualizar el problema a tratar, conocer que técnicas y métodos se utilizan para la resolución de dicho tipo de problemas, y que se ha realizado en casos similares.

2. Crear una base de datos con muestras reales y con datos simulados, para poder así aumentar la cantidad de registros que se tienen en la base de datos y ampliar el espectro de modelos que se puede implementar.

3. Realizar estudio de las variables obtenidas del Serious Game basado en técnicas de Feature Selection y Feature Engineering (explicadas en detalle en el marco conceptual) y desarrollar modelos de predicción de segmento etario utilizando las variables estudiadas.

4. Entregar el conjunto de variables que mejor caracteriza el desempeño de funciones ejecutivas.

5. Realizar una evaluación de impacto social y económico para concluir el trabajo de título, identificando a que poblaciones afectará positivamente los resultados del presente trabajo y en que monto.

Cabe destacar que el tercer objetivo es el más importante de la investigación, ya que lleva a responder la hipótesis planteada. Este estudio consistió en evaluar distintas métricas asociadas a las variables, como por ejemplo , las correlaciones, las varianzas, la información mutua, entre otras, para así ir realizando filtros en la base de datos original y finalmente quedar con un conjunto de variables que mejor estime el segmento etario al cual pertenece cada registro.

1.5. Marco conceptual

A continuación se definirán y explicarán ciertos conceptos, técnicas y algoritmos que se utilizarán en el presente trabajo de título. Se introduce y se detalla además el proceso KDD en el cual se basa la metodología aplicada en el trabajo de título.

1.5.1. Proceso KDD

El descubrimiento de conocimiento desde bases de datos (KDD, Knowledge Discovery in Databases) es una metodología o proceso, que permite descubrir patrones desde una fuente de datos. Esta metodología combina técnicas de bases de datos, inteligencia artificial, estadística, entre otros [31]. Por lo tanto, esta metodología se aplica a problemas en donde se tiene una gran cantidad de datos, ya sea en tamaño o en dimensionalidad[3], y dado ese tamaño no es factible que un humano analice ese volumen de datos, por lo que se utilizan técnicas computacionales de análisis, con el fin de obtener una base de datos más compacta y factible de interpretar por expertos. Algunos ejemplos del uso de la metodología KDD son [33]:

- Catalogar imágenes del espacio para un posterior análisis de los objetos que existen.
- Descubrir volcanes en Venus utilizando un mapa global del planeta completo.
- Interpretación de los datos del ADN humano, utilizando una base con 200.000 diferentes secuencias de proteínas

Existen distintos enfoques sobre cómo se aborda la metodología o proceso KDD. A continuación se presentan los pasos de la metodología según Biswas [34]:

1. Integración: consiste en recolectar datos de distintas fuentes, como por ejemplo, distintas bases de datos, ya sea internas de una empresa o públicas, e integrarlas y almacenarlas en un sólo lugar.
2. Selección: debido a que en el paso de Integración se recolectan datos de distintas fuentes, es probable que haya mucha información que no sea de utilidad para la investigación que se está realizando. Por lo tanto, en esta etapa se busca filtrar y seleccionar sólo los datos relevantes.
3. Limpieza/Preprocesamiento de datos: se tiene que verificar que los datos seleccionados estén correctamente estructurados, no tengan valores nulos, sean consistentes, etc. A este proceso se le conoce como la limpieza o preprocesamiento de los datos.
4. Transformación: en orden de poder aplicar técnicas de data mining o Machine Learning, se debe reestructurar los datos en un formato adecuado para trabajar.
5. Minería: se utilizan técnicas de data mining para analizar los datos y poder detectar patrones en estos.

[3] La dimensión de una base de datos puede verse como que tantas variables o atributos caracterizan un objeto o registro, por lo tanto, mientras más variables tenga un objeto, tendrá una dimensionalidad mayor [32]

6. Interpretación y evaluación: una vez obtenido los patrones de los datos, se procede a transformar estos en conocimiento para poder concluir sobre algún fenómeno en específico, o poder tomar decisiones en algún negocio.

7. Visualización: este último paso ayuda a entender de mejor manera los resultados obtenidos del proceso KDD, entregando herramientas visuales como gráficos y tablas.

A continuación, en la figura 1.7 se puede ver el resumen gráfico del proceso KDD, obtenido de [34].

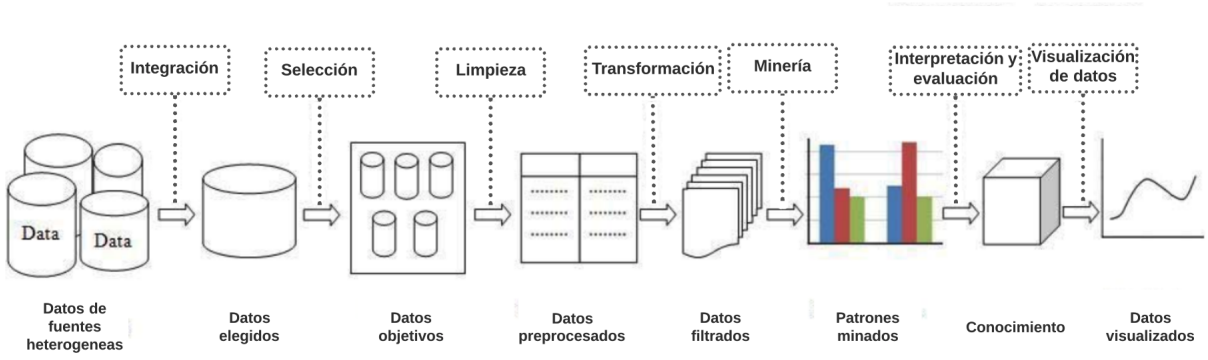


Figura 1.7: Imágen ilustrativa que muestra las distintas etapas del proceso KDD. *Fuente: (Biswas, Saha Bhattacharyya, 2016)* [34]

1.5.2. Bases de datos

En los últimos años el uso de Internet y la disponibilidad de acceder a este ha incrementado exponencialmente como se puede apreciar en la figura 1.8. Ramakrishnan & Gehrke (2000) mencionan en su libro que la cantidad de datos disponible en aquellos tiempos era gigantesca y que cada vez era mayor. Además, reconocen que los datos son un activo fundamental para las organizaciones, ya que permite generar información y es con la información que se toman decisiones [35]. Sin embargo, la presencia de cantidades grandes de datos, al punto de no poder procesarlos ni almacenarlos hace que pasen de ser un activo valioso a una carga [35]. Ramakrishnan & Gehrke definen las bases de datos como: "*colección de datos, que típicamente describe las actividades de una o más organizaciones relacionadas*" [35].

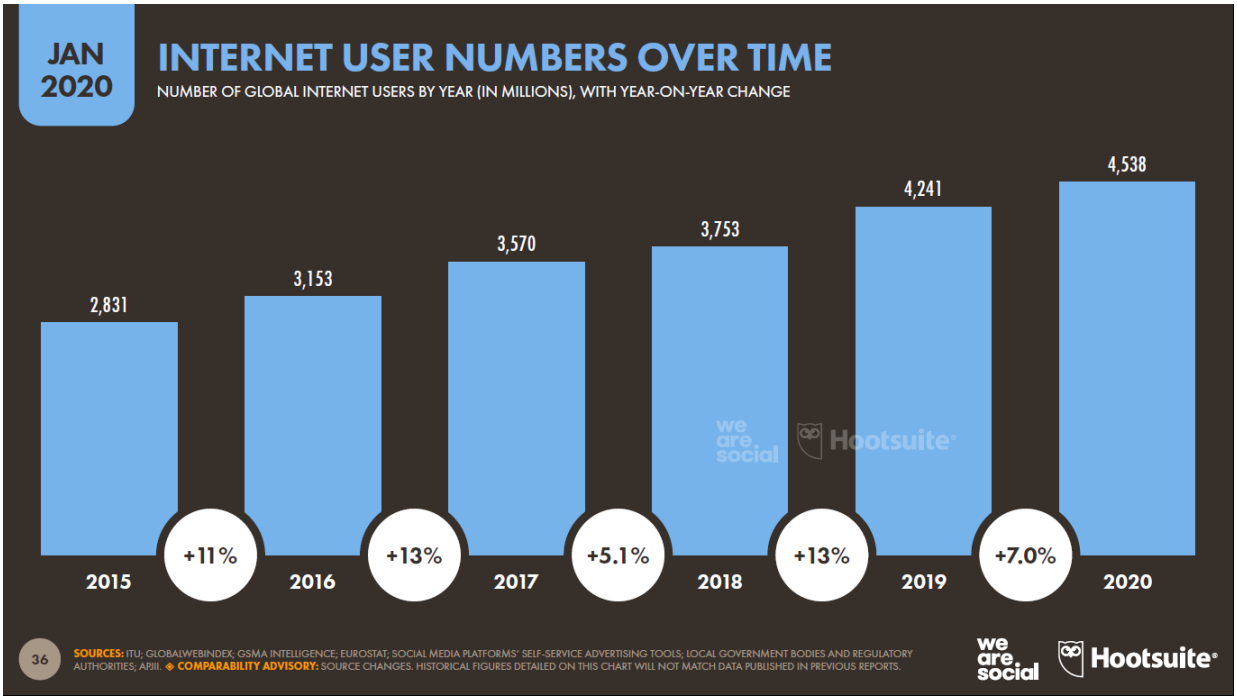


Figura 1.8: Aumento del uso de Internet a lo largo de los años. *Fuente: We are social, 2020* [36]

Por otro lado, las bases de datos suelen funcionar en conjunto con Sistemas de Administración de Bases de Datos (Database Management Systems, DBMS), softwares que permiten gestionar los datos de manera robusta y eficiente [35]. Además, los DBMS permiten tratar los datos con un alto nivel de abstracción, es decir, que permite aislar ciertos objetos de estudios y poder segmentar la información que se desea analizar [37].

Una base de datos almacena los datos según una cierta estructura. Esta estructura suele ser en base a tablas, en donde se tienen filas que corresponden a observaciones, y columnas que corresponden a variables. Se puede dar el caso en que se tiene una única tabla en donde se almacenan todos los datos, lo que sería una base de datos simple, así como también se puede dar el caso en donde se tienen varias tablas, las cuales deben estar relacionadas entre ellas por medio de índices identificadores.

1.5.3. Python

Python es un lenguaje de programación que se suele utilizar en el mundo de la ciencia de los datos. Funciona principalmente en base a la importación de librerías, es decir, se importan funcionalidades especificas según el problema que se esté resolviendo. Python es un lenguaje basado en OOP (object-oriented programming, o programación orientada a objetos en español), lo que quiere decir que cuando se importan librerías o si se crean ciertas funciones o clases, se puede acceder a estas a través de la creación de un objeto de dicha clase [40]. Por ejemplo, existe una librería que permite crear tablas (o DataFrames) llamada Pandas, por lo tanto al importar dicha librería, para poder acceder a las funcionalidades de esta se debe crear un *objeto* de dicha librería. Ya creado el *objeto*, para poder aplicar alguna de las funcionalidades que existen en la librería se aplican *métodos* sobre dicho objeto, y es de

esta manera que se trabaja en Python, creando objetos y aplicando las funciones de dichos objetos. En el código 1.1 se puede apreciar cómo se importa una librería, cómo se crea un objeto y cómo se aplica un método sobre un objeto.

Código 1.1: Ejemplo de importación de una librería, creación de un objeto y aplicación de un método

```
1 import pandas as pd
2
3 x = pd.DataFrame([0,1,2])
4
5 x.columns = ['Primera_columna']
```

Jupyter Notebook

Jupyter es un proyecto open-source (es gratis y de libre uso) creado para ayudar a los científicos de datos y de la computación a interactuar con distintos lenguajes de programación [41]. Una de las funcionalidades que ofrece el proyecto Jupyter es una interfaz de programación llamada *Jupyter Notebook*. Esta interfaz permite programar código en Python y correr una consola en la misma interfaz, pero a través de lo que se denomina "celda". Por lo tanto, al momento de programar se puede escribir el código en distintas celdas, y cada celda se puede ejecutar independiente de las demás, pero al estar todo dentro de un mismo entorno, permite utilizar el resultado de una celda en otra, facilitando así la programación y la verificación del código [42]. En la figura 1.9 se puede apreciar una leve variante del código 1.1 escrito en una celda de Jupyter Notebook, y en la otra celda simplemente se ejecuta un método (cambiar el nombre de la columna) sobre la variable creada en la celda anterior, arrojando el resultado abajo de esta.

Figura 1.9: Ejemplo de la programación en Jupyter Notebook. *Fuente: Elaboración propia*

1.5.4. Framework Django

Django es un entorno de trabajo Web (Web Framework) que permite desarrollar sitios webs. Utilizando este sistema se pueden desarrollar aplicaciones web de alta calidad, con varias funcionalidades de manera relativamente sencilla, sin tantas configuraciones ni complicaciones [43]. Django funciona con el lenguaje de programación Python, y permite regular o moderar archivos HTML para desplegar la información en un sitio web. HTML es un lenguaje de programación de sitios webs. Este es interpretado por los navegadores web como Google Chrome y Mozilla Firefox entre otros. Sus siglas significan *HyperText Markup Language* (Lenguaje de Marcas de Hipertexto), por lo tanto, es un lenguaje en el que se referencian otros elementos como enlaces hacia otros textos u otros documentos HTML [44]. Equipo Vértice, autores del libro *Diseño básico de páginas web en HTML* (2009), dicen lo siguiente: "HTML sirve para estructurar documentos (títulos, párrafos, listas, etc.), pero no describe la apariencia o el diseño de un documento sino que ofrece las herramientas necesarias para dar formato, según la capacidad del servidor en el que se almacenan las páginas web y la capacidad del navegador ..." [44].

El desarrollo web se viene desarrollando desde mediados del siglo XX, partiendo en un inicio con el desarrollo del hipertexto, luego del lenguaje HTML basado en el hipertexto que se utilizó para construir sitios web simples y ya más adelante en la historia se trabajaba con servidores web, en donde los sitios web se desarrollaban programando en HTML el contenido, y con un archivo en Python se regulaba el comportamiento de dicho HTML para poder desplegarlo en un navegador Web, es decir, se iba desarrollando un entorno de trabajo más completo y con mayor funcionalidad, pero aún de forma tosca [45]. Django facilita dicho proceso implementando un entorno de trabajo predefinido en el que con simplemente correr un servidor, y configurar la ubicación de los archivos HTML se puede programar un sitio web completo [43]. Por otro lado, una de las funcionalidades que provee Django es la configuración y manejo de bases de datos. Django tiene soporte para bases de datos como PostgreSQL, SQLite3, MySQL y Oracle [43]. Por lo tanto, Django permite generar sitios web, o aplicaciones web con relativa sencillez, además es modular y escalable lo que permite incorporar nuevas funcionalidades con el tiempo sin tener que reestructurar todo el proyecto de Django, y se puede configurar una base de datos de manera nativa para poder almacenar los datos que se generan desde el sitio web desarrollado. ¿Pero cómo se pueden recibir datos enviados desde otro sitio, o desde otra aplicación para ser trabajados en Django y almacenados en la base de datos? Por ejemplo, en el caso de un sitio web de e-commerce, ¿Cómo se configura para que el pago en línea se conecte con el sitio web, y se asocie al usuario que está comprando?, ¿Cómo se hace el envío y recepción de información?

1.5.4.1. Comunicación con bases de datos y otros sitios

Las APIs (Application Programming Interface, o Interfaz de Programación de Aplicaciones) son, como dice su nombre, una interfaz para implementar funcionalidades que los programadores (no necesariamente web) utilizan para realizar variadas tareas [46]. Por otro lado, REST (Representational State Transfer, o Transferencia de Estado Representacional) es un estilo de arquitectura para sistemas distribuidos de hipermedia. La hipermedia se refiere a la presentación simultanea de información, la cual se puede encontrar en sitios web y con la cual el usuario de dicho sitio web obtiene opciones y selecciona acciones [47]. Están los

llamados métodos, que se podrían considerar acciones a seguir con respecto a la hipermedia. Algunos de estos métodos son *GET* y *POST*, en donde el primero hace referencia a obtener información de la hipermedia mientras que el segundo hace referencia a entregar información a la hipermedia [47]. Por lo tanto, una API REST es una interfaz a través de la cual se puede enviar y recibir información (hipermedia), y es este tipo de interfaces las que facilitan el envío de información de un sitio web a otro. Entonces en el ejemplo mencionado anteriormente sobre el sitio web de e-commerce, este realiza el envío de información del usuario para efectuar el pago en línea a través de una API REST, y de igual manera recibe los datos del pago a través de una API REST.

Django no posee de manera nativa una funcionalidad del tipo API REST, por lo que no se podría realizar este tipo de configuraciones de manera sencilla para generar un "canal de comunicación" entre el sitio web que se está desarrollando y el resto de la red. Sin embargo, existe un proyecto llamado Django REST Framework que efectivamente permite incorporar una interfaz de comunicación a otros sitios web en un proyecto de Django [48]. Por lo tanto, con esta interfaz que se puede adicionar a Django, se puede construir un sitio web con varias funcionalidades, conexiones a bases de datos, y con un canal de comunicación a otros sitios webs. Es importante destacar que este "canal de comunicación" se hace posible a través de las denominada URIs (Uniform Resource Identifiers, o Identificador de Recursos Uniformes), que corresponden a una sintaxis inventada por Tim Berners-Lee (uno de los padres de la World Wide Web) para asignar a cada recurso web un identificador único [49], es decir, un valor que sirve para identificar un elemento único de la web. Por lo tanto, las API REST se logran comunicar con otro elemento de la web utilizando este identificador único.

1.5.5. Reducción de dimensionalidad: Feature Selection y Feature Engineering

Existen diversos problemas en los que se tienen datos de alta dimensionalidad, lo que quiere decir que un objeto es descrito por una gran cantidad de variables o atributos [32]. Esto implica un desafío para los encargados de analizar dichos datos, razón por la cual se ha desarrollado un proceso llamado *selección de características* (más conocido como *Feature Selection* en inglés) [50]. Dicho proceso consiste en obtener un subconjunto de variables desde un conjunto mayor, es decir, reducir la dimensionalidad de la base de datos. Hay estudios que indican que una correcta realización en el proceso de Feature Selection puede mejorar el desempeño de los modelos entrenados con el nuevo subconjunto de variables [51].

Debido a que en el presente trabajo se trabajó con datos etiquetados (se sabe el valor que se desea predecir), se acota el marco conceptual a algoritmos de tipo supervisado (posteriormente se explicará sobre el significado de un algoritmo supervisado), además este tipo de selección basado en algoritmos supervisados suele estar orientado a problemas de clasificación, como es el caso del presente trabajo [50].

Debido a que este tipo de técnicas se centran en detectar las variables más relevantes, es decir, que puedan caracterizar de mejor manera la clase a la cual se quiere clasificar, pero sin caer en la redundancia (no tener variables que tienen un impacto similar sobre la clase), es que se busca medidas de similitud entre variables. De acuerdo con Cai et al. (2018), el mejor criterio de evaluación para problemas de clasificación sería el Error Bayesiano, sin embargo

este puede ser complejo de calcular, por lo que también proponen una serie de otras medidas de similitud o de relación como las siguientes:

- Coeficiente de correlación entre las variables
- Coeficiente de correlación de Pearson
- Información mutua
- Incertidumbre simétrica
- Distancia de la información
- Distancia euclidiana

Por otro lado, es importante definir la diferencia entre un algoritmo basado en filtros, contenido o mixto. En primer lugar, cuando se hace una selección de variables en base a filtros, lo que se obtiene es un subconjunto de variables del conjunto original, en donde estas variables para cumplir con los requisitos de ser importantes deben cumplir ciertos criterios, como por ejemplo tener una alta correlación con la clase a la cual se desea clasificar, y una baja correlación entre las demás variables en el subconjunto final.

Los algoritmos de tipo contenido (wrapper en inglés), no sólo realizan el proceso de seleccionar las variables más importantes y entregar el subconjunto final, sino que dentro del mismo algoritmo se entrena al modelo final de clasificación que se desea. La principal ventaja de dicho enfoque es una alta precisión en el modelo final, debido a que se selecciona el mejor conjunto de variables posibles, sin embargo, las principales desventajas son el costo en tiempo y en recursos que conlleva dicho tipo de algoritmos y la poca capacidad de generalización fuera de la muestra de entrenamiento [50]. Los algoritmos de tipo wrapper se pueden separar en 2 tipos principalmente, aquellos en los que se parte con todas las variables originales y a medida que avanza el algoritmo se van eliminado variables para reducir la dimensionalidad (más conocido por su nombre en inglés *backward elimination*), y aquellos en los que se parte con un subconjunto pequeño de variables y a medida que avanza el algoritmo se van agregando nuevas variables al set (más conocido como *forward selection*) [52].

Luego, se tienen modelos mixtos, en donde se combinan los algoritmos de filtro y los algoritmos de tipo contenido. De esta manera, primero se realiza un filtro, obteniendo un subconjunto de variables que remueve las variables irrelevantes de la base, y luego sobre el subconjunto obtenido se aplican los algoritmos de tipo contenido, obteniendo así un subconjunto final de variables que maximizan la precisión del modelo, en un tiempo no tan elevado de procesamiento, ya que no se trabaja con una alta dimensionalidad [50].

Finalmente, se tienen los algoritmos de tipo embedded (incrustado en español, sin embargo no se utiliza mucho ese término). Este tipo de algoritmos es similar al tipo wrapper en el sentido de que en ambos está relacionado un modelo predictivo. Sin embargo, a diferencia del método wrapper en donde progresivamente se van eliminando/agregando variables de acuerdo con el poder predictivo del modelo en cuestión, en los algoritmos de tipo embedded es el mismo modelo el que realiza la selección de variables, en el proceso de entrenamiento (más adelante se explicará en mayor detalle en que consiste el entrenamiento de un modelo)

[52]. Existen modelos específicos que por su naturaleza de entrenamiento califican como modelos de tipo embedded, como por ejemplo los árboles de decisión, en donde el proceso de entrenamiento consiste en seleccionar variables de acuerdo con un cierto criterio [52].

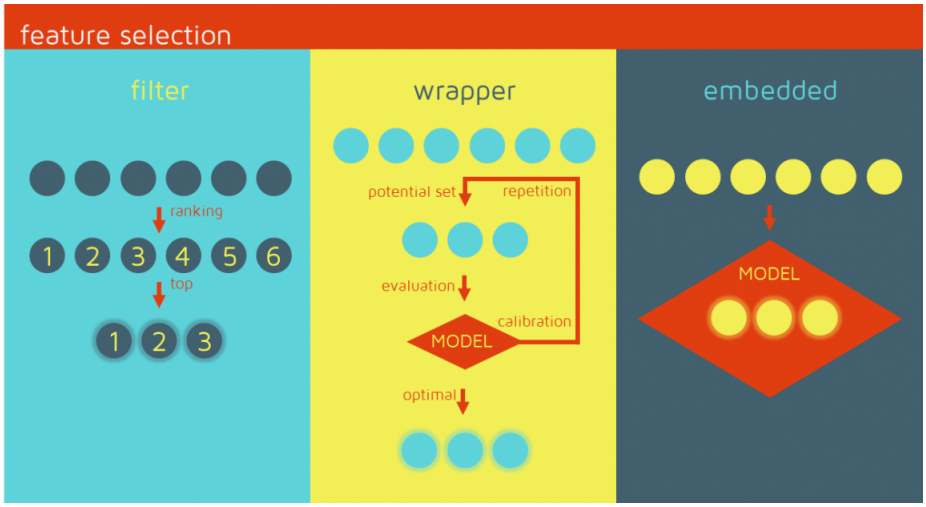


Figura 1.10: Figura resumen de los métodos de Feature Selection. *Fuente: QuantDare, 2019* [54]

El proceso de Feature Selection es parte del proceso llamado Feature Engineering. De acuerdo con Garla & Brandt (2012), el proceso de Feature Engineering involucra la selección de un subconjunto de variables importantes (Feature Selection), y/o la combinación de distintas variables para generar una nueva variable, es decir, la transformación de variables [55]. Un ejemplo de Feature Engineering se puede encontrar en el paper *Feature Engineering and Classifier Ensemble for KDD Cup 2010* de Yu et. al. (2010) [56]. En dicho paper se expone sobre la competencia KDD Cup, en donde distintos participantes deben realizar una tarea en específico relacionada al área de Data Mining, y el participante que entregue los mejores resultados es el ganador. En particular, en la competencia del año 2010 el problema era predecir el desempeño de estudiantes en base a la información pasada de estos en una plataforma de educación. Por lo tanto, inicialmente se tiene una base de datos con los registros de los estudiantes en esta plataforma, y los autores del paper deben realizar el proceso de Feature Engineering en donde transforman los datos originales (comportamiento de estudiantes en una plataforma de educación en donde cada registro es la solución o un paso de una solución a un problema específico) en nuevas variables como el nombre del estudiante, el nombre de unidad, nombre de sección, nombre del problema, nombre del paso, entre otros [56].

El proceso de Feature Engineering se puede realizar con datos de una única tabla, en donde se ejecutan ciertas funciones sobre variables existentes de la tabla para generar nuevas variables que aporten más información como en el caso de Yu et. al. (2010) [56], así como también se puede realizar en el contexto en donde se tienen múltiples tablas como una base de datos relacional, y se deba llevar la información de interés a una única tabla [57]. De acuerdo con sitios como Kaggle, en donde se realizan distintas competencias sobre el análisis de datos y proyectos de Data Science, el proceso de Feature Engineering en donde se determinan que variables son importantes para el problema que se quiere resolver, es el más costoso en tiempo y dinero, y es el proceso en el que se necesita más conocimiento sobre los datos, sobre

el contexto del problema y donde se necesita mayor experiencia e intuición [57].

Hasta el año 2015, los expertos en análisis de datos, mejor conocidos como científicos de datos, tenían que hacer el proceso de Feature Engineering por su cuenta, es decir, entender la relación entre las distintas tablas o variables de una base de datos, calcular distintas métricas, aplicar funciones y transformaciones a las variables, y finalmente seleccionar las variables creadas en base a los datos de las distintas tablas que mejor desempeño le den al modelo predictivo en cuestión que se haya estado desarrollando [58]. Además, los datos crudos de las bases de datos pueden ser estructurados (un dato numérico con algún significado específico y con rangos definidos) o no estructurados (series de tiempo, imágenes, texto, etc), siendo el proceso de Feature Engineering más complicado y largo en bases no estructuradas [58]. Sin embargo, el año 2015 salió una publicación realizada por científicos del MIT, en donde se desarrolla el llamado *Deep Feature Synthesis (DFS)*, en el cual se automatiza el proceso de Feature Engineering [59]. Posterior a dicha publicación, en IBM se desarrolló otro algoritmo capaz de realizar el proceso de Feature Engineering, llamado *One Button Machine (OneBM)*, y en la actualidad ya existen varios algoritmos y librerías que permiten realizar dicho proceso de manera automática, definiendo ciertos parámetros, incluso teniendo varias tablas como sucede en una base de datos relacional.

1.5.6. Machine Learning y modelos de clasificación

Hoy en día, debido a la gran cantidad de datos que se generan en los distintos rubros, tanto económicos, académicos, de salud, entre otros, en conjunto con el aumento en la capacidad de procesamiento de los computadores, es que se abre la posibilidad de crear modelos computacionales que aprendan de los datos existentes, y logren predecir datos futuros o clasificar los datos en alguna categoría establecida [63]. A esto se le conoce como Machine Learning, ya que se le enseña a una máquina a predecir o clasificar, en base a un entrenamiento.

Existen distintos tipos de modelos o algoritmos de Machine Learning, según el tipo de entrenamiento que se lleve a cabo. A continuación se muestran distintos algoritmos que muestra Ayodele (2010) en su libro *Types of Machine Learning Algorithms* [64]:

- **Aprendizaje supervisado:** Este tipo de algoritmos debe ser entrenado para lograr estimar el resultado deseado, en base a los datos suministrados. El entrenamiento se hace utilizando una etiqueta que indica cual es el resultado deseado. Se suele utilizar para modelos de clasificación.

- **Aprendizaje no supervisado:** A diferencia del aprendizaje supervisado, este tipo de algoritmos no cuentan con una etiqueta, sino que analiza patrones dentro de los datos suministrados.

- **Aprendizaje semi supervisado:** Combina ambos tipos de algoritmos, con etiquetas y sin etiquetas, para así generar un modelo.

- **Aprendizaje por refuerzo:** En este tipo de Machine Learning, el algoritmo aprende a cómo actuar según una cierta observación de los datos. Cada modificación que realice el modelo genera un impacto positivo o negativo, que le indica al algoritmo que reforzar y que no.

Principalmente se utilizan los 2 primeros tipos, aprendizaje supervisado y no supervisado.

El proceso de forma simplificada que se sigue al crear y entrenar un modelo es primero dividir la base de datos en (por lo menos) 2 subconjuntos, uno de entrenamiento y uno de prueba. El primero, como dice su nombre se utiliza para "enseñarle" a la máquina las características de la base, y así esta pueda detectar patrones de dicha base. Esto con el fin de posteriormente poder aplicar lo que "aprendió" el modelo para: clasificar otros datos (subconjunto de prueba) en alguna categoría en específico determinada por el problema con el que se está trabajando; predecir algún valor sobre otros datos (subconjunto de prueba) como por ejemplo, una decisión que tomará un cliente en base a ciertas características de este. En otras palabras, con la muestra de entrenamiento el modelo aprende que patrones existen o que combinación de variables permite clasificar o predecir un cierto valor (etiqueta), y luego ese modelo se utiliza para poder clasificar o predecir (según los valores que hayan tenido las etiquetas en la base de entrenamiento) nuevos datos de acuerdo con las variables de dichos datos. Al hacer ese proceso en la muestra de prueba se puede comparar lo que predice el modelo según lo que aprendió con el valor real que debería predecir (etiqueta), y de esta manera detectar un error de predicción, y claro está que a menor error se tiene un mejor desempeño, ya que logra predecir correctamente los valores reales. Por lo tanto, el ideal de un modelo de Machine Learning es poder clasificar o predecir fuera de la muestra de entrenamiento, es decir, que se puedan generalizar los resultados.

Sin embargo, existe un problema bastante común conocido como sobreajuste (overfitting en inglés), el cual consiste en que el modelo aprende de manera sesgada, es decir, que aprende demasiado especifico a la muestra de entrenamiento, perdiendo así la capacidad de generalizar. Un ejemplo de esto podría ser la clasificación de imágenes, en donde el modelo en cuestión debe predecir si una imagen es un auto o no. Entonces, si la base de entrenamiento consiste únicamente de autos rojos, al momento de pasarle al modelo un auto azul, aunque este tenga ruedas, puertas, ventanas, y todas las características que tiene un auto, el modelo dirá que no es un auto porque no es rojo, es decir, aprendió demasiado ajustado a los valores que se tienen en la base de entrenamiento y no puede generalizar.

Una manera que existe para hacerse cargo de dicho problema es la implementación de *K-fold Cross Validation* o validación cruzada. Esta técnica consiste en segmentar la base en K subconjuntos, y luego en la primera iteración se entrena el modelo con K-1 subconjuntos y se utiliza el restante para probar el desempeño del modelo entrenado; luego en la segunda iteración nuevamente se entrena al modelo con otros K-1 subconjuntos, y se prueba con el otro restante, y así sucesivamente por K iteraciones. De esta manera el modelo no aprende según la primera asignación de la muestra de entrenamiento, sino que constantemente va modificando la muestra de entrenamiento, evitando así el sesgo y el sobre ajuste. Finalmente, el error asociado a K-fold Cross Validation se calcula promediando los errores obtenidos para cada iteración, por lo que si ese error es bajo, efectivamente se tiene un modelo no sobre ajustado [65].

Algoritmos de clasificación

Existen múltiples algoritmos de clasificación, en donde cada uno tiene sus respectivas ventajas y desventajas. A continuación se presenta una lista de los algoritmos más utilizados

en el área de Machine Learning:

Naive Bayes

Este algoritmo asigna un objeto a una clase según el valor que tienen las variables que caracterizan a dicho objeto. Estas variables se conocen como *features*. En Naive Bayes se asume que las variables son independientes unas de otras, lo que es poco realista en problemas del mundo real. Sin embargo, a pesar de que teóricamente puede que no se cumpla dicho supuesto, este algoritmo ha demostrado un gran desempeño en distintos problemas de distintas industrias como la clasificación de texto, diagnóstico médico y gestión de sistemas [66]. El clasificador Naive Bayes tiene 3 principales ventajas por sobre otros clasificadores: la estructura esta predefinida, y no se necesita realizar un aprendizaje o modificación de la estructura según el problema específico; es muy eficiente para trabajar con grandes bases de datos sobre todo cuando las variables no están fuertemente correlacionadas (se hace más verídico el supuesto de independencia entre las variables); se requiere de una muestra pequeña de entrenamiento para estimar los parámetros necesarios [67].

Al momento de utilizar Naive Bayes es necesario entender cierta notación que hace referencia a las variables, a las clases y las probabilidades de que dado un conjunto de variables, se tenga una clase en específico. En primer lugar se tiene $\mathbf{X}$, que corresponde a un vector cuyos elementos son las variables del problema. A este vector se le conoce como vector de entrada, ya que es lo que se le entrega al modelo tanto para entrenar como para predecir las clases. También, se definen las clases como C_k , en donde k es igual a la cantidad de clases.

$$\mathbf{X} = (x_1, x_2, ..., x_n) \tag{1.2}$$

Por lo tanto, lo que se debe calcular en primera instancia es la probabilidad condicional de obtener una clase en particular, dado el vector de entrada que se tenga. El teorema de Bayes indica lo siguiente:

$$P(C_k|\mathbf{X}) = \frac{P(\mathbf{X}|C_k)P(C_k)}{P(\mathbf{X})} \tag{1.3}$$

Para todas las clases se tiene el mismo denominador P($\mathbf{X}$), el cuál toma el valor:

$$P(\mathbf{X}) = P(C_1)\prod_{i-1}^{n} P(x_i|C_1) + P(C_2)\prod_{i-1}^{n} P(x_i|C_2) + ... + P(C_k)\prod_{i-1}^{n} P(x_i|C_k) \tag{1.4}$$

Luego, asumiendo la independencia de las variables, se tiene:

$$P(\mathbf{X}|C_k) = \prod_{i-1}^{n} P(x_i|C_k) \tag{1.5}$$

Finalmente, reemplazando todo lo anterior en una misma ecuación se obtiene la ecuación

1.6, que indica la probabilidad de obtener una clase según las variables de entrada.

$$P(C_k|\mathbf{X}) = \frac{P(C_k)\prod_{i-1}^{n} P(x_i|C_k)}{P(C_1)\prod_{i-1}^{n} P(x_i|C_1) + P(C_2)\prod_{i-1}^{n} P(x_i|C_2) + ... + P(C_k)\prod_{i-1}^{n} P(x_i|C_k)} \quad (1.6)$$

Ya con la probabilidad modelada se debe definir el clasificador $\hat{y}$, que tomando todas las probabilidades de obtener una clase de acuerdo con un set de variables, selecciona la clase que tenga una mayor probabilidad.

$$\hat{y} = argmax_{k\in 1,...,K} \frac{P(C_k)\prod_{i-1}^{n} P(x_i|C_k)}{P(C_1)\prod_{i-1}^{n} P(x_i|C_1) + ... + P(C_k)\prod_{i-1}^{n} P(x_i|C_k)} \quad (1.7)$$

Support Vector Machine (SVM)

SVM es un algoritmo que en base a varios datos de prueba, puede aprender a clasificar. SVM se puede explicar a través de funciones matemáticas y la optimización de estas, sin embargo, el concepto general de SVM se puede entender explicando 4 conceptos: un hiperplano separador, el margen máximo del hiperplano, los márgenes suaves y la función Kernel (o núcleo) [68].

El algoritmo de SVM, para determinar la diferencia entre clases utiliza lo que se conoce como el hiperplano separador [68]. La manera más simple de entender que es un hiperplano separador sería la de simplificar el problema a que únicamente se tienen 2 variables en el vector de entrada $\mathbf{X}$, es decir, $\mathbf{X} = (\ x_1\ ,\ x_2\ )$. Al tener únicamente 2 variables, cada observación se puede representar en un gráfico de 2 dimensiones como se puede apreciar en la figura 1.11-a, en donde se tiene una variable llamada ZYX y otra llamada MARCKSL1. En dicho gráfico se representan todas las observaciones que se tienen, según la clase a la cuál pertenecen. En el ejemplo de la figura 1.11 hay elementos de la clase verde y hay elementos de la clase roja. Un hiperplano separador corresponde al plano (o recta en el caso de 2 dimensiones) que permite separar ambas clases, quedando una clase a un lado del hiperplano, y la otra clase al otro lado del hiperplano, como aparece en la figura 1.11-b. Dicho ejemplo considera solamente 2 variables, por lo que sería un problema en 2 dimensiones, sin embargo, si aumentamos a 3 variables, se tendría un gráfico de 3 dimensiones, de manera tal que el hiperplano en ese caso si sería un plano como tal, y así sucesivamente se pueden ir agregando variables y dimensiones, y el hiperplano "cortará" el espacio dejando las clases bien definidas a cada lado del hiperplano [68].

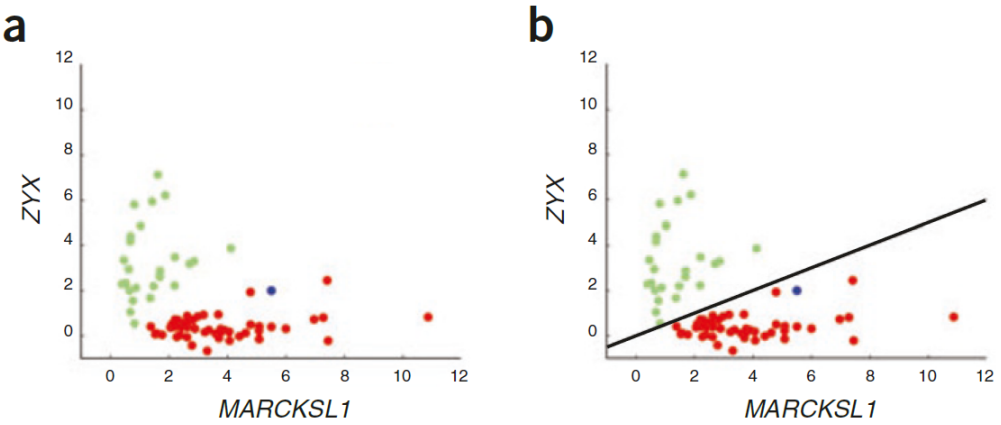


Figura 1.11: (**a**) Gráfico en 2 dimensiones en donde se plasman las observaciones con un color para cada clase (2 clases); (**b**) Línea recta que representa al hiperplano separador en 2 dimensiones. Dicho hiperplano separa ambas clases (verde y rojo) *Fuente: Noble, 2006* [68]

Sin embargo, no existe una única recta posible o un único hiperplano posible entre 2 clases distintas, y es por esto que parte del algoritmo de SVM consiste en optimizar la posición que tendrá el hiperplano. En la figura 1.12 se puede apreciar que hay 2 clases, círculos y cruces. En dicha figura claramente se puede trazar un hiperplano (recta) que divida el espacio en 2, dejando a una clase a un lado del hiperplano y a la otra clase al otro lado del hiperplano. Además, se puede notar que existen 3 observaciones que están encerradas dentro de un cuadrado, las cuales son los puntos extremos de dicha clase, es decir, los que están en el límite del espacio que define la clase. Dichas observaciones son las conocidas como *vectores de soporte* (de ahí viene el nombre del algoritmo), y es con estos vectores de soporte que se construyen los márgenes o la "frontera" de las clases [69]. El objetivo del algoritmo de SVM es maximizar la distancia entre márgenes, es decir, separar lo más posible la distancia entre las distintas clases, y según esta distancia entre márgenes es que se determina la posición del hiperplano separador. Al seleccionar este hiperplano en particular, en donde se maximizan las distancias entre los márgenes, se maximiza también la habilidad del modelo para poder predecir la correcta clasificación [68].

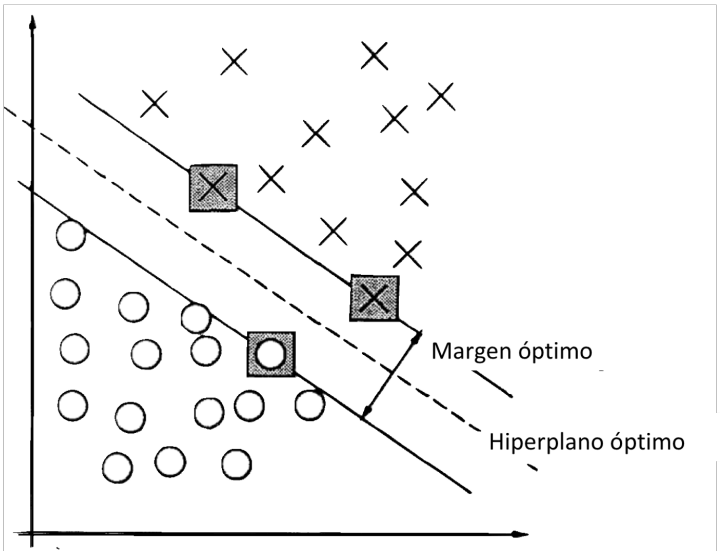


Figura 1.12: (**a**) Gráfico en 2 dimensiones en donde se puede apreciar cuales son los vectores de soporte para cada clase, los márgenes asociados a estos vectores y el hiperplano generado *Fuente: Cortes & Vapnik, 1995* [69]

Los ejemplos mencionados anteriormente se cumplen cuando los datos están perfectamente separados y efectivamente se puede trazar un hiperplano separador perfecto, en donde cada clase quede a un lado del hiperplano. Sin embargo, esto no siempre se da, y puede haber casos en donde no existe un hiperplano que pueda dividir perfectamente el espacio dejando una clase a cada lado del hiperplano. Es por esto que existen los márgenes suaves. Este concepto quiere decir que el margen funciona como una membrana semipermeable, dejando así pasar algunas observaciones de la otra clase. De esta manera, el algoritmo puede simplemente ignorar algunos (pocos) datos y no los considera a la hora de definir los márgenes, y así si se puede generar un hiperplano separador, con un leve error de clasificación para algunas observaciones [68].

Finalmente se tiene el caso en que dada la distribución de los datos, no es posible separarlos con un hiperplano ni siquiera suavizando los márgenes, como se puede ver en la figura 1.13-a, en donde los datos están en una línea (dimensión) y se tiene datos mezclados de las clases. Para solucionar este problema, se aplica una función Kernel, que lo que hace es agregar una dimensión extra, como se puede ver en la figura 1.13-b. Al hacer esto, los datos se redistribuyen por la nueva dimensión y de esta manera si se puede trazar el hiperplano separador [68]. Lo anterior puede aplicarse a cualquier número de dimensiones, y la función Kernel siempre aumentará dicha cantidad de dimensiones para "ver los datos desde otro punto de vista" para poder así asignar el hiperplano separador.

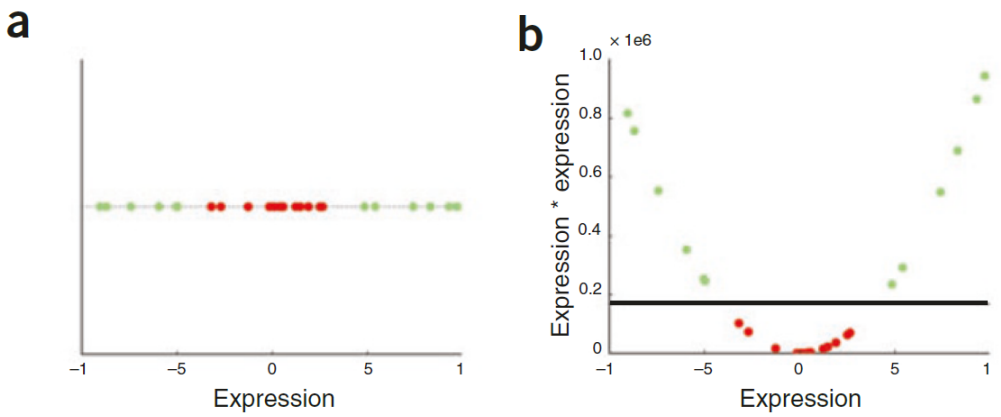


Figura 1.13: (**a**) Visualización de las observaciones en una dimensión, diferenciando por clase (verde y rojo); (**b**) Visualización de los mismos datos que (**a**), con la dimensión extra generada por la aplicación de un Kernel. *Fuente: Noble, 2006* [68]

El algoritmo SVM entonces, busca trazar el hiperplano óptimo, que mejor separe las distintas clases (no necesariamente binaria como se muestra en los ejemplos), y en base a los hiperplanos trazados, posteriormente puede predecir la clase C_k a la que pertenece un vector de entrada $\mathbf{X}$ fuera de la muestra de entrenamiento.

Árboles de decisión

Los árboles de decisión también son algoritmos de clasificación. Estos funcionan determinando una serie de reglas para las variables del vector $\mathbf{X}$, y según se vayan cumpliendo dichas reglas se determina a que clase pertenece una observación. Por lo tanto, según los valores que tomen ciertas variables, se va recorriendo el árbol hasta llegar al último nodo, que corresponde a una de las posibles clases [70].

Existen muchas maneras distintas de diseñar y construir un árbol de decisión como *Bottom-up* (desde abajo hacia arriba) y *Top-Down* (desde arriba del árbol hacia abajo) , además de tener distintos criterios de optimalidad [70]. En este apartado no se detallarán las distintas maneras de construir un árbol de decisión, más bien se explica que es un árbol de decisión, sus componentes y cómo el clasificador "recorre" el árbol para poder asignar una clase en específico a una observación.

Un árbol de decisión lleva el nombre de árbol debido a la estructura de nodos y vértices que tiene, que gráficamente se pueden interpretar como las ramas del árbol. Sin embargo, generalmente estos "árboles" están invertidos, quedando la raíz del árbol en la parte superior como se puede apreciar en la figura 1.14. Sólo hay un nodo raíz, que es aquel nodo que no tiene vértices de entrada, sino que sólo salidas, como el nodo que está en la parte superior del árbol en la figura 1.14. Dicho nodo contiene todas las clases, por lo que es ahí desde donde se empieza a segmentar y recorrer el árbol para llegar a una clasificación final. Se entiende como recorrer el árbol el ir verificando las condiciones que hay en cada nodo y según se cumpla o no, se va por un vértice (rama) o el otro, hasta llegar a los últimos nodos de la parte inferior, conocidos como *hojas* [70]. Un punto importante que se debe cumplir en los

árboles de decisión es que debe existir un único camino entre la raíz y cualquier otro nodo del árbol.

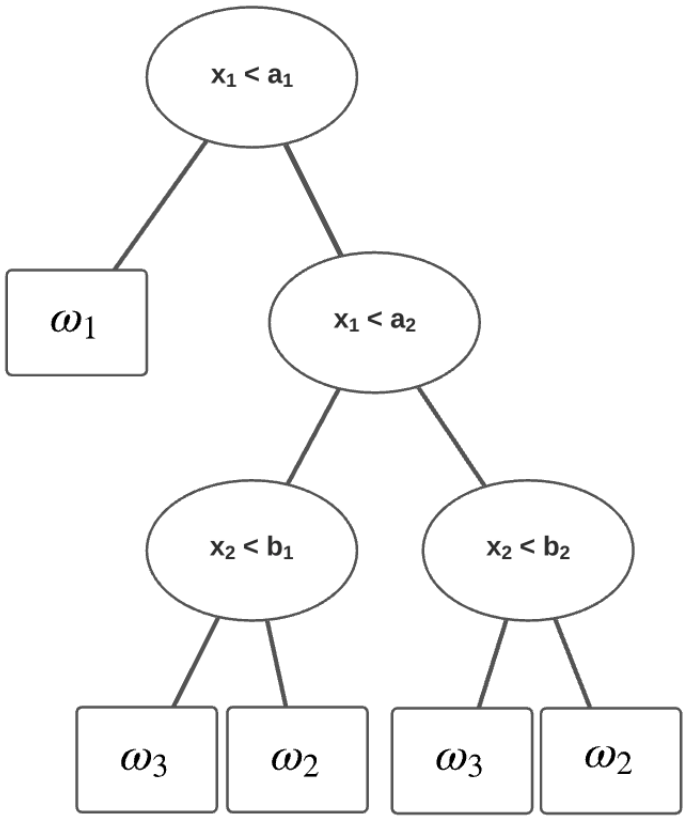


Figura 1.14: Representación gráfica de un árbol de decisión *Fuente: Safavian & Landgrebe, 1991* [70]

Como se mencionó anteriormente, existen varias maneras de construir un árbol. Generalmente se suelen construir de manera dinámica, es decir, paso a paso, calculando con cada nueva iteración cual es el próximo paso. Es necesario definir ciertos criterios de corte, para que cada vez que se llegue a un nuevo nodo, se pueda seguir dividiendo hacia abajo el árbol. Cómo se puede apreciar en la figura 1.14 en cada nodo existe algún criterio, y que según se cumpla ese criterio o no, se determina a que rama del árbol se sigue.

Random Forest

El algoritmo Random Forest (bosque aleatorio) consiste en la combinación de varios árboles de decisión (por eso el nombre de bosque, varios árboles), y que en base a la clasificación individual que hace cada árbol del bosque, se elige la clasificación más popular y ese será el resultado del clasificador Random Forest [71]. Debido al tipo de modelo que es un Random Forest, se tiene que a mayor cantidad de árboles que se tengan, mejora la convergencia del algoritmo, es decir, se mejora la clasificación, lo que implica que a mayor cantidad de árboles no se sufre de sobreajuste.

La implementación de este algoritmo consiste en primer lugar, generar varias muestras "bootstrap" (no existe una traducción directa de este término), lo que significa que del set de

datos original se extraen aleatoriamente x observaciones, luego se reemplazan en el conjunto de datos, y posterior a eso se repite el procedimiento K veces, quedando así una base final distinta a la original. Debido a que aleatoriamente se extraen valores y luego aleatoriamente se reemplazan dichos valores en la base, pueden suceder 2 cosas: hay valores de la base original que no están en la base final debido a que no fueron seleccionados aleatoriamente, pero si fueron reemplazados; en la base final hay valores repetidos, ya que pueden haber sido seleccionados aleatoriamente en más de una iteración. Los registros que no llegan a la base final se les conoce como observaciones fuera de la bolsa (*Out-of-bag observations*) y posteriormente se utilizan para medir el rendimiento del modelo [72]. Aproximadamente el 63 % de los datos originales quedan en la muestra final de bootstrap, por lo menos con una ocurrencia (los valores se pueden repetir) [72].

Como se mencionó anteriormente, Random Forest consiste en un conjunto de árboles de decisión, por lo que una vez construida la muestra bootstrap se procede a construir todos los árboles del bosque. Para construir cada árbol de decisión se utiliza la muestra bootstrap. El proceso para construir dicho árbol consiste en elegir aleatoriamente una cantidad de variables de la muestra bootstrap para la bifurcación de cada nodo. Por lo tanto, en el primer nodo en vez de considerar todas las variables existentes en la muestra, aleatoriamente se elige una cantidad de variables a considerar para hacer la división del nodo. Una vez elegido el criterio de separación, se continúa construyendo el árbol eligiendo para cada nuevo nodo un subconjunto aleatorio de las variables restantes para elegir el nuevo criterio de separación. Se procede de esta manera hasta construir el árbol completo. Una vez terminada la construcción del primer árbol se procede a construir más árboles aplicando el mismo proceso de selección aleatoria de variables para cada nodo, asegurando así que haya árboles distintos. Ya construidos todos los árboles del bosque, se procede a pasar datos reales al Random Forest, y la manera en la que este modelo clasifica es que pasa la observación a clasificar por todos los árboles construidos con la muestra bootstrap, de manera tal que cada árbol entrega una clasificación para dicha observación. Finalmente, el modelo asigna la clase que obtuvo mayor frecuencia entre todas las asignaciones de los árboles del bosque.

Finalmente para medir el desempeño del modelo se hace pasar por el Random Forest las out-of-bag observations, y se puede comparar la clase real con la clase que predice el modelo.

Redes neuronales artificiales (ANN)

Las redes neuronales artificiales (*Artificial Neural Networks*) surgen de la inspiración de las redes neuronales biológicas que presentan los seres vivos. Estas otorgan la capacidad de resolver problemas perceptuales complejos, gracias a la capacidad de paralelismo, aprendizaje, generalización, adaptabilidad, procesamiento de información contextual, entre otras capacidades.

Una neurona está compuesta de 3 elementos principalmente: Soma, axón y dendritas, y el conjunto de los millones de neuronas que tienen los seres humanos (y otras especies) y sus interconexiones son las que permiten realizar habilidades complejas como reconocer la cara de una persona por ejemplo. En la figura 1.15 se puede apreciar el bosquejo de una neurona y sus partes. Las neuronas funcionan en conjunto con otras neuronas, y estas se comunican a través de las llamadas sinapsis. La sinapsis ocurre con la interacción entre la dendrita de una neurona y la hebra del axón de otra neurona, y al hacer la sinapsis se transmite un pulso

eléctrico desde la hebra del axón hacía la dendrita de la otra neurona. El cerebro humano contiene alrededor de 10^{11} neuronas y entre 10^{14} y 10^{15} conexiones entre neuronas, en donde cada neurona se conecta con 1.000 a 10.000 otras neuronas.

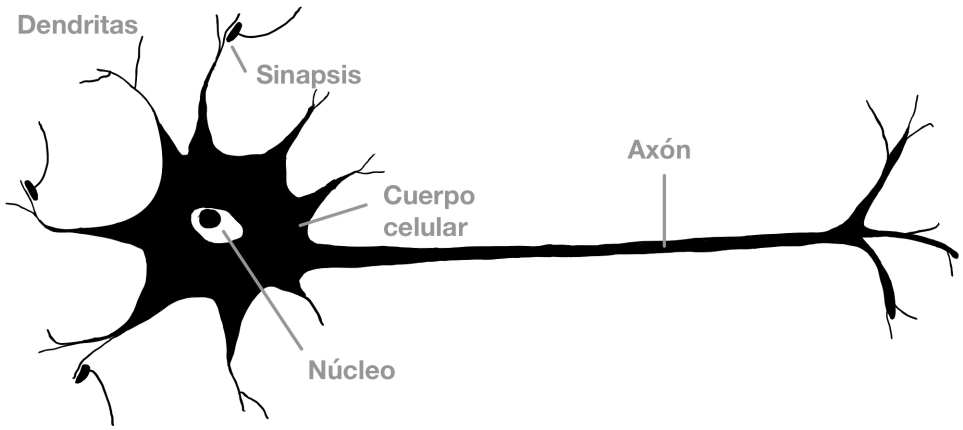


Figura 1.15: Bosquejo de una neurona biológica con sus componentes
Fuente: Jain, Mao & Mohiuddin, 1996 [73]

Las neuronas artificiales gráficamente tienen un aspecto similar a las biológicas, cómo se puede apreciar en la figura 1.16. Las neuronas artificiales toman el nombre de *perceptrón*, y al igual que una neurona tiene distintos componentes. En primer lugar, se tiene el vector de entrada $\mathbf{X}$, y cada variable x_i se conecta al "soma" del perceptrón a través de conexiones, por lo que todas estas conexiones se pueden considerar las dendritas del perceptrón. En un perceptrón, la sinapsis se modela con los pesos o *weights* w_i , los cuales permiten darle más o menos importancia al valor de una variable x_i . La recepción de toda la información de las "dendritas" se modela como la combinación lineal de variables x_i con los pesos w_i , y es lo que se aprecia como $\sum$ en la figura 1.16. La actividad que ocurre en el soma de una neurona puede ser representada por la función de activación, la cual toma de entrada el resultado de la combinación lineal entre pesos y variables, y de salida devuelve un único valor que depende exclusivamente de qué tipo de función sea. Existen varias funciones de activación como la sigmoídea, gaussiana, o de umbral por mencionar algunas. Por lo tanto, lo que entra en un perceptrón es el vector de entrada $\mathbf{X}$ y lo que sale es un valor $\mathbf{y}$.

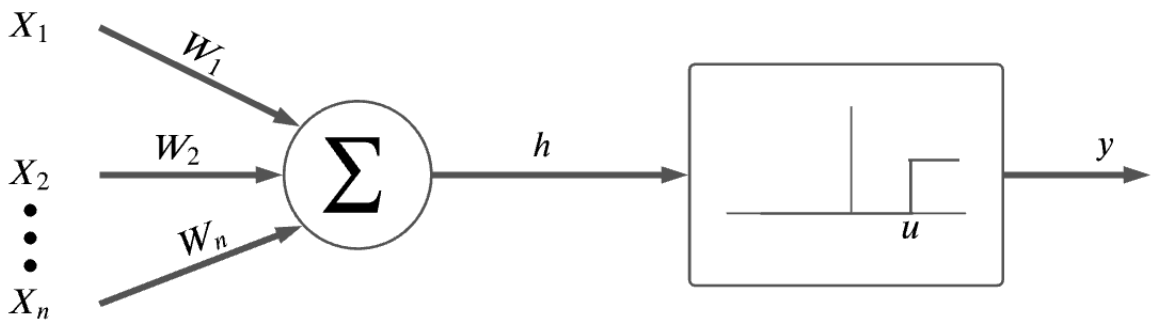


Figura 1.16: Bosquejo de una neurona artificial o perceptrón *Fuente: Jain, Mao & Mohiuddin, 1996* [73]

Uno de los métodos más utilizados en redes neuronales es el de una red neuronal de múltiples capas, en donde se tienen varios perceptrones conectados entre ellos por capas. En la figura 1.17 se puede apreciar una visualización de una red multicapa, en donde cada nodo de esa red corresponde a un perceptrón, y una capa se entiende como una columna de perceptrones. Es importante distinguir la primera capa con la última, las cuales toman los nombres de capa de entrada y capa de salida respectivamente. La capa de entrada corresponde a las variables del vector de entrada $\mathbf{X}$, mientras que la capa de salida suele ser el resultado del modelo, es decir, que si el modelo busca predecir un número, la capa de salida será un perceptrón cuyo valor $\mathbf{y}$ (1.16) corresponderá al número predicho, mientras que si el modelo busca predecir una clase, en la capa de salida se tendría un perceptrón por cada clase del problema, por lo menos en una versión clásica de redes neuronales [73].

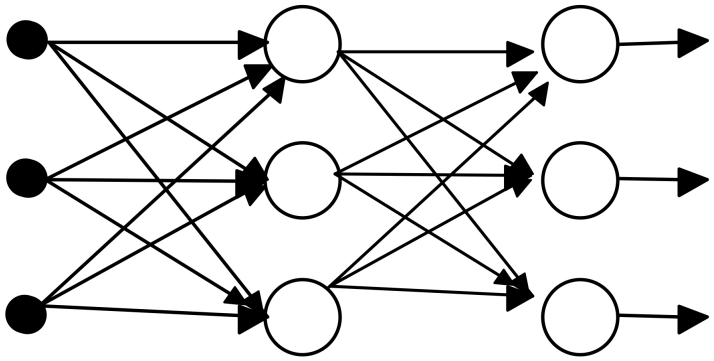


Figura 1.17: Bosquejo de una red de múltiples capas *Fuente: Jain, Mao & Mohiuddin, 1996* [73]

La manera general en la que este tipo de algoritmos "aprende" y detecta patrones es cambiando los pesos w_i asignados a cada conexión entre perceptrones. Antes de actualizar los pesos del modelo es importante tener una medida de error del modelo, que tan bien o mal predice. Para calcular este error se tiene que comparar el valor real que se utiliza como etiqueta, con el valor que se predice. Si la diferencia es muy alta quiere decir que el modelo

predice muy mal, y para reajustarlo y hacer que prediga mejor, se tiene que actualizar los pesos de cada conexión. Hay más de una manera de actualizar los pesos de la red, pero por lo general se modifican según la magnitud del error que se tiene [73]. Por lo tanto, la manera en que se entrena el modelo es la siguiente:

- Se pasa una observación con todas sus variables al modelo a través de la capa de entrada.
- Una vez que dicha observación pasa por toda la red llega a la capa de salida, en la que se predice el resultado del modelo.
- Se compara el valor que predice el modelo con el valor real de dicha observación, y si la diferencia es considerable se deben actualizar los pesos del modelo (se reajusta el modelo).
- Con el valor de la diferencia entre la predicción y el valor real se recorre la red hacía atrás actualizando los pesos según algún criterio específico que considera la diferencia de la predicción con lo real.
- Ya actualizado los pesos se introduce una nueva observación en la red y se sigue iterando hasta que la diferencia entre lo predicho y lo clasificado sea mínima o hasta que se cumpla alguna otra condición en caso de que el modelo nunca converja por la complejidad de los datos.

Dicho algoritmo de actualización de los pesos, en donde se recorre la red hacia "adelante" y luego hacia "atrás", se conoce como *Back-propagation* (propagación hacia atrás) [73].

Un aspecto importante a considerar de las redes neuronales, es que este tipo de algoritmos tienen un funcionamiento de "caja negra", es decir, que todo el procesamiento que realiza el modelo, las transformaciones matemáticas, la aplicación de funciones, la combinación de los pesos, y varios otros parámetros que tiene una red neuronal, no es accesible por lo que esa información se pierde, quedando así solamente el resultado del modelo que es la predicción del valor numérico o la clase [74]. Si bien existen técnicas para poder obtener las computaciones que hace una red neuronal, siguen siendo complicadas de interpretar [74].

Métodos de evaluación de algoritmos de clasificación

Cada algoritmo de los mencionados en el apartado anterior tiene distintas variaciones según como se ajusten los parámetros de este, por lo que para un mismo algoritmo se pueden tener muchos modelos distintos en donde cada uno entregue resultados distintos. Además, existen varios algoritmos distintos que no se presentan en el trabajo de título, lo que quiere decir que se tienen muchas maneras de resolver un problema en particular con Machine Learning. Si bien hay ciertos algoritmos que tienen un mejor desempeño que otros para ciertos problemas en específico, como Naive Bayes que suele tener buenos resultados al aplicarlo en problemas de diagnóstico médico [74], se puede dar que otro algoritmo bajo ciertas condiciones presente un mejor desempeño. Pero ¿cómo podemos medir el desempeño de un modelo y compararlo con otros para saber cuál es mejor?

Existen varias métricas que permiten evaluar modelos de clasificación de Machine Learning, como la exactitud (*accuracy)*, la precisión, recall, entre otros. A continuación se presentará el detalle de dichas métricas para evaluar los modelos de clasificación, pero antes es importante explicar algunos conceptos. A modo de simplificar la explicación se considera un problema de clasificación binaria, es decir, se tienen 2 posibles clases.

- **Verdaderos Positivos (TP):** Número de instancias en los que se predice correctamente la clase 1.
- **Falsos Positivos (FP):** Número de instancias en las que se predice incorrectamente, retornando la clase 2 cuando el valor real era la clase 1.
- **Verdaderos Negativos (TN):** Número de instancias en que se predice correctamente la clase 2.
- **Falsos Negativos (FN):** Número de instancias en que se predice incorrectamente, retornando la clase 1 cuando el valor real era la clase 2.

A modo de simplificar la interpretación de dichos valores TP, FP, TN y FN estos se representan en una matriz llamada matriz de confusión, que se puede apreciar en la figura 1.18. Dicha matriz representa que tanto clasifica correcta e incorrectamente un modelo, y en base a los valores de TP, FP, TN y FN se pueden construir algunas de las métricas que se explican a continuación [75].

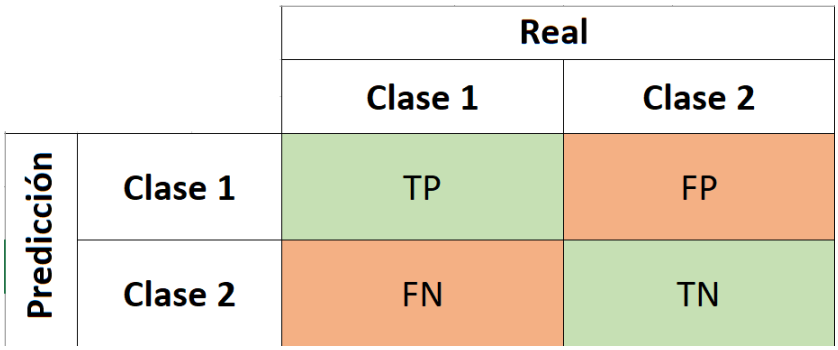


Figura 1.18: Matriz de confusión de 2 clases. *Fuente: Elaboración propia*

Exactitud (Accuracy)

La exactitud, más conocido por su nombre en inglés *accuracy* es el indicador que representa la cantidad de clasificaciones correctas del total de clasificaciones, es decir, indica la probabilidad de predecir correctamente [76]. Para calcular el accuracy se usa la siguiente fórmula:

$$accuracy = \frac{\text{Total de clasificados correctamente}}{\text{Total de registros}} = \frac{TP + TN}{TP + TN + FP + FN} \tag{1.8}$$

Precisión

La precisión también llamada valor predictivo positivo es aquel valor que indica la proporción de valores positivos predichos correctamente sobre el total de valores positivos predichos [76]. Por lo tanto, esta métrica indica cuantos valores efectivamente pertenecen a una clase de todas las asignaciones a dicha clase por parte del modelo. Puede suceder que haya clases de mayor interés a predecir, como por ejemplo el diagnóstico de una enfermedad correctamente, por sobre indicar que una persona esta sana, en cuyo caso interesa que la precisión de la clase "padece la enfermedad" sea alta. La fórmula de precisión es:

$$precision = \frac{\text{Total de clasificados correctos para la clase positiva}}{\text{Total de clasificados a la clase positiva}} = \frac{TP}{TP + FP} \tag{1.9}$$

Recall

El recall o tasa de verdaderos positivos indica la probabilidad del modelo de detectar casos realmente positivos [76]. Es decir, a cuantos se clasifican correctamente a la clase positiva, del total de registros positivos que hay. Por ejemplo, en el diagnóstico de una enfermedad interesa tener un alto recall, ya que esto significaría que del total de pacientes que sufren de una condición se está detectando a varios de estos y se clasifican de forma correcta. La fórmula de recall es:

$$Recall = \frac{\text{Total de clasificados correctos para la clase positiva}}{\text{Total de registros de la clase positiva}} = \frac{TP}{TP + FN} \tag{1.10}$$

F1-score

El puntaje F1 indica un desempeño general del modelo al combinar la precisión y el recall [76].

$$F1\text{-}Score = \frac{2xPrecisionxRecall}{Precision + Recall} \tag{1.11}$$

Curva ROC y AUC

Existen otras 2 maneras clásicas de evaluar modelos de clasificación, que son las curvas ROC (*Receiver Operating Characteristic*) y AUC que es el área bajo la curva de la curva ROC (*Area Under the Curve*). La curva ROC es un gráfico que tiene 2 ejes: la tasa de verdaderos positivos (TPR) o recall y la tasa de falsos positivos (FPR) conocido como *fallout*) [75]. Las fórmulas de ambas tasas son las siguientes:

$$TPR = \frac{TP}{TP + FN} \tag{1.12}$$

$$FPR = \frac{FP}{FP + TN} \tag{1.13}$$

Por lo tanto, la curva ROC evalúa el intercambio que hay entre ambas tasas. En la figura 1.19 se puede ver un gráfico de curva ROC, y su respectiva área bajo la curva. Debido a que dicho gráfico recorre los ejes de TPR y FPR, en donde a medida que crece el TPR es bueno pero a medida que crece el FPR es malo, si en el eje horizontal se ubica el FPR y en el vertical el TPR, lo que se busca es que la curva se desplace por el sector superior izquierdo, lo que indicaría un alto TPR y un bajo FPR. Por otro lado, mientras más cercana este la curva al sector superior izquierdo mayor será el área bajo la curva, como se puede apreciar en la figura 1.19, por lo tanto, a mayor AUC se tiene un mejor desempeño general del modelo [75]. Es importante destacar que el valor AUC toma valores entre 0 y 1.

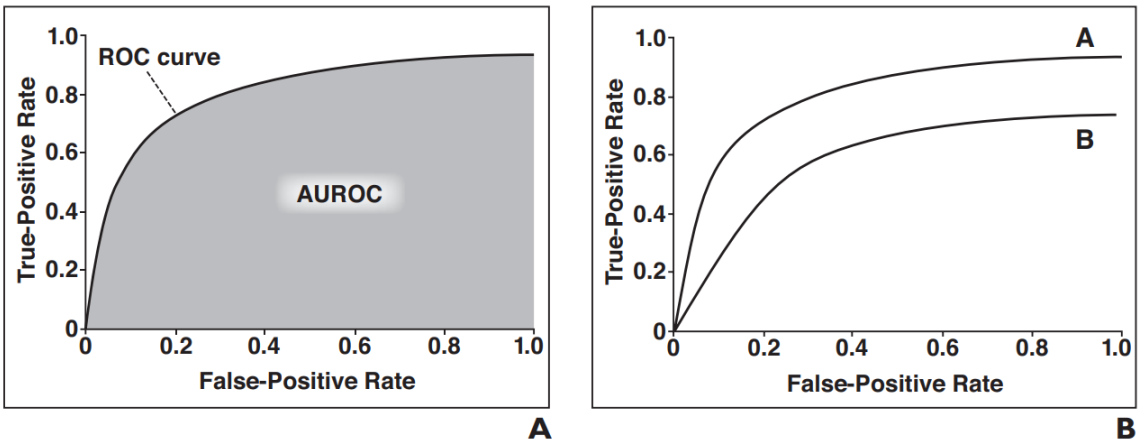


Figura 1.19: (**a**) Gráfico que muestra la curva ROC y en color más oscuro el área bajo esa curva. (**b**) 2 curvas ROC, en donde la curva A tiene un mayor AUC. *Fuente: Handelman et. al., 2019* [76]

1.6. Metodología

Cada actividad importante por realizar en el trabajo de título se puede representar como una etapa del proceso KDD. En primera instancia se recopila la información con la que se trabajará, es decir, la etapa de integración. Debido a que el mecanismo de evaluación que se está desarrollando en el proyecto Neuronat es una plataforma Serious Game, este se puede utilizar y difundir a través de un enlace de Internet. Esto permite poder generar una base de datos con las variables reales del proyecto, y con datos reales de personas. Por lo tanto, el primer paso es recolectar datos de distintas personas utilizando la plataforma Neuronat.

La segunda y tercera etapa del proceso KDD consiste en la selección y limpieza de datos, respectivamente, la cual se hace una vez ya tomadas todas las muestras requeridas, para así lograr determinar si es que existen ciertos registros anormales. Para esto se realiza un análisis exploratorio de datos. En un proyecto de datos, la etapa de selección consiste en determinar con que base se trabajará, y se puede dar el caso en que haya un proyecto con

múltiples fuentes de datos, también está el caso en que se tiene sólo una fuente de datos como sucede en el presente trabajo, sin embargo, a pesar de darse este caso, los datos deben ser trabajados y pulidos para obtener la base final con la que se trabajará, por lo tanto el proceso de selección es ese, la selección de la única fuente de datos que exporta la plataforma, y la limpieza de esta.

Debido a que los datos que se exportan de la plataforma Neuronat se almacenan en una base de datos relacional con varias tablas, es necesario transformar dicha estructura para finalmente obtener una sola tabla con las variables importantes que se utilizarán para entrenar el modelo. En conjunto con el equipo clínico del proyecto, se desarrolló una serie de variables médicas importantes que se pueden obtener del Serious Game, por lo tanto, con estos datos "crudos" se calcularon las variables definidas (se calcularon aproximadamente 70 variables de las 300 definidas), obteniendo así una tabla única. Además, se hizo un proceso en paralelo, en donde se construyó otra tabla con todas las posibles combinaciones de variables que se pueden generar con los datos "crudos" que se exportan de la plataforma, obteniendo así una segunda tabla con más de 600 variables. Lo anterior corresponde a la cuarta etapa del proceso KDD, la transformación de datos.

El quinto paso del proceso KDD corresponde a la minería, y es en este paso donde se realiza la principal investigación del trabajo de título. Con la base de datos ya generada se procedió a aplicar distintos modelos y técnicas para reducir la dimensionalidad de la base. Debido a que se definió un alto número de variables, puede haber complicaciones a la hora de analizar e interpretar los datos y los resultados [33]. Es por esto que se realizó el proceso de Feature Selection, para determinar qué variables eran las más relevantes. Una vez realizado el estudio sobre las variables, para concluir cuales son más relevantes, se procedió a entrenar modelos de predicción del segmento etario utilizando los distintos conjuntos de variables obtenidos del estudio.

Finalmente, los últimos 2 pasos corresponden a la interpretación y evaluación, y a la visualización. Una vez realizados los modelos que predicen el segmento etario, se evaluó cuál de estos era el que tenía un mejor desempeño desde el punto de vista del recall (exhaustividad), y ese modelo que presentó el mejor desempeño, fue el que entregó el conjunto de variables que mejor representa la DE en situaciones cotidianas. Para la visualización de los resultados, se resume en distintas tablas las métricas de evaluación de cada modelo. Esas tablas se pueden encontrar en el Capítulo 5.

Por último, se consideró todo el potencial mercado que podría beneficiarse de la investigación que se realizó. Esto para poder estimar el impacto que tendría el nuevo conocimiento, y a cuantas personas podría afectar, tanto pacientes como médicos y el sistema. Para realizar esto se investigó sobre las estadísticas de las enfermedades y condiciones presentadas en la figura A.1 del anexo, para así saber cuántas personas sufren de dichas condiciones, ya que todas estas personas podrían beneficiarse de los descubrimientos que se hagan en el presente trabajo de título.

1.7. Resultados esperados

Del presente proyecto de título, se espera obtener los siguientes resultados:

- Una base de datos, con registros reales y simulados del Serious Game Neuronat.
- Un conjunto de variables psicosociales que caracterice una DE en situaciones cotidianas.
- Un modelo que permita predecir el segmento etario en base a una serie de variables sobre una persona, con un accuracy y recall aceptable, es decir, que sea mejor que elegir un segmento etario al azar o incluso peor que eso.
- Una estimación de cuantas personas podrían verse afectadas por el trabajo realizado y su implementación en el proyecto Neuronat, y una estimación económica del impacto de la aplicación del proyecto.

1.7.1. Alcances

En primer lugar, es importante establecer que los datos con los que se trabajó no son confiables en su totalidad, por lo que el estudio realizado sirve netamente como una exploración inicial de lo que se puede lograr con la plataforma Neuronat.

Por otro lado, si bien se está dentro del contexto del proyecto Neuronat, los modelos implementados y los resultados obtenidos no son aplicables en su totalidad al proyecto en si, ya que no se trabajó con pacientes reales como si debe suceder en el proyecto Neuronat. A pesar de lo anterior, los modelos desarrollados fueron construidos de manera genérica, es decir, que al cambiar la base con la que se trabaje (de personas sanas de distintas edades a pacientes diagnosticados con Disfunción Ejecutiva y sujetos control) no se debe modificar en gran medida los algoritmos desarrollados, por lo que se puede utilizar el presente trabajo como guía para lo que se desarrolle en un futuro en el proyecto Neuronat.

Además, se pretende dar un enfoque social al trabajo realizado, es decir, se determina a cuanta gente podría afectar positivamente el proyecto Neuronat con resultados favorables como los obtenidos en el presente trabajo. Es por esto que no se realiza un plan de difusión del proyecto, o un pricing asociado a la plataforma Neuronat para su posterior venta a servicios privados de salud. Es importante destacar que el análisis de impacto social realizado contiene varios supuestos y aproximaciones realizadas por el autor, y debe considerarse como tal, una estimación de un potencial segmento de pacientes que podría beneficiarse de los resultados obtenidos del proyecto Neuronat.

Capítulo 2

Estado del arte

Hoy en día se tiene una cierta manera de evaluar la Disfunción Ejecutiva, la cuál es principalmente tradicional, es decir, se siguen utilizando medios convencionales como test de lápiz y papel, o entrevistas para evaluar la presencia de una DE. En el primer apartado del presente capítulo se detallará cuáles son las maneras actuales, tanto convencionales y no convencionales de evaluar una Disfunción Ejecutiva.

Posterior a eso, se expone sobre distintas aplicaciones de técnicas computacionales como Machine Learning en el área de la medicina, en particular, en el diagnóstico médico.

2.1. Evaluación y diagnóstico de la Disfunción Ejecutiva

Actualmente la Disfunción Ejecutiva se diagnostica por medio de una serie de pruebas que realizan los neuropsicólogos, además de varias entrevistas que realiza el paciente con distintos profesionales [2]. Según Vayas & Carrera (2012) se utilizan las siguientes pruebas y medidas para evaluar, diagnosticar y hacer seguimiento de las DE de los pacientes:

- **Mini-mental test de Folstein:** Esta prueba se utiliza inicialmente con los pacientes debido a que evalúa varios dominios cognitivos, sin embargo apenas evalúa las funciones ejecutivas, por lo que se hace para tener una evaluación inicial de los pacientes y entender cuál es el problema que estos sufren. En el anexo A.2 se puede ver el Mini-mental test de Folstein.

- **Test del reloj:** Se hace a los pacientes dibujar un reloj con todos sus números, manecillas y que indique una hora en particular. Este test evalúa funciones ejecutivas entre otras cosas, y presenta una escala de desempeño, en donde según el puntaje obtenido, la persona padece de algo o no. En la figura 2.1 (**i**) se puede apreciar distintos dibujos de relojes realizados por pacientes.

- **Trail making test parte B:** Se le pide al paciente que empareje números con letras consecutivamente en orden creciente. Si se tienen más de 13 errores en la lista generada se considera un posible deterioro. También se evalúa el tiempo que se demora el paciente en completar el test. En la figura 1.5 se puede ver a un paciente haciendo el Trail making test.

- **Tests de fluencia verbal:** Dada una letra, el paciente debe decir todas las palabras que pueda que partan con esa letra en un minuto. Si se tienen menos de 30 palabras (únicas, no sirve si son repetidas o conjugaciones de palabras dichas anteriormente) se considera un potencial deterioro, aunque hay que tener en consideración la edad y el nivel de escolaridad de los pacientes, ya que se espera un mejor desempeño en pacientes jóvenes con un alto nivel de escolaridad.

- **Frontal assesment battery (FAB)**: Test que consiste en la aplicación de subtests que evalúan la conceptualización, flexibilidad mental, programación motora, control inhibitorio entre otros. El objetivo de este test es evaluar las funciones del lóbulo frontal del cerebro [79].

- Se indica además que los médicos deben estar atentos al comportamiento del paciente mientras este realiza los tests de evaluación. También se realizan entrevistas de seguimiento tanto a los pacientes como a familiares o seres cercanos al paciente.

Por otro lado, Rabinovici [4] identifica una serie de test para la evaluación de distintas funciones ejecutivas en específico, como por ejemplo para memoria de trabajo se utilizan tests de repetición de actividades como repetir números en un cierto orden (digit span subtest), tocar ciertos objetos repetitivamente en un cierto orden (Corsi block tapping test) o contar y recordar puntos en una secuencia de láminas (dot counting task). Para la función de inhibición se tienen tests orientados a la respuesta rápida de estímulos visuales como decir el color de una palabra que es el nombre de otro color (Stroop test), o indicar la dirección a la cual apunta la flecha de al medio, en una imagen con varias flechas apuntando a otras direcciones lo más rápido posible (Flanker test). En la autorregulación se tienen los tests de trail making explicado anteriormente, el Wisconsin Card Sorting Test (WCST) en donde se tiene que ordenar cartas según las reglas que va cambiando el examinador, o el set shifting test en donde se tiene que unir objetos por color o forma de acuerdo con reglas repentina que indica un computador. Finalmente, para la función de fluencia está el ya explicado test de fluencia verbal. Los test de Stroop, Wisconsin Card Sorting y Flanker se pueden apreciar en las figuras A.3, A.4 y A.5 respectivamente en la sección de anexos.

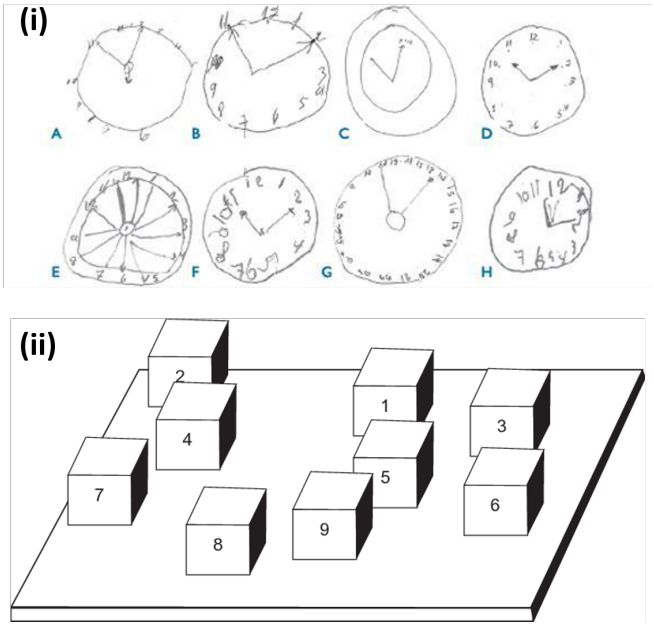


Figura 2.1: (**i**) Varias muestras de test del reloj. (**ii**) Corsi block tapping test, en donde el paciente debe tocar los bloques repetidamente en el orden numérico. *Fuentes: Ciberwatch* [77] *y Berch, Krikorian & Huha (1998)* [78]

Sin embargo, dichos test cuentan con la misma limitación, que al realizarse en un entorno clínico se deja de lado la cotidianeidad, lo que puede llevar al error en el diagnóstico de DE en situaciones cotidianas. Es por esto que se han desarrollado otros mecanismos para diagnosticar o evaluar las funciones ejecutivas como el test BRIEF, en donde se entrevista a los padres y profesores de niños y adolescentes que puedan tener alguna Disfunción Ejecutiva. En base a las respuestas de éstos, se crean distintos puntajes y se evalúa el desempeño de los pacientes. Este test ha demostrado tener una alta validez ecológica, lo que quiere decir que lo que indica el test se puede extrapolar a lo que sucede en el mundo real, sin embargo, este test está orientado a niños y adolescentes principalmente [18].

Otro mecanismo de evaluación consiste en la evaluación y observación naturalista de los pacientes, en donde se analiza el comportamiento de estos en actividades reales y cotidianas, como por ejemplo en el estudio realizado por McAlister & Schmitter-Edgecombe, en donde se evaluó a adultos y jóvenes sanos en la realización de actividades cotidianas en un departamento del campus de Washington State University [21]. Este tipo de estudios son los óptimos debido a que presentan la mayor validez ecológica ya que no son experimentos en un laboratorio en los que se tenga que extrapolar sus resultados al mundo real, sino que es el mundo real en sí, sin embargo, presentan el gran problema de que no son fáciles de implementar en un entorno clínico habitual como se pudo apreciar en el experimento de McAlister & Schmitter-Edgecombe en donde se utilizó un departamento, no una consulta clínica [21].

Finalmente se encuentra el uso de entornos virtuales, o uso de herramientas de realidad virtual para evaluar a los pacientes. El caso del Mall Virtual (VMall) es el que presenta más documentación. Este consiste en una plataforma que permite a los pacientes comprar y elegir productos, y realizar actividades cotidianas a través de una aplicación [20]. Esta se utiliza

en el tratamiento de pacientes que han sufrido un accidente cerebrovascular, entregando resultados prometedores debido al interés que genera en los pacientes y la realización de actividades reales a diferencia de los test convencionales [20]. El entorno del mall virtual también fue utilizado para investigación sobre la validez ecológica de los entornos virtuales al comparar el desempeño de pacientes al realizar un test dentro del entorno virtual (VMall), y al realizar el mismo test pero en un mall físico. Los resultados de dicho estudio indican que existe una alta correlación entre los resultados del entorno virtual y el mall real, pudiendo concluir que efectivamente sí se tiene validez ecológica en el VMall, abriendo la posibilidad de generar evaluaciones virtuales que sí tengan validez ecológica y permitan diagnosticar la presencia de DE [30]. Sin embargo, no se puede encontrar mucha documentación al respecto de este tipo de test de evaluación, lo que indica que su uso no es muy frecuente, y de lo que se sabe es que son caros de implementar y existen tipos de pacientes que no aceptan muy bien este tipo de pruebas, como pacientes con enfermedades neurodegenerativas. En la figura 2.2 se puede apreciar una captura de la plataforma VMall.

Figura 2.2: Captura de la plataforma Virtual Mall. *Fuente: Rand et. al (2005)* [20]

No se encontró documentación sobre la implementación de modelos de Machine Learning para ayudar en la evaluación y diagnóstico de DE, sin embargo sí existen variados estudios sobre la implementación de técnicas de Machine Learning aplicadas a la medicina en otras áreas, como por ejemplo en análisis de imágenes para diagnosticar Alzheimer [8], o el uso de Machine Learning para diagnosticar ansiedad y depresión en niños [9], entre otros.

2.2. Uso de técnicas computacionales y Machine Learning en la medicina

Cómo se mencionó anteriormente, existen varios estudios de la aplicación de técnicas de Machine Learning en problemas relacionados al área de la medicina. En particular, para el presente trabajo de título las áreas de la medicina más relevantes serían la de diagnóstico de enfermedades o condiciones, la de salud mental y la neurociencia, debido a que el objetivo último del proyecto Neuronat, y el aporte que se desea realizar en el presente estudio es

lograr determinar las características o variables que permitan diagnosticar una Disfunción Ejecutiva.

El 2001, Igor Kononenko publica su estudió sobre el uso de Machine Learning en el diagnóstico médico, revisando la historia, el estado del arte y los próximos pasos que se deberían seguir [74]. En dicho estudio Kononenko afirma la importancia del uso de técnicas de Machine Learning en el diagnóstico médico, ya que estas se pueden utilizar para asistir y apoyar a los médicos en su diagnóstico, aumentando así la velocidad a la cual se diagnostica y la precisión con la que se diagnostica [74]. Kononenko en particular habla de la relevancia de los métodos *Naive Bayes, redes neuronales y árboles de decisión.*

- **Naive Bayes:** Según Kononenko, este es el modelo de Machine Learning que presenta mejores resultados en varios casos, y en caso de no entregar el mejor resultado, si da una buena base para poder comparar otros modelos. El aspecto más positivo de Naive Bayes de acuerdo con los médicos colaboradores de Kononenko es que las "decisiones" que toma Naive Bayes se pueden interpretar como la suma de la información obtenida de todos los atributos, en donde esta información puede ser positiva o negativa. Por lo tanto, cada variable aporta con distintos pesos a la decisión que toma el modelo (clase a asignar o valor a predecir), y según los médicos, esto es similar a como ellos realizan un diagnóstico. En la figura 2.3 se puede apreciar el ejemplo que muestra Kononenko.

- **Redes neuronales:** Este tipo de algoritmos entregan buenos resultados de diagnóstico, sin embargo tienen el gran problema de ser una caja negra, por lo que no es simple interpretar los resultados y determinar cuál es el efecto de cada variable. Se han desarrollado métodos para poder extraer los patrones que se detectan de las redes neuronales con el fin de detectar las reglas de clasificación o predicción, sin embargo estas reglas suelen ser muy largas y complejas, lo que dificulta su interpretación.

- **Árboles de decisión:** La principal ventaja de este tipo de algoritmos es su fácil interpretación, sin embargo el aspecto negativo de los árboles de decisión es que para poder ser interpretables, se reduce considerablemente el tamaño del árbol, ignorando quizás el efecto de otras variables menos significativas pero igualmente relevantes.

Table 2
Semi-naive Bayes: an explanation of a decision in the femoral neck fracture recovery problem[a]

Attribute value	For decision (bit)	Against decision (bit)
Age: 70–80	0.07	
Sex: female		−0.19
Mobility before injury: fully mobile	0.04	
State of health before injury: other	0.52	
Mechanism of injury: simple fall		−0.08
Additional injuries: none	0.00	
Time between injury and operation >10 days	0.42	
Fracture classification according to Garden: Garden III		−0.30
Fracture classification according to Pauwels: Pauwels III		−0.14
Transfusion: yes	0.07	
Antibiotic profilaxis: yes		−0.32
Hospital rehabilitation: yes	0.05	
General complications: none		−0.00
Combination		
Time between injury and examination <6 h + hospitalization time between 4 and 5 weeks	0.21	
Artroplastic + anticoagulant therapy: yes	0.63	

[a] Decision: no complications (correct).

Figura 2.3: Ejemplo de Naive Bayes en diagnóstico médico. *Fuente: Kononenko (2001)* [74]

En particular, es importante revisar el estado del arte en el uso de técnicas de Machine Learning en el diagnóstico de enfermedades o patologías asociadas al área de salud mental, dada la índole de la Disfunción Ejecutiva. Shatte et. al., realizó el año 2018 (publicado el 2019) un estudio sobre el uso de técnicas de Machine Learning en la salud mental, revisando distintos tópicos como la detección y diagnóstico de una enfermedad o condición, pronostico y tratamiento de enfermedades, salud pública e investigación y administración clínica [80]. Para el caso de detección y diagnóstico se revisaron estudios que apuntan al desarrollo de herramientas de prediagnóstico (se utilizan para evaluar a un paciente que potencialmente padezca de la enfermedad), y herramientas de identificación de riesgo de padecer alguna enfermedad en un futuro (detectar que tan probable es que una persona desarrolle una cierta patología). Algunas de las enfermedades tratadas para el estudio de detección y diagnóstico son Alzheimer, ansiedad, autismo, Parkinson, traumatismo craneoencefálico, depresión y esquizofrenia, que son enfermedades que pueden llevar a una Disfunción Ejecutiva [4] [80]. De acuerdo con Shatte et. al., el Machine Learning tiene el potencial de mejorar procesos clínicos, y de generar nuevos descubrimientos acerca de la salud mental y el bienestar. Sin embargo es importante destacar que el uso de Machine Learning en ningún caso reemplaza otros mecanismos de evaluación, sino que aporta un gran valor a la investigación acerca de la salud mental y apoya a los especialistas. Además, una de las limitaciones que se tiene hasta el momento en el uso de técnicas de Machine Learning en el diagnóstico de enfermedades asociadas a la salud mental es la aplicación en entornos reales y cotidianos, es decir, fuera de los experimentos de laboratorios en el cual se enmarcan las investigaciones, y es justo dicho problema el que se intenta abordar en el presente trabajo de título y en el proyecto Neuronat.

Capítulo 3

Generación y almacenamiento de los datos

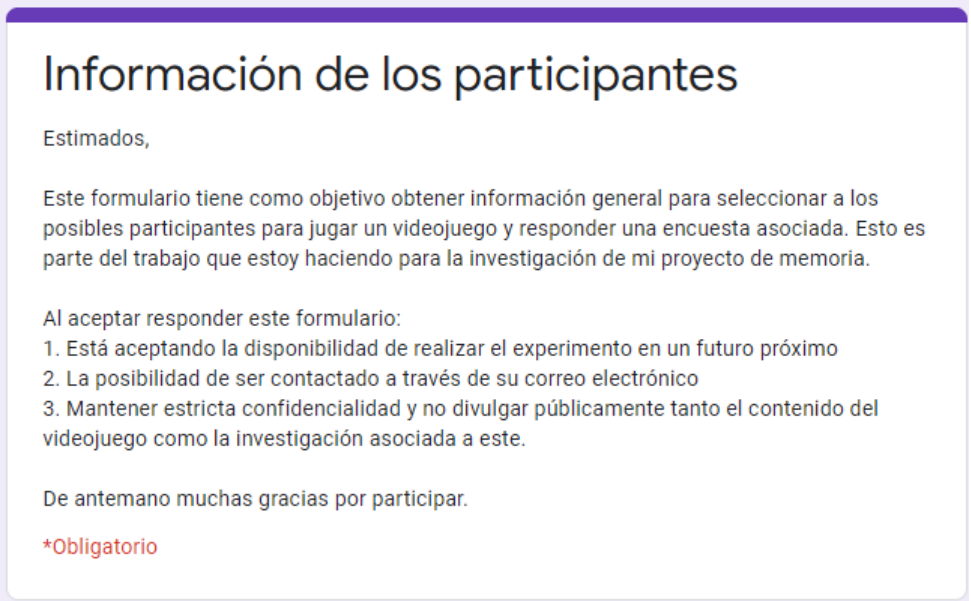
El proyecto Neuronat es un proyecto FONDEF que está siendo desarrollado por la Facultad de Medicina de la Universidad de Chile y el Web Intelligence Centre. Como se comentó en el Capítulo 1, dicho proyecto busca apoyar en el diagnóstico de Disfunción Ejecutiva en contextos cotidianos a través de una plataforma Serious Game, es decir, un videojuego serio que es utilizado con fines distintos al ocio, como en este caso es un diagnóstico médico [13]. El contexto del Serious Game es un restorán, en donde el jugador, o potencial paciente toma el rol del mozo de dicho restorán. Parte del proyecto, es la toma de muestra de pacientes reales que estén diagnosticados con Disfunción Ejecutiva, y de sujetos control (personas sanas).

El presente trabajo de título, apoya al proyecto Neuronat en la definición de las variables clínicas que permiten caracterizar un desempeño menos eficiente de las funciones ejecutivas, lo cual se puede extrapolar al padecimiento de una Disfunción Ejecutiva. Para el ya mencionado trabajo de título se necesitaba de una base de datos para poder aplicar la metodología KDD, sin embargo, debido a la complejidad de obtener muestras de pacientes reales como se hará en el proyecto, es que se decidió obtener muestras de sujetos sanos, de distintos rangos etarios (18-29, 30-40, 41-51 y 52-62), para así poder utilizar la edad como un indicador de desempeño de función ejecutiva.

Antes que nada, es importante explicar por qué es complejo obtener una muestra de pacientes reales, y por qué es factible obtener muestras de personas sanas. Para obtener las muestras se utilizó la misma plataforma Neuronat, y debido a la pandemia del COVID-19 la toma de muestras debía realizarse de forma remota. Es por esto que, para obtener muestras se tenía que difundir un enlace a través del cual las personas podrían acceder desde Internet a la plataforma. En el caso de pacientes reales, primero deben estar previamente diagnosticados con Disfunción Ejecutiva por médicos. Además, el daño sufrido por estas personas no debe ser tan profundo debido a que eso generaría casos muy extremos en los resultados de la plataforma, más conocido como *outliers*, y esto hace que disminuya considerablemente la población objetivo. Sin embargo, a pesar de ser casos no tan graves de Disfunción Ejecutiva, si sería necesaria la constante supervisión de los pacientes a la hora de realizar el test con la plataforma, debido a que parte de los síntomas que pueden sufrir los pacientes son la pérdida de atención, la repetición de una tarea, la imposibilidad de terminar o empezar tareas entre otras, lo cual haría que el desarrollo mismo del Serious Game no avance [3]. Por

otro lado, si las muestras obtenidas provinieran de personas sanas no se tendría que estar necesariamente supervisando el desarrollo del test con la plataforma, ya que no sufrirían los síntomas mencionados anteriormente. Entonces, con la correcta instrucción, personas sanas podrían completar correctamente el Serious Game Neuronat, y así generar datos reales, y como se abordó en el capítulo 1 y 2, existen estudios que reflejan que entre personas sanas, la edad si es un indicador de desempeño de distintas funciones ejecutivas, teniendo las personas mayores una disminución en su desempeño en comparación con los más jóvenes, por lo que se utilizó la edad de las personas para efectivamente estimar que variables caracterizan de mejor manera el desempeño de las funciones ejecutivas.

Por lo tanto, se obtuvieron datos de personas sanas para el estudio del presente trabajo de título. El primer paso para la obtención de los datos fue la definición de los criterios de aceptación de los sujetos de prueba. Estos criterios se definieron en conjunto con el equipo clínico del proyecto Neuronat, debido a que es fundamental que las personas que participen del estudio no padezcan, o hayan padecido recientemente condiciones como alcoholismo, uso de drogas, Parkinson, Alzheimer, entre otras. Esto debido a que el padecimiento de estas condiciones puede afectar en el desempeño de las funciones ejecutivas, "ensuciando" así los datos. Una vez definidos los criterios de aceptación, se procedió a crear un formulario en línea para obtener información de distintas personas (potenciales sujetos de prueba), para corroborar si estas personas cumplían con los requisitos que se definieron en conjunto con el equipo clínico. El formulario de Google puede ser difundido y contestado por Internet. En este se le deja en claro a los encuestados el contexto en el cual se realiza dicho formulario, y se les informa que de acceder a contestarlo están aceptando los términos que se imponen, que son: mantener la confidencialidad tanto del proyecto Neuronat como del trabajo de título realizado y estar disponible en caso de que se llegue a contactarlos para poder realizar el experimento. En ese formulario se solicita información personal como el género y la edad, además de preguntas médicas relacionadas con el historial médico de las personas. En la Figura 3.1 se puede apreciar el inicio del formulario con el contexto explicado y las condiciones que deben aceptar.



Información de los participantes

Estimados,

Este formulario tiene como objetivo obtener información general para seleccionar a los posibles participantes para jugar un videojuego y responder una encuesta asociada. Esto es parte del trabajo que estoy haciendo para la investigación de mi proyecto de memoria.

Al aceptar responder este formulario:
1. Está aceptando la disponibilidad de realizar el experimento en un futuro próximo
2. La posibilidad de ser contactado a través de su correo electrónico
3. Mantener estricta confidencialidad y no divulgar públicamente tanto el contenido del videojuego como la investigación asociada a este.

De antemano muchas gracias por participar.

*Obligatorio

Figura 3.1: Contexto y consentimiento informado del formulario que se utilizó para buscar participantes de la toma de datos con el videojuego. *Fuente: Elaboración propia*

Utilizando el formulario presente en la Figura 3.1 se recopilaron datos de 148 personas, de los cuales 123 cumplen con los criterios previamente definidos en conjunto con el equipo clínico. En la figura 3.2 se puede apreciar la cantidad de personas que hay por segmento etario.

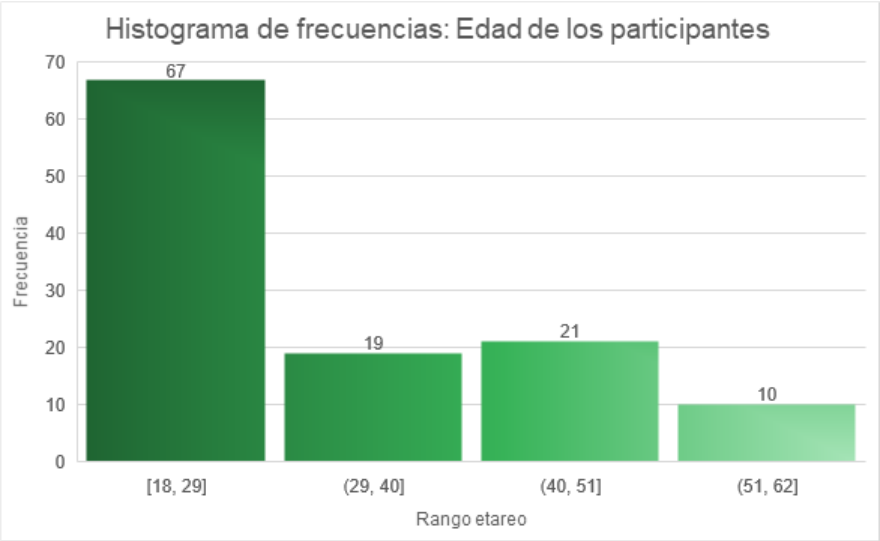


Figura 3.2: Histograma que muestra la distribución de personas según el segmento etario al cual pertenecen. *Fuente: Elaboración propia*

Al ver el gráfico de la figura 3.2 se puede apreciar que las clases están claramente desbalanceadas, es decir, que no todos los segmentos tienen la misma cantidad de observaciones. Debido a que los modelos entrenados utilizan dicha variable de segmento etario como variable a predecir, el hecho de tener una clase desbalanceada puede producir un sesgo en la muestra,

llegando así a aumentar el peso de dicha clase al momento de predecir. Debido a lo anterior, y al tiempo requerido para tomar muestras de 123 personas, es que se decidió acotar la muestra a 40 personas, teniendo 10 personas por clase. Para esto se eligió aleatoriamente 10 personas de cada segmento, para disminuir lo más posible los sesgos que pueda presentar la base. De esta manera, queda un gráfico como el de la figura 3.3, en donde se tiene la misma cantidad de personas para cada clase o segmento etario. Existen otras maneras de hacerse cargo de una muestra desbalanceada, como por ejemplo simular datos, ya sea replicando datos existen con leves variaciones (suele aplicarse a imágenes, en donde de una sola imagen se pueden generar 3 nuevas al rotar la original), o simulando nuevos registros en base a los valores de las variables que se tienen de la clase menos populada. Dichas técnicas se descartan debido a la poca cantidad de datos que se tienen como para simular grandes volúmenes de datos (por ejemplo en el caso del último rango etareo, para poder tener la misma cantidad de datos que el primer rango etareo se deben crear 57 registros en base a los valores de 10 registros que se tienen originalmente). Otra manera de hacerse cargo de clases desbalanceadas es utilizar algoritmos de Machine Learning que según la documentación suelen tener buenos resultados para problemas de clasificación con clases desbalanceadas como lo son árboles de decisión o modelos con penalización, en donde se toma en cuenta la cantidad de registros por clase. Más adelante se explica que efectivamente se implementa el algoritmo de Random Forest, a pesar de tener las clases ya balanceadas luego de reducir la cantidad inicial de datos y de simular algunos registros.

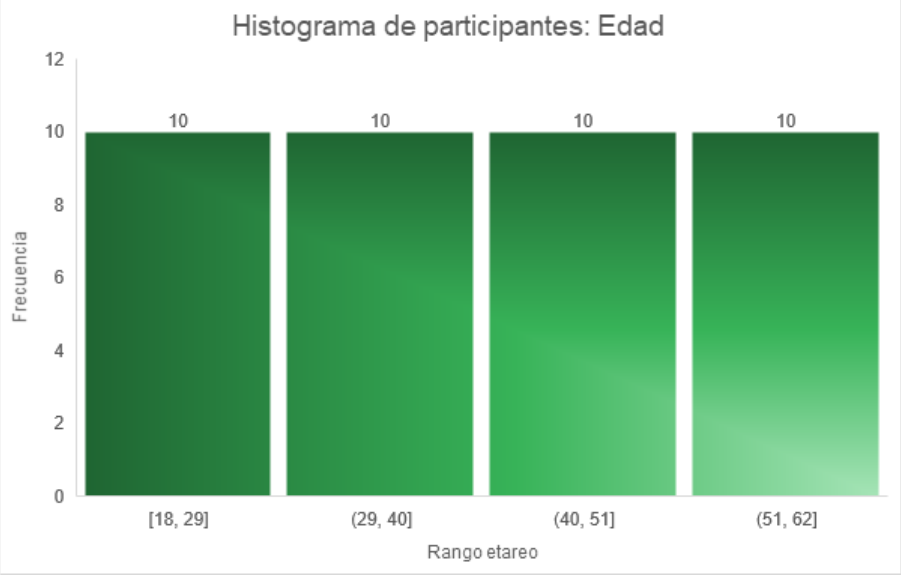


Figura 3.3: Histograma que muestra la distribución de personas según el segmento etario al cual pertenecen - filtrado. *Fuente: Elaboración propia*

Una vez realizada la encuesta, se procedió a elaborar un tutorial de cómo funciona la plataforma Neuronat, para así poder explicar a las personas que participaron del estudio cómo funcionaba la plataforma, cuál era el objetivo y que interacciones debían hacer y cómo hacerlas. Este tutorial constaba de 2 partes, en donde la primera era un documento escrito en el que se mostraba el contexto del proyecto Neuronat, el objetivo del trabajo de título y se explicaban las distintas funcionalidades de la plataforma a través de imágenes de la

misma. La segunda parte del tutorial consistía en un video donde se explicaba visualmente cómo funciona la plataforma además de los objetivos del proyecto Neuronat y del videojuego mismo. En la Figura 3.4 se puede apreciar una imagen del video subido a YouTube.

Figura 3.4: Tutorial en video de la plataforma Neuronat subido a YouTube. *Fuente: Elaboración propia*

Con los sujetos de prueba seleccionados y los tutoriales ya elaborados se procedió a contactar a las personas para fijar un horario en el que se puedan conectar a realizar la prueba.

Cada registro de la plataforma se fue almacenando en una base de datos alojada en un servidor del WIC. De esta manera, se podría acceder a dichos datos para trabajarlos y crear los modelos que se plantearon en el capítulo 1. Para lo anterior, es fundamental entender cuál es la estructura de los datos que entrega como resultado la plataforma, y cuál es la estructura de la base de datos en la que se almacenan dichos valores obtenidos de la plataforma.

El Serious Game Neuronat como fue explicado anteriormente está ambientado en un restorán. En dicho restorán el jugador toma el rol de mozo y debe atender a distintos clientes. La estructura del juego se divide en tipos de actividades que debe realizar el mozo como la toma de orden o realizar la transacción final de una mesa. Es por esto, que los datos generados también se pueden separar en este tipo de actividades, por lo que hay una cantidad acotada de variables asociadas al proceso de toma de orden, al igual que hay variables asociadas a la entrega de orden, a la transacción final, y así con todas las actividades fundamentales que debe realizar el jugador.

Por otro lado, el proyecto Neuronat busca dar libertad a los pacientes que utilicen la plataforma, en el sentido de que estos puedan elegir libremente el camino a seguir en el transcurso del juego, es decir, si quieren atender a la mesa 2 antes que la 1 que puedan hacerlo. Sin embargo, hay ciertas restricciones sobre las libertades que se le dan a los jugadores, ya que de lo contrario, no se aseguraría el fin de las actividades (el jugador podría hacer lo que quisiera y no seguir la línea general de la historia del Serious Game). Es por esto, que se arman distintos caminos posibles que puede seguir un jugador dentro de la plataforma. En

la Figura 3.5 se puede ver una representación de todos los potenciales caminos que puede seguir un jugador, teniendo así 32 finales posibles (puntos rojos). Por lo tanto, es importante obtener la información sobre el camino que sigue cada jugador, y es por esto que se le asignó un valor a cada nodo del árbol de la Figura 3.5, y finalmente el conjunto de nodos que se almacena en la base de datos representa el camino que siguió un jugador.

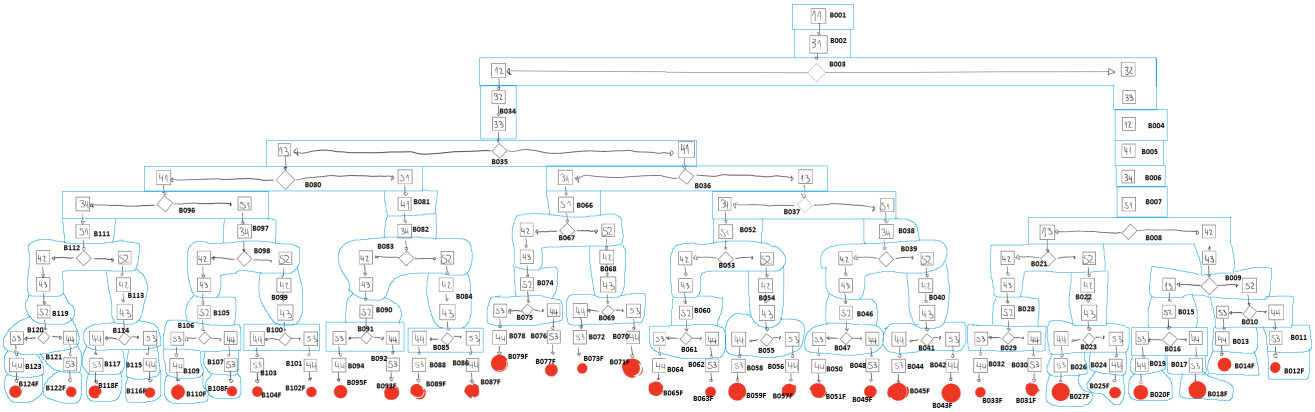


Figura 3.5: Representación visual en forma de árbol que representa los distintos caminos que puede tomar un jugador dentro de la plataforma Neuronat. *Fuente: Web Intelligence Centre*

Finalmente, es importante destacar que dentro del contexto de Neuronat existen 5 clientes que pueden interactuar en simultáneo en un mismo nodo. Por ejemplo, en un nodo en específico del árbol mostrado en la Figura 3.5 puede suceder que la mesa 1 esté en la actividad de toma de orden, mientras que la mesa 5 esté en la actividad de transacción.

Dado todo lo anterior, se tiene que los datos generados en la plataforma siguen una estructura como la que se ve en el código B.1 de anexos. En dicha estructura se tiene información sobre los estados del juego, las variables de cada estado y los datos generales de cada jugador, es decir, los datos demográficos, los cuales se almacenan en un nivel paralelo al de los estados. Las variables son exportadas en un formato JSON.

Para poder almacenar todos los valores explicados anteriormente en una base de datos se crea un modelo de base de datos. Dada la estructura que tiene el archivo JSON que exporta la plataforma Neuronat, una base de datos de tipo relacional es lo más conveniente, ya que se tiene distintos elementos o tipos de datos que siguen una estructura consistente a lo largo de todo el desarrollo de Neuronat, pudiendo así ser identificados como entidades. En la figura 3.6 se puede apreciar el diagrama Entidad Relación que modela la base de datos que almacena los datos de la plataforma Neuronat.

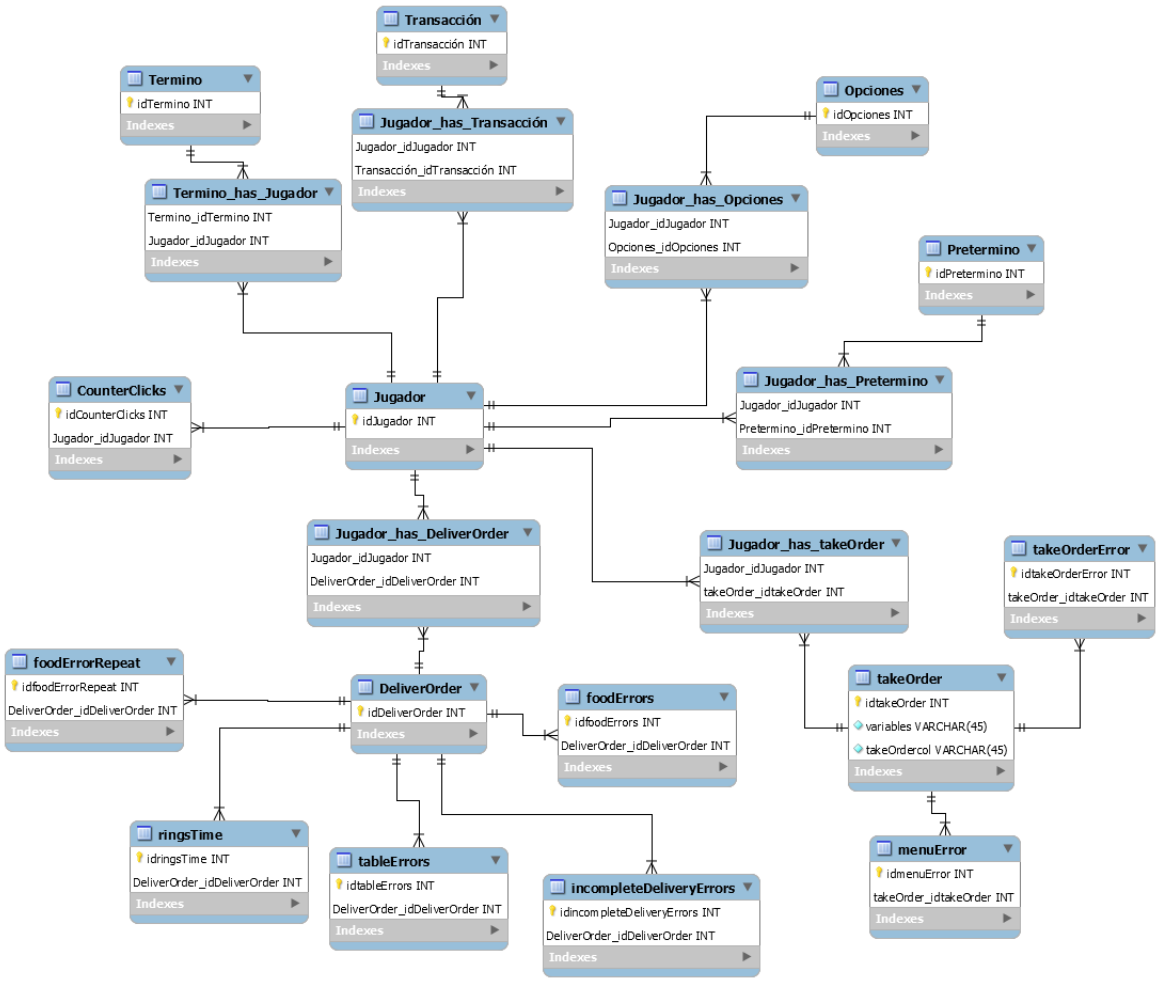


Figura 3.6: Diagrama Entidad Relación de la base de datos que almacena los datos de la plataforma Neuronat. *Fuente: Elaboración propia*

La Figura 3.6 es simplemente el modelo o la ilustración de la estructura que tiene la base de datos, por lo que posterior a la definición del modelo, se tuvo que implementar. Además, el videojuego está alojado en un servidor de Amazon Web Services (AWS), y se puede acceder a este a través de un enlace de Internet. Lo anterior implica que el videojuego no se encuentra alojado en el mismo sitio en donde se implementó la base de datos (servidor del WIC), por lo que se debían enviar los datos desde el videojuego a la base de datos. Es por esto que se creó una API REST que permite el traspaso de datos a través de un enlace de Internet conocido como *endpoint*. Cómo ya se explicó en el marco conceptual una API REST es una interfaz a través de la cual se puede enviar y recibir información utilizando las ya mencionadas URI. Por lo tanto, un *endpoint* es simplemente la ruta de Internet con la cual se puede conectar cualquier aplicación o sitio web con la base de datos. Cada *endpoint* puede tener asociados distintos métodos, siendo los principales GET y POST, en donde GET se suele utilizar para obtener datos desde la base de datos y POST se utiliza para crear nuevos datos en la base de datos.

Para configurar la base de datos siguiendo la estructura del modelo de la Figura 3.6 y la

API REST necesaria para conectar el videojuego con la base, se utilizó Django y la aplicación Django rest-framework, debido a la simplicidad de su uso y configuración, y la experiencia en este framework que se tiene. Lo primero que se hizo fue configurar el modelo de base de datos, en donde cada clase corresponde a una entidad. En el código B.2 de anexos se puede apreciar un ejemplo de 2 clases correspondiente a 2 entidades del modelo de la Figura 3.6.

Ya programado el modelo de base de datos se procedió a implementar Django rest-framework. La implementación de dicha aplicación consta principalmente de 3 factores: los serializers, las funciones views y las urls. Los serializers permiten interpretar cada elemento de una entidad de la base de datos, por lo tanto, existe un serializer por cada clase del modelo que se tenga. Las funciones views por otro lado, son funciones que determinan que se hace con cada *endpoint* que se tenga, es decir, se le asigna un método a cada ruta que se tenga. Esas funciones llaman a los serializers para poder interpretar los datos de la base y se puedan visualizar en la API REST. Se tiene una función específica (*postNeuronatData*) programada para recibir datos desde la plataforma y almacenarlos en la base de datos. Finalmente, las urls asocian las funciones de views con la ruta que tendrá cada *endpoint*. En la Figura 3.7 se puede apreciar cómo se ve la API REST en el *endpoint* "take_order", que utilizando el método GET obtiene todos los valores de la entidad Take Order. En los códigos B.3 y B.4 del anexo se puede apreciar como se ve una clase de serializer y la función *postNeuronatData*.

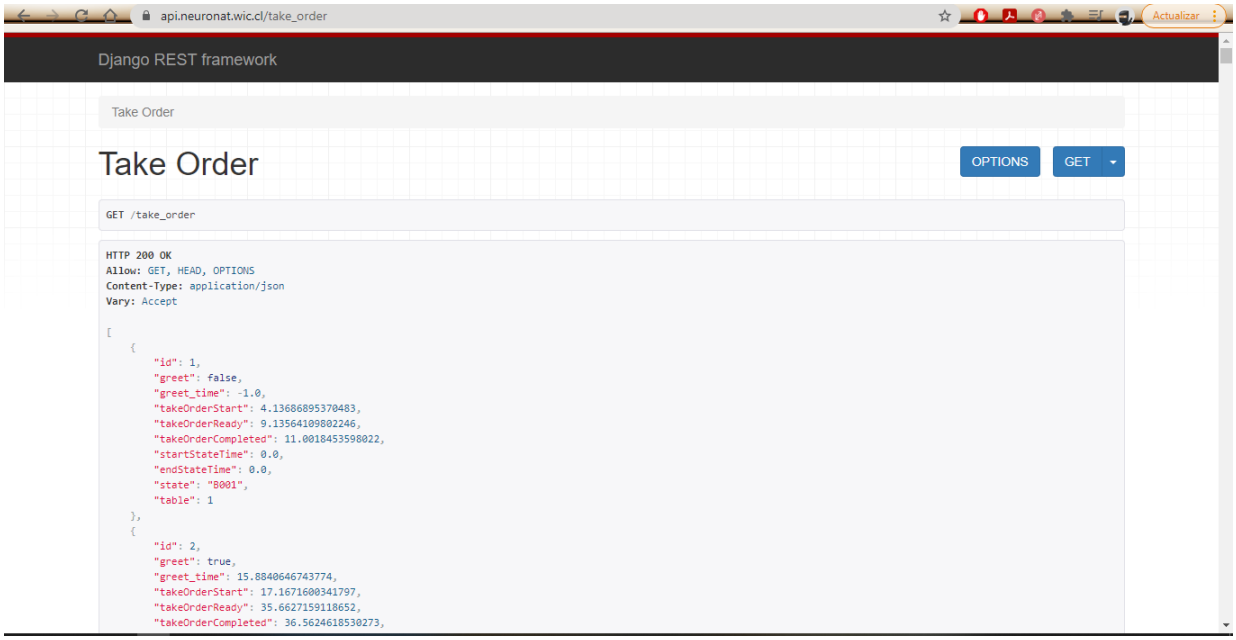


Figura 3.7: Visualización de la API REST en el *endpoint* de la tabla Take Order. *Fuente: Elaboración propia*

Sin embargo, todo lo anterior se configuró de manera local, lo que significa que no se puede acceder a la base a través de Internet, por lo que se debe asignar un dominio de Internet al proyecto de Django. El ingeniero a cargo del proyecto Neuronat en el WIC realizó dicha labor, y configuró el proyecto de Django para que se pueda acceder a este utilizando la dirección *api.neuronat.wic.cl/* en Internet, quedando de esta manera lista la base de datos para poder recibir los datos que se generan en la plataforma Neuronat. En la figura 3.8 se

puede apreciar el esquema general que resume la interacción entre los distintos elementos del proyecto de Django creado.

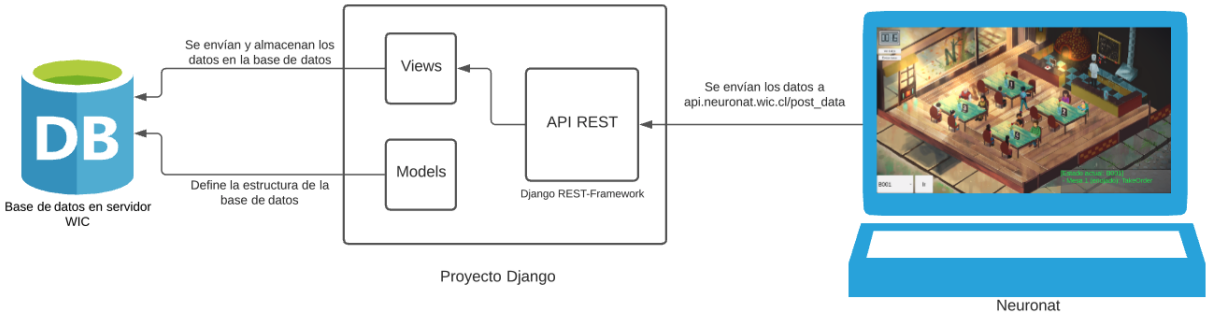


Figura 3.8: Esquema del proyecto Django. *Fuente: Elaboración propia*

Con todo lo anterior listo y la plataforma Neuronat lista, se difundió el enlace que permite jugar en la plataforma a los 40 seleccionados. Lamentablemente, no todos contestaron, por lo que se contactó a más gente de la población de 123 personas que cumplían con los requisitos planteados en el formulario de la figura 3.1, intentando siempre mantener las clases lo más balanceadas posible. A pesar de todos los intentos, por el hecho de tener una fecha límite no se pudo seguir con la toma de datos, quedando así 27 registros, con la distribución de edades que aparece en la figura 3.9.

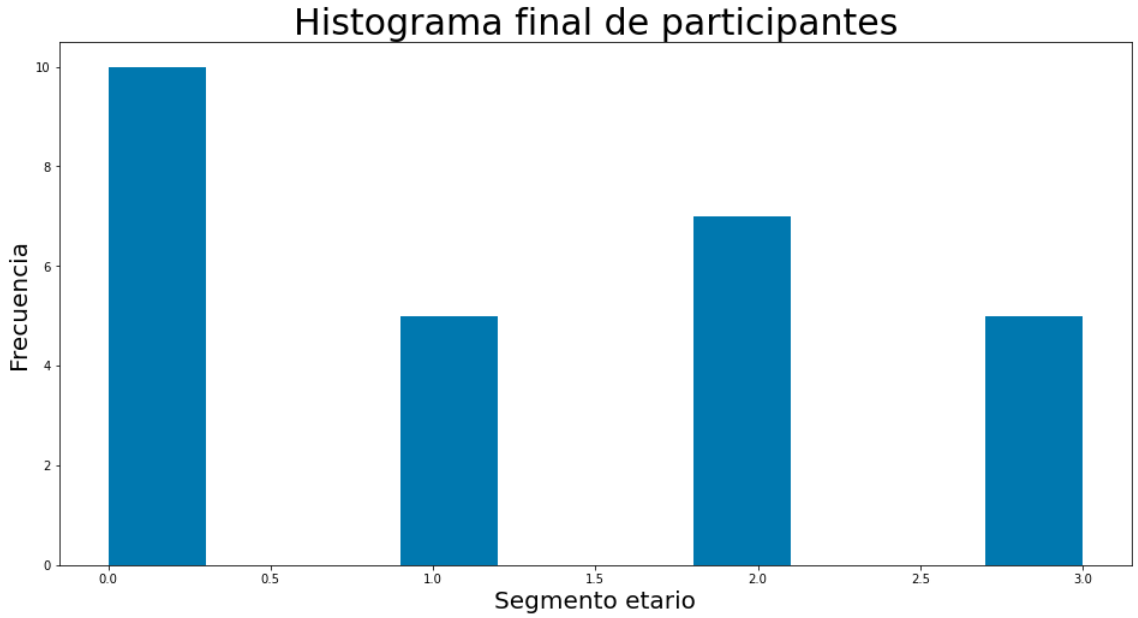


Figura 3.9: Histograma final de los participantes según el segmentario etario al cual pertenecen. *Fuente: Elaboración propia*

Para poder aumentar la cantidad de registros obtenidos, se separó la muestra en conjunto de entrenamiento y conjunto de test (en el próximo capítulo se profundiza más al respecto), quedando así el primero con 22 registros y el segundo con 5. Luego, al estar desbalanceadas las clases se aplicó una librería de Python llamada SMOTE a ambos conjuntos, la cual permite simular registros para dejar así todas las clases con la misma cantidad de datos (o muy parecidos). Esto hace que finalmente se tengan 38 registros en total (11 simulados), 32 de entrenamiento y 6 de test.

Capítulo 4

Selección de variables y modelos de clasificación

En el presente capitulo se mostrarán principalmente los distintos conjuntos de datos con los que se trabajará, y los modelos entrenados con dichos modelos. Para la construcción de los conjuntos de datos se utilizarán técnicas de Feature Engineering (ingeniería de características) y Feature Selection (selección de características). En particular se construirán 2 bases iniciales, una automática y otra manual. Luego aplicando filtros y wrappers de Feature Selection se reducirán las variables y se generarán distintos conjuntos de datos. En el tercer apartado del capítulo se explicará cómo se construyen los distintos modelos de Machine Learning utilizando los conjuntos de variables generados.

4.1. Ingeniería de características

Cómo se mencionó en el Marco Conceptual del capítulo 1, la ingeniería de características corresponde a la aplicación de técnicas para construir variables en base a datos existentes. En el capítulo anterior se mostró cuál es la estructura de los datos que se exporta desde la plataforma Neuronat, y cómo se almacenan estos datos en un servidor del Web Intelligence Centre. En este apartado se muestra cómo se trabajaron dichos datos para poder construir 2 bases, en donde las filas corresponden a los jugadores, mientras que las columnas a las variables calculadas, como por ejemplo, tiempo en resolver alguna actividad, cantidad de errores al resolver un problema, etc. De esas 2 bases una se construyó de forma automática y la otra de forma manual.

4.1.1. Ingeniería de características automática

Para la creación de nuevas variables a partir de una base con varias tablas o entidades es importante detectar cuál es el nivel de las filas que se desea. Por ejemplo, en un típico problema en una empresa quizás se desee tener una granularidad de ventas, es decir, que cada registro en la tabla final sea una venta, y las columnas estén en función de las ventas. Un ejemplo de columnas podría ser el dinero de una venta, que corresponde a la suma de los precios de los productos vendidos en una transacción. Lo importante a entender es que la granularidad es importante, y que las variables se calculan en función de cada registro utilizando ciertos elementos matemáticos como sumas, multiplicaciones, conteos, mínimos,

máximos, etc.

En el caso de este trabajo de título se desea que cada registro sea un jugador, para así poder calcular las métricas de desempeño de cada jugador. Como se vio en el capítulo anterior, la base de datos sigue la estructura mostrada en la Figura 3.6, en donde la entidad Jugador es la entidad central del modelo. Por lo tanto, las métricas que se crean utilizan como base los registros de dicha tabla, y se aplican las funciones matemáticas mencionadas anteriormente sobre las otras tablas. En el siguiente ejemplo se explica un caso, el cuál se puede ver ilustrado en la figura 4.1:

Existe una relación entre los jugadores y las distintas etapas del juego, como la toma de orden, o la entrega de los pedidos. A su vez, algunas actividades tienen otras subrelaciones, como en el caso de toma de orden que se relaciona con otra tabla que hace referencia a los errores en la toma de orden. Por lo tanto, para un mismo jugador se tienen por lo menos 4 registros de toma de orden, ya que les toma el pedido a las 4 mesas y cada una de estas deja su registro, y además cada toma de orden puede estar asociada a cero o más errores, los cuales se registran en la tabla de errores de toma de orden. Por lo tanto, si uno quisiera tener métricas asociadas a la cantidad de errores que se tiene en las tomas de orden de las distintas mesas lo que se hace es: para la toma de orden de una mesa de un jugador específico, se cuentan cuantos registros de errores de toma de orden hay relacionados a esa toma de orden específica. Por ejemplo, en la tabla de error de toma de orden hay 3 registros de errores relacionados a la toma de orden de la mesa 1 del jugador 1, por lo que en la base final aparecería una variable con el valor 3, que indicaría efectivamente la cantidad de errores al momento de tomar la orden de la mesa 1 para ese jugador.

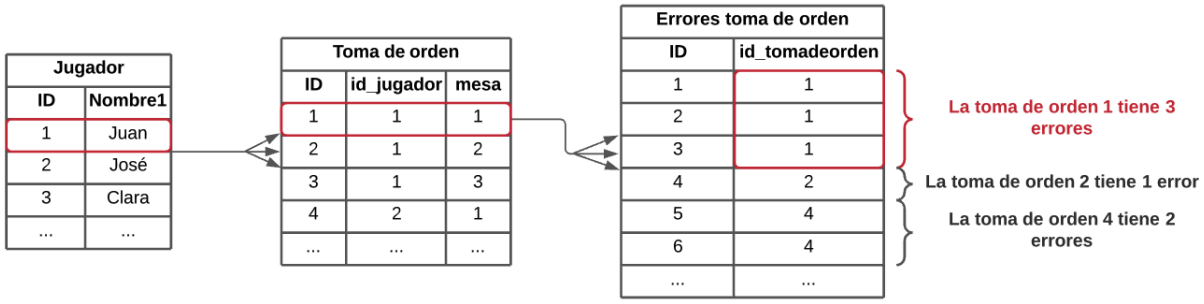


Figura 4.1: Ejemplo de la relación entre tablas y cómo se calculan las métricas. *Fuente: Elaboración propia*

Lo importante del ejemplo recién mencionado es notar que la operación que se utiliza para la construcción de la métrica es contar una cantidad de registros ligadas a una toma de orden específica y a un jugador específico, y de esta manera se pueden aplicar otras operaciones como la suma, multiplicación, máximo, mínimo, promedio entre otras, dejando fijos los jugadores y otras actividades. Por lo tanto, considerando que existen 19 tablas aparte de jugador, que es la tabla bajo la cual se quieren construir las métricas, se tiene una gran cantidad de posibles combinaciones de operaciones entre tablas y atributos de esas tablas.

La ingeniería de características automática lo que hace es efectivamente hacer todas esas posibles combinaciones que se tienen entre las distintas tablas, y calcula todas las sumas de todas las tablas de todas las variables, y todos los promedios de todas las tablas de todas las variables, y así sucesivamente con todas las operaciones disponibles y aplicables según el tipo de atributos que se tengan. Existen librerías de Python que permiten realizar este tipo de cálculos. La utilizada en el desarrollo del trabajo de título es *Feature Tools*, debido a lo simple y rápido de implementar.

Para poder implementar dicha librería, se configuran los datos de una cierta manera, para que estos estén en el formato permitido por Feature Tools. Luego se representa la base de datos en formato de la librería, quedando así 16 entidades de Feature Tools y 15 relaciones entre tablas, representando así la mayoría de las tablas del modelo de la Figura 3.6 (la mayoría debido a que hay algunas tablas que no presentan valores por lo que no se incluyen en el modelo final, como tableErrors). Ya programado el código para representar toda la base de datos en entidades de Feature Tools, con el comando *dfs* (Deep Feature Selection, o selección de características profundas, haciendo referencia a la profundidad en el recorrido que hace la librería en toda la base de datos) se hacen todas las posibles combinaciones de variables. Lo anterior resulta en una base de datos con 622 columnas, en donde 618 de esas columnas son métricas calculadas, y las otras 4 son variables asociadas a los jugadores como la edad, género, nivel educacional y el alias que se utilizó en la plataforma. En la tabla 4.1 se puede apreciar un pequeño extracto de cómo se ve la base final. En dicha tabla se puede apreciar que el nombre de las variables corresponde a la operación aplicada sobre la variable específica de una tabla, como el conteo y la moda de una cierta variable para ese jugador.

Tabla 4.1: Extracto de la base generada de forma automática con Feature Tools. *Fuente: Elaboración propia*

edad	**genero**	**educacion**	**COUNT(jugadorOpciones)**	**MODE(jugadorOpciones.opciones)**
61	Masculino	Superior completa	0.0	NaN
61	Masculino	Superior completa	2.0	1.0
25	Masculino	Superior incompleta	2.0	3.0
26	Femenino	Superior completa	2.0	5.0
24	Masculino	Superior incompleta	2.0	7.0

Si bien tener muchas variables es algo positivo ya que permite aplicar distintos tipos de técnicas para mejorar los modelos, y se pueden hacer muchas combinaciones posibles entre las variables para generar distintos modelos, el hecho de que sean variables que se generan haciendo todas las posibles operaciones de todos los atributos de las tablas del modelo de bases de datos hace que sea complicado de interpretar el significado de una variable al momento de evaluar los modelos que se realizaron. Como se puede apreciar en la tabla 4.2, hay variables que son la combinación de distintas operaciones como la suma de las desviaciones estándar del tiempo que hay entre las campanas en la entrega de los pedidos, lo cual no es sencillo de interpretar desde un punto de vista médico, que finalmente es el objetivo del presente trabajo de título, detectar que variables permiten representar el desempeño de las funciones ejecutivas. A pesar de lo anterior, se utilizó esta base para comprobar si con los datos que se generan con la plataforma se puede caracterizar el desempeño de las funciones ejecutivas y permite predecir el segmento etario de los jugadores.

Tabla 4.2: Ejemplo de variable generada con Feature Tools. *Fuente: Elaboración propia*

SUM(jugadorDelivery.deliveryOrder.STD(ringsTime.time))
0.000000
1.307871
14.139719
14.137960
7.069493

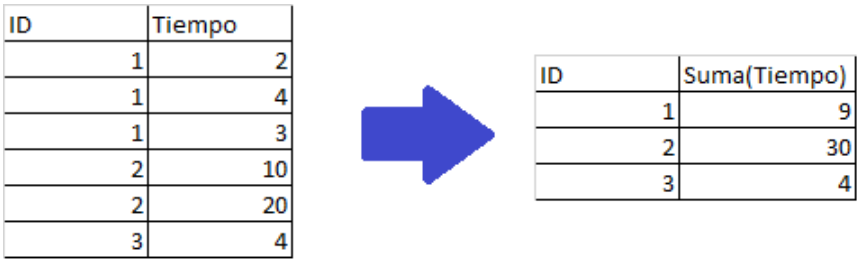
4.1.2. Ingeniería de características manual

El equipo de expertos del proyecto Neuronat construyó un conjunto de métricas que deberían ser evaluadas y generadas con los datos que exporta la plataforma Neuronat. Este conjunto de métricas consta de 310 variables aproximadamente, y consiste en variables temporales, en donde se mide el tiempo que se demora un jugador en realizar una actividad, numéricas cómo la cantidad de errores a la hora de realizar una actividad o el número de intentos, categóricas para representar el instante en el que se resuelve una cierta actividad o problemática (si se resuelve en primera instancia, segunda instancia o tercera instancia) y binarias para representar si algo se hizo o no como por ejemplo el saludo al iniciar la atención de una mesa (el jugador se presenta ante los clientes o directamente les toma la orden u otra cosa). También se consideran los 5 clientes que tiene el restorán de la plataforma y todos los hitos de cada mesa (toma de orden, ir a la cocina a buscar los pedidos, entregar los pedidos a los clientes, realizar la transacción final).

Por lo tanto, para crear la base manual se generan las métricas definidas por los expertos utilizando los datos generados por la plataforma. Afortunadamente, los pasos de preprocesamiento de los datos son los mismos que los primeros pasos en la creación de la base automática, por lo tanto se utiliza lo ya creado con anterioridad.

Con las tablas de la base de datos ya importadas, se construyen las distintas métricas. En este punto es importante destacar ciertos puntos. Lo primero, es que debido a la construcción de la estructura de la plataforma, los datos del segundo cliente del videojuego no se están exportando de la plataforma, por lo que no se cuenta con estos. Lo segundo, es que la estructura de las métricas construidas por los expertos es, primero por cliente, y luego por hito del cliente, es decir, las primeras variables son sobre el primer cliente, y en particular en el orden de toma de orden, ir a la cocina a buscar los pedidos, entregar los pedidos a los clientes y hacer la transacción final (por lo menos en un cliente típico, que no es el caso de los clientes 1, 2 y 5), luego vienen las variables del cliente 3 (ya que no se cuentan con los datos de la clienta 2) en el orden mencionado, luego cliente 4 y 5. En tercer y último lugar, el equipo clínico definió todas las métricas posibles, lo que implica que hay varias de estas que son redundantes, o se construyen en base a otras variables (lo que las hace variables dependientes y altamente correlacionadas entre esas variables). Lo anterior implica que en la base construida se tengan menos variables que las 310 construidas por los expertos. Además, se sigue el orden impuesto originalmente en donde se ordena desde el primer cliente hasta el último y por cada cliente se ordenan las variables por hitos cronológicos.

Habiendo aclarado lo anterior se procede a explicar la construcción de las métricas. La librería Pandas de Python tiene varias funciones programadas que se pueden utilizar, llamadas métodos. Debido a que dicha librería se centra en el trabajo con tablas, los métodos están relacionados igualmente al trabajo de tablas (generalmente), y en particular para la construcción de la base manual interesan 2 métodos: Merge y Group By. La función Merge se utiliza para unir 2 tablas en base a ciertos criterios, es decir, permite fusionar 2 tablas utilizando una variable en común entre ambas tablas (llaves). Por otro lado, la función Group By permite realizar operaciones matemáticas como la suma, promedio, máximo o mínimo sobre las variables de una tabla en base a una variable en particular. En la Figura 4.2 se muestra un ejemplo de cómo se vería la agrupación de una tabla en base de la variable ID. En ese ejemplo, la tabla agrupada (derecha) muestra los valores únicos de la variable ID (variable que fue agrupada), y debido a que en la otra variable Tiempo se tenían más valores, se aplica la operación Suma a esta variable, quedando la nueva variable Suma(Tiempo). Una manera de interpretar esto es que para un único valor de la ID, por ejemplo 1, la suma de sus tiempos es 9 (2+4+3).



ID	Tiempo
1	2
1	4
1	3
2	10
2	20
3	4

ID	Suma(Tiempo)
1	9
2	30
3	4

Figura 4.2: Ejemplo de cómo funciona el método Group By. *Fuente: Elaboración propia*

Por lo tanto, siguiendo la estructura de las métricas construidas por el equipo clínico, se construyen primero las métricas de la toma de orden del cliente 1. Es importante notar que existen 4 niveles en las tablas que almacenan los datos asociados a la toma de orden, los cuales se pueden ver reflejados de forma simplificada en la figura 4.1, salvo el segundo nivel que se omite por simplicidad. En primer lugar está la tabla *Jugador*, que tiene la información sobre los jugadores como la edad y el género; luego está la tabla intermedia entre *Jugador* y *Toma de Orden*, cuya función es hacer la relación entre que registros de toma de orden están asociadas a que jugador (por ejemplo un jugador tiene 4 registros de toma de orden, entonces es en esa tabla intermedia en donde se indica que jugador tiene que registros de toma de orden); después está la tabla de toma de orden en donde se registran todos los datos asociados a la toma de orden, como el momento en que se comienza a realizar esa actividad o si es que el jugador se presentó o no; y finalmente están las 2 tablas que registran los errores asociados a la toma de orden, que indican que errores obtenidos pertenecen a que toma de orden en particular. Por lo tanto, para construir las métricas asociadas a la toma de orden se debe interactuar con estos 4 niveles, y para hacer esta interacción se utilizan los métodos Merge y Group By. En la misma figura 4.1, en la llave que aparece en rojo a la derecha se puede apreciar el valor de agrupar las variables según el jugador y la toma de orden, dando como resultado 3 errores en la toma de orden de la mesa 1, que se relaciona con el jugador 1.

Lo primero que se hizo fue obtener las métricas asociadas a las tablas del último nivel,

es decir, del nivel de los errores en la toma de orden, y luego se asociaron esas métricas a la toma de orden correspondiente. Por ejemplo, la primera métrica calculada fue "el primer error obtenido en el menú general", la cual hace referencia a la primera opción que elige el jugador en el menú general, que no sea la toma de orden. Para construir esta métrica se hizo una agrupación sobre la variable *Order* de la tabla *menuErrors*, luego, considerando que existe una variable temporal que indica en que momento sucedió cada registro, se obtuvo el mínimo de los tiempos para cada agrupación, para así quedar con el registro que sucedió primero para cada orden. El resultado de esta operación es una tabla en donde cada registro es una orden, y cada orden tiene asociado un elemento del menú general, que corresponde a la primera opción seleccionada por el jugador que no sea la toma de orden. En el código B.6 de anexos se puede apreciar el código de cómo se construye dicha métrica, mientras que en la tabla 4.3 se muestra el resultado de la agrupación de la tabla *menuErrors*.

Tabla 4.3: Tabla generada en la construcción de la métrica utilizando el método Group By. *Fuente: Elaboración propia*

order	**type**	**time**	**pk**
5	candy	20.168570	1
8	candy	192.805527	2
11	candy	477.734222	6
12	candy	71.314720	8
15	tip	306.682251	9
22	deliver_order	287.056122	10
25	candy	351.924927	11

En el código B.6, aparece también como se une la tabla 4.3 con la tabla *takeOrder1* utilizando el método Merge. Como se mencionó anteriormente, el método Merge fusiona 2 tablas utilizando una variable en común entre ambas, que en este caso corresponde a *order*. En la tabla *takeOrder1* se tienen todos los registros de toma de orden del cliente 1, por lo tanto, cómo sólo se toma el pedido al cliente 1 una vez en el juego, cada registro de *takeOrder1* corresponde a un jugador distinto. Además, la tabla toma de orden tiene una variable que permite identificar únicamente a sus registros llamada *pk*, por lo que no puede haber 2 registros con el mismo pk. En la tabla 4.3 se puede apreciar que la primera columna es order, y los valores que aparecen ahí corresponden a la variable *pk* de la tabla *takeOrder1*, es decir, que la fila cuyo valor *order* es 5, hace referencia a la toma de orden cuyo valor *pk* es 5. Por lo tanto, el resultado de unir ambas tablas es una nueva tabla que contiene todas las columnas de *takeOrder1* con todas las columnas de la tabla 4.3. Existe la posibilidad de que al hacer esa combinación de ambas tablas suceda que haya más registros en *takeOrder1* que en la tabla 4.3, en cuyo caso, los registros que no consigan un par de la otra tabla se rellenan con valores nulos. Por ejemplo, si en la tabla *takeOrder1*, el primer registro cuyo *pk* es 1 no tiene ningún error de menú en el juego, implicará que en la tabla 4.3 no haya ningún registro con valor 1 en la variable *order*, por lo que al hacer la mezcla de ambas tablas, la toma de orden con *pk* 1 no tendrá ningún valor asociado a las columnas provenientes de la otra tabla como *type* y *time*. El resultado de esa tabla se puede apreciar en la tabla 4.4.

Tabla 4.4: Tabla generada al unir *takeOrder1* con la tabla construida anteriormente. *Fuente: Elaboración propia*

	greet	table	state	pk_x	greet_boolean1	type	time	pk_y
0	False	1	B001	1	0	NaN	NaN	NaN
4	True	1	B001	5	1	candy	20.168570	1.0
11	True	1	B001	12	1	candy	71.314720	8.0
18	True	1	B001	19	1	NaN	NaN	NaN
25	True	1	B001	26	1	deliver_order	5.746121	12.0
32	True	1	B001	33	1	NaN	NaN	NaN
38	True	1	B001	39	1	NaN	NaN	NaN
44	False	1	B001	45	0	NaN	NaN	NaN
50	True	1	B001	51	1	NaN	NaN	NaN
56	False	1	B001	57	0	NaN	NaN	NaN

Finalmente, se reemplazaron los valores nulos que se ven en la tabla 4.4, por el valor "take_order", debido a que si aparece un valor nulo en la columna *type*, quiere decir que el jugador seleccionó correctamente la opción "Toma de orden" en el menú general del juego. Ya realizado ese proceso, se procedió a eliminar las variables no relevantes de dicha tabla como *time*, *pk_y*, *State* y *table*, y se transformaron las variables de texto como *greet* y *type* a variables numéricas, binarias o categóricas según corresponda, quedando así una tabla sólo con valores importantes y numéricos para posteriormente poder analizarlo. Se realizó el mismo proceso para construir todas las métricas necesarias, utilizando el método Group By y luego indexando las métricas construidas a la tabla *takeOrder1*. Ya construida la tabla con todas las métricas se obtiene una tabla como 4.5, y el siguiente paso fue relacionar esa tabla con la tabla *Jugador*, para así asociar las variables demográficas a las métricas calculadas.

Tabla 4.5: Extracto de la tabla final con todas las métricas construidas. *Fuente: Elaboración propia*

	table	takeOrderID	greet_boolean1	type_label1	take_order_boolean1	MenuErrorCount1	persevMenu1	takeOrderErrorCount1	persevTakeOrder1
0	1	1	0	2	1	0.0	0	1.0	0
1	1	5	1	0	0	1.0	0	0.0	0
2	1	12	1	0	0	1.0	0	0.0	0
3	1	19	1	2	1	0.0	0	0.0	0
4	1	26	1	1	0	2.0	0	0.0	0
5	1	33	1	2	1	0.0	0	0.0	0
6	1	39	1	2	1	0.0	0	0.0	0
7	1	45	0	2	1	0.0	0	0.0	0
8	1	51	1	2	1	0.0	0	0.0	0
9	1	57	0	2	1	0.0	0	0.0	0

Se siguió el mismo procedimiento para todas las métricas de los hitos (toma de orden, ir a la cocina, entregar los pedidos y la transacción final) y de los clientes, obteniendo así una tabla final con 72 columnas, de las cuales 68 son métricas construidas de las que fueron definidas por el equipo clínico.

Como se comentó al inicio de esta sección, en un inicio se definieron alrededor de 310 métricas, sin embargo se pudieron generar 68 de estas, debido a 2 razones: las métricas del segundo cliente no se exportan de la plataforma y existen variables redundantes. La mesa de la clienta 2 cuenta con 102 variables, de las cuales ninguna pudo ser calculada por lo recién mencionado, mientras que en la mesa del cliente 1 se tienen 27 variables de las cuales se calcularon 12, en la mesa del cliente 5 igualmente se tienen 27 variables y se calcularon

12 (mesas idénticas en donde sólo cambia la expresión del cliente), en la mesa del cliente 3 se tienen 76 variables de las cuales se calcularon 22 y en la mesa del cliente 4 se tienen 76 variables igualmente pero se calcularon 22 (mismo caso que entre la mesa 1 y 5, son casi idénticas pero cambiar aspectos de forma en estos clientes). En las tablas 4.3, 4.4 y 4.5 se puede apreciar el proceso de construcción de las métricas calculadas y cómo se ven estas. En el anexo C.1 se puede apreciar el glosario de todas las variables calculadas con su significado y sus posibles valores.

4.2. Selección de características

El objetivo general del trabajo de título es poder determinar que variables permiten caracterizar de mejor manera el desempeño de las funciones ejecutivas, y es por esto que se proponen distintos conjuntos de variables, para posteriormente evaluar que combinación de variables entrega un mejor resultado. Con las 2 tablas que se utilizan como base, ya formadas, se procede a realizar el proceso de construcción de distintos conjuntos de variables.

Dentro del proceso KDD, en el preprocesamiento de los datos se suelen implementar técnicas de *Feature Selection* (selección de características o variables en español), en donde se seleccionan las variables más relevantes de un conjunto inicial. Cómo se mencionó en el capítulo 1, en la sección de Marco Conceptual, existen 4 métodos de Selección de Características, filtros, wrappers, mixtos y embedded. A modo de resumen, los filtros aplican ciertos criterios a la base para filtrar variables, como por ejemplo según la varianza. Los métodos Wrapper van agregando o eliminando variables desde un conjunto inicial de variables, y en cada iteración comprueban que tan bien se entrena un modelo según las variables de esa iteración. Los métodos mixtos combinan los filtros y los wrapper, entregando un conjunto ya filtrado al algoritmo wrapper. Finalmente, en los métodos embedded es el mismo modelo de Machine Learning el que elige las variables según criterios propios del modelo. En el presente estudio se implementan los métodos filtros y wrapper (mixto) para generar distintos conjuntos de datos.

4.2.1. Métodos de filtro

Cómo ya se ha mencionado antes, los métodos de filtros permiten como dice su nombre filtrar variables de acuerdo con un cierto criterio. Para el caso del presente trabajo se utilizan 4 criterios distintos: Varianza cero y duplicados, correlación, información mutua y la combinación entre correlación e información mutua.

Varianza cero y variables duplicadas

La varianza indica que tan dispersos están los datos a lo largo de los distintos registros, es decir, que tanto varían. Una alta varianza indica que puede haber una gran diferencia entre los valores de una variable, mientras que una baja varianza puede indicar que los valores de esa variable no varían tanto, o en el caso extremo, son fijos. Por lo tanto, cuando la varianza de una variable toma el valor cero, indica que los valores de esa variable no cambian a lo largo de los registros, lo que quiere decir que esa variable no tiene ninguna relevancia para poder determinar el desempeño de una función ejecutiva (si esa variable es constante no indica ningún cambio entre las distintas poblaciones que hay en la muestra).

Por otro lado, se puede dar el caso en que haya 2 variables idénticas, es decir, que para todas las filas esas 2 variables tengan los mismos valores. Nuevamente esto implica que una de esas 2 variables está de más y no aporta en nada para el posterior análisis.

Por lo tanto, los primeros filtros que se aplicaron fueron los recién mencionados, en donde se eliminan todas aquellas variables que tengan una varianza menor al 2 %, y todas aquellas variables que tengan un par (o más) repetidos, es decir, se deja una de todas las repetidas. No existe un valor por excelencia para poner como umbral en el filtro de varianza, y se suele recomendar valores bajos según el dataset que se tenga. En este caso, se eligió un 2 % debido a que permite filtrar más variables que poniendo una restricción más estricta de varianza cero, que es el caso por defecto en la función *VarianceThreshold* de Scikit-Learn, y sigue siendo un valor lo suficientemente bajo como para considerar dichas variables como constantes.

Antes de implementar ambos filtros, lo primero que se hizo fue segmentar las 2 bases que se tienen en muestras de entrenamiento y muestras de evaluación (más conocidos por sus nombres en inglés, train y test). Como recordatorio, es importante realizar dicha separación ya que es necesario evaluar los modelos que se utilizan posteriormente, y en particular, interesa que el modelo pueda predecir fuera de la muestra de la que aprendió, es decir, que pueda generalizar. Es por esto que existe una muestra de entrenamiento, la cual se utiliza para que el modelo aprenda y detecte patrones según los datos que se tienen, y una muestra de evaluación en donde el modelo se enfrenta a datos que nunca ha "visto" y tiene que predecir según lo que "aprendió". Luego, debido a que se implementan modelos de Machine Learning Supervisado (se tienen etiquetas), se puede comparar el valor real con el valor predicho, para determinar que tan bien predice el modelo entrenado.

Con ambas bases ya segmentadas en las muestras de entrenamiento y evaluación, se procedió a implementar los filtros de Varianza y de variables duplicadas. Finalmente es importante destacar que las bases resultantes al aplicar estos filtros se utilizan como base para los próximos filtros a implementar, es decir, ya no se utilizan las bases originales con 72 y 622 variables, sino que las ya filtradas según varianza y variables duplicadas.

Correlación

Las correlaciones indican que tan fuerte es la relación entre 2 variables. Un ejemplo de correlación aplicado al contexto del trabajo realizado es el de tiempo de ejecución del videojuego y presencia de Disfunción Ejecutiva. Digamos que se tienen 2 variables, una que indica el tiempo que se demora un jugador y otra que indica la probabilidad de padecer Disfunción Ejecutiva (una variable real que adopta valores entre 0 y 1). Si se analiza la evolución de la variable temporal en función de la probabilidad de padecer una DE, se debería apreciar que a medida que esta incrementa, la probabilidad también lo hace. Por lo tanto, debido a que ambas variables tienen comportamientos similares bajo ciertos criterios, se puede decir que ambas están altamente correlacionadas. Hay que destacar que el hecho de que 2 variables estén relacionadas, no quiere decir que existe causalidad, es decir, el hecho de que una persona se demore en resolver el videojuego no quiere decir que tenga una Disfunción Ejecutiva, y a su vez, el hecho de padecer una Disfunción Ejecutiva no necesariamente se verá reflejado en el tiempo de ejecución, simplemente son 2 fenómenos que están altamente relacionados debido a un tercer factor que es necesario indagar más al respecto para averiguar cual es.

Por otro lado, al momento de analizar datos y entrenar modelos, debido a que 2 variables altamente relacionadas tienen comportamientos similares, se considera necesaria una de estas variables, ya que con una se captura el efecto deseado, y la otra variable sería redundante. Es por esto que se filtran las variables altamente correlacionadas.

Para hacer esto se crea una función que obtiene todas las variables con un alto nivel de correlación, y este alto nivel de correlación se determina con un parámetro de la función creada. Es importante notar que los valores de correlación van desde -1 hasta 1, en donde los valores negativos indican una correlación negativa (si una variable aumenta su valor, la otra lo disminuye), y los valores positivos indican una correlación positiva (si una variable aumenta su valor, la otra también lo aumenta). Además, mientras más cercano a 1 (o -1 en el caso de una correlación negativa) mayor será la fuerza de la dependencia y relación entre las variables. Por lo tanto, la función creada primero determina una matriz de correlación utilizando un método de la librería Pandas, sobre las bases obtenidas del filtro anterior. Luego se recorre está matriz que indica las correlaciones entre 2 variables según la posición, y se almacena en un vector las variables que tengan un valor absoluto de correlación mayor al umbral definido. Finalmente, se tendrá un vector con todas las variables que presentan una alta correlación con otra variable y se eliminan estas de las bases. Cabe destacar que este paso y todos los que se verán en los siguientes apartados se realizan 2 veces, uno para la base manual y otro para la base automática.

Información mutua

La información mutua es una medida de dependencia entre 2 variables, que mientras mayor sea su valor indica dependencia entre ambas variables, y cuando es cero o cercana a cero indica independencia. Por lo tanto, se calcula la información para todas las variables independientes con respecto a la variable dependiente, es decir, la que se quiere estimar que en este caso corresponde al segmento etario de las personas.

Al calcular la información mutua entre las variables y la etiqueta se determina si las variables de la base son dependientes o independientes de lo que se quiere estimar. Finalmente se dejan las variables con mayores valores de información mutua ya que son las que presentan un mayor grado de dependencia con la variable a estimar. Es importante que se tenga ese nivel de dependencia entre las variables de la base y la variable a estimar ya que lo que se quiere es detectar las variables que tengan un efecto sobre lo que se quiere estimar, y es por esto que no sirve tener variables independientes a la etiqueta, ya que no aportan información en poder predecir un valor de esta.

Para realizar este filtro se aplican principalmente 2 funciones de la librería Scikit-Learn mencionada anteriormente, *mutual_info_classif* (asigna los valores de información mutua a todas las variables con respecto a la variable a estimar) y *SelectPercentile* (selecciona el X % de variables con mayores valores de información mutua, en donde X es un valor que se debe definir en la función). El X seleccionado en el trabajo fue de un 40 %, por un par de razones. En primer lugar el objetivo de hacer Feature Selection es reducir la dimensionalidad, por lo que desde un inicio se deseaba reducir en más de un 50 % la cantidad de variables. Luego se deseaba obtener las variables con mayor valor de Información Mutua que serían las primeras en el orden que aplica la función *SelectPercentile*. Sin embargo, posteriormente se aplicarán otros métodos de Feature Selection sobre la base obtenida en este paso, por

lo que se elige un valor alto como un 40 % para así poder tener más variables, con las que los algoritmos wrapper (explicados posteriormente) podrán hacer más combinaciones entre estas. Aplicando estas 2 funciones se obtienen 2 bases (manual y automática) con un nuevo conjunto de variables, las cuales presentan una alta dependencia con la variable a estimar.

Correlación e Información mutua

Además, se decidió hacer una combinación de 2 criterios, para así generar una base de datos con variables que tengan baja correlación entre sí y que tengan un alto valor de información mutua con la variable a estimar. De esta manera se genera un nuevo conjunto de variables.

4.2.2. Métodos Wrapper

Cómo ya se ha mencionado antes, los métodos wrapper de Feature Selection utilizan algún algoritmo o modelo de clasificación para determinar qué conjunto de variables son los que permiten predecir mejor el valor de una cierta variable. A grandes rasgos existen principalmente 2 maneras de abarcar un método wrapper, uno es el llamado *Forward* (para adelante) y el otro es el llamado *Backward* (hacia atrás). El método *Forward* lo que hace es que teniendo un conjunto inicial de variables, a medida que avanza el algoritmo va agregando más variables a este conjunto, mientras que el método *Backward* es todo lo contrario, es decir, se parte de un gran conjunto de variables y a medida que el algoritmo avanza se van eliminando variables. Por lo tanto, teniendo un conjunto inicial de datos con una cantidad determinada de variables se entrena un modelo para que pueda predecir algún valor, luego se evalúan los resultados de ese modelo y se almacena dicha combinación de variables junto al resultado del modelo en distintas variables. Ya realizado el proceso anterior se sigue con la nueva iteración del algoritmo en donde se agrega o elimina una variable dependiendo de si es *Forward* o *Backward*. Una vez terminadas todas las iteraciones, se procede a comparar las distintas combinaciones de variables con sus resultados para finalmente elegir el conjunto de variables que entrega mejores resultados de predicción.

Para la implementación de los algoritmos wrapper forward y backward se utiliza la librería *mlxtend*, la cual tiene la función llamada *SequentialFeatureSelector* (SFS) que dependiendo de los parámetros que se le introduzcan realiza el proceso de Feature Selection utilizando el método forward o backward. En este punto es importante mencionar que para que funcione la función SFS, se debe incluir dentro de los parámetros de ésta un modelo de Machine Learning, razón por la cuál en el desarrollo del código implementado se definen los modelos a utilizar en este apartado, sin embargo la generación de modelos se explicará en un próximo apartado de este capítulo. Lo que sí es importante a destacar en este punto es que se utilizaron 3 modelos distintos, Support Vector Machine (SVM), Random Forest (RF) y Naive Bayes (NB).

En primer lugar se utiliza la base manual y se implementa el método Forward utilizando el modelo de SVM, y los 4 conjuntos de variables obtenidos al aplicar los filtros que fueron explicados en la sección anterior (varianza y duplicados, correlación, información mutua y correlación con información mutua). Luego se implementa el método Backward con el mismo modelo de SVM y los mismos 4 conjuntos de variables. Una vez terminado el proceso de Feature Selection para un modelo, se procede a implementar el mismo proceso pero con otro

modelo, en este caso RF, y ya finalizado ese se implementa con NB. De esta manera, se tendrían 8 nuevos conjuntos por modelo implementado, quedando así 24 conjuntos de datos nuevos. Finalmente, se realiza el mismo procedimiento para la base automática, quedando así 48 conjuntos de datos nuevos.

Por lo tanto, si se consideran los conjuntos de variables obtenidos en la primera sección de filtros, se tendrían en total 56 conjuntos de variables distintos (los 48 obtenidos de los wrapper, más 4 filtros para la base manual y 4 filtros para la base automática). Por otro lado, considerando que se tienen 3 modelos de Machine Learning que serán implementados, la cantidad de conjuntos aumenta aún más, sin embargo, no es tan directo como multiplicar 56 por 3, ya que los conjuntos de los wrapper están directamente ligados a un modelo en particular, es decir, el conjunto de variables obtenidos para SVM no se aplican para RF, ya que ese conjunto de variables está optimizado para SVM y no para RF. Por lo tanto, considerando ambas bases (manual y automática), todos los conjuntos de variables obtenidos, y los 3 algoritmos de Machine Learning que se implementan, se entrenaron en total 72 modelos con distintas combinaciones de variables y algoritmos. En la tabla 4.6 se resumen todos los modelos que se implementan, con el correspondiente algoritmo, base y conjunto de variables.

Tabla 4.6: Tabla resumen de todos los modelos implementados

Manual			Automática		
SVM	RF	NB	SVM	RF	NB
Var	Var	Var	Var	Var	Var
Corr	Corr	Corr	Corr	Corr	Corr
IM	IM	IM	IM	IM	IM
IMCrr	IMCrr	IMCrr	IMCrr	IMCrr	IMCrr
$F_{svm}(Var)$	$F_{rf}(Var)$	$F_{nb}(Var)$	$F_{svm}(Var)$	$F_{rf}(Var)$	$F_{nb}(Var)$
$F_{svm}(Corr)$	$F_{rf}(Corr)$	$F_{nb}(Corr)$	$F_{svm}(Corr)$	$F_{rf}(Corr)$	$F_{nb}(Corr)$
$F_{svm}(IM)$	$F_{rf}(IM)$	$F_{nb}(IM)$	$F_{svm}(IM)$	$F_{rf}(IM)$	$F_{nb}(IM)$
$F_{svm}(IMCrr)$	$F_{rf}(IMCrr)$	$F_{nb}(IMCrr)$	$F_{svm}(IMCrr)$	$F_{rf}(IMCrr)$	$F_{nb}(IMCrr)$
$B_{svm}(Var)$	$B_{rf}(Var)$	$B_{nb}(Var)$	$B_{svm}(Var)$	$B_{rf}(Var)$	$B_{nb}(Var)$
$B_{svm}(Corr)$	$B_{rf}(Corr)$	$B_{nb}(Corr)$	$B_{svm}(Corr)$	$B_{rf}(Corr)$	$B_{nb}(Corr)$
$B_{svm}(IM)$	$B_{rf}(IM)$	$B_{nb}(IM)$	$B_{svm}(IM)$	$B_{rf}(IM)$	$B_{nb}(IM)$
$B_{svm}(IMCrr)$	$B_{rf}(IMCrr)$	$B_{nb}(IMCrr)$	$B_{svm}(IMCrr)$	$B_{rf}(IMCrr)$	$B_{nb}(IMCrr)$

Var: Modelo con el conjunto de variables resultantes de aplicar el filtro de varianza y de variables duplicadas.

Corr: Modelo con el conjunto de variables resultantes de aplicar el filtro de correlación.

IM: Modelo con el conjunto de variables resultantes de aplicar el filtro de información mutua.

IMCrr: Modelo con el conjunto de variables resultantes de aplicar el filtro de la combinación entre correlación e información mutua.

F_X(Y): Modelo con el conjunto de variables resultantes de aplicar el método Forward con el algoritmo X y con el filtro Y de entrada al método Forward.

B_X(Y): Modelo con el conjunto de variables resultantes de aplicar el método Backward con el algoritmo X y con el filtro Y de entrada al método Backward.

4.3. Modelos de clasificación

Ya terminada la construcción de los distintos conjuntos de variables se procede a entrenar todos los modelos presentados en la tabla 4.6. Cómo ya se sabe, se implementan 3 algoritmos de clasificación distintos: *Support Vector Machine* (SVM), *Random Forest* (RF) y *Naive Bayes* (NB). La librería de Python anteriormente mencionada, Scikit-Learn, tiene funciones asociadas a cada uno de esos algoritmos, por lo tanto se utilizan estas funciones para la construcción de los modelos.

A modo general, se crea una función para cada algoritmo a implementar, en donde cada función recibe como parámetros 4 conjuntos de variables: X_train, X_test, y_train, y_test, en donde los X corresponden a los conjuntos de variables y los y corresponden a la variable a predecir; por otro lado, los *train* corresponden a los conjuntos que se utilizan para entrenar al modelo, por eso existe un X y un y (variables independientes y variable dependiente, respectivamente), y los *test* son los conjuntos que se utilizan para evaluar los modelos entrenados. Por lo tanto, se reciben como parámetros los conjuntos de variables obtenidos en el paso anterior. Luego, se importan las funciones de Scikit-Learn para cada algoritmo a utilizar en cada función, lo que quiere decir que hay una función para SVM, otra función para RF y otra función para NB.

Antes de poder entrenar un modelo, es necesario definirlo, lo que se hace con las funciones que se importan de Scikit-Learn. Cada algoritmo cuenta con una serie de parámetros para personalizarlos y ajustarlos de mejor manera según el tipo de datos que se tienen y según el contexto del problema con el que se está trabajando. Existen versiones simples de cada algoritmo en donde todos los parámetros se dejan en sus valores por defectos, pero los modelos entrenados con esa combinación de parámetros por defectos suelen no presentar un gran rendimiento. Es por esto que en las funciones programadas (para SVM y Random Forest) se implementan 2 métodos de optimización de parámetros, los cuáles prueban un algoritmo con distintas combinaciones de parámetros y el resultado final es el modelo con la combinación de parámetros que entregó mejores resultados. Dichos métodos de optimización de parámetros son *RandomSearch* y *GridSearch*.

RandomSearch realiza distintas combinaciones de parámetros de forma aleatoria, mientras que *GridSearch* realiza todas las posibles combinaciones de parámetros que se tienen, obteniendo un mejor resultado pero con un costo computacional mayor. Por ejemplo, digamos que un modelo cuenta con 4 parámetros, y se le asignan 4 valores a cada parámetro; *RandomSearch* hará una cantidad acotada de iteraciones, en donde cada iteración es una combinación única y aleatoria de esos 4 parámetros, mientras que *GridSearch* realiza todas las combinaciones de esos 4 parámetros con esos 4 posibles valores por parámetro.

Por lo tanto, el procedimiento realizado para las funciones de SVM y Random Forest fue, crear posibles valores para los distintos parámetros del algoritmo asociado a esa función, luego realizar el proceso de *RandomSearch* ya que es más rápido y permite hacer un sondeo inicial. Los resultados de *RandomSearch* indican que combinación de parámetros entregan el mejor resultado, por lo tanto se utilizan esos valores para volver a definir posibles valores para los distintos parámetros de ese modelo. Una vez lista la definición de los parámetros, se procede a ejecutar *GridSearch*, para así poder probar todas las posibles combinaciones de

valores, pero partiendo de la base de que se utilizan los valores arrojados por *RandomSearch*, optimizando así la búsqueda de *GridSearch* ya que no tendrá que probar combinaciones que dan malos resultados. Finalmente, el resultado de la función desarrollada es el modelo que entregó los mejores resultados en *GridSearch*.

Finalmente, es importante recalcar que ambos métodos de optimización de parámetros cuentan con un campo que es *Cross Validation*, que como ya se vio en el Marco conceptual, es una técnica que se utiliza para mejorar el desempeño de un modelo y disminuir el sobreajuste, mejorando así la generalización del modelo. Dentro de las funciones de *cross validation* que existen en Scikit-Learn se pueden encontrar 2 grandes categorías, regular y estratificado, siendo el regular recomendado para problemas de regresión en donde no existen clases, ya que hace una asignación completamente aleatoria para cada muestra, mientras que el estratificado se recomienda para problemas con clases como el presente en este trabajo de título, ya que realiza una asignación aleatoria pero considerando las clases. En la Figura 4.3 se puede ver la diferencia entre *cross validation* regular y estratificado.

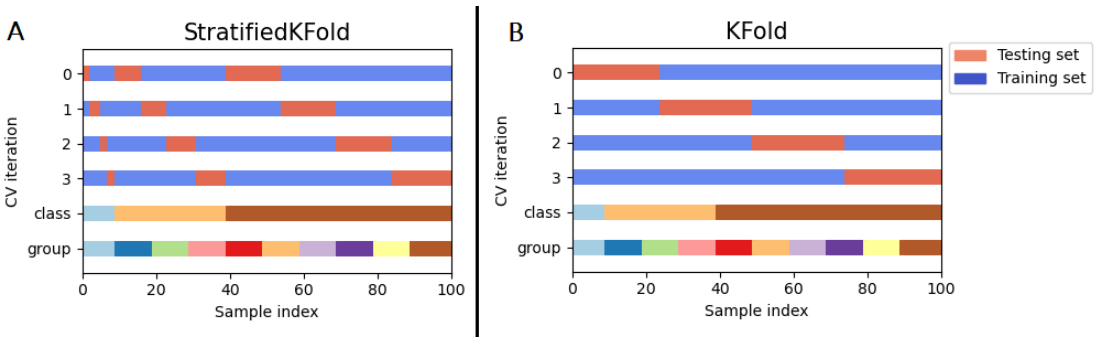


Figura 4.3: (**a**) Ejemplo de *Cross validation* estratificado, en donde las muestras asignadas a Test consideran la distribución de las clases; (**b**) Ejemplo no estratificado o regular de *Cross Validation*, en donde se asigna aleatoriamente un segmento al conjunto de Test, sin considerar las clases. *Fuente: Scikit-Learn* [82]

Por lo tanto, para *RandomSearch* se utiliza la versión repetida de Kfold estratificado, la cual repite n veces el proceso de *Cross validation* y cada repetición se asigna aleatoriamente los conjuntos de test y train. Para *GridSearch* se utiliza la versión estratificada de *ShuffleSplit*, que es un método de realizar la validación cruzada en donde se revuelven los datos (shuffle en inglés) y una vez revuelto se asigna de forma aleatoria (considerando las clases) los registros a test y train.

Support Vector Machine

Para no tener que definir manualmente los modelos (con su respectiva optimización de parámetros) para cada conjunto de variables se creó una función llamada *run_svm(X_train, X_test, y_train, y_test)*, la cuál de forma genérica define un modelo de Support Vector Machine, optimiza los parámetros primero mediante *RandomSearch*, luego mediante *GridSearch* y finalmente devuelve el modelo con la combinación de parámetros que entrega los mejores resultados según el criterio de accuracy, a pesar de estar dentro de un contexto médico en donde se suele priorizar el recall. Se elige accuracy por sobre recall debido a que no se está

trabajando con una muestra de pacientes reales, y el objetivo de los modelos implementados no es diagnosticar una enfermedad o condición como si es en el caso del proyecto Neuronat, sino más bien es poder detectar las variables que permiten caracterizar mejor el desempeño de las funciones ejecutivas, por lo tanto interesa que pueda identificar y diferenciar de la mejor manera posible a modo general, más que para una clase en particular como es en el caso del recall. En el código B.5 del anexo se puede apreciar la función creada. Cabe destacar que los valores iniciales de los parámetros fueron creados buscando un amplio rango de posibles valores para cada parámetro.

Random Forest

Al igual que en el caso de SVM, también se construye una función para el algoritmo de Random Forest, llamada *run_randomForest(X_train, X_test, y_train, y_test)*. Esta función cumple el mismo propósito de generalizar la creación de modelos con parámetros optimizados, para no tener que definir manualmente cada modelo para todos los conjuntos de variables que se tienen. El resultado de esta función es el modelo que mejor resultados entregó según su combinación de parámetros utilizando el mismo criterio de accuracy explicado en SVM.

Naive Bayes

El algoritmo de Naive Bayes a diferencia de Random Forest y Support Vector Machine no cuenta con tantos parámetros para su configuración, dependiendo de la versión de Naive Bayes implementada. La librería Scikit-Learn dispone de 5 implementaciones distintas: *GaussianNB*, *BernoulliNB*, *CategoricalNB*, *MultinomialNB* y *ComplementNB*. BernoulliNB realiza el proceso de entrenamiento y clasificación asumiendo que las variables entregadas al modelo son binarias, y en caso de no serlo las transforma en variables binarias, sin embargo, las bases utilizadas presentan distintos tipos de datos, no sólo datos binarios por lo que se descarta dicho algoritmo. En contraste a BernoulliNB, CategoricalNB realiza el proceso asumiendo que todas las variables son categóricas, es decir, que se tiene una cantidad predefinida de posibles valores para cada variable, lo cual tampoco se tiene en las bases, por lo que también se descarta esta implementación. ComplementNB es una variante de MultinomialNB que suele presentar mejores resultados por lo que se descarta MultinomialNB. De cualquier manera ambas implementaciones suelen presentar buenos resultados en la tarea de clasificación de textos que no es lo que se está haciendo en el presente trabajo. Finalmente GaussianNB asume una distribución normal de las variables, lo que quiere decir que acepta variables que tengan un rango de valores, lo cual se aproxima más que las otras implementaciones.

Dado que no se tiene una optimización de parámetros, cada modelo se entrena muy rápido, por lo que, se crea una función llamada *run_nb(X_train, X_test, y_train, y_test)*, que toma los conjuntos de variables que se tienen y entrena 4 de los 5 modelos antes mencionados. No se considera el modelo de CategoricalNB debido a que este arrojó varios errores de configuración que no pudieron ser resueltos. El resultado de la función run_nb() es el mejor modelo de los 4 entrenados, y para seleccionar el mejor se compara en primer lugar los accuracy del conjunto de test, y se elige el modelo que tenga el mayor valor para esta métrica. En el caso de que haya más de un modelo con el mismo valor máximo, se procede a analizar el accuracy del conjunto de entrenamiento, para determinar cuál es el mejor modelo. Si aun así los valores son iguales, se tienen 2 modelos idénticos, por lo que se elige cualquiera de los 2.

Por lo tanto, con las funciones creadas y el proceso de optimización, se entrenaron todos los modelos de la tabla 4.6, utilizando la función que corresponda según el caso. En el próximo capítulo se verán los resultados de todos los modelos y se evaluará que combinación de variables en conjunto con que algoritmo entrenan al modelo que entregue mejores resultados.

Capítulo 5

Resultados, evaluación y discusión

En este capítulo se presentan los resultados de todos los modelos implementados en la tabla 4.6, para posteriormente poder evaluar según distintos criterios cual de todos los modelos es el que permite clasificar de mejor manera el segmento etario. Se mostrarán los resultados desde el punto de vista de la base manual y de la base automática, ya que los resultados de ambas bases tienen distintas interpretaciones. Finalmente se realiza una evaluación de impacto social y económica, para poder estimar a cuantas personas puede afectarle positivamente los resultados de la investigación, y en que monto monetario aproximado.

5.1. Evaluación de modelos

Debido a que se tenían bastantes conjuntos de variables, modelos y bases distintas, se decidió separar el trabajo realizado en 2 apartados, uno para cada base (manual y automática), lo que permite además tener distintas interpretaciones, ya que la base automática al tener tantas variables (al ser combinaciones de operaciones matemáticas de los datos exportados por el videojuego), es difícil interpretar el aporte de cada variable sobre la decisión tomada por los modelos, pero sí permite poder representar de mejor manera el fenómeno que se desea caracterizar que en este caso es el desempeño de funciones ejecutivas, por lo tanto, se puede obtener modelos con mejores resultados; mientras que en el caso de la base manual al tener menos variables, pero con significado clínico (ya que se construyeron dichas variables en conjunto con el equipo clínico) se tienen modelos con peor desempeño que en el caso de la base automática, pero las variables asociadas a cada modelo si pueden ser interpretadas. A pesar de que en la base manual se tenga un peor desempeño que en la base automática, si se logran detectar patrones de los datos, ya que se predice con resultados mayores al azar, lo cual será presentado en los próximos apartados.

5.1.1. Base manual

A modo de recuerdo, la base manual es aquella base que se construyó utilizando las métricas que se desarrollaron desde el equipo clínico en base a los datos que se exportan desde la plataforma Neuronat. Dicha base consta de alrededor de 70 variables las cuales tienen una interpretación específica como por ejemplo “la cantidad de errores al tomar la orden de la mesa 1”.

Utilizando esta base se construyeron 36 modelos distintos, asociados a 3 algoritmos distintos, y 12 conjuntos de variables, en donde esos 12 conjuntos de variables se crearon a partir de 4 filtros realizados, sometidos a 2 algoritmos wrapper (forward y backward), quedando así los 4 filtros solos, 4 forward y 4 backward (12 conjuntos de variables). En primer lugar se revisarán los resultados asociados al algoritmo de Support Vector Machine, luego Random Forest y finalmente Naive Bayes.

Support Vector Machine

Como ya se dijo, se tienen 12 modelos SVM asociados a 12 conjuntos de variables obtenidos a través del proceso de Feature Selection. Los 4 conjuntos obtenidos de los filtros son: *Unique*, *Uncorr*, *Mi* y *MiCorr*, en donde el primero hace referencia a la base sin variables con valores constantes ni duplicadas, el segundo a la base sin lo anterior y sin variables correlacionadas, el tercero a la base según el filtro de información mutua, y el cuarto a la base que se obtiene al combinar los filtros de correlación y de información mutua. Utilizando esos 4 conjuntos se obtuvieron 4 nuevos conjuntos al implementar el algoritmo Forward de Feature Selection y otros 4 conjuntos al implementar el algoritmo Backward de Feature Selection, obteniendo así los otros 8 conjuntos: *Forward Unique*, *Forward Uncorr*, *Forward Mi*, *Forward MiCorr*, *Backward Unique*, *Backward Uncorr*, *Backward Mi* y *Backward MiCorr*.

En las tablas 5.1 y 5.2 se pueden apreciar las distintas métricas de evaluación para los modelos entrenados con los 12 conjuntos de datos. Se separan en 2 tablas ya que hay métricas asociadas al modelo en general y métricas ponderadas asociadas al poder predictivo por clase de los modelos. Además, se muestran las métricas para los conjuntos de entrenamiento y test. En las tablas C.2 y C.3 de la sección de anexos se puede apreciar el desglose del poder predictivo de los modelos por clase, siendo la primera tabla para el conjunto de entrenamiento y la segunda para el conjunto de test.

Tabla 5.1: Métricas asociadas a los modelos de SVM

Modelos	**Entrenamiento**		**Test**		**# Variables**
	Accuracy	**AUC**	**Accuracy**	**AUC**	
Unique	0.875	0.930	0.333	0.756	39
Uncorr	0.875	0.936	0.333	0.756	36
Mi	0.875	0.917	0.5	0.606	16
MiCorr	0.781	0.914	0	0.337	7
Forward Unique	0.906	0.948	0.166	0.712	34
Forward Uncorr	1	0.992	0.333	0.481	4
Forward Mi	0.843	0.902	0.333	0.594	12
Forward MiCorr	0.688	0.867	0.166	0.394	5
Backward Unique	0.938	0.915	0.5	0.756	25
Backward Uncorr	0.844	0.922	0.333	0.756	34
Backward Mi	0.875	0.917	0.5	0.606	16
Backward MiCorr	0.688	0.868	0.166	0.394	6

En el marco conceptual se revisaron las distintas métricas que se utilizan para evaluar mo-

delos de Machine Learning, y se menciona que en el caso de problemas médicos, en particular de diagnóstico interesa revisar la métrica de Recall, ya que esta determina a cuantos registros se logra identificar correctamente en cada clase, es decir, a cuantos pacientes se logra diagnosticar correctamente. Es por esto que en el proyecto Neuronat se debe revisar cuidadosamente dicha métrica. Sin embargo, en el contexto del presente trabajo, si bien se está dentro de un contexto médico, los datos no están asociados a pacientes, y las etiquetas de cada clase no tienen un significado médico, simplemente es el segmento etario de las personas, por lo que la métrica de Recall si bien se analiza, no se considera como la más importante del problema, a diferencia de métricas como AUC o Weighted F1-Score que miden el desempeño general del problema considerando la capacidad predictiva por clase. El accuracy de un modelo da una buena percepción general del desempeño de este, sin embargo es necesario entrar al detalle de las clases, ya que si se tiene una clase con 99 muestras y otra con 1 muestra, el modelo tendrá un alto accuracy ya que podrá predecir siempre que una muestra pertenece a la clase 1 y estará bien un 99 % de las veces, y no predecir correctamente ningún registro de la otra clase.

Tabla 5.2: Métricas ponderadas asociadas a las clases de los modelos de SVM

Modelos	**Entrenamiento**				**Test**			
	Recall		**F1-Score**		**Recall**		**F1-Score**	
	M	**W**	**M**	**W**	**M**	**W**	**M**	**W**
Unique	0.88	0.88	0.88	0.88	0.25	0.33	0.25	0.33
Uncorr	0.88	0.88	0.88	0.88	0.25	0.33	0.25	0.33
Mi	0.88	0.88	0.88	0.88	0.50	0.50	0.46	0.50
MiCorr	0.78	0.78	0.78	0.78	0	0	0	0
Forward Unique	0.91	0.91	0.91	0.91	0.12	0.17	0.10	0.13
Forward Uncorr	1	1	1	1	0.25	0.33	0.20	0.27
Forward Mi	0.84	0.84	0.85	0.85	0.38	0.33	0.25	0.25
Forward MiCorr	0.69	0.69	0.65	0.65	0.12	0.17	0.12	0.17
Backward Unique	0.94	0.94	0.94	0.94	0.50	0.50	0.46	0.50
Backward Uncorr	0.84	0.84	0.84	0.84	0.25	0.33	0.25	0.33
Backward Mi	0.88	0.88	0.88	0.88	0.50	0.50	0.46	0.50
Backward MiCorr	0.69	0.69	0.65	0.65	0.12	0.17	0.12	0.17

M: Macro Average (métrica que se obtiene al calcular la ponderación simple entre los puntajes de Recall, Precisión o F1-Score)

W: Weighted Average (métrica que se obtiene al calcular la ponderación utilizando los tamaños de las clases y los valores de Recall, Precisión o F1-Score)

Otro aspecto importante a considerar son los resultados de entrenamiento y test, ya que en un típico problema de Machine Learning se debe tener en especial consideración los resultados de test, ya que son estos los que indican que tan bien puede generalizar un modelo. Si bien en este caso se interpreta de la misma manera, el hecho de tener pocos registros en la muestra de test (6 muestras) y tantas clases hace que el modelo simplemente eligiendo al azar obtenga buenos o malos resultados, por lo tanto no se debe asignar tanto peso a los resultados de

test como en otro problema de Machine Learning, aunque si debe ser considerado en cierta magnitud. Por otro lado, dado que el objetivo del presente trabajo es determinar las variables que mejor caractericen el desempeño de las funciones ejecutivas (reflejadas en la edad), el hecho de que el conjunto de entrenamiento de buenos resultados representa que el modelo si puede detectar patrones en los datos y esos patrones caracterizan el desempeño de las funciones ejecutivas.

Por lo tanto, considerando lo anterior, a modo general los modelos que presentan un mejor desempeño son *Mi*, *Backward Unique* y *Backward Mi*, ya que estos presentan los valores más altos en las 4 métricas presentadas en las tablas 5.1 y 5.2, considerando ambos conjuntos de entrenamiento y test, sin embargo, *Mi* y *Backward Mi* son 2 modelos idénticos, con los mismos resultados y mismas variables (porque *Backward Mi* se construye a partir de *Mi*, por lo que si tienen la misma cantidad de variables son exactamente las mismas), y es por esto que simplemente se deja *Mi*. En el capítulo de discusión se analizará más en profundidad los 2 modelos que presentan mejores resultados.

Random Forest

Al igual que en el caso de Support Vector Machine, en Random Forest se presentan los mismos 12 conjuntos de variables, con la excepción de que los conjuntos forward y backward se generan utilizando los modelos de Random forest aplicado sobre los 4 filtros antes mencionados. En las tablas 5.3 y 5.4 se muestran las métricas asociadas a los modelos de Random Forest, y en los anexos en las tablas C.4 y C.5 se muestra el desglose de las métricas Recall y F1-Score por clase.

Tabla 5.3: Métricas asociadas a los modelos de Random Forest

Modelos	**Entrenamiento**		**Test**		**# Variables**
	Accuracy	**AUC**	**Accuracy**	**AUC**	
Unique	1	1	0.333	0.706	39
Uncorr	1	1	0.333	0.675	36
Mi	1	1	0.5	0.675	16
MiCorr	1	1	0.5	0.762	7
Forward Unique	1	1	0.5	0.525	9
Forward Uncorr	1	1	0.5	0.613	10
Forward Mi	1	1	0.333	0.581	2
Forward MiCorr	1	1	0.166	0.531	4
Backward Unique	1	1	0.667	0.769	14
Backward Uncorr	1	1	0.5	0.756	15
Backward Mi	1	1	0.5	0.775	9
Backward MiCorr	1	1	0.333	0.556	4

Tabla 5.4: Métricas ponderadas asociadas a las clases de los modelos de Random Forest

Modelos	Entrenamiento				Test			
	Recall		F1-Score		Recall		F1-Score	
	M	W	M	W	M	W	M	W
Unique	1	1	1	1	0.38	0.33	0.29	0.28
Uncorr	1	1	1	1	0.38	0.33	0.29	0.28
Mi	1	1	1	1	0.38	0.33	0.25	0.25
MiCorr	1	1	1	1	0.50	0.50	0.42	0.39
Forward Unique	1	1	1	1	0.50	0.50	0.52	0.52
Forward Uncorr	1	1	1	1	0.50	0.50	0.42	0.39
Forward Mi	1	1	1	1	0.38	0.33	0.38	0.33
Forward MiCorr	1	1	1	1	0.12	0.17	0.10	0.13
Backward Unique	1	1	1	1	0.62	0.67	0.58	0.67
Backward Uncorr	1	1	1	1	0.50	0.50	0.42	0.44
Backward Mi	1	1	1	1	0.50	0.50	0.37	0.38
Backward MiCorr	1	1	1	1	0.25	0.33	0.17	0.22

M: Macro Average (métrica que se obtiene al calcular la ponderación simple entre los puntajes de Recall, Precisión o F1-Score)

W: Weighted Average (métrica que se obtiene al calcular la ponderación utilizando los tamaños de las clases y los valores de Recall, Precisión o F1-Score)

Analizando las 2 tablas anteriores, se puede llegar a la conclusión de que los 2 mejores modelos son *Backward Unique* y *Backward Uncorr* ya que estos son los que presentan mayores valores de accuracy y AUC en la muestra de Test (ya que en entrenamiento son todos iguales) y mayores valores de Weighted Recall y Weighted F1-Score.

Naive Bayes

Finalmente, se presentan los resultados de los 12 modelos de Naive Bayes en las tablas 5.5 y 5.6, y en las tablas C.6 y C.7 de los anexos.

Tabla 5.5: Métricas asociadas a los modelos de Naive Bayes

Modelos	Entrenamiento		Test		# Variables
	Accuracy	AUC	Accuracy	AUC	
Unique	0.967	0.999	0.333	0.462	39
Uncorr	0.969	0.997	0.333	0.462	36
Mi	0.906	0.974	0.333	0.412	16
MiCorr	0.813	0.970	0.333	0.356	7
Forward Unique	0.719	0.908	0.5	0.712	14
Forward Uncorr	0.750	0.901	0.5	0.712	12
Forward Mi	0.625	0.872	0.5	0.675	6
Forward MiCorr	0.813	0.954	0.5	0.6	4
Backward Unique	0.813	0.927	0.5	0.594	24
Backward Uncorr	0.813	0.917	0.5	0.744	23
Backward Mi	0.938	0.987	0.333	0.388	8
Backward MiCorr	0.781	0.958	0.333	0.294	5

Tabla 5.6: Métricas ponderadas asociadas a las clases de los modelos de Naive Bayes

Modelos	Entrenamiento				Test			
	Recall		F1-Score		Recall		F1-Score	
	M	W	M	W	M	W	M	W
Unique	0.97	0.97	0.97	0.97	0.25	0.33	0.20	0.27
Uncorr	0.97	0.97	0.97	0.97	0.25	0.33	0.20	0.27
Mi	0.91	0.91	0.91	0.91	0.25	0.33	0.20	0.27
MiCorr	0.81	0.81	0.82	0.82	0.25	0.33	0.12	0.17
Forward Unique	0.72	0.72	0.73	0.73	0.50	0.50	0.42	0.47
Forward Uncorr	0.75	0.75	0.76	0.76	0.50	0.50	0.42	0.47
Forward Mi	0.62	0.62	0.64	0.64	0.50	0.50	0.42	0.47
Forward MiCorr	0.81	0.81	0.81	0.81	0.50	0.50	0.33	0.33
Backward Unique	0.81	0.81	0.82	0.82	0.50	0.50	0.42	0.47
Backward Uncorr	0.81	0.81	0.82	0.82	0.50	0.50	0.42	0.47
Backward Mi	0.94	0.94	0.94	0.94	0.25	0.33	0.20	0.27
Backward MiCorr	0.78	0.78	0.79	0.79	0.25	0.33	0.12	0.17

M: Macro Average (métrica que se obtiene al calcular la ponderación simple entre los puntajes de Recall, Precisión o F1-Score)

W: Weighted Average (métrica que se obtiene al calcular la ponderación utilizando los tamaños de las clases y los valores de Recall, Precisión o F1-Score)

Para los modelos de Naive Bayes se tiene un caso complicado de interpretar, ya que varios modelos presentan altos valores en la muestra de entrenamiento y bajos en la muestra de test, o viceversa, por lo que se tendrá que optar por un criterio sobre el otro. En este

caso se prioriza la métrica de AUC, ya que muestra un desempeño general de los modelos, quedando de esta manera los modelos *Forward Unique*, *Forward Uncorr* y *Backward Uncorr*. Se eligen dichos modelos ya que si bien no presentan los valores más altos en entrenamiento, si son los que tienen valores más altos en test sin disminuir considerablemente el desempeño en entrenamiento.

Resultados generales sobre la base manual

Por lo tanto, se tienen 7 modelos que presentan buenos resultados, lo que indica que independiente de que modelo sea mejor, con los datos que se generan en la plataforma y con las métricas construidas con el equipo clínico de Neuronat se puede diferenciar a las personas según su segmento etario, lo que indica que el videojuego está capturando el desempeño de las funciones ejecutivas.

Los modelos seleccionados son *SVM Mi*, *Backward Unique*, *RFC Backward Unique*, *RFC Backward Uncorr*, *NB Forward Unique*, *NB Forward Uncorr* y *NB Backward Uncorr*. En la tabla 5.7 se puede apreciar un resumen de los mejores modelos en orden de mejor a peor, y en anexos en la tabla C.8 se puede apreciar las variables que conforman cada conjunto con los que se entrenaron los modelos.

Tabla 5.7: Resumen de mejores modelos para la base manual

		Random Forest		**Support Vector Machine**		**Naive Bayes**		
		Backward Unique	**Backward Uncorr**	**Backward Unique**	**Mi**	**Backward Uncorr**	**Forward Uncorr**	**Forward Unique**
E	**Acc.**	1	1	0.94	0.88	0.81	0.75	0.72
	AUC	1	1	0.92	0.92	0.92	0.90	0.91
	W. Recall	1	1	0.94	0.88	0.81	0.75	0.72
	W. F1	1	1	0.94	0.88	0.82	0.76	0.73
T	**Acc.**	0.67	0.50	0.50	0.50	0.50	0.50	0.50
	AUC	0.77	0.76	0.76	0.61	0.74	0.71	0.71
	W. Recall	0.67	0.50	0.50	0.50	0.50	0.50	0.50
	W. F1	0.67	0.44	0.50	0.50	0.47	0.47	0.47

E: Resultados asociados a la muestra de entrenamiento.

T: Resultados asociados a la muestra de test.

Acc: Métrica accuracy.

W: Weighted

Como se puede apreciar en los resultados de la tabla 5.7 el modelo *Backward Unique* con el algoritmo Random Forest es el que presenta un mejor desempeño en prácticamente todos los criterios de evaluación. Es importante definir un criterio de que es un buen modelo, o cuales son buenos resultados ya que si existe una herramienta que permite clasificar segmentos etarios con un 90 % de accuracy en la muestra de test, el modelo *Backward Unique* de Random Forest que se entrenó tiene malos resultados en comparación. Para el caso del trabajo realizado y el problema identificado, al desarrollar el estudio de variables utilizando 4 segmentos

etarios como etiqueta para entrenar modelos, no se tiene otro punto de comparación más que la probabilidad de obtener cada clase (segmento etario) según como estén distribuidos los datos, que en este caso, al tener clases balanceadas gracias al método SMOTE utilizado para aumentar los registros de clases con menos datos, se puede estimar que la probabilidad de que un registro pertenezca a una clase cualquiera es de aproximadamente un 25 %. Por lo tanto, bajo ese punto de vista todos los modelos de la tabla 5.7 son mejores que el caso base en donde se elige aleatoriamente una clase, lo que quiere decir que con los datos obtenidos y el procesamiento de estos se puede entrenar un modelo que detecte patrones que caractericen cada clase.

Por otro lado, es importante revisar qué variables son las que permiten caracterizar de mejor manera las clases. Para esto se revisan las variables de los primeros 2 modelos, *Backward Unique* y *Backward Uncorr* del algoritmo Random Forest, y luego se ven las variables que más aparecen en común entre todos los modelos de la tabla 5.7. Para esto se utiliza la tabla de anexos C.8.

Los 2 mejores modelos tienen 14 y 15 variables, de las cuales comparten 10. Estas 10 variables se pueden apreciar en la tabla 5.8. El modelo *Backward Unique* cuenta con las 10 variables de la tabla recién mencionada y además con las variables: *option_label1*, *takeOrderTotalTime3*, *totalTime3* y *dilemma_y*, mientras que el modelo *Backward Uncorr* tiene las 10 variables de la tabla 5.8 y *takeOrderTotalTime1*, *persevTakeOrder3*, *dilemma_x*, *deliveryTime3* y *transactionTime3*. Se puede apreciar de estas variables que el tiempo de resolución de hitos del videojuego parece tener una importancia significativa a la hora de tomar una decisión para el primer modelo, sin embargo el tipo de variable que más se repite es el de cantidad de errores, en particular en la actividad de toma de orden. El otro tipo de variables que se encuentran en ambos modelos son las 2 variables demográficas y decisiones específicas que toma el mesero como si se presenta o no (*greet_boolean*), o como resuelve los 2 dilemas que se presentan durante el juego.

Tabla 5.8: Variables de los 2 mejores modelos para la base manual

Genero	**takeOrderButtonTime1**
Educacion	**greet_boolean3**
greet_boolean1	**takeOrderErrorCount3**
MenuErrorCount1	**first_bell3**
takeOrderErrorCount1	**persevDeliveryErrors3**

Por otro lado, si se cuentan las variables más populares entre los 7 mejores modelos se tienen las siguientes variables: *Educacion*, *greet_boolean1*, *MenuErrorCount1*, *takeOrderTotalTime1*, *takeOrderButtonTime1*, *takeOrderErrorCount3*, *takeOrderTotalTime3* y *deliveryTime3*. Se consideran variables populares si aparecen en por lo menos 5 modelos de los 7 mejores. 5 de las 8 variables más populares aparecen en la tabla 5.8, poniendo énfasis nuevamente en las variables temporales y de cantidad de errores.

5.1.2. Base automática

A continuación se presenta el mismo tipo de tablas presentadas en el apartado de la base manual para los mismos 12 conjuntos de variables, pero ahora aplicados sobre la base automática que cuenta con poco más de 600 variables en un inicio.

Support Vector Machine

En las tablas 5.9 y 5.10 se puede apreciar el resumen de las 4 métricas principales sobre los conjuntos de variables de la base automática. De estas tablas se puede apreciar un aumento de los valores generales de las métricas en comparación con los obtenidos en la base manual, lo cual se debe a que se tiene una dimensionalidad mucho mayor que en la otra base, no en cantidad de registros ya que se tienen los mismos, pero si en cantidad de columnas o variables. Por lo tanto, por cada registro se tiene mucha más información para poder caracterizar las etiquetas.

Tabla 5.9: Métricas asociadas a los modelos de SVM para la base automática

Modelos	**Entrenamiento**		**Test**		**# Variables**
	Accuracy	**AUC**	**Accuracy**	**AUC**	
Unique	1	1	0.333	0.531	255
Uncorr	0.969	0.956	0.667	0.875	72
Mi	1	1	0.167	0.438	102
MiCorr	0.938	0.977	0.50	0.825	15
Forward Unique	0.969	0.962	0.667	0.594	33
Forward Uncorr	0.969	0.918	0.667	0.713	62
Forward Mi	0.969	0.977	0.50	0.731	3
Forward MiCorr	1	1	0.333	0.512	13
Backward Unique	1	1	0.167	0.50	15
Backward Uncorr	0.938	0.927	0.833	0.906	70
Backward Mi	1	1	0.333	0.60	38
Backward MiCorr	0.938	0.977	0.50	0.825	15

Tabla 5.10: Métricas ponderadas asociadas a las clases de los modelos de SVM para la base automática

Modelos	Entrenamiento				Test			
	Recall		F1-Score		Recall		F1-Score	
	M	W	M	W	M	W	M	W
Unique	1	1	1	1	0.38	0.33	0.42	0.39
Uncorr	0.97	0.97	0.97	0.97	0.62	0.67	0.57	0.60
Mi	1	1	1	1	0.12	0.17	0.17	0.22
MiCorr	0.94	0.94	0.94	0.94	0.50	0.50	0.52	0.52
Forward Unique	0.97	0.97	0.97	0.97	0.62	0.67	0.54	0.61
Forward Uncorr	0.97	0.97	0.97	0.97	0.62	0.67	0.62	0.67
Forward Mi	0.97	0.97	0.97	0.97	0.50	0.50	0.38	0.42
Forward MiCorr	1	1	1	1	0.38	0.33	0.27	0.24
Backward Unique	1	1	1	1	0.12	0.17	0.17	0.22
Backward Uncorr	0.94	0.94	0.94	0.94	0.88	0.83	0.87	0.82
Backward Mi	1	1	1	1	0.38	0.33	0.42	0.39
Backward MiCorr	0.94	0.94	0.94	0.94	0.50	0.50	0.52	0.52

M: Macro Average (métrica que se obtiene al calcular la ponderación simple entre los puntajes de Recall, Precisión o F1-Score)

W: Weighted Average (métrica que se obtiene al calcular la ponderación utilizando los tamaños de las clases y los valores de Recall, Precisión o F1-Score)

En las tablas C.9 y C.10 de los anexos se puede encontrar el desglose por clase de las métricas presentadas en la tabla C.9.

Según lo presentado en las tablas anteriores se puede apreciar que aquellos modelos con una alta capacidad predictiva para la muestra de entrenamiento tienen los peores desempeños en la muestra de test, lo que quiere decir que probablemente haya sobreajuste en esos modelos, es decir, el modelo aprendió tan bien para las muestras de entrenamiento que ya no puede generalizar y predice mal para nuevos valores. Es por esto que dichos modelos se descartan, ya que no sirve que no puedan predecir registros nuevos. De los demás modelos restantes los 3 que destacan son aquellos relacionados con el filtro de correlación, es decir los modelos *Uncorr*, *Forward Uncorr* y *Backward Uncorr*, en donde claramente el mejor de estos 3 es ***Backward Uncorr***, alcanzando valores sobre 90 % en el conjunto de entrenamiento y sobre 80 % en la muestra de test.

Random Forest

A continuación se presentan las 2 tablas resumen de las métricas asociadas al algoritmo de Random Forest. Al igual que en las tablas recién presentadas de SVM, se puede apreciar un incremento general de los resultados de los modelos en comparación con los mismos de la base manual.

Tabla 5.11: Métricas asociadas a los modelos de Random Forest para la base automática

Modelos	Entrenamiento		Test		# Variables
	Accuracy	AUC	Accuracy	AUC	
Unique	1	1	0.667	0.887	255
Uncorr	1	1	0.50	0.794	72
Mi	1	1	0.833	0.887	102
MiCorr	1	1	0.667	0.844	15
Forward Unique	1	1	0.167	0.650	5
Forward Uncorr	1	1	0.333	0.419	36
Forward Mi	1	1	0.333	0.631	4
Forward MiCorr	1	1	0.667	0.750	11
Backward Unique	1	1	0.333	0.631	12
Backward Uncorr	1	1	0.667	0.825	68
Backward Mi	1	1	0.333	0.713	54
Backward MiCorr	1	1	0.667	0.812	14

Tabla 5.12: Métricas ponderadas asociadas a las clases de los modelos de Random Forest para la base automática

Modelos	Entrenamiento				Test			
	Recall		F1-Score		Recall		F1-Score	
	M	W	M	W	M	W	M	W
Unique	1	1	1	1	0.62	0.67	0.62	0.67
Uncorr	1	1	1	1	0.38	0.50	0.30	0.40
Mi	1	1	1	1	0.88	0.83	0.83	0.83
MiCorr	1	1	1	1	0.75	0.67	0.67	0.56
Forward Unique	1	1	1	1	0.12	0.17	0.12	0.17
Forward Uncorr	1	1	1	1	0.25	0.33	0.25	0.33
Forward Mi	1	1	1	1	0.38	0.33	0.38	0.33
Forward MiCorr	1	1	1	1	0.75	0.67	0.67	0.56
Backward Unique	1	1	1	1	0.38	0.33	0.42	0.39
Backward Uncorr	1	1	1	1	0.62	0.67	0.62	0.67
Backward Mi	1	1	1	1	0.38	0.33	0.42	0.39
Backward MiCorr	1	1	1	1	0.75	0.67	0.67	0.56

M: Macro Average (métrica que se obtiene al calcular la ponderación simple entre los puntajes de Recall, Precisión o F1-Score)

W: Weighted Average (métrica que se obtiene al calcular la ponderación utilizando los tamaños de las clases y los valores de Recall, Precisión o F1-Score)

En las tablas C.11 y C.12 del anexo se puede apreciar más detalle sobre el desempeño de

los modelos por clase.

Al igual que en lo presentado en las tablas 5.3 y 5.4 para la base manual, los resultados de entrenamiento tienen todos un 100 % de Accuracy, Recall y F1-Score, además de un valor 1 para AUC. Es por esto que se comparan los modelos según sus valores de test. En un principio se descartan todos los modelos que tengan resultados inferiores a un 50 %, quedando así 6 modelos restantes. Claramente el mejor modelo de los 6 es el filtro ***Mi***, que tiene 102 variables. De los 5 restantes, los 3 asociados al conjunto *MiCorr* presentan valores idénticos en la tabla 5.12, por lo tanto, de esos 3 se elige *MiCorr* ya que presenta un valor AUC mayor a los otros 2. Finalmente, la única diferencia entre *Unique* y *Backward Uncorr* es el valor AUC de test, que es mayor en el modelo *Unique*. Por lo tanto, los 3 mejores modelos de este apartado son *Mi*, *MiCorr* y *Unique*.

Naive Bayes

Finalmente, en las tablas 5.13 y 5.14, y las tablas C.13 C.14 de anexos se muestran los resultados de los modelos Naive Bayes aplicados sobre la base automática.

Tabla 5.13: Métricas asociadas a los modelos de Naive Bayes para la base automática

Modelos	**Entrenamiento**		**Test**		**# Variables**
	Accuracy	**AUC**	**Accuracy**	**AUC**	
Unique	0.875	0.991	0.50	0.544	255
Uncorr	0.938	0.992	0.50	0.675	72
Mi	0.969	1	0.333	0.487	102
MiCorr	0.875	0.986	0.667	0.938	15
Forward Unique	1	1	0.50	0.569	102
Forward Uncorr	0.906	0.978	0.50	0.775	7
Forward Mi	1	1	0.667	0.894	11
Forward MiCorr	0.969	1	0.667	0.938	8
Backward Unique	0.969	0.992	0.50	0.497	16
Backward Uncorr	0.906	0.978	0.50	0.762	24
Backward Mi	0.563	0.635	0.50	0.425	11
Backward MiCorr	1	1	0.50	0.938	9

Tabla 5.14: Métricas ponderadas asociadas a las clases de los modelos de Naive Bayes para la base automática

Modelos	Entrenamiento				Test			
	Recall		F1-Score		Recall		F1-Score	
	M	W	M	W	M	W	M	W
Unique	0.88	0.88	0.88	0.88	0.38	0.50	0.33	0.43
Uncorr	0.94	0.94	0.94	9.94	0.38	0.50	0.29	0.39
Mi	0.97	0.97	0.97	0.97	0.25	0.33	0.20	0.27
MiCorr	0.88	0.88	0.87	0.87	0.75	0.67	0.67	0.56
Forward Unique	1	1	1	1	0.38	0.50	0.33	0.43
Forward Uncorr	0.91	0.91	0.90	0.90	0.50	0.50	0.45	0.43
Forward Mi	1	1	1	1	0.62	0.67	0.54	0.64
Forward MiCorr	0.97	0.97	0.97	0.97	0.62	0.67	0.58	0.61
Backward Unique	0.97	0.97	0.97	0.97	0.38	0.50	0.29	0.39
Backward Uncorr	0.91	0.91	0.91	0.91	0.50	0.50	0.45	0.43
Backward Mi	0.56	0.56	0.52	0.52	0.38	0.50	0.33	0.44
Backward MiCorr	1	1	1	1	0.50	0.50	0.33	0.33

M: Macro Average (métrica que se obtiene al calcular la ponderación simple entre los puntajes de Recall, Precisión o F1-Score)

W: Weighted Average (métrica que se obtiene al calcular la ponderación utilizando los tamaños de las clases y los valores de Recall, Precisión o F1-Score)

En este caso sucede prácticamente lo mismo que con Random Forest, en donde todos los modelos salvo uno tienen altos valores de entrenamiento, por lo que la decisión sobre que modelos son mejores se toma en base a los resultados de test. Por lo general los modelos no dan malos resultados como en otros casos, sin embargo, descartando los "peores" se obtiene que los 3 mejores son *Forward MiCorr*, *MiCorr* y *Forward Mi*, en donde el primero es el que entrega mejores resultados con 8 variables.

Resultados generales sobre la base automática

Para la base automática no hace sentido interpretar las variables, ya que como se ha mencionado antes, las variables son un conjunto de operaciones matemáticas que no tiene un significado teórico. Lo importante a notar en este apartado es que la gran mayoría de modelos entrega resultados positivos, considerablemente mayor al caso base en donde se selecciona aleatoriamente una clase (25 % de probabilidad aproximadamente), teniendo por lo general sobre un 60 % en gran parte de las métricas. Esto es importante ya que permite confirmar que con los datos exportados por el videojuego se puede caracterizar el desempeño de una función ejecutiva. Otro punto relevante para considerar es que a medida que se aumenta la dimensionalidad mejora el desempeño de los modelos, y la dimensionalidad se puede aumentar en cantidad de variables o en cantidad de registros, lo que da pie a mejoras considerables en futuras iteraciones del proyecto Neuronat. Finalmente, en la tabla 5.15 se puede encontrar un resumen de los 9 mejores modelos, 3 por cada algoritmo, obtenidos de la base automática, en

donde el modelo entrenado con la base Backward Uncorr y con el algoritmo SVM es el mejor de estos, y el modelo con la base Mi y el algoritmo Random Forest es el segundo mejor.

Tabla 5.15: Resumen de mejores modelos para la base automática

		Random Forest			Support Vector Machine			Naive Bayes		
		Mi	**MiCorr**	**Unique**	**Backward Uncorr**	**Forward Uncorr**	**Uncorr**	**Forward MiCorr**	**MiCorr**	**Forward Mi**
E	**Acc.**	1	1	1	0.94	0.97	0.97	0.97	0.88	1
	AUC	1	1	1	0.93	0.92	0.96	1	0.99	1
	W. Recall	1	1	1	0.94	0.97	0.97	0.97	0.88	1
	W. F1	1	1	1	0.94	0.97	0.97	0.97	0.87	1
T	**Acc.**	0.83	0.67	0.67	0.83	0.67	0.67	0.67	0.67	0.67
	AUC	0.89	0.84	0.89	0.91	0.71	0.88	0.94	0.94	0.90
	W. Recall	0.83	0.67	0.67	0.83	0.67	0.67	0.67	0.67	0.67
	W. F1	0.83	0.56	0.67	0.82	0.67	0.60	0.61	0.56	0.64

E: Resultados asociados a la muestra de entrenamiento.

T: Resultados asociados a la muestra de test.

Acc: Métrica accuracy.

W: Weighted

5.2. Discusión

En los apartados anteriores se presentan los resultados de los modelos generados, además de un breve análisis sobre estos resultados.

Si bien en los resultados numéricos se puede apreciar que efectivamente existe una tendencia por parte de los distintos modelos en base a los distintos conjuntos de datos y algoritmos de caracterizar correctamente las distintas clases, lo que se puede interpretar como que efectivamente con los datos obtenidos de la plataforma Neuronat se puede representar el desempeño de funciones ejecutivas, hay que tener claro un par de cosas antes de obtener conclusiones reales.

En primer lugar, la plataforma con la que se recopilaron los datos fue entregada en una versión muy básica y prototipo, lo que implica que pueden haber errores en los datos obtenidos en el sentido que los jugadores (sujetos de prueba) podrían haberse confundido a la hora de jugar, ya que faltaban indicaciones visuales y textuales, habían elementos gráficos y funcionales de más, que puede llevar a la confusión de los jugadores al pensar que estos son parte del juego, había una mesa dentro del contexto del juego que tenía el error de que esta no se iba y además los datos referentes a esa mesa no se exportaban de la plataforma, entre otros. En cualquier proyecto en el que se toman datos, la herramienta o forma de tomar datos es fundamental para asegurar una buena calidad de datos, ya que con datos de mala calidad se obtienen resultados de mala calidad, y por eso el popular dicho en el rubro de Data Science “garbage in, garbage out” (basura entra, basura sale).

Luego, debido a que la plataforma funcional fue entregada a principios de febrero de 2021,

se tuvo poco tiempo para tomar datos con esta (aproximadamente 3 semanas), por lo que de los 40 registros que se tenía planeado obtener, se obtuvieron 27, y cuando se tiene una cantidad tan pequeña de datos en un problema de Machine Learning, el hecho de perder 13 registros más puede ser bastante perjudicial, ya que los modelos no tienen suficientes registros para "aprender correctamente". Si bien se simularon algunos datos para lograr obtener en total 38 registros, siguen siendo pocos datos para entrenar correctamente un modelo de Machine Learning.

Finalmente, y lo más importante, para poder comprobar si se cumple o no la hipótesis de investigación se debe comparar el desempeño en la plataforma Neuronat de 4 grupos distintos, en donde la única diferencia demográfica de esos grupos es la edad. Para poder realizar una comparación válida entre estos grupos, estos deben presentar las mismas condiciones contextuales e idealmente un mismo entorno, es decir, que sólo se evalúe el desempeño en la plataforma y todas las demás variables sean constantes. Lamentablemente, debido a la pandemia del COVID-19 esto no se pudo dar, ya que la plataforma tuvo que ser contestada remotamente a través de un enlace en Internet, lo que quiere decir que todas las personas estaban en entornos distintos bajo condiciones distintas (un computador más lento o rápido, un sistema operativo distinto, un ambiente con más ruido exterior, una tercera persona ayudando en el desarrollo del videojuego, o cualquier otra variable ambiental/contextual que se pueda presentar). Lo anterior es particularmente importante a la hora de trabajar con grupos diferenciados por la edad, ya que al ser una interacción con una plataforma tecnológica es de esperar que las personas más jóvenes de la muestra la entiendan mejor, aprendan más rápido como usarla, o sea más intuitivo para ellos el uso de la plataforma, en comparación con las personas de mayor edad en la muestra (62 años). A este fenómeno se le llamará *sesgo etario*, ya que efectivamente induce diferencias en el resultado de las variables dependiendo del grupo. Las diferencias en estos resultados pueden darse en las variables temporales (ya que una persona que tiene más dificultades con la tecnología se va a demorar más en entender cómo funciona un videojuego y como interactuar con este) y en las variables de cantidad de errores (por ejemplo la cantidad de errores en la toma de orden, ya que al no entender completamente como interactuar con el videojuego aprieta opciones incorrectas, generando así más errores). Sin embargo, se plantea como hipótesis (que no podrá ser validada o rechazada en el presente trabajo) que existen variables que no dependen del *sesgo etario* como por ejemplo la decisión que toma un jugador en un momento especifico como las variables *dilemma_x* y *dilemma_y* que representan si el jugador le informó a los clientes que no queda vino o no, y si el jugador le da una respuesta positiva o negativa a un cliente conflictivo, respectivamente; o también variables que representan el comportamiento de un jugador, como por ejemplo si saluda o no a los clientes, o si recuerda darle el dulce a los clientes al terminar de atender la mesa. Se plantea dicha hipótesis ya que esas variables no están relacionadas directamente con las interacciones más complejas del videojuego, sino que son más bien decisiones que toma una persona que se representa en la plataforma como 2 botones: opción A u opción B.

Por lo tanto, aunque los resultados numéricos obtenidos de los modelos indiquen que efectivamente existen diferencias entre segmentos etarios, y que estas diferencias se manifiestan en los valores de las variables del videojuego, no se puede asegurar que esos resultados sean válidos al punto de poder confirmar que el videojuego permite detectar el desempeño de funciones ejecutivas. A pesar de lo anterior, si existen ciertos indicios de que el videojuego

permite diferenciar segmentos etarios en base a las variables obtenidas del videojuego, como la mencionada anteriormente de que existen variables que no necesariamente se ven afectadas por el *sesgo etario*, o principalmente porque los modelos presentan buenos resultados de entrenamiento para todas las clases, no solamente para las clases extremas que serían las que se verían afectadas por el *sesgo etario*.

Capítulo 6

Evaluación de impacto social y económica

En el apartado 1.3.4 del presente trabajo de título se calcula un impacto estimado directo asociado a la reducción de costos de tratamientos y de licencias laborales para pacientes con traumatismo craneoencefálico (TEC), ya que en el proyecto Neuronat en el cual se enmarca el trabajo de título, se trabaja con pacientes que sufren de dicha condición, debido a que esta puede provocar una Disfunción Ejecutiva (DE) [2]. Sin embargo, la herramienta desarrollada en el proyecto Neuronat no es específica para pacientes de TEC, sino que busca detectar el desempeño de distintas funciones ejecutivas, para así detectar si es que existe algún trastorno en alguna de estas. Esto implica que con la plataforma Neuronat y con los resultados obtenidos del presente trabajo de título, se podría evaluar la presencia de Disfunción Ejecutiva transversalmente a lo largo de varias otras enfermedades o condiciones sabidas que pueden provocar una DE, como las que aparecen en la figura A.1 del anexo.

Para la evaluación que realizada se consideran las siguientes enfermedades/condiciones: Parkinson, Alzheimer, Esclerosis Lateral Amiotrófica (ELA o ALS en inglés), Traumatismo Craneoencefálico (TEC), Tumor cerebral, Epilepsia, Esclerosis múltiple, Desorden depresivo mayor, Esquizofrenia y VIH. En primera instancia se presenta la incidencia a nivel mundial de cada enfermedad, para estimar cuantos casos nuevos se tienen por año. Luego, se presenta el costo promedio por tratamiento de la enfermedad, el cuál es distinto al tratamiento de la DE, simplemente para demostrar lo costoso que ya es de por sí cada enfermedad. Posterior a eso, y tomando el supuesto de que un 10 % de los pacientes para todas las enfermedades desarrollan una DE, se calcula la cantidad de personas con Disfunción Ejecutiva, a nivel mundial y a nivel nacional. El 10 % mencionado es un valor conservador, ya que varias de las enfermedades mencionadas anteriormente suelen desarrollar alguna DE como el Alzheimer [83], ELA [84] o epilepsia [85]. Finalmente, se asume que un 1 % de los pacientes que desarrollan una DE son mal diagnosticados con DE debido a lo poco preciso de las herramientas actuales de evaluación de DE. Luego, considerando que el tratamiento de una Disfunción Ejecutiva en Chile tiene un valor aproximado de 3 millones de pesos de acuerdo con lo enunciado en el apartado 1.3.4, se calcula un costo a nivel mundial que se tendría por pacientes mal diagnosticados. Es importante señalar que dicho costo de 3 millones de pesos es un costo fijado en Chile, el cuál puede variar considerablemente según el país, sin embargo, no se cuenta con el dato de cuál es el costo del tratamiento de DE en otros países por lo que se utiliza de igual manera el costo chileno, teniendo en cuenta que los resultados obtenidos son una aproximación. Finalmente,

se realiza la misma secuencia de pasos y cálculos pero con datos nacionales.

Por lo tanto, se procede con el primer y segundo paso, enunciando las incidencias mundiales y nacionales para las distintas enfermedades y los costos de tratamientos anuales asociados a estas. Todas las incidencias presentadas están calculadas por cada 100.000 habitantes.

- La enfermedad de Parkinson en el año 2005 tenía una incidencia mundial de 38,4 casos [86], mientras que en Chile al año 2017 se tienen 12 casos aproximadamente [87]. Se estima que los gastos para esta enfermedad oscilan entre 4 y 4,5 millones de pesos [88], y sin considerar una cirugía a la que se someten varios pacientes de Parkinson llamada Estimulación Cerebral Profunda, que tiene un costo aproximado de 20 millones de pesos, los cuales no tienen cobertura ni ISAPRE ni FONASA [89].

- La enfermedad de Alzheimer presenta una incidencia de 420 casos a nivel mundial [90], mientras que en Chile se tiene que un 1,06 % de la población total padece de alguna demencia, y que el 70 % aproximadamente de las demencias corresponden a la enfermedad de Alzheimer [91][92]. Existen variados estudios que indican los costos asociados al tratamiento del Alzheimer. El 2010 se publicó en Francia un estudio que indica que los costos de tratamiento de Alzheimer varían entre los 14,3 millones de pesos y 26,2 millones de pesos, dependiendo de la gravedad de la condición [93]. Por otro lado, en China se hizo un estudio el 2015 que establece que el gasto anual por paciente es de aproximadamente 14 millones de pesos [94]. Finalmente, un estudio realizado por la doctora Andrea Slachevsky (junto a otros investigadores) indica que el costo promedio de tratamiento de Alzheimer en Chile al mes varía entre 511.072 pesos para los estratos socioeconómicos altos y 749.720 pesos para los estratos socioeconómicos bajos, quedando así un rango entre 6,1 millones de pesos y 9 millones de pesos al año [95].

- La Esclerosis lateral amiotrófica (ELA) presenta 1,75 casos a nivel mundial [96] y 2 casos en Chile [97]. Un estudio realizado entre los años 2015 y 2018 indicó que el costo anual por paciente varía entre 13.667 dólares en Dinamarca y 69.475 en Estados Unidos, equivalente a 10 millones de pesos y 51 millones de pesos, respectivamente [98].

- Para el caso de TEC, los valores mundiales y naciones fueron enunciados anteriormente y corresponden a 939 casos y 200 casos, respectivamente. De acuerdo con Tuominen et. al. (2012), los costos asociados al tratamiento de TEC en Finlandia son en total 17,5 millones de pesos, considerando cirugías, tratamientos y otros gastos médicos [99].

- A nivel mundial se tiene una incidencia de 10,82 casos de tumor cerebral [100] mientras que para Chile se tienen 3,5 casos [101]. El 2010 en Europa se estimaba que el costo total, directo e indirecto de un tumor cerebral era de 21.590 euros, lo que equivale hoy en día a 19,2 millones de pesos [102].

- En epilepsia se tiene una incidencia global de 61,44 casos [103] y en Chile se tienen 114 casos [104]. Según un estudio, en Estados Unidos los costos directos de un tratamiento de epilepsia en niños varían entre los 3,2 millones de pesos si las convulsiones son leves, a 7,5 millones de pesos si las convulsiones son graves, mientras que para adultos en esas mismas condiciones los costos varían entre 3,6 millones de pesos a 6,3 millones de pesos. Por otro lado, los costos indirectos para niños varían entre los 15,1 millones de

pesos y 30 millones de pesos, y en adultos entre 10,3 millones de pesos y 20,9 millones de pesos [105].

- La esclerosis múltiple cuenta con 3,6 casos en el mundo [106], mientras que en Chile la incidencia es de 0,9 casos [107]. Al igual que en varias otras condiciones, los costos asociados a esta enfermedad en Europa varían según la gravedad de los pacientes, en donde para casos leves se tiene un costo aproximado de 13,8 millones de pesos, para moderados 15,5 millones de pesos y graves 13,3 millones de pesos. Los montos mencionados corresponden al gasto directo, y si se considera la totalidad del gasto este asciende a 20,4 millones, 33,2 millones y 51,4 millones para pacientes leves, moderados y severos, respectivamente [108].

- Para el caso del Desorden depresivo mayor se tienen 3.000 casos a nivel mundial [109] y en Chile se tienen 5.700 casos [110]. En Estados Unidos, para el año 2017 se estimaba que los costos anuales por paciente con Desorden depresivo mayor eran en promedio 8,9 millones de pesos, y se establecía el máximo en 9,9 millones, estimando así un posible mínimo de tratamiento entre 7,5 y 8 millones de pesos [111].

- La incidencia para la Esquizofrenia es de 15,2 casos en el mundo [112] y en Chile es de 12 casos [113]. Al año 2016 se estimaba que los costos (directos e indirectos) anuales para pacientes que sufren de esquizofrenia en distintos países variaban desde los 4,3 millones de pesos (Tailandia) hasta los 70 millones de pesos (Noruega) [114].

- Finalmente, para el caso de VIH se tienen 1400 casos en el mundo [115], mientras que en Chile se tienen 50 casos [116]. Los costos anuales de la terapia Antirretroviral con la cual se trata el VIH en Estados Unidos variaban entre los 26,5 y 35,3 millones de pesos aproximadamente por paciente el año 2018 [117].

Considerando que la población mundial a diciembre del 2020 es de 7.800.000.000 personas aproximadamente [118], y la población chilena es de 19.500.000 personas aproximadamente [25], en la tabla 6.1 se calcula la cantidad de casos anuales para las distintas enfermedades, además del costo asociado al tratamiento de cada enfermedad.

Tabla 6.1: Incidencias de distintas enfermedades que pueden provocar una Disfunción Ejecutiva.

Enfermedad	**Incidencia Mundial**	**Incidencia nacional**	**Costo promedio CLP**
Enfermedad de Parkinson	2.803.200	2.340	4.250.000
Enfermedad de Alzheimer	30.660.000	144.690	13.900.000
Esclerosis lateral amiotrófica	127.750	390	30.500.000
Traumatismo craneoencefálico	68.547.000	39.000	17.500.000
Tumor cerebral	789.860	682,5	19.200.000
Epilepsia	4.485.120	22.230	24.225.000
Esclerosis múltiple	262.800	175,5	35.000.000
Desorden depresivo mayor	219.000.000	1.111.500	8.900.000
Esquizofrenia	1.109.600	2.340	37.150.000
VIH	102.200.000	5.000	30.900.000

En la tabla 6.1 se puede apreciar la cantidad de personas que sufren las condiciones mencionadas anteriormente junto al costo promedio de dicha enfermedad. Como ya se ha mencionado, se muestra el costo del tratamiento de las enfermedades simplemente para enunciar el hecho de que los tratamientos son bastante costosos, y si bien dichos costos no necesariamente son los que existen en Chile, a nivel mundial si es un gasto considerable. También es importante mencionar que dependiendo del país, dichos costos no son pagados en su totalidad por los pacientes, ya que puede haber ayudas estatales, seguros, distintos tipos de coberturas, etc, pero de cualquier manera alguna institución o persona debe costear los tratamientos de las enfermedades, por lo que el costo sigue existiendo aunque no lo pague el paciente en su totalidad.

Debido a que se escapa del marco de la investigación, se realizan algunos supuestos para dimensionar el beneficio económico y social que podría resultar de implementar un sistema de evaluación como el que se desarrolla en el proyecto Neuronat, el cuál usa como base los resultados del presente trabajo de título, en vez de hacer un análisis más exhaustivo sobre los costos y porcentajes asociados al padecimiento de Disfunción Ejecutiva en otros países y para otras enfermedades. En primer lugar, como se mencionó anteriormente, un 10 % de los pacientes de cada enfermedad desarrolla una Disfunción Ejecutiva. Esto se puede apreciar en la columna "DE Mundial" y "DE Nacional" de la tabla 6.2. Luego, se asume que un 1 % de las personas con DE, no padecen de esta condición, es decir, están mal diagnosticadas. Finalmente, con esa cantidad de pacientes mal diagnosticados y el costo de tratamiento de Disfunción Ejecutiva, que según el equipo clínico del proyecto Neuronat tiene un costo aproximado de 3 millones de pesos, se calcula el costo total que pagan las personas por un tratamiento que no necesitan. Dicho costo es de 1.290 billones de pesos por año a nivel

mundial, y 4 billones de pesos al año en Chile.

Tabla 6.2: Estimación de casos de pacientes con Disfunción Ejecutiva en Chile y en el mundo

Enfermedad	**DE Mundial**	**DE nacional**	**Mal diagnosticados mundialmente**	**Mal diagnosticados nacionalmente**
Enfermedad de Parkinson	280.320	234	2.803,2	2,34
Enfermedad de Alzheimer	3.066.000	14.469	30.660	144,69
Esclerosis lateral amiotrófica	12.775	39	127,75	0,39
Traumatismo craneoencefálico	6.854.700	3.900	68.547	39
Tumor cerebral	78.986	68,25	789,86	0,6825
Epilepsia	448.512	2.223	4.485,12	22,23
Esclerosis múltiple	26.280	17,55	262,8	0,1755
Desorden depresivo mayor	21.900.000	111.150	219.000	1.111,5
Esquizofrenia	110.960	234	1.109,6	2,34
VIH	10.220.000	500	102.200	5
Total	**42.998.533**	**132.834,8**	**429.985,33**	**1.328,348**

Otro análisis que se puede realizar con los datos de la tabla 6.2, es simplemente enunciar la cantidad de pacientes a los cuales se les puede aplicar el test Neuronat y a los cuales se les podría detectar tempranamente el padecimiento de Disfunción Ejecutiva, generando así tratamientos tempranos, y reduciendo o minimizando los daños propios de la DE en conjunto con la enfermedad padecida. Esta cantidad de personas es de 42.998.533 a nivel mundial y 132.835 a nivel nacional, y eso considerando que sólo un 10 % de la población enunciada desarrolla una DE. No se realiza un análisis económico atribuible a un tratamiento temprano, debido a la cantidad de enfermedades distintas y sus distintas evoluciones y características.

Por lo tanto, se puede estimar que los resultados obtenidos del presente trabajo de título en conjunto con lo desarrollado en el proyecto Neuronat, puede llegar a afectar a más de 130 mil personas al año en Chile, obteniendo diagnósticos y tratamientos tempranos para estas personas, disminuyendo así el daño sufrido y los tiempos de tratamiento. Es importante recordar que el daño sufrido se ve reflejado en la gran mayoría de las actividades cotidianas de las personas, como relacionarse con otros, recordar tareas y actividades, tener la capacidad de planificar, entre otros. El hecho de que el diagnóstico sea tardío y acentúe los daños implica que los pacientes padecerán dichos problemas por más tiempo, lo que les quita independencia por más tiempo, y podría hacer que sus gastos sean mayores, ya que al perder la independencia necesariamente requieren de un tercero que los ayude en algunas actividades, lo cual podría tener un costo como contratar a un enfermero o kinesiólogo por ejemplo. Además, al no tener las capacidades suficientes para poder trabajar (por las consecuencias antes mencionadas) los pacientes pueden llegar a sufrir una disminución de sus ingresos ya sea porque tienen que

hacer uso de sus licencias médicas que no necesariamente son equivalentes a un sueldo, o en el peor de los casos pueden perder sus trabajos.

Por otro lado, considerando que un 1 % de las personas diagnosticadas con DE están erróneamente diagnosticadas, se tiene un costo de 4 billones de pesos al año en Chile, los cuales deben ser pagados por los pacientes injustificadamente (ya que no padecen realmente de Disfunción Ejecutiva). Si con los resultados obtenidos del presente documento en conjunto con lo desarrollado en el proyecto Neuronat se logra reducir en un 20 % la cantidad de diagnósticos erróneos se tendría un 20 % de ahorro en los gastos asociados al tratamiento de Disfunción Ejecutiva por parte de los pacientes, que corresponde a un ahorro anual de 800 millones de pesos. Finalmente, si el proyecto Neuronat logra ser aplicado a nivel mundial, se afectaría a casi 43 millones de personas, reduciendo los tiempos de tratamiento de estas y mejorando así su calidad de vida, además de ahorrar 258 billones de pesos al año considerando una disminución del 20 % de diagnósticos erróneos.

Capítulo 7

Conclusión

En el presente capítulo se concluye el trabajo realizado, indicando los principales resultados y si se cumple la hipótesis de investigación. Finalmente se expone las limitaciones que se tuvieron durante el desarrollo del trabajo, y se proponen sugerencias acerca del trabajo futuro para el proyecto Neuronat.

7.1. Conclusiones

En el capítulo 1 del presente documento se plantearon 5 objetivos específicos para poder cumplir el objetivo general que es *Seleccionar el conjunto de variables psicosociales que permite caracterizar el desempeño de funciones ejecutivas en situaciones cotidianas*. Dichos objetivos se cumplieron en un 100 %, y a continuación se explica brevemente lo obtenido:

- Del análisis del estado del arte se logró llegar a la conclusión de que si bien hay nuevas maneras de evaluar y diagnosticar Disfunciones Ejecutivas, estas no son aplicables en todos los contextos. Además, se encontró que existen varios problemas similares en el mundo de la salud mental que son abordados desde el área de la ciencia de los datos, sin embargo, no se tiene registro de algún caso real o de estudio en donde se apliquen técnicas de Machine Learning para el diagnóstico de Disfunciones Ejecutivas.

- Al generar la base de datos que almacena los registros obtenidos de la plataforma Neuronat, y al trabajar con esos datos para obtener una única tabla, se logró apreciar que con la forma en la que se exportan los datos de la plataforma, y el contenido de estos, no es posible construir todas las métricas que desarrollaron los expertos del proyecto Neuronat. Esto debido a que hay datos faltantes para poder generar dichas métricas como en el caso de los datos de la mesa 2, que no exporta ningún dato.

- Del estudio de las variables realizado se obtiene principalmente que las variables más importantes son aquellas que reflejan la cantidad de errores que comete el jugador en la plataforma, junto con los tiempos de ejecución de actividades.

- El modelo de la base manual que obtuvo mejores resultados fue *Random Forest Backward Unique*, con un 100 % de Accuracy, Weighted Recall y Weighted F1-Score, y un 1 en AUC en la muestra de entrenamiento, y un 67 % de Accuracy, Weighted Recall y Weighted F1-Score en la muestra de test, y un 0.77 de AUC en esa misma muestra, lo cual son resultados considerablemente mejor al caso base considerado en donde se

obtiene las clases de manera aleatoria (25 % por clase aproximadamente). Las métricas de los modelos con mejor desempeño se pueden ver en las tablas 5.8 y C.8 del anexo.

- Finalmente, de la evaluación de impacto social y económico del Capítulo 6, se logra estimar una gran cantidad de potenciales pacientes que podrían beneficiarse del proyecto Neuronat en caso de tener buenos resultados (los cuáles podrían ser obtenidos tomando como base lo desarrollado en el presente trabajo) llegando a 130 mil personas en Chile y más de 40 millones en el mundo al año. También se estima un ahorro para el sistema en costos asociados al tratamiento de Disfunciones Ejecutivas, llegando a un ahorro anual de 800 millones de pesos en Chile y 258 billones de pesos en el mundo. Por lo tanto, se puede concluir que de ser exitoso el proyecto, y que pueda diagnosticar tempranamente una Disfunción Ejecutiva, se beneficiaría una gran cantidad de personas e instituciones en Chile y en el mundo.

Por lo tanto, al cumplir los 5 objetivos específicos planteados se puede afirmar que se cumple el objetivo general del trabajo de título, lo cual se puede ver representado en las tablas 5.8 y C.8. Sin embargo, el presente trabajo es también un trabajo de investigación, lo que implica que existe una hipótesis de investigación que es "*Con las variables que produce la plataforma Neuronat se debe poder estimar el desempeño de las funciones ejecutivas, reflejadas a través de diferencias en los resultados de segmentos etarios distintos*". Si bien los resultados obtenidos de los modelos entrenados son positivos en el sentido de que efectivamente se pueden detectar patrones en los datos que permitan diferenciar a los distintos segmentos etarios, los hechos mencionados en el apartado de Discusión del capítulo 5 hacen que se rechace la hipótesis de investigación no por los resultados, sino que por la calidad de los datos, ya que no se puede asegurar en un 100 % que los datos obtenidos estén correctos por la manera en que estos se obtuvieron (de manera remota). A pesar de lo anterior, si existen indicios de que con los datos obtenidos del videojuego se puede diferenciar a distintos grupos etarios sin la presencia del *sesgo etario* mencionado en la Discusión, por lo que se cree posible aprobar la hipótesis de investigación, pero en una próxima etapa en donde haya más control en la toma de datos.

7.2. Trabajo futuro

Según el trabajo realizado y las conclusiones obtenidas, se proponen los siguientes puntos para el trabajo futuro del proyecto Neuronat:

- En primer lugar y más importante, **tomar datos utilizando la plataforma en un ambiente controlado**, en donde todos los sujetos de prueba se encuentren en un mismo contexto, con las variables control constantes entre todos los jugadores, para que de esta manera sólo se obtenga el efecto deseado en los datos. Para el caso del trabajo de título se trabajó con una etiqueta de segmento etario, sin embargo el ideal es trabajar con pacientes reales en donde la etiqueta que se entrega a los modelos es una etiqueta binaria de si es paciente o es control.

- Luego, se sugiere replicar los análisis realizados una vez la plataforma esté más avanzada en su desarrollo, ya que para el trabajo realizado se utilizó la plataforma en su versión más básica, lo que también puede implicar datos de mala calidad, ya sea por cómo se

generan o por el trasfondo contextual de los jugadores en donde la plataforma puede causar confusión. Además, hay ciertos datos que pueden ser relevantes asociados a actividades específicas del juego que no se están exportando, por lo que se pierden. En particular todos los datos asociados a la segunda mesa del juego, datos sobre ciertos errores específicos como la entrega de orden en mesas distintas o en el menú general.

- Finalmente, se sugiere revisar las métricas generadas por el equipo clínico en conjunto con el WIC, ya que varias de estas son redundantes y terminan siendo eliminadas por el proceso de Feature Selection. Por ejemplo, al ver las variables de los mejores modelos presentados en la tabla 5.8 de anexos, se puede apreciar que la gran mayoría están relacionadas a las mesas 1 y 3, ya que se da el caso en que las mesas 5 y 4 son prácticamente idénticas a las 1 y 3 respectivamente, por lo tanto al aplicar filtros como el de correlación, en donde se eliminan variables altamente correlacionadas, se eliminan varias variables de las mesas 5 y 4 (ya que esas variables se analizan después que las variables de las mesas 1 y 3). Lo que se podría hacer para este caso es reorganizar las variables para orientarlas a actividades o tipo de actividad más que a variables por mesa, de esta manera, una misma actividad de la mesa 3 y 4 se puede agrupar en una sola variable, capturando así el efecto global de dicha actividad.

Bibliografía

[1] Dewan, M. C., Rattani, A., Gupta, S., Baticulon, R. E., Hung, Y., Punchak, M., Agrawal, A., Adeleye, A. O., Shrime, M. G., Rubiano, A. M., Rosenfeld, J. V., Park, K. B. (2018). Estimating the global incidence of traumatic brain injury, Journal of Neurosurgery JNS, 130(4), 1080-1097. Retrieved Jun 16, 2020, from https://thejns.org/view/journals/j-neurosurg/130/4/article-p1080.xml

[2] Vayas Abascal, Rocío, Carrera Romero, Luis. (2012). Disfunción Ejecutiva: Síntomas y relevancia de su detección desde Atención Primaria. Revista Clínica de Medicina de Familia, 5(3), 191-197. https://dx.doi.org/10.4321/S1699-695X2012000300007

[3] Pineda, D. (2000). La función ejecutiva y sus trastornos. Revista de neurología, 30(8), 764-768.

[4] Rabinovici, G. D., Stephens, M. L., Possin, K. L. (2015). Executive dysfunction. CONTINUUM: Lifelong Learning in Neurology, 21(3 Behavioral Neurology and Neuropsychiatry), 646.

[5] Schmuckler, M. A. (2001). What is ecological validity? A dimensional analysis. Infancy, 2(4), 419-436.

[6] McDonald, B. C., Flashman, L. A., Saykin, A. J. (2002). Executive dysfunction following traumatic brain injury: neural substrates and treatment strategies. NeuroRehabilitation, 17(4), 333-344.

[7] Sajda, P. (2006). Machine Learning for detection and diagnosis of disease. Annu. Rev. Biomed. Eng., 8, 537-565.

[8] Khedher, L., Ramírez, J., Górriz, J. M., Brahim, A., Segovia, F., Alzheimer s Disease Neuroimaging Initiative. (2015). Early diagnosis of Alzheimer s disease based on partial least squares, principal component analysis and support vector machine using segmented MRI images. Neurocomputing, 151, 139-150.

[9] R. S. McGinnis et al., Rapid Anxiety and Depression Diagnosis in Young Children Enabled by Wearable Sensors and Machine Learning,"2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), Honolulu, HI, 2018, pp. 3983-3986, doi: 10.1109/EMBC.2018.8513327.

[10] Web Intelligence Centre. (s. f.). Home. Recuperado 6 de junio de 2020, de https://wic.uchile.cl/

[11] Facultad de Medicina Universidad de Chile. (s. f.). Facultad - Facultad de Medicina - Universidad de Chile. Recuperado 6 de junio de 2020, de http://www.medicina.uchile.cl/facultad

[12] Gobierno de México. (s. f.). Salud Mental y Discapacidad psicosocial. Recuperado 6 de junio de 2020, de https://www.gob.mx/conadis/articulos/salud-mental-y-discapacidad-psicosocial#:%7E:text=Salud%20Mental%20y%20Discapacidad%20psicosocial,Discapacidad%20%7C%20Gobierno%20%7C%20gob.mx

[13] Tecnológico de Monterrey. (2018, 28 septiembre). ¿Qué son los Serious Games? — Observatorio de Innovación Educativa. Recuperado 6 de junio de 2020, de https://observatorio.tec.mx/edu-news/que-son-los-serious-games

[14] Newman, R. (2013). Cinematic game secrets for creative directors and producers: inspired techniques from industry legends. Taylor Francis.

[15] Cimino, C. R. (2000). Principles of neuropsychological interpretation. Clinician's guide to neuropsychological assessment, 69-109.

[16] Superintendencia de salud - Gobierno de Chile. (s. f.). ¿Qué es una licencia médica? Superintendencia de salud. Recuperado 21 de julio de 2020, de http://www.supersalud.gob.cl/consultas/667/w3-article-4550.html

[17] Superintendencia de Seguridad Social - Gobierno de Chile. (s. f.). Subsidios por incapacidad laboral de origen común. SUSESO. Recuperado 21 de julio de 2020, de https://www.suseso.cl/606/w3-propertyvalue-568.htmlpresentacion

[18] Gioia, G. A., Isquith, P. K. (2004). Ecological assessment of executive function in traumatic brain injury. Developmental neuropsychology, 25(1-2), 135-158.

[19] Universidad Internacional de Valencia. (2018, 21 marzo). ¿Qué es la neuropsicología y qué hace un neuropsicólogo? | VIU. https://www.universidadviu.com/int/actualidad/nuestros-expertos/que-es-la-neuropsicologia-y-que-hace-un-neuropsicologo

[20] Rand, D., Katz, N., Shahar, M., Kizony, R., Weiss, P. L. (2005). The virtual mall: A functional virtual environment for stroke rehabilitation. Annual Review of Cybertherapy and Telemedicine: A decade of VR, 3, 193-198.

[21] McAlister, C., Schmitter-Edgecombe, M. (2013). Naturalistic assessment of executive function and everyday multitasking in healthy older adults. Aging, Neuropsychology, and Cognition, 20(6), 735-756.

[22] Rabinowitz, A. R., Levin, H. S. (2014). Cognitive sequelae of traumatic brain injury. The Psychiatric Clinics of North America, 37(1), 1.

[23] Leo P, McCrea M. Epidemiology. In: Laskowitz D, Grant G, editors. Translational Research in Traumatic Brain Injury. Boca Raton (FL): CRC Press/Taylor and Francis Group; 2016. Chapter 1. Available from: https://www.ncbi.nlm.nih.gov/books/NBK326730/

[24] Facultad de Medicina - Universidad de Chile. (2016). TEC. Recuperado 16 de junio de 2020, de https://sintesis.med.uchile.cl/index.php/component/content/article/101-revision/r-de-urgencias/1900-tec

[25] Instituto Nacional de Estadísticas. (s. f.). Inicio. Inicio. Recuperado 23 de julio de 2020, de https://www.ine.cl/

[26] Boudreaux, K. (2003). Finanzas.

[27] Ministerio de Desarrollo Social - Gobierno de Chile Ministerio de Salud - Gobierno de Chile. (2015, febrero). Guía para la presentación de proyectos: Servicios de Atención Primaria de Urgencia de Alta Resolución (SAR). http://sni.ministeriodesarrollosocial.gob.cl/download/guia-para-la-presentacion-de-proyectos-servicios-de-atencion-primaria-de-urgencia-de-alta-resolucion-sar/?wpdmdl=915

[28] Superintendencia de Seguridad Social (SUSESO) - Gobierno de Chile. (2019, noviembre). Estadísticas Nacionales de LM y SIL 2018. Régimen de Licencias médicas y Subsidios por incapacidad laboral (SIL). https://www.suseso.cl/608/w3-article-580746.html

[29] Buckland, M., Gey, F. (1994). The relationship between recall and precision. Journal of the American society for information science, 45(1), 12-19.

[30] Rand, D., Rukan, S. B. A., Weiss, P. L., Katz, N. (2009). Validation of the Virtual MET as an assessment tool for executive functions. Neuropsychological rehabilitation, 19(4), 583-602.

[31] Williams, G. J., Huang, Z. (1996, October). A case study in knowledge acquisition for insurance risk assessment using a KDD methodology. In Proceedings of the Pacific Rim Knowledge Acquisition Workshop, Dept. of AI, Univ. of NSW, Sydney, Australia (pp. 117-129).

[32] M. J. Fonseca and J. A. Jorge, Ïndexing high-dimensional data for content-based retrieval in large databases,.[Ei]ghth International Conference on Database Systems for Advanced Applications, 2003. (DASFAA 2003). Proceedings., Kyoto, Japan, 2003, pp. 267-274, doi: 10.1109/DASFAA.2003.1192391

[33] Fayyad, U. M., Haussler, D., Stolorz, P. E. (1996, August). KDD for Science Data Analysis: Issues and Examples. In KDD (pp. 50-56).

[34] Biswas, T., Saha, M., Bhattacharyya, S. (2016). An Approach to Implement Data Mining in Service Oriented Methodology towards Amelioration of Society. International Journal of Innovations in Engineering and Technology (IJIET), 6(3).

[35] Ramakrishnan, R., Gehrke, J. (2000). Database management systems. McGraw-Hill.

[36] We are social. (2020, enero). Digital 2020. https://wearesocial.com/digital-2020

[37] Silberschatz, A., Korth, H. F., Sudarshan, S. (1997). Database system concepts (Vol. 5). New York: McGraw-Hill.

[38] Chen, P. P. S. (1976). The entity-relationship model—toward a unified view of data. ACM transactions on database systems (TODS), 1(1), 9-36.

[39] Teorey, T. J., Lightstone, S. S., Nadeau, T., Jagadish, H. V. (2011). Database modeling and design: logical design. Elsevier.

[40] Python Software Foundation. (n.d.). The Python Tutorial — Python 3.9.2 documentation. The Python Tutorial. Retrieved February 24, 2021, from https://docs.python.org/3/tutorial/index.htmltutorial-index

[41] Project Jupyter. (n.d.). Project Jupyter. About Us. Retrieved February 24, 2021, from https://jupyter.org/about

[42] Jupyter Project. (n.d.). The Jupyter Notebook — Jupyter Notebook 6.2.0 documentation. The Jupyter Notebook. Retrieved February 24, 2021, from https://jupyter-notebook.readthedocs.io/en/stable/notebook.html

[43] Holovaty, A., Kaplan-Moss, J. (2009). The definitive guide to Django: Web development done right. Apress.

[44] Equipo Vértice. (2009). Diseño básico de páginas web en HTML. Editorial Vértice.

[45] Scharl, A. (2012). Evolutionary web development. Springer Science Business Media.

[46] Robillard, M. P. (2009). What makes APIs hard to learn? Answers from developers. IEEE software, 26(6), 27-34.

[47] R. (2019, 25 diciembre). What is REST. REST API Tutorial. https://restfulapi.net/

[48] Vainikka, J. (2018). Full-stack web development using Django REST framework and React.

[49] Masse, M. (2011). REST API Design Rulebook: Designing Consistent RESTful Web Service Interfaces. .ºReilly Media, Inc.".

[50] Cai, J., Luo, J., Wang, S., Yang, S. (2018). Feature Selection in Machine Learning: A new perspective. Neurocomputing, 300, 70-79.

[51] Langley, P. (1994, November). Selection of relevant features in Machine Learning. In Proceedings of the AAAI Fall symposium on relevance (Vol. 184, pp. 245-271).

[52] Guyon, I., Elisseeff, A. (2003). An introduction to variable and Feature Selection. Journal of Machine Learning research, 3(Mar), 1157-1182.

[53] Xu, R., Wunsch, D. (2008). Clustering (Vol. 10). John Wiley Sons.

[54] QuantDare. (2019, 3 diciembre). What is the difference between feature extraction and Feature Selection? https://quantdare.com/what-is-the-difference-between-feature-extraction-and-feature-selection/

[55] Garla, V. N., Brandt, C. (2012). Ontology-guided feature engineering for clinical text classification. Journal of biomedical informatics, 45(5), 992-998.

[56] Yu, H. F., Lo, H. Y., Hsieh, H. P., Lou, J. K., McKenzie, T. G., Chou, J. W., ... Weng, J. Y. (2010). Feature engineering and classifier ensemble for KDD cup 2010. KDD cup, 11.

[57] A. Fatima, F. Ali Khan, A. Raza and A. Basit Kamran, .ªutomated Feature Synthesis from Relational Database for Data Science Related Problems,"2018 International Conference on Frontiers of Information Technology (FIT), Islamabad, Pakistan, 2018, pp. 71-75, doi: 10.1109/FIT.2018.00020.

[58] Lam, H. T., Thiebaut, J. M., Sinn, M., Chen, B., Mai, T., Alkan, O. (2017). One button machine for automating feature engineering in relational databases. arXiv preprint arXiv:1706.00327.

[59] J. M. Kanter and K. Veeramachaneni, "Deep feature synthesis: Towards automating Data Science endeavors,"2015 IEEE International Conference on Data Science and Advanced Analytics (DSAA), Paris, 2015, pp. 1-10, doi: 10.1109/DSAA.2015.7344858.

[60] Chen, Z. S., Zhu, B., He, Y. L., Yu, L. A. (2017). A PSO based virtual sample generation

method for small sample sets: Applications to regression datasets. Engineering Applications of Artificial Intelligence, 59, 236-243.

[61] Wedyan, M., Crippa, A., Al-Jumaily, A. (2019). A Novel Virtual Sample Generation Method to Overcome the Small Sample Size Problem in Computer Aided Medical Diagnosing. Algorithms, 12(8), 160.

[62] Xie, Y., Li, M. (2010, January). Application of gray forecasting model optimized by genetic algorithm in electricity demand forecasting. In 2010 Second International Conference on Computer Modeling and Simulation (Vol. 4, pp. 275-277). IEEE.

[63] Murphy, K. P. (2012). Machine Learning: a probabilistic perspective. MIT press.

[64] Ayodele, T. O. (2010). Types of Machine Learning algorithms. New advances in Machine Learning, 3, 19-48.

[65] Rodriguez, J. D., Perez, A., Lozano, J. A. (2009). Sensitivity analysis of k-fold cross validation in prediction error estimation. IEEE transactions on pattern analysis and machine intelligence, 32(3), 569-575.

[66] Rish, I. (2001, August). An empirical study of the naive Bayes classifier. In IJCAI 2001 workshop on empirical methods in artificial intelligence (Vol. 3, No. 22, pp. 41-46).

[67] Feng, X., Li, S., Yuan, C. et al. Prediction of Slope Stability using Naive Bayes Classifier. KSCE J Civ Eng 22, 941–950 (2018). https://doi.org/10.1007/s12205-018-1337-3

[68] Noble, W. What is a support vector machine?. Nat Biotechnol 24, 1565–1567 (2006). https://doi.org/10.1038/nbt1206-1565

[69] Cortes, C., Vapnik, V. (1995). Support-vector networks. Machine Learning, 20(3), 273-297.

[70] Safavian, S. R., Landgrebe, D. (1991). A survey of decision tree classifier methodology. IEEE transactions on systems, man, and cybernetics, 21(3), 660-674.

[71] Breiman, L. (2001). Random forests. Machine Learning, 45(1), 5-32.

[72] Cutler, D. R., Edwards Jr, T. C., Beard, K. H., Cutler, A., Hess, K. T., Gibson, J., Lawler, J. J. (2007). Random forests for classification in ecology. Ecology, 88(11), 2783-2792.

[73] Jain, A. K., Mao, J., Mohiuddin, K. M. (1996). Artificial neural networks: A tutorial. Computer, 29(3), 31-44.

[74] Kononenko, I. (2001). Machine Learning for medical diagnosis: history, state of the art and perspective. Artificial Intelligence in medicine, 23(1), 89-109.

[75] Chen, M., Hao, Y., Hwang, K., Wang, L., Wang, L. (2017). Disease prediction by Machine Learning over big data from healthcare communities. Ieee Access, 5, 8869-8879.

[76] Handelman, G. S., Kok, H. K., Chandra, R. V., Razavi, A. H., Huang, S., Brooks, M., ... Asadi, H. (2019). Peering into the black box of artificial intelligence: evaluation metrics of Machine Learning methods. American Journal of Roentgenology, 212(1), 38-43.

[77] Ciberwatch.es. (2015, February 26). El Test del Reloj puede diagnosticar problemas de Alzheimer. Ciberwatch.Es - Tienda de Relojes Online. https://www.ciberwatch.es/blog/noticias/el-test-del-reloj-puede-diagnosticar-problemas-

de-alzheimer/

[78] Berch, D. B., Krikorian, R., Huha, E. M. (1998). The Corsi block-tapping task: Methodological and theoretical considerations. Brain and cognition, 38(3), 317-338.

[79] The FAB A frontal assessment battery at bedside B. Dubois, A. Slachevsky, I. Litvan, B. Pillon Neurology Dec 2000, 55 (11) 1621-1626; DOI: 10.1212/WNL.55.11.1621

[80] Shatte ABR, Hutchinson DM, Teague SJ (2019). Machine Learning in mental health: a scoping review of methods and applications. Psychological Medicine 1–23. https://doi.org/10.1017/S0033291719000151

[81] He, Y. L., Wang, P. J., Zhang, M. Q., Zhu, Q. X., Xu, Y. (2018). A novel and effective nonlinear interpolation virtual sample generation method for enhancing energy prediction and analysis on small data problem: A case study of Ethylene industry. Energy, 147, 418-427.

[82] Scikit-Learn. (s. f.). 3.1. Cross-validation: evaluating estimator performance — Scikit-Learn 0.24.1 documentation. Recuperado 29 de enero de 2021, de https://Scikit-Learn.org/stable/modules/cross$_v$*alidation.html*

[83] Collette, F., Van der Linden, M., Salmon, E. (1999). Executive dysfunction in Alzheimer's disease. Cortex, 35(1), 57-72.

[84] Watermeyer, T.J., Brown, R.G., Sidle, K.C.L. et al. Executive dysfunction predicts social cognition impairment in amyotrophic lateral sclerosis. J Neurol 262, 1681–1690 (2015). https://doi.org/10.1007/s00415-015-7761-0

[85] Rzezak, P., Fuentes, D., Guimarães, C. A., Thome-Souza, S., Kuczynski, E., Guerreiro, M., Valente, K. D. (2009). Executive dysfunction in children and adolescents with temporal lobe epilepsy: is the Wisconsin Card Sorting Test enough?. Epilepsy Behavior, 15(3), 376-381.

[86] Savica R, Grossardt BR, Bower JH, Ahlskog JE, Rocca WA. Time Trends in the Incidence of Parkinson Disease. JAMA Neurol. 2016;73(8):981–989. doi:10.1001/jamaneurol.2016.0947

[87] Ministerio de Salud, Gobierno de Chile. (2017). INFORME DE EVALUACIÓN CIENTÍFICA BASADA EN LA EVIDENCIA DISPONIBLE, Condición de Salud: Enfermedad de Parkinson, Tecnología Sanitaria Evaluada: Estimulación cerebral profunda. https://www.minsal.cl/wp-content/uploads/2017/10/parkinson-OK.pdf

[88] Marin-Medina, Daniel Stiven, Quintero-Moreno, Juan Felipe, Valencia-Vásquez, Aníbal, Duque-Salazar, Catalina, Gil-Restrepo, Andrés Felipe, Castaño-Montoya, Juan Pablo, García-Rodríguez, Daniela, Carmona-Villada, Hans. (2018). Estimulación cerebral profunda en enfermedad de Parkinson. Iatreia, 31(3), 262-273. https://dx.doi.org/10.17533/udea.iatreia.v31n3a04

[89] S.D.N Chile. (2018, junio). Especial de Parkinson: La cirugía que hospitales y clínicas trabajan por hacer accesible. Sociedad de Neurocirugía de Chile. https://www.neurocirugiachile.org/noticias/especial-de-parkinson-la-cirugia-que-hospitales-y-clinicas-trabajan-por-hacer-accesible/

[90] Ponjoan, A., Garre-Olmo, J., Blanch, J., Fages, E., Alves-Cabratosa, L., Martí-Lluch, R., ... Ramos, R. (2020). Is it time to use real-world data from primary care in Alzheimer's disease?. Alzheimer's research therapy, 12, 1-9.

[91] Escuela de Medicina, Pontificia Universidad Católica de Chile. (2019, 14 agosto). Epidemiología, diagnóstico y pruebas cognitivas de demencias en APS. Escuela de Medicina. https://medicina.uc.cl/publicacion/epidemiologia-diagnostico-y-pruebas-cognitivas-de-demencias-en-aps/

[92] Albala, C. (2015). La situación de la demencia en Chile y el mundo. Congreso Nacional de Geriatría y Gerontología de Chile 2015, Santiago, Chile. http://compuerta.cl/geriatriacongreso2015/presentaciones/pdf/parque%201/3$_viernes$%2024/$pre_7.pdf$

[93] Rapp, T., Andrieu, S., Chartier, F., Deberdt, W., Reed, C., Belger, M., Vellas, B. (2018). Resource use and cost of alzheimer's disease in France: 18-month results from the GERAS observational study. Value in Health, 21(3), 295-303.

[94] Jia, J., Wei, C., Chen, S., Li, F., Tang, Y., Qin, W., ... Zhou, A. (2018). The cost of Alzheimer's disease in China and re-estimation of costs worldwide. Alzheimer's Dementia, 14(4), 483-491.

[95] Universidad de Chile. (2015, 27 agosto). Costos de tratamiento de demencia en Chile son mayores para los estratos más pobres. https://www.uchile.cl/noticias/114673/tratamiento-de-demencia-en-chile-es-mas-caro-para-estratos-mas-pobres

[96] Benoît Marin, Farid Boumédiene, Giancarlo Logroscino, Philippe Couratier, Marie-Claude Babron, Anne Louise Leutenegger, Massimilano Copetti, Pierre-Marie Preux, Ettore Beghi, Variation in worldwide incidence of amyotrophic lateral sclerosis: a meta-analysis, International Journal of Epidemiology, Volume 46, Issue 1, February 2017, Pages 57–74, https://doi.org/10.1093/ije/dyw061

[97] Ministerio de Salud, Gobierno de Chile. (2017b). INFORME DE EVALUACIÓN CIENTÍFICA BASADA EN LA EVIDENCIA DISPONIBLE Condición de Salud: Esclerosis Lateral Amiotrófica Tecnología Sanitaria Evaluada: Apoyo profesional, ayudas técnicas y alimentación entera. https://www.minsal.cl/wp-content/uploads/2017/10/ELA.pdf

[98] Song, H., Liu, J. C., Cao, Z. P., Luo, W. J., Chen, J. Y. (2020). Medical cost and healthcare utilization of amyotrophic lateral sclerosis in China: A cohort study based on hospital data from 2015 to 2018. Medicine, 99(47), e23258. https://doi.org/10.1097/MD.0000000000023258

[99] Tuominen, R., Joelsson, P., Tenovuo, O. (2012). Treatment costs and productivity losses caused by traumatic brain injuries. Brain Injury, 26(13-14), 1697-1701.

[100] Paula de Robles, Kirsten M. Fiest, Alexandra D. Frolkis, Tamara Pringsheim, Callie Atta, Christine St. Germaine-Smith, Lundy Day, Darren Lam, Nathalie Jette, The worldwide incidence and prevalence of primary brain tumors: a systematic review and meta-analysis, Neuro-Oncology, Volume 17, Issue 6, June 2015, Pages 776–783, https://doi.org/10.1093/neuonc/nou283

[101] Ministerio de salud, Gobierno de Chile. (s. f.). Descripción y Epidemiología: Tumores primarios del sistema nervioso central en personas de 15 años o más. CuidémonosEntreTodos. Recuperado 13 de diciembre de 2020, de https://diprece.minsal.cl/leinformamos/auge/acceso-guias-clinicas/guias-clinicas-desarrolladas-utilizando-manual-metodologico/tumores-primarios-del-sistema-nervioso-central-en-personas-de-15-anos-o-mas/descripcion-y-epidemiologia/

[102] Olesen, J., Gustavsson, A., Svensson, M., Wittchen, H. U., Jönsson, B., CDBE2010 Study

Group, European Brain Council. (2012). The economic cost of brain disorders in Europe. European journal of neurology, 19(1), 155-162.

[103] Prevalence and incidence of epilepsy A systematic review and meta-analysis of international studies Kirsten M. Fiest, Khara M. Sauro, Samuel Wiebe, Scott B. Patten, Churl-Su Kwon, Jonathan Dykeman, Tamara Pringsheim, Diane L. Lorenzetti, Nathalie Jetté Neurology Jan 2017, 88 (3) 296-303; DOI: 10.1212/WNL.0000000000003509

[104] Ministerio de Salud, Gobierno de Chile. (2014, julio). GUÍA CLÍNICA AUGE EPILEPSIA ADULTOS. https://www.minsal.cl/sites/default/files/files/GUIA%20CLINICA$_EPILEPSIA$%20$ADULTOS_web.p$

[105] Hussain, S. A., Ortendahl, J. D., Bentley, T. G., Harmon, A. L., Gupta, S., Begley, C. E., ... Knoth, R. L. (2020). The economic burden of caregiving in epilepsy: An estimate based on a survey of US caregivers. Epilepsia, 61(2), 319-329.

[106] Temporal trends in the incidence of multiple sclerosis A systematic review Alvaro Alonso, Miguel A. Hernán Neurology Jul 2008, 71 (2) 129-135; DOI: 10.1212/01.wnl.0000316802.35974.34

[107] Ministerio de Salud, Gobierno de Chile. (2019, enero). Protocolo 2019: Tratamiento de Segunda Línea Basado en Fingolimod o Natalizumab o Alemtuzumab o Cladribina u Ocrelizumab para personas con Esclerosis Múltiple Recurrente Remitente con falla a tratamiento con inmunomoduladores y Tratamiento con Ocrelizumab para personas con Esclerosis Múltiple Primaria Progresiva. https://www.minsal.cl/wp-content/uploads/2019/03/04$_protocolo-Esclerosis-M\%C3\%BAltiple.pdf$

[108] Kobelt, G., Thompson, A., Berg, J., Gannedahl, M., Eriksson, J. (2017). New insights into the burden and costs of multiple sclerosis in Europe. Multiple Sclerosis Journal, 23(8), 1123–1136. https://doi.org/10.1177/1352458517694432

[109] Ferrari, A. J., Somerville, A. J., Baxter, A. J., Norman, R., Patten, S. B., Vos, T., Whiteford, H. A. (2013). Global variation in the prevalence and incidence of major depressive disorder: a systematic review of the epidemiological literature. Psychological medicine, 43(3), 471.

[110] Salvo, L. (2014). Magnitud, impacto y estrategias de enfrentamiento de la depresión, con referencia a Chile. Revista médica de Chile, 142(9), 1157-1164.

[111] Gauthier, G., Guérin, A., Zhdanava, M. et al. Treatment patterns, healthcare resource utilization, and costs following first-line antidepressant treatment in major depressive disorder: a retrospective US claims database analysis. BMC Psychiatry 17, 222 (2017). https://doi.org/10.1186/s12888-017-1385-0

[112] John McGrath, Sukanta Saha, David Chant, Joy Welham, Schizophrenia: A Concise Overview of Incidence, Prevalence, and Mortality, Epidemiologic Reviews, Volume 30, Issue 1, 1 November 2008, Pages 67–76, https://doi.org/10.1093/epirev/mxn001

[113] Organización panamericana de la salud, Chile. (2007). Salud mental. Pan American Health Organization / World Health Organization. https://www.paho.org/chi/index.php?option=com$_contentview = articleid = 180 : salud-mentalItemid = 1005$

[114] Jin, H., Mosweu, I. (2017). The societal cost of schizophrenia: a systematic review.

Pharmacoeconomics, 35(1), 25-42.

[115] Hayes, R. J., Donnell, D., Floyd, S., Mandla, N., Bwalya, J., Sabapathy, K., ... Piwowar-Manning, E. (2019). Effect of universal testing and treatment on HIV incidence—HPTN 071 (PopART). New England Journal of Medicine, 381(3), 207-218.

[116] Ministerio de Salud, Gobierno de Chile. (s. f.). Descripción y Epidemiología: VIH/SIDA. CuidémonosEntreTodos. Recuperado 13 de diciembre de 2020, de https://diprece.minsal.cl/le-informamos/auge/acceso-guias-clinicas/guias-clinicas-desarrolladas-utilizando-manual-metodologico/sindrome-de-la-inmunodeficiencia-adquirida-vih-sida/descripcion-y-epidemiologia/

[117] McCann, N. C., Horn, T. H., Hyle, E. P., Walensky, R. P. (2020). HIV antiretroviral therapy costs in the United States, 2012-2018. JAMA Internal Medicine, 180(4), 601-603.

[118] Worldometer. (s. f.). World Population Clock: 7.8 Billion People (2020) - Worldometer. Recuperado 13 de diciembre de 2020, de https://www.worldometers.info/world-population/:%7E:text=7.8%20Billion%20(2020),Nations%20estimates%20elaborated%20by%20Worl

Anexo A

Figuras

Potenciales causas de DE

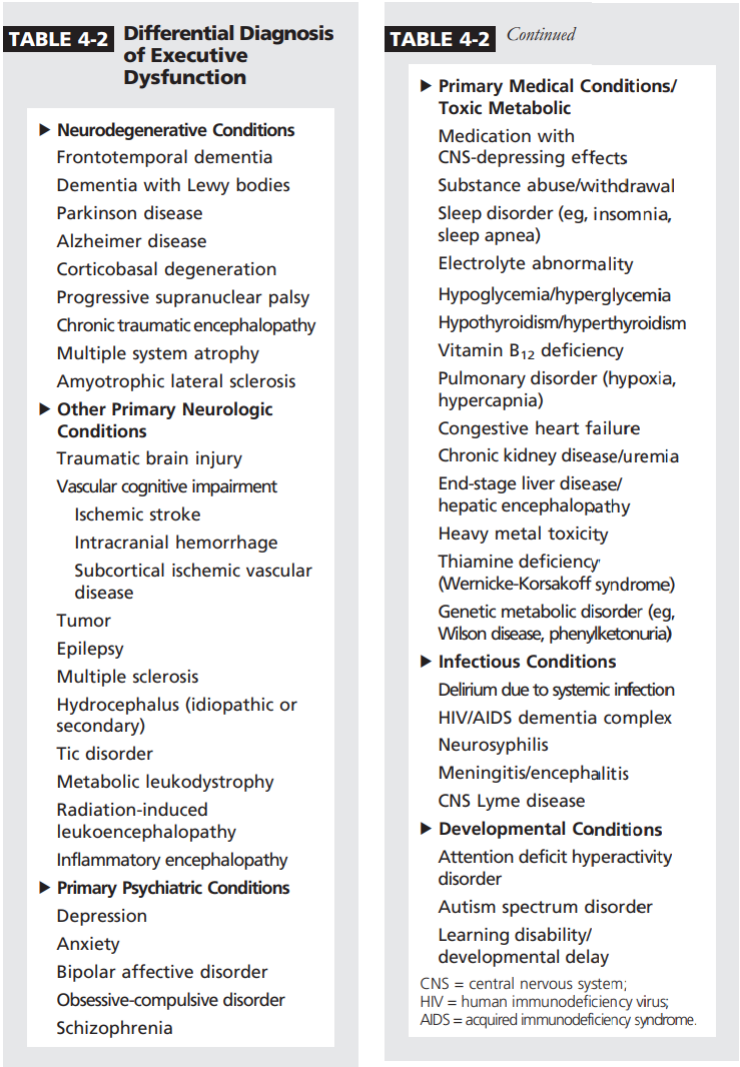


TABLE 4-2 Differential Diagnosis of Executive Dysfunction

- ▶ **Neurodegenerative Conditions**
  - Frontotemporal dementia
  - Dementia with Lewy bodies
  - Parkinson disease
  - Alzheimer disease
  - Corticobasal degeneration
  - Progressive supranuclear palsy
  - Chronic traumatic encephalopathy
  - Multiple system atrophy
  - Amyotrophic lateral sclerosis
- ▶ **Other Primary Neurologic Conditions**
  - Traumatic brain injury
  - Vascular cognitive impairment
    - Ischemic stroke
    - Intracranial hemorrhage
    - Subcortical ischemic vascular disease
  - Tumor
  - Epilepsy
  - Multiple sclerosis
  - Hydrocephalus (idiopathic or secondary)
  - Tic disorder
  - Metabolic leukodystrophy
  - Radiation-induced leukoencephalopathy
  - Inflammatory encephalopathy
- ▶ **Primary Psychiatric Conditions**
  - Depression
  - Anxiety
  - Bipolar affective disorder
  - Obsessive-compulsive disorder
  - Schizophrenia

TABLE 4-2 *Continued*

- ▶ **Primary Medical Conditions/ Toxic Metabolic**
  - Medication with CNS-depressing effects
  - Substance abuse/withdrawal
  - Sleep disorder (eg, insomnia, sleep apnea)
  - Electrolyte abnormality
  - Hypoglycemia/hyperglycemia
  - Hypothyroidism/hyperthyroidism
  - Vitamin B_{12} deficiency
  - Pulmonary disorder (hypoxia, hypercapnia)
  - Congestive heart failure
  - Chronic kidney disease/uremia
  - End-stage liver disease/ hepatic encephalopathy
  - Heavy metal toxicity
  - Thiamine deficiency (Wernicke-Korsakoff syndrome)
  - Genetic metabolic disorder (eg, Wilson disease, phenylketonuria)
- ▶ **Infectious Conditions**
  - Delirium due to systemic infection
  - HIV/AIDS dementia complex
  - Neurosyphilis
  - Meningitis/encephalitis
  - CNS Lyme disease
- ▶ **Developmental Conditions**
  - Attention deficit hyperactivity disorder
  - Autism spectrum disorder
  - Learning disability/ developmental delay

CNS = central nervous system; HIV = human immunodeficiency virus; AIDS = acquired immunodeficiency syndrome.

Figura A.1: Tabla que muestra posibles causas de una Disfunción Ejecutiva. Fuente: (Rabinovici, Stephens Possin, 2015) [4]

Test de diagnóstico de Disfunción Ejecutiva

CUADRO IV. EXAMEN MMSE DE FOLSTEIN (EVALÚA ESTADO MENTAL)

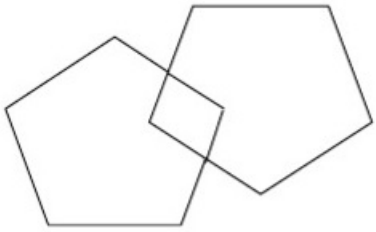
ORIENTACIÓN	**PUNTOS**
¿Qué año-estación-fecha-día-mes es?	(5)
¿Dónde estamos? (estado-país-cuidad-hospital-piso)	(5)
MEMORIA INMEDIATA	
Repetir 3 nombres ("mesa", "llave", "libro"). Repetirlos de nuevo hasta que aprenda los tres nombres y anotar el número de ensayos.	(3)
ATENCIÓN Y CÁLCULO	
Restar 7 a partir de 100, 5 veces consecutivas. Como alternativa, deletrear "mundo" al revés.	(5)
RECUERDO DIFERIDO	
Repetir los 3 nombres aprendidos antes.	(3)
LENGUAJE Y CONSTRUCCIÓN	
Nombrar un lápiz y un reloj mostrados	(2)
Repetir la frase "Ni si es, ni no es, ni peros"	(1)
Realizar correctamente las tres órdenes siguientes: "Tome este papel con la mano derecha, dóblelo por la mitad y póngalo en el suelo"	(3)
Leer y ejecutar la frase "Cierre los ojos"	(1)
Escribir una frase con sujeto y predicado	(1)
Copiar este dibujo: 	(1)
Puntuación total:	

Figura A.2: Mini-mental test de Folstein.

VERDE	VERDE	ROJO	AZUL
AZUL	ROJO	AZUL	VERDE
VERDE	AZUL	ROJO	ROJO
ROJO	ROJO	VERDE	AZUL
AZUL	VERDE	AZUL	VERDE
ROJO	AZUL	VERDE	AZUL
AZUL	VERDE	ROJO	VERDE
VERDE	ROJO	AZUL	ROJO

Figura A.3: Stroop test

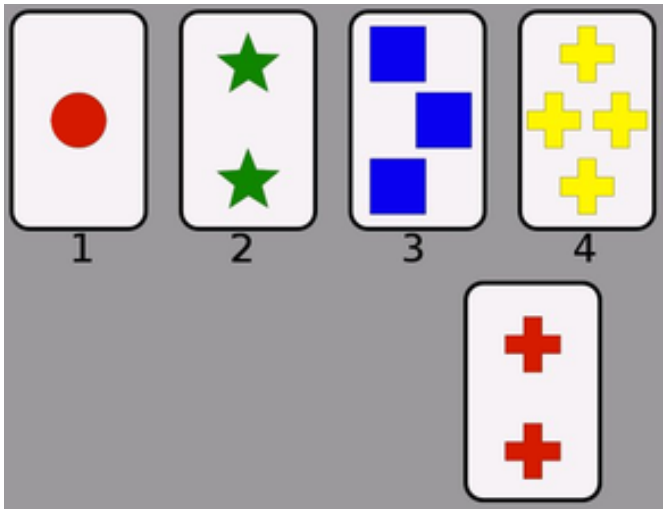


Figura A.4: Wisconsin Card Sorting Test (WCST).

Figura A.5: Flanker test.

Anexo B

Códigos

Archivo JSON exportado de la plataforma Neuronat

Código B.1: Ejemplo en json.


```json
{
  "player": {
    "nombre": " Jos ",
    "edad": 100,
    "genero": "M",
    "educacion": "Superior incompleta"
  },
  "statesData": [
    {
      "gameState": "B001",
      "metricsTable1Type": "TakeOrder",
      "metricsTable1": {
        "greet": {
          "accomplished": false,
          "time": -1
        },
        "takeOrderStart": 6.783209323883057,
        "takeOrderReady": 11.532215118408203,
        "takeOrderCompleted": 12.182025909423828,
        "takeOrderErrors": [],
        "menuErrors": [],
        "startStateTime": 0,
        "endStateTime": 0
      },
      "metricsTable2Type": "S1",
      "metricsTable2": {},
      "metricsTable3Type": "D",
      "metricsTable3": {},
      "metricsTable4Type": "D",
      "metricsTable4": {},
      "metricsTable5Type": "Null",
```

```
      "metricsTable5": {}
    },
    {
      "gameState": "B002",
      "metricsTable1Type": "D",
      "metricsTable1": {},
      "metricsTable2Type": "S1",
      "metricsTable2": {},
      "metricsTable3Type": "TakeOrder",
      "metricsTable3": {
        "greet": {
          "accomplished": false ,
          "time": -1
        },
        "takeOrderStart": 18.71379280090332,
        "takeOrderReady": 40.89201736450195,
        "takeOrderCompleted": 41.80937194824219,
        "takeOrderErrors": [],
        "menuErrors": [],
        "startStateTime": 12.182025909423828,
        "endStateTime": 0
      },
      "metricsTable4Type": "D",
      "metricsTable4": {},
      "metricsTable5Type": "Null",
      "metricsTable5": {}
    },
    {
      "gameState": "B003",
      "metricsTable1Type": "D",
      "metricsTable1": {},
      "metricsTable2Type": "S1",
      "metricsTable2": {},
      "metricsTable3Type": "DeliverOrder",
      "metricsTable3": {
        "dilemma": true,
        "tableErrors": [],
        "foodErrors": [],
        "foodErrorsRepeat": [
          {
            "time": 94.0687484741211,
            "foodCode": "gnocchi",
            "personIndex": 1
          }
        ],
        "incompleteDeliveryErrors": [],
        "ringsTime": [
          41.80937194824219,
          51.71365737915039
        ],
```

```
        "conflictOptions": [],
        "deliverOrderStart": 81.1383285522461,
        "deliverOrderCompleted": 95.36911010742188,
        "orderReady": 69.5244140625,
        "startStateTime": 41.80937194824219,
        "endStateTime": 0
      },
      "metricsTable4Type": "D",
      "metricsTable4": {},
      "metricsTable5Type": "Null",
      "metricsTable5": {}
    },
    {
      "gameState": "B004",
      "metricsTable1Type": "SpeakOptions",
      "metricsTable1": {
        "speakOptionStart": 133.059814453125,
        "speakOptionCompleted": 135.9257354736328,
        "option": "Disculpe por la demora, hay problemas en la cocina, pero le regalaremos
↪ un postre por este inconveniente",
        "startStateTime": 95.36911010742188,
        "endStateTime": 0
      },
      "metricsTable2Type": "S2",
      "metricsTable2": {},
      "metricsTable3Type": "D",
      "metricsTable3": {},
      "metricsTable4Type": "D",
      "metricsTable4": {},
      "metricsTable5Type": "Null",
      "metricsTable5": {}
    },
    {
      "gameState": "B005",
      "metricsTable1Type": "D",
      "metricsTable1": {},
      "metricsTable2Type": "S4",
      "metricsTable2": {},
      "metricsTable3Type": "D",
      "metricsTable3": {},
      "metricsTable4Type": "TakeOrder",
      "metricsTable4": {
        "greet": {
          "accomplished": false ,
          "time": -1
        },
        "takeOrderStart": 192.51266479492188,
        "takeOrderReady": 214.90771484375,
        "takeOrderCompleted": 215.62428283691406,
        "takeOrderErrors": [],
```

```
        "menuErrors": [],
        "startStateTime": 135.9257354736328,
        "endStateTime": 0
      },
      "metricsTable5Type": "Null",
      "metricsTable5": {}
    },
    {
      "gameState": "B006",
      "metricsTable1Type": "D",
      "metricsTable1": {},
      "metricsTable2Type": "S4",
      "metricsTable2": {},
      "metricsTable3Type": "Transaction",
      "metricsTable3": {
        "candy": {
          "delivered": true,
          "time": 221.78936767578125
        },
        "tip": {
          "included": false,
          "time": -1
        },
        "goodByeCompleted": 224.00784301757812,
        "tableReady": 215.62428283691406,
        "startStateTime": 215.62428283691406,
        "endStateTime": 0
      },
      "metricsTable4Type": "D",
      "metricsTable4": {},
      "metricsTable5Type": "Null",
      "metricsTable5": {}
    },
    {
      "gameState": "B007",
      "metricsTable1Type": "D",
      "metricsTable1": {},
      "metricsTable2Type": "S6",
      "metricsTable2": {},
      "metricsTable3Type": "Null",
      "metricsTable3": {},
      "metricsTable4Type": "D",
      "metricsTable4": {},
      "metricsTable5Type": "TakeOrder",
      "metricsTable5": {
        "greet": {
          "accomplished": false,
          "time": -1
        },
        "takeOrderStart": 264.42950439453125,
```

```
        "takeOrderReady": 270.5114440917969,
        "takeOrderCompleted": 271.3804626464844,
        "takeOrderErrors": [],
        "menuErrors": [],
        "startStateTime": 224.00784301757812,
        "endStateTime": 0
      }
    },
    {
      "gameState": "B008",
      "metricsTable1Type": "PreEnd",
      "metricsTable1": {
        "startStateTime": 271.3804626464844,
        "endStateTime": 0
      },
      "metricsTable2Type": "S6",
      "metricsTable2": {},
      "metricsTable3Type": "Null",
      "metricsTable3": {},
      "metricsTable4Type": "DeliverOrder",
      "metricsTable4": {
        "dilemma": false,
        "tableErrors": [],
        "foodErrors": [],
        "foodErrorsRepeat": [],
        "incompleteDeliveryErrors": [],
        "ringsTime": [
          271.3804626464844
        ],
        "conflictOptions": [],
        "deliverOrderStart": -1,
        "deliverOrderCompleted": -1,
        "orderReady": -1,
        "startStateTime": 271.3804626464844,
        "endStateTime": 0
      },
      "metricsTable5Type": "D",
      "metricsTable5": {}
    },
    {
      "gameState": "B021",
      "metricsTable1Type": "Null",
      "metricsTable1": {},
      "metricsTable2Type": "S6",
      "metricsTable2": {},
      "metricsTable3Type": "Null",
      "metricsTable3": {},
      "metricsTable4Type": "DeliverOrder",
      "metricsTable4": {
        "dilemma": false,
```

```
        "tableErrors": [],
        "foodErrors": [
          {
            "time": 307.419677734375,
            "foodCode": "vaso_agua",
            "personIndex": 3
          }
        ],
        "foodErrorsRepeat": [],
        "incompleteDeliveryErrors": [],
        "ringsTime": [
          280.29278564453125
        ],
        "conflictOptions": [
          0
        ],
        "deliverOrderStart": 291.0738220214844,
        "deliverOrderCompleted": 311.81878662109375,
        "orderReady": 285.94415283203125,
        "startStateTime": 280.29278564453125,
        "endStateTime": 0
      },
      "metricsTable5Type": "D",
      "metricsTable5": {}
    },
    {
      "gameState": "B028",
      "metricsTable1Type": "Null",
      "metricsTable1": {},
      "metricsTable2Type": "S6",
      "metricsTable2": {},
      "metricsTable3Type": "Null",
      "metricsTable3": {},
      "metricsTable4Type": "D",
      "metricsTable4": {},
      "metricsTable5Type": "SpeakOptions",
      "metricsTable5": {
        "speakOptionStart": 331.6979675292969,
        "speakOptionCompleted": 333.1982421875,
        "option": "Su pedido  est  atrasado y no es mi culpa. Tiene que esperar",
        "startStateTime": 311.81878662109375,
        "endStateTime": 0
      }
    },
    {
      "gameState": "B029",
      "metricsTable1Type": "Null",
      "metricsTable1": {},
      "metricsTable2Type": "S6",
      "metricsTable2": {},
```

```
      "metricsTable3Type": "Null",
      "metricsTable3": {},
      "metricsTable4Type": "Transaction",
      "metricsTable4": {
        "candy": {
          "delivered": false,
          "time": -1
        },
        "tip": {
          "included": false,
          "time": -1
        },
        "goodByeCompleted": 339.61187744140625,
        "tableReady": 333.1982421875,
        "startStateTime": 333.1982421875,
        "endStateTime": 0
      },
      "metricsTable5Type": "PreEnd",
      "metricsTable5": {
        "startStateTime": 333.1982421875,
        "endStateTime": 0
      }
    },
    {
      "gameState": "B030",
      "metricsTable1Type": "Null",
      "metricsTable1": {},
      "metricsTable2Type": "S6",
      "metricsTable2": {},
      "metricsTable3Type": "Null",
      "metricsTable3": {},
      "metricsTable4Type": "Null",
      "metricsTable4": {},
      "metricsTable5Type": "End",
      "metricsTable5": {}
    }
  ],
  "counterButtonClicks": []
}
```

Código de Models

Código B.2: Ejemplo de modelos en Django

```
from django.db import models

class takeOrder(models.Model):
    greet = models.BooleanField(null=True)
    greet_time = models.FloatField(null=True)
```

```
    takeOrderStart = models.FloatField(null=True)
    takeOrderReady = models.FloatField(null=True)
    takeOrderCompleted = models.FloatField(null=True)
    startStateTime = models.FloatField(null=True)
    endStateTime = models.FloatField(null=True)
    state = models.CharField(null=True, max_length=50)
    table = models.IntegerField(null=True)

class takeOrderErrors(models.Model):
    peopleIndex = models.IntegerField(null=True)
    foodCode = models.CharField(null=True, max_length=50)
    time = models.FloatField(null=True)
    order = models.ForeignKey(takeOrder, on_delete=models.CASCADE, null=True)
```

Código de Serializers

Código B.3: Ejemplo de cómo se ven las clases de Serializers

```
from rest_framework import serializers
from .models import *

class takeOrderSerializer(serializers.ModelSerializer):
    class Meta:
        model = takeOrder
        fields = ('id','greet','greet_time','takeOrderStart','takeOrderReady','
    ↪ takeOrderCompleted','startStateTime','endStateTime','state','table')

class takeOrderErrorsSerializer(serializers.ModelSerializer):
    class Meta:
        model = takeOrderErrors
        fields = ('id','peopleIndex','foodCode','time','order')
```

Función postNeuronatData en views.py de la API REST

Código B.4: Función postNeuronatData que permite almacenar los datos de la plataforma Neuronat en la base de datos

```
class postNeuronatData(APIView):
    parser_classes = [JSONParser]
    def post(self, request, format=None):
        player = request.data['player']
        states = request.data['statesData']
        button = request.data['counterButtonClicks']
        jugador = Jugador(alias= player['nombre'],edad=player['edad'],genero=player['
    ↪ genero'], educacion=player['educacion'])
        jugador.save()
        for click in button:
            counterClicks = CounterClicks(clickTime=click, jugador=jugador)
```

```
            counterClicks.save()
        for state in states:
            estado = state['gameState']
            for key in state:
                if 'metricsTable' in key and state[key] != {} and 'Type' not in key:
                    table = key
                    table = table.replace('metricsTable', '')
                    table = int(table)
                    action = state[key+'Type']
                    if action=='TakeOrder':
                        greet = state[key]['greet']['accomplished']
                        greet_time = state[key]['greet']['time']
                        takeOrderStart = state[key]['takeOrderStart']
                        takeOrderReady = state[key]['takeOrderReady']
                        takeOrderCompleted = state[key]['takeOrderCompleted']
                        startStateTime = state[key]['startStateTime']
                        endStateTime = state[key]['endStateTime']
                        takeorder = takeOrder(greet=greet, greet_time=greet_time,
↪ takeOrderStart=takeOrderStart,takeOrderReady=takeOrderReady,
↪ takeOrderCompleted=takeOrderCompleted,startStateTime=startStateTime,
↪ endStateTime=endStateTime, state=estado, table=table)
                        takeorder.save()
                        for error in state[key]['takeOrderErrors']:
                            takeordererror = takeOrderErrors(peopleIndex=error['
↪ peopleIndex'], foodCode=error['foodCode'], time=error['time'], order=takeorder)
                            takeordererror.save()
                        for error in state[key]['menuErrors']:
                            menuerror = menuErrors(type=error['type'], time=error['time'],
↪ order=takeorder)
                            menuerror.save()
                        jugadortakeorder = jugadorTakeOrder(jugador=jugador,takeOrder=
↪ takeorder)
                        jugadortakeorder.save()
                    elif action=='SpeakOptions':
                        speakOptionStart = state[key]['speakOptionStart']
                        speakOptionCompleted = state[key]['speakOptionCompleted']
                        option = state[key]['option']
                        startStateTime = state[key]['startStateTime']
                        endStateTime = state[key]['endStateTime']
                        opciones = Opciones(speakOptionStart=speakOptionStart,
↪ speakOptionCompleted=speakOptionCompleted,option=option,startStateTime=
↪ startStateTime,endStateTime=endStateTime, state=estado, table=table)
                        opciones.save()
                        jugadoropciones = jugadorOpciones(jugador=jugador, opciones=
↪ opciones)
                        jugadoropciones.save()
                    elif action == 'DeliverOrder':
                        if len(state[key]['conflictOptions'])==0:
                            dilemma = state[key]['dilemma']
                            if dilemma==True:
```

```
52                                 dilemma = 0
53                             else:
54                                 dilemma = 1
55                             deliverOrderStart = state[key]['deliverOrderStart']
56                             deliverOrderCompleted = state[key]['deliverOrderCompleted']
57                             orderReady = state[key]['orderReady']
58                             startStateTime = state[key]['startStateTime']
59                             endStateTime = state[key]['endStateTime']
60                             delivery = DeliveryOrder(dilemma=dilemma, deliverOrderStart
    ↪ =deliverOrderStart, deliverOrderCompleted=deliverOrderCompleted, orderReady=
    ↪ orderReady,startStateTime=startStateTime,endStateTime=endStateTime, state=
    ↪ estado, table=table)
61                             delivery.save()
62                             for error in state[key]['foodErrorsRepeat']:
63                                 fooderrorsrepeat = foodErrorsRepeat(time=error['time'],
    ↪ foodCode=error['foodCode'], personIndex=error['personIndex'], deliveryOrder=
    ↪ delivery)
64                                 fooderrorsrepeat.save()
65                             for error in state[key]['ringsTime']:
66                                 ringstime = RingsTime(time=error,deliveryOrder=delivery)
67                                 ringstime.save()
68                             for error in state[key]['tableErrors']:
69                                 tablerror = tableErrors(time=error['time'] ,
    ↪ expectedTableName=error['expectedTableName'] ,tableName=error['tableName'] ,
    ↪ deliveryOrder=delivery)
70                                 tablerror.save()
71                             for error in state[key]['foodErrors']:
72                                 fooderror = foodErrors(foodCode=error['foodCode'],time=
    ↪ error['time'], personIndex = error['personIndex'], deliveryOrder=delivery)
73                                 fooderror.save()
74                             for error in state[key]['incompleteDeliveryErrors']:
75                                 incompletedeliveryerror = incompleteDeliveryErrors(time=
    ↪ error['time'],foodCodesReady=error['foodCodesReady'], deliveryOrder=delivery)
76                                 incompletedeliveryerror.save()
77                             jugadordelivery = jugadorDelivery(jugador=jugador,
    ↪ deliveryOrder=delivery)
78                             jugadordelivery.save()
79                         else:
80                             dilemma = state[key]['conflictOptions'][0]
81                             deliverOrderStart = state[key]['deliverOrderStart']
82                             deliverOrderCompleted = state[key]['deliverOrderCompleted']
83                             orderReady = state[key]['orderReady']
84                             startStateTime = state[key]['startStateTime']
85                             endStateTime = state[key]['endStateTime']
86                             delivery = DeliveryOrder(dilemma=dilemma, deliverOrderStart
    ↪ =deliverOrderStart,
87                                                      deliverOrderCompleted=
    ↪ deliverOrderCompleted, orderReady=orderReady,
88                                                      startStateTime=startStateTime,
    ↪ endStateTime=endStateTime,
```

```
                                                    state=estado, table=table)
                        delivery.save()
                        for error in state[key]['foodErrorsRepeat']:
                            fooderrorsrepeat = foodErrorsRepeat(time=error['time'],
↪ foodCode=error['foodCode'],
                                                                personIndex=error['
↪ personIndex'],
                                                                deliveryOrder=delivery
↪ )
                            fooderrorsrepeat.save()
                        for error in state[key]['ringsTime']:
                            ringstime = RingsTime(time=error, deliveryOrder=delivery
↪ )
                            ringstime.save()
                        for error in state[key]['tableErrors']:
                            tablerror = tableErrors(time=error['time'],
                                                    expectedTableName=error['
↪ expectedTableName'],
                                                    tableName=error['tableName'],
↪ deliveryOrder=delivery)
                            tablerror.save()
                        for error in state[key]['foodErrors']:
                            fooderror = foodErrors(foodCode=error['foodCode'], time=
↪ error['time'],
                                                   personIndex=error['personIndex'],
↪ deliveryOrder=delivery)
                            fooderror.save()
                        for error in state[key]['incompleteDeliveryErrors']:
                            incompletedeliveryerror = incompleteDeliveryErrors(time=
↪ error['time'],

↪ foodCodesReady=error[
                                                                                ,
↪ foodCodesReady'],

↪ deliveryOrder=delivery)
                            incompletedeliveryerror.save()
                        jugadordelivery = jugadorDelivery(jugador=jugador,
↪ deliveryOrder=delivery)
                        jugadordelivery.save()
                    elif action == 'PreEnd':
                        startStateTime = state[key]['startStateTime']
                        endStateTime = state[key]['endStateTime']
                        pretermino = Pretermino(startStateTime=startStateTime,
↪ endStateTime=endStateTime, state=estado, table=table)
                        pretermino.save()
                        jugadorpretermino = jugadorPretermino(jugador=jugador,
↪ pretermino=pretermino)
                        jugadorpretermino.save()
                    elif action == 'Transaction':
```

```
                        candy = state[key]['candy']['delivered']
                        candyTime = state[key]['candy']['time']
                        tip = state[key]['tip']['included']
                        tipTime = state[key]['tip']['time']
                        goodbyeCompleted = state[key]['goodByeCompleted']
                        tableReady = state[key]['tableReady']
                        startStateTime = state[key]['startStateTime']
                        endStateTime = state[key]['endStateTime']
                        transaccion = Transaccion(candy=candy,candyTime=candyTime,tip
    ↪ =tip,tipTime=tipTime,goodbyeCompleted=goodbyeCompleted,tableReady=
    ↪ tableReady,startStateTime=startStateTime,endStateTime=endStateTime, state=
    ↪ estado, table=table)
                        transaccion.save()
                        jugadortransaccion = jugadorTransaccion(jugador=jugador,
    ↪ transaccion=transaccion)
                        jugadortransaccion.save()
                    else:
                        startStateTime = state[key]['startStateTime']
                        endStateTime = state[key]['endStateTime']
                        termino = Termino(startStateTime=startStateTime, endStateTime
    ↪ =endStateTime, state=estado, table=table)
                        termino.save()
                        jugadortermino = jugadorTermino(jugador= jugador, termino=
    ↪ termino)
                        jugadortermino.save()
        return Response(request.data)
```

Función creada para entrenar modelos de SVM

Código B.5: Función que define un modelo de Support Vector Machine

```
def run_svm(X_train, X_test, y_train, y_test):
    # C
    C = [.0001, .001, .01]
    # gamma
    gamma = [.0001, .001, .01, .1, 1, 10, 100]
    # degree
    degree = [1, 2, 3, 4, 5]
    # kernel
    kernel = ['linear', 'rbf', 'poly']
    # probability
    probability = [True]
    # Create the random grid
    random_grid = {'C': C,
                   'kernel': kernel,
                   'gamma': gamma,
                   'degree': degree,
                   'probability': probability
```

```
                }
# First create the base model to tune
svc = svm.SVC(random_state=8)
# Definition of the random search

cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3,random_state=100)
random_search = RandomizedSearchCV(estimator=svc,
                                   param_distributions=random_grid,
                                   n_iter=50,
                                   scoring='accuracy',
                                   cv=cv,
                                   verbose=1,
                                   random_state=8)
svm_y_train = np.array(list(y_train[5]))
# Fit the random search model
random_search.fit(X_train, svm_y_train)

print("The best hyperparameters from Random Search are:")
print(random_search.best_params_)
print("")
print("The mean accuracy of a model with these hyperparameters is:")
print(random_search.best_score_)

rbf = random_search.best_params_.copy()
poly = random_search.best_params_.copy()
linear = random_search.best_params_.copy()

rbf['probability'] = [rbf['probability']]
rbf['kernel'] = ['rbf']
rbf['gamma'] = [rbf['gamma']*0.8, rbf['gamma'], rbf['gamma']*1.2]
rbf['degree'] = [rbf['degree']*0.8, rbf['degree'], rbf['degree']*1.2]
rbf['C'] = [rbf['C']*0.8,rbf['C'],rbf['C']*1.2]

poly['probability'] = [poly['probability']]
poly['kernel'] = ['poly']
poly['gamma'] = [poly['gamma']*0.8, poly['gamma'], poly['gamma']*1.2]
poly['degree'] = [poly['degree']*0.8, poly['degree'], poly['degree']*1.2]
poly['C'] = [poly['C']*0.8,poly['C'],poly['C']*1.2]

linear['probability'] = [linear['probability']]
linear['kernel'] = ['linear']
linear['gamma'] = [linear['gamma']*0.8, linear['gamma'], linear['gamma']*1.2]
linear['degree'] = [linear['degree']*0.8, linear['degree'], linear['degree']*1.2]
linear['C'] = [linear['C']*0.8, linear['C'], linear['C']*1.2]

param_grid = [linear, poly, rbf]

# Create a base model
svc = svm.SVC(random_state=8)

```

```
    # Manually create the splits in CV in order to be able to fix a random_state (
    ↪ GridSearchCV doesn't have that argument)
    cv_sets = StratifiedShuffleSplit(n_splits = 5, test_size = .33, random_state = 8)

    # Instantiate the grid search model
    grid_search = GridSearchCV(estimator=svc,
                               param_grid=param_grid,
                               scoring='accuracy',
                               cv=cv_sets,
                               verbose=1)

    # Fit the grid search to the data
    grid_search.fit(X_train, svm_y_train)

    print("")
    print("The best hyperparameters from Grid Search are:")
    print(grid_search.best_params_)
    print("")
    print("The mean accuracy of a model with these hyperparameters is:")
    print(grid_search.best_score_)

    best_svc = grid_search.best_estimator_
    best_svc.fit(X_train, y_train)
    svc_pred = best_svc.predict(X_test)

    # Training accuracy
    print("The training accuracy is: ")
    print(accuracy_score(y_train, best_svc.predict(X_train)))


    # Test accuracy
    print("The test accuracy is: ")
    print(accuracy_score(y_test, svc_pred))
    print('Cantidad de features: ')
    print(X_train.shape[1])
    return best_svc
```

Generación de métricas

Código B.6: Código de Python que muestra cómo se construye una métrica utilizando el método Group By

```
#Se selecciona el primer error cometido en el menú universal en caso de existir.
#Si no existe error se rellena con 'take_order'
first_menu_error = menuErrors.groupby(['order']).min()
takeOrder1 = pd.merge(takeOrder1,
                      first_menu_error,
                      right_index=True,
                      left_on='pk',
```

```
                        how='left'
                        )
takeOrder1['type'] = takeOrder1['type'].fillna('take_order')

#y luego se categoriza con valores enteros (0-4)
le = preprocessing.LabelEncoder()
takeOrder1['type_label1'] = le.fit_transform(takeOrder1['type'])
takeOrder1['take_order_boolean1'] = np.where(takeOrder1['type']=='take_order',1,0)
takeOrder1 = takeOrder1.rename(columns={'pk_x':'takeOrderID'})
```

Anexo C

Tablas

C.1. Variables de la base manual

Tabla C.1: Glosario de todas las variables generadas en la base manual

Variables	Definición	Valores
greet_boolean1	En el primer contacto con la mesa el cliente debe presentarse.	1 = mozo se presenta, 0 = mozo no se presenta (sólo se considera 1 si se hace al primer intento)
type_label1	Eleccion de cualquiera de las actividades a realizar. 1: presentarse, 2 tomar pedido, 3 entregar pedido, 4 cierre de mesa, 5 cortesia de la casa	Puntaje de 1 a 5 según opcion tomada
take_order_boolean1	Si selecciona inmediatamente la opción toma de orden en el menú universal se considera como 1, de lo contrario es un cero	1 = mozo selecciona tomar orden, 0 = mozo selecciona otra opción del menú universal (cualquier otra)
MenuErrorCount1	Cantidad de errores hasta elegir la opción correcta.	0 = 0 Error, 1 = 1 Error, 2 = 2 Errores, 3 = 3 Errores
persevMenu1	Si el jugador en sus 3 intentos elige la misma opción, se considera como perseverancia.	1 = perseverancia (3 errores iguales), 0 = no perseverancia
TakeOrderErrorCount1	Cantidad de errores al momento de tomar la orden en primer intento	X errores y de total de 2 posibles
persevTakeOrder1	Si en los 3 intentos elige la misma comida y/o bebida, se está en el caso de perseverancia.	1 = perseverancia (3 errores iguales), 0 = no perseverancia
TakeOrderTotalTime1	Tiempo que se demora el jugador en todo el proceso de toma de orden.	Número real que corresponde a un tiempo.
TakeOrderButtonTime1	Tiempo que se demora el jugador en apretar el botón, desde que terminó de tomar la orden correctamente.	Número real que corresponde a un tiempo.
option_label1	Opción elegida con el cliente enojado cuando no llega su comida. 2 opciones.	0 = respuesta 1 (. . . ...), 2 = respuesta (. . . ...)
OptionTime1	Cuánto se demora el jugador en seleccionar la respuesta.	Número real que corresponde a un tiempo.
totalTime1	Cantidad de tiempo que se demora el jugador en atender todas las etapas de la mesa.	Número real que corresponde a un tiempo.
greet_boolean3	En el primer contacto con la mesa el cliente debe presentarse.	1 = mozo se presenta, 0 = mozo no se presenta (sólo se considera 1 si se hace al primer intento)
type_label3	Eleccion de cualquiera de las actividades a realizar: 1: presentarse, 2 tomar pedido, 3 entregar pedido, 4 cierre de mesa, 5 cortesia de la casa	Puntaje de 1 a 5 según opcion tomada
take_order_boolean3	Si selecciona inmediatamente la opción toma de orden en el menú universal se considera como 1, de lo contrario es un cero	1 = mozo selecciona tomar orden, 0 = mozo selecciona otra opción del menú universal (cualquier otra)
MenuErrorCount3	Cantidad de errores hasta elegir la opción correcta.	0 = 0 Error, 1 = 1 Error, 2 = 2 Errores, 3 = 3 Errores
persevMenu3	Si el jugador en sus 3 intentos elige la misma opción, se considera como perseverancia.	1 = perseverancia (3 errores iguales), 0 = no perseverancia
TakeOrderErrorCount3	Cantidad de errores al momento de tomar la orden en primer intento	Número real que corresponde a la cantidad de errores
persevTakeOrder3	Si el jugador repite alguna opción en alguno de los intentos, esta variable toma el valor 1.	1 = al menos una repetición en cualquier intento, 0 = no repetición.
TakeOrderTotalTime3	Tiempo que se demora el jugador en todo el proceso de toma de orden.	Número real que corresponde a un tiempo.
TakeOrderButtonTime3	Tiempo que se demora el jugador en apretar el botón, desde que terminó de tomar la orden correctamente.	Número real que corresponde a un tiempo.
dilemma_x	Resultado del dilema del vino.	0 = avisa sobre el vino, 1= no avisa
first_bell3	Si se va en primera instancia a la cocina a buscar el pedido o no.	1 = Se va en primera instancia, 0 = Se va despues
kitchen_time3	Cantidad de tiempo que el jugador está en el pop de cocina. Es una aproximación, ya que no se puede calcular exactamente: se resta el tiempo de OrderReady con el tiempo del último timbre.	Número real que corresponde a un tiempo.
deliverErrors3	Numero total de elementos mal registrados en la entrega de orden	X de 8 = primer registro.
persevDeliveryErrors3	Si se elige 3 veces la misma opción se considera perseverancia.	1 = perseverancia (3 errores iguales), 0 = no perseverancia
repeatFoodErrors3	Cantidad de veces que se le entrega a un cliente el plato que pidio ya habiendo servido su plato.	Numero entero que representa la cantidad de estos errores. Asociado a la entidad foodErrorsRepeat
deliveryTime3	Tiempo que se demora el jugador en entregar la orden, desde que se abre el pop up hasta que se cierra.	Número real que corresponde a un tiempo.
attend_table3	En base a las 4 variables anteriores se puede definir categóricamente un momento en que se decide ir a la mesa a hacer la transacción: 1 si se va en primera instancia antes de los 10 segundos en la primera señal (cubiertos), 2 si se va a la segunda señal (clienta lo llama), y 3 si el jefe tuvo que intervenir.	0 = Se ignora a la mesa en primera instancia, 2 = se atiende en primera instancia, 3 = No hay más opción que atender la mesa, no puede ser ignorada.
candy_pretip3	Si se le entrega el dulce antes de la propina o no.	1 = se acuerda de entregar el dulce antes de la propina, 0 = no entrega dulce
candy_posttip3	Si se le entrega el dulce después de la propina o no.	1 = Da el dulce, 0 = no da nunca el dulce.
tip_binary3	Si el jugador pide propina o no.	1 = jugador pide propina, 0 = jugador no pide propina.
transactionTime3	Tiempo que se demora el jugador en cerrar la mesa desde que se selecciona la mesa hasta que el cliente se va	Número real que corresponde a un tiempo.
totalTime3	Tiempo que se demora el jugador en toda la mesa	Número real que corresponde a un tiempo.
greet_boolean4	En el primer contacto con la mesa el cliente debe presentarse.	1 = mozo se presenta, 0 = mozo no se presenta (sólo se considera 1 si se hace al primer intento)
type_label4	Eleccion de cualquiera de las actividades a realizar: 1: presentarse, 2 tomar pedido, 3 entregar pedido, 4 cierre de mesa, 5 cortesia de la casa	Puntaje de 1 a 5 según opcion tomada
take_order_boolean4	Si selecciona inmediatamente la opción toma de orden en el menú universal se considera como 1, de lo contrario es un cero	1 = mozo selecciona tomar orden, 0 = mozo selecciona otra opción del menú universal (cualquier otra)
MenuErrorCount4	Cantidad de errores hasta elegir la opción correcta.	0 = 0 Error, 1 = 1 Error, 2 = 2 Errores, 3 = 3 Errores
persevMenu4	Si el jugador en sus 3 intentos elige la misma opción, se considera como perseverancia.	1 = perseverancia (3 errores iguales), 0 = no perseverancia
TakeOrderErrorCount4	Cantidad de errores al momento de tomar la orden en primer intento	Número real que corresponde a la cantidad de errores
persevTakeOrder4	Se considera repetición si el jugador selecciona por lo menos 2 veces la misma comida/bebida en un mismo intento.	1 = al menos una repetición en cualquier intento, 0 = no repetición.
TakeOrderTotalTime4	Tiempo que se demora el jugador en todo el proceso de toma de orden.	Número real que corresponde a un tiempo.
TakeOrderButtonTime4	Segundo en el que el jugador decide seleccionar el botón, desde que terminó de tomar la orden correctamente.	Número real que corresponde a un tiempo.
dilemma_y	El cliente alega de que eso no es lo que pidió. El mozo debe responderle una de las 2 posibles opciones, 0 o 1.	0= se le ofrece un 10% de descuento, 1 = se le dice que no puede hacer nada y que se coma la comida.
first_bell4	Si se va en primera instancia a la cocina a buscar el pedido o no.	1 = Se va en primera instancia, 0 = Se va despues
kitchen_time4	Tiempo que se demora el jugador en el pop de la cocina	Número real que corresponde a un tiempo.
deliverErrors4	Numero total de elementos mal registrados en la entrega de orden	X de 8 = primer registro.
persevDeliveryErrors4	Si en los 3 intentos elige la misma comida y/o bebida, se está en el caso de perseverancia.	1 = perseverancia (3 errores iguales), 0 = no perseverancia
repeatFoodErrors4	Cantidad de veces que se le entrega a un cliente el plato que pidio ya habiendo servido su plato.	Numero entero que representa la cantidad de estos errores. Asociado a la entidad foodErrorsRepeat
deliveryTime4	Tiempo que se demora el jugador en entregar la orden, desde que se abre el pop up hasta que se cierra.	Número real que corresponde a un tiempo.
attend_table4	En base a las 4 variables anteriores se puede definir categóricamente un momento en que se decide ir a la mesa a hacer la transacción: 1 si se va en primera instancia antes de los 10 segundos en la primera señal (cubiertos), 2 si se va a la segunda señal (clienta lo llama), y 3 si el jefe tuvo que intervenir.	0 = Se ignora a la mesa en primera instancia, 2 = se atiende en primera instancia, 3 = No hay más opción que atender la mesa, no puede ser ignorada.
candy_pretip4	Si se le entrega el dulce antes de la propina o no.	1 = se acuerda de entregar el dulce antes de la propina, 0 = no entrega dulce o
candy_posttip4	Si se le entrega el dulce después de la propina o no.	1 = Da el dulce, 0 = no da nunca el dulce.
tip_binary4	Si el jugador pide propina o no.	1 = jugador pide propina, 0 = jugador no pide propina.
transactionTime4	Tiempo que se demora el jugador en cerrar la mesa desde que se selecciona la mesa hasta que el cliente se va	Número real que corresponde a un tiempo.
totalTime4	Tiempo que se demora el jugador en completar la mesa.	Número real que corresponde a un tiempo.
greet_boolean5	En el primer contacto con la mesa el cliente debe presentarse.	1 = mozo se presenta, 0 = mozo no se presenta (sólo se considera 1 si se hace al primer intento)
type_label5	Eleccion de cualquiera de las actividades a realizar: 1: presentarse, 2 tomar pedido, 3 entregar pedido, 4 cierre de mesa, 5 cortesia de la casa	Puntaje de 1 a 5 según opcion tomada
take_order_boolean5	Si selecciona inmediatamente la opción toma de orden en el menú universal se considera como 1, de lo contrario es un cero	1 = mozo selecciona tomar orden, 0 = mozo selecciona otra opción del menú universal (cualquier otra)
MenuErrorCount5	Cantidad de errores hasta elegir la opción correcta. Mínimo 0, máximo 3. A la tercera el juego le indica la actividad a seguir.	0 = 0 Error, 1 = 1 Error, 2 = 2 Errores, 3 = 3 Errores
persevMenu5	Si el jugador en sus 3 intentos elige la misma opción, se considera como perseverancia.	1 = perseverancia (3 errores iguales), 0 = no perseverancia
TakeOrderErrorCount5	Cantidad de errores al momento de tomar la orden en primer intento	X errores y de total de 2 posibles
persevTakeOrder5	Si en los 3 intentos elige la misma comida y/o bebida, se está en el caso de perseverancia.	1 = perseverancia (3 errores iguales), 0 = no perseverancia
TakeOrderTotalTime5	Tiempo que se demora el jugador en todo el proceso de toma de orden.	Número real que corresponde a un tiempo.
TakeOrderButtonTime5	Tiempo que se demora el jugador en apretar el botón, desde que terminó de tomar la orden correctamente.	Número real que corresponde a un tiempo.
option_label5	Opción elegida con el cliente enojado cuando no llega su comida. 2 opciones.	0 = respuesta 1 (. . . ...), 2 = respuesta (. . . ...)
OptionTime5	Cuanto se demora el jugador en seleccionar la respuesta, desde que se hace clic en la mesa hasta que efectivamente elige una opción.	Número real que corresponde a un tiempo.
totalTime5	Cantidad de tiempo que se demora el jugador en atender todas las etapas de la mesa.	Número real que corresponde a un tiempo.

Resumen métricas de evaluación: Base manual

Tabla C.2: Métricas asociadas a los modelos de SVM, muestra de entrenamiento

Modelos	Recall				F1-Score			
	0	**1**	**2**	**3**	**0**	**1**	**2**	**3**
Unique	0.75	1	0.88	0.88	0.75	0.89	0.93	0.93
Uncorr	0.75	1	0.88	0.88	0.75	0.89	0.93	0.93
Mi	0.88	0.88	0.88	0.88	0.78	0.88	0.93	0.93
MiCorr	0.5	1	1	0.62	0.53	0.80	1	0.77
Forward Unique	0.88	1	0.88	0.88	0.82	0.94	0.93	0.93
Forward Uncorr	1	1	1	1	1	1	1	1
Forward Mi	0.75	0.88	0.88	0.88	0.71	0.82	0.93	0.93
Forward MiCorr	0.12	1	0.75	0.88	0.17	0.76	0.86	0.82
Backward Unique	1	1	0.88	0.88	0.89	1	0.93	0.93
Backward Uncorr	0.62	1	0.88	0.88	0.67	0.84	0.93	0.93
Backward Mi	0.88	0.88	0.88	0.88	0.78	0.88	0.93	0.93
Backward MiCorr	0.12	1	0.75	0.88	0.17	0.76	0.86	0.82

Tabla C.3: Métricas asociadas a las clases de los modelos de SVM, muestra de test

Modelos	Recall				F1-Score			
	0	**1**	**2**	**3**	**0**	**1**	**2**	**3**
Unique	0.5	0	0.5	0	0.5	0	0.5	0
Uncorr	0.5	0	0.5	0	0.5	0	0.5	0
Mi	0.5	1	0.5	0	0.67	0.67	0.5	0
MiCorr	0	0	0	0	0	0	0	0
Forward Unique	0.5	0	0	0	0.4	0	0	0
Forward Uncorr	1	0	0	0	0.8	0	0	0
Forward Mi	0.5	1	0	0	0.5	0.5	0	0
Forward MiCorr	0.5	0	0	0	0.5	0	0	0
Backward Unique	0.5	0	0.5	1	0.5	0	0.67	0.67
Backward Uncorr	0.5	0	0.5	0	0.5	0	0.5	0
Backward Mi	0.5	1	0.5	0	0.67	0.67	0.5	0
Backward MiCorr	0.5	0	0	0	0.5	0	0	0

Tabla C.4: Métricas asociadas a los modelos de Random Forest, muestra de entrenamiento

Modelos	Recall				F1-Score			
	0	**1**	**2**	**3**	**0**	**1**	**2**	**3**
Unique	1	1	1	1	1	1	1	1
Uncorr	1	1	1	1	1	1	1	1
Mi	1	1	1	1	1	1	1	1
MiCorr	1	1	1	1	1	1	1	1
Forward Unique	1	1	1	1	1	1	1	1
Forward Uncorr	1	1	1	1	1	1	1	1
Forward Mi	1	1	1	1	1	1	1	1
Forward MiCorr	1	1	1	1	1	1	1	1
Backward Unique	1	1	1	1	1	1	1	1
Backward Uncorr	1	1	1	1	1	1	1	1
Backward Mi	1	1	1	1	1	1	1	1
Backward MiCorr	1	1	1	1	1	1	1	1

Tabla C.5: Métricas asociadas a las clases de los modelos de Random Forest, muestra de test

Modelos	Recall				F1-Score			
	0	**1**	**2**	**3**	**0**	**1**	**2**	**3**
Unique	0.5	1	0	0	0.5	0.67	0	0
Uncorr	0.5	1	0	0	0.5	0.67	0	0
Mi	0.5	1	0	0	0.5	0.5	0	0
MiCorr	1	1	0	0	0.67	1	0	0
Forward Unique	0.5	1	0.5	0	0.4	1	0.67	0
Forward Uncorr	1	1	0	0	0.67	1	0	0
Forward Mi	0.5	1	0	0	0.5	1	0	0
Forward MiCorr	0.5	0	0	0	0.4	0	0	0
Backward Unique	1	1	0.5	0	1	0.67	0.67	0
Backward Uncorr	1	1	0	0	1	0.67	0	0
Backward Mi	1	1	0	0	0.8	0.67	0	0
Backward MiCorr	1	0	0	0	0.67	0	0	0

Tabla C.6: Métricas asociadas a los modelos de Naive Bayes, muestra de entrenamiento

Modelos	Recall				F1-Score			
	0	**1**	**2**	**3**	**0**	**1**	**2**	**3**
Unique	0.88	1	1	1	0.93	0.94	1	1
Uncorr	0.88	1	1	1	0.93	0.94	1	1
Mi	0.88	1	0.88	0.88	0.82	0.94	0.93	0.93
MiCorr	0.88	0.88	0.62	0.88	0.70	0.88	0.77	0.93
Forward Unique	0.62	0.75	0.75	0.75	0.62	0.60	0.86	0.86
Forward Uncorr	0.75	0.75	0.75	0.75	0.67	0.67	0.86	0.86
Forward Mi	0.38	0.75	0.75	0.62	0.40	0.55	0.86	0.77
Forward MiCorr	0.62	0.88	0.88	0.88	0.71	0.82	0.88	0.82
Backward Unique	0.88	0.88	0.75	0.75	0.78	0.78	0.86	0.86
Backward Uncorr	0.88	0.88	0.75	0.75	0.78	0.78	0.86	0.86
Backward Mi	0.88	1	1	0.88	0.88	0.94	1	0.93
Backward MiCorr	0.88	0.88	0.62	0.75	0.67	0.88	0.77	0.86

Tabla C.7: Métricas asociadas a las clases de los modelos de Naive Bayes, muestra de test

Modelos	Recall				F1-Score			
	0	**1**	**2**	**3**	**0**	**1**	**2**	**3**
Unique	1	0	0	0	0.8	0	0	0
Uncorr	1	0	0	0	0.8	0	0	0
Mi	1	0	0	0	0.8	0	0	0
MiCorr	1	0	0	0	0.5	0	0	0
Forward Unique	0.5	1	0.5	0	0.67	0.5	0.5	0
Forward Uncorr	0.5	1	0.5	0	0.67	0.5	0.5	0
Forward Mi	0.5	1	0.5	0	0.67	0.5	0.5	0
Forward MiCorr	1	1	0	0	0.67	0.67	0	0
Backward Unique	0.5	1	0.5	0	0.67	0.5	0.5	0
Backward Uncorr	0.5	1	0.5	0	0.67	0.5	0.5	0
Backward Mi	1	0	0	0	0.8	0	0	0
Backward MiCorr	1	0	0	0	0.5	0	0	0

Tabla C.8: Variables de los mejores modelos para la base manual

	Random Forest		Support Vector Machine		Naive Bayes			Total
	Backward Unique	Backward Uncorr	Backward Unique	Mi	Backward Uncorr	Forward Uncorr	Forward Unique	
Genero	O	O	O	X	X	X	X	**3**
Educacion	O	O	O	X	O	O	O	**6**
greet_boolean1	O	O	O	X	O	O	O	**6**
MenuErrorCount1	O	O	O	X	O	O	O	**6**
takeOrderErrorCount1	O	O	O	X	O	X	X	**4**
takeOrderTotalTime1	X	O	O	O	O	O	O	**6**
takeOrderButtonTime1	O	O	O	O	O	O	O	**7**
option_label1	O	X	O	X	O	O	O	**5**
OptionTime1	X	X	O	X	O	O	O	**4**
totalTime1	X	X	O	O	O	X	X	**3**
greet_boolean3	O	O	O	X	O	X	X	**4**
takeOrderErrorCount3	O	O	O	O	O	O	O	**7**
takeOrderTotalTime3	O	X	O	O	O	O	O	**6**
persevTakeOrder3	X	O	O	X	O	X	O	**4**
takeOrderButtonTime3	X	X	O	O	O	X	X	**3**
repeatFoodErrors	X	X	X	O	X	X	X	**1**
dilemma_x	X	O	X	X	O	O	O	**4**
first_bell3	O	O	O	X	O	X	X	**4**
kitchen_time3	X	X	X	X	O	O	O	**3**
deliveryErrors3	X	X	O	X	X	X	X	**1**
persevDeliveryErrors3	O	O	X	X	O	X	X	**3**
deliveryTime3	X	O	O	O	X	O	O	**5**
candy_pretip3	X	X	X	X	O	X	X	**1**
candy_posttip3	X	X	O	X	O	X	X	**2**
tip_binary3	X	X	O	X	X	X	X	**1**
attend_table3	X	X	X	O	X	X	X	**1**
transactionTime3	X	O	O	O	X	X	X	**3**
totalTime3	O	X	O	O	X	X	O	**4**
greet_boolean4	X	X	X	X	O	X	X	**1**
takeOrderErrorCount4	X	X	O	X	O	X	X	**2**
takeOrderTotalTime4	X	X	O	O	X	X	X	**2**
takeOrderButtonTime4	X	X	O	O	X	X	X	**2**
deliverErrors4	X	X	X	O	X	X	X	**1**
repeatFoodErrors4	X	X	X	O	X	X	X	**1**
dilemma_y	O	X	X	X	X	X	X	**1**
deliveryTime4	X	X	X	O	X	X	X	**1**
Variables	**14**	**15**	**25**	**16**	**22**	**12**	**14**	

O: La variable pertenece al modelo

X: La variable no pertenece al modelo

Resumen métricas de evaluación: Base automática

Tabla C.9: Métricas asociadas a las clases de los modelos de SVM para la base automática, muestra de entrenamiento

Modelos	Recall				F1-Score			
	0	**1**	**2**	**3**	**0**	**1**	**2**	**3**
Unique	1	1	1	1	1	1	1	1
Uncorr	1	1	0.88	1	1	0.94	0.93	1
Mi	1	1	1	1	1	1	1	1
MiCorr	1	0.88	0.88	1	0.89	0.93	0.93	1
Forward Unique	1	1	0.88	1	1	0.94	0.93	1
Forward Uncorr	1	1	0.88	1	1	0.94	0.93	1
Forward Mi	1	1	0.80	1	1	0.94	0.93	1
Forward MiCorr	1	1	1	1	1	1	1	1
Backward Unique	1	1	1	1	1	1	1	1
Backward Uncorr	0.88	1	0.88	1	0.93	0.89	0.93	1
Backward Mi	1	1	1	1	1	1	1	1
Backward MiCorr	1	0.88	0.88	1	0.89	0.93	0.93	1

Tabla C.10: Métricas asociadas a las clases de los modelos de SVM para la base automática, muestra de test

Modelos	Recall				F1-Score			
	0	**1**	**2**	**3**	**0**	**1**	**2**	**3**
Unique	0.50	1	0	0	0.67	1	0	0
Uncorr	1	1	0	0.50	0.80	1	0	0.50
Mi	0.50	0	0	0	0.67	0	0	0
MiCorr	0.50	0	1	0.50	0.40	0	1	0.67
Forward Unique	1	1	0	0.50	1	0.67	0	0.50
Forward Uncorr	1	0	1	0.50	1	0	1	0.50
Forward Mi	1	1	0	0	1	0.50	0	0
Forward MiCorr	0.50	1	0	0	0.40	0.67	0	0
Backward Unique	0.50	0	0	0	0.67	0	0	0
Backward Uncorr	1	1	1	0.50	0.80	1	1	0.67
Backward Mi	0.50	1	0	0	0.67	1	0	0
Backward MiCorr	0.50	0	1	0.50	0.40	0	1	0.67

Tabla C.11: Métricas asociadas a las clases de los modelos de Random Forest para la base automática, muestra de entrenamiento

Modelos	Recall				F1-Score			
	0	**1**	**2**	**3**	**0**	**1**	**2**	**3**
Unique	1	1	1	1	1	1	1	1
Uncorr	1	1	1	1	1	1	1	1
Mi	1	1	1	1	1	1	1	1
MiCorr	1	1	1	1	1	1	1	1
Forward Unique	1	1	1	1	1	1	1	1
Forward Uncorr	1	1	1	1	1	1	1	1
Forward Mi	1	1	1	1	1	1	1	1
Forward MiCorr	1	1	1	1	1	1	1	1
Backward Unique	1	1	1	1	1	1	1	1
Backward Uncorr	1	1	1	1	1	1	1	1
Backward Mi	1	1	1	1	1	1	1	1
Backward MiCorr	1	1	1	1	1	1	1	1

Tabla C.12: Métricas asociadas a las clases de los modelos de Random Forest para la base automática, muestra de test

Modelos	Recall				F1-Score			
	0	**1**	**2**	**3**	**0**	**1**	**2**	**3**
Unique	1	1	0	0.50	1	1	0	0.50
Uncorr	1	0	0	0.50	0.80	0	0	0.40
Mi	1	1	1	0.50	1	1	0.67	0.67
MiCorr	1	1	1	0	0.67	1	1	0
Forward Unique	0.50	0	0	0	0.50	0	0	0
Forward Uncorr	1	0	0	0	1	0	0	0
Forward Mi	0.50	1	0	0	0.50	1	0	0
Forward MiCorr	1	1	1	0	0.67	1	1	0
Backward Unique	0.50	1	0	0	0.67	1	0	0
Backward Uncorr	1	1	0	0.50	1	1	0	0.50
Backward Mi	0.50	1	0	0	0.67	1	0	0
Backward MiCorr	1	1	1	0	0.67	1	1	0

Tabla C.13: Métricas asociadas a las clases de los modelos de Naive Bayes para la base automática, muestra de entrenamiento

Modelos	Recall				F1-Score			
	0	**1**	**2**	**3**	**0**	**1**	**2**	**3**
Unique	0.88	0.88	0.88	0.88	0.82	0.93	0.93	0.82
Uncorr	0.88	1	0.88	1	0.88	0.94	0.93	1
Mi	1	1	1	0.88	0.94	1	1	0.93
MiCorr	1	0.88	0.88	0.75	0.94	0.88	0.88	0.80
Forward Unique	1	1	1	1	1	1	1	1
Forward Uncorr	0.75	1	0.88	1	0.80	0.94	0.88	1
Forward Mi	1	1	1	1	1	1	1	1
Forward MiCorr	1	0.88	1	1	0.94	0.93	1	1
Backward Unique	1	1	1	0.88	0.94	1	1	0.93
Backward Uncorr	0.88	1	0.75	1	0.82	0.94	0.86	1
Backward Mi	1	0.25	0.75	0.25	0.76	0.40	0.55	0.36
Backward MiCorr	1	1	1	1	1	1	1	1

Tabla C.14: Métricas asociadas a las clases de los modelos de Naive Bayes Forest para la base automática, muestra de test

Modelos	Recall				F1-Score			
	0	**1**	**2**	**3**	**0**	**1**	**2**	**3**
Unique	1	0	0	0.50	0.80	0	0	0.50
Uncorr	1	0	0	0.50	0.67	0	0	0.50
Mi	1	0	0	0	0.80	0	0	0
MiCorr	1	1	1	0	0.67	1	1	0
Forward Unique	1	0	0	0.50	0.80	0	0	0.50
Forward Uncorr	1	1	0	0	0.80	1	0	0
Forward Mi	1	0	1	0.50	1	0	0.50	0.67
Forward MiCorr	1	0	1	0.50	0.67	0	1	0.67
Backward Unique	1	0	0	0.50	0.67	0	0	0.50
Backward Uncorr	1	1	0	0	0.80	1	0	0
Backward Mi	1	0	0	0.50	0.67	0	0	0.67
Backward MiCorr	1	0	1	0	0.67	0	0.67	0