UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

# CLUSTERING DE RECETAS CULINARIAS GENERADAS POR GIUSEPPE

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL EN COMPUTACIÓN

JUAN ANDRÉS MORENO CORTEZ

PROFESOR GUÍA:
JORGE PEREZ ROJAS

MIEMBROS DE LA COMISIÓN:
ALEXANDRE BERGEL
KARIM PICHARA BAKSAI
JOSÉ M. SAAVEDRA RONDO

SANTIAGO DE CHILE
2020

# Resumen

The Not Company es una empresa de *Foodtech* chilena cuya misión fundamental es utilizar solo ingredientes vegetales para reproducir productos alimenticios que comúnmente utilizan ingredientes animales. El ciclo de desarrollo de un producto nuevo comienza con el sistema de Inteligencia Artificial de NotCo: Giuseppe. Giuseppe genera miles de combinaciones de ingredientes vegetales (en este contexto, las fórmulas) que buscan reproducir el sabor, textura, aroma y retro gusto de un alimento en particular. Posteriormente, el equipo de *Machine Learning* de la empresa selecciona un pequeño subconjunto de fórmulas, las que finalmente son entregadas al equipo de Chefs, quienes las analizan y cocinan, para dar con aquella fórmula que reproduce más fielmente el producto objetivo.

El presente documento detalla el desarrollo de un algoritmo que apoya un nuevo proceso de selección de fórmulas. El algoritmo desarrollado encuentra automáticamente familias de fórmulas con *perfiles sensoriales* similares, y que en consecuencia, comparten un conjunto de ingredientes en común. Para lograr esto, representamos cada fórmula por un vector numérico de *características moleculares* las cuales buscan reproducir indirectamente el *perfil sensorial*[1] de la fórmula. Posteriormente, reducimos la dimensión de los vectores utilizando el algoritmo de reducción de dimensionalidad UMAP [14] y luego utilizamos el algoritmo de *clustering* HDBSCAN [3] para identificar, en el espacio reducido, las familias de fórmulas con *perfiles sensoriales* equivalentes.

Para validar con los Chefs si efectivamente los *clusters* obtenidos corresponden a fórmulas con *perfiles sensoriales* similares, utilizamos la técnica de validación de *clusters* con expertos propuesta en el trabajo de Hatzivassiloglou y McKeown [7]. Aplicamos la metodología de validación sobre 5 alimentos distintos, cárnicos y no cárnicos. El algoritmo desarrollado obtuvo un puntaje $F_1$ promedio de 0,55, este número aumenta a 0,64 si solo consideramos los alimentos no cárnicos. Encapsulamos el algoritmo en una herramienta de línea de comando la cual, dado un conjunto de fórmulas, retorna las $k$ fórmulas con *perfiles sensoriales* distintos pero más similares al producto objetivo en cuestión. Adicionalmente, para cada una de las $k$ familias de fórmulas *sensorialmente equivalentes* asociadas entregamos las $m$ mejores fórmulas.

Las fórmulas seleccionadas por la herramienta de línea de comando son subidas a una herramienta web interna de la empresa en la cual los Chefs pueden participar activamente del proceso de selección de fórmulas, a través de la exploración de las familias de fórmulas seleccionadas.

La solución desarrollada es de carácter exploratoria, considera el conjunto completo de fórmulas y su resultado permite involucrar completamente a los Chefs en el proceso de selección de fórmulas. El resultado obtenido por la solución desarrollada en el proceso de validación con expertos entrega evidencia de que el vector de características moleculares puede ser utilizado para representar las fórmulas y encontrar aquellas con *perfiles sensoriales* similares.

[1] Sabor, color, aroma, textura y retro gusto de un alimento.

*A mi Mamá, a mi Papá, a mis hermanos, a todos quienes quiero como mi familia y a mi belle. Gracias, gracias, gracias por su cariño, su contención y su apoyo.*

# Tabla de contenido

# Índice de tablas

# Índice de ilustraciones

# 1. Introducción

Para una mejor comprensión de este documento primero describimos en la Sección 1.1 a la empresa The Not Company, su razón de ser y el problema central que busca resolver: reproducir alimentos convencionales utilizando solo ingredientes vegetales. Posteriormente, en la Sección 1.2 presentamos a Giuseppe, el sistema de Inteligencia Artificial que encuentra las combinaciones de ingredientes vegetales que reproducen fielmente algún otro alimento convencional específico.

En la Sección 1.2 describimos también el proceso de selección de fórmulas, el cual consiste en seleccionar manualmente las combinaciones de ingredientes vegetales generadas por Giuseppe (las fórmulas) que serán entregadas a los Chefs para ser cocinadas y degustadas. El trabajo desarrollado se centró en el proceso de selección de fórmulas y por tanto concluimos la Sección 1.2 con los problemas identificados con el proceso de selección actual.

Finalmente, en la Sección 1.3 presentamos los objetivos específicos del trabajo desarrollado en este trabajo de título. Los objetivos planteados buscan mejorar el proceso de selección de fórmulas actual mediante la resolución de los problemas identificados. Presentamos además una breve descripción de la solución desarrollada junto con los resultados obtenidos por esta.

## 1.1. NotCo: motivación y problemas

### 1.1.1. The Not Company: *raison d'être*

Se estima que el reino de las plantas contiene mas de 300,000 especies distintas [18], de las cuales cerca de 20,000 son consideradas comestibles por el ser humano [6]. A pesar de esto, la componente vegetal de la dieta del ser humano se ha basado en aproximadamente, 20 especies de plantas distintas [6]. Existen, por ende, recursos no aprovechados.

Por otro lado, y de manera opuesta, en la industria ganadera existen recursos mal aprovechados. Una muestra de ello son los litros de agua necesarios para producir una kilo caloría utilizando carne de vaca y una kilo caloría utilizando vegetales: $10,19$ y $1,34$ [8], respectivamente.

The Not Company (desde ahora en adelante NotCo) es una empresa de *Foodtech* que nace de esta problemática y con la misión de cambiar la forma en que los seres humanos producen su comida. NotCo se dedica a desarrollar tecnología e investigación con el fin de encontrar maneras de combinar ingredientes vegetales para replicar perfectamente alimentos de origen animal como la leche, el queso o la carne. NotCo propone una nueva forma de producir comida que elimina al animal, y los recursos necesarios asociados a este, del proceso de producción de comida.

### 1.1.2. La Combinación Perfecta de Ingredientes Vegetales

El problema central de NotCo es lograr encontrar la combinación de ingredientes vegetales perfecta, es decir, el conjunto de ingredientes vegetales y las proporciones en que deben combinarse estos para reproducir perfectamente algún otro alimento.

Para ejemplificar la complejidad de este problema, supongamos que bastase con encontrar

los ingredientes vegetales que deben estar presentes en un fórmula para reproducir algún otro alimento. Si existiesen tan solo 10 ingredientes vegetales distintos el problema consiste en probar 1024 combinaciones distintas de ingredientes, lo que quizás es lograble en un margen de tiempo razonable si se contase con una cantidad no razonable de Chefs para preparar cada una de las 1024 fórmulas.

Si solo el $0,5\,\%$ de las $\sim$ 300,000 especies de plantas que existen fueran comestibles, la cantidad de combinaciones existentes de ingredientes vegetales ya supera con creces la cantidad estimada de átomos en el universo. Comprobamos entonces que resulta infactible realizar una búsqueda exhaustiva sobre el espacio de las combinaciones de ingredientes vegetales para dar con una fórmula que reproduzca perfectamente algún otro alimento.

## 1.2. Giuseppe: IA aplicada a alimentos

### 1.2.1. Giuseppe: Un explorador Inteligente

Giuseppe es un algoritmo de Inteligencia Artificial propietario desarrollado en NotCo el cual explora, de una manera inteligente, el espacio de combinaciones de ingredientes vegetales posibles para dar con aquellas formulaciones que reproducen de mejor manera el sabor, color, aroma y textura de un alimento de origen animal específico.

Para facilitar la comprensión de este documento y del trabajo realizado, introduciremos, a continuación, tres conceptos centrales.

**Definición 1. Perfil Sensorial de un alimento** : Descripción de un alimento en términos de su sabor, color, aroma, textura y retro gusto. Diremos que dos alimentos distintos tienen *perfiles sensoriales* similares si estos tienen un sabor, color, aroma, textura y retro gusto similares.

**Definición 2. Vector de características moleculares** : Vector de características numéricas que describen física, química y nutricionalmente un alimento (vegetal o no vegetal).

**Definición 3. *Target*** : Vector de *características moleculares* de un alimento objetivo que se desea reproducir. Corresponde a la entrada de Giuseppe.

**Definición 4. Fórmula** : Resultado entregado por Giuseppe. Se compone de dos partes: un conjunto de ingredientes vegetales y en las proporciones que deben combinarse para reproducir el *target* entregado como entrada a Giuseppe, y un vector de *características moleculares* que representa a dicha fórmula. El vector de *características moleculares* se construye como la suma ponderada de los vectores de características de los ingredientes vegetales y las proporciones de cada uno.

Ejecutar a Giuseppe comprende entregar como entrada el *target* asociado al alimento de que se desea reproducir y comenzar el proceso de generación de fórmulas. El resultado de este proceso de generación es un conjunto de miles de fórmulas distintas que buscan replicar

el *target* pero basándose solo en ingredientes vegetales.

Giuseppe logra reducir el tamaño del espacio de soluciones al orden de los miles. Aun así, resulta infactible probar en la cocina cada una de estas miles de fórmulas generadas para encontrar aquella que reproduce más fielmente el *perfil sensorial* de *target*. Es por esto que, una vez concluido el proceso de generación y, por ende obtenido el conjunto de miles de fórmulas de resultado, comienza el proceso de selección de fórmulas.

### 1.2.2. El Proceso de Selección de fórmulas

El proceso de selección de fórmulas consiste en determinar el subconjunto mínimo de fórmulas del resultado de Giuseppe. Un subconjunto fórmulas es mínimo si contiene aquellas fórmulas que con alta probabilidad, una vez preparadas por el equipo de Chefs, reproducirán satisfactoriamente el *perfil sensorial* del *target* asociado.

### 1.2.3. Situación Actual

A la fecha de desarrollo de este proyecto, la selección del subconjunto de fórmulas es un proceso manual que lleva a cabo el equipo de *Machine Learning*. El proceso consiste en ordenar todas las fórmulas generadas por su similitud al *target* y luego, mediante una inspección superficial de las fórmulas ordenadas, seleccionar un subconjunto de fórmulas similares al *target* con ingredientes distintos y con mayor probabilidad de éxito.

De esta manera, el equipo de *Machine Learning* guía su selección utilizando la similitud como un *proxy* para medir la probabilidad de éxito de cada fórmula. Esta manera de seleccionar logra reducir el tamaño del espacio de soluciones del problema desde los miles a las decenas. Sin embargo, este proceso tiene defectos y por lo tanto existe espacio para mejorar.

### 1.2.4. Problemas de la Situación Actual

La similitud que utiliza el equipo de *Machine Learning* para ordenar las fórmulas en el proceso de selección es en realidad la distancia Euclidiana entre cada fórmula y su *target* asociado. Esta distancia entre fórmula y target se calcula sobre un espacio de alta dimensión (el espacio de las *características moleculares*) y por ende, sufre del llamado *curse of dimensionality*. La consecuencia directa del *curse of dimensionality* en este contexto es que todas las fórmulas resultan, en promedio, igual de similares al *target* en cuestión. En consecuencia, la similitud no es una medida efectiva, por sí sola, para juzgar la probabilidad de éxito de una fórmula, pues todas resultan, en promedio, igual de similares al *target* en cuestión.

El proceso de selección de fórmulas requiere, por su naturaleza, conocimiento gastronómico, pues consiste en identificar aquellas fórmulas que una vez *cocinadas* reproducirán más fielmente el *perfil sensorial* del target en cuestión. El problema es que el proceso de selección de fórmulas está a cargo del equipo de *Machine Learning* únicamente y ningún integrante del equipo tiene conocimiento gastronómico. Luego, la única manera que tienen para juzgar la probabilidad de éxito de una fórmula en la cocina es evaluando la similitud de la fórmula a su *target* asociado, la cual sufre de las dolencias explicadas anteriormente. Específicamente, el problema aquí es que los Chefs, quienes sí tienen conocimiento gastronómico, no están involucrados en el proceso de selección de fórmulas.

Por último, utilizar una lista ordenada de fórmulas por similitud permite ver relaciones de orden entre las fórmulas generadas pero no permite apreciar relaciones mas complejas que existen dentro de todo el conjunto de fórmulas generadas. Ejemplo de estas relaciones complejas son las familias de fórmulas con *perfiles sensoriales* similares. Dada la manera en que Giuseppe explora el espacio de ingredientes este tiende a generar distintos conjuntos bases de ingredientes y miles de variaciones para cada uno. Cada conjunto base de ingredientes caracteriza a una familia de fórmulas con *perfiles sensoriales* similares, y cada variación del conjunto base se traduce en una fórmula integrante de aquella familia.

## 1.3. Mejorando el proceso de Giuseppe

### 1.3.1. Objetivos Específicos del Trabajo

El objetivo principal de este trabajo de título fue diseñar e implementar un algoritmo que apoye un nuevo proceso de selección de fórmulas que sea de carácter exploratorio, que involucre a los Chefs, y que logre identificar las distintas familias de fórmulas con *perfiles sensoriales* similares dentro del conjunto completo de fórmulas generadas como resultado por Giuseppe para un *target* en particular.

Los objetivos específicos de este trabajo de título fueron:

1. Desarrollar un algoritmo de selección que identifique las familias de fórmulas generadas para un mismo *target* con *perfiles sensoriales* similares
2. Incrementar la diversidad, en términos de ingredientes, de las fórmulas consideradas en el proceso de selección con respecto a la situación actual
3. Presentar el resultado del algoritmo a desarrollar de manera tal que los Chefs puedan explorar el conjunto de fórmulas generado a través de las familias de fórmulas identificadas.

### 1.3.2. Descripción de la Solución Desarrollada

La solución presentada en este documento consistió en un algoritmo que, dado el conjunto completo de fórmulas generado por Giuseppe para un *target* especifico, detecta automáticamente las familias de fórmulas con *perfiles sensoriales* similares y con ingredientes en común como *clusters* de densidades variables, representando cada fórmula por su vector de *características moleculares*.

Para encontrar los *clusters* de familias de fórmulas utilizamos el algoritmo de *clustering* jerárquico y basado en densidad *Hierarchical Density Based Spatial Clustering for Applications with Noise*, desarrollado por Campello et al. [3]. HDBSCAN identifica las regiones localmente densas y topológicamente persistentes del espacio como *clusters* distintos, por ende no asume una cantidad definida de *clusters* a encontrar ni tampoco una forma de *clusters* especifica.

Para lidiar con el *curse of dimensionality* asociado al cálculo de distancias en el espacio de características y al mismo tiempo realzar las regiones densas de puntos, utilizamos el algoritmo de reducción de dimensionalidad *Uniform Manifold Approximation and Projection*, desarrollado por McInnes et al. [14]. UMAP fue desarrollado con un fundamento matemático que asegura una equivalencia teórica entre los espacios topológicos de los datos en alta y baja

dimensión. [14]

Para validar si las agrupaciones de fórmulas detectadas corresponden efectivamente a conjuntos de fórmulas con *perfiles sensoriales* similares utilizamos el proceso de validación de *clusters* con expertos propuesto en el trabajo de Hatzivassiloglou y McKeown [7].

### 1.3.3. Resultados Obtenidos

Mediante el proceso de validación de *clusters* de Hatzivassiloglou y McKeown [7] comparamos el rendimiento promedio de nuestra solución con el rendimiento de un conjunto de Chefs sobre la tarea *"encontrar agrupaciones disjuntas de fórmulas con perfiles sensoriales similares"* para 5 *targets* distintos. La solución desarrollada obtuvo un puntaje promedio $F_1$ de 0,55, una precisión promedio de 0,49 y una exhaustividad promedio de 0,64, considerando el resultado obtenido por el grupo de Chefs como el resultado ideal de la tarea.

El producto final de este proyecto es un paquete de *Python* el cual contiene una herramienta de línea de comando que recibe como entrada un conjunto de fórmulas generadas para un *target* $t$, identifica todas las familias de fórmulas presentes en el conjunto y retorna como resultado las $k$ fórmulas de familias distintas con mayor similitud a $t$, junto con los $m$ integrantes más similares a $t$ de cada una de las $k$ familias.

Los resultados se hacen disponibles a través de una herramienta web interna de la empresa. En ella, los Chefs son capaces de explorar las familias de fórmulas identificadas para cada *target*, combinando efectivamente su criterio experto con el resultado completo generado por Giuseppe para encontrar la fórmula que reproduce más fielmente el color, aroma, textura y sabor del *target* asociado.

Lo que queda de este documento se estructura de la siguiente manera: en el capítulo 2 introducimos y explicamos los algoritmos y conceptos centrales utilizados para el desarrollo de este proyecto. En el capítulo 3 describimos el problema abordado, la relevancia de contar con una solución, los requisitos técnicos y los requisitos de calidad de esta. En el capítulo 4 presentamos la solución desarrollada en detalle con su diseño y justificaciones correspondientes, además de un análisis de los *clusters* encontrados por la solución para un conjunto de *targets* de prueba. En el capítulo 4 mostramos además los resultados obtenidos por el algoritmo. Finalmente en el capítulo 5 presentamos las conclusiones obtenidas del proyecto junto con el trabajo futuro.

# 2. Marco Teórico

El desarrollo de la solución descrita en este documento se puede dividir en dos partes fundamentales: reducción de dimensionalidad y llevar a cabo una tarea de *clustering*. En la presente sección del documento explicaremos, con la profundidad necesaria, los distintos algoritmos utilizados, junto con breves justificaciones de su aplicabilidad para construir una solución que resuelve el problema propuesto en este proyecto.

## 2.1. *Curse of Dimensionality*

El espacio de características escogido para representar cada fórmula generada por Giuseppe contiene más de 50 dimensiones. A continuación explicaremos las consecuencias que conlleva trabajar con datos de alta dimensión y por qué es necesario reducir la dimensión de estos.

El termino *Curse of Dimensionality* fue acuñado por el matemático Richard Ernest Bellman en 1961 en su libro *"Adaptive Control Processes: a guided tour"* [2] para referirse al hecho de que muchos algoritmos que funcionan bien con datos de baja dimensión se vuelven intratables al considerar datos de alta dimensión.

En el trabajo de Pedro Domingos [4], se detallan algunas de las consecuencias del *Curse of Dimensionality* en el contexto del aprendizaje de máquinas:

1. **Dificultad para generalizar aumenta exponencialmente con la dimensión de los datos**: Un conjunto de prueba de tamaño fijo cubre una pequeña fracción del espacio que los contiene. Si consideramos, por ejemplo, el espacio de los vectores binarios de largo 100 y un conjunto de entrenamiento de $10^{12}$ puntos, este último solo cubre una pequeña fracción equivalente al $10^{-18}$ del total del espacio.
2. **El razonamiento basado en similitud pierde validez**: El vecindario de los $k$ vecinos más cercanos de un punto $x$ en alta dimensión puede describirse como la hiper-esfera centrada en $x$ que contiene a los $k$ vecinos más cercanos. A medida que la dimensión de los datos crece, el volumen de la hiper-esfera comienza a crecer cada vez más y sucede que la mayoría de los vecinos se concentra hacia el borde la hiper-esfera. En consecuencia, en altas dimensiones, los $k$ vecinos más cercanos de $x$ se encuentran todos a casi la misma distancia de $x$.

El trabajo de Pedro Domingos [4] también se refiere a un fenómeno que contra resta en cierta medida los efectos del *Curse of Dimensionality*: el *Blessing of non-uniformity*. El *Blessing of non-uniformity* nota que en la mayoría de los casos reales los datos no se encuentran distribuidos uniformemente en el espacio, sino que más bien tienden a concentrarse en o en una aproximación de un *Manifold* de menor dimensión [4].

Todd Rowland define el concepto de *Manifold* en su artículo *"Manifold"* en el sitio web *mathworld* [17] como un espacio topológico localmente Euclidiano. Rowland se vale también de la explicación de Poincaré quien describió los *Manifolds* como objetos que a bajas escalas parecen ser planos. Un ejemplo de *Manifold* es la esfera terrestre.

## 2.2. Reducción de Dimensionalidad

Para lidiar con el *Curse of Dimensionality* se utilizan algoritmos específicos que logran, de una u otra manera, reducir la cantidad de dimensiones de los datos y, que además, preservan alguna propiedad de interés que poseen los datos en la dimensión original. A esta tarea se le llama reducción de dimensionalidad.

Específicamente, la reducción de dimensionalidad es un proceso que dado un conjunto de datos $D$, con puntos $n$-dimensionales, busca una representación de ellos $m$-dimensional con $m \ll n$, que preserve la estructura relevante del conjunto de datos original $D$.

La revisión sistemática de van der Maaten et al. [13] sobre reducción de dimensionalidad, divide los algoritmos de reducción de dimensión en dos categorías: lineales y no lineales. La revisión se refiere principalmente a los algoritmos de la segunda categoría argumentando que estos algoritmos obtienen mejores resultados que sus contra partes lineales sobre conjuntos de datos reales. La razón de esta diferencia nace del hecho que los conjuntos de datos reales (i.e. no sintéticos) suelen formar *Manifolds* complejos y no lineales. Aun así, la revisión concluye que no existe claridad sobre si una categoría de algoritmo funciona mejor que la otra en todos los casos.

## 2.3. UMAP: Uniform Manifold Approximation and Projection

UMAP es un algoritmo de reducción de dimensionalidad no lineal desarrollado por McInnes et al. [14] que construye representaciones topológicas de los datos tanto en la dimensión original como en baja dimensión. En el caso de los datos en alta dimensión, la estructura topológica busca aproximar el *Manifold* sobre el cual, se supone, se encuentran los datos. La representación topológica de los datos en baja dimensión se inicializa con valores aleatorios. Una vez construidas las representaciones topológicas de los datos en alta y baja dimensión, se optimiza la posición de los puntos en baja dimensión para minimizar la entropía cruzada entre las representaciones topológicas de los datos en alta y baja dimensión.

Una de las ventajas de UMAP es que cuenta con un fundamento matemático que asegura que una serie de propiedades topológicas relevantes del *Manifold* que describen los puntos en alta dimensión, son efectivamente capturadas por la representación topológica construida para los datos en baja dimensión.

En lo que resta de esta subsección explicaremos los aspectos relevantes del funcionamiento de UMAP, para comprender cómo utilizar el método y elegir correctamente sus hiperparámetros. Las definiciones a continuación serán útiles para comprender con mayor facilidad el funcionamiento del algoritmo.

**Definición 5. Bola**: Conjunto definido con la geometría de una esfera de radio $r$. Contiene a todo elemento $x$ que se encuentre a una distancia menor o igual a $r$ del centro de la bola.

**Definición 6. Espacio Métrico**: Conjunto $C$ dotado de una función de distancia $g$, donde $g(x, y)$ es la distancia entre los puntos $x \in C$ e $y \in C$.

**Definición 7. Distancia Geodésica**: Distancia mínima entre dos puntos sobre una superficie.

**Definición 8. Espacio Topológico**: Johaness et al. [12] define un espacio topológico como un conjunto $X$ junto con una colección de subconjuntos abiertos $\tau$ que satisfacen cuatro condiciones:

1. El conjunto vacío está en $\tau$
2. $X$ está en $\tau$
3. La intersección de una cantidad finita de conjuntos de $\tau$ también está en $\tau$
4. La unión de una cantidad arbitraria de conjuntos de $\tau$ también está en $\tau$

**Definición 9. Cobertura Abierta de un espacio topológico**: Colección de conjuntos abiertos cuya unión contiene a todo el espacio topológico.

El proceso de construcción de la representación topológica de los datos (en alta o en baja dimensión) se puede explicar con dos pasos fundamentales:

1. **Aproximar el *Manifold*** sobre el cual, se supone, se encuentran los datos, pero de manera local, es decir, en torno a cada punto.
2. **Construir la representación topológica** del *Manifold* como la unión de las aproximaciones locales.

### 2.3.1. Aproximación del *Manifold*

De manera similar al caso de los *Laplacian Eigenmaps* [1], UMAP requiere, para funcionar correctamente, que los datos se encuentren uniformemente distribuidos sobre el *Manifold*. En la práctica, esto rara vez se cumple con conjuntos de datos no sintéticos. Para lidiar con esto, UMAP define una medida de distancia particular en torno a cada punto tal que los puntos se distribuyen aproximadamente de manera uniforme en el *Manifold* con respecto a estas distancias.

Si asumimos que los puntos del conjunto de entrada $X$ se encuentran uniformemente distribuidos sobre el *Manifold*, entonces cualquier bola de volumen fijo debería contener, aproximadamente, la misma cantidad de puntos, independiente de dónde se centre esta en el *Manifold*. De la misma manera, una bola centrada en un punto $x \in X$, que contenga exactamente a los $k$ vecinos más cercanos de $x$ debería tener el mismo volumen, independiente de la elección de $x$.

El algoritmo de aproximación del *Manifold* uniforme define una medida de distancia distinta para cada $x \in X$ tal que la bola unitaria (de radio unitario y, por ende, de volumen fijo) que rodea a cada $x$ contiene siempre a los $k$ vecinos más cercanos. Cabe destacar aquí que estas bolas son unitarias solo con respecto a las medidas de distancia que definiremos en torno a cada $x \in X$, y no con respecto a la medida de distancia heredada del espacio ambiente que contiene al Manifold.

Para formalizar esto, consideremos un punto $x_i \in \mathcal{M}$, donde $\mathcal{M}$ es el *Manifold* sobre el cual, se supone, se encuentran los puntos. Además sea $B$ la bola abierta centrada en $x_i$. Se

[4] `https://umap-learn.readthedocs.io/en/latest/how_umap_works.html`

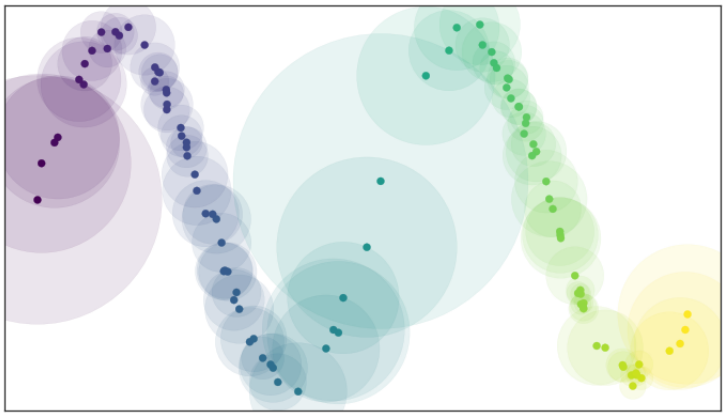

Figura 1: *Ejemplo en dos dimensiones de bolas unitarias construidas considerando los 5 vecinos más cercanos. Si bien los radios en el espacio ambiente no son unitarios, sí lo son con respecto a las nociones de distancia locales definidas en torno a cada punto. Imagen tomada de la documentación de la implementación de UMAP* [4].

define la distancia geodésica aproximada *dentro de la bola* $B$ entre dos puntos $x_i$ y $x_j$ como

$$d_{\mathcal{M}}(x_i, x_j) = \frac{1}{r_B} d_{\mathcal{R}^n}(x_i, x_j),$$

donde $r_B$ es el radio (medido en el espacio ambiente que contiene al *Manifold*) de la bola $B$ con centro $x_i$ que contiene a los $k$ vecinos más cercanos de $x_i$ y $d_{\mathcal{R}^n}$ es la medida de distancia heredada del espacio que contiene al *Manifold*.

Es importante notar aquí que la distancia entre dos puntos $x_{i^*}$, $x_{j^*}$ que pertenecen a la bola unitaria $B$ centrada en $x_i$ no está definida. Sin embargo, y como veremos luego, existen dos definiciones de distancia entre $x_{i^*}$ y $x_{j^*}$, una definida con respecto a la bola centrada en $x_{i^*}$ y otra definida con respecto a la bola centrada en $x_{j^*}$.

La cantidad $k$ de vecinos más cercanos que debe contener cada bola unitaria definida en torno a cada punto, es uno de los hiperparámetros de UMAP. Intuitivamente, este hiperparámetro se utiliza para definir qué tan granular debe ser la aproximación del *Manifold* en alta dimensión. Valores pequeños de $k$ harán que UMAP se concentre en la estructura local del *Manifold*, en detrimento de la estructura global. La Figura 1 muestra las bolas unitarias construidas cuando se consideran los 5 vecinos más cercanos para cada punto.

En el contexto de este proyecto, la naturaleza del hiperparámetro $k$ resulta conveniente pues este puede entenderse como, aproximadamente, la cantidad $k$ de fórmulas que deseamos considerar como las más similares para cada fórmula. De esta manera, valores mayores de $k$ harán menos restrictiva nuestra noción de similitud local y, por ende, el *Manifold* aproximado capturará relaciones más globales entre las fórmulas.

Si bien esta manera de definir distancias de manera local hace cierta la suposición de distribución uniforme, también trae consigo otro problema: cada bola $B$ asociada a cada punto $x_i$ define un espacio métrico distinto con métricas que pueden no ser compatibles. Se tiene, en esencia, un conjunto de espacios métricos distintos que deben ser unidos de alguna manera para crear una estructura consistente y global de los datos.

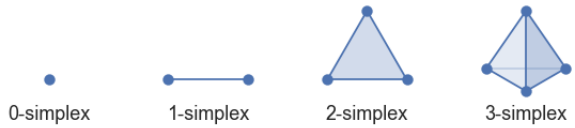


Figura 2: *Distintos n-símplices. Imagen tomada de la documentación de la implementación de UMAP.*

### 2.3.2. Construcción de la representación Topológica

Para combinar los distintos espacios métricos construidos en torno a cada punto (las bolas) en una sola estructura global, UMAP convierte cada espacio métrico en conjuntos difusos de *símplices*, para finalmente tomar la unión de estos conjuntos y obtener una representación global de los datos.

El fundamento de la correctitud de este procedimiento esta basado en topología algebraica y teoría de categorías. Para efectos de este proyecto bastará con explicar cómo se construyen los conjuntos difusos de *símplices* y cómo se toma la unión de estos. Los lectores interesados en los fundamentos teóricos pueden consultar la publicación original de McInnes et al. [14].

Los *símplices* son utilizados para construir objetos $m$-dimensionales. Geométricamente un $m$-*simplex* es un *simplex*[5] de $m$ dimensiones y se construye tomando la cobertura convexa de $m+1$ puntos separados. La Figura 4 muestra cómo se ven los 4 primeros *símplices* de menor dimensión.

Una colección de *símplices* pegados de una manera en particular forman un *simplicial complex*. Dada una cobertura abierta de un espacio topológico $\mathcal{T}$ es posible construir un tipo particular de *simplicial complex*, un *Čech complex*. El proceso de construcción de un *Čech complex* se explica a continuación.

- Cada elemento en la cobertura abierta es un 0-*simplex*
- Se crea un 1-*simplex* para cada par de elementos en la cobertura abierta con intersección no nula
- Se crea un 2-*simplex* entre cada trio de elementos en la cobertura abierta con intersección no nula
- Se continúa este proceso iterativamente formando *símplices* de mayor dimensión

La Figura 3 muestra el *Čech complex* obtenido a partir de las bolas unitarias mostradas en la Figura 1.

Por sencillo que parezca esta manera de construir un *simplicial complex*, el Teorema del Nervio [16] asegura que el *Čech complex* construido es equivalente de manera homotópica a la cobertura abierta de $\mathcal{T}$. Específicamente, esto quiere decir que existen funciones continuas que permiten transformar la cobertura abierta de $\mathcal{T}$ en el *Čech complex* construido y vice versa.

[5] *Simplex* es el singular de *símplices*.

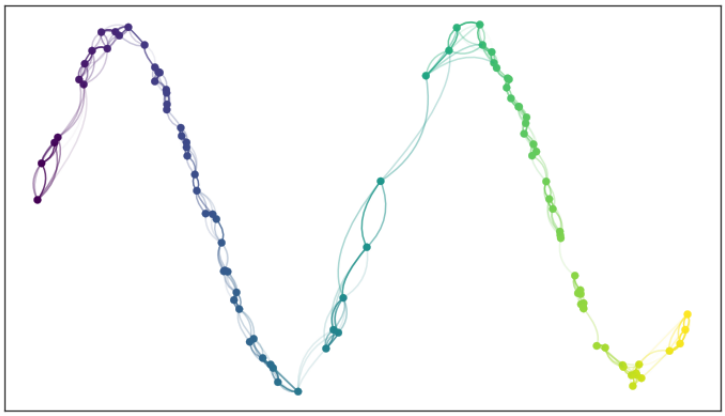

Figura 3: *Čech complex* construido a partir de los conjuntos difusos construidos a partir de las bolas unitarias mostradas en la Figura 1, considerando la suposición de conectividad local. Una mayor intensidad de color en la arista indica mayor probabilidad de existencia.

UMAP construye un *Čech complex* a partir del conjunto de bolas abiertas y así se obtiene una garantía teórica de que la representación construida captura la topología del *Manifold* en alta dimensión. Dado que resulta intratable para un computador construir el *Čech complex* considerando *símplices* de dimensiones mayores a 2, UMAP se basa solo en los *0-símplices* y *1-símplices* los cuales pueden representarse convenientemente como un grafo, donde los nodos son los *0-símplices* y las aristas los *1-símplices*.

Posteriormente, UMAP transforma cada conjunto abierto (cada bola) en un conjunto difuso pues, bajo esta representación, es posible asignar un número entre 0 y 1 a cada 1-*simplex*. Así, si $B_{x_i}$ y $B_{x_j}$ son dos 0-*simplex*, el peso asignado al 1-*simplex* que los conecta es igual al grado de pertenencia de $x_j$ a $B_{x_i}$ y se interpreta como la probabilidad de existencia del 1-*simplex*[6].

La teoría desarrollada para fundamentar UMAP especifica exactamente cómo medir el grado de pertenencia de un punto $x_j$ a la bola unitaria $B$ centrada en el punto $x_i$ con radio $r_B$ en el espacio ambiente:

$$\mu_B(x_j) = \exp\left(-\frac{1}{r_B} d_{\mathcal{R}^n}(p,q)\right) = \exp(-d_{\mathcal{M}}(x_i, x_j)). \tag{1}$$

A la función $\mu_B : X \to [0,1]$ se le denomina *función de pertenencia* del conjunto difuso $B$, pues $\mu_B(x)$ mide el grado de pertenencia del punto $x$ al conjunto difuso $B$. Producto de como está definida la función de pertenencia, un punto cercano al centro de la bola tendrá un valor de pertenencia cercano a 1, mientras que un punto cercano al borde de la bola tendrá un valor de pertenencia cercano a 0. En consecuencia, mientras más lejanos se encuentren los puntos, menor sera la probabilidad de existencia del 1-*simplex* que conecta a sus bolas respectivas.

[6]Es importante notar que también existe otro 1-*simplex* que conecta a $B_{x_i}$ y $B_{x_j}$ pero con un valor asignado igual a la pertenencia de $x_i$ a $B_{x_j}$.

Una complicación que surge con esta manera de medir pertenencia es que, producto del *curse of dimensionality*, la mayoría de los puntos tienden a estar aislados y, por ende, la mayoría de las pertenencias (y, por tanto las probabilidades de existencia de los 1-*símplices*) toma valores cercanos a cero. Para lidiar con esto, UMAP supone que el *Manifold* está localmente conectado.

La suposición de conectividad local se traduce en que la pertenencia de los puntos a una bola cualquiera $B$, centrada en un punto cualquiera $p$ empieza a decaer a partir del primer vecino más cercano de $p$. Considerando esta suposición, definimos, a continuación, la medida de distancia en torno a cada $x_i \in X$.

$$d_i(x_j, x_k) = \begin{cases} d_{\mathcal{M}}(x_j, x_k) - \rho & \text{si } i = j \\ \infty & \text{si no,} \end{cases}$$

donde $\rho$ es la distancia al vecino más cercano de $x_i$. Bajo esta definición, la distancia de un punto a su vecino más cercano es igual a cero, y como consecuencia, la probabilidad de existencia del 1-*simplex* (ver Ecuación 1) que conecta a sus bolas es igual a 1. De esta manera, para cada bola $B_{x_i}$ definida para aproximar localmente el *Manifold*, se tiene un conjunto difuso de 1-*símplices* $FSC_{x_i}$ :

$$FSC_{x_i} = \{(x_i, x_j, \mu_{B_{x_i}}(x_j))\}_{j=0}^{m}.$$

Por último, resta tomar la unión de los $FSC_{x_i}$ para obtener una representación del *Manifold*. Primero se debe notar el hecho de que para cada par de bolas $B_{x_i}$, $B_{x_j}$ con intersección no nula existen dos 1-*simplex*, cada uno con un peso asignado por la función de pertenencia de $B_{x_i}$ o de $B_{x_j}$. La Figura 4 muestra un ejemplo de dos bolas con intersección no nula, cada bola define su propia medida de distancia y, por ende, la distancia entre sus centros depende de la medida que se utilice.

Notando, además, que las pertenencias toman valores entre 0 y 1, y que estas se interpretan como la probabilidad de existencia del 1-*simplex* al que se asocian, la unión de los $FSC_{x_i}$ se define como la probabilidad conjunta de las probabilidades de existencia de ambos 1-*símplices*:

$$M(\mu_{B_i}, \mu_{B_j}) = \mu_{B_i} + \mu_{B_j} - \mu_{B_i}\mu_{B_j}.$$

### 2.3.3. Optimización de la Representación en baja dimensión

La representación de los puntos en baja dimensión puede ser construida a partir de una inicialización de coordenadas aleatorias para cada punto en baja dimensión. Las representaciones topológicas obtenidas en ambos casos pueden representarse como vectores de probabilidad indexados por tuplas de 0-*símplices*, pues ambas representaciones comparten los mismos puntos, y por ende, los mismos 0-*símplices*.

UMAP luego comienza un proceso de optimización en el cual ajusta la disposición de los puntos en la baja dimensión con el objetivo de minimizar la entropía cruzada entre ambos

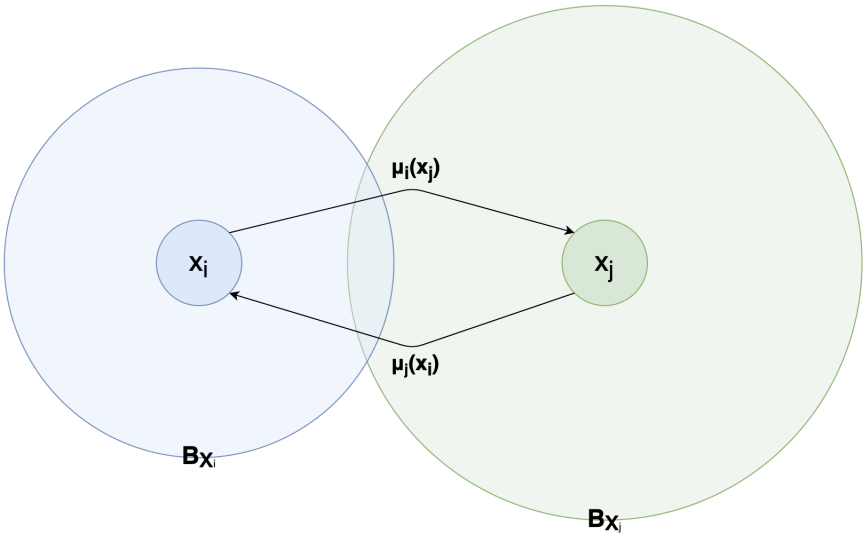


Figura 4: Ejemplo de los dos 1-*símplices* que se forman entre dos bolas $B_{x_i}$ y $B_{x_j}$ con intersección no nula. Cada 1-*simplex* tiene asociado el valor de pertenencia de un punto a la otra bola.

vectores de probabilidad. Específicamente, el proceso de optimización consiste en un algoritmo de dibujo de grafos basado en encontrar el equilibrio entre fuerzas atractivas y repulsivas.

Aquí UMAP introduce el último hiperparámetro relevante, la distancia mínima a la que pueden quedar, una vez concluido el proceso de optimización, un par de puntos cualquiera en baja dimensión. Valores menores de este hiperparámetro resultan en representaciones en baja dimensión con aglomeraciones más concentradas de puntos.

En este proyecto se asignó un valor igual a cero a este último hiperparámetro de UMAP. Esto permitió obtener una representación en baja dimensión que contrae, aún más, las regiones de puntos densas, facilitando de esta manera la tarea de *clustering* posterior.

## 2.4. Análisis de Clusters

Dado un conjunto de datos $D$, el análisis de *clusters* consiste en encontrar subconjuntos de datos de $D$. Cada subconjunto corresponde a un *cluster* distinto y cada *cluster* es tal que los datos dentro de él son similares entre si y al mismo tiempo disimilares a los datos en otros *clusters*. El conjunto de *clusters* obtenido como resultado de este proceso corresponde a un *clustering* de $D$. Por último, la búsqueda de *clusters* no la realiza un humano sino un algoritmo de *clustering* [9].

Concluimos entonces, que la tarea de encontrar grupos de fórmulas con *perfiles sensoriales* similares corresponde a efectuar un análisis de *clusters* sobre un conjunto de fórmulas generadas por Giuseppe para un *target* específico.

### 2.4.1. Taxonomía de Algoritmos de Clustering

A continuación describiremos brevemente una taxonomía de los algoritmos de *clustering* desarrollada en el trabajo de Fahad et al. [5].

- **Basado en Particiones** : Estos algoritmos dividen el conjunto de datos en un número predefinido de particiones, cada una corresponde a un *cluster* distinto. Los *clusters* encontrados contienen al menos un dato y cada dato pertenece exactamente a un *cluster*. Esta última característica puede ser relajada dependiendo del algoritmo. En el libro de Jiawei et al. *"Data Mining Concepts and Techniques"* [9] se nota que los algoritmos basados en particiones suelen solo ser capaces de encontrar *clusters* con formas esféricas.
- **Basado en Jerarquías** : Estos algoritmos representan los datos utilizando un dendrograma que contiene un dato en cada hoja y el conjunto completo de datos en la raíz. El *cluster* inicial que contiene todos los puntos se divide progresivamente en *clusters* más pequeños a medida que continúa la jerarquía en el dendograma. Los métodos jerárquicos pueden ser aglomerativos (*bottom-up*) o divisivos (*top-down*). Un algoritmo aglomerativo parte con un *cluster* por dato, es decir, por las hojas del dendrograma, y combina recursivamente los pares de *clusters* más apropiados. Los algoritmos divisivos parten con un solo *cluster* que contiene a todos los puntos, es decir, por la raíz del dendrograma, y divide recursivamente los *clusters* más apropiados. Ambos algoritmos continúan su proceso hasta que se alcanza un criterio de término. La principal desventaja de los algoritmos jerárquicos es que una vez que se hace una división o combinación, esta no puede deshacerse.
- **Basado en Densidad** : Los datos se separan basado en regiones de densidad, conectividad y bordes. Un *cluster* de puntos se define como una componente densa y conexa, y este crece en cualquier dirección en que aumente la densidad. Por ende, estos algoritmos pueden descubrir *clusters* de formas arbitrarias.
- **Basados en Grillas** : Se divide el espacio de los datos utilizando una grilla y luego se calculan estadísticas para cada componente de la grilla. La principal ventaja de estos algoritmos es que recorren solo una vez el conjunto de datos para crear la grilla y luego la complejidad depende de la definición de la grilla y las estadísticas a calcular para cada componente de la grilla. Por último, el *clustering* se efectúa sobre las medidas tomadas sobre la grilla y no sobre los puntos individualmente.
- **Basado en Modelos** : Este tipo de algoritmos ajustan un modelo matemático predefinido (es decir, sus parámetros asociados) al conjunto de datos. Se basan en la suposición de que los datos fueron generados por una mezcla de distribuciones de probabilidad. En esta categoría entran los métodos estadísticos y los basados en redes neuronales.

La elección del algoritmo de *clustering* a utilizar en este proyecto debe considerar principalmente el carácter exploratorio requerido de la solución. Este requisito implica que no podemos asumir nada sobre la cantidad, la forma, el tamaño, ni la existencia incluso de *clusters* en los datos.

Adicionalmente, el algoritmo de *clustering* escogido debe ser no paramétrico, o en su defecto, contener hiperparámetros que no requieran ser ajustados específicamente para cada entrada recibida. El algoritmo de selección debe ser capaz de identificar grupos de fórmulas con *perfiles sensoriales* similares, si es que los existen, en conjuntos de fórmulas generados

para *targets* que pueden corresponder a categorías de alimentos diametralmente distintas.

### 2.4.2. HDBSCAN : Hierarchical Density-based Clustering for Applications with Noise

HDBSCAN es un algoritmo de *clustering* jerárquico basado en densidad desarrollado por Campello et al. [3]. Su carácter dual (jerárquico y basado en densidad) deriva del hecho que HDBSCAN genera una jerarquía completa de *clusters* basados en distintos umbrales de densidad. Desde esta jerarquía se extrae luego el conjunto de *clusters* más estables, posiblemente con distintas densidades.

El carácter basado en densidad de HDBSCAN le permite identificar *clusters* con formas arbitrarias, por otro lado, su carácter jerárquico le permite explorar un rango de umbrales de densidades que definen distintos *clusters* cada uno. Por último, el proceso de extracción de *clusters* logra identificar aquellos *clusters* más persistentes a lo largo de los distintos umbrales de densidad considerados, de modo que, no es necesario saber a-priori la cantidad de *clusters*, sino que esta queda definida por las características propias de los datos.

Considerando las características de HDBSCAN mencionadas anteriormente, y el carácter exploratorio requerido de la solución, es que escogimos HDBSCAN como el algoritmo a utilizar para encontrar grupos de fórmulas con *perfiles sensoriales* similares.

A continuación explicaremos el funcionamiento de HDBSCAN y la intuición detrás de sus dos únicos hiperparámetros $min_{pts}$ y $min_{clSize}$. Lo que resta de la descripción de este algoritmo fue tomada del trabajo de Campello et al. [3].

Para extraer una jerarquía de *clusters* desde el conjunto de puntos HDBSCAN construye, en primer lugar, un grafo completo $G_{m_{pts}}$. Cada nodo de $G_{m_{pts}}$ es un dato, cada arista $w_{p,q}$ tiene como valor asociado el *mutual reachability distance* entre los datos $x_p$ y $x_q$. El *mutual reachability distance* entre $x_p$ y $x_q$ se define como

$$d_{mreach}(x_p, x_q) = max\{d_{core}(x_p), d_{core}(x_q), d(x_p, x_q)\} \tag{2}$$

donde $d_{core}(x_p)$ es el *core distance* de $x_p$ y corresponde a la distancia de $x_p$ a su $m_{pts}$-vecino más cercano (incluyendo a $x_p$). De esta manera, puntos localizados en regiones densas tendrán valores de *core distance* pequeños, mientras que puntos localizados en regiones poco densas tendrán valores de *core distance* mayores.

Bajo la medida de distancia presentada en la Ecuación 2, los puntos que pertenecen a regiones densas, es decir, aquellos puntos con valores bajos de *core distance* permanecen a su distancia original. Por otro lado, los puntos que pertenecen a regiones menos densas son separados a una distancia que es al menos sus valores de *core distance*.

Una vez construido el grafo $G_{m_{pts}}$ se construye el *Minimum Spanning Tree* de $G_{m_{pts}}$ y finalmente con el algoritmo mostrado a continuación se extrae una jerarquía en forma de dendrograma desde el $MST$:

1. Se asignan todos los puntos a la raíz del dendrograma como un único *cluster* que

contiene a todos los puntos

2. Remover iterativamente todas las aristas del $MST$ en orden decreciente de acuerdo sus pesos:
   (a) Antes de remover una arista se asigna el peso de la arista a remover al valor de escala actual del dendograma (el eje y).
   (b) Después de remover la arista, se le asignan etiquetas a las componentes conexas que contienen a ambos nodos relacionados a la arista eliminada. Para avanzar en la jerarquía se asigna una nueva etiqueta de *cluster* a la o las componentes conexas si tienen más de un vértice. Una componente con un solo nodo se considera como ruido.

En cada iteración se elimina una arista del $MST$ con *mutual reachability distance* $w$, por ende, todas las aristas restantes tienen *mutual reachability distances* asociadas menores o iguales a $w$. En consecuencia, las componentes conexas que se forman al eliminar una arista corresponden a los *clusters* de puntos con valores de *core distance* menores o iguales a $w$.

Cada vez que se elimina una arista pueden ocurrir una de las siguientes situaciones con las componentes conexas: (i) una de las componentes se encoge pero permanece conexa, hasta un cierto umbral en el que luego o bien (ii) se divide en componentes mas pequeñas o (iii) desaparece. Para limpiar un poco el dendrograma resultante de *clusters* espurios, HDBSCAN elimina las componentes conexas resultantes que contengan menos puntos que un hiperparámetro del algoritmo $m_{\text{clSize}}$.

Finalmente el dendrograma simplificado pasa por un proceso de optimización que busca el conjunto de *clusters* no anidados más estables a lo largo de los distintos umbrales de densidad recorridos.

En conclusión, HDBSCAN es un algoritmo de *clustering* jerárquico, basado en densidad, que dado dos hiperparámetros: $min_{pts}$ y $min_{clSize}$, logra encontrar *clusters* de densidades variables y formas arbitrarias.

En este proyecto, $min_{pts}$ nos permite definir globalmente que tan conservadores deseamos ser en nuestra búsqueda de familias de fórmulas con *perfiles sensoriales* similares pues a mayores valores de $min_{pts}$, más fórmulas serán declaradas como ruido. Por otro lado el hiperparámetro $min_{clSize}$ nos permite definir el tamaño mínimo requerido de una familia.

### 2.4.3. Tendencia de Puntos a formar Clusters

Considerando el tiempo que toma un proceso de validación de *clusters* y, además, que los algoritmos de *clustering* pueden encontrar *clusters* espurios, resulta de suma utilidad conocer la tendencia de los puntos a agruparse naturalmente antes de ejecutar cualquier algoritmo. En el libro de Jain et al. *"Algorithms for Clustering Data"* se define el concepto *Tendencia de Clustering* como un problema que consiste en decidir si un conjunto de datos presenta una tendencia natural a formar grupos.

Los tests de *Tendencia de Clustering* se plantean en realidad como tests de aleatoriedad espacial. Bajo esta concepción, si un conjunto de puntos se distribuye uniformemente o aleatoriamente en el espacio no se debe intentar clusterizar dicho conjunto.

### 2.4.4. Estadístico de Hopkins

El Estadístico de Hopkins fue introducido por primera vez en el trabajo de Lawson et al. [10]. El Estadístico de Hopkins mide la probabilidad que un conjunto de datos dado haya sido generado por una distribución uniforme. Para lograr esto se comparan las distribuciones de distancias al vecino más cercano para un subconjunto de los datos escogidos al azar con un conjunto de puntos generados con posiciones que siguen una distribución uniforme.

Para calcular el Estadístico de Hopkins primero se debe seleccionar al azar un subconjunto $\{x_i\}_{i=1}^m$ de los datos $D$ y generar un conjunto de puntos $\{y_j\}_{j=1}^m$ a partir de una distribución uniforme $d$ dimensional, con $d$ la dimensión de los datos. Luego se calcula la distancia de cada punto en $\{x_i\}_{i=1}^m$ a su vecino más cercano en $\{x_i\}_{i=1}^m$ y la distancia de cada en $\{y_j\}_{j=1}^m$ a su vecino más cercano en $\{x_i\}_{i=1}^m$. Finalmente, el Estadístico de Hopkins en $d$ dimensiones se define como:

$$H = \frac{\sum U_j^d}{\sum U_j^d + \sum W_j^d}.$$

$U_j$ es la distancia desde $y_j$ a su vecino más cercano en $\{x_i\}_{i=1}^m$ y $W_j$ es la distancia desde $x_j$ a su vecino más cercano en $\{x_i\}_{i=1}^m$. Luego, si el conjunto de datos $D$ estuviera distribuido uniformemente $\sum U_j^d$ y $\sum W_j^d$ tenderían a tener valores similares, y por ende, $H$ tomaría valores cercanos a 0.5.

Las hipótesis nula $H_0$ y alternativa $H_1$ asociadas al test estadístico son las siguientes:

- $H_0$ : El conjunto de datos $D$ está uniformemente distribuido
- $H_1$ : El conjunto de datos $D$ no está uniformemente distribuido

Así el test estadístico consiste en calcular iterativamente $H$ tomando muestras aleatorias distintas de $D$ en cada iteración y rechazar la hipótesis nula siempre que $H > 0{,}5$.

## 2.5. Métricas de Validación de un *Clustering*

Las métricas de evaluación de *clusterings* se dividen en dos categorías: las medidas de evaluación internas y las medidas de evaluación externas. Para utilizar medidas de evaluación externas se debe contar con asignaciones reales de cada punto a cada *cluster*. En aplicaciones reales, las métricas de evaluación externas no suelen ser muy útiles pues en la mayoría de los casos no se cuenta con asignaciones reales para cada dato.

Por otro lado, las métricas de evaluación internas solo utilizan los datos entregados como entrada al algoritmo de *clustering* y las asignaciones generadas por este. Típicamente, los métodos de validación interna evalúan mejor a *clusterings* con similitudes entre *clusters* bajas y similitudes altas entre puntos de un mismo cluster.

### 2.5.1. Índice de Jaccard

El Índice de Jaccard mide la similitud entre dos conjuntos $A$ y $B$ comparando su intersección sobre su unión:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}.$$

En este proyecto medimos el índice de Jaccard entre los ingredientes presentes en cada par de fórmulas asignadas a un mismo *cluster*. Medir esto nos permitió tener una noción de que tan similares son las fórmulas de un mismo *cluster* en términos de sus ingredientes.

Si bien existen interacciones complejas entre ingredientes de una fórmula, la principal característica que hace que dos fórmulas tengan *perfiles sensoriales* similares es la cantidad de ingredientes en común. Un *clustering* con índices de Jaccard promedio por *cluster* altos, significa que los *clusters* identificados comparten, en promedio, una gran cantidad de ingredientes en común y, por tanto, sus fórmulas constituyentes tenderán a tener *perfiles sensoriales* similares. Aun así, solo medir la cantidad de ingredientes en común no es suficiente, pues dos fórmulas pueden tener *perfiles sensoriales* equivalentes utilizando ingredientes distintos pero equivalentes como, por ejemplo, dos tipos de aceites distintos.

### 2.5.2. DBCV: Density Based Clustering Validation

DBCV es una medida de validación interna desarrollada por Moulavi et al. [15] capaz de validar *clusters* de formas arbitrarias. DBCV computa la región de menor densidad dentro de cada *cluster*, la que es utilizada como medida de similitud entre los puntos de un mismo *cluster*. Para la medida de similitud entre *clusters* DBCV computa la región de mayor densidad entre cada par de *clusters*.

DBCV construye un índice de validez $V_C(C_i)$ para cada *cluster* $C_i$ del *clustering* $C$. Este índice de validez toma valores positivos cuando la densidad dentro del *cluster* $C_i$ es mayor que la densidad que separa $C_i$ de los demás *clusters*.

Finalmente, el valor de DBCV para un *clustering* $C = \{C_i\}_{i=1}^{l}$ se calcula como un promedio ponderado de los índices de validez $V_C(C_i)$ calculados para cada *cluster*:

$$DBCV(C) = \sum_{i=1}^{l} \frac{|C_i|}{|O|} V_C(C_i).$$

$O$ corresponde al conjunto completo de datos, incluido aquellos que fueron catalogados como ruido. Por ende, mientras menor sea el porcentaje de los datos con *clusters* asignados, peor será la evaluación de DBCV del *clustering*. Finalmente, notamos que DBCV toma valores entre $-1$ y $+1$, donde valores cercanos a 1 indican un *clustering* basado en densidad de mejor calidad.

En este proyecto utilizamos DBCV para medir la calidad de los *clusterings* encontrados por HDBSCAN pues estos pueden ser de tamaño y forma arbitraria. Adicionalmente, dado que DBCV considera los puntos sin *cluster* asignados, si obtenemos valores de DBCV negativos o cercanos a 0 puede significar que HDBSCAN no es el algoritmo indicado para la tarea o que nuestra elección de hiperparámetros de HDBSCAN resulta en un *clustering* demasiado conservador.

## 2.6. Evaluación de un *Clustering* con expertos

Cuando en una tarea de *clustering* se cuenta con las asignaciones correctas para cada dato, evaluar la calidad del *clustering* se reduce a comparar, utilizando alguna medida de evaluación, las etiquetas asignadas automáticamente por el algoritmo de *clustering* con las etiquetas correctas disponibles. Sin embargo, en la mayoría de los casos, no se cuenta con las asignaciones correctas y, por ende, medir la bondad del resultado de un algoritmo de *clustering* no es directo.

Nuestro proyecto corresponde justamente a uno de esos casos en los que se quiere realizar *clustering* sobre un conjunto de puntos sin conocer las asignaciones reales de estos. Específicamente, en este proyecto buscamos encontrar grupos de fórmulas con *perfiles sensoriales* similares. Como dijimos en la Sección 2.5.1 no basta con medir los ingredientes en común entre fórmulas de cada *cluster* pues existen también ingredientes distintos pero con *perfiles sensoriales* equivalentes. Para comprobar que los *clusters* encontrados contienen fórmulas con *perfiles sensoriales* similares se requiere de un Chef experto capaz de determinar si, efectivamente, dentro de un mismo *cluster* existen fórmulas con *perfiles sensoriales* similares.

La solución a desarrollar recibe un conjunto de fórmulas generadas por Giuseppe para un *target* específico como entrada y entrega como salida los *clusters* de fórmulas con *perfiles sensoriales* similares. Para evaluar la calidad del *clustering* generado con los Chefs existen dos alternativas: presentar el *clustering* generado por la solución a los Chefs y pedirles que estimen la calidad del *clustering* con respecto al objetivo establecido, ó pedirle a los Chefs que completen la misma tarea de *clustering* y luego comparar el *clustering* obtenido por la solución desarrollada con el generado por los Chefs.

La primera alternativa deriva en resultados de evaluación más sesgados pues los humanos encuentran fácilmente argumentos para justificar comportamientos (en este caso, agrupamientos) de datos observados. Por otro lado, la segunda alternativa es más atractiva pues requiere que los Chefs creen su propio modelo mental para justificar las agrupaciones de fórmulas que encuentran. Considerando lo anterior, decidimos utilizar la segunda alternativa pues los Chefs son los más calificados para definir que pares de fórmulas contienen *perfiles sensoriales* similares, por lo que su criterio de agrupamiento será considerado como el correcto.

Para comparar el *clustering* obtenido por la solución desarrollada con el obtenido por los Chefs utilizamos el proceso de validación descrito en el trabajo de Hatzivassiloglou y McKeown [7]. Este proceso de comparación de *clusters* fue ideado originalmente para comparar *clusters* de adjetivos identificados automáticamente con *clusters* de adjetivos generados por un grupo de expertos. A continuación describiremos el proceso de validación.

Dado un conjunto de datos $D$ y un *clustering* de $D$ generado por el sistema que se desea evaluar, el proceso de validación comienza entregándole el conjunto de datos $D$ al grupo de expertos y pidiéndoles que completen la misma tarea de *clustering* que lleva a cabo el sistema a evaluar. Posteriormente, la comparación de ambos *clusterings* se reduce a una serie de preguntas con respuesta si o no, donde cada pregunta tiene una respuesta correcta (la entregada por el modelo generado de los expertos) y una respuesta asignada por el modelo a evaluar.

| | Sistema | $E_1$ | $E_2$ | $E_3$ | $GC$ | $GI$ |
|---|---|---|---|---|---|---|
| $(x, y)$ | si | si | si | no | 0,67 | 0,33 |
| $(x, z)$ | no | si | si | si | 1,0 | 0,0 |
| $(y, z)$ | si | no | no | no | 0,0 | 1,0 |

Tabla 1: Ejemplo de preguntas de asociación de pares de datos.

| | Experto | |
|---|---|---|
| | **SI** | **NO** |
| **Sistema - SI** | x | y |
| **Sistema - NO** | w | z |

Tabla 2: Ejemplo de Matriz de confusión.

Para cada par de datos $x, y$ la pregunta es: ¿fueron $x, y$ asignados al mismo *cluster*? Dado que cada experto puedo generar *clusterings* distintos, las respuestas correctas de los expertos se representan con dos números: $GC$ (grado de correctitud) y $GI$ (grado de incorrectitud). $GC$ indica el porcentaje de expertos que sí asigno $x, y$ dentro del mismo *cluster* y $GI$ el porcentaje de los expertos que no. Por ejemplo, si la validación cuenta con la participación de 3 expertos y 2 de 3 expertos identifica un par de datos $x, y$ dentro de un mismo grupo se tiene un valor de $GC = 0{,}67$. La Tabla 1 muestra un breve ejemplo con 3 datos y 3 expertos.

Utilizar $GC$ y $GI$ permite asociar un puntaje a cada asociación basado en la popularidad que tuvo la asociación entre los expertos. Por ejemplo, en la Tabla 1, la asociación de los datos $x, z$ en un mismo grupo tiene un valor de $GC = 1$ pues todos los expertos agruparon los datos $x, z$ dentro del mismo grupo.

Una vez calculados los valores de $GC$ y $GI$ rellenamos una matriz de confusión como la mostrada en la Tabla 2. La primera columna se rellena utilizando la suma de los $GC$ y la segunda columna utilizando la suma de los $GI$. Para la primera fila se consideran solo los pares para los que el sistema responde "si" y para la segunda fila solo los pares para los cuales el sistema responde "no". Siguiendo la lógica previamente descrita, el valor de $x$ en la Tabla 2 correspondería a la suma de los $GC$ de aquellos pares de fórmulas que sí fueron asignadas a un mismo grupo por la solución desarrollada.

Por último, se utilizan las ecuaciones presentadas a continuación para calcular la precisión, exhaustividad y medida F1 alcanzado por el *clustering* del sistema.

$$\begin{aligned} \text{Precisión} &= \frac{x}{x+y} \\ \text{Exhaustividad} &= \frac{x}{x+w} \\ F_1 &= \left( \frac{2}{\text{Precisión}^{-1} + \text{Exhaustividad}^{-1}} \right) \end{aligned} \tag{3}$$

# 3. Problema Abordado

## 3.1. Descripción del Proceso de Selección de fórmulas

Giuseppe representa tanto los alimentos de origen animal, como los ingredientes vegetales con vectores de *características moleculares* (ver Definición 2) cuyos contenidos específicos se escapan del alcance de este proyecto. A las combinaciones de ingredientes vegetales generadas se les denominan, en este contexto, fórmulas. Cada fórmula se compone de una lista de ingredientes vegetales con sus respectivas proporciones, un vector de *características moleculares* y un puntaje que mide la similitud entre dicha fórmula y el alimento que busca reproducir esta.

En una ejecución razonable, Giuseppe encuentra del orden de miles de fórmulas distintas para un mismo *target*. Considerando que miles de ingredientes vegetales distintos generan innumerables combinaciones distintas posibles, el aporte de Giuseppe al problema es indiscutible. Aun así, no es factible en términos de costo y tiempo cocinar cada una de estas miles de fórmulas distintas para dar con la que mejor reproduce el *target*.

Para lidiar con esto, los conjuntos de fórmulas resultantes de Giuseppe deben pasar por un proceso de selección. El propósito de este proceso es identificar el subconjunto mínimo de fórmulas distintas, en términos de sus *perfiles sensoriales*, con mayor probabilidad de éxito. Una fórmula se dice exitosa si una vez preparada logra reproducir satisfactoriamente el *perfil sensorial* de su *target* asociado. Entregar a los Chefs una selección de fórmulas con estas características permite aprovechar eficientemente su tiempo en la cocina pues le permite experimentar con fórmulas distintas y con alta probabilidad de éxito.

En NotCo el proceso de selección está a cargo del área de *Machine Learning*, específicamente, a cargo del equipo de ingenieros, quienes no cuentan con ningún entrenamiento gastronómico. Como consecuencia, la única forma objetiva que se tiene para juzgar la probabilidad de éxito de una fórmula es evaluar la similitud reportada de la fórmula a su *target*.

El proceso de selección consiste primero en ordenar las fórmulas por su similitud al *target* y segundo en recorrer esta lista ordenada hasta seleccionar una cantidad, a juicio del ingeniero a cargo, suficiente de fórmulas distintas y similares al *target* en cuestión.

## 3.2. Problemas Identificados

A continuación presentamos los problemas identificados con el proceso de selección de fórmulas descrito en la Sección 3.1.

1. **La similitud no es una medida efectiva ni suficiente para juzgar la probabilidad de éxito de una fórmula**: La similitud es en realidad la distancia Euclidiana entre una fórmula y su *target* asociado, en el espacio de *características moleculares*. Este espacio es de alta dimensión y, por tanto, ocurre que la mayoría de las distancias calculadas resultan ser muy similares. Como consecuencia, si bien es posible ordenar las fórmulas por similitud, no es posible seleccionar fórmulas *únicamente* por similitud pues todas resultan, en promedio, igual de similares al target en cuestión.

2. **Proceso de Selección de fórmulas a cargo del equipo de *Ingenieros*** : El proceso de selección de fórmulas consiste en identificar aquellas fórmulas que, con mayor probabilidad, una vez *cocinadas* reproducirán el sabor, color, aroma, textura y retro gusto de un target específico. En consecuencia, un proceso de selección efectivo requiere de conocimiento gastronómico. El problema es que el proceso de selección está a cargo de los ingenieros del equipo de *Machine Learning*, cuyos integrantes no cuentan con conocimiento ni entrenamiento gastronómico. Luego, la única manera que tiene el equipo de ingenieros para guiar el proceso de selección de fórmulas es utilizar la similitudes de la fórmulas al *target* en cuestión.

3. **El proceso de selección solo considera un subconjunto de tamaño variable de las fórmulas generadas**: El proceso de selección consiste actualmente en ordenar las fórmulas generadas por su similitud al *target* y luego, mediante una inspección superficial de las fórmulas ordenadas, seleccionar un subconjunto de fórmulas similares al *target* y con ingredientes distintos. Seleccionar de esta manera deriva en selecciones potencialmente poco representativas.

4. **Pérdida de información relevante al utilizar una lista ordenada**: Utilizar una lista ordenada de fórmulas por similitud permite ver relaciones de orden entre las fórmulas generadas pero no permite apreciar relaciones mas complejas que existen dentro de todo el conjunto de fórmulas generadas. Ejemplo de estas relaciones complejas son las familias de fórmulas con *perfiles sensoriales* similares. Dada la manera en que Giuseppe explora el espacio de ingredientes este tiende a generar distintos conjuntos bases de ingredientes y miles de variaciones para cada uno. Cada conjunto base de ingredientes caracteriza a una familia de fórmulas, y cada variación del conjunto base se traduce en una fórmula integrante de aquella familia.

## 3.3. Ventajas de un nuevo Proceso de Selección

Un nuevo proceso de selección debería permitir que sea un Chef quien seleccione el subconjunto de fórmulas a cocinar mediante una exploración acabada del conjunto completo de resultados. Este nuevo proceso de selección derivará en un criterio de selección que combina el conocimiento experto del Chef con las similitudes reportadas por Giuseppe, amortiguando así los problemas que adolecen a la similitud.

A su vez, este nuevo criterio de selección compuesto podría traducirse en aumentos significativos de las probabilidades de éxito de las fórmulas seleccionadas, lo que significaría una reducción en los tiempos de desarrollo de nuevos productos.

Por otro lado, un proceso de selección que permita a los Chefs identificar las familias de fórmulas con sus variaciones de ingredientes respectivas es sumamente valioso por dos razones:

1. Permite seleccionar fórmulas con ingredientes distintos para el proceso de preparación. Contar con fórmulas con ingredientes distintos permite a los Chefs experimentar paralelamente con preparaciones distintas.
2. Permite a los Chefs encontrar variaciones para una fórmula en caso de encontrarse con

ingredientes no disponibles o que no son adecuados para la preparación en una fórmula en particular.

Finalmente, el que el proceso de selección considere el conjunto completo de fórmulas generado por Giuseppe permite incrementar la seguridad sobre la representatividad de las fórmulas seleccionadas. Es decir, permite tener mayor confianza en que el proceso de selección está identificando todas las fórmulas interesantes y que no se está perdiendo información relevante.

## 3.4. Requisitos

Considerando los problemas identificados con el proceso de selección actual y los beneficios que tendría solucionar estos problemas, es que nace este proyecto de título con el objetivo principal de desarrollar un algoritmo que apoye un nuevo proceso de selección de fórmulas.

Este nuevo proceso debe ser de carácter exploratorio, debe considerar el conjunto completo de resultados generados por Giuseppe, debe ser capaz de identificar las familias de fórmulas generadas caracterizadas por *perfiles sensoriales* distintos y su resultado debe permitir involucrar a los Chefs en el proceso de selección, para que sean ellos quienes a través de un complemento de su conocimiento experto con las similitudes reportadas por Giuseppe, decidan finalmente qué fórmulas pasarán al proceso de preparación.

Como requisitos técnicos la solución debe desarrollarse en *Python* para integrarse de manera directa tanto con los modelos como con las herramientas internas ya implementadas. Además, el diseño debe abstraerse de cualquier algoritmo específico utilizado.

En términos de su uso, la solución será utilizada por el equipo de *Machine Learning*, pero el resultado entregado por esta debe ser al menos legible por Chefs y en un formato consumible por una herramienta web encargado de desplegar estos.

Adicionalmente la solución debe ser capaz de funcionar correctamente, vale decir, entregar resultados válidos, sin requerir ajuste alguno específico al *target* o al conjunto de fórmulas generado para que este sea recibido como entrada.

En términos de calidad, todo código escrito debe estar cubierto por pruebas e implementado siguiendo buenas prácticas de diseño de software además de estar debidamente versionado y con sus dependencias especificadas inequívocamente.

# 4. Solución Desarrollada

La solución desarrollada consistió en un algoritmo que, dado el conjunto completo de fórmulas generado por Giuseppe para un *target* específico, detecta automáticamente las familias de fórmulas como *clusters* de densidades variables en un espacio de características reducido de las fórmulas.

Para facilitar la comprensión del desarrollo de la solución construida detallaremos primero la información contenida en cada fórmula generada por Giuseppe. Como dijimos en la Sección 3.1, cada una de estas fórmulas se compone de un diccionario de ingredientes, un vector de características y la similitud de la fórmula al *target* para el que fue generada:

1. **Similitud** al *target*: Distancia Euclidiana entre los vectores de *características moleculares* de la fórmula y del *target*.
2. Diccionario de **Ingredientes**: contiene los ingredientes presentes en la fórmula con sus respectivos porcentajes. Por ejemplo :
   `{"tomate": 0.4, "mayonesa": 0.3, "palta": 0.4}`
3. Vector de ***características moleculares***: Vector numérico que representa la fórmula en términos físico, químico y nutricionales. La apuesta de NotCo es que este vector logra representar indirectamente el *perfil sensorial*[7] de la fórmula. Este es el vector que se utiliza para calcular la similitud de cada fórmula a su *target* asociado.

## 4.1. Metodología de Trabajo

La base de datos de *targets* contiene miles de alimentos distintos para los cuales Giuseppe puede generar fórmulas. Para el desarrollo de este trabajo consideramos un subconjunto de 150 *targets* escogidos al azar. Para cada uno de estos 150 *targets* generamos 5000 fórmulas, construyendo así el conjunto de datos utilizado para evaluar las implementaciones desarrolladas en este proyecto.

A continuación describiremos la metodología seguida para el desarrollo de la solución presentada en este trabajo.

1. **Selección de espacio de características**
2. **Reducción de dimensión**
3. **Comprobación de tendencia de puntos a formar *Clusters***
4. **Comprobación visual de existencia de *Clusters***
5. **Análisis de *Clusterings* obtenidos**
6. **Validación de *Clusterings* con Chefs**
7. **Diseño de Algoritmo de Selección**

## 4.2. Selección de espacio de características

Como explicamos en el inicio del Capítulo 4 cada fórmula generada por Giuseppe contiene sus ingredientes vegetales y un vector de *características moleculares*. Existen, por tanto, al

[7] Referido al color, aroma, sabor, textura y retro-gusto.

menos dos maneras distintas para representar una fórmula generada por Giuseppe.

El objetivo de este proyecto es lograr identificar familias de fórmulas con *perfiles sensoriales* similares. Considerando que fórmulas con ingredientes en común tienden a tener *perfiles sensoriales* similares, una eventual solución podría ser representar cada fórmula por su vector de ingredientes presentes y luego efectuar el *clustering* sobre esta representación. Los *clusters* identificados revelarían entonces las familias de fórmulas con un alto porcentaje de ingredientes común. Sin embargo, una familia de fórmulas no solo contiene ingredientes iguales, sino que también contiene además ingredientes distintos pero *sensorialmente equivalentes* e ingredientes totalmente distintos.

Producto de lo anterior es que decidimos efectuar el *clustering* sobre el espacio de *características moleculares* de las fórmulas, es decir, utilizando el vector de *características moleculares* para representar cada fórmula. Esta decisión significa que nos apoyamos en la hipótesis de que fórmulas que sean similares en este espacio molecular tendrán ingredientes en común y tendrán *perfiles sensoriales* similares. Esta hipótesis será comprobada con los Chefs al momento de validar los *clusters* encontrados.

## 4.3. Reducción de Dimensión

Representar cada fórmula por su vector de *características moleculares* para llevar a cabo la tarea de *clustering* plantea una hipótesis atractiva que de cumplirse significaría que hemos encontrado una solución al problema central planteado en este proyecto. Sin embargo, esta decisión tiene un costo: producto de la dimensión del vector de *características moleculares* debemos lidiar con el *curse of dimensionality*.

Existe una plétora de algoritmos que lidian con el *curse of dimensionality* a través de transformaciones lineales y/o no lineales del espacio en alta dimensión a uno de menor dimensión. Algunos de ellos son: PCA, KernelPCA, Isomap, LLE, Autoencoders, t-SNE.

Para guiar la selección del algoritmo de reducción de dimensionalidad a utilizar, consideramos dos requisitos importantes de la solución. El primer requisito es que la solución debe ser de carácter exploratoria, esto significa que cualquier reducción de dimensionalidad utilizada debe, en lo posible, no introducir estructuras espurias en los datos. El segundo requisito es que la solución debe ser robusta frente a conjuntos de fórmulas generados para distintos *targets*. De este requisito se desprende que el algoritmo de reducción de dimensionalidad a utilizar debe tener hiperparámetros cuya selección debe depender del contexto de generación de los datos más que de los datos mismos. En otras palabras, los hiperparámetros deben ser robustos frente a conjuntos de datos distintos pero provenientes de un mismo contexto.

El *blessing of non-uniformity* es otro concepto que guió nuestra elección de algoritmo de reducción de dimensionalidad. Este concepto contra resta en cierta medida muchas de las complicaciones introducidas por el *curse of dimensionality* y se refiere al hecho de que en muchos problemas de la vida real los puntos en altas dimensiones no se distribuyen uniformemente en el espacio sino que más bien tienden a concentrarse en un *manifold* de menor dimensión.

Considerando lo anterior, el algoritmo seleccionado para hacer la reducción de dimensión

fue UMAP. Los resultados obtenidos por UMAP al reducir los conjuntos de datos de prueba MNIST (Lecun et al. [11]) y FMNIST (Xiao et al. [19]) a dos dimensiones entregan evidencia de que UMAP logra reducir la dimensión de los datos manteniendo la estructura global y local de estos sin introducir estructuras espurias, cumpliendo con el carácter exploratorio requerido. Por otro lado, y a primera vista, los hiperparámetros de UMAP `min_dist` y `n_neighbors` pueden ser definidos acorde al contexto de generación de fórmulas y no con respecto a cada target en particular, cumpliendo con la robustez requerida.

Más aún, definir `min_dist` cercano a 0 resulta útil para la tarea posterior de *clustering* pues `min_dist` define que tan cerca pueden quedar los puntos en baja dimensión, acentuando la densidad de las regiones densas. Así mismo, los creadores de UMAP sugieren valores entre 20 y 30 para el hiperparámetro `n_neighbors` para tareas de *clustering* pues permite a UMAP capturar en buena medida tanto la estructura global como local de los datos.

Definimos entonces UMAP como el algoritmo de reducción de dimensionalidad escogido. A continuación, mostramos los valores de los hiperparámetros utilizados en este proyecto:

1. `min_dist` : 0.0
2. `n_neighbors` : 30
3. `target_dim` : 20
4. `metric` : euclidean

Notamos aquí que por restricciones de tiempo no fue posible explorar combinaciones de distintos valores para estos hiperparámetros, ni tampoco experimentar con otras técnicas de reducción de dimensión. Dejamos estas experimentaciones como trabajo futuro.

## 4.4. Tendencia de fórmulas a formar *Clusters*

Llevar a cabo una tarea de *clustering* de principio a fin es costoso en términos de tiempo. La tarea de *clustering* comprende seleccionar y ejecutar un algoritmo de *clustering*, además de definir y aplicar una metodología de validación de *clusters* obtenidos. El proceso de validación puede depender de la disponibilidad de tiempo de terceros, expertos en el área, facultados para validar los *clusters* encontrados. Debido a esto es importante poder corroborar con cierto grado de certeza que los puntos sobre los cuales se ejecutará el algoritmo de *clustering* tienden a agruparse naturalmente, sin necesidad de tener que identificar estos para así evitar perder tiempo encontrando *clusters* espurios.

En nuestro proyecto utilizamos el Estadístico de Hopkins (Lawson et al. [10]) para medir la tendencia que tiene un conjunto de fórmulas generado por Giuseppe a agruparse en *clusters* naturalmente.

El Estadístico de Hopkins mide, en realidad, la tendencia de los puntos a ser generados por un proceso de *Poisson* y, por ende, a estar distribuidos de manera uniformemente aleatoria. El estadístico toma valores entre 0 y 1, donde un valor cercano a 1 significa que los puntos no están distribuidos de manera uniforme en el espacio. Específicamente, valores del estadístico mayores a 0.75 indican no uniformidad con una confianza del 90 %.

Recordemos que ya hemos tomado dos decisiones de diseño importantes en el desarrollo de

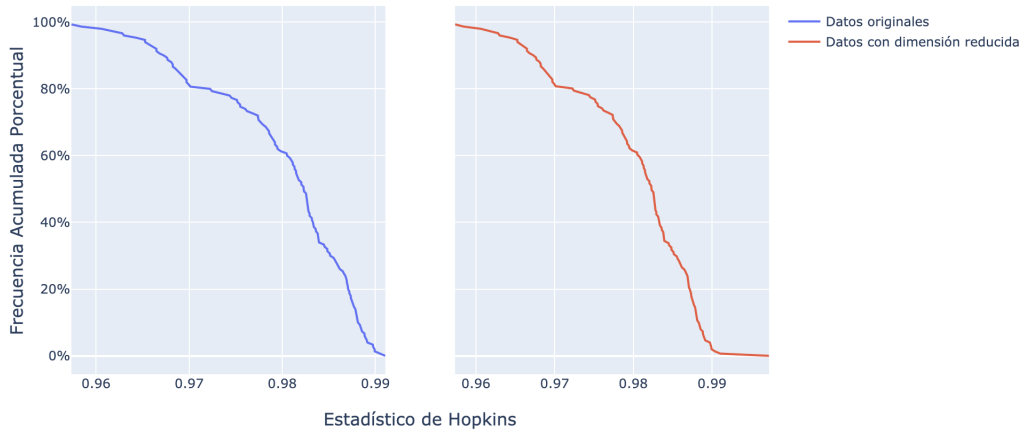


Figura 5: Frecuencia acumulada porcentual de los Estadísticos de Hopkins calculados para cada conjunto de fórmulas del conjunto de prueba. El punto X,Y indica que el Y % de los conjuntos de fórmulas tiene un Estadístico de Hopkins mayor o igual a X.

este proyecto: representar una fórmula por su vector de *características moleculares* y reducir la dimensión de este vector utilizando UMAP. Tenemos, por ende, un vector de características original y otro vector de dimensión reducida.

En la Figura 5 mostramos la frecuencia acumulada porcentual de los Estadísticos de Hopkins obtenidos para cada conjunto de fórmulas representadas tanto por sus vectores moleculares originales como por sus vectores con dimensión reducida.

Comprobamos que, tanto para los datos originales, como para los datos con dimensión reducidas, más del 95 % de los conjuntos de fórmulas presenta valores para el Estadístico de Hopkins mayores o iguales a 0,95. Por tanto, podemos afirmar que los datos no se encuentran uniformemente distribuidos con una confianza superior al 90 %.

Los estadísticos de los datos con dimensión reducida tienden a ser levemente más cercanos a 1 en comparación al espacio sin dimensión reducida. En la Figura 5 esto se puede comprobar notando la diferencia en el fin de ambas curvas. Esto se debe a que el algoritmo de reducción de dimensión UMAP tiende a contraer regiones de puntos densas en el espacio original, lo que produce que el estadístico tome valores más cercanos a 1 ya que los datos parecen estar aún menos uniformemente distribuidos.

## 4.5. Comprobación Visual de existencia de *Clusters*

El Estadístico de Hopkins sirve para comprobar no-uniformidad de los datos. Nos permite saber si los datos están distribuidos de manera aleatoria o no, pero no nos dice si efectivamente existen *clusters* ni tampoco nos habla de la forma de estos (globulares, elipsoidales, lineales, etc.).

Generalmente, distintos algoritmos de *clustering* tienden a tener un sesgo a detectar *clusters* de formas específicas. Por ejemplo, el algoritmo de *clustering K-Means* puede ajustar únicamente *clusters* con formas globulares a los datos. En la misma línea, el algoritmo de mezcla de Gaussianas *GMM*, por su naturaleza basada en distribuciones Gaussianas, solo

puede ajustar *clusters* con formas elipsoidales a los datos.

Del párrafo anterior se desprende que para seleccionar el algoritmo de *clustering* más apropiado para este proyecto es vital conocer la forma de los *clusters* en los que se agrupan las fórmulas generadas por Giuseppe para un mismo target.

Para comprobar la existencia y forma de *clusters* en los conjuntos de fórmulas generados por Giuseppe utilizamos un procedimiento de comprobación visual. Este proceso consistió primero en seleccionar 10 conjuntos de fórmulas generadas para 10 *targets* distintos. Luego, redujimos los vectores de *características moleculares* a 2 dimensiones utilizando UMAP. Finalmente visualizamos cada conjunto de fórmulas por separado, en un gráfico de puntos. Utilizamos estas visualizaciones para explorar cada conjunto y comprobar la existencia y forma de los *clusters* formados.

La Figura 6 muestra los resultados del proceso de comprobación de *clusters*. Cada fila es un conjunto de fórmulas generado por Giuseppe para un *target* distinto. La primera columna contiene el conjunto de datos original y las dos siguientes *clusters* o conjuntos de *clusters* identificados.

De la Figura 6 se desprende que las fórmulas se agrupan en *clusters* elipsoidales (fila 1, columna 2), lineales (fila 8, columna 1), globulares (fila 2, columna 1), circulares (fila 5, columna 1) y con forma de estrella (fila 5, columna 2). Notamos además que, en general, las fórmulas tienden a generar un *cluster* principal y luego una serie de *clusters* mas pequeños.

## 4.6. Análisis de *Clusters* obtenidos

Hasta este punto escogimos una manera de representar cada fórmula, justificamos reducir la dimensión de la representación escogida, comprobamos que las fórmulas no se distribuyen de manera aleatoria y corroboramos la existencia de *clusters* de fórmulas de distintas formas.

Resta decidir el algoritmo de *clustering* a utilizar para finalmente evaluar los clusters obtenidos. La decisión debe considerar los requisitos mencionados en la Sección 3.4 junto con las características de los conjuntos de fórmulas detalladas en las Secciones 4.4 y 4.5.

Consideramos que el algoritmo de *clustering* HDBSCAN es una opción atractiva puesto que logra encontrar *clusters* de formas y densidades variables, no requiere de un hiperparámetro que especifique la cantidad de *clusters* a ajustar a los datos y los hiperparámetros que sí son requeridos `min_pts` y `min_clSize` pueden ser definidos de acuerdo al contexto del problema y no con respecto a cada conjunto de fórmulas entregado como entrada.

A continuación especificamos los valores de los hiperparámetros utilizados en este proyecto:

1. `min_pts` : 5
2. `min_clSize` : 5

Al definir `min_clSize` igual a 5 suponemos que toda familia de fórmulas con *perfiles sensoriales* similares contiene al menos 5 fórmulas. Esta suposición es razonable dada la manera en que Giuseppe explora el espacio de ingredientes vegetales. Similarmente, al definir

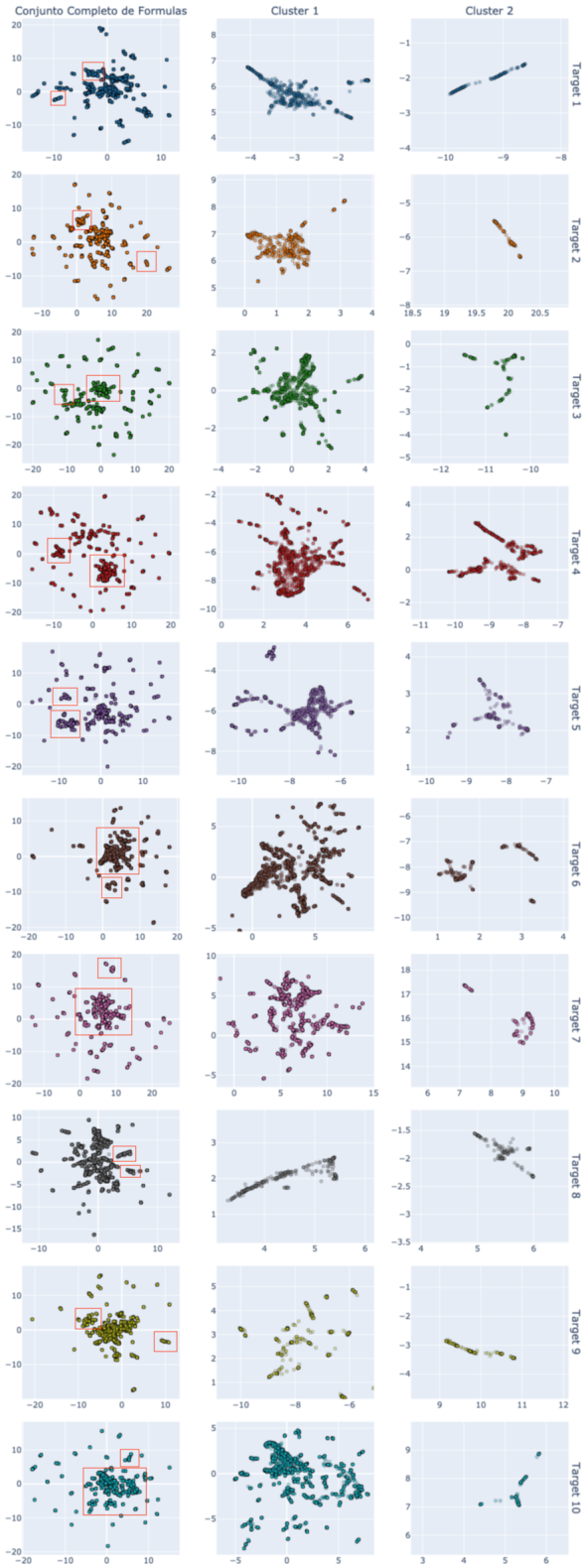


Figura 6: Gráfico de puntos para 10 conjuntos de fórmulas distintos, junto con *clusters* identificados de distintos tamaños y formas.

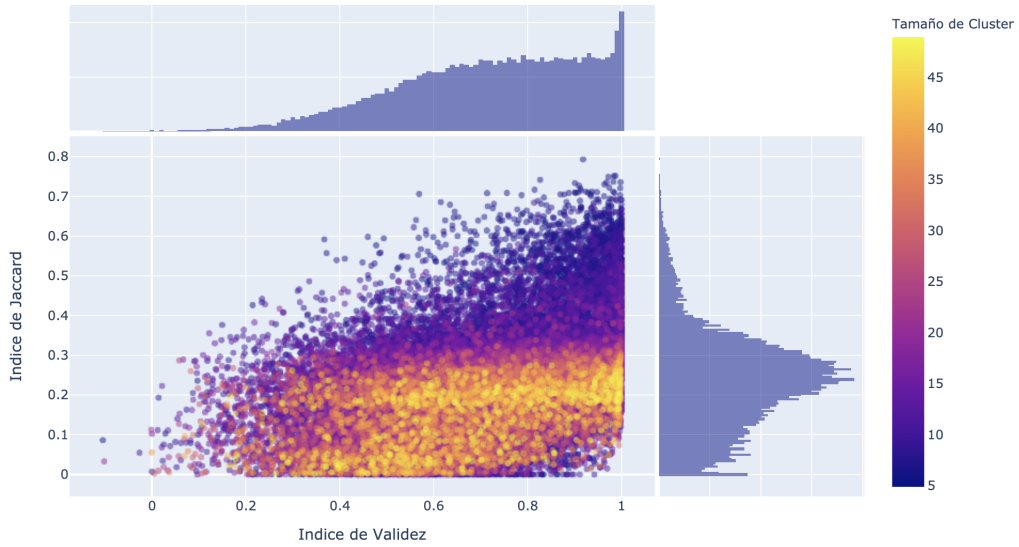


Figura 7: *Clusters* de fórmulas graficados en términos de sus índices de validez y de Jaccard.

el valor de `min_pts` igual a 5 estamos suponiendo que basta con la distancia al quinto vecino más cercano de cada punto para caracterizar correctamente la densidad de la región en la que se encuentra dicho punto.

El objetivo principal de este proyecto es lograr encontrar, dentro de un conjunto de fórmulas generadas por Giuseppe para un mismo *target*, *clusters* de fórmulas con *perfiles sensoriales* similares. Para comprobar que las fórmulas de una mismo *cluster* tienen *perfiles sensoriales* similares necesitamos de la ayuda del juicio experto de los Chefs. Por otro lado, sí es posible comprobar si los *clusters* de fórmulas identificados tienen ingredientes en común sin ayuda de los Chefs.

Para comprobar que los *clusters* encontrados contienen fórmulas con ingredientes en común utilizamos el índice de Jaccard. Específicamente, calculamos el índice de Jaccard con respecto a la presencia de ingredientes entre cada par de fórmulas pertenecientes a un mismo *cluster*. Luego tomamos el promedio de los índices calculados para cada familia, obteniendo finalmente el índice de Jaccard promedio por familia de fórmulas identificadas.

Para medir qué tan bien definido está cada *cluster* $C_i$ utilizamos el índice de validez $V_C(C_i)$ presentado en la Sección 2.5.2. Este índice toma valores entre 0 y 1, aquellos *clusters* con densidades internas elevadas y que además están claramente separados de los demás *clusters* toman valores del índice cercanos a 1.

Antes de presentar los índices de validez y de Jaccard de los *clusters* identificados por HDBSCAN cabe notar que eliminamos todos aquellos *clusters* con más de 50 fórmulas pues comprobamos que el 98 % de los *clusters* contiene 50 o menos fórmulas (ver Figura 8).

La Figura 7 muestra un gráfico de puntos con todos los *clusters* identificados por HDBSCAN para los 150 conjuntos de fórmulas del conjunto de prueba. En la Figura 7 cada punto es un *cluster* de uno de los 150 *targets* del conjunto de *targets* de prueba. El color de ca-

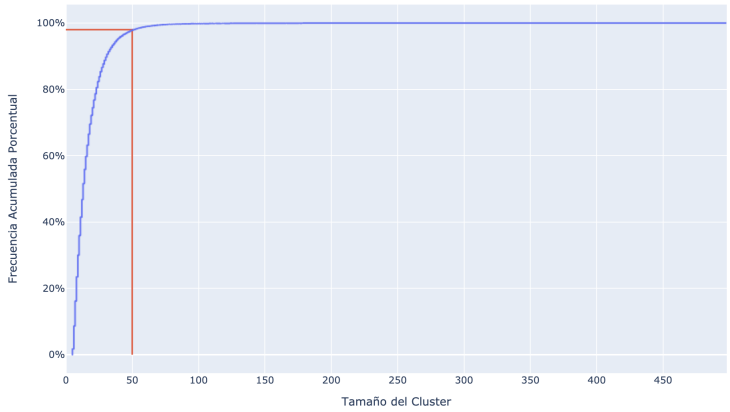


Figura 8: Frecuencia acumulada porcentual de tamaños de *clusters*

da punto indica el tamaño del cluster correspondiente, esto es, la cantidad de fórmulas que pertenecen a dicho *cluster*. El eje x contiene el índice de validez DBCV de cada *cluster* (ver Sección 2.5.2) mientras que el eje y contiene el índice de Jaccard de cada *cluster* (ver Sección 2.5.1). Incluimos además los histogramas de cada eje para analizar las distribuciones de cada uno.

Pudimos comprobar que el índice Jaccard promedio se distribuye normalmente con una media cercana a 0,3, lo que significa que, en promedio, las fórmulas de un mismo *cluster* tienen un 30 % de sus ingredientes en común. 30 % de ingredientes en común es un porcentaje razonable pues fórmulas de una misma familia pueden contener también ingredientes distintos pero con *perfiles sensoriales* similares, además de ingredientes completamente distintos y no equivalentes. Estos últimos ingredientes son los que le otorgan la variabilidad a las fórmulas de una misma familia. Por otro lado, existe una relación lineal positiva entre ambos índices, lo que significa que *clusters* mejor definidos tienden también a contener fórmulas con mas ingredientes en común. Además, los *clusters* de mayor tamaño tienden a tener índices de Jaccard menores, lo que es esperable pues a mayor cantidad de fórmulas en un *cluster*, mayor es la cantidad de ingredientes distintos y, por ende, los ingredientes en común entre pares de fórmulas, es menor.

El histograma marginal de los índices de validez de la Figura 7 revela que más del 50 % de los *clusters* presenta índices de validez superiores a 0,6, indicando que los *clusters* encontrados por HDBSCAN se encuentran bien definidos.

En conclusión, gran parte de los *clusters* identificados por HDBSCAN se encuentran bien delimitados y definidos, además las fórmulas de un mismo *cluster* tienen, en promedio, un 30 % de sus ingredientes en común. Resta comprobar ahora la existencia de fórmulas con *perfiles sensoriales* similares dentro de una misma familia.

## 4.7. Validación de *Clusters* con Chefs

El objetivo del proceso de validación de *clusters* es comparar el desempeño de la solución desarrollada con el desempeño de un grupo de Chefs sobre la misma tarea de *clustering*: *"Dado un conjunto de fórmulas, encontrar grupos de fórmulas con perfiles sensoriales similares"*.

| *Target* | Cantidad de fórmulas | Cantidad de *clusters* | $\mu$(Fórmulas por *cluster*) | $\sigma$(Fórmulas por *cluster*) | Cantidad de Ingredientes distintos |
|---|---|---|---|---|---|
| Barra de Caramelo | 16 | 4 | 4.0 | 1.22 | 1124 |
| Carne de Pollo | 18 | 5 | 3.6 | 0.48 | 1303 |
| Pizza de Queso | 20 | 4 | 5.0 | 0.70 | 1123 |
| Carne de Hamburguesa | 23 | 5 | 4.6 | 0.80 | 1239 |
| Galleta de Avena | 20 | 4 | 5.0 | 1.0 | 1133 |

Tabla 3: Subconjuntos de fórmulas seleccionadas para proceso de validación con Chefs, junto con los *clusters* encontrados por la solución desarrollada.

Los Chefs conocen de ingredientes, sus sabores, texturas, aromas, además de los resultados de combinar ingredientes de distintas maneras. Por lo tanto, consideramos los *clusters* de fórmulas encontrados por los Chefs como los grupos de fórmulas con *perfiles sensoriales* similares efectivamente presentes en el conjunto de fórmulas dado.

Para validar los *clusters* encontrados por la solución desarrollada utilizamos el método de validación de *clusters* con expertos descrito en la Sección 2.6. El método de validación consistió en pedirle a un grupo de 3 Chefs que completaran, cada uno, la misma tarea de *clustering* que lleva a cabo la solución desarrollada, para 5 *targets* distintos escogidos al azar: Barra de Caramelo, Carne de Pollo, Pizza de Queso, Carne de Hamburguesa y Galleta de Avena. Posteriormente, y para cada *target*, evaluamos la calidad del *clustering* encontrado por la solución desarrollada comparando este con el *clustering* encontrado por los Chefs.

Finalmente, la tarea de *clustering* que completan los expertos y la solución desarrollada no son totalmente iguales. En primer lugar, los Chefs utilizan los ingredientes presentes en cada fórmula para representar estas, a diferencia de la solución desarrollada, la cual utiliza los vectores de *características moleculares*. Decidimos entregarle los ingredientes de cada fórmula a los Chefs pues consideramos que esta representación es la más amigable para un experto cuando se le pide encontrar grupos de fórmulas con *perfiles sensoriales* equivalentes. En segundo lugar, y como veremos en la siguiente sección, debido a restricciones de tiempo y las diferencias entre las capacidades de cómputo de una máquina y de un humano, las fórmulas entregadas a los Chefs para que completen la tarea de *clustering* son un pequeño subconjunto del conjunto de fórmulas entregado a la solución desarrollada para completar la misma tarea de *clustering*.

### 4.7.1. Algoritmo de Construcción de Conjunto de Fórmulas para Validación

La solución desarrollada puede procesar conjuntos de más de 1000 fórmulas con facilidad y en cuestión de segundos. Desafortunadamente, resulta infactible en términos de tiempo pedirle a un grupo de Chefs que encuentre *clusters* de fórmulas en un conjunto de más 1000 elementos.

Producto de lo anterior, desarrollamos el Algoritmo 1. Este Algoritmo recibe como entrada un conjunto de fórmulas[8] para las cuales la solución desarrollada ya encontró *clusters* de fórmulas y selecciona $K \sim U(a, b)$ *clusters* distintos, luego, para cada *cluster* selecciona $Q \sim U(c, d)$ fórmulas. Las propiedades de la distribución uniforme y la linealidad de la esperanza matemática nos dice que el conjunto de fórmulas resultante del Algoritmo 1 tiene una cardinalidad esperada igual a $(b - a + 1)(d - c + 1)$. Para obtener conjuntos de fórmulas

[8]Generadas por Giuseppe para un mismo *target*.

**Algorithm 1:** Algoritmo de construcción de conjunto de fórmulas de validación

**Result:** Sampled formulae
**Input:** Clustered target formulae
Sampled formulae ⟵ ∅
k ⟵ Sample one number from U(3,7)
c ⟵ Select K distinct clusters from Clustered target formulae
**foreach** *cluster in C* **do**
    q ⟵ Sample one number from U(3,7)
    s ⟵ Sample Q formulae from cluster
    append s to Sampled formulae
**end**

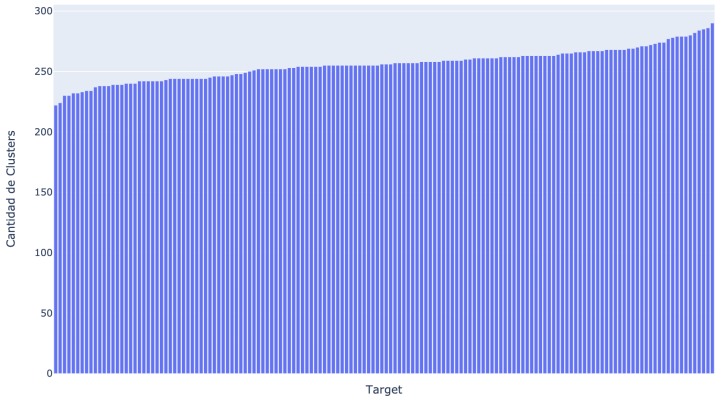


Figura 9: Cantidad de *clusters* distintos encontrados por la solución en cada conjunto de fórmulas generadas para el conjunto de 150 *targets* de prueba. Cada barra está asociada a un *target* y su altura indica la cantidad de *clusters* encontrados por la solución en el conjunto de fórmulas generado para dicho *target*.

agrupables en un tiempo razonable por un humano utilizamos el Algoritmo 1 con $a = c = 3$ y $b = d = 7$, derivando en conjuntos con una cardinalidad esperada igual a 25.

Cabe destacar que el Algoritmo 1 selecciona fórmulas basándose en los *clusters* encontrados por la solución desarrollada pues el proceso de validación que utilizamos (ver Sección 2.6) requiere de pares de fórmulas asignadas a *clusters* distintos y pares de fórmulas asignadas al mismo *cluster*. Considerando que la solución desarrollada suele encontrar más de 200 *clusters* distintos para conjuntos de 5000 fórmulas (ver Figura 9), si seleccionáramos 25 fórmulas al azar probablemente acabaríamos con fórmulas que pertenecen a alrededor de 25 *clusters* distintos. La Tabla 3 describe los conjuntos de fórmulas seleccionados por el Algoritmo 1 para los 5 *targets* distintos utilizados en el proceso de validación.

El proceso de validación consistió entonces en entregar el subconjunto de fórmulas seleccionado para un *target* a cada Chef y pedirle a este que, dado el subconjunto de fórmulas entregado, complete la siguiente tarea de *clustering*: *"Encontrar grupos de fórmulas con perfiles sensoriales similares"*. Dentro de las instrucciones especificamos además que el número de fórmulas por *cluster* puede ser variable, así como también el número de *clusters* existen-

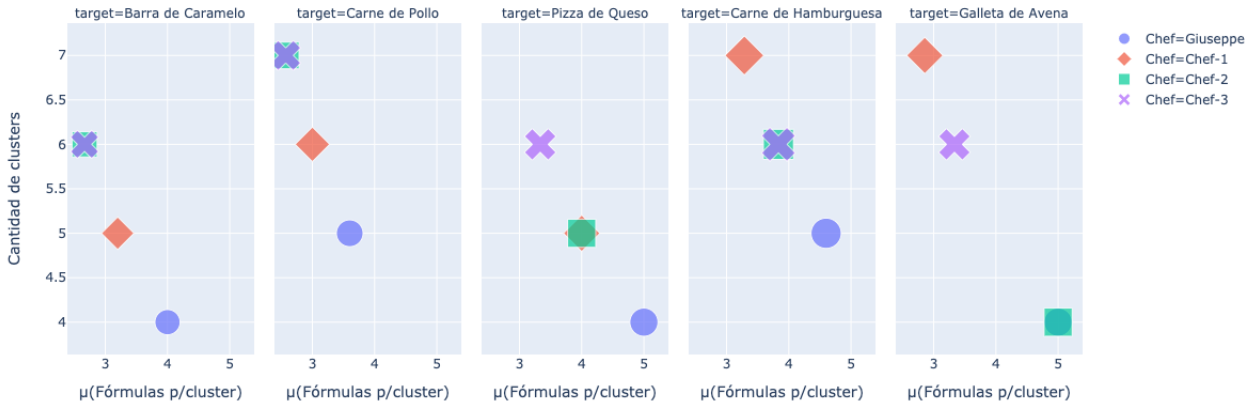


Figura 10: Comparación superficial de *clusters* encontrados por Chefs y la solución desarrollada (referida en el gráfico como Chef Giuseppe) en los conjuntos de fórmulas de validación.

| **Target** | **Precisión** | **Exhaustividad** | **Puntaje $F_1$** |
|---|---|---|---|
| Barra de Caramelo | 0.49 | 0.78 | 0.60 |
| Carne de Pollo | 0.31 | 0.38 | 0.34 |
| Pizza de Queso | 0.53 | 0.61 | 0.57 |
| Carne de Hamburguesa | 0.48 | 0.57 | 0.52 |
| Galleta de Avena | 0.66 | 0.84 | 0.74 |
| **PROMEDIO** | 0.49 | 0.64 | 0.55 |

Tabla 4: Precisión, exhaustividad y puntaje $F_1$ alcanzado por la solución desarrollada en los 5 *targets* de prueba.

tes por *target*. Adicionalmente, también les explicamos a los Chef que no todas las fórmulas necesariamente deben pertenecer a un *cluster*, es decir, los Chefs podían formar un grupo de fórmulas catalogadas como ruido. Se les dio todo el tiempo que necesitaran para completar la tarea y posteriormente recolectamos los resultados. Efectuamos el proceso de validación una vez por *target* para cada Chef.

### 4.7.2. Resultados de Validación

Antes que nada comparamos superficialmente los *clusters* encontrados por la solución desarrollada con los *clusters* encontrados por los Chefs para cada *target*. De la Figura 10 podemos ver que los Chefs tendieron a encontrar una mayor cantidad de *clusters* por *target* en comparación a la solución desarrollada, excepto en el *target* 5, en el cual la cantidad de *clusters* de la solución coincide con la cantidad de *clusters* del Chef #2. Dado que la solución desarrollada tiende a encontrar menos *clusters* que los Chefs sucede naturalmente que la cantidad de fórmulas promedio por *cluster*, encontradas por la solución desarrollada, es también mayor. Por otro lado, dos de los tres Chefs consultados tiene cantidades de *clusters* y cantidad de fórmulas promedio por *cluster* coincidentes en tres de los cinco *targets*.

La Tabla 4 muestra la precisión, exhaustividad y puntaje $F_1$ alcanzado por la solución desarrollada para los 5 *targets* de prueba seleccionados. Lo primero que notamos es que el

| Target | Precisión | Exhaustividad | Puntaje $F_1$ |
|---|---|---|---|
| Barra de Caramelo | 0.49 | 0.78 | 0.60 |
| Pizza de Queso | 0.53 | 0.61 | 0.57 |
| Galleta de Avena | 0.66 | 0.84 | 0.74 |
| **PROMEDIO** | 0.56 | 0.74 | 0.64 |

Tabla 5: Precisión, exhaustividad y puntaje $F_1$ alcanzado por la solución desarrollada en los *targets* de prueba no cárnicos.

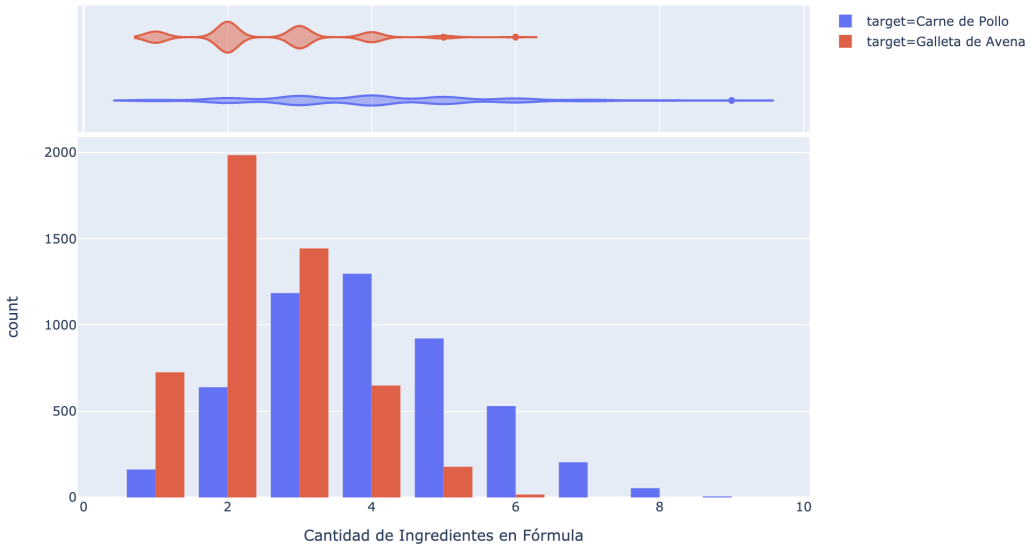


Figura 11: Histograma de cantidad de ingredientes por fórmula generada para dos de los cinco *targets* de validación: Carne de Pollo y Galleta de Avena.

rendimiento más bajo se obtuvo con los dos *targets* cárnicos seleccionados: Carne de Pollo y Carne de Hamburguesa, con una precisión, exhaustividad y puntaje $F_1$ de 0,31, 0,38, 0,34 y 0,48, 0,57, 0,52, respectivamente. Por otro lado, el target con mejores resultados fue la Galleta de Avena con una precisión de 0,66, una exhaustividad de 0,84 y un puntaje $F_1$ de 0,74.

Tabla 5 muestra los mismos resultados que la Tabla 4 pero sin considerar los alimentos cárnicos. De las Tabla 4 y 5 se desprende que la precisión, exhaustividad y puntaje $F_1$ promedio aumentan desde 0,49, 0,64, 0,55 a 0,56, 0,74 y 0,64, respectivamente, cuando no consideramos los *targets* cárnicos: Carne de Pollo y Carne de Hamburguesa.

Más aún, el que observemos buenos resultados para algunos *targets* (Galleta de Avena, Pizza de Queso, Barra de Caramelo) y no tan buenos para otros (Carne de Pollo y Carne de Hamburguesa) puede sugerir que la resolución requerida para aproximar correctamente el *manifold* de alta dimensión así como las características de los *clusters* formados varían de acuerdo a la categoría del *target*. Si esto fuera cierto, sería necesario ajustar los hiperpárametros de UMAP y de HDBSCAN dependiendo de la categoría a la que pertenezca el *target* (por ejemplo, alimento cárnico v.s alimento lácteo).

La diferencia en desempeño para alimentos cárnicos y no cárnicos puede deberse también a la complejidad de las fórmulas en términos de sus ingredientes. La Figura 11 muestra los histogramas de cantidades de ingredientes por fórmula para los *targets* Galleta de Avena y Carne de Pollo. Podemos ver que la cantidad de ingredientes por fórmula para Carne de Pollo tiende a ser mayor a la cantidad de ingredientes por fórmula para Galleta de Avena. En la misma línea, de la Tabla 3 podemos observar que la cantidad de ingredientes distintos totales presentes en las fórmulas de Carne de Pollo (1303) también supera a la cantidad de ingredientes distintos totales presentes en las fórmulas de Galleta de Avena (1133).

Una mayor variabilidad de ingredientes puede significar también una mayor variabilidad en los vectores de *características moleculares* de las fórmulas generadas y, por ende, aproximar el *manifold* no es igual de sencillo en ambos casos. La calidad de la representación en baja dimensión de las fórmulas depende directamente de la calidad de la aproximación del *manifold* en alta dimensión. Obtener una representación de las fórmulas en baja dimensión es importante pues, el algoritmo de *clustering* trabaja con esta representación, y en consecuencia, la validez de las familias de fórmulas identificadas por el algoritmo de *clustering* quedará sujeta a la calidad de la representación en baja dimensión.

Por último, la Figura 10 demuestra inmediatamente que la cantidad de *clusters* encontrada por la solución desarrollada no es la óptima, los Chefs lograron identificar al menos dos *clusters* adicionales para cada *target*. Esta diferencia en cantidad de *clusters* se ve reflejada también en los valores de precisión y exhaustividad alcanzados por la solución. Considerando que los valores de precisión y exhaustividad son aceptables, creemos que los *clusters* no identificados por la solución corresponden a *clusters* anidados (uno dentro del otro). Estos *clusters* anidados fueron separados efectivamente por los Chefs pero no por la solución.

## 4.8. Algoritmo de Selección de Fórmulas

Hasta este punto desarrollamos una solución capaz de encontrar, con cierto grado de efectividad, los *clusters* de fórmulas con *perfiles sensoriales* similares existentes dentro de un conjunto de fórmulas generadas por Giuseppe para un *target* particular. Sin embargo, y como explicamos en la Sección 1.2, Giuseppe genera del orden de miles de fórmulas distintas en cada ejecución, y no es factible entregar un conjunto tan extenso de fórmulas a los Chefs para que estos las exploren.

Producto de lo anterior, fue necesario diseñar un algoritmo de selección de fórmulas, el cual recibe como entrada el conjunto completo de miles de fórmulas generadas por Giuseppe con sus *clusters* de fórmulas *sensorialmente similares* ya identificados y retorna un subconjunto reducido de fórmulas de familias distintas y altamente similares al *target* en cuestión.

El algoritmo 2 detalla la lógica de selección de fórmulas implementada. El algoritmo de selección consiste en primero ordenar las fórmulas *clusterizadas* por su similitud al *target* asociado. Posteriormente, bajo este orden el algoritmo selecciona las $k$ primeras fórmulas de *clusters* distintos. Finalmente, cada uno de los $k$ *clusters* asociados a las fórmulas seleccionadas son también ordenados por similitud al *target* y se extraen las $m$ primeras fórmulas de cada uno.

El algoritmo de selección descrito tiene como resultado un conjunto de aproximadamente

**Algorithm 2:** Algoritmo de selección de fórmulas

**Result:** Selected formulae
**Input: Clustered formulae**, **K**, **M**
Clustered formulae ⟵ *sortBySimilarity*(**Clustered formulae**)
S ⟵ Select **K** *first* formulae from *different* clusters from **Clustered formulae**
Selected formulae ⟵ ∅
**foreach** *formula in S* **do**
    c ⟵ Get *formula* cluster members
    c ⟵ *sortBySimilarity*(c)
    append first **M** formulae in c to Selected formulae
**end**

$k \times m$ fórmulas pertenecientes a $k$ familias con *perfiles sensoriales* característicos distintos.

### 4.8.1. Comparación cuantitativa de procesos de selección de fórmulas

A continuación compararemos el proceso de selección actual con el nuevo proceso de selección propuesto en este trabajo. Como explicamos en la Sección 1.2.3 el proceso de selección actual corresponde a ordenar las fórmulas por similitud al *target* en cuestión, y luego, a través de una inspección superficial seleccionar un subconjunto de fórmulas distintas.

Para comparar la situación actual con el proceso de selección propuesto, medimos la cantidad de ingredientes distintos presentes en las $k$ fórmulas más similares a su *target* (situación actual) con las $k$ fórmulas con mayor similitud a su *target* pero de *clusters* distintos. Repetimos este análisis para los mismos 5 *targets* de prueba utilizados en el proceso de validación de *clusters* con expertos descritos en la Sección 4.7.

Los resultados de la comparación de ambos procesos de selección se muestran en la Figura 12. En ella podemos ver que en todos los *targets* de prueba la cantidad de ingredientes distintos del conjunto de fórmulas seleccionado por la solución propuesta en este trabajo es superior a si se ocupase el proceso de selección actual. La Figura 12 brinda evidencia de que el proceso de selección propuesta logra efectivamente captar fórmulas con una mayor diversidad de ingredientes.

## 4.9. Arquitectura de la Solución

La solución desarrollada se compone de 4 módulos lógicos distintos, cada uno encapsulado en un paquete de Python distinto: el *parser* de fórmulas, el *embedder* de fórmulas, el *clusterer* de fórmulas y, finalmente el seleccionador de fórmulas. El *parser* de fórmulas es utilizado para leer archivos de fórmulas generados por Giuseppe y transformar estos a un formato tabular conveniente de Pandas[9] en Python. El *embedder* encapsula la lógica de reducción de dimensión, abstrayéndose, al mismo tiempo, de cualquier algoritmo en específico. No obstante, el único algoritmo soportado, a la fecha, por el *embedder* de fórmulas es UMAP, pues UMAP fue el algoritmo de reducción de dimensión escogido para este proyecto. En la misma línea, el *clusterer* de fórmulas encapsula la lógica de *clustering* de fórmulas, y a la fecha, solo soporta

[9] `https://pandas.pydata.org/`

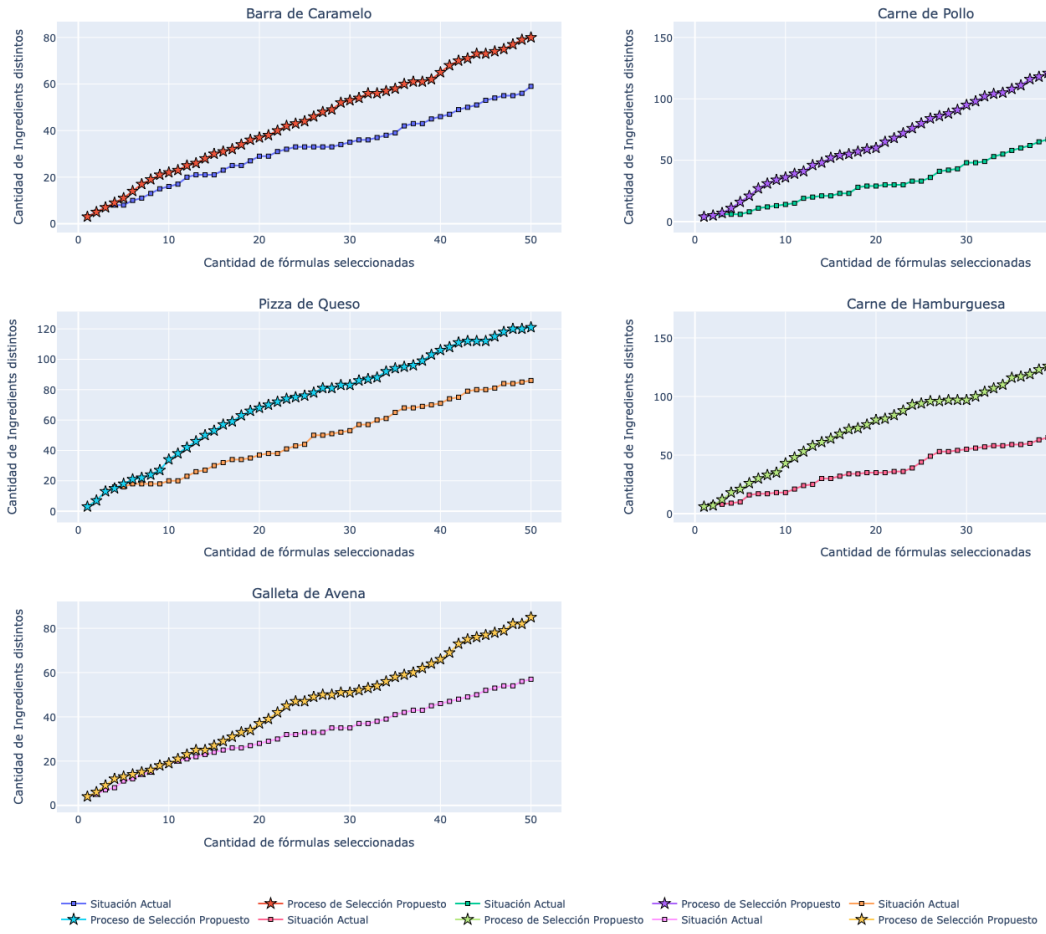


Figura 12: Comparación de cantidad de ingredientes distintos capturados por el proceso de selección actual v.s el propuesto.

el algoritmo de *clustering* HDBSCAN.

El seleccionador de fórmulas encapsula toda la lógica de selección de fórmulas interesantes. Dado un archivo con fórmulas generadas por Giuseppe para un *target* en particular, el seleccionador de fórmulas utiliza el *parser*, el *embedder* y el *clusterer* de fórmulas previamente descritos para encontrar todas las familias de fórmulas con *perfiles sensoriales* similares. El seleccionador incluye además el algoritmo de selección (ver Algoritmo 2) el cual utiliza para finalmente retornar un subconjunto de fórmulas interesantes con *perfiles sensoriales* distintos.

El seleccionador de fórmulas es capaz de soportar distintas lógicas que definen la noción de *fórmula interesante*, sin embargo, a la fecha de desarrollo de este proyecto, solo se soporta una única noción embrionaria de las $k$ *fórmulas interesantes*: las $k$ fórmulas más similares al *target* en cuestión. Los lectores interesados en los detalles técnicos de la arquitectura pueden consultar el apéndice A.1.

Finalmente, el módulo del seleccionador de fórmulas contiene una herramienta de línea de comando que recibe como entrada un conjunto de fórmulas generadas junto con los paráme-

tros $k$ y $m$ del algoritmo de selección (ver Algoritmo 2).

# 5. Conclusiones

El objetivo de este proyecto de título fue desarrollar un algoritmo que apoye un nuevo proceso de selección de fórmulas, el cual debe ser de carácter exploratorio, debe considerar el conjunto completo de fórmulas generadas por Giuseppe, debe ser capaz de identificar las distintas familias de fórmulas generadas, cada una definida por un *perfil sensorial* similar y su resultado debe permitir involucrar a los Chefs en el proceso de selección.

Utilizar HDBSCAN para descubrir los *clusters* deriva en una solución exploratoria pues HDBSCAN no asume una cantidad de *clusters* específica a encontrar, ni tampoco una forma particular de estos. Más bien, HDBSCAN identifica los *clusters* de fórmulas como regiones del espacio persistentemente densas a lo largo de distintos umbrales de densidad.

La solución considera además el conjunto completo de fórmulas generadas por Giuseppe y logra identificar, hasta cierto punto, las distintas familias de fórmulas, logrando un puntaje $F_1$ promedio de 0,55 cuando se compara el rendimiento de la solución con el rendimiento de los Chefs sobre la misma tarea en 5 *targets* de prueba. Desprendemos de el resultado mencionado anteriormente, que la solución desarrollada cumple solo parcialmente el objetivo específico numero uno.

El algoritmo de selección desarrollado (ver Algoritmo 2), permite seleccionar un conjunto de $\approx k \times m$ fórmulas, el cual contiene fórmulas de $k$ familias distintas, las $k$ fórmulas más similares al *target* en cuestión, y las $m$ fórmulas más similares al *target* en cuestión por cada familia seleccionada. Los resultados mostrados en la Figura 12 dan evidencia de que el algoritmo de selección propuesto logra seleccionar fórmulas con una mayor cantidad de ingredientes distintos en comparación a la situación actual, cumpliendo entonces con el objetivo específico numero dos.

El tercer y ultimo objetivo específico también fue cumplido. El resultado del algoritmo desarrollado es presentado a los Chefs a través de una herramienta web interna de la empresa, en la cual los Chefs pueden consultar las fórmulas seleccionadas para cada *target* y explorar las $k$ familias identificadas junto con las $m$ fórmulas que componen cada familia. Sin embargo, dado que se seleccionan a lo más $k \times m$ fórmulas, el proceso de selección no considera todas las fórmulas generadas, sino más bien un subconjunto de fórmulas de $k$ familias distintas. A pesar de esto, la cantidad de fórmulas exploradas por los Chefs es mayor a la situación actual, la cual contempla solo una inspección superficial del conjunto de fórmulas ordenadas.

Concluimos entonces que la solución desarrollada tiene un rendimiento aceptable pues es de carácter exploratoria, considera el conjunto completo de fórmulas para encontrar las familias de fórmulas con *perfiles sensoriales* similares, las que logra identificar pero hasta cierto punto. El resultado de la solución permite involucrar completamente a los Chefs en el proceso de selección de fórmulas ya que el subconjunto de fórmulas seleccionadas esta organizado por familia y además se pone a disposición de los Chefs a través de una herramienta web gráfica interna de la empresa. Finalmente, el resultado obtenido por la solución en el proceso de validación con expertos entrega evidencia de que el vector de características moleculares puede ser utilizado para representar las fórmulas y encontrar aquellas con *perfiles sensoriales* similares.

# 6. Trabajo Futuro

Como trabajo futuro, proponemos primero tratar de categorizar los *targets*, para luego seleccionar un conjunto de *targets* de prueba de manera representativa con respecto a las clases ya identificadas.

Posteriormente, podríamos comprobar si se obtienen mejores resultados ajustando los hiperparámetros de los algoritmos de reducción de dimensión y *clustering* para cada categoría de *target* encontrada. En la misma línea, sería interesante experimentar con otros algoritmos de reducción de dimensión y/o de *clustering* y comparar los resultados obtenidos con el mismo proceso de validación.

Es más, el proceso de validación aún es perfectible, sería particularmente interesante poder entregar conjuntos de fórmulas más grandes a los Chefs y lograr identificar cuando una solución desarrollada anida dos *clusters* que los Chefs separaran correctamente. Una opción para identificar estos *clusters* anidados es contar la cantidad de *clusters* distintos identificados por los Chefs que son asignados al mismo *cluster* por la solución desarrollada y luego experimentar con los hiperparámetros de HDBSCAN para comprobar si es posible separar los *clusters* anidados.

Por otro lado, la lógica de selección de fórmulas interesantes puede mejorarse considerando una combinación de la similitud de las fórmulas con el grado de novedad de la combinación de ingredientes presentes. En los sistemas de recomendación se utilizan medidas como la diversidad, serenidad y novedad para evaluar las recomendaciones que serán presentadas a los usuarios así como también para evaluar estas.

Otra métrica que sería interesante calcular en el futuro es la representatividad de la selección de fórmulas entregada a los Chefs, en términos de los ingredientes, o del espacio de características. Esta medida es relevante pues dice de manera concreta la seguridad con la que se está cubriendo el espacio de fórmulas generadas para encontrar el subconjunto de fórmulas seleccionadas, para NotCo es de vital importancia maximizar la confianza en que no existen fórmulas interesantes no detectadas.

Finalmente, dado que este proyecto trató fundamentalmente de desarrollar un algoritmo sería deseable en un futuro evaluar la complejidad del algoritmo, al menos experimentalmente, para saber cómo escala el algoritmo completo de selección de fórmulas (*embed* - *cluster* - *select*) con respecto a la cantidad de fórmulas, el tamaño de la dimensión original, el tamaño de la dimensión reducida y el impacto de los hiperparámetros.

# Bibliografía


[1] Belkin, Mikhail y Partha Niyogi: *Laplacian Eigenmaps and Spectral Techniques for Embedding and Clustering*. En *NIPS*, 2001.

[2] BELLMAN, RICHARD: *Adaptive Control Processes: A Guided Tour*. Princeton University Press, 1961, ISBN 9780691079011. http://www.jstor.org/stable/j.ctt183ph6v.

[3] Campello, Ricardo J. G. B., Davoud Moulavi y Joerg Sander: *Density-Based Clustering Based on Hierarchical Density Estimates*. En Pei, Jian, Vincent S. Tseng, Longbing Cao, Hiroshi Motoda y Guandong Xu (editores): *Advances in Knowledge Discovery and Data Mining*, páginas 160–172, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg, ISBN 978-3-642-37456-2.

[4] Domingos, Pedro: *A Few Useful Things to Know About Machine Learning*. Commun. ACM, 55(10):78–87, Octubre 2012, ISSN 0001-0782. http://doi.acm.org/10.1145/2347736.2347755.

[5] Fahad, A., N. Alshatri, Z. Tari, A. Alamri, I. Khalil, A. Y. Zomaya, S. Foufou y A. Bouras: *A Survey of Clustering Algorithms for Big Data: Taxonomy and Empirical Analysis*. IEEE Transactions on Emerging Topics in Computing, 2(3):267–279, Sep. 2014.

[6] Future, Plants for a: *Edible*. https://pfaf.org/user/edibleuses.aspx.

[7] Hatzivassiloglou, Vasileios y Kathleen R. McKeown: *TOWARDS THE AUTOMATIC IDENTIFICATION OF ADJECTIVAL SCALES: CLUSTERING ADJECTIVES ACCORDING TO MEANING*. En *31st Annual Meeting of the Association for Computational Linguistics*, páginas 172–182, Columbus, Ohio, USA, Junio 1993. Association for Computational Linguistics. https://www.aclweb.org/anthology/P93-1023.

[8] Hoekstra, Arjen Y.: *The hidden water resource use behind meat and dairy*. Animal Frontiers, 2(2):3–8, Abril 2012, ISSN 2160-6056. https://doi.org/10.2527/af.2012-0038.

[9] Jiawei Han, Micheline Kamber, Jian Pei: *Data Mining: Concepts and Techniques 3rd Edition*, páginas 444–450. Elsevier, Waltham, MA, 2012, ISBN 9780123814791.

[10] Lawson, Richard G. y Peter C. Jurs: *New index for clustering tendency and its application to chemical problems*. Journal of Chemical Information and Computer Sciences, 30(1):36–41, 1990. https://pubs.acs.org/doi/abs/10.1021/ci00065a010.

[11] Lecun, Y., L. Bottou, Y. Bengio y P. Haffner: *Gradient-based learning applied to document recognition*. Proceedings of the IEEE, 86(11):2278–2324, Nov 1998, ISSN 1558-2256.

[12] Lipp, Johannes y Eric W Weisstein: *Topological Space*. http://mathworld.wolfram.com/TopologicalSpace.html.

[13] Maaten, Laurens van der, Eric Postma y H. Herik: *Dimensionality Reduction: A Com-*

*parative Review*. Journal of Machine Learning Research - JMLR, 10, Enero 2007.

[14] McInnes, Leland, John Healy y James Melville: *UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction*. arXiv e-prints, página arXiv:1802.03426, Feb 2018.

[15] Moulavi, Davoud, Pablo A Jaskowiak, Ricardo Campello, Arthur Zimek y Joerg Sander: *Density-Based Clustering Validation*. Abril 2014.

[16] nLab authors: *nerve theorem*. `http://ncatlab.org/nlab/show/nerve%20theorem`, Noviembre 2019. Revision 16.

[17] Rowland, Todd: *Manifold*. `http://mathworld.wolfram.com/Manifold.html`.

[18] Willis, K.J (ed.): *State of the World's Plants 2017*, 2017, ISBN 978-1-84246-647-6.

[19] Xiao, Han, Kashif Rasul y Roland Vollgraf: *Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms*, 2017.

# Apendice

# A. Arquitectura de la Solución

La solución desarrollada tiene dos componentes principales: la reducción de dimensión y el *clustering* de las fórmulas con su dimensión reducida. No obstante, existen dos componentes adicionales necesarias para cumplir el objetivo planteado en este proyecto, un *parser* de fórmulas y un *seleccionador* de fórmulas. En esta sección explicaremos cada componente de la solución desarrollada en un orden lógico y finalmente explicaremos como se integran todas las componentes desarrolladas dentro de una sola herramienta de linea de comando encargada de recibir conjuntos de fórmulas y entregar como resultado las familias identificadas.

De acuerdo los requisitos técnicos planteados en la Sección 3.4 todas las componentes de la solución fueron desarrolladas en Python. Además, considerando la adopción de la empresa de tipos de datos tabulares todos los componentes desarrollados pueden recibir datos tabulares en *DataFrames* de *Pandas* o en arreglos multidimensionales de *Numpy*.

## A.1. *Parser* de Fórmulas

El conjunto de fórmulas generado por Giuseppe, dado un *target* especifico, es entregado en un único archivo en formato JSON. Este archivo contiene una lista de diccionarios donde cada fórmula generada es un diccionario distinto. Como explicamos en la Sección 3.1, cada fórmula se compone de una lista de ingredientes vegetales con sus respectivas concentraciones, un vector de *características moleculares* y la similitud de la fórmula al *target* que busca reproducir dicha fórmula. A modo de ejemplo, mostramos en la Figura 13 un extracto de un archivo de fórmulas generado por Giuseppe.

Los archivos JSON de fórmulas generados por Giuseppe contienen una estructura que los hace medianamente legibles por un ser humano. Sin embargo, en este proyecto necesitamos extraer los vectores de *características moleculares* y los ingredientes presentes en cada fórmula. Para lograr esto, encapsulamos la lógica de reestructuración y ordenamiento de los datos en un paquete llamado *fParser*. Este paquete contiene una sola clase llamada *FormulaParser* cuyas instancias reciben como entrada la ubicación del archivo JSON de fórmulas y entregan como salida una tres-tupla cuyos contenidos especificamos a continuación:

- Meta-data de fórmulas: *DataFrame* de *Pandas* con las columnas ID de Fórmula, Similitud al *target*, Ingredientes y Vector de *características moleculares*. Las dos ultimas columnas contienen diccionarios donde las llaves son los nombres de los ingredientes/-características moleculares y los valores asociados las cantidades de los ingredientes/-características moleculares.
- Ingredientes: *DataFrame* de *Pandas* con tantas columnas como ingredientes distintos existan en el archivo de fórmulas entregado y tantas filas como fórmulas existan en el archivo de fórmulas entregado. Cada fila contiene tantos valores no nulos como ingredientes tenga la fórmula asociada a la fila.
- *Características Moleculares*: *DataFrame* de *Pandas* con una columna por *característica molecular* y tantas filas como fórmulas existan en el archivo de fórmulas entregado.

```
{
  "id": "11",
  "delta": 0.6272861442585452,
  "ingredients": {
    "Ingredient_1_1": 0.26404084011591883,
    "Ingredient_1_2": 0.0023833559339390873,
    "Ingredient_1_3": 0.002701468206264985,
    "Ingredient_1_4": 0.6664830272870856,
    "Ingredient_1_5": 0.0022579673060317837,
    "Ingredient_1_6": 0.062133341150759705
  },
  "features": {
    "feature_1": 0.15228421679883258,
    "feature_2": 0.12137189991235264,
    "feature_3": 0.4266695907736041,
    "feature_4": 0.4557185299179193 3,
    "feature_5": 0,
    ...
  }
},
{
  "id": "14",
  "delta": 0.6144296486211551,
  "ingredients": {
    "Ingredient_2_1": 0.360615083185449,
    "Ingredient_2_2": 0.0016890079241304904,
    "Ingredient_2_3": 0.0023037271472223543,
    "Ingredient_2_4": 0.6044785074889061,
    "Ingredient_2_5": 0.002367408562459869,
    "Ingredient_2_6": 0.02760773233826715,
    "Ingredient_2_7": 0.000938533353565134
  },
  "features": {
    "feature_1": 0.30533958915459664,
    "feature_2": 0.381001814450619,
    "feature_3": 0.4095489313683589,
    "feature_4": 0.6170137923268035,
    "feature_5": 0,
    ...
  }
},
```


Figura 13: Extracto de archivo de fórmulas generado por Giuseppe. Los nombres originales de los ingredientes y de las *características moleculares* fueron cambiados por nombres genéricos.

Utilizamos el *DataFrame* de meta-data para mantener una asociación entre ID de fórmula, sus ingredientes y su vector de *características moleculares* así como también para guardar, posteriormente, los *clusters* asignados a cada fórmula. Por otro lado, utilizamos el *DataFrame* de Ingredientes para representar cada fórmula por sus ingredientes presentes y computar los índices de Jaccard promedio por *cluster* mostrados en la Figura 7. Por último, utilizamos el *DataFrame* de *características moleculares* para obtener la representación escogida en la Sección 4.2 para cada fórmula: los vectores de *características moleculares*, los cuales corresponden a las filas del *DataFrame* de *características moleculares*.

## A.2. *Embedder* de Fórmulas

En la Sección 4.3 explicamos la necesidad de reducir la dimensión de las fórmulas dada la representación escogida para estas: los vectores de *características moleculares*. Encapsulamos la lógica de reducción de dimensión en un paquete llamado *fEmbedder*, este paquete contiene una única clase llamada *FormulaEmbedder*.

El diseño de la clase *FormulaEmbedder* sigue el estándar adoptado por los modelos de la librería *scikit-learn* de Python, es decir, la clase *FormulaEmbedder* contiene tres funciones: `fit`, `transform` y `fit_transform`. La función `fit` se utiliza para entregarle un conjunto de

datos al paquete para que posteriormente se reduzca su dimensión llamando a la función `transform`. La función `fit_transform` ejecuta ambos pasos mencionados anteriormente de una sola vez y sobre el conjunto de datos entregado como entrada. Además, el constructor de la clase *FormulaEmbedder* tiene un parámetro `method` el cual especifica el algoritmo a utilizar para reducir la dimensión de los datos.

La salida de las funciones `transform` y `fit_transform` corresponde a una matriz de *numpy* con tantas filas como fórmulas fueron entregadas a la función `fit` ó `fit_transform`, respectivamente, y tantas columnas como dimensiones tienen los datos con su dimensión reducida. Es importante notar aquí que la cantidad de dimensiones a las que serán reducidos los datos debe ser especificada a través del constructor de la clase *FormulaEmbedder*.

Si bien a la fecha de desarrollo de este proyecto el único algoritmo soportado es UMAP, diseñar la clase *FormulaEmbedder* de esta manera permite, en el futuro, añadir fácilmente soporte para otros algoritmos de reducción de dimensión.

## A.3. *Clusterer* de Fórmulas

La lógica del proceso de *clustering* de las fórmulas, el cual lleva a identificar las familias de fórmulas con *perfiles sensoriales* similares fue encapsulada en un paquete llamado *fClusterer*. Este paquete contiene una sola clase llamada *FormulaClusterer*.

La clase *FormulaClusterer* sigue la misma filosofía de diseño que la clase *FormulaEmbedder* pues contiene tres funciones principales `fit`, `predict` y `fit_predict`, las cuales son análogas a las funciones `fit`, `transform` y `fit_transform` de la clase *FormulaEmbedder*, con la diferencia que la clase *FormulaClusterer* busca *predecir* una asignación de *cluster* para cada punto y no *transformar* los puntos. Así mismo, el constructor de la clase *FormulaClusterer* también tiene un parámetro `method` el cual especifica el algoritmo de *clustering* a utilizar.

Adicionalmente, la clase *FormulaClusterer* contiene otra función: `clustering_validity`. Esta función recibe un conjunto de datos y sus *clusters* asociados y computa alguna medida de validez del *clustering* dependiente de la naturaleza del algoritmo de *clustering* escogido. Por ejemplo, cuando el algoritmo de *clustering* utilizado es HDBSCAN (i.e. `method = HDBSCAN`), la medida de validez calculada es DBCV.

## A.4. Seleccionador de Fórmulas

Cada una de las clases *FormulaParser*, *FormulaEmbedder* y *FormulaClusterer* encapsula únicamente sus responsabilidades. La clase *FormulaParser* recibe la ruta a un archivo JSON y retorna los *DataFrames* descritos en la Sección A.1. La clase *FormulaEmbedder* recibe una matriz de *Numpy* con una fórmula por fila y una *característica molecular* por columna a través de sus funciones `fit` ó `fit_transform`, y retorna, a través de sus funciones `transform` ó `fit_transform`, otra matriz de *Numpy* con la misma asignación de filas, pero con tantas columnas como dimensiones tienen los datos con dimensión reducida. Finalmente, la clase *FormulaClusterer* también recibe una matriz de *Numpy* con una fórmula por fila y tantas columnas como características tengan las fórmulas, a través de sus funciones `fit` ó `fit_predict`, y retorna, a través de sus funciones `predict` ó `fit_predict`, un vector de *Numpy* con la misma asignación de filas, pero con una sola columna que contiene el *cluster*

asignado a cada fórmula.

Desarrollamos un paquete llamado *fSelector* que contiene una sola clase llamada *FormulaSelector*, la función de esta clase es integrar las clases *FormulaParser*, *FormulaEmbedder* y *FormulaClusterer*. Esta clase recibe por ende, la ruta al archivo de fórmulas, requerida por la clase *FormulaParser*, además de los algoritmos (i.e. los valores de `method`) de reducción de dimensión y *clustering* junto con sus hiperparametros requeridos. La clase *FormulaSelector* se encarga de *parsear* las fórmulas, reducir su dimensión y posteriormente *clusterizar* las fórmulas con su dimensión reducida. De manera análoga a las clases *FormulaEmbedder* y *FormulaClusterer*, la clase *FormulaSelector* contiene tres funciones principales: `fit`, `select` y `fit_select`.

La clase *FormulaSelector* se encarga de agregar una columna adicional al *DataFrame* de meta-data de las fórmulas, la cual contiene el *cluster* (la familia) asignado(a) a cada fórmula. Tenemos entonces, el mismo conjunto de fórmulas original pero con un *cluster* asignado a cada fórmula. Generalmente, el conjunto de fórmulas original contiene mas de 1000 fórmulas distintas, y por ende, es necesario aplicar una lógica de selección de fórmulas que permita seleccionar un subconjunto de fórmulas interesantes de un tamaño tal que los Chefs puedan revisar en un tiempo razonable. A las fórmulas interesantes les llamamos en este contexto, fórmulas representantes pues decimos que son las fórmulas representativas de sus familias.

La lógica de selección de las fórmulas representantes corresponde a en primer lugar ordenar las fórmulas por sus similitudes al *target* y luego, en segundo lugar, identificar las $k$ fórmulas mas interesantes como las $k$ fórmulas de familias distintas más similares al *target* en cuestión (el *target* para el cual fueron generadas las fórmulas originalmente). Posteriormente, para cada una de las $k$ familias correspondientes a las $k$ fórmulas seleccionadas, seleccionamos también las $m$ mejores fórmulas de cada familia.

La clase *FormulaSelector* permite entonces, dado un conjunto de fórmulas generadas por Giuseppe para un *target* específico, seleccionar un subconjunto de a lo más $k \times m$ fórmulas de $k$ familias distintas y que sabemos que contiene a las $k$ fórmulas de familias distintas más similares al *target* en cuestión.

Concluimos entonces, que la clase *FormulaSelector* lleva a cabo el proceso de selección de fórmulas descrito en la Sección 3.1 pues logra identificar un subconjunto de fórmulas con ingredientes distintos, ya que pertenecen a $k$ familias distintas. Además, el subconjunto de fórmulas seleccionado es interesante pues contiene a las $k$ fórmulas mas similares al *target*. Por otro lado, el subconjunto seleccionado incluye además a las $m$ fórmulas mas similares al *target* dentro de cada una de las $k$ familias correspondientes a las fórmulas seleccionadas.

## A.5. Herramienta de Linea de Comando

En NotCo se generan fórmulas para nuevos *targets* diariamente. Esto quiere decir que todos los días se generan nuevos conjuntos de miles de fórmulas para *targets* distintos, y como dijimos en la Sección 3.1, cada uno de estos conjuntos de fórmulas debe pasar por el proceso de selección de fórmulas. Como explicamos en la Sección A.4, la clase *FormulaSelector* es capaz de completar el proceso de selección automáticamente.

Considerando el ritmo de generación de fórmulas diario es que nació la necesidad de desarrollar una herramienta de linea de comando que envuelve a la clase *FormulaSelector*. Esta herramienta de linea de comando utiliza el seleccionador de fórmulas para identificar las $k$ fórmulas representantes (las mas interesantes) y, además de eso, retornar $m$ integrantes de cada una de las $k$ familias de las $k$ fórmulas seleccionadas. Adicionalmente, la herramienta de linea de comando permite especificar los algoritmos de reducción de dimensión y *clustering* junto con sus hiperpárametros. A continuación describimos el parámetro y las opciones de la herramienta de linea de comando desarrollada:

- `src` : Único parámetro de la herramienta. Corresponde a la ruta absoluta al archivo JSON que contiene el conjunto de fórmulas generadas por Giuseppe para un *target* específico.
- `embed_method` : Parámetro para especificar el algoritmo de reducción de dimensión a utilizar. A la fecha de desarrollo de este proyecto solo UMAP es soportado.
- `target_dim` : Cantidad de dimensiones a las que se reducirán las fórmulas especificadas por el parámetro `source`.
- `n_neighbors` : Hiperpárametro de UMAP, para mas detalles consultar la Sección 2.3.1.
- `min_dist` : Hiperpárametro de UMAP, para mas detalles consultar la Sección 2.3.1.
- `cluster_method` : Parámetro para especificar el algoritmo de *clustering* a utilizar. A la fecha de desarrollo de este proyecto solo HDBSCAN es soportado.
- `min_samples` : Hiparámetros de HDBSCAN, para mas detalles consultar la Sección 2.4.2.
- `min_cluster_size` : Hiparámetros de HDBSCAN, para mas detalles consultar la Sección 2.4.2.
- `select_k` : Cantidad $k$ de fórmulas que el *FormulaSelector* seleccionará como representantes.
- `select_m` : Cantidad $m$ de fórmulas a seleccionar por cada familia de las $k$ fórmulas seleccionadas como representantes.

La herramienta de linea de comando genera como resultado un directorio nuevo que contiene un archivo con todas las fórmulas seleccionadas con la meta-data correspondiente para cada una (identificador, similitud al *target*, ingredientes, vector de *características moleculares* y *cluster* asignado). Las fórmulas seleccionadas son subidas a una herramienta web interna de la empresa, en la cual los Chefs pueden revisar las familias y las fórmulas seleccionadas. La Figura 14 muestra una parte de la herramienta web interna en la que se pueden ver dos fórmulas representantes para el *target blood*, es posible explorar las familias de cada fórmula haciendo click en el botón titulado *More formulae in this family*.

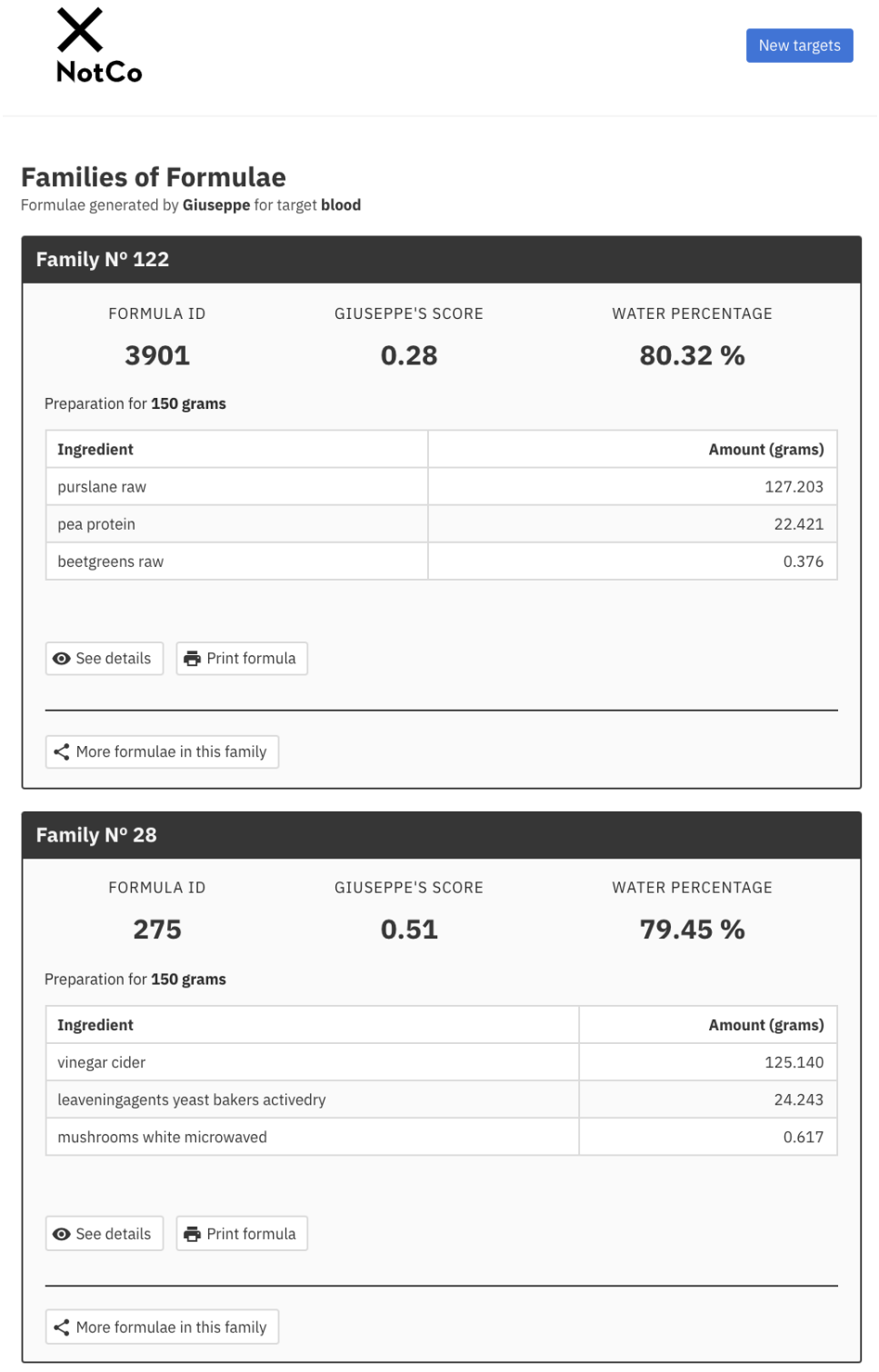


Figura 14: Herramienta web interna utilizada por los Chefs para explorar las fórmulas seleccionadas por la solución desarrollada así como también sus respectivas familias.