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

OPTIMIZACIÓN DEL MANTENIMIENTO PREVENTIVO DE FLOTAS EN BASE A TÉCNICAS DE CLUSTERING Y APRENDIZAJE SUPERVISADO

MEMORIA PARA OPTAR AL TÍTULO DE

INGENIERO CIVIL MECÁNICO

GASTÓN BÜTIKOFER LAGOS

PROFESOR GUÍA:

ENRIQUE LÓPEZ DROGUETT

MIEMBROS DE LA COMISIÓN:

VIVIANA MERUANE NARANJO

EDUARDO SALAMANCA HENRÍQUEZ

SANTIAGO DE CHILE

2017

RESUMEN DE LA MEMORIA PARA OPTAR
AL TÍTULO DE INGENIERO CIVIL MECÁNICO
POR: GASTÓN BÜTIKOFER LAGOS
FECHA: 2017
PROF. GUÍA: ENRIQUE LÓPEZ DROGUETT

OPTIMIZACIÓN DEL MANTENIMIENTO PREVENTIVO DE FLOTAS EN BASE A TÉCNICAS DE CLUSTERING Y APRENDIZAJE SUPERVISADO

El proyecto consistió en desarrollar una metodología para la optimización del plan de mantenimiento preventivo de una flota heterogénea subdividiendo ésta en subpoblaciones homogéneas. La flota de estudio de este trabajo fue una flota de camiones para la gran minería de la que se tiene una base de datos con información de horómetros de camiones, historial de motores y muestras de espectrometría. Se decidió trabajar en torno a los motores QSK.

La metodología consistió principalmente en la obtención de datos de la flota junto al estudio de ésta, la aplicación del algoritmo para segmentar flota, luego gracias a la subdivisión se entrenó un clasificador, utilizando los resultados del proceso de segmentación, para el diagnóstico de un motor en funcionamiento asignándolo a un subgrupo. Se calculó la confiabilidad de cada subgrupo y finalmente se modeló cada subgrupo obteniendo así tiempos óptimos para la toma de decisión con respecto al momento de enviar un motor a mantenimiento.

Para la obtención de las subpoblaciones se trabajó con k-means++ mediante el software python 3.0 a una base de datos de la flota con información de las covariables relevantes a los mecanismos de falla. Se obtuvo 4 clústers cada uno de estos asociados a un mecanismo de falla diferente: falla por edad, falla destructiva debido a filtración líquido de enfriamiento, falla en elementos del motor y filtración de agua. El clasificador para nuevos elementos de la flota fue una red neuronal MLP. El grafo de la red neuronal es de una entrada con 24 neuronas, una capa oculta de 14 neuronas y salida de 4 neuronas.

Luego se determinó el plan óptimo de mantenimiento preventivo de cada subpoblación minimizando costos asociados a reemplazos de equipos o minimizando la indisponibilidad de los equipos. Los tiempos óptimos calculados son inferiores al MTTF por lo que se propone mantenimiento preventivo. Las confiabilidades de los subgrupos difieren, lo que es un gran problema al trabajar con la flota como si fuera homogénea.

Esta metodología identificó subpoblaciones homogéneas con respecto a la totalidad de la flota permitiendo así un correcto estudio de la confiabilidad de la flota aportando información para la toma de decisiones y una correcta administración.

En memoria de Fernando Lagos, José Luis Gunther Bütikofer y Tango.

AGRADECIMIENTOS

En primer lugar, quisiera agradecer a mi familia: Mis padres, Gastón y Teresa; Mis hermanos, Federico y Raimundo; Mis abuelas: María Teresa y Arecia; Mis primos y tíos.

En segundo lugar, agradecer a todas las personas con las que he compartido a lo largo de mi vida en el colegio, en Bartolomé de las Casas, en scout, en la universidad, en Tongoy…

En tercer lugar, dar gracias a todos los profesores que he tenido. Siempre seré un estudiante agradecido de su dedicación y entrega, aprendí mucho de ellos y los tendré siempre presente.

Y finalmente a mi profesor guía Enrique López por haberme aceptado como memorista. En el desarrollo del trabajo, siempre estuvo presente mostrando interés y siendo un apoyo importante. Además, agradecer a S.A, I.M y J.D por su aporte.

TABLA DE CONTENIDO

1 Introducción 1
1.1 Motivación 1
1.2 Objetivos 1
1.2.1 Objetivo General 1
1.2.2 Objetivos Específicos 2
1.3 Alcances 2
2 Metodología 3
2.1 Obtención Base de Datos 4
2.2 Estudio de Flota 4
2.3 Tratamiento de Datos 4
2.4 Subdividir Población en Grupos Homogéneos 4
2.5 Entrenar Clasificador 4
2.6 Distribución de Tiempo de Falla 5
2.7 Modelamiento de la confiabilidad en cada sub población 5
2.8 Análisis de Resultado y Conclusiones 5
3 Antecedentes y Discusión Bibliográfica 6
3.1 Flota de Estudio: Camiones para la gran minería 6
3.1.1 Introducción 6
3.1.2 Motor QSK 6
3.1.3 El análisis de Aceite 7
3.2 Agrupación de datos en Clústers K-MEANS ++ 9
3.2.1 k-means y k-means++ 9
3.2.2 Definiciones 10
3.2.3 Algoritmo k-means 10
3.2.4 Algoritmo k-means++ 11
3.2.5 Silhouette Score 11
3.3 Clasificador Multi Capas 12
3.3.1 Descripción 12
3.3.2 Evaluación de Resultados 14
3.4 Confiabilidad por Clúster 15
3.4.1 Distribuciones sobre tiempos de falla 15
3.4.2 P-valor 15

3.5 Optimización de Plan de Mantenimiento de cada Clúster 15
4 Resultados 17
4.1 Base de datos 17
4.2 Clúster 19
4.3 Clasificador 20
4.4 Descripción de Subgrupos 22
4.5 Plan de Mantenimiento 27
5 Análisis de Resultados 28
6 Conclusiones 31
7 Bibliografía 32
ANEXOS 33
Anexo A Código Primera Etapa Tratamiento de Datos 33
Anexo B Código Segunda Etapa Tratamiento de Datos 38
Anexo C Código Cluster 43
Anexo D Multilayer Perceptron 46

Índice de Tablas

Tabla 3-1 Criterios seguridad por elemento según resultados de análisis de aceite 8

Tabla 4-1 Evaluación Clasificador MLP 21

Tabla 4-2 Características Subgrupo Fe y Cu 22

Tabla 4-3 Características Subgrupo Ni y Sn 22

Tabla 4-4 Características Subgrupo Al y Si 23

Tabla 4-5 Características Subgrupo Pb y Cr 23

Tabla 4-6 Características Subgrupo Na y K 24

Tabla 4-7 Características Subgrupo B y Ag 24

Tabla 4-8 Características Subgrupo Horas 25

Tabla 4-9 Ajuste distribución Normal 25

Tabla 4-10 Ajuste distribución Lognormal 25

Tabla 4-11 Ajuste distribución Weibull 25

Tabla 4-12 MTTF por Clúster 26

Tabla 4-13 Resultados Confiabilidad a 18000 y 16721 horas 27

Tabla 4-14 Resultados tiempo óptimo por Clúster 27

Índice de Figuras

Figura 2-1 Metodología empleada en el trabajo 3

Figura 3-1 Neurona Artificial 12

Figura 3-2 Arquitectura MLP con una capa oculta (adaptada [6]) 13

Figura 3-3 Ejemplo Matriz de Confusión 14

Figura 4-1 Resultados Clúster ordenados por Clase 19

Figura 4-2 Silhouette Score 20

Figura 4-3 Matriz de Confusión 21

Figura 4-4 Distribuciones de Probabilidad 26

1 Introducción

Uno de los grandes desafíos de la gestión de una flota es la predicción de la confiabilidad y, con base a esto, realizar un buen plan de mantención preventiva. Diferentes condiciones de operación, diferentes cargas de funcionamiento, ambientes dañinos u otros imprevistos producen que haya heterogeneidad dentro de la flota.

Los métodos modernos de modelación basados en datos generalmente consideran esta incertidumbre sobre un análisis unidad por unidad. Al contabilizar la incertidumbre con este análisis, un analista puede instituir el mantenimiento basado en la condición de los equipos en una flota o subconjunto de esta.

Por lo que es importante saber clasificar estos equipos en grupos y realizar la evaluación de la confiabilidad según grupo lo que permite bajar costos y mejorar el plan de mantenimiento preventivo. Una forma de realizar esta clasificación es a través de la segmentación de la flota en sub-poblaciones en base a técnicas de *clustering* de las covariables relevantes a los mecanismos de falla. Luego, realizar la modelación de la confiabilidad. Esta metodología permite la clasificación de un nuevo equipo al ser integrado en la flota para el cual no se tiene experiencia operacional. Esto se realiza a través de técnicas de aprendizaje supervisado (como *Support Vector Machines o Multilayer Perceptron*) utilizando como etiquetas los resultados de segmentación entregados por la etapa de clustering. Con este resultado, se puede determinar a cuál clúster (sub población) este equipo pertenece. De esta manera, se logra tomar mejores decisiones sobre políticas de mantenimiento preventivo, pues la misma será desarrollada para la realidad operacional de cada clúster perteneciente a la flota de equipos.

1.1 Motivación

Modelos actuales de confiabilidad son ajustados para toda la flota y no toman en cuenta las condiciones ambientales y de operación de cada sub-población en la flota resultando en planes de mantención alejadas de la realidad y por lo tanto con altos costos.

Hay una necesidad de desarrollar y optimizar un calendario de mantención preventiva para cada sub-población de equipos de la flota. Éste debe considerar las covarariables relevantes a los mecanismos de falla.

1.2 Objetivos

1.2.1 Objetivo General

Modelación de la confiablidad de una flota con la respectiva optimización del mantenimiento preventivo a través del uso de técnicas de *clustering* y aprendizaje

supervisado para el modelamiento de la confiablidad dentro de cada sub-población de equipos.

1.2.2 Objetivos Específicos

Los objetivos específicos son:

- Uso de técnicas de clustering para agrupar elementos de una flota heterogénea en grupos homogéneos.
- Uso técnicas de clasificación para asignar los equipos a los clústers correspondientes.
- Utilizar métricas de la estimación de confiablidad con el fin de desarrollar una estrategia óptima de mantenimiento preventivo. En este trabajo el mantenimiento preventivo se refiere al reemplazo programado de un componente o sistema dado.

1.3 Alcances

Obtención de un plan de mantenimiento preventivo óptimo para una flota utilizando herramientas en estado de arte que permita tomar decisiones estratégicas en la gestión de una flota.

2 Metodología

La metodología empleada en el trabajo, para cumplir con los objetivos se resume en un esquema en la Figura 2-1.

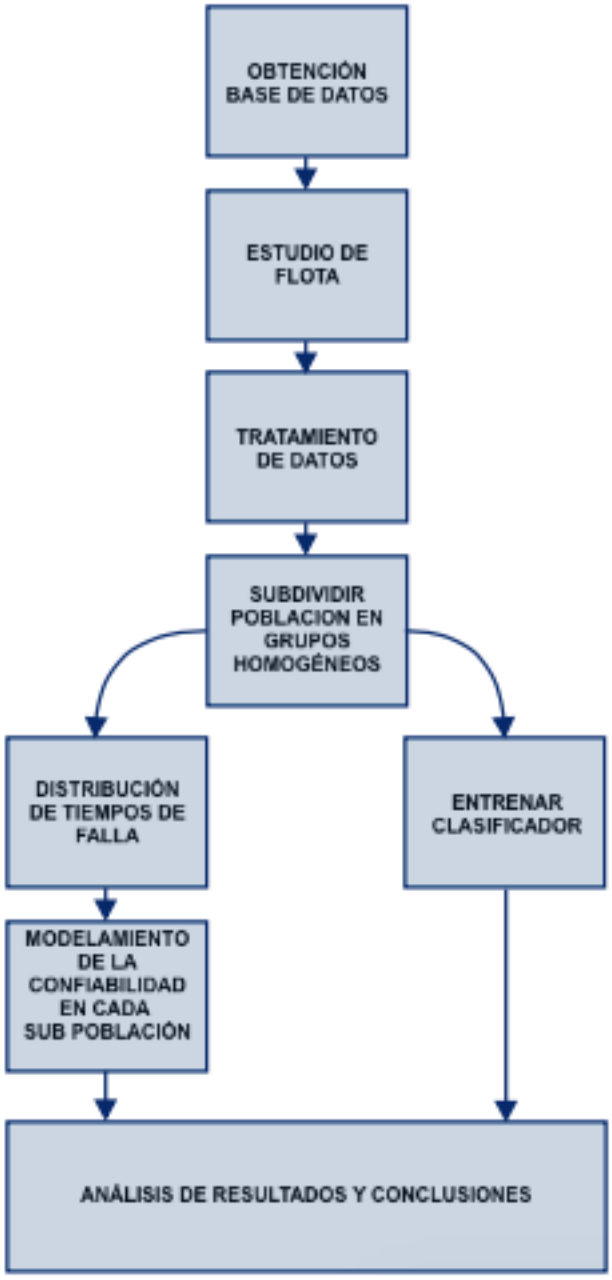

Figura 2-1 Metodología empleada en el trabajo

2.1 Obtención Base de Datos

Obtener base de datos sobre flota, se deben tener covariables relevantes a los mecanismos de falla. La flota a estudiar debe ser heterogénea es decir los equipos de la flota deben tener condiciones de operación diferentes. En el trabajo se obtuvo una base de datos de camiones utilizados en la minería. Los datos brutos consisten: horómetros de los camiones, información de los estados de los motores y análisis de aceite de los camiones. Para el análisis de aceite se tiene muestras de espectrometría.

2.2 Estudio de Flota

Se deben estudiar los mecanismos de falla de los motores. Para esto se debe realizar una revisión bibliográfica. Además, se debe analizar la base de datos obtenida y proyecciones del trabajo.

2.3 Tratamiento de Datos

En esta etapa se debe realizar un tratamiento de datos que permita relacionar la información, se trabaja con información entregada por la empresa.

2.4 Subdividir Población en Grupos Homogéneos

Para la segmentación de la flota se debe probar algoritmos para realizar clúster como k-means++, iterar hasta encontrar subdivisión de la población, esto deber ser testeado y validada. Para esto se deberá determinar la cantidad de clúster esperada. Otro factor relevante es determinar el formato de la base de datos para la entrada del algoritmo. El algoritmo será implementado en python.

2.5 Entrenar Clasificador

Con los resultados obtenidos por el proceso de segmentación se debe entrenar un clasificador que permita diagnosticar el estado de un motor en funcionamiento, gracias al análisis de aceite. Se trabaja principalmente con clasificadores disponibles tanto en TensorFlow y scikit-learn para implementar en python. El clasificador debe ser evaluado y validado.

2.6 Distribución de Tiempo de Falla

Asignar a cada subgrupo una distribución de probabilidad y evaluar el ajuste. Calcular la confiabilidad de cada subgrupo.

2.7 Modelamiento de la confiabilidad en cada sub población

Gracias a los resultados del proceso de clustering se realiza el modelamiento de la confiabilidad de cada clúster. Utilizando como referencia lo propuesto por A. Jardine y A. Tsang.

2.8 Análisis de Resultado y Conclusiones

Una vez realizado los pasos anteriores se debe analizar los resultados presentado un plan de mantenimiento de la flota, que considere los resultados del trabajo y aspectos económicos.

3 Antecedentes y Discusión Bibliográfica

3.1 Flota de Estudio: Camiones para la gran minería

3.1.1 Introducción

Los camiones de alta capacidad de carga utilizados en el transporte de cobré, carbón u otros minerales son parte fundamental del rubro minero. El estudio de éstos es importante debido a sus altos costos y tiempo requerido para estar operativo desde orden de compra, superiores a un año. Se deben tomar en cuenta varios aspectos para aumentar su vida útil y evitar imprevistos que conlleven a pérdidas.

Las pérdidas por la implementación de un plan de mantenimiento correctivo son altas porque los costos por falla implican, además de la reparación del elemento, la paralización de la actividad y un mayor tiempo de indisponibilidad del camión. Uno de los elementos críticos de los camiones son los motores. Estos requieren un seguimiento constante y cuidadores regulares, como, por ejemplo, el cambio de aceite utilizado, como lubricación, cada ciertas horas de uso.

3.1.2 Motor QSK

Los motores QSK son motores Diésel fabricados por Cummins empleados principalmente en camiones en minería. Las siglas QSK corresponden a: [1]

- Q: Sistema de inyección Quantum.
- S: Sistema
- K: Serie de motores - correspondiente a Culatas Individuales.

En este trabajo se considerará como objeto de estudio el sistema de lubricación y enfriamiento mediante aceite de los motores QSK.

El sistema trabaja con lubricación forzada a descansos de bancada, bielas y ejes de levas y otras piezas móviles [1]. Una de las ventajas de este sistema es que se puede cambiar el aceite contaminado por uno limpio, esto permite reducir las concentraciones de diferentes contaminantes, disminuyendo estas concentraciones y así reduciendo la velocidad de desgaste de diversos componentes del motor.

La circulación del aceite comienza en el aspirado de una bomba desde el cárter, impulsándolo al sistema. Luego el aceite pasa por un sistema de filtrado. Una vez filtrado el aceite, éste lubrica diferentes elementos y mecanismos del motor. Finalmente, el aceite retorna al cárter.

Según fuente del dueño de la flota los motores QSK tienen una vida útil de 18000 horas.

3.1.3 El análisis de Aceite

Los análisis de aceite de motor entregan información relevante del estado del motor y del aceite. Estos estudios son relevantes para identificar los elementos del motor que bajan el rendimiento y pueden llegar a fallar provocando pérdidas. Un plan de mantenimiento preventivo debe integrar los estudios anteriores, por ejemplo, haciendo un seguimiento de la calidad y composición del aceite [2].

Una de las causas de este deterioro es el desgaste por el funcionamiento del motor de elementos que por fricción desprenden metales. Los aceites contaminados por estas partículas terminan agravando el estado del motor.

Los efectos de desgaste, cómo la fricción de las paredes de los cilindros, el desgaste del cigüeñal o de los cojinetes, producen el desprendimiento de elementos de las capas de ciertos componentes del motor, cómo bujes o sellos, elevando las concentraciones en el aceite de los elementos que se desprenden. Las muestras de espectrometría del aceite entrega información de las concentraciones de diferentes elementos en partes por millón (ppm). El encargado del mantenimiento debe estar pendiente de las concentraciones de los diferentes elementos y que no pasen ciertos valores críticos en función de cada elemento [2].

En la Tabla 3-1 se presentan los estados del aceite según las concentraciones de los elementos en ppm [3]. Esta información va a permitir realizar diferentes métodos de diagnóstico.

Tabla 3-1 Criterios seguridad por elemento según resultados de análisis de aceite

Elemento	Precaución	Crítico	Unidad
Fierro	25	40	ppm
Cobre	5	15	ppm
Níquel	3	5	ppm
Estaño	3	5	ppm
Aluminio	5	10	ppm
Silicio	7	15	ppm
Plomo	5	10	ppm
Cromo	3	6	ppm
Sodio	15	35	ppm
Potasio	30	40	ppm
Boro	20	25	ppm
Plata	3	5	ppm
Viscosidad alta a 100ºC	12	16	cst
Viscosidad alta a 40ºC	87	130	cst
TBN	5	5	mgKOH/gr
TAN	3,5	4,5	mgKOH/gr
Oxidación	15	20	abs/cm
Nitración	15	20	abs/cm
Nitración	0,15	0,2	0.1 abs/ 0.1 mm
Sulfatación	15	20	abs/cm
Sulfatación	0,15	0,2	0.1 abs/ 0.1 mm
Dilución x FO flash point	189	189	Temp inf.
Dilución x FO	1,5	3	%
Hollín	150	250	abs/cm
Hollín	1,5	2,5	abs/0.1 cm
Hollín	3	7,5	%
Agua	0,2	0,2	%
Indice PQ	30	50	Adimencional

La contaminación del aceite se debe a la presencia de diferentes elementos en éste. La causa de la presencia de estos elementos se puede deber a problemas internas, externos, de fabricación o por acciones del mantenimiento. Los principales contaminantes de aceite son [4]:

- Partículas metálicas, cómo las partículas desprendidas por desgaste de diferentes elementos del motor sometidas a fricción.
- Óxidos metálicos, debido a la oxidación de las partículas metálicas desprendidas.
- Óxido de nitrógeno, debido a la oxidación del nitrógeno atmosférico durante la combustión.
- Polvo atmosférico e impurezas, contaminación externa. La presencia de estos elementos se puede deber a problemas con el filtro de aire, orificios o conductos con fuga entre otros.
- Productos carbonosos, producto del paso de los gases de la combustión, que además facilitan la degradación del aceite, en el cárter.
- Agua, debido a la condensación del vapor producida por la combustión o fugas internas del sistema de refrigeración.
- Combustible, su presencia se puede deber debido a fallas con los inyectores, por la combustión o motor frío.
- Residuos u otros.

Cada uno de estos, además de ser causa de un problema en el motor, son un problema al estar circulando un aceite contaminado en el motor. La circulación produce desgaste abrasivo, rugosidad en superficies, facilitar la degradación del aceite, disminución de la capacidad de carga del aceite debido a cambios en la viscosidad del aceite, espesamiento del aceite debido a la nitración del aceite, entre otros [4].

En este trabajo para caracterizar los subgrupos obtenidos por el proceso de clustering se consulta el libro *Diagnóstico de motores diésel mediante el análisis del aceite usado*, del doctor Bernardo Tormos [4].

3.2 Agrupación de datos en Clústers K-MEANS ++

Un clúster es un subconjunto no vacío de un conjunto de datos que se agruparon, generalmente, por un algoritmo según la similitud o la distancia entre elementos.

3.2.1 k-means y k-means++

El algoritmo de k-means es uno de los principales métodos utilizados en la agrupación de datos. El algoritmo no es complejo y sólo requiere como parámetro la cantidad k de clúster [5].

Las principales características del algoritmo son:

- Tiene buenos resultados con subconjuntos convexos.
- Algoritmo computacionalmente rápido.

La idea de k-means es minimizar la suma del cuadrado de los errores ∅:

$$\emptyset = \sum_{x \in X} \min_{c \in C} \|x - c\|^2 \tag{3.1}$$

El conjunto X representa la base de datos a segmentar y el conjunto C es el conjunto que contiene a los centroides de los Clústers definidos por el algoritmo. En cada iteración se actualizan los valores de los centroides cambiando de esta manera los Clústers.

3.2.2 Definiciones

La base de datos se representa por $X = \{x_1, x_2, \dots, x_n\}$. La distancia se define como $d(x, y) = \|x - y\|^2$. En el caso de querer segmentar la base de datos en k clústers C_i se define el centroide c_i del clúster C_i :

$$c_i = \frac{1}{|C_i|} \sum_{x \in C_i} x \tag{3.2}$$

Los clústers se redefinen en cada iteración de la siguiente forma:

$$C_i = \{\, x \in X \mid d(x, c_i) \leq d(x, c_j\,) \quad \forall j \neq i\} \tag{3.3}$$

Se define la distancia de un elemento $x \in X$ al representante de cluster más cercano $D(x)$ como:

$$D(x) = \min_{c_i} d(x, c_i) \tag{3.4}$$

3.2.3 Algoritmo k-means

Las dos formas de inicializar k-means son seleccionando con probabilidad uniforme los representantes c dentro de R^d o bien con una segmentación inicial de k clústers de X.

El algoritmo sigue los siguientes pasos [5]:

1. Si la inicialización comienza con clústers iniciales pasar al paso 3. Si no, se determinan los k centroides, puntos semillas, con probabilidad uniforme en el espacio en el que se encuentra X. Estos puntos semillas van a permitir la construcción de los k clústers.
2. Definir para cada $i = \{1,2, \dots, k\}$ los clústers C_i, asignando a cada elemento del conjunto X el clúster más cercano C_i.
3. Para cada $i = \{1,2, \dots, k\}$ calcular c_i el centro de masa del clústes C_i.
4. Repetir paso 2 y 3 hasta que los clústers no tengan cambios con respecto a la iteración anterior.

3.2.4 Algoritmo k-means++

La diferencia entre k-means y k-means++ es en el paso 1 del algoritmo. En el caso de k-means++ se selecciona el primer representante con probabilidad uniforme dentro del conjunto X y el resto se calcula con probablidad proporcional a $D(x)^2$, como por ejemplo con la probabilidad: $\frac{D(x)^2}{\sum_{x \in X} D(x)^2}$. Este algoritmo permite distribuir de mejor manera los puntos semillas para realizar la segmentación puesto que asigna mayor probabilidad de ser seleccionados a los puntos más alejados a los clústers [5].

El algoritmo sigue los siguientes pasos:

1. Definir punto semilla c_i con probabilidad uniforme dentro de la base de datos X para los puntos semillas restantes se calculan con probabilidad proporcional a $D(x)^2$.
2. Se realizan los mismos pasos de k-means a partir del paso 2.

3.2.5 Silhouette Score

Para validar los resultados hay diferentes métricas de clúster. El *Silhoutte Score* es una métrica que no requiere las etiquetas verdaderas, trabaja sólo con los resultados del proceso de segmentación. El valor Silhouette entrega información de la relación de un dato etiquetado con su clúster con respecto a los otros [6].

Para el cálculo del valor Silhouette se definen los siguientes parámetros:

- a: distancia promedio intra clúster.
- b: distancia promedio al clúster más cercano

El valor Silhouette para un clúster viene dado por:

$$S = \frac{b-a}{\max\{a,b\}} \tag{3.5}$$

El valor Silhouette está acotado por -1 y 1, si es cercano a -1 se tiene que los resultados de la segmentación no son correctos, cercano a 0 indica que los clústers podrían estar solapados y cercano a 1 es que los clústers están bien separados. Las ventajas de este método es que trabaja bien con conjuntos de diámetros relativamente pequeños y disjuntos. Valores altos indican que los clústers están suficientemente separados entre ellos y distancia intra clúster pequeña con respecto a la distancia inter clúster. Por otra parte, esta métrica es útil para evaluar los resultados al utilizar k-means. Si un algoritmo de clustering presenta malos resultados, se debe buscar alternativas como por ejemplo DBSCAN.

El Silhouette Score entrega el promedio de los valores Silhouette por clúster.

3.3 Clasificador Multi Capas

3.3.1 Descripción

El clasificador multi capas (Multilayer perceptron) es una red neuronal para clasificación [7][8]. Las capas presentes en estas redes neuronales son la capa de entrada, capa de salida y las capas ocultas. La capa de entrada corresponde a las neuronas que reciben la información de entrada consideradas para la clasificación, las capas ocultas realizan diferentes operaciones de la capa entrada y la capa de salida retorna una etiqueta para dicha entrada.

Para fijar ideas, sea el vector $\vec{x} = [x_1, x_2, \dots, x_n]$ la información de entrada, y sean $\overrightarrow{W} = [w_1, w_2, \dots, w_n]$ los pesos asociados a las neuronas de la capa de entrada. La unidad básica de la red neuronal se conoce como perceptrón, consiste en hacer una suma ponderada a de la entrada mediante los pesos de las neuronas de la primera capa, i.e.:

$$a = \sum_{i=1}^{n} w_i . x_i \tag{3.6}$$

Una vez realizada la suma ponderada de la entrada la activación de la neurona quedará determinada por la *función de activación* como por ejemplo ReLU o Softmax. Así lo último que realiza el perceptron es aplicar la función de activación a la suma ponderada. En la Figura 3-1 se puede observar un perceptrón simple, que realiza los pasos recién explicados.

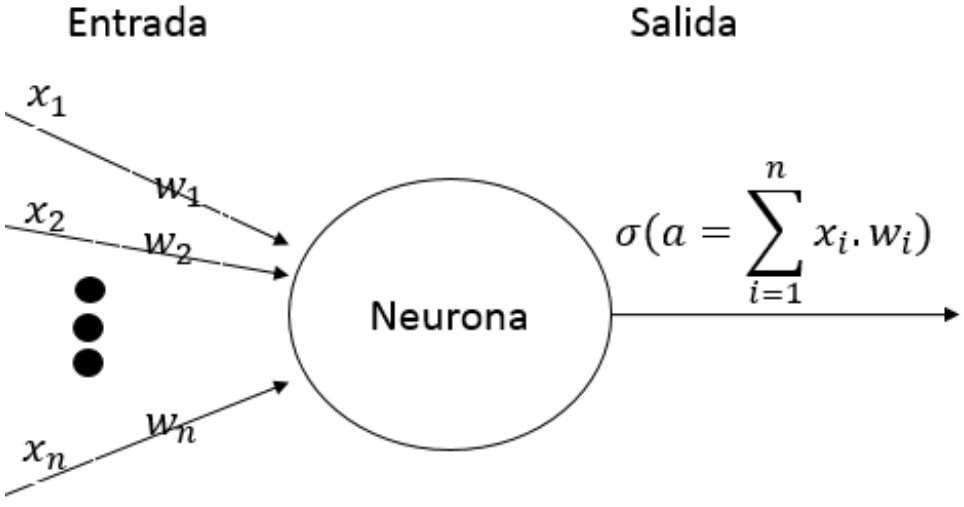

Figura 3-1 Neurona Artificial

Esta arquitectura se puede extender obteniendo así un clasificador multi capas (MLP). La idea es entonces formar redes neuronales a partir de diferentes capas con múltiples

perceptrones. En la Figura 3-2 se presenta un perceptron multicapa(MLP). El funcionamiento del MLP es el mismo funcionamiento de múltiples perceptrones actuando capa a capa.

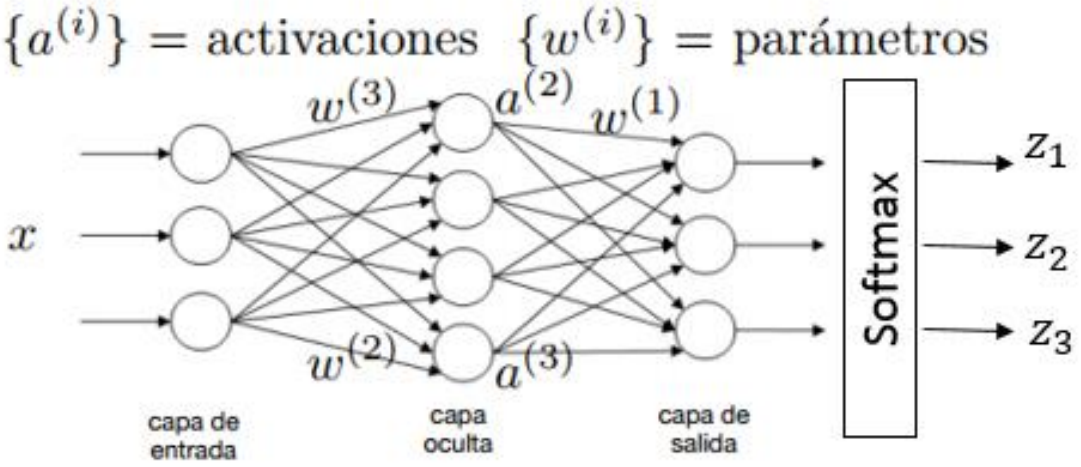

Figura 3-2 Arquitectura MLP con una capa oculta (adaptada [6])

La salida del MLP es la clasificación realizada por la red a la información de entrada.

El entrenamiento de la red consistirá entonces en el ajuste de los pesos y parámetros que definen la red. Durante el entrenamiento de la MLP la información se propaga hacia delante, produciendo una función de costo que llamaremos $J(\theta)$, donde $J(\theta)$ cumple el rol de cuantificar la precisión de la clasificación. La idea del entrenamiento consiste en encontrar los pesos y parámetros que definen la MLP mediante la minimización de la función de costo $J(\theta)$. Usualmente la minimización de $J(\theta)$ implica encontrar el gradiente. Esto se realiza mediante el algoritmo de *backpropagation* que esencialmente consiste en un algoritmo que calcula y evalúa las derivadas parciales que definen el gradiente. Así una vez obtenido el gradiente $\nabla J(\theta)$ el entrenamiento se realiza resolviendo el problema de optimización mediante, por ejemplo, gradient descent method.

Las n de etiquetas deben ser codificadas numéricamente. Para esto, se enumeran las etiquetas y se les asigna a cada una vector de dimensión n. Si se tiene una etiqueta i se le asigna el vector de dimensíon n tal que su componente i sea igual a 1 y el resto iguales a 0.

En resumen, la metodología de entrenamiento es:

- Construir el grafo que define al MLP.
- Separar la base de datos en conjunto de entrenamiento y conjunto de prueba.
- Entrenamiento de la red MLP.
- Análisis de resultados mediante el conjunto de prueba.

3.3.2 Evaluación de Resultados

Para la evaluación del clasificador se ven los resultados obtenidos con el conjunto de prueba. La matriz de confusión es una forma gráfica para visualizar las tasas de acierto sobre el conjunto de prueba.

La matriz de confusión tiene dimensiones $n\ x\ n$, donde n es la cantidad de etiquitas entregadas al clasificador. La componente i, j de la matriz es el porcentaje o cantidad de elementos pertenecientes al clúster i y que fueron etiquetados como j por la MLP. El resultado ideal sería obtener una matriz diagonal.

En la Figura 3-3, se muestra un ejemplo con 3 etiquetas. Se observa que cada fila corresponde a una clase y los valores de las celdas corresponden a la cantidad de elementos de etiqueta i clasificados como j. [9]

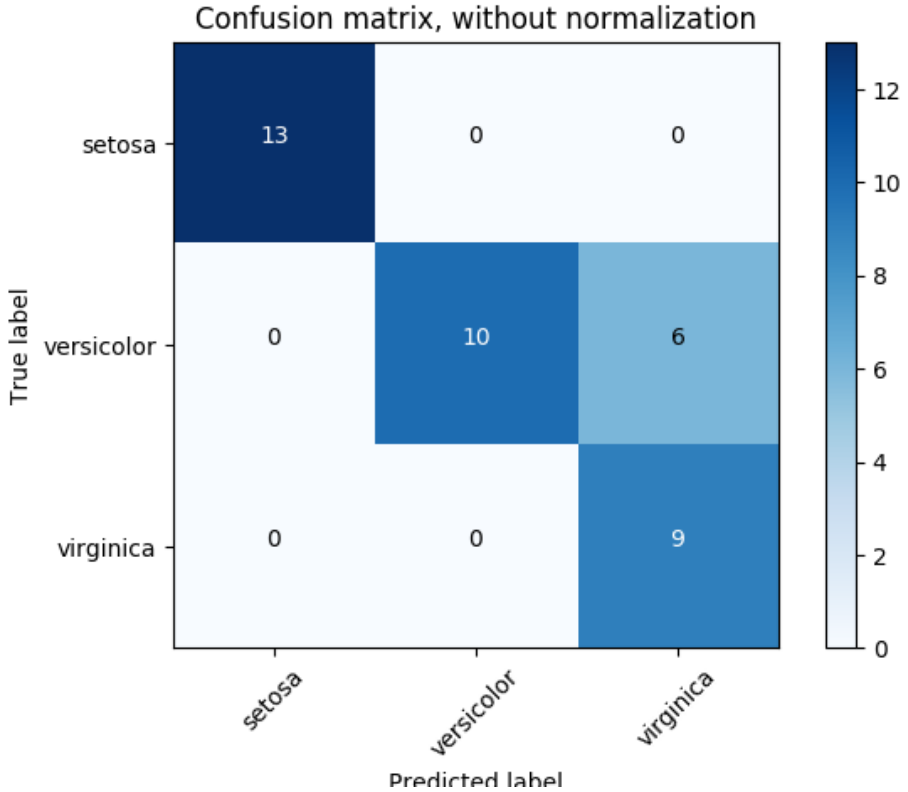

Figura 3-3 Ejemplo Matriz de Confusión

3.4 Confiabilidad por Clúster

3.4.1 Distribuciones sobre tiempos de falla

Para hacer un análisis de confiabilidad de los diferentes clústers obtenidos se asignan distribuciones de probabilidad a cada uno sobre los tiempos de falla. Las distribuciones más utilizadas para este fin son, por ejemplo, las distribuciones Weibull, Lognormal o Normal. Esta etapa permite calcular la confiabilidad $R(t)$ y el MTTF de cada subgrupo.

3.4.2 P-valor

El cálculo del p-valor permite evaluar el ajuste por distribución de probabilidad a un conjunto [10]. Sea X el modelo probabilistico que los tiempos de falla y x una observación dada. El p-valor es una probabilidad condicional definida de la siguiente forma:

$$P(X \leq x|H_0)$$

De esta manera el p-valor indica que tan probable que bajo la hipótesis H_0 el modelo explique los datos mediante la observación x. La hipótesis es aceptada si el p-valor es superior al nivel de significancia deseado. Un p-valor cercano a 1 significa que la hipótesis H_0 tiene alta probabilidad de explicar el modelo.

3.5 Optimización de Plan de Mantenimiento de cada Clúster

Para la optimización del plan de mantenimiento de un grupo homogéneo, cada clúster, se va a trabajar con lo propuesto en el capítulo 2 del libro Maintenance, Replacement, and Reliability Theory and Applications de A. Jardine y A. Tsang [11].

Uno de los casos de interés es el de optimización con reemplazo según edad de equipo sujeto a breakdown considerando los tiempos requeridos para realizar los reemplazos en casos preventivos y de falla.

El modelo en este caso debe considerar:

- El costo total de hacer el reemplazo preventivo C_p.
- El costo total de hacer el reemplazo después de la falla C_f.
- El tiempo medio requerido para hacer el reemplazo preventivoT_p.
- El tiempo medio requerido para hacer el reemplazo después de la falla T_f.
- La función densidad de probabilidad de los tiempos de falla del equipo $f(t)$.
- MTTF cuando el reemplazo preventivo se realiza en el tiempo t_p.

El modelo consiste en realizar un reemplazo una vez que el equipo haya alcanzado un tiempo t_p o reemplazo cuando falle. El objetivo es determinar el t_p óptimo para minimizar el costo esperado total por unidad de tiempo.

Los costos $C(t_p)$ en función de t_p se puede calcular con la siguiente ecuación:

$$C(t_p) = \frac{C_p.R(t_p) + C_f.(1 - R(t_p))}{(t_p+T_p).R(t_p) + (M(t_p) + T_f).(1 - R(t_p))} \quad (3.7)$$

En donde $R(t_p)$ es la confiabilidad del equipo.

El otro caso de interés tiene como objetivo minimizar los tiempos de inactividad (downtime) en función de t_p. Al trabajar con reemplazos debido a un plan preventivo los tiempos de inactividad dependen de los tiempos de reemplazo

Se debe considerar:

- El tiempo medio de inactividad para hacer el reemplazo preventivo T_p.
- El tiempo medio de inactividad para hacer el reemplazo después de la falla T_f.
- La función densidad de probabilidad de los tiempos de falla del equipo $f(t)$.

El tiempo de inactividad $D(t_p)$ es:

$$D(t_p) = \frac{H(t_p)T_f + T_p}{t_p + T_p} \quad (3.8)$$

En donde $H(t_p)$ es el número de fallas en el intervalo $(0\ ;\ t_p)$.

Modelo para determinar el tiempo t_p para minimizar el tiempo de inactividad $D(t_p)$ es:

$$D(t_p) = \frac{T_p.R(t_p) + T_f.(1 - R(t_p))}{(t_p+T_p).R(t_p) + (M(t_p) + T_f).(1 - R(t_p))} \quad (3.9)$$

4 Resultados

4.1 Base de datos

La base de datos estudiada para este trabajo consiste en el estudio 48 motores QSK. La base de datos está subdividida en tres:

- Historial Motores
- Horómetros Camiones
- Muestras Espectrometría

Cada uno de estos grupos tiene 50 archivos y la información que se tiene es hasta enero del 2016. El historial por cada motor consiste en un archivo en formato csv que presenta los camiones en los que ha trabajado el motor y periodos fuera de servicio. También se tiene la información de las horas hasta la falla en cada periodo y en qué estado está hasta la fecha. Los horómetros de los camiones presentan las fechas en las que ha trabajado el camión indicando la cantidad de horas de trabajo. Las muestras de espectrometría presentan las concentraciones en ppm de 20 elementos:

- Hierro
- Cromo
- Cobre
- Estaño
- Plomo
- Aluminio
- Níquel
- Plata
- Titanio
- Potasio
- Vanadio
- Silicio
- Sodio
- Boro
- Zinc
- Calcio
- Magnesio
- Bario
- Fosforo
- Molibdeno

Además, se tiene información sobre concentraciones de hollín, agua, petróleo e información de la oxidación y nitración del aceite. Sin embargo, esta información no se considera en el estudio debido a que los resultados no están expresados en el formato deseado, el dueño realizó una transformación de la cual no se tiene información que no permite integrar esa información en este trabajo.

Para realizar la segmentación de la flota se debe generar la entrada al algoritmo k-means+. Para poder tener resultados correctos se debe realizar un tratamiento de los datos y relacionar la información disponible tomando en cuenta si las concentraciones de los diferentes elementos están en niveles normales, de precaución o críticos.

Para crear la entrada al algoritmo de segmentación se trabaja con python. Un paso previo a la generación de la entrada al algoritmo es integrar y filtrar la información de los motores que está dispersa en varios archivos creando un Excel. Para esto se crea un archivo Excel por cada motor, el código en python, ver Anexo A, realiza los siguientes pasos:

- Crea un archivo Excel por cada motor, cada pestaña corresponde a un camión en el que estuvo un periodo hasta la falla. Por lo tanto, se descartó la información de los motores que estaban en funcionamiento hasta enero del 2016. Se generaron 35 archivos de motores QSK. Las componentes a considerar son: K, Mo, P, Ba, Mg, Ca, Zn, B, Na, Si, V, Ti, Ag, Ni, Al, Pb, Sn, Cu, Cr, Fe, Horas, Fecha Inicial, Fecha Final. Se descartan los otros 8 elementos no considerados debido a que no se tiene información de su estado en función de la concentración en ppm.
- Completas celdas de cada elemento utilizando los archivos de muestras de espectrometría e historial de motores, con éste último se determinan los rangos por camión. Las celdas para fecha inicial y final se completan con el historial.
- Completa columna de Horas utilizando los horómetros de los camiones.

El aceite se cambia aproximadamente cada 500 horas. Tomando en cuenta este dato entregado por la empresa, se genera el archivo de entrada. Éste considera los estados de precaución y críticos acumulados de un motor hasta la falla. Las filas de este archivo corresponden al estado del aceite considerando los cambios de aceite anteriores. De esta manera se tiene la cantidad de estados de precaución, críticos y concentraciones que ha tenido un motor hasta la falla. Debido a que un motor en sus primeras horas de funcionamiento desde la última reparación no presenta, usualmente, niveles críticos, se filtran estos datos. El código, ver Anexo B, sigue los siguientes pasos:

- Se genera el archivo para la entrada al proceso de segmentación y se crean por cada elemento las columnas de niveles de precaución y críticos acumulados en un periodo con un motor. Además, se incluyen columnas con la concentración en ppm, la suma acumulada desde la última reparación, las horas desde la última reparación, el tiempo hasta la falla en el periodo correspondiente y la clase a la que pertenece. Ésta última columna va a permitir filtrar las primeras horas de trabajo debido a que la información de esos intervalos no permite determinar qué tipo de falla va a tener el motor. Cada fila corresponde a un intervalo de 500 horas.
- El valor de la celda para el estado del aceite se rellena comparando en un intervalo de 500 horas las concentraciones con los niveles de precaución y críticos. Si el valor corresponde al estado de precaución se suma 1 a los niveles de precaución y crítico, si está sobre el nivel crítico se suma 5 al nível crítico y por último si está bajo el nivel de precaución se deja igual a los resultados obtenidos en el intervalo anterior. El clasificador obtiene buenos resultados gracias a esta fórmula que se obtuvo en un proceso iterativo.
- Por periodo se calcula el promedio de los niveles de precaución y críticos para cada elemento. Un intervalo es clase 1 si al menos cuatro componentes de los niveles de precaución y críticos están sobre la media. De no ser así son considerados clase 0.

Estos criterios para generar el archivo de entrada fueron variando en función de los resultados del proceso de segmentación. En total hay 511 datos de los cuales 308 son clase 1 y se estudian 32 ciclos de motores hasta la falla.

4.2 Clúster

Para la segmentación de la flota se trabaja en python con la biblioteca scikit-learn. El número de clústers seleccionado fue de 4, ver código Anexo C. La segmentación consideró los datos clase 1. En la Figura 4-1 se presentan los resultados, se ordenaron los datos por etiqueta.

Las componentes consideradas para el proceso de clustering son los tiempos hasta la falla de los ciclos estudiados. El clúster 1 contiene 8 ciclos, el clúster 2 contiene 10 ciclos, el clúster 3 contiene 11 ciclos y el clúster 4 contiene 3 ciclos.

Figura 4-1 Resultados Clúster ordenados por Clase

En la Figura 4-2 se presenta el valor del Silhouette Score para diferentes números de clústers. El valor del Silhouette Score para 3 clústers es el mínimo y luego su valor crece a mayor cantidad de clústers, en todos los casos el valor es positivo. El Silhouette Score para 4 clústers es de 0.647, se trabaja con 4 clústers debido a la poca cantidad de datos. El aumento del valor del Silhouette Score se debe a que al aumentar la cantidad de clústers estos contienen menor número de ciclos disminuyendo así la distancia intraclúster y aumentando la distancia interclúster. Estas distancias dependen de las horas hasta la falla, a menor cantidad mejores resultados del Silhouette Score.

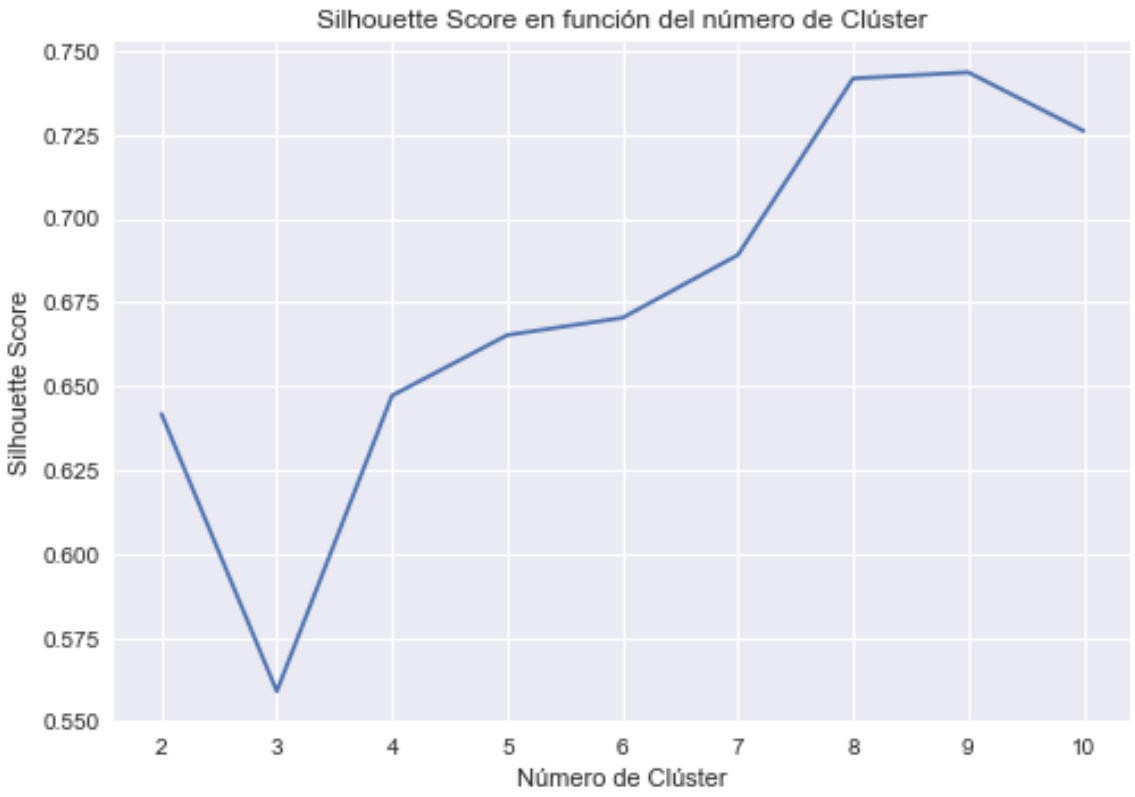

Figura 4-2 Silhouette Score

4.3 Clasificador

La red MLP consiste en una capa de entrada de 24 neuronas, una capa oculta de 14 neuronas y una capa de salida de 4 neuronas, se trabajó con la biblioteca TensorFlow en python. La función de activación utilizada fue ReLU, la función de costo softmax y el método de entrenamiento fue por descent gradient method. Las componentes consideradas para la clasificación fueron los niveles de precauciones y críticos. Los datos se subdividieron en conjunto de datos para entrenamiento (30%) y conjunto de datos para validación (70%). En la Figura 4-3 se presenta la matriz de confusión y en la Tabla 4-1 están los resultados de la evaluación por clase. Para el código ver Anexo D.

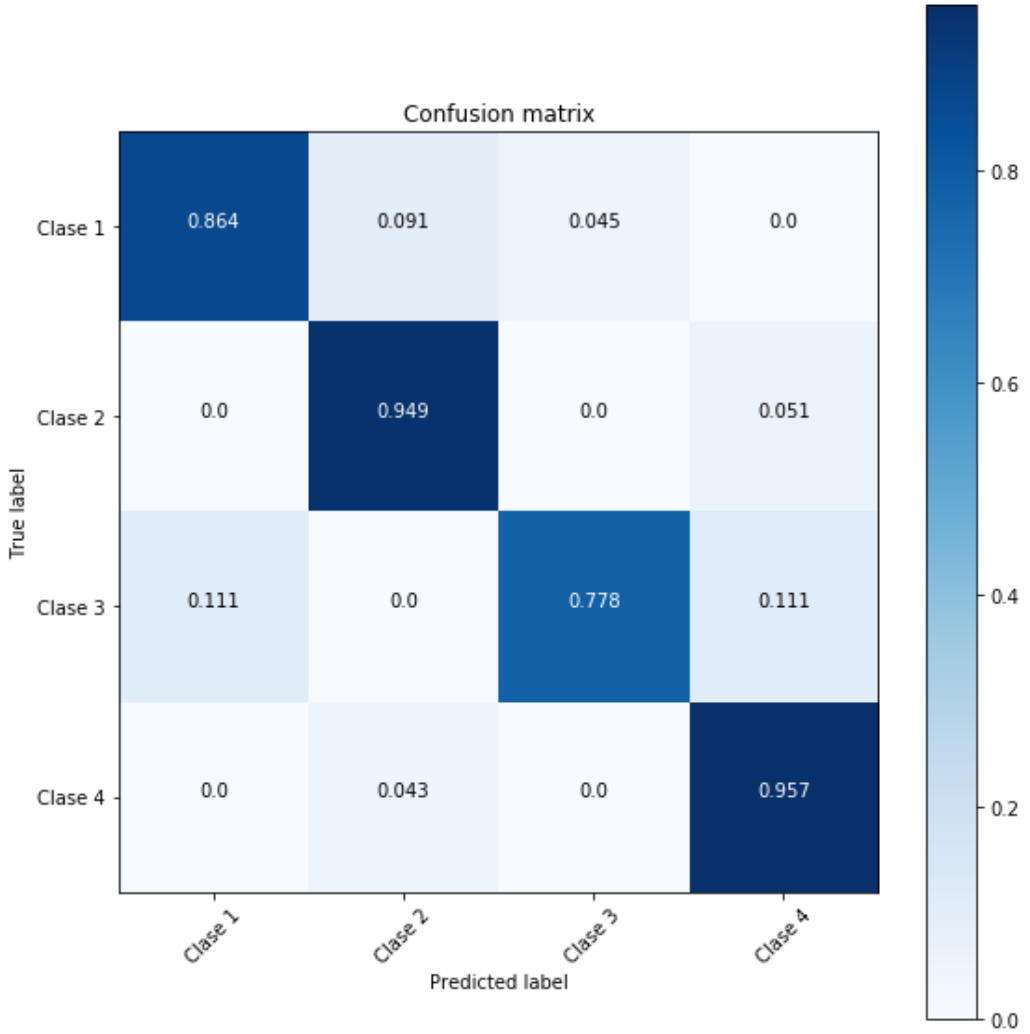

Figura 4-3 Matriz de Confusión

Tabla 4-1 Evaluación Clasificador MLP

Clase	Precisión	Recall	f1-score	Support
1	**0,95**	**0,86**	**0,90**	**22**
2	**0,93**	**0,95**	**0,94**	**39**
3	**0,88**	**0,78**	**0,82**	**9**
4	**0,88**	**0,96**	**0,92**	**23**
Promedio	**0,91**	**0,91**	**0,91**	**93**

4.4 Descripción de Subgrupos

Una vez realizado el proceso de segmentación se calcula la media, desviación estándar y mediana de cada componente para cada clúster. Los resultados se presentan de la Tabla 4-2 hasta la Tabla 4-8.

Tabla 4-2 Características Subgrupo Fe y Cu

			Clúster 1	Clúster 2	Clúster 3	Clúster 4				Clúster 1	Clúster 2	Clúster 3	Clúster 4
Fe	Precaution	mean	5,49	10,34	2,61	8,46	Cu	Precaution	mean	2,32	7,41	1,21	5,17
		std	6,67	6,64	3,97	1,02			std	2,87	6,89	2,16	3,86
		50%	3,00	9,00	1,00	8,00			50%	1,00	4,00	1,00	4,00
	Critical	mean	5,59	10,78	2,67	8,05		Critical	mean	3,42	12,57	4,33	12,15
		std	8,19	8,81	5,36	2,04			std	6,64	13,33	8,75	15,00
		50%	3,00	8,00	0,00	8,00			50%	1,00	9,00	0,00	4,00
	ppm creado	mean	210,74	309,47	134,21	291,06		ppm	mean	41,26	73,64	25,88	51,00
		std	58,44	84,66	59,58	85,61			std	52,71	53,34	23,08	20,47
		50%	210,00	304,00	127,00	306,00			50%	34,00	62,00	19,00	47,00
	Total, ppm	mean	1.285,96	1.854,30	872,76	1.790,95		Total ppm	mean	136,93	224,38	91,85	182,71
		std	414,74	482,56	382,45	567,29			std	209,01	137,69	77,41	98,54
		50%	1.273,00	1.828,00	873,00	1.864,00			50%	100,00	204,00	69,00	147,00

Tabla 4-3 Características Subgrupo Ni y Sn

			Clúster 1	Clúster 2	Clúster 3	Clúster 4				Clúster 1	Clúster 2	Clúster 3	Clúster 4
Ni	Precaution	mean	0,37	0,68	0,09	0,51	Sn	Precaution	mean	0,05	0,00	0,09	0,00
		std	0,49	0,98	0,29	0,62			std	0,47	0,00	0,38	0,00
		50%	0,00	0,00	0,00	0,00			50%	0,00	0,00	0,00	0,00
	Critical	mean	0,95	0,64	0,06	0,46		Critical	mean	2,60	0,00	0,03	0,00
		std	1,91	0,97	0,24	0,59			std	4,50	0,00	0,17	0,00
		50%	0,00	0,00	0,00	0,00			50%	0,00	0,00	0,00	0,00
	ppm	mean	13,79	16,85	11,27	18,80		ppm	mean	3,45	1,86	1,85	1,58
		std	5,14	7,52	8,63	6,97			std	4,03	1,24	2,88	1,94
		50%	13,00	16,00	12,00	18,00			50%	2,00	2,00	0,00	1,00
	Total ppm	mean	17,64	20,58	15,48	21,68		Total ppm	mean	5,99	1,86	2,15	1,86
		std	6,98	9,92	11,19	7,83			std	9,87	1,24	3,50	2,36
		50%	17,00	20,00	16,00	20,00			50%	2,00	2,00	0,00	1,00

Tabla 4-4 Características Subgrupo Al y Si

			Clúster 1	Clúster 2	Clúster 3	Clúster 4				Clúster 1	Clúster 2	Clúster 3	Clúster 4
Al	Precaution	mean	0,48	1,38	0,61	1,08	Si	Precaution	mean	16,66	24,80	16,61	23,34
		std	0,88	1,52	0,83	1,11			std	6,08	9,61	8,75	15,94
		50%	0,00	1,00	0,00	1,00			50%	17,00	24,00	16,00	21,00
	Critical	mean	0,38	1,67	0,45	1,40		Critical	mean	20,19	29,06	14,58	24,54
		std	0,81	2,14	0,75	2,02			std	12,18	14,72	11,06	18,23
		50%	0,00	1,00	0,00	1,00			50%	19,00	28,00	16,00	19,00
	ppm	mean	39,77	55,76	24,70	57,12		ppm	mean	123,01	150,64	78,58	157,71
		std	12,73	17,58	11,85	26,04			std	44,62	50,71	41,19	74,41
		50%	39,00	54,00	26,00	54,00			50%	115,00	144,00	83,00	152,00
	Total ppm	mean	139,71	198,07	87,79	192,55		Total ppm	mean	533,73	717,45	383,76	739,85
		std	42,30	60,40	37,78	83,77			std	151,84	214,81	160,77	324,83
		50%	136,00	196,00	85,00	189,00			50%	532,00	701,00	405,00	738,00

Tabla 4-5 Características Subgrupo Pb y Cr

			Clúster 1	Clúster 2	Clúster 3	Clúster 4				Clúster 1	Clúster 2	Clúster 3	Clúster 4
Pb	Precaution	mean	26,74	39,91	23,15	49,37	Cr	Precaution	mean	0,00	0,37	0,00	0,17
		std	15,43	25,88	16,20	32,36			std	0,00	0,99	0,00	0,38
		50%	24,00	35,00	25,00	39,00			50%	0,00	0,00	0,00	0,00
	Critical	mean	32,79	67,34	41,33	73,29		Critical	mean	0,00	0,35	0,00	0,15
		std	35,12	69,29	44,09	69,72			std	0,00	0,97	0,00	0,36
		50%	24,00	37,00	35,00	52,00			50%	0,00	0,00	0,00	0,00
	ppm	count	65,71	97,05	52,36	97,54		ppm	mean	18,64	24,66	13,00	26,78
		mean	25,58	42,50	31,68	53,41			std	7,55	7,65	6,95	12,92
		std	63,00	89,00	49,00	80,00			50%	17,00	24,00	14,00	26,00
	Total ppm	count	417,70	605,14	333,61	648,48		Total ppm	mean	82,85	110,31	49,73	119,89
		mean	174,98	262,10	181,53	344,26			std	25,48	30,61	23,64	51,17
		std	395,00	545,00	347,00	560,00			50%	82,00	109,00	45,00	118,00

Tabla 4-6 Características Subgrupo Na y K

			Clúster 1	Clúster 2	Clúster 3	Clúster 4				Clúster 1	Clúster 2	Clúster 3	Clúster 4
Na	Precaution	mean	1,00	4,42	1,45	6,35	K	Precaution	mean	0,00	0,12	0,00	0,00
		std	1,99	5,75	1,94	6,64			std	0,00	0,33	0,00	0,00
		50%	0,00	2,00	1,00	1,00			50%	0,00	0,00	0,00	0,00
	Critical	mean	0,81	29,26	1,21	13,09		Critical	mean	6,64	6,18	11,06	9,23
		std	1,83	36,58	1,87	19,07			std	11,15	10,88	14,18	13,95
		50%	0,00	5,00	0,00	1,00			50%	0,00	0,00	0,00	0,00
	ppm	mean	63,11	222,62	50,48	116,09		ppm	mean	5.674,88	3.942,57	5.844,39	4.064,65
		std	23,63	190,03	25,34	60,24			std	8.494,03	8.828,47	7.331,67	5.990,35
		50%	61,00	134,00	53,00	113,00			50%	99,00	134,76	90,00	133,00
	Total ppm	mean	345,40	976,72	271,15	636,38		Total ppm	mean	7.494,01	6.614,78	8.903,24	7.588,74
		std	132,15	769,46	139,15	386,74			std	10.764,22	12.610,30	10.994,27	10.331,87
		50%	351,00	581,00	289,00	621,00			50%	665,00	875,00	544,00	999,00

Tabla 4-7 Características Subgrupo B y Ag

			Clúster 1	Clúster 2	Clúster 3	Clúster 4				Clúster 1	Clúster 2	Clúster 3	Clúster 4
B	Precaution	mean	0,00	0,00	0,00	0,00	Ag	Precaution	mean	0,00	0,12	0,00	0,00
		std	0,00	0,00	0,00	0,00			std	0,00	0,33	0,00	0,00
		50%	0,00	0,00	0,00	0,00			50%	0,00	0,00	0,00	0,00
	Critical	mean	0,00	0,00	0,00	0,00		Critical	mean	0,00	0,12	0,00	0,00
		std	0,00	0,00	0,00	0,00			std	0,00	0,32	0,00	0,00
		50%	0,00	0,00	0,00	0,00			50%	0,00	0,00	0,00	0,00
	ppm	mean	11,19	19,04	5,88	17,06		ppm	mean	0,58	1,20	0,85	1,37
		std	7,55	10,28	4,79	6,78			std	0,60	1,94	1,42	1,23
		50%	9,00	16,00	8,00	18,00			50%	1,00	0,00	0,00	1,00
	Total ppm	mean	17,32	44,55	12,97	31,06		Total ppm	mean	0,58	1,45	0,85	1,37
		std	10,24	32,84	10,52	17,24			std	0,60	2,57	1,42	1,23
		50%	15,00	28,00	18,00	26,00			50%	1,00	0,00	0,00	1,00

Tabla 4-8 Características Subgrupo Horas

		Clúster 1	Clúster 2	Clúster 3	Clúster 4
Horas	mean	16717,73973	23358,32117	8790,151515	28513,24615
	std	2055,036307	1513,743475	2983,084039	441,7101719
	50%	17350	23869	9886	28401

En las Tabla 4-9, Tabla 4-10 y Tabla 4-11 se presentan 3 alternativas de distribuciones para cada clúster. La distribución se determinó considerando los valores del p-valor. Para calcular el p-valor se trabaja con R.

Tabla 4-9 Ajuste distribución Normal

Normal				
	Clúster 1	**Clúster 2**	**Clúster 3**	**Clúster 4**
mu	16718,75	23358,32	8790,15	28513,24
sigma	2055,03	1514,75	2983,08	441,71
p-valor	$7{,}5 \times 10^{-5}$	0,93	0,83	0,95

Tabla 4-10 Ajuste distribución Lognormal

Lognormal				
	Clúster 1	**Clúster 2**	**Clúster 3**	**Clúster 4**
mu	9,68	10,04	8,28	10,25
sigma	0,12	0,07	1,65	0,01
p-valor	0,94	0,92	0,17	0,95

Tabla 4-11 Ajuste distribución Weibull

Weibull				
	Clúster 1	**Clúster 2**	**Clúster 3**	**Clúster 4**
alpha	17046,10	23923,13	8242,07	28786,99
beta	8,54	15,25	1,81	47,14
p-valor	0,94	0,93	0,40	0,91

En la Tabla 4-12 se presenta los resultados del MTTF por clúster según la distribución.

Tabla 4-12 MTTF por Clúster

MTTF [hrs]				
	Clúster 1	**Clúster 2**	**Clúster 3**	**Clúster 4**
Weibull	16103	23113	7326	28447
Normal	16117	23358	8790	28153
Lognormal	16166	23169	15560	28512

Para cada clúster se seleccionó el ajuste con mayor p-valor, de esta manera se ajusta con una distribución Lognormal al clúster 1 y con distribución Normal a los clústers 2, 3 y 4. En la Figura 4-4 Distribuciones de Probabilidad se presentan las distribuciones propuestas a cada clúster.

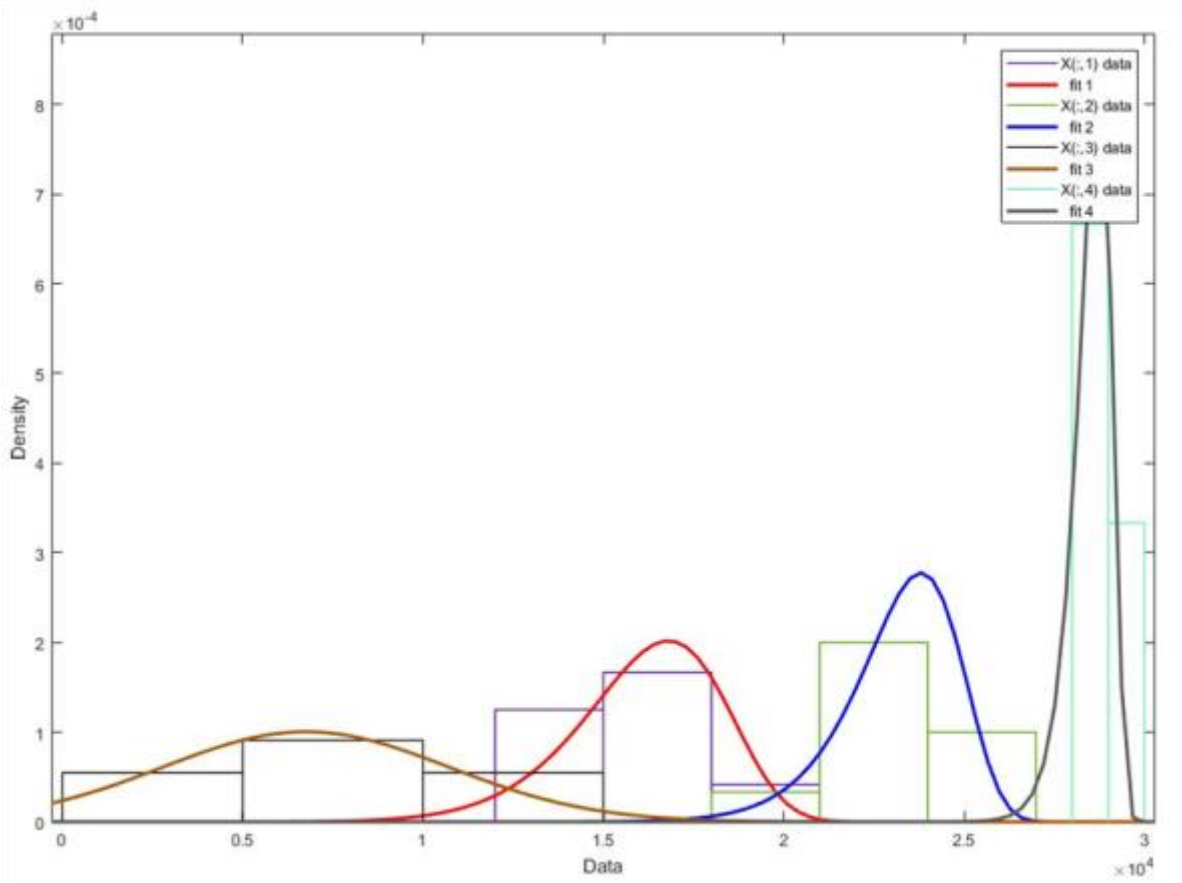

Figura 4-4 Distribuciones de Probabilidad

4.5 Plan de Mantenimiento

Para el plan de mantenimiento se trabaja con la ecuación (3.7) para minimizar costos. Se considera el costo de reparación una vez fallado el motor C_f como 3 veces mayor al costo de reparación sin falla C_p obteniéndose así la siguiente función objetivo:

$$f_{objetivo} = \frac{C(t_p)}{C_p} = \frac{R(t_p) + 3.(1 - R(t_p))}{(t_p + T_p).R(t_p) + (M(t_p) + T_f).(1 - R(t_p))} \quad (4.1)$$

Se considera que los clústers 2, 3 y 4 siguen distribución normal y el clúster 1 sigue distribución Lognormal. Las funciones de confiabilidad $R(T)$ se calcula de la siguiente forma:

Distribución Normal

$$R(t) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(t-\mu)^2}{2\sigma}} \quad (4.2)$$

Distribución Lognormal

$$R(t) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(\ln(t)-\mu)^2}{2\sigma}} \quad (4.3)$$

Se calcula la confiabilidad de cada clúster para 18000 horas y para 16271 horas qué es el promedio de las horas hasta la falla de los ciclos estudiados. En Tabla 4-13 se presentan los resultados.

Tabla 4-13 Resultados Confiabilidad a 18000 y 16721 horas

Clúster	1	2	3	4
R(18000)	0,4205	0,9992	0,0023	1,0000
R(16721)	0,4903	1,0000	0,0083	1,0000

Para el cálculo de los tiempos óptimos se trabaja con el solver de Excel. Los resultados se presentan en la Tabla 4-14.

Tabla 4-14 Resultados tiempo óptimo por Clúster

Clúster	1	2	3	4
Tiempo Óptimo t_p [hrs]	14149	5834	26983	19755

5 Análisis de Resultados

Para obtener un plan de mantenimiento preventivo en base a técnicas de clustering y aprendizaje supervisado la etapa de tratamiento de datos resulta ser la más compleja. Hay varias formas y métodos para generar el archivo de entrada a los algoritmos de segmentación y de clasificación, para obtener resultados correctos estas entradas deben poder diferenciar los datos entre ellos. Se decide tratar la flota de camiones en torno a los motores debido a que no se tiene información de las condiciones de operación de los camiones y dar como componente el camión en que se trabaja no diferencia los datos. En cambio, resulta más simple centrarse en los motores al utilizar la información del estado del aceite y trabajar con intervalos de 500 horas, correspondientes aproximadamente a un cambio de aceite según fuente de la empresa.

Las variables consideradas para el proceso de clustering se determinaron en un proceso iterativo. Se probaron diferentes entradas al proceso de clustering y también se consideraron diferentes algoritmos de clustering. En la sección de resultados se presenta el método que entregó resultados interpretables y correctos. Los datos clase 0, datos descartados, corresponden a las primeras horas de trabajo de un motor. Se descartan debido a que los niveles de precaución y críticos son bajos, excepto en algunos casos donde el deterioro del motor es rápido en pocas horas, y no se relacionan con un mecanismo de falla. El criterio de ponderar por 5 los casos críticos es para distanciar más los datos que presentan altos valores de concentración en ciertos intervalos, de esta manera se diferencian más los estados sanos en ciertos componentes. La selección del valor 5 se debe a que con ese valor se obtuvo un buen resultado en el proceso de clasificación. Esta etapa entrega un archivo de entrada con 308 datos clase 1. De los 308 datos hay varios datos que corresponden al mismo periodo, los periodos analizados son 32.

Se observa en la Figura 4-1 que los clústers están bien separados con respecto a las horas hasta la falla, los clústers 1, 2, 3 y 4 tienen respectivamente 8, 10, 11 y 3 periodos. El valor del Silhouette Score está sobre 0,5 lo que es bueno. Al contrastar ese valor, ver Figura 4-2, con el obtenido para otras configuraciones se observa que en 3 es el mínimo y a partir de 4 el valor crece. Se decide trabajar con 4 clúster debido a la poca cantidad de datos lo que implica que hayan clústers con muy pocos periodos y las distancia intraclúster serían cada vez más pequeñas debido a esto último. Además, esos 4 clúster sí se pueden diferenciar. La configuración propuesta permite realizar un diagnóstico en torno a 4 tipos de fallas diferentes. Se debe considerar que el proceso de clustering es no supervisado, por lo que hay menos métricas para la evaluación del proceso.

Se observa a partir de la Tabla 4-2 hasta la Tabla 4-8 que en general el hierro es un elemento con concentraciones elevadas en el aceite. El clúster 1, con respecto a los otros subgrupos, tiene bajas concentraciones de los otros elementos, medianas bajas y su vida es de aproximadamente 16 mil horas por lo que se puede deducir que su falla se debe a elementos del motor tales como paredes de cilindro, desgaste de camisas, desgaste de válvulas o tren de engranaje. El clúster 2 presenta concentraciones elevadas de silicio y sodio. Esto nos indica que hubo filtraciones de agua por el sodio y entrada de polvo debido al silicio. El hierro sigue siendo un factor relevante y presenta concentraciones altas, de todas maneras, hay que considerar que los motores del clúster 2 duran más que los del clúster 1 y 3. El clúster

3 es el caso más destructivo, el promedio de horas hasta la falla es de 8790 horas y las concentraciones de hierro y potasio son comparables a las concentraciones obtenidas en el resto de los motores que trabajaron más horas. Se deduce que el deterioro es rápido y el diagnóstico para este clúster es una falla en el sistema de enfriamiento con filtraciones del líquido de enfriamiento que contiene potasio. Además, este subgrupo es el que tiene los casos más dispersos en cuanto horas hasta la falla su desviación estándar es de aproximadamente 3000 horas. El clúster 4 tiene concentraciones elevadas de varios elementos, sin embargo, los periodos considerados en este clúster son los que duran más, 28 mil horas en promedio, por lo que no resulta sorprendente que estos niveles sean altos. Las concentraciones de sodio son menores que las del clúster 2, se descarta filtraciones de agua. Se puede deducir que la falla de estos motores es por edad y es el clúster más pequeño, tres casos.

El clasificador MLP obtiene buenos resultados a pesar de trabajar con pocos datos. Este clasificador no toma en cuenta las horas hasta la falla, por lo que resulta interesante que pueda clasificar los motores a un clúster tomando en cuenta la cantidad de estados de precaución y críticos desde la última reparación. La matriz de confusión, Figura 4-3, se aproxima a una matriz diagonal, los valores en las celdas están sobre el 90% a excepción del clúster 3. Éste último es el que presenta mayor problema en el momento de clasificar, sin embargo, debido a lo comentado en el párrafo anterior, es un caso crítico debido a las altas concentraciones de diferentes elementos con menos de 10 mil horas de uso. En la Tabla 4-1 se presentan en formato tabla los resultados de la MLP, todos los parámetros están sobre el 90% en promedio.

Se ajustaron 3 distribuciones para cada clúster, se eligieron los ajustes con mayor p-valor. Se puede observar que los p-valores son elevados por lo que las hipótesis son buenas, considerando que generalmente se rechaza la hipótesis con resultados inferiores a 0,1. A partir de las Tabla 4-9, Tabla 4-10 y Tabla 4-11 se determina que los clústers 2, 3 y 4 se ajustan con una distribución Normal y el clúster 1 con una distribución Lognormal. Esto permite calcular el MTTF, ver Tabla 4-12, de cada subgrupo y además gracias a esto se tiene un modelo de confiabilidad para cada clúster.

La confiabilidad de cada clúster se calcula para el tiempo esperado según fuente del dueño de 18000 horas y el tiempo promedio de fallas de los 32 casos estudios de 16271 horas. El clúster 1 y clúster 3 tiene confiabilidad de que los motores duren 18000 horas de 42 y 0,02% respectivamente y en el caso de que duren más de 16271 horas su confiabilidad es de 49 y 0,08% respectivamente. Estos dos clústers contienen aproximadamente el 60% de los periodos estudiados. En el caso de los clústers 2 y 3 la confiabilidad de que duren más de 18000 horas y 16271 horas son altas, no se espera que fallen antes. La diferencia de confiabilidad entre clústers es relevante por lo que se justifica realizar un plan de mantenimiento que no considere la misma confiabilidad para toda la flota. Son muchos los casos en que la falla se dio antes de lo esperado, el encargado de mantenimiento debe integrar estos factores.

Para utilizar el modelo propuesto en la sección 2.5 Optimización de Plan de Mantenimiento de cada Clúster se debe tener información de los costos tanto de mantenimiento preventivo como del mantenimiento correctivo además de tener definidos los tiempos de indisponibilidad en los dos tipos de mantenimiento. Como se comenta en la sección de resultados, se trabaja con el supuesto de que el costo por mantenimiento correctivo

es 3 veces mayor al del mantenimiento preventivo. Con esto se obtiene una nueva función objetivo a minimizar para encontrar los tiempos óptimos t_p para realizar mantenimiento. En todos los casos el tiempo óptimo t_p es menor al MTTF del clúster, se prefiere realizar un plan de mantenimiento preventivo utilizando esta información.

En el caso del clúster 3, debido a la dispersión de los periodos considerados en éste, no se recomienda esperar hasta ese tiempo óptimo. El motor está dañado gravemente y se propone enviar a mantenimiento.

Por lo tanto, se logra realizar diagnósticos para cada clúster que ayuda en la toma de decisión del tiempo óptimo para enviar el motor a mantención. Sin embargo, hay que considerar que falta información relevante para los mecanismos de falla, tales como porcentajes de agua, nitración o incluso la viscosidad del aceite. El trabajo con el dueño de la flota para la integración de esta información en el modelo es necesario para un buen plan de mantenimiento acorde a las condiciones de operación.

6 Conclusiones

Resulta interesante la metodología propuesta en el presente trabajo. El empleo de técnicas de aprendizaje supervisado y clustering logran competir e incluso reemplazar la opinión del experto para los mantenimientos al considerar las condiciones de operación de los equipos/máquinas.

Con la información disponible es posible proponer un plan de mantenimiento preventivo. Se logra obtener una correcta segmentación de la flota heterogénea en estudio. El algoritmo de clustering identifica diferentes subgrupos con características comunes asociándoles las posibles causas de falla. Por otra parte, el clasificador permite clasificar los motores que estén en funcionamiento, esto es una ventaja para el dueño de la flota debido a que podrá realizar un mejor diagnóstico de su flota y evitar pérdidas por fallas imprevistas. El aporte de la combinación de ambas técnicas es relevante para una buena administración de la flota.

La cantidad de estados de precaución y críticos de diferentes elementos, son determinados como parámetros que sí son relevantes para explicar la diferencia entre motores de la flota en un periodo. Es con esa información con la que se caracterizan los subgrupos para luego así realizar un diagnóstico por subpoblaciones.

Las cuatro subpoblaciones determinadas en el trabajo fallan por diferentes mecanismos. El más catastrófico es por la falla del sistema de enfriamiento, clúster 3. Las otras fallas son por horas de uso en el caso del clúster 4, filtraciones de agua en el caso del clúster 2 y por fallas de elementos del motor en el caso del clúster 1. Los tiempos óptimos en los cuatro casos son inferiores al MTTF por lo que se propone en los cuatro casos mantenimiento preventivo.

Estudiando los clústers 1 y 2 se puede concluir que hay varias fallas que ocurren antes de lo esperado considerando el tiempo esperado de falla del motor según el dueño de la flota o incluso del promedio de las horas de fallas de los casos estudiados en el presente trabajo. Por esto mismo es importante segmentar la flota heterogénea en subpoblaciones homogéneas, permitiendo así un plan de mantenimiento sujeto a las condiciones operación y características de cada subgrupo. Además, la confiabilidad por cada clúster se acerca más a lo esperado de cada clúster, hay que considerar que si se calcula la confiabilidad de la flota en su totalidad se tendría que un gran porcentaje de la flota fallaría antes de lo esperado. Esta segmentación permite evitar pérdidas por la indisponibilidad de camiones traducidas, por ejemplo, en costos por ingresos no percibidos o costos mayores por reparar motores con falla.

Para mejorar el plan de mantenimiento es fundamental realizar un trabajo con la empresa dueña de la flota con el fin de integrar otros factores relevantes, como por ejemplo, porcentajes de agua, hollín, petróleo, tiempos de indisponibilidad, costos entre otros elementos. La metodología utilizada en el presente trabajo permite la incorporación de más información esperando así un buen análisis de la confiabilidad de la flota y una correcta administración.

7 Bibliografía

[1] ANGLO AMERICAN, Curso de Mantención de Camiones Komatsu 930E – 830E, p*áginas 6 a 9*.

[2] Spectros Scientific *(2014) https://www.spectrosci.com/default/assets/File/SpectroSci_OilAnalysisHandbook_FINAL_2014-08.pdf*

[3] Dueño flota camiones, *tabla criterios seguridad,* entregada junto a la base de datos.

[4] Bernardo Tormos, *Diagnóstico de motores diésel mediante el análisis del aceite usado*, Cápitulos: 5 y 6.

[5] Arthur, D.; Vassilvitskii, S (2007) *k-means++: The Advantages of Careful Seeding*. *http://ilpubs.stanford.edu:8090/778/1/2006-13.pdf*

[6] scikit-learn, Silhouette Score,

http://scikit-learn.org/stable/modules/generated/sklearn.metrics.silhouette_score.html#sklearn.metrics.silhouette_score

[7] Jean-Pierre Lévy Mangy (2008), *Las Redes Neuronales Artificiales Fundamentos Teóricos y aplicaciones prácticas*. *Capítulo* 4, *https://books.google.cl/books?id=X0uLwi1Ap4QC&pg=PA60&lpg=PA60&dq=perceptron+multicapa+fcfm&source=bl&ots=gNHAjolr_k&sig=hmyQ9DkooXreyuoxtA5ZecqJUF8&hl=es&sa=X&redir_esc=y#v=onepage&q=perceptron%20multicapa%20fcfm&f=false*

[8] Ian Goodfellow and Yoshua Bengio and Aaron Courville(2016), *Deep Learning textbook , http://www.deeplearningbook.org* ,Capítulo: 6- Deep Feedforward Networks

[9] scikit-learn, Ejemplo Matriz de Confusión, *http://scikit-learn.org/stable/auto_examples/model_selection/plot_confusion_matrix.html#sphx-glr-auto-examples-model-selection-plot-confusion-matrix-py*

[10] Mark J. Schervish(1996), *P Values: What They Are and What They Are Not*., *http://www.jstor.org/stable/2684655?seq=1#page_scan_tab_contents*

[11] Andrew K.S Jardine, Albert H.C Tsang, *Maintenance, Replacement, and Realiability Theory and Applications (Second Edition)* Capítulos : 2.6 y 2.7.

ANEXOS

Anexo A Código Primera Etapa Tratamiento de Datos

```
# -*- coding: utf-8 -*-
"""
This program will work exactly like data_merging_v2.0, but it will use
a different file to read the trucks' historial to avoid the error that exist in
the original files.
@author: Sébastien ALONSO
Versión MAC Gastón BUTIKOFER
"""

    # %% Libraries importation

    import os
    import pandas as pd
    import numpy as np
    import openpyxl
    import xlrd
    from datetime import datetime


    # %% Path and useful variables definitions

    merging_path = r'/Users/GBL/Desktop/datamot/Mergedd Dataa'

    #historial_motores_path = r'D:\Documents\Stage\Travail\Data Haul Trucks ' + \
    #r'Motors\v2.0\Historial Motores'

    historial_camiones_path = r'/Users/GBL/Desktop/datamot/Sintesis de los motores instalados'

    espectrometria_path = r'/Users/GBL/Desktop/datamot/Muestras espectrometria'


    horometro_path = r'/Users/GBL/Desktop/datamot/Horometro camiones sin errores'

    col_name = ['K', 'Mo', 'P', 'Ba', 'Mg', 'Ca', 'Zn', 'B', 'Na', 'Si',
                'V', 'Ti', 'Ag', 'Ni', 'Al', 'Pb', 'Sn', 'Cu', 'Cr', 'Fe', 'Hours',
                'Initial Date', 'Final Date']

    normalized_path = r'D:\Documents\Stage\Travail\Data Haul Trucks ' + \
    r'Motors\v2.0\Normalized data'


    # %% Functions definition
```

```
43.
44.      #============================================================
   ==================
45.      # Now the create excel function only create an excel if the
   motor used by the
46.      # truck failed and adds sheet tothis excel only if the motor
   has failed. Thus
47.      # it will be useless to filter the excels because this
   function does it
48.      # implicitely.
49.      #============================================================
   ==================
50.
51.      def create_excel(writting_path):
52.          "This function create one excel per truck with one sheet
   per motor used \
53.          by the truck in the file given in as argument"
54.
55.          file_list = os.listdir(espectrometria_path)
56.          for file in file_list:
57.              excel = openpyxl.Workbook()
58.              truck_hist = pd.read_excel(historial_camiones_path +
   '\\' + file + '.xlsx')
59.              motor_list = truck_hist['Motor']
60.              if len(motor_list) >= 2:
61.                  if len(motor_list) % 2 == 0:
62.                      motor_list = np.unique(motor_list)
63.                      for motor in motor_list:
64.                          excel.create_sheet(motor)
65.                      excel.remove_sheet(excel.get_sheet_by_name('S
   heet'))
66.                      excel.save(writting_path + '\\' + file +
   '_spectro.xlsx')
67.                  else:
68.                      for i in range(len(motor_list)-1):
69.                          if motor_list[i] == motor_list[i+1]:
70.                              excel.create_sheet(motor_list[i])
71.                      excel.remove_sheet(excel.get_sheet_by_name('S
   heet'))
72.                      excel.save(writting_path + '\\' + file +
   '_spectro.xlsx')
73.          print('Excels created')
74.
75.
76.      def filling_excels():
77.          "This function will fill the excel with the relevant
   datas."
78.
79.          truck_list = os.listdir(merging_path)
80.          al1 = openpyxl.styles.Alignment(horizontal='center',
   vertical='center')
81.
82.          for truck in truck_list:
83.              excel = openpyxl.load_workbook(merging_path + '\\' +
   truck)
84.              sheet_list = excel.get_sheet_names()
```

```
85.             spec_path = espectrometria_path + '\\' + truck[0:6] +
   '\\' + truck[0:6] + \
86.             'espectrometria.xlsx'
87.             spectro = openpyxl.load_workbook(spec_path)
88.             component = spectro.get_sheet_names()
89.             del component[0]
90.     #=============================================================
   ==================
91.     # The previous line delete the first value of the component
   list because the
92.     # first sheet of the espectrometria file do not correspond to
   an oil component
93.     #=============================================================
   ==================
94.             for sheet in sheet_list:
95.                 wrk_sheet = excel[sheet]
96.                 wrk_sheet.merge_cells('A1:T1')
97.                 wrk_sheet['A1'] = 'Components'
98.                 wrk_sheet['A1'].alignment = al1
99.                 wrk_sheet.merge_cells('U1:W1')
100.                wrk_sheet['U1'] = 'Time'
101.                wrk_sheet['U1'].alignment = al1
102.                for i in range(len(col_name)):
103.                    wrk_sheet.cell(row=2, column=i + 1).value =
   col_name[i]
104.                    wrk_sheet.cell(row=2, column=i + 1).alignment
   = al1
105.    #=============================================================
   ==================
106.    # The previous section define the column titles, now we fill
   the excel.
107.    # To do this, we need the dates of installation and failure
   of the motor, we will
108.    # read those dates in the excel that we created to avoid the
   inherent mistakes
109.    # of the original files.
110.    #=============================================================
   ==================
111.                truck_hist_name = historial_camiones_path + '\\'
   + truck[0:6] + '.xlsx'
112.                truck_hist = pd.read_excel(truck_hist_name)
113.                motor_list = truck_hist['Motor']
114.                date_list = truck_hist['Fecha']
115.                dates = []
116.                for i in range(len(motor_list)):
117.                    if motor_list[i] == sheet:
118.                        dates.append(datetime.strptime(date_list[
   i], '%d-%m-%Y'))
119.                date_init = min(dates)
120.                date_end = max(dates)
121.    #=============================================================
   ==================
122.    # Now we have the limit dates, so we will collect the data
   between them.
123.    #=============================================================
   ==================
124.                test_result = []
```

```
125.                 date_instance = type(date_init)
126.                 for cp in component:
127.                     ws_spec = spectro[cp]
128.                     a = []
129.                     for cell in ws_spec['A']:
130.                         if isinstance(cell.value, date_instance)
   and \
131.                         date_init <= cell.value <= date_end:
132.                             a.append(ws_spec.cell(row=cell.row,
   column=2).value)
133.                     test_result.append(a)
134.                     del a
135.     #=============================================================
   ==================
136.     #  This part of the function calculate and collect the
   motor's worked hours:
137.     #=============================================================
   ==================
138.                 # Here we create a list with all the spectrometry
   dates
139.                 test_date = []
140.                 ws_spec2 = spectro[component[1]]
141.                 for cell in ws_spec2['A']:
142.                     if isinstance(cell.value, date_instance) and
   date_init <=\
143.                     cell.value <= date_end:
144.                         test_date.append(cell.value)
145.                 # Here the program read the horometer motor file
146.                 name_h = truck[0:5] + '-HOR.csv'
147.                 hor_path = horometro_path + '\\' + name_h
148.                 horometro = pd.read_csv(hor_path,
   engine='python', sep='","')
149.                 date = np.array(horometro['Entry Date'].dropna())
150.                 h = np.array(horometro['Statistic
   Value'].dropna())
151.                 h_res = []
152.     #=============================================================
   ==================
153.     # Here we sum all the worked hours between two test dates and
154.     # add the result to a list
155.     #=============================================================
   ==================
156.                 for i in range(len(test_date) - 1):
157.                     aux = 0
158.                     for j in range(len(date)):
159.                         if test_date[i+1] <=
   datetime.strptime(date[j], '%d-%m-%Y')\
160.                         < test_date[i]:
161.                             aux = aux + float(h[j].replace(',',
   '.'))
162.                     h_res.append(aux)
163.
164.                 aux = 0
165.                 for i in range(len(date)):
166.                     if date_end <= datetime.strptime(date[i],
   '%d-%m-%Y') <=\
167.                     test_date[-1]:
```

```
                        aux = aux + float(h[i].replace(',', '.'))

                h_res.append(aux)
                h_res.reverse()
                for i in range(1, len(h_res)):
                    h_res[i] = int(h_res[i] + h_res[i-1])
                h_res.reverse()


                # We add the cumulative hour list to the list containing the test
                # values
                test_result.append(h_res)
                test_result.append([date_init])
                test_result.append([date_end])
                for i in range(len(test_result)):
                    for j in range(len(test_result[i])):
                        # The program write all the results in the excel
                        wrk_sheet.cell(row=j+1+2, column=i+1).value = test_result[i][j]
                wrk_sheet['V3'].alignment = al1
                wrk_sheet['W3'].alignment = al1
                excel.save(merging_path + '\\' + truck)
            print(truck + ' done')
        print('Excels filled')
```

Anexo B Código Segunda Etapa Tratamiento de Datos

```
1.
2. # %% Importar bilbiotecas
3.
4. import os
5. import numpy as np
6. import pandas as pd
7. import openpyxl
8. import time
9.
10.      # %% path
11.
12.      data_path = r'/Users/GBL/Desktop/datamot/Merged data 14 de
   julio'
13.
14.      final_path =
   r'/Users/GBL/Desktop/datamot/Split_filtered_data02AGOSTO4col.xlsx'
15.
16.
17.
18.      # %% Program
19.
20.      #================================================================
   ==================
21.      # Generar Excel
22.      #================================================================
   ==================
23.
24.      ti = time.clock()
25.
26.      main_compo = ['Fe', 'Cu', 'Ni', 'Sn', 'Al', 'Si', 'Pb', 'Cr',
   'Na', 'K',
27.                    'B', 'Ag']
28.      levels = [[25, 40], [5, 15], [3, 5], [3, 5], [5, 10], [7,
   15], [5, 10],
29.                [3, 6], [15, 35], [30, 40], [20, 25], [3, 5]]
30.
31.      al1 = openpyxl.styles.Alignment(horizontal='center',
   vertical='center')
32.
33.      excel_fin = openpyxl.Workbook()
34.      wrk_sheet = excel_fin['Sheet']
35.      wrk_sheet.merge_cells('A1:AV1')
36.      wrk_sheet['A1'] = 'Components state'
37.      wrk_sheet['A1'].alignment = al1
38.      wrk_sheet.merge_cells('AW1:AW3')
39.      wrk_sheet['AW1'] = 'Hours'
40.      wrk_sheet['AW1'].alignment = al1
41.      wrk_sheet.merge_cells('AX1:AX3')
42.      wrk_sheet['AX1'] = 'hor'
43.      wrk_sheet['AX1'].alignment = al1
44.      wrk_sheet.merge_cells('AY1:AY3')
45.      wrk_sheet['AY1'] = 'Classes'
46.      wrk_sheet['AY1'].alignment = al1
47.
```

```
48.
49.
50.      for i in range(len(levels)):
51.          wrk_sheet.merge_cells(start_row=2, start_column=4*i + 1,
   end_row=2,
52.                                end_column=4*i+4)
53.          wrk_sheet.cell(row=2, column=4*i + 1).value =
   main_compo[i]
54.          wrk_sheet.cell(row=2, column=4*i + 1).alignment = al1
55.          wrk_sheet.cell(row=3, column=4*i + 1).value =
   'Precaution'
56.          wrk_sheet.cell(row=3, column=4*i + 2).value = 'Pre +
   5xCrtitical'
57.          wrk_sheet.cell(row=3, column=4*i + 3).value = 'ppm'
58.          wrk_sheet.cell(row=3, column=4*i + 4).value = 'Total ppm'
59.          wrk_sheet.cell(row=3, column=4*i + 1).alignment = al1
60.          wrk_sheet.cell(row=3, column=4*i + 2).alignment = al1
61.          wrk_sheet.cell(row=3, column=4*i + 3).alignment = al1
62.          wrk_sheet.cell(row=3, column=4*i + 4).alignment = al1
63.
64.
65.      excel_fin.save(final_path)
66.
67.
68.      #==========================================================
   ==================
69.      # Rellenar excel
70.      #==========================================================
   ==================
71.
72.      truck_list = os.listdir(data_path)
73.      Matrix = []
74.      truck_list.pop(0)
75.      truck_list.pop(-1)
76.      truck_list.pop(-1)
77.      truck_list.pop(-1)
78.
79.      for truck in truck_list:
80.          filename = data_path + '/' + truck
81.          print(filename)
82.          excel = pd.read_excel(filename, sheetname=None,
   skiprows=[0])
83.          sheet_list = list(excel.keys())
84.          for sheet in sheet_list:
85.              Lines = []
86.              dataframe = excel[sheet]
87.              hours = list(dataframe['Hours'])
88.              components = np.array(dataframe.iloc[:, 0:20],
   dtype=int)
89.
90.      #==========================================================
   ==================
91.      # Econtrar intervalos de 500  horas
92.      #==========================================================
   ==================
93.
94.              index = [0]
```

```
95.             hours.reverse()
96.             quotient = [0]
97.             for i in range(1, len(hours)):
98.                 quotient.append(hours[i] // 1000)
99.                 if quotient[i] > quotient[i-1]:
100.                    index.append(i)
101.            hours.reverse()
102.
103.            nb_line = np.shape(dataframe)[0]
104.            index = index[::-1]
105.            for i in range(1, len(index)-1):
106.                df = dataframe.iloc[index[i]:nb_line, :]
107.                line = []
108.                linep=[]
109.                p=[0,0,0]
110.                pin=[0,0,0]
111.                score = 0
112.                maxs=[0,0,0]
113.                for cp in main_compo:
114.                    prec_nb = 0
115.                    crit_nb = 0
116.                    prec_lvl = levels[main_compo.index(cp)][0]
117.                    crit_lvl = levels[main_compo.index(cp)][1]
118.                    total = 0
119.                    total1 = 0
120.                    for i in range(len(df[cp])):
121.                        if prec_lvl <= list(df[cp])[i] <
   crit_lvl:
122.                            prec_nb += 1
123.                            crit_nb +=1
124.                        elif crit_lvl <= list(df[cp])[i]:
125.                            crit_nb +=5
126.                        if list(df[cp])[i-1]-list(df[cp])[i] > 0
   :
127.                            total += (list(df[cp])[i-1]-
   list(df[cp])[i])
128.                        total1 += list(df[cp])[i]
129.
130.                    line.append(prec_nb)
131.                    line.append(crit_nb)
132.                    line.append(total)
133.                    line.append(total1)
134.
135.
136.                line.append(dataframe['Hours'][0])
137.
138.                line.append(dataframe['Hours'][i])
139.                Lines.append(line)
140.
141.            n = len(index)
142.            df = dataframe
143.            line = []
144.            linep=[]
145.            p=[0,0,0]
146.            pin=[0,0,0]
147.            for cp in main_compo:
148.                prec_nb = 0
```

```
149.                 crit_nb = 0
150.                 prec_lvl = levels[main_compo.index(cp)][0]
151.                 crit_lvl = levels[main_compo.index(cp)][1]
152.                 total = 0
153.                 total1 = 0
154.                 for i in range(len(df[cp])):
155.                     if prec_lvl <= list(df[cp])[i] < crit_lvl:
156.                         prec_nb += 1
157.                         crit_lvl +=1
158.                     elif crit_lvl <= list(df[cp])[i]:
159.                         crit_nb +=5
160.                     if list(df[cp])[i-1]-list(df[cp])[i] > 0 :
161.                             total += (list(df[cp])[i-1]-
   list(df[cp])[i])
162.                     total1 += list(df[cp])[i]
163.                 line.append(prec_nb)
164.                 line.append(crit_nb)
165.                 linep.append(crit_nb)
166.                 line.append(total)
167.                 line.append(total1)
168.
169.
170.             line.append(dataframe['Hours'][0])
171.             line.append(dataframe['Hours'][0])
172.             Lines.append(line)
173.             Lines = np.array(Lines)
174.
175.
176.             mean, std = [], []
177.             for i in range(len(Lines[:])-1):
178.                 mean.append(np.mean(Lines[:, i]))
179.                 std.append(np.std(Lines[:, i]))
180.
181.
182.             Lines = list(Lines)
183.             for i in range(len(Lines)):
184.                 Lines[i] = list(Lines[i])
185.
186.             for i in range(len(Lines)-1):
187.                 score = 0
188.                 for j in range(len(mean)):
189.                     if Lines[i][j] > mean[j]:
190.                         score += 1
191.                 if score >= 4:
192.                     Lines[i].append(1)
193.                 else:
194.                     Lines[i].append(0)
195.             Lines[-1].append(1)
196.         for i in range(len(Lines)):
197.             Matrix.append(Lines[i])
198.
199.
200.     excel_fin = openpyxl.load_workbook(final_path)
201.     wrk_sheet = excel_fin['Sheet']
202.     for i in range(len(Matrix)):
203.         wrk_sheet.append(Matrix[i])
204.     excel_fin.save(final_path)
```

```
205.
206.     tf = time.clock()
207.
208.     print('Done in ', tf - ti, 's')
209.
210.
```

Anexo C Código Cluster

```
1.
   #####código clúster + evaluación
2. import pandas as pd
3. from sklearn.cluster import KMeans
4. from sklearn import metrics
5. import matplotlib.pyplot as plt
6. import seaborn as sns
7. import numpy as np
8. import pml.unsupervised.clustering
9. import time
10.      import os
11.      BD =
   r'/Users/GBL/Desktop/datamot/Split_filtered_data02AGOSTO4col.xlsx'
12.      savefile =
   r'/Users/GBL/Desktop/datamot/kmeanscluster02agosto.xlsx'
13.
14.      colnames=['Fe','Cu','Ni','Sn','Al','Si','Pb','Cr','Na','K','B
   ','Ag']
15.      colnames2=[0]
16.
17.      for i in range(0,len(colnames)):
18.          colnames2.append(colnames[i]+' Precaution')
19.          colnames2.append(colnames[i]+' Pre + 5xCritical')
20.          colnames2.append(colnames[i]+' ppm')
21.          colnames2.append(colnames[i]+' Total ppm')
22.
23.
24.
25.      del colnames2[0]
26.      colnames2.append('Hours')
27.      colnames2.append('hor')
28.      colnames2.append('Classes')
29.
30.      hdb=pd.read_excel(io = BD,
31.                        header=2,
32.                        sheetname='Sheet',
33.                        names=None,
34.                        parse_cols=None,
35.                        parse_dates=False,
36.                        date_parser=None,
37.                        na_values=None,
38.                        thousands=None,
39.                        convert_float=True,
40.                        has_index_names=None,
41.                        converters=None,
42.                        true_values=None,
43.                        false_values=None,
44.                        engine=None,
45.                        squeeze=False)
46.
47.      hdb.columns = colnames2
48.      hdb=hdb[hdb.Classes != 0]
49.
50.
```

```
51.      start_time = time.time()
52.
53.      ##col para clúster
54.      colcl=[]
55.      for i in range (0,len(colnames)):
56.          colcl.append(colnames2[4*i+1])
57.      colcl.append(colnames2[-3])   #hora
58.
59.
60.      Hdbscan = KMeans(n_clusters=4, random_state=0,init='k-
   means++').fit(hdb[colcl])
61.
62.      print("--- %s seconds ---" % (time.time() - start_time))
63.
64.
65.      hdb['labels'] = Hdbscan.labels_
66.
67.
68.
69.
70.
71.      db2 = hdb.sort_values('labels', axis=0, ascending=True,
   inplace=False, kind='quicksort', na_position='last')
72.
73.      print("--- %s seconds ---" % (time.time() - start_time))
74.
75.      ###################################################
76.      ##evaluación
77.
78.      #SILHOUETTE SCORE
79.      #ev  sil http://scikit-
   learn.org/stable/modules/generated/sklearn.metrics.silhouette_score
   .html
80.      X=hdb[colcl]
81.      labels=Hdbscan.labels_
82.      SIL=metrics.silhouette_score(hdb[colcl], labels,
   metric='euclidean')
83.
84.
85.
86.
87.
88.
89.      sil=np.zeros(9)
90.      for i in range(2,11):
91.          kmeans = KMeans(n_clusters=i, random_state=0,init='k-
   means++').fit(hdb[colcl])
92.          hdb['labels'] = kmeans.labels_
93.          X=hdb[colcl]
94.          labels=kmeans.labels_
95.          sil[i-2]=metrics.silhouette_score(hdb[colcl], labels,
   metric='euclidean')
96.      print(sil)
97.
98.      ###################################################
99.      ######resultados evaluación
100.     x=[2,3,4,5,6,7,8,9,10]
```

```
    plt.plot(x,sil)
    plt.show()
    plt.title('Silhouette Score')
    plt.savefig(r'/Users/GBL/Desktop/Silhouette Score.png')
    plt.clf()
    ##################################################
    ###graficar
    tart_time = time.time()
    print("--- %s seconds ---" % (time.time() - start_time))


    labels = db2['labels']
    palette = sns.color_palette('deep', np.unique(labels).max() + 1)
    colors = [palette[x] if x >= 0 else (0.0, 0.0, 0.0) for x in labels]
    print("--- %s seconds ---" % (time.time() - start_time))
    print('scatter')

    plt.scatter(range(len(db2['Hours'])), db2['Hours'], c=colors)
    #plt.savefig(gpath)
    print('pairplot')

    print("--- %s seconds ---" % (time.time() - start_time))
    plt.show()
    print("--- %s seconds ---" % (time.time() - start_time))
    ##################################################
    ##guardar gráficos
    gpath = r'/Users/GBL/Desktop/datamot/graph0208/'    ####crear carpeta para gráficos
    directory = os.path.dirname(gpath)
    os.mkdir(directory)

    for c in colnames2:
        pathgr=gpath + c + '.png'
        plt.scatter(range(len(db2[c])), db2[c], c=colors)
        plt.title(c)
        plt.savefig(pathgr)
        plt.clf()

    plt.scatter(range(len(db2['Hours'])), db2['Hours'], c=colors)

    plt.show()


    ##################################################


    ##################################################
    ##guardar resultados
    excel_fin = pd.ExcelWriter(savefile)
    db2.to_excel(excel_fin,'Sheet1')
    excel_fin.save()



    ##################################################
```

Anexo D Multilayer Perceptron

Este código es una adaptación del código propuesto en la segunda tarea del curso MA5203-1 Aprendizaje de Máquinas Probabilístico, del professor Felipe Tobar.

```
##### Cargar Datos


import tensorflow as tf
import pandas as pd
import numpy as np
from sklearn.cross_validation import train_test_split
import matplotlib.pylab as plt
import time


      FILE_PATH = r'/Users/GBL/Desktop/datos y codigos 2007/kmeanscluster02agosto.xlsx'
      data=pd.read_excel(io = FILE_PATH,
                         header=0,
                         sheetname='Sheet1',
                         names=None,
                         parse_cols=None,
                         parse_dates=False,
                         date_parser=None,
                         na_values=None,
                         thousands=None,
                         convert_float=True,
                         has_index_names=None,
                         converters=None,
                         true_values=None,
                         false_values=None,
                         engine=None,
                         squeeze=False)

      print("Raw data loaded successfully...\n")

      del data['Hours']
      del data['Classes']
      del data['hor']
      colnames=['Fe','Cu','Ni','Sn','Al','Si','Pb','Cr','Na','K','B','Ag']

      for i in range(0,len(colnames)):
          del data[colnames[i]+' Total ppm']
          del data[colnames[i]+' ppm']
         # del data[colnames[i]+' Total ppm']

      data=data.as_matrix(columns=None)

      data.shape
      ########################
```

```
47.      ##tratar datos pre entrenamiento
48.      X = data[:, :24].astype(np.float32)
49.      Y = data[:, 24].astype(int)
50.
51.      print('Dimension de los datos de entrada: ', X.shape)
52.      print('Dimension de los dato de salida: ', Y.shape)
53.
54.      X_train, X_test, y_train, y_test = train_test_split(X, Y,
   test_size=0.30)
55.
56.      print('Ejemplos de entrenamiento: %i' %X_train.shape[0])
57.      print('Ejemplos de test: %i' %X_test.shape[0])
58.
59.      def one_hot_encode(y, n_classes=4):
60.          '''
61.          DESCRIPCION: Función que transforma las clases (variable
   categórica) en una codificación numérica.
62.
63.          INPUT:
64.
65.          (y)          Vector 1-darray con las clases
66.          (n_classes)  Cantidad de clases, por defecto 2.
67.
68.          OUTPUT:
69.          (encode_y)   Vector n_classes-darray con la codificación
   para cada ejemplo
70.
71.          '''
72.          encode_y = np.zeros((y.shape[0], n_classes), dtype=int)
73.          for i in range(0, y.shape[0]):
74.              clase = np.argmax(y[i])
75.              encode_y[i, y[i]] = 1
76.
77.          return encode_y
78.      Y_train = one_hot_encode(y_train)
79.      Y_test = one_hot_encode(y_test)
80.      for i in range(10):
81.          print('El ejemplo número', i, 'con etiqueta',
   y_train[i],'ha sido codificado en:', Y_train[i, :] )
82.
83.      print('Dimension etiquetas luego del encoder: (%i, %i)'
   %(Y_train.shape))
84.
85.
86.
87.      ##########################
88.
89.      ##########################
90.      ## ESTRUCTURA GRAFO
91.
92.      #definición cantidad de neuronas:
93.      n_hidden =  14 # Neuronas de la capa oculta
94.      n_input = 24 # Neuronas de la capa de entrada (n_features)
95.      n_classes = 4 # Clase A o B
96.
97.      #placeholder entrada
98.      x = tf.placeholder(tf.float32, [None, n_input])
```

```
99.
100.     #placeholder salida
101.     y = tf.placeholder(tf.float32, [None, 4])
102.
103.     learning_rate = 0.001 # Tasa de aprendizaje
104.     # Crear red
105.     def multilayer_perceptron(x, weights, biases):
106.         # Primera capa oculta con funcion de activacion ReLU
107.         layer_1 = tf.add(tf.matmul(x, weights['h1']),
   biases['b1'])
108.
109.
110.         layer_3 = tf.nn.relu(layer_1)
111.
112.         out_layer = tf.matmul(layer_3, weights['out']) +
   biases['out']
113.         return out_layer
114.
115.     # Generalmente los parámetros de la red se inicializan
   aleatoriamente.
116.     weights = {
117.         'h1': tf.Variable(tf.random_normal([n_input, n_hidden])),
118.
119.         'out': tf.Variable(tf.random_normal([n_hidden,
   n_classes]))
120.     }
121.     biases = {
122.         'b1': tf.Variable(tf.random_normal([n_hidden])),
123.
124.         'out': tf.Variable(tf.random_normal([n_classes]))
125.     }
126.
127.     # Construir el modelo
128.     pred = multilayer_perceptron(x, weights, biases)
129.
130.     #función de costo
131.     cost =
   tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(logits=pred,
   labels=y))
132.
133.     #optimización y entrenamiento
134.     optimizer = tf.train.GradientDescentOptimizer(learning_rate)
135.     train = optimizer.minimize(cost)
136.
137.     #evaluación del modelo
138.     correct_prediction = tf.equal(tf.argmax(pred, 1),
   tf.argmax(y, 1))
139.     accuracy = tf.reduce_mean(tf.cast(correct_prediction,
   "float"))
140.
141.     init = tf.global_variables_initializer()
142.
143.
144.     ##########################
145.
146.     ##########################
147.     ###EJECUCIÓN
```

```
148.
149.     training_epochs = 13000 # épocas de entrenamiento
150.     display_step = 1000 # iteraciones para metrics-display
151.
152.     # Ejecución del grafo
153.
154.     with tf.Session() as sess:
155.         sess.run(init)
156.         init_time = time.time()
157.         saver = tf.train.Saver()
158.
159.         # Fit all training data
160.         for epoch in range(training_epochs):
161.             sess.run(train, feed_dict={x: X_train, y: Y_train})
162.             #Display logs per epoch step
163.             if epoch % display_step == 0:
164.                 c, train_ac = sess.run([cost, accuracy],
   feed_dict={x: X_train, y:Y_train})
165.                 test_ac = accuracy.eval({x: X_test, y: Y_test})
166.                 print("Epoch: %04d  Cost: %f  Train Accuracy:
   %f  Test Accuracy: %f " % ((epoch), c, train_ac, test_ac))
167.         training_time = time.time() - init_time
168.
169.         saver.save(sess, 'my_test_model')
170.
171.
172.         pred_one_hot = sess.run(pred, feed_dict={x: X_test})
173.         y_pred = np.argmax(pred_one_hot, axis=1)
174.
175.         print("Optimization Finished!")
176.         print('Training time: %f seconds' % training_time)
177.
178.
179.     ########################
180.
181.     ########################
182.     ###Evaluar
183.
184.     import itertools
185.     from sklearn.metrics import confusion_matrix,
   classification_report
186.     def plot_confusion_matrix(cm, classes, normalize=False,
   title='Confusion matrix', cmap=plt.cm.Blues):
187.         """
188.         DESCRIPCION: Funcion que sirve para plotear una matriz de
   confusion.
189.
190.         INPUT:
191.         (cm)         Matriz de confusion (n-darray)
192.         (classes)    Nombre de las clases, lista de strings con
   el nombre de cada clase.
193.         (title)      Titulo de la matriz de confusion, string.
194.         (cmap)       Color de la matriz de confusion
195.
196.         """
197.         plt.figure(figsize=(8,8))
198.         plt.title(title)
```

```
199.         tick_marks = np.arange(len(classes))
200.         plt.xticks(tick_marks, classes, rotation=45)
201.         plt.yticks(tick_marks, classes)
202.
203.         if normalize:
204.             cm = cm.astype('float') / cm.sum(axis=1)[:,
   np.newaxis]
205.             cm = np.round(cm, decimals=3)
206.
207.             print("Normalized confusion matrix")
208.         else:
209.             print('Confusion matrix, without normalization')
210.
211.         #print(cm)
212.
213.         thresh = cm.max() / 2.
214.         for i, j in itertools.product(range(cm.shape[0]),
   range(cm.shape[1])):
215.             plt.text(j, i, cm[i, j],
216.                      horizontalalignment="center",
217.                      color="white" if cm[i, j] > thresh else
   "black")
218.         plt.imshow(cm, interpolation='nearest', cmap=cmap)
219.
220.         plt.colorbar()
221.
222.         plt.tight_layout()
223.         plt.ylabel('True label')
224.         plt.xlabel('Predicted label')
225.
226.     cm = confusion_matrix(y_true=y_test, y_pred=y_pred)
227.     plot_confusion_matrix(cm, ['Clase 1', 'Clase 2','Clase 3',
   'Clase 4'], normalize=True)
228.     clc_report = classification_report(y_test, y_pred)
229.     print(clc_report)
230.     plt.show()
231.
232.
233.
234.
235.     ########################
```