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

DETERMINACIÓN DE MODELOS DE FALLAS EN BOMBA MEDIANTE APRENDIZAJE AUTOMÁTICO

TESIS PARA OPTAR AL GRADO DE MAGISTER EN CIENCIAS DE LA INGENIERÍA
MENCIÓN MECÁNICA

JORGE ARTURO MARÍN CASTILLO

PROFESORA GUÍA:

VIVIANA MERUANE NARANJO

MIEMBROS DE LA COMISIÓN:

JUAN CRISTÓBAL ZAGAL

RUBÉN FERNÁNDEZ URRUTIA

SANTIAGO DE CHILE

2021

RESUMEN DE LA TESIS PARA OPTAR AL GRADO DE:
Magister en Ciencias de la Ingeniería, Mención Mecánica
POR : Jorge Marín Castillo
FECHA : 2021
PROFESORA GUÍA : Viviana Meruane Naranjo.

DETERMINACIÓN DE MODELOS DE FALLAS EN BOMBA MEDIANTE APRENDIZAJE AUTOMÁTICO

En los últimos años ha surgido el desarrollo de Inteligencia Artificial (IA), en diferentes ámbitos, siendo una metodología que permite procesar gran cantidad de datos e infiriendo conocimiento en muy poco tiempo de procesamiento informático. Similarmente el mantenimiento según condición se basa en evaluar datos para que los especialistas infieran el estado de salud de la maquinaría, siendo esta actividad estratégica en la gran Minería. La industria minera, es una actividad cada vez más exigida y requiere de una evaluación permanente de la salud operacional de sus máquinas consideradas como críticas para el proceso productivo. Los especialistas en monitoreo de condición son escasos en el país y solo se logra cubrir turnos en jornadas diurna. Por otra parte, los costos de los sensores son cada vez más económicos y los equipos estas están que cada vez más sensorizados. En sistemas de control automatizado (DCS su sigla en inglés), siendo esta una capacidad ya instalada y disponible.

El propósito de esta investigación es utilizar esta capacidad instalada de sensores, para obtener una metodología de diagnóstico automatizada y permanente, aplicando IA la salud de las maquinas, existiendo un vacío de conocimiento en torno a cómo tratar tal volumen de información que entregan dichos sensores cada cierta cantidad de segundos, día y noche, los 365 días del año. Este estudio tiene por final llenar ese vacío.

Con la aplicación de IA se busca alertar oportunamente mediante la detección de comportamientos sintomático anormales del estado de salud de las máquinas evitando fallas catastróficas, logrando de esta manera la continuidad operacional de la minera.

Para tal fin, se usaron datos de sensores de una bomba de impulsión de agua desalada de 3500 kw, equipo crítico para el proceso productivo, relativamente nuevo, sin un historial de fallas significativas. La técnica de diagnóstico de anormalidades se aplica a un volumen del orden de 9 millones de datos operacionales aportados por la minera, entregados por: 8 sensores de vibración, 4 sensores de temperatura, un sensor de caudal y tres sensores de corriente de alimentación del motor, durante un periodo de 3 meses de recolección.

Para obtener los resultados esperados, se procesan los datos iniciando con limpieza de datos, luego sincronizándolos en el tiempo, efectuando una metodología de reducción de variable para finalmente aplicar una metodología de detección de anormalidades, luego se analizan los resultados ya que el autor tiene conocimiento del comportamiento sintomático de fallas de la bomba.

Como resultado se logra detectar una anormalidad la que se comprueba en los archivos del historial del equipo que posee la minera, que efectivamente existió una falla durante la noche que evoluciono en menos de 3 horas. De esta manera se valida la metodología.

Se concluye que es posible detectar fallas con datos reales para un equipo minero. Esta metodología permitirá a la Minera tener un control sobre la bomba, día y noche sin una mayor inversión en instalación de sensores, solo aplicando conocimiento y desarrollo informático.

DEDICATORIA:

Para Alma Hila

AGRADECIMIENTOS:

A mi familia, a la Minera que aporto con los datos para este estudio brindando información y dando las facilidades para probar el algoritmo. También a las personas de la empresa MCM Chile que me han permitido el conocimiento del monitoreo de condición.

TABLA DE CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE ILUSTRACIONES

1 Introducción

La gran Minería, se desarrolla en Chile principalmente en la alta cordillera del Desierto de Atacama. El producto que ahí se elabora es el concentrado de cobre. Se requiere agua para este proceso. La escasez de este recurso, en el desierto de Atacama, hace necesario trabajar con agua de mar directamente o bien desalada, la que debe ser trasportada a distancia, del orden de los 200 km, y llegar a una altura de 3000 msnm, donde se encuentran los centros mineros.

Las bombas de impulsión de agua desalada son activos críticos en el proceso de producción de concentrado de cobre que deben funcionar día y noche, todos los días del año en forma ininterrumpida. La estrategia del Monitoreo de Condición es vital para asegurar una operación confiable de estos equipos, siendo los analistas de monitoreo de condición un factor clave en este proceso.

En la actualidad la detección oportuna de síntomas que indique una potencial falla catastrófica se realiza mediante técnicos analistas en el país. Solo hay dos centros de formación superior que forman los técnicos en monitoreo de condición y de acuerdo con estimaciones provenientes del mundo productivo, como la asociación gremial SOFOFA, en nuestro país existe un déficit que fluctúa entre los 600.000 y 700.000 técnicos. Esa cifra incluye la demanda de poco más de un 10% para el sector minería [15], Esta realidad de déficit de técnicos afecta a los técnicos en monitoreo de condición, y en la gran mayoría de los casos estos son formados al interior de las empresas.

Dada la gran demanda por tener una industria nacional competitiva en los mercados internacionales, es necesario suplir este déficit de profesionales y optimizar este recurso, mediante sistemas informáticos que automaticen con un buen nivel de asertividad los diagnósticos. De acuerdo a la experiencia profesional, un 80% de los diagnósticos son sencillos y no requieren un gran entrenamiento. El 20% restantes, requieren un nivel de expertise mayor para su análisis. Es aquí donde los sistemas informáticos de la línea de (IA) mediante detección de anomalías de datos resultan ser una buena alternativa a investigar para automatizar el diagnóstico y optimizar la escasez de profesionales asertivos. Considerando que las técnicas IA, no es transparente el proceso del algoritmo, por lo que requiere de un especialista para analizar los datos, lo cual resulta relevante para este estudio la experiencia en monitoreo de condición del autor.

1.1 Objetivos Generales

Obtener una metodología de detección oportuna de fallas en equipos críticos para la Minera, entrenando algoritmos de detección de anomalías, aplicados a una bomba de impulsión de agua desalada sensorizada y validando los resultados con anormalidades previamente detectadas.

1.2 Objetivos Específicos

Entrenar con casos reales DCS detector de anomalías. Para lo cual se requiere:

1.-Analizar los datos obtenidos durante tres meses, cada segundo del DCS de bomba de impulsión de agua, efectuando una limpieza de datos si se requiere.

2.-Entrenar algoritmo detector de anomalías

3.-Validar los resultados con anormalidades previamente detectadas

1.3 Antecedentes

Los estudios al respecto se basan en datos obtenidos en laboratorio [4], [5] y no en datos reales. Este estudio se basa en datos reales. Se ha conseguido el patrocinio de una gran minera del cobre de Chile, quienes han aportado con datos reales con objeto de ser analizados. Estos datos corresponden a una bomba de impulsión de agua que no tiene un historial de falla significativo por ser relativamente nueva (dos años de uso). En un análisis previo de la data ofrecida por la Minera, relacionada con datos de vibración on line y off line que dispone, y que son de gran calidad para el análisis (espectros de vibración), están almacenados en software que no permiten la extracción de los datos, lo cual es requisito fundamental para aplicar técnicas de detección de anomalías, en cambio si se tienen datos extraíbles de Sistema de Control Distribuido, más conocido por sus siglas en inglés DCS (Distributed Control System), estos datos son de menor calidad, los cuales no son de gran valor para el vibroanálisis (solo valor RMS), pero tienen la ventaja que son extraíbles y se dispone de otras variables como temperatura, amperaje de motor y caudal, datos que pueden ser correlacionados. Es por esta razón que para este estudio solo se utilizaron datos de vibración RMS y no espectrales.

El aprendizaje automático es un tipo de IA en que algoritmos computacionales tienen la capacidad de aprender, sin ser programados explícitamente y pueden cambiar cuando se exponen a nuevos datos.

Estos sistemas de aprendizaje automático buscan relaciones entre los datos para encontrar patrones. Sin embargo, en lugar de extraer los datos para la comprensión humana – como es el caso de las aplicaciones de minería de datos – el aprendizaje automático utiliza esos datos para detectar patrones y ajustar las acciones del programa en consecuencia.

Los algoritmos del aprendizaje automático se clasifican a menudo como supervisados, semisupervisado o no supervisados [4]. Los algoritmos supervisados pueden aplicar lo que se ha aprendido en el pasado mediante una data que están etiquetados, en el contexto de este estudio, se conoce el estado de salud para cada registro, los semisupervisados son similares a los anteriores y se diferencian en que la data de aprendizaje solo se tienen datos etiquetados y no etiquetados. Los algoritmos no supervisados pueden extraer inferencias de conjuntos de datos y se aplican para datos no etiquetados, que es el caso de este estudio, ya que no hay registro del estado de salud de la bomba cada vez que se registraron datos de vibración y operacionales.

Los algoritmos agrupan una serie de vectores según un criterio utilizando, en general, criterios de distancia o similitud.

Reducción de dimensionalidad

Cuando se tienen muchas variables y forman un hiperespacio mayor el plano (2D) o tridemensional (3D) es recomendable reducir la dimensionalidad [6], que consiste en descomponer valores singulares de los datos para proyectarlos en un espacio dimensional más bajo. Las razones por las que interesa reducir la dimensionalidad son:

- Identificar y eliminar las variables irrelevantes.
- No siempre el mejor modelo es el que más variables tiene en cuenta.
- Mejorar el rendimiento computacional, lo que se traduce en un ahorro en coste y tiempo.
- Reducir la complejidad, lo que lleva a facilitar la comprensión del modelo y sus resultados.

La reducción de datos a menos dimensiones a menudo hace que los algoritmos de análisis sean más eficientes y puede ayudar a los algoritmos de aprendizaje automático a hacer predicciones más precisas. Por otra parte, reducir los datos a un pequeño número de dimensiones es útil para fines de visualización gráfica.

En general los métodos de reducción de dimensionalidad requieren de la estandarización de los datos ya que, si no se realizan, podrían no tener un buen comportamiento si las características individuales no son parecidas numéricamente a los datos estándar distribuidos normalmente. Por ejemplo, muchos elementos utilizados en la función objetivo de un algoritmo de aprendizaje, suponen que todas las características están centradas alrededor de 0 y tienen una varianza en el mismo orden. Si una característica tiene una varianza que es de órdenes de magnitud mayor que otras, podría dominar la función objetivo y hacer que el estimador no pueda aprender de otras características correctamente como se esperaba. Para estandarizar los datos se utilizó la biblioteca de Python “scaler.fit_transform” que estandariza los datos de manera que su distribución tenga un valor medio 0 y una desviación estándar de 1.

Los métodos de reducción de variables estudiados son los:

- PCA
- Kernel PCA
- Factor Analysis
- LDA
- MDS
- ISOMAP
- LLE

La descripción de ellos es la siguiente:

1.3.1 PCA

El Análisis de componentes principales (PCA), resume el conjunto de características sin depender de la salida (etiquetas). PCA intenta encontrar las direcciones de la varianza máxima en el conjunto de datos. En un conjunto de grande de variables, hay muchas variables que son simplemente duplicadas de otras variables o tienen una alta correlación con otras variables. Dichas variables son básicamente redundantes y pueden ignorarse. La función de PCA es encontrar variables tan altamente correlacionadas o duplicadas y crear un nuevo conjunto de variables donde haya una

correlación mínima entre las variables, en otras palabras, un conjunto de variables mínima que describa muy bien el proceso. Dado que la variación entre las características no depende de la salida, por lo tanto, PCA no tiene en cuenta las etiquetas de salida [12]

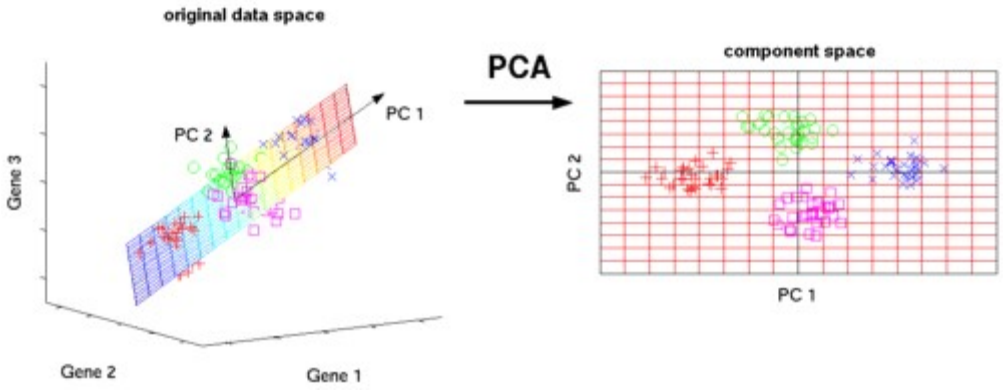


Figura 1: Esquema de reducción de variable 3D a 2D mediante PCA

Como en PCA la primera componente principal, es la que explica la mayor varianza de los datos y corresponda al primer eje. La elección del número de componentes principales que mejor representan los datos es posible cuantificando la suma acumulada de los valores propios para cada eje de componente principal y si nos definimos un valor un valor optimo por ejemplo 85%, en que las componentes principales explican en torno al 85% de la varianza de los datos.

1.3.2 Kernel PCA

Para explicar esta metodología se realizará mediante un ejemplo simple de un set de datos de dos formas de media luna

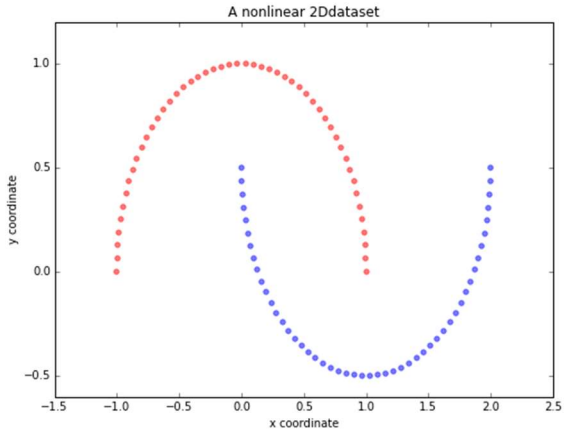


Figura 2: Cambio de variables funciones no lineales con PCA

Dado que las dos formas de media luna son inseparables linealmente, al aplicar el PCA "clásico" el cual no encuentra una buena representación de los datos en el espacio 2D (como se observa en la figura 3), las componentes principales resultantes no producen un subespacio donde los datos están bien separados linealmente. En la figura, los colores azul y rojo no corresponden a etiquetas, solo se agregan con fines de visualización para indicar el grado de separación.

Mediante la reducción de dimensionalidad a través de Kernel PCA aplicados a los datos de media luna. La elección de los parámetros del algoritmo depende del conjunto de datos a separar y se puede obtener a través de técnicas de ajuste de hiperparámetros como la búsqueda de cuadrícula [6]. El ajuste de hiperparámetros es un tema amplio en sí mismo. Mediante Kernel PCA logra separar en nuevas variables como el mostrado en la siguiente figura siguiente.

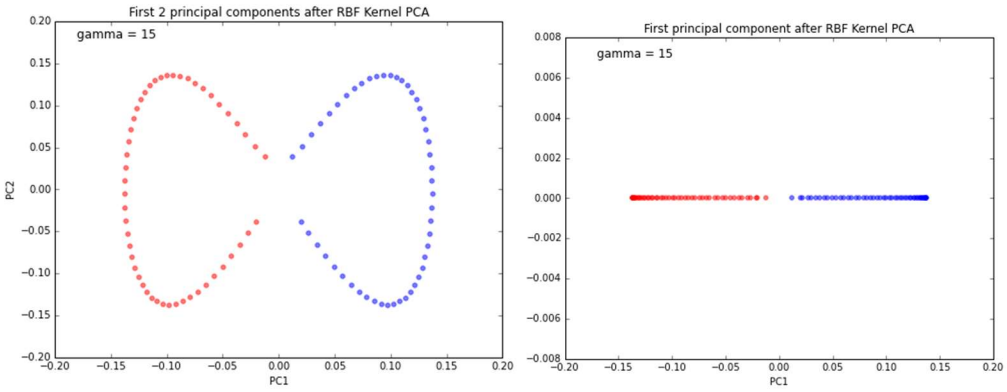


Figura 3: Cambio de variables funciones no lineales mediante Kernel PCA

Podemos ver claramente que la proyección a través del kernel PCA produjo un subespacio donde las clases están bien separadas. Dicho subespacio se puede utilizar como entrada para modelos de clasificación lineal.

1.3.3 Factor Analysis

El concepto clave del Factor Analysis es que múltiples variables observadas tienen patrones de respuestas similares porque todas están asociadas con una variable latente (es decir, no medida directamente). Por ejemplo, las personas pueden responder de manera similar a las preguntas sobre ingresos, educación y ocupación, que están asociadas con la variable no latente estado socioeconómico [7]. En cada análisis factorial, hay la misma cantidad de factores que variables.

Cada factor captura una cierta cantidad de la varianza general en las variables observadas, y los factores siempre se enumeran en orden de cuánta variación explican. El valor propio es una medida de cuánto de la varianza de las variables observadas explica un factor. Cualquier factor con un valor propio ≥1 explica más varianza que una sola variable observada. Entonces, si el factor para el estado socioeconómico tuviera un valor propio alto, explicaría varias variables. Este factor, que captura la mayor parte de la varianza, podría utilizarse en otros análisis.

1.3.4 LDA

El análisis discriminante lineal (LDA, por sus siglas en ingles), intenta encontrar un límite de decisión alrededor de cada grupo de una etiqueta. Luego proyecta los puntos de datos a nuevas dimensiones de manera que los grupos estén tan separados entre sí como sea posible y los elementos individuales dentro de un grupo estén lo más cerca posible del centroide del grupo. Las nuevas dimensiones se clasifican en función de su capacidad para maximizar la distancia entre los grupos y minimizar la distancia entre los puntos de datos dentro de un grupo y sus centroides. Estas nuevas dimensiones forman los discriminantes lineales del conjunto de características.

La aplicación de este método se descarta ya que los datos de estudio de esta tesis no poseen etiquetas.

1.3.5 MDS

El Escalamiento multidimensional (MDS, Multi-Dimension Scaling, de sus siglas en ingles), lo que hace MDS es encontrar un conjunto de vectores en el espacio p-dimensional de modo que la matriz de distancias euclidianas entre ellos corresponda lo más cerca posible a alguna función de la matriz de entrada de acuerdo con una función de criterio llamada el estrés [8].El algoritmo se asigna puntos a coordenadas arbitrarias en el espacio p-dimensional, calcula distancias euclidianas entre todos los pares de puntos, para formar una nueva matriz de distancia la que es comparada con la matriz original.

Compare esta matriz con la matriz D de entrada al evaluar la función de estrés. Cuanto menor es el valor, mayor es la correspondencia entre los dos. Se ajustan las coordenadas de cada punto en la dirección que mejor estresen al máximo. Este proceso se repite hasta que el estrés no disminuya.

A modo de ejemplo, se dispone como dato de entrada las distancia entre ciudades las cuales se grafican en un espacio bidemensional (ver siguiente figura), de esta manera se reduce de 3 a 2 dimensiones, la solución es buena con poca reducción de dimensión y pocos datos, al aumentar esta variables la solución no es exacta y se debe iterar:

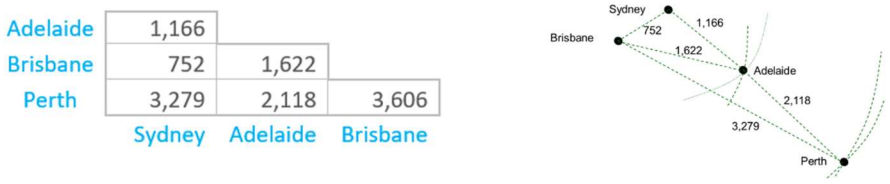


Figura 4: Cambio de variable mediante análisis discriminante lineal MDS de 3 a 2 dimensiones

1.3.6 ISOMAP

El ISOMAT consiste en determinar el número de vecinos (neighbors) en un espacio matemático donde viene recreado localmente un espacio euclidiano, basado según las distancias euclidianas. Para cada punto, asignaremos los vecinos (determinados según el número de vecinos fijados, por ejemplo: en función de un radio fijo)

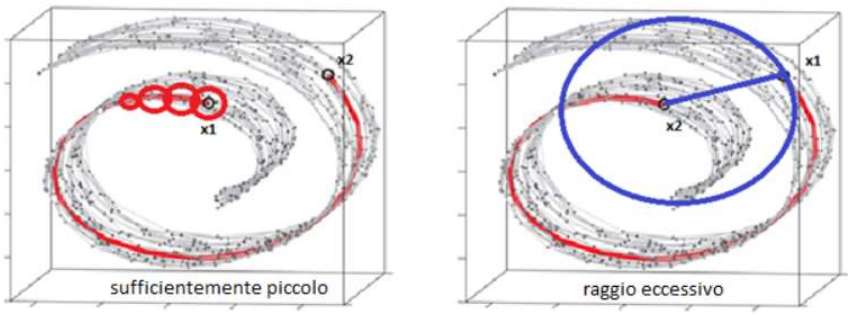


Figura 5: Distancia a vecinos en espacio euclidiano (fig. izquierda) y distancia en espacio geodésico (fig. lado derecho)

calcularemos la distancia geodésica (ver figura anterior lado derecho). Por lo tanto, calcularemos la distancia de cada par de puntos, pero no trabajará en un espacio euclidiano, corresponderá con el camino más corto entre todos los caminos posibles para conectar dos puntos [9]. Una vez encontradas las distancias geodésicas se aplica un MDS (escalado multidimensional) a la matriz de distancia de las geodésicas para reconstruirlo en un nuevo espacio euclidiano que mantendrá la misma estructura geométrica. El algoritmo depende principalmente de la elección de algunos parámetros (la elección del número de vecinos, por ejemplo), ISOMAP es recomendable cuando se tiene idea sobre la estructura de datos porque, comparado con otros métodos lineales, es computacionalmente más intensivo.

1.3.7 LLE

Por sus siglas en inglés (Local Linear Embedding - LLE), es un algoritmo que comienza por encontrar un conjunto de los vecinos más cercanos de cada punto. Calcula un conjunto de pesos para cada punto que mejor describe el punto como una combinación lineal de sus vecinos. Por último, se utiliza una técnica de optimización basada en el vector propio para encontrar la incorporación de bajas dimensiones de puntos, de tal manera que cada punto que sigue se describe con la misma combinación lineal de sus vecinos. LLE.

LLE calcula las coordenadas de un punto en función de sus vecinos. El punto original se reconstruye mediante una combinación lineal, dado por la matriz de ponderación de sus vecinos. El error de reconstrucción está dado por la diferencia entre el valor real y el ponderado por sus vecinos. [11]

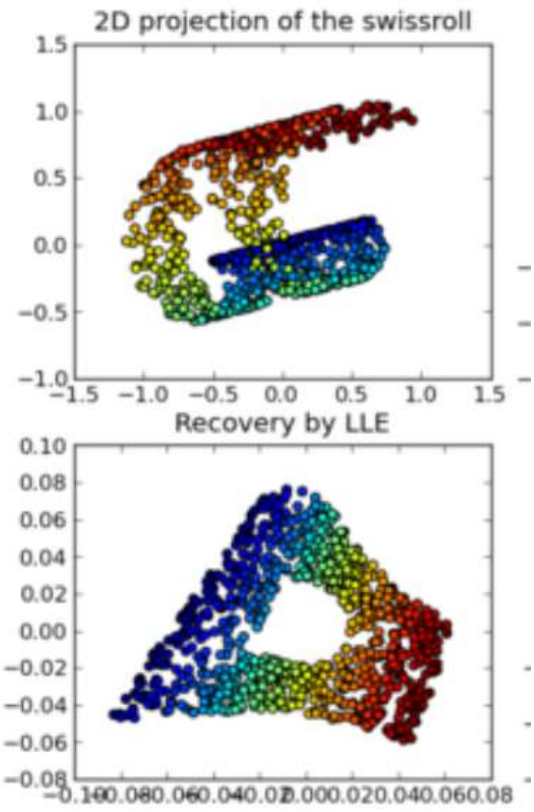


Figura 6: Reducción dimesionalidad mediante algoritmo LLE

1.4 Detector de anormalidad

Muchas aplicaciones requieren poder decidir si una nueva observación pertenece a la misma distribución que las observaciones existentes (lo que se considera una normalidad), o debe considerarse como diferente (anormalidad). A menudo, esta capacidad se utiliza para limpiar conjuntos de datos reales. Se deben hacer dos distinciones importantes:

Como modelo predictivo de detector de anormalidad, los datos de entrenamiento contienen valores anómalos (outliers) que se definen como observaciones que están lejos de los demás.

- Los estimadores de detección de valores anómalos intentan ajustarse a las regiones donde los datos de entrenamiento son los más concentrados, ignorando las observaciones desviadas.

Para efecto de este estudio se usarán los siguientes métodos de detector de anormalidades:

1.4.1 Elliptic Envelope

Elliptic Envelope es una metodología para detectar datos anormales, que asume que los datos normales provienen de una distribución conocida (por ejemplo, una distribución Gaussiana).

Se definen las observaciones anómalas como observaciones que se encuentran lo suficientemente lejos de la forma ajustada.

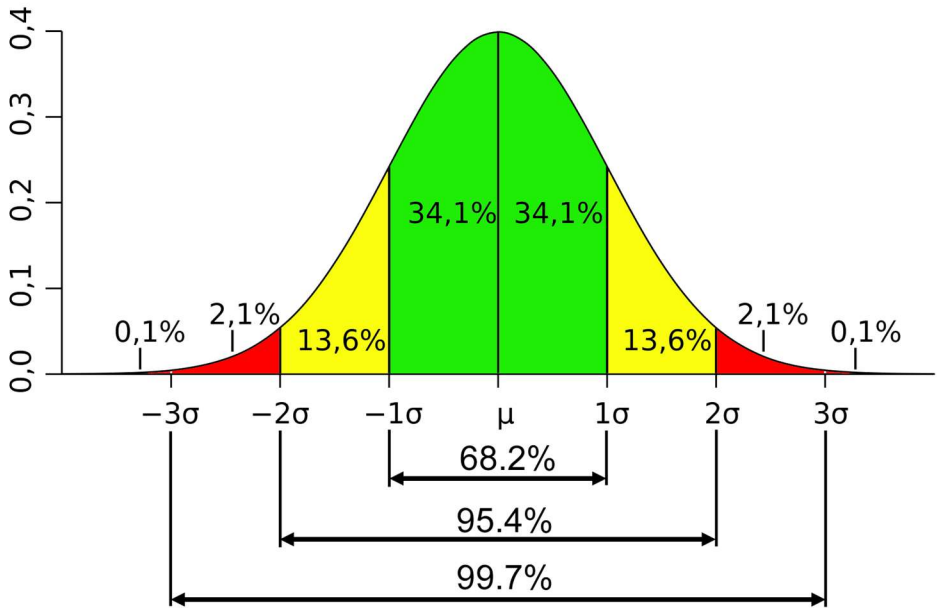


Figura 7: Modelo predictivo EllipticEnvelope

Se ocupó la biblioteca EllipticEnvelope de la biblioteca Scikit-learn de Python, que tiene la variable de ajuste "outlier_frac" que corresponde fracción de valores anómalos y que varía entre 0 y 1, se recomienda un valor inicial de valor 0,1. El código de llamado esta en anexo.

1.4.2 Isolation Forest

El método "Isolation Forest" trata de aislar los valores anómalos al hacer sucesivas divisiones de los datos. Es un algoritmo de referencia para ajustar datos en conjuntos de datos de alta dimensión. Es posible 'aislar' observaciones seleccionando aleatoriamente una entidad y luego seleccionando aleatoriamente un valor dividido entre los valores máximo y mínimo de la entidad seleccionada. Cuando un bosque de árboles aleatorios produce colectivamente longitudes de camino más cortas para muestras particulares, es muy probable que sean anomalías

Se selecciona un parámetro al azar y luego se selecciona al azar un valor de ese parámetro para dividir los datos en dos grupos

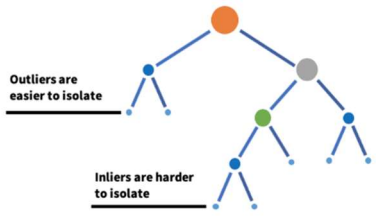


Figura 8: Modelo *Isolation Forest*

Se ocupó la biblioteca "IsolationForest" de la biblioteca Scikit-learn de Python, que tiene la variable de ajuste "n_estimators" que corresponde al número de estimadores base en el conjunto", se recomienda un valor de 100, la variable "outlier_frac" que corresponde fracción de valores

anómalos y que varía entre 0 y 1, se recomienda un valor inicial de valor 0,1. El código de llamado esta en anexo.

1.4.3 Local Outlier Factor

El método Local Outlier Factor (LOF) calcula una puntuación (denominada factor de valores anómalos locales) que refleja el grado de anomalía de las observaciones. Mide la densidad local de un dato dado con respecto a sus vecinos. La idea es detectar las muestras que tienen una densidad sustancialmente menor que sus vecinos.

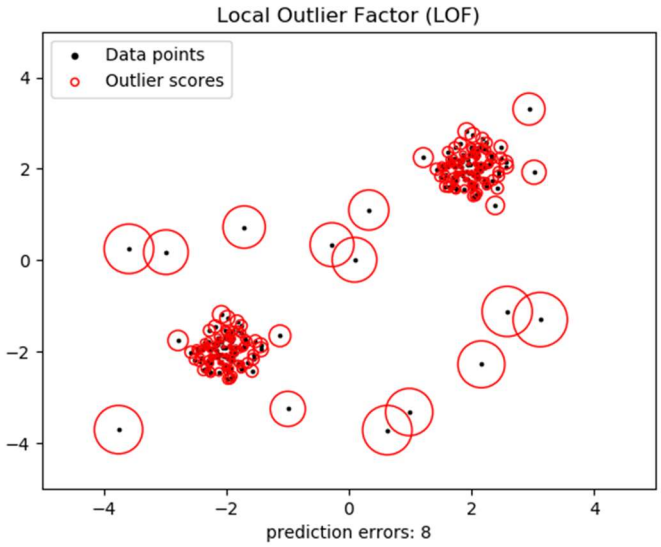


Figura 9: Modelo Local Outlier Factor (LOF)

La puntuación LOF de una observación es igual a la razón de la densidad local promedio de sus "k" vecinos más cercanos, y su propia densidad local: se espera que una instancia normal tenga una densidad local similar a la de sus vecinos, mientras que para los datos anómalos se espera que tengan una densidad local mucho menor.

El número de vecinos considerados suele ser:

1. Mayor que el número mínimo de muestras que debe contener un clúster.
2. Menor que el número máximo de muestras cercanas que podrían ser datos anómalos.

Se ocupó la biblioteca "LocalOutlierFactor" de la biblioteca Scikit-learn de Python, que tiene las variables de ajuste "n_neighbors" que correponde a la cantidad de vecinos y "contamination" que corresponde a la fracción de anómalos. El código de llamado está en anexo.

1.4.4 SVM

Las máquinas de vector de soporte más conocida como SVM (de la sigla en ingles Support Vector Machines), corresponden intuitivamente a los puntos que definen el margen máximo de separación del hiperplano que separa las clases. Se llaman vectores, en lugar de puntos, porque estos «puntos» tienen tantos elementos como dimensiones tenga nuestro espacio de entrada. Es decir, estos puntos

multi-dimensionales se representan con vector de “n” dimensiones. En el siguiente ejemplo se explica conceptualmente como operan

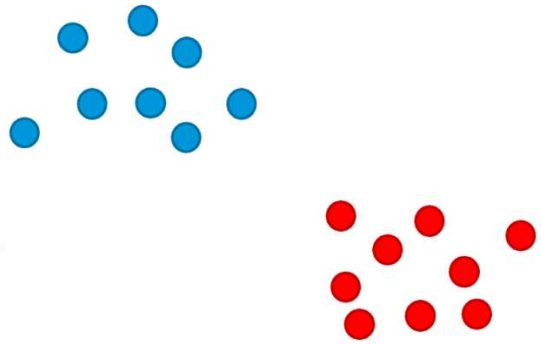


Figura 10: Dos clase azul y rojo, el SVM busca encontrar la frontera que mejor separa ambas clases

Los SVM son los puntos que definen el margen máximo (no necesariamente una línea) que separa las clases, tal como lo muestra la siguiente figura.

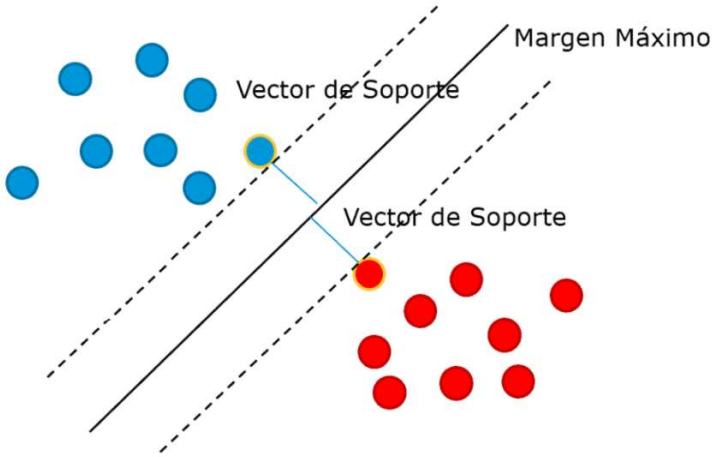


Figura 11: frontera de máxima separación de las clases

Se ocupó la biblioteca “svm.OneClassSVM” de la biblioteca Scikit-learn de Python, que tiene la variable de ajuste “nu” que corresponde al límite superior en la fracción de errores de entrenamiento y un límite inferior de la fracción de vectores de soporte, “kernel” que especifica el tipo de kernel que se utilizará en el algoritmo y el parámetro gamma, que corresponde al coeficiente del kernel seleccionado. El código de llamado esta en anexo.

2 Metodología

Se han definido tres etapas para el desarrollo del tema:

- La primera etapa consiste en el levantamiento de los datos reales de una bomba del DCS de la minera, en esta etapa se realiza una exploración de la data y se depura.

- Una segunda etapa, considerando que los equipos solo tienen 2 años en operación sin muchos casos de falla, se va a construir un sistema de aprendizaje automático sin etiquetas. Se evaluará considerando que la data contiene datos normales en su mayoría, ya que las bombas no han presentado volúmenes significativos de fallas durante su corta vida útil.
- En una tercera etapa, se evaluará la capacidad de desempeño del modelo predictivo con la separación de datos anormales y normales, de acuerdo al conocimiento que posee el autor de las bombas y validando los resultados con anormalidades previamente detectadas

Se usarán para el desarrollo de este estudio algoritmos de aprendizaje automático para detectar el estado de salud de las bombas denominado detector de anormalidades o de detección de novedades ya que solo se cuenta con datos normales en su mayoría.

2.1 Etapa 1: Limpieza de datos:

Aplicado a una data entregada por la Minera y extraída del sistema DCS que posee, principalmente data de vibración RMS, y datos de su entorno sintomático, que son: variables de temperatura de descansos, caudal de la bomba y amperaje de cada fase del motor. En total 16 variables medidos entre noviembre de 2018 y marzo de 2019. El procedimiento metodológico es el siguientes:

Se realiza una limpieza de los datos eliminando datos NaN, (no es un número) y eliminación de datos asociados a un mismo tiempo. Los datos de cada variable están tomados en instante de tiempo diferente y periodicidad diferente.

Sincronización de datos.

Debido a que los datos de cada variable no están tomados en el mismo instante de tiempo y tampoco coinciden las frecuencias de muestreo, como también no son coincidentes la hora de inicio y fin del tren de datos de cada sensor. Si se considera aquellos datos que coincide en el tiempo en el muestreo de las 16 variables, se reduce demasiado la cantidad de datos, por lo que se opta por realizar una sincronización de los datos de acuerdo con un tiempo patrón.

El criterio para seleccionar el tiempo patrón (tiempo de referencia) es el siguiente:

- Se selecciona el tiempo de inicio de la variable en que se comienza a registrar más tarde
- Se selecciona el tiempo final con la variable que tiene el primer tiempo final
- Se selecciona la variable con el mínimo tren de datos establecido para el inicio y fin dados por los criterios anteriores.

La sincronización se realizó con el lenguaje de programación de código abierto, orientado a objetos, Python utilizando la biblioteca de numpy interp, que es una biblioteca de numpy que devuelve la interpolación del set de datos medidos en un sensor a una función con puntos de datos discretos, correspondiente al tiempo de referencia definido de acuerdo al criterio de más arriba. En la gráfica 1 los puntos x de color azul corresponde a los valores medidos por el sensor de vibración en el motor (valores reales) la línea verde corresponde a una interpolación lineal y la naranja a una interpolación cubica. Se observa que la interpolación cubica y lineal son similares por lo que se optó por usar interpolación lineal. De esta manera se puede obtener un valor de vibración para cualquier tiempo, en el caso del estudio al set de tiempo de referencia.

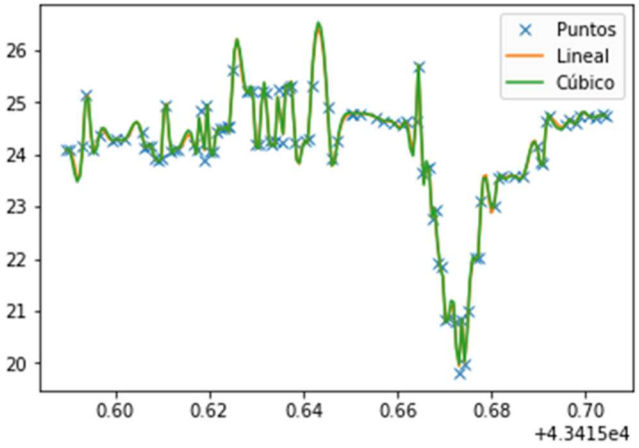


Figura 12: Grafica de interpolación en valores RMS

Reducción de dimensionalidad

Se usaron los métodos de reducción de variables descritos en antecedentes, que son:

- PCA
- Kernel PCA
- Factor Analysis
- LDA
- MDS
- ISOMAP
- LLE

Se evalúa la eficiencia de cada método con el set de datos sincronizado y depurado. Esta eficiencia se evalúa bajo el criterio de:

- Tiempo de procesamiento en computador descrito en anexo.
- Capacidad grafica de aglomerar mejor los datos
- Capacidad de distinguir valores anómalos.

De esta manera se selecciona un método de reducción de variables. En el capítulo siguiente en resultados están los valores obtenidos tanto para reducción en dos y tres dimensiones.

2.2 Etapa 2 : Construcción de sistema detección de anomalías

Considerando que los equipos solo tienen 2 años en operación sin muchos casos de falla, se va a construir un sistema de aprendizaje automático sin etiquetas. Se evaluará considerando que la data

contiene datos normales en su mayoría, ya que las bombas no han presentado volúmenes significativos de fallas durante su corta vida útil.

Se usaron los métodos de detector de anomalías descritos en antecedentes, seleccionando aquellos que separan en fronteras mejor definidas entre normales y anormales.

Para tal efecto se separa el set de datos al azar para construcción del modelo dejando un 70% para entrenamiento del modelo y 30% restante para validar el modelo

2.3 Etapa 3: avaluación del desempeño del modelo

Se valida la capacidad de desempeño del modelo predictivo con la separación de datos anormales y normales, de acuerdo al conocimiento que posee el autor de las bombas, considerando las condiciones meteorológicas de instalación desempeño normal vibracional y validando los resultados con posibles fallas ocurridas en el periodo de tiempo que pertenecen. También se validad el desempeño con el 20% de los datos que se dejaron para estos efectos.

3 Resultados

3.1 Detalle técnico del equipo en estudio

El agua desalada del mar es producida mediante el proceso de osmosis inversa. Esta es trasladada mediante bombeo a la planta concentradora distante 200 km desde la planta desaladora de la minera que está a orillas del océano Pacifico. En la instalación minera se encuentra ubicada en la cordillera de los Andes, y el agua se debe trasladar mediante un ducto de 200 km, para lo cual hay cuatro estaciones de bombeo cada una con 9 bombas similares, las que operan dependiendo de la demanda de agua de la minera. El equipo al cual la minera aporto los datos los datos corresponde a una de estas bombas que tiene las siguientes características:

Motor:

RPM	: 2986 rpm
Potencia	: 3500 Kw
Descanso	: tipo metalados

Bomba:

Tipo	: centrifuga
Cantidad de etapas	: 5
Descanso	: tipo metalados
Caudal	:1001 M3/hr

Figura 13: Bomba en estudio

Se trabajará con los datos de vibración temperatura, amperaje de motor y caudal. Los sensores de vibración de proximidad se encuentran instalados bajo el estandar ISO 7919 “Mechanical vibration -- Evaluation of machine vibration by measurements on rotating shafts”, y API 640 cuyo esquema de instalación se describe en la siguiente figura.

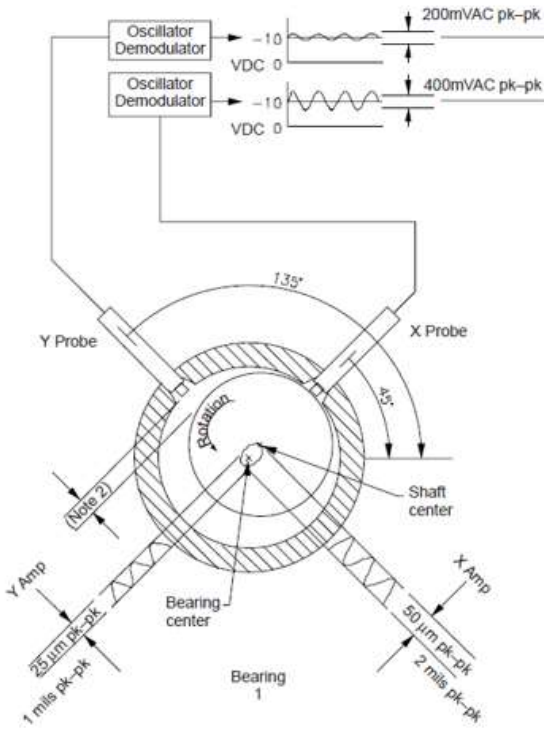


Figura 14: Especificación de instalación norma ISO 7919

3.2 Limpieza de datos

Del sistema DCS se obtienen data de variables de un período de 4 meses, desde el 11 de noviembre 2018 al 11 de marzo de 2018, que permitirá efectuar un sistema computacional de detector de anomalías. La data obtenida del sistema corresponde a las siguientes variables en formato (ver tabla 1):

Variable	Planilla Excel	Unidad	Cantidad registros
Amperaje de fase de motor	AMPERAJE	Amper (amp)	729554
Vibraciones proxímetros motor	PROXIMIDAD MOTOR	um (micra)	716512
Vibraciones proxímetros bomba	PROXIMIDAD BOMBA	um (micra)	767.647
Caudal	CAUDAL	m3/hr (metro cúbico hora)	279512
Temperatura descansos.	TEMPERATURAS EQUIPO (MOTOR + BOMBA).	°C (grados Celsius)	441197

Tabla 1: Características de la data entregada por empresa minera.

A continuación, se muestra el proceso de limpieza de datos (ver figura 15). El análisis de datos se realiza con el programa Python. Se observa que hay datos repetidos para un mismo tiempo, Nan y que los datos del proxímetro "x" no están tomados en el mismo instante de tiempo del proxímetro "y" estos se registran en el sistema DCS cada 10 segundos y corresponden a valor RMS pk – pk

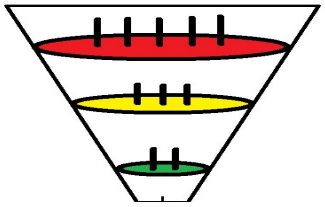


Figura 15: Filtrado de datos

La tabla 2 muestra la reducción de datos

	Registros iniciales	Eliminando NaN	Eliminando repetidos
Próximetro 1x	716511	461705	114056
Próximetro 1y	716511	716511	130324
Próximetro 2x	716511	372389	101682
Próximetro 2y	716511	438860	96617
Próximetro 3x	767647	733888	733888
Próximetro 3y	767647	767647	767647
Próximetro 4x	767647	729544	729544
Próximetro 4y	767647	746683	746683
Ampf1	279509	30211	30211
Ampf2	279509	27861	27861
Ampf3	279509	31952	31952
temp1	441997	323441	323441
temp2	441997	307385	307385
temp3	441997	441197	441197
temp4	441997	293469	293469
Caudal	279511	279511	279511
		variable con mayor cantidad datos	767647
		variable con menor cantidad datos	27861

Tabla 2: Ejemplo de limpieza de datos de vibración

Los datos repetidos corresponden a un error de extracción de datos entregados, ya que figuran en la base de datos dos o más valores de vibración para un mismo registro de tiempo. Como criterio se dejó el primer dato en el tiempo en estos casos. Antes de eliminar los datos repetidos en el tiempo, se eliminaron los datos sin registro de número (NaN) los que ocurren por errores del sistema de registro del DCS, o bien por probable error de comunicación.

Por ejemplo, en los datos correspondiente al proxímetro del motor lado libre en dirección x hay 254.806 registros sin datos numéricos (Nan) y eliminando los registros con duplicidad de datos para un mismo tiempo quedan 114056 datos para análisis.

Sincronización de datos

Para definir el tiempo de referencia para efectos de la sincronización, se usa los siguientes criterios.

- Adaptar el tiempo de la variable que contiene menos datos.
- El inicio del rango de tiempo de referencia, corresponde al tiempo de la variable que se inicia más tardíamente en su registro de datos.
- El final del rango de tiempo de referencia, corresponde al tiempo de la variable que primero termina su registro de datos.

Con los criterios anteriores se asegura que en el resto de las variables se van a encontrar registro de datos en el tiempo de referencia. La variable usada de referencia corresponde al de amperaje del motor de la fase 2 la cual tiene la menor cantidad de datos que son 27810 registros. Se tomó el criterio de usar un tiempo de referencia de la variable con menos datos, para obtener un menor error en la extracción de datos con variables interpoladas linealmente. En anexo 1 está el código Python para definir el tiempo de referencia.

3.3 Resultados de reducción de dimensionalidad de datos

Los sets de datos de 16 variables forman un hiperespacio abstracto. Considerando que se usa un criterio de evaluar el detector mediante visualización y a posterior con cada variable ver su comportamiento gráfico, se aplica una reducción de dimensionalidad a 2 dimensiones (2D) y a 3 dimensiones (3D). Los resultados con la reducción de dimensionalidad son los siguientes

3.3.1 PCA

Se aplicó a la data en estudio de 16 variables reducidas con este método a 2 y 3 dimensiones, las cuales se pueden graficar y apreciar visualmente (ver figura 16), mediante el código entregado en el anexo1.

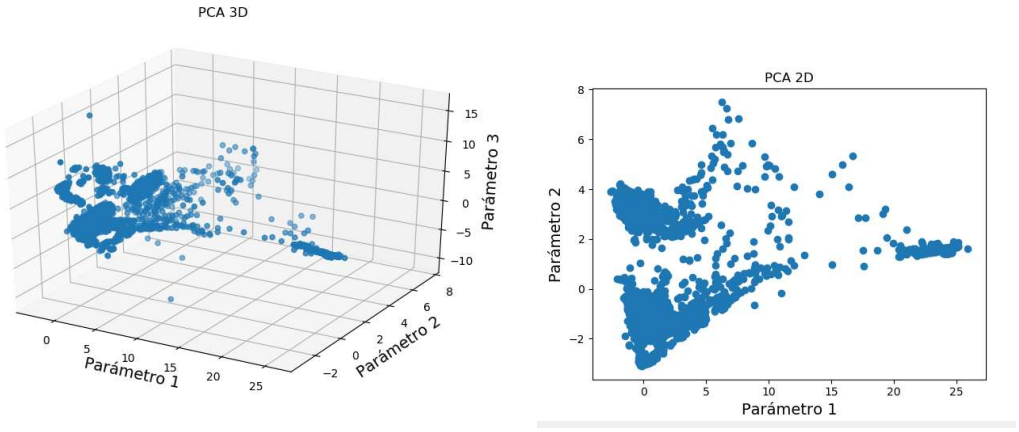


Figura 16: Cambio de variables funciones no lineales con PCA a 3D y 2D

Los tiempos de procesamiento, fueron los siguientes:

Tiempo de proceso PCA 3D: 0.085 segundos.

Tiempo de proceso PCA 2D: 2.16 segundos.

Al graficar la varianza acumulada para 15 componentes principales (ver figura 17) se observa que con las dos primeras componentes principales se obtiene una variancia acumulada de un 88,1% , que está por sobre el 85% que se consideró como aceptable.

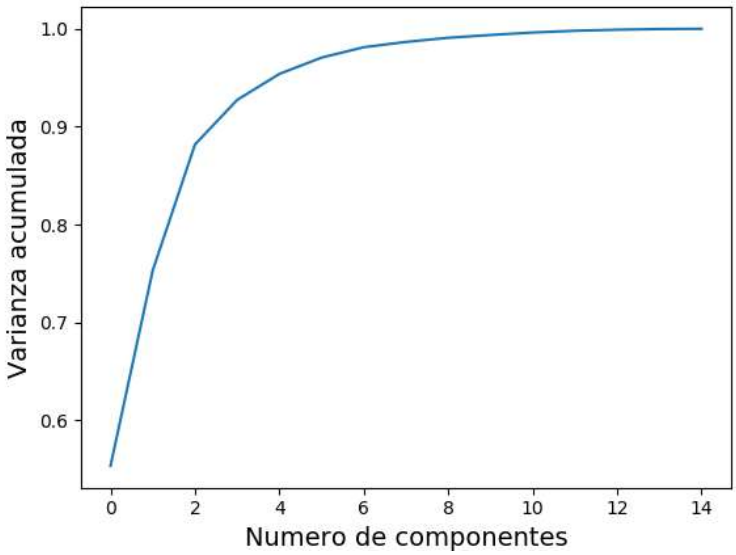


Figura 17: Varianza acumulada 15 componentes principales

3.3.2 Kernel PCA

Se aplicó a la data en estudio de 16 variables reducidas con este método a 2 y 3 dimensiones, las cuales se pueden graficar y apreciar visualmente (ver figura 17) , mediante el código entregado en el anexo.

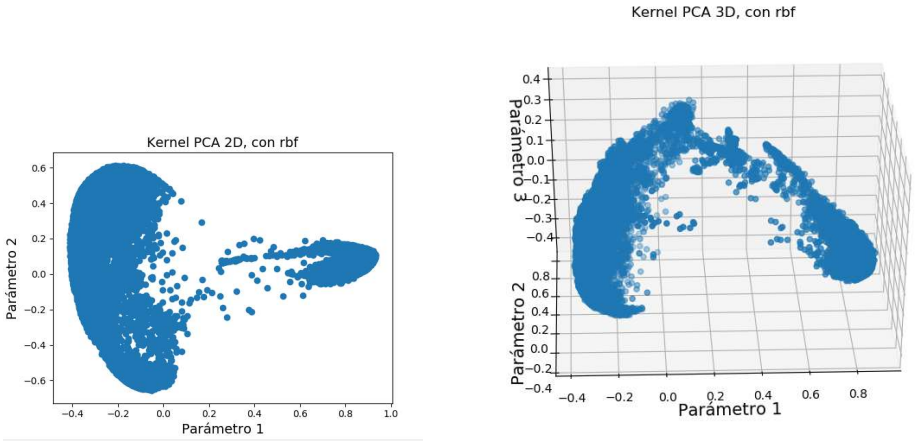


Figura 18: Cambio de variables funciones no lineales con Kernel PCA a 3D y 2D

El tiempo de procesamiento fueron los siguientes:

Tiempo de proceso Kernel PCA 2D, con rbf : 2996.41segundos, 22,99 minutos

Tiempo de proceso Kernel PCA 3D, con rbf : 1379.50 segundos, 17,37 Minutos

3.3.3 Factor Analysis

Se aplicó a la data en estudio de 16 variables, reducidas con este método a 2 y 3 variables, las cuales se pueden graficar y apreciar visualmente (ver figura 18), mediante el código entregado en el anexo.

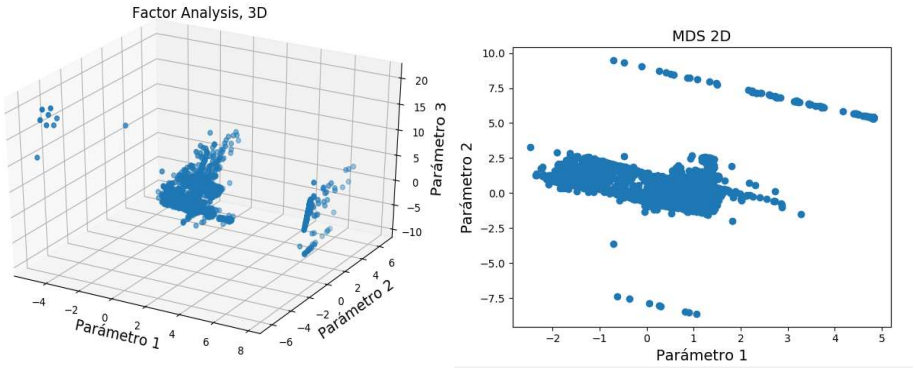


Figura 19: Cambio de variables funciones no lineales con Factor Analisis a 3D y 2D

Tiempo de proceso Factor Analysis, 2D : 44.96 segundos.

Tiempo de proceso Factor Analysis, 3D : 14.27 segundos.

MDS

Se aplicó a la data en estudio de 16 variables, reducidas con este método a 2 y 3 variables, las cuales se pueden graficar y apreciar visualmente (ver figura 19) mediante el código entregado en el anexo.

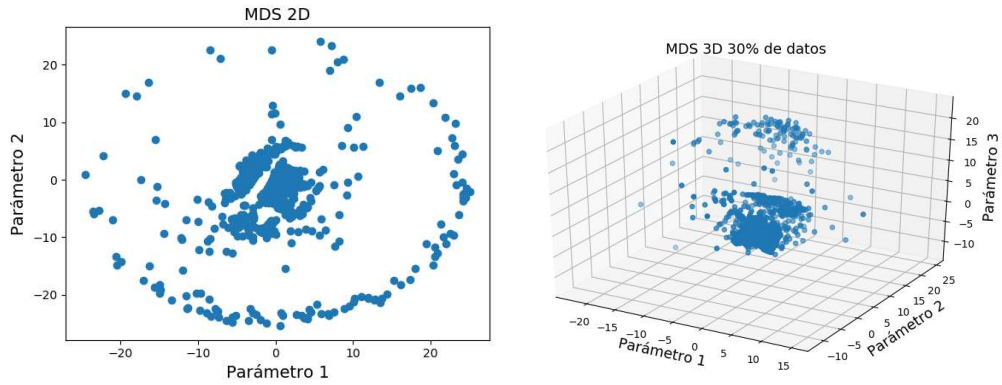


Figura 20: Cambio de variables funciones con MDS a 3D y 2D

Para la cantidad de datos el computador usado no es capaz de efectuar el proceso, arrojando un error de saturación de memoria, para poder procesar se tomó solo un 30% de la data, correspondiente 8358 registro de un total de 27860 registros.

Tiempo de proceso MDS, 2D, con 30 % de datos: 3249.64 segundos. (54 minutos)

Tiempo de proceso MDS, 3D, con 30% de datos : 3223.38 segundos. (53,7 minutos)

3.3.4 ISOMAP

Se aplicó a la data en estudio de 16 variables, reducidas con este método a 2 y 3 variables, las cuales se pueden graficar y apreciar visualmente (ver figura 20), mediante el código entregado en el anexo.

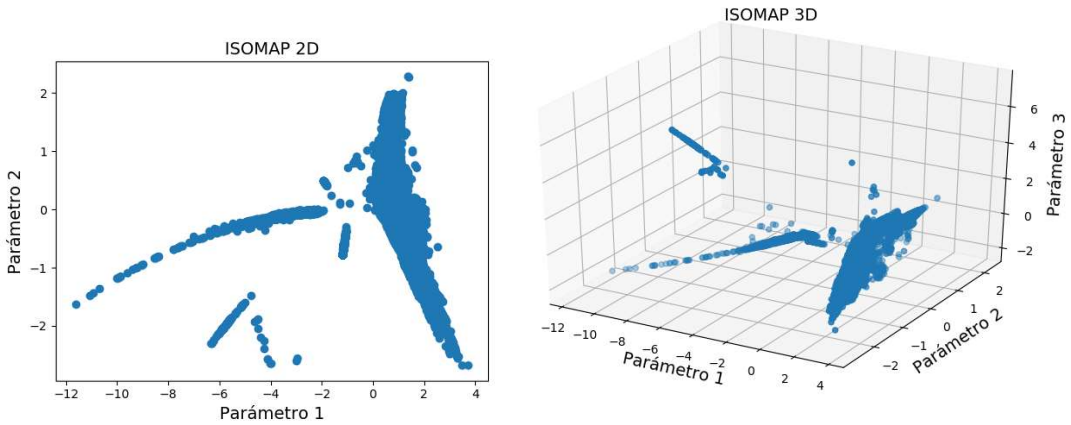


Figura 21: Cambio de variables funciones con ISOMAT a 3D y 2D

Tiempo de proceso ISOMAP 2D : 3013.96 segundos. 50,23 minutos

Tiempo de proceso ISOMAP 3D : 3351.97 segundos, 55,85 minutos

3.3.5 LLE

Se aplicó a la data en estudio de 16 variables, reducidas con este método a 2 y 3 variables, las cuales se pueden graficar y apreciar visualmente (ver figura 21), mediante el código entregado en el anexo.

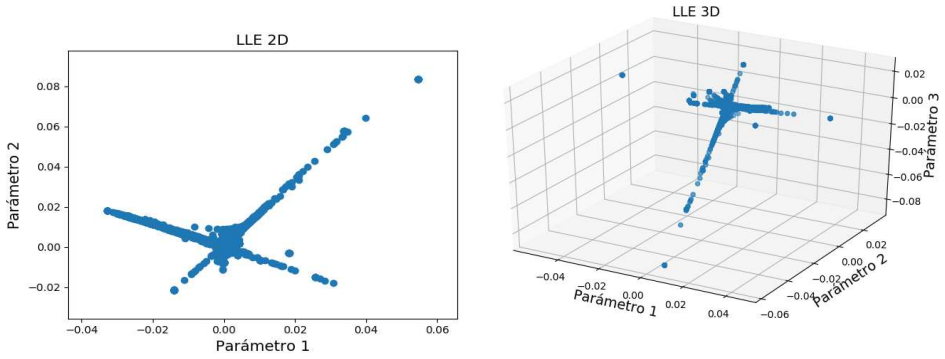


Figura 22: Reducción Componentes principales método LLE, de los 16 sensores de motobomba

Tiempo de proceso LLE 2D : 8.18 segundos.

Tiempo de proceso LLE 3D : 8.219 segundos.

Por los tiempos altos de procesamiento de los métodos del método Kernel-PCA y MDS, se utilizará para el análisis los siguientes métodos:

- PCA
- Factor Analysis
- LLE

3.4 Resultados del modelo predictivo detector de anormalidad

El procesamiento de los datos se realizó mediante programa en Python (ver anexo). Para una mejor visualización se realizó análisis con reducción a dos variables. Los siguientes son los resultados de los datos para entrenamiento de los modelos con los diferentes algoritmos de detección de anomalías, para cada método de reducción de variables antes descrito. En cada método se procesaron con diferentes valores de parámetros propios de cada método, siendo los siguientes valores de parámetros los que se logrón un mejor desempeño.

- Elliptic Envelope : contamination=0,022
- Isolation Forest : n_estimators=100, max_samples='auto', contamination=0,022
- LocalOutlierFactor : n_neighbors=20, contamination=0,022
- SVM : nu=0.02, kernel="rbf", gamma='auto'

A modo de ejemplo y para graficar la metodología de criterio de selección visual de los datos graficados se observa la determinación del parámetro " Contaminatión" en 0,04. En la graficas de más abajo (ver figura 22) se presentan 6 graficas cada una con diferentes valores de este parámetro, que van de 0,017 a 0,04 siendo este último valor el que se selecciona. Se observa en las gráficas,

una zona de concentración de datos y otra de datos dispersos. Con valores de "contamination" se obtiene que hay datos dispersos que son tomados como normales (azules), al aumentar este parámetro hasta llegar a un valor de 0,04 los datos dispersos considerados como normales disminuyen, pero aumenta una zona de alta concentración que son considerados anormales (puntos naranjos).

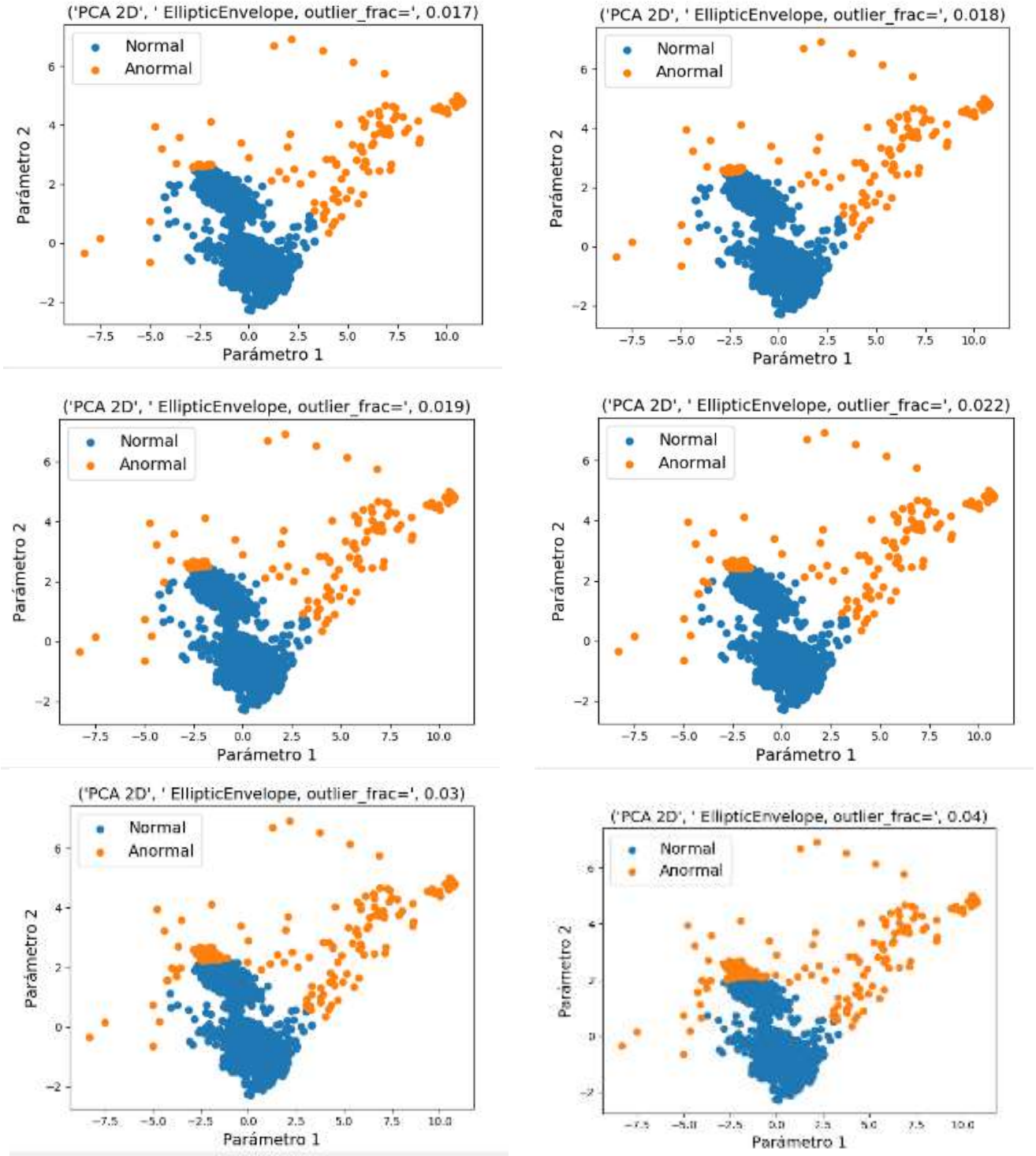


Figura 23: Selección del parámetro "contamination" en Elliptic Envelope, bajo un criterio de apreciación visual gráfico.

De similar manera se procede a ajustar cada parámetro y de modelo de detección de anomalías usados para los 3 métodos de reducción de variables a dos dimensiones, en la tabla 3 se grafican los resultados obtenidos para cada combinación modelo y reducción de variable, en total 12 alternativas de reducción de variables:

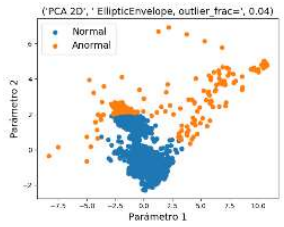
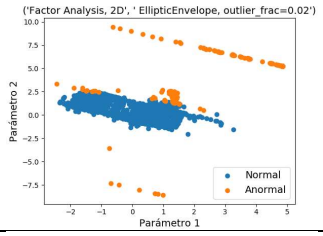
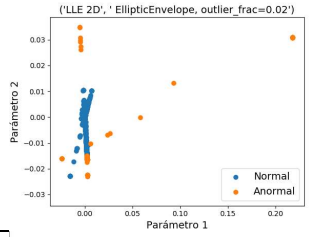
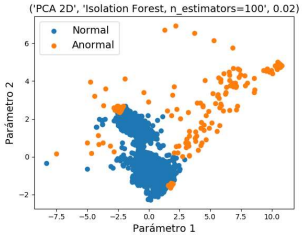
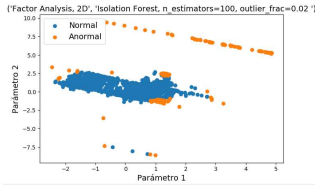
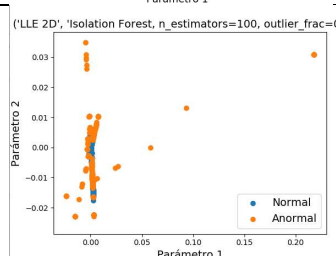
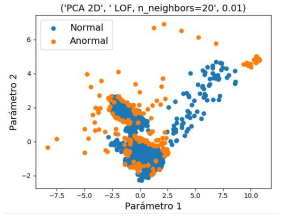
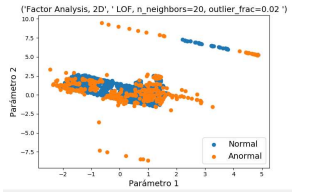
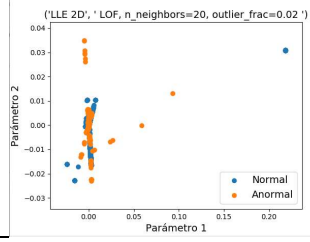
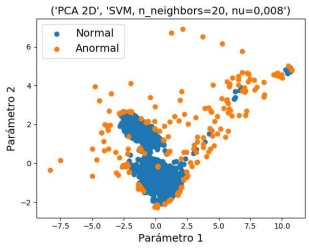
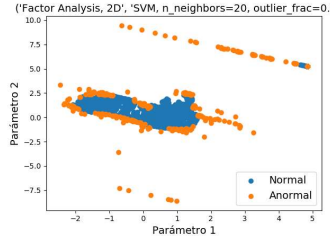
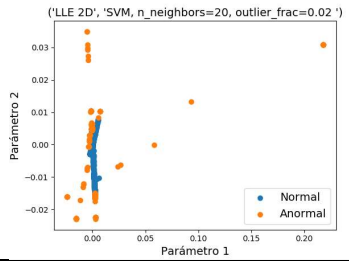
Modelo	• PCA 2D	• Factor Analysis 2D	• LLE 2D
Elliptic Envelope			
Isolation Forest			
LocalOutlier Factor			
SVM			

Tabla 3: Tabla comparativa de técnicas de detector de anormalidades usadas

En la tabla 3 anterior, cada grafica es la obtenida en base a diferentes pruebas de parámetros de setup para una separación optima de anómalos v/s normales. Se observa que el método que mejor logra separar corresponde al método de reducción de dimesionalidad es el PCA para los cuatro modelos de anormalidad. Se considera que los que están más agrupados son los normales y los dispersos son los anormales. Bajo este criterio, el modelo que es mejor detector de anormalidad es el Elliptic Envelopecon con el método de reducción de dimensionalidad PCA 2D, además, es el que posee un menor tiempo de procesamiento computacional.

Para efecto de visualizar en un gráfico los valores de cada variable o variables sincronizadas en el tiempo, tanto para los valores normales y anormales, se mantuvo en el tratamiento de los datos los

valores de tiempo de referencia de cada registro, de esta manera, se pueden graficar las variables físicas cronológicamente para efectos de validación, evaluando si tienen relación con el comportamiento lógico mecánico de cada variable, que se analiza más adelante.

3.5 Validación dela modelo usado

3.5.1.1 Validación con set de datos de Validación:

Con la división al azar del set de datos con un 80% para entrenamiento y 20% para validación se procede a aplicar el modelo de detección de anomalías, se observa la similitud con la detección de anormalidades del set de datos de entrenamiento.

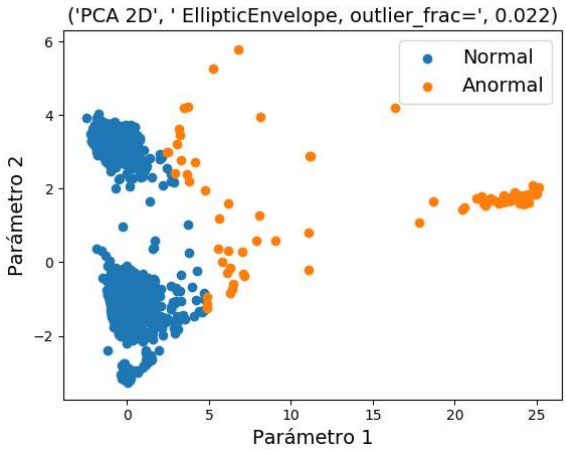


Figura 24: anormalidades del set de datos de validación

El modelo detecta, de un total de 5449 datos de validación, de los cuales 5326 datos normales y 123 datos anormales que representa un 2,2 % de anormalidad. Para el entrenamiento se usó un 80% de los datos que son 22288 datos, de los cuales 21797 son clasificados como normales y 491 como anormales lo que representa también un 2,2% de anormalidad

3.5.2 Grafica de orbita a partir de datos RMS

Una vez sincronizados los datos se grafica los desplazamientos RMS x-y en un descanso solo los datos normales. A modo de ejemplo para el descanso 1 se obtiene la siguiente gráfica:

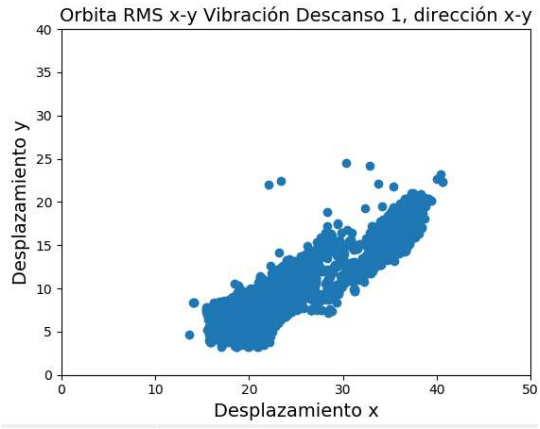


Figura 25: Grafica de descanso motor lado libre x-y de datos normales

Los sensores e desplazamientos x-y en cada descanso, están dispuesto a 90| de acuerdo a norma. Al graficar se observa, que la forma de este clusters es coincidente con la órbita obtenida por sistema de monitoreo espectral instalado en los equipos y que se muestra en la siguiente figura 24. Al girar en 45° la gráfica anterior, se observa la similitud entre ellas.

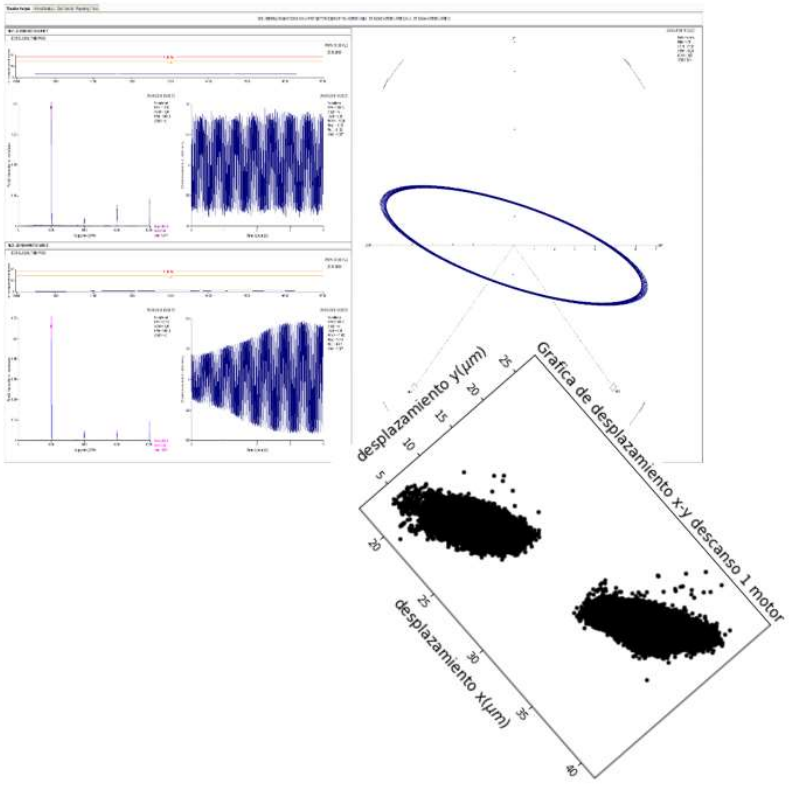


Figura 26: Grafica de proximetros x-y sincronizados y comparada con orbita de sistema de monitoreo on- line

Este resultado valida la aplicación de la técnica. porqué un espectro de vibración de motor en condiciones normales predomina la vibración a la frecuencia correspondiente a la velocidad de giro tal como se muestra en la siguiente figura 25:

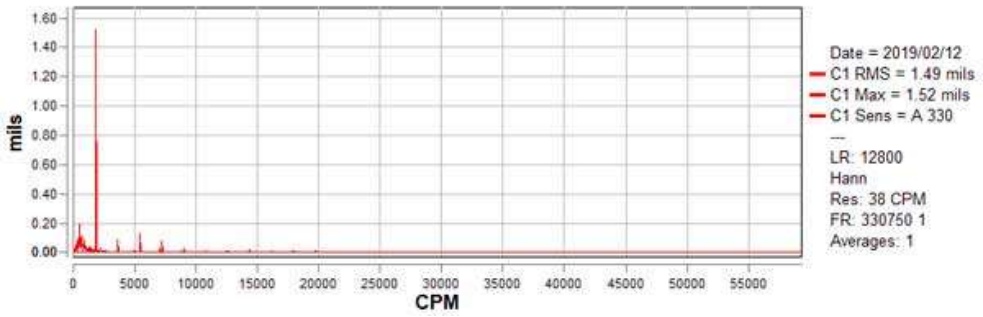


Figura 27: Espectro de vibraciones típico de frecuencia sin falla de un motor

El valor RMS será, en este caso normal, proporcional a la frecuencia fundamental del espectro, entonces si tenemos muchos valores RMS en condiciones normales (como lo son los datos de este estudio) para diferentes posiciones del centro del eje del motor y correspondientes a un mismo descanso y sí estos están sincronizados temporalmente, al graficar mucho de estos datos se formará una zona normal similar a la órbita tomada puntualmente en forma espectral.

Para los restantes descansos el proceso de validación es similar.

3.5.3 Comportamiento de la temperatura en oscilaciones diarias

En el desierto de Atacama, se observa lo cíclico de la temperatura ambiente en un día [4], está en un día puede variar fácilmente 30ºC , tal como lo muestra la siguiente grafica 26:

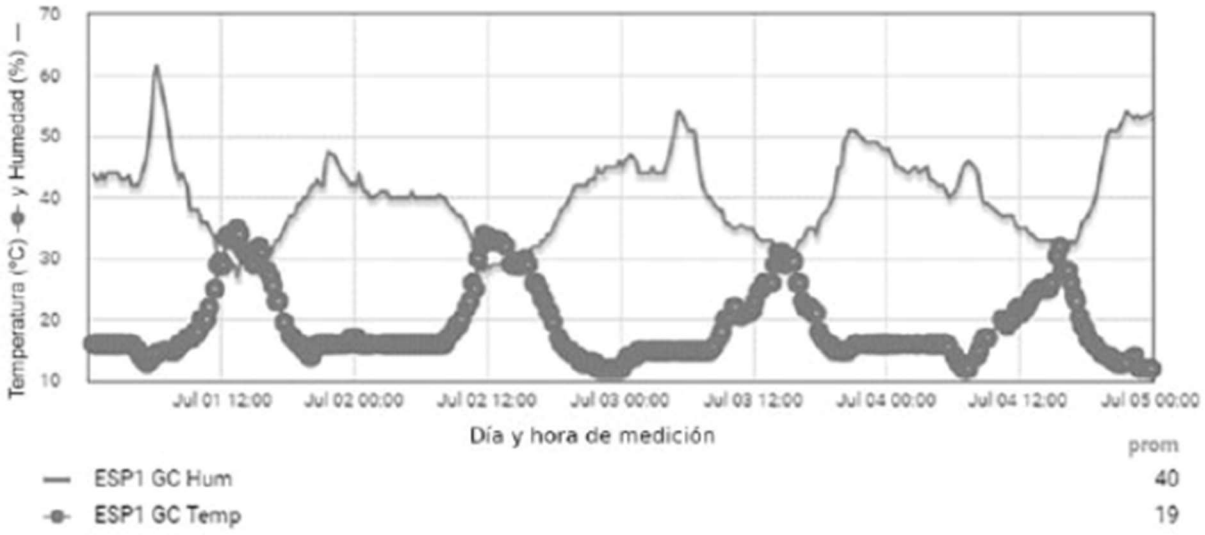


Figura 28: Variación de temperatura ene le día [4]

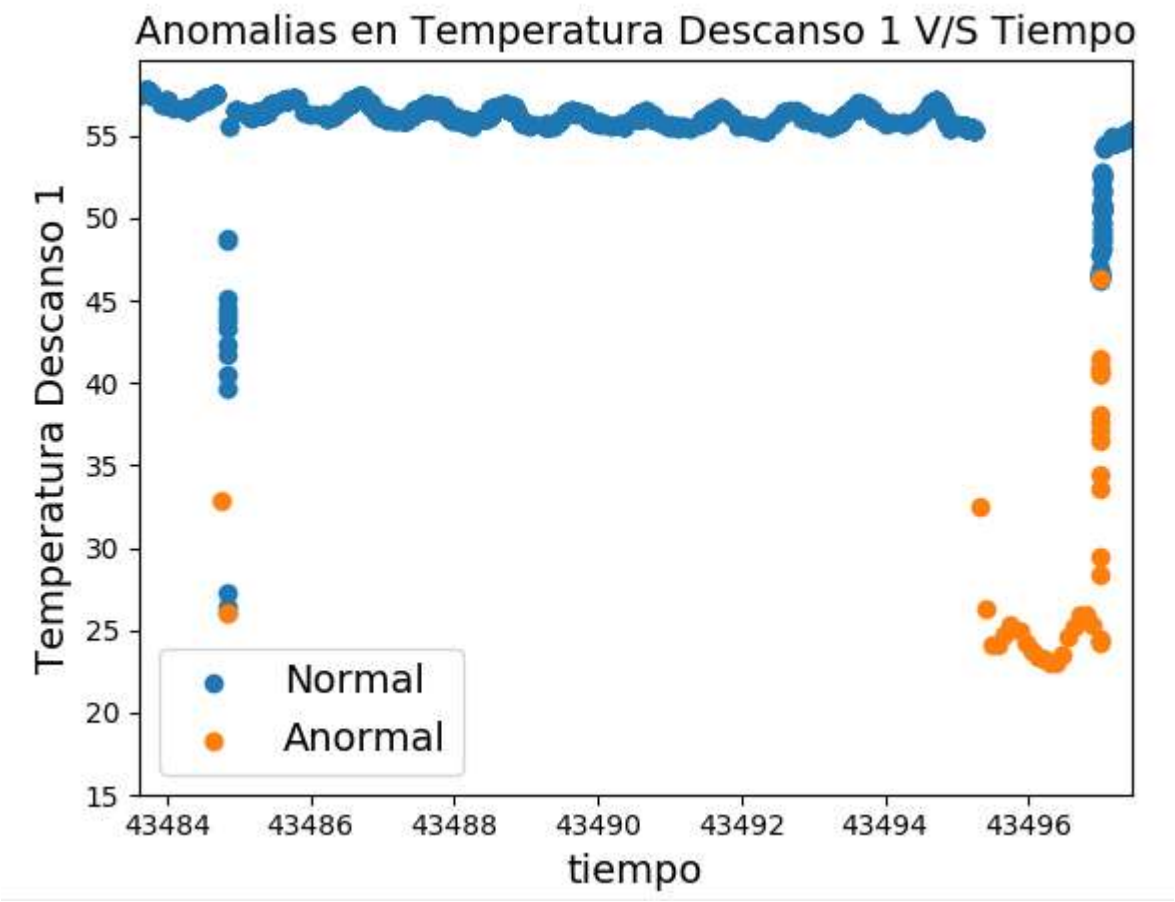


Figura 29: Fluctuaciones de Temperatura Descanso motor lado acople, diarios

El detector de anomalías evalúa estas oscilaciones como normales no así las detenciones, tal como se aprecia en la figura anterior correspondiente a la temperatura del descanso del motor lado bomba.

3.5.4 Verificación de datos de detección de anormalidad de datos físicos:

Con el set de datos de entrenamiento de vibración de la bomba y con el algoritmo de mejor comportamiento en cuando tiempos de procesamiento informático se obtiene los siguientes resultados:

Importante indicar que los datos de esta tesis provienen de planillas Excel en ella las fechas se cambiaron a números y así se importaron a Python. Excel almacena las fechas como números de serie secuenciales para que se puedan usar en cálculos. De manera predeterminada, la fecha 1 de enero de 1900 es el número de serie 1. Por otra parte, Python contabiliza los días desde el primero de enero de 1970. Por lo tanto, entre el contador Excel y el de Python hay 25568 días.

Las siguientes graficas corresponden a los datos detectados anómalos para cada variable física:

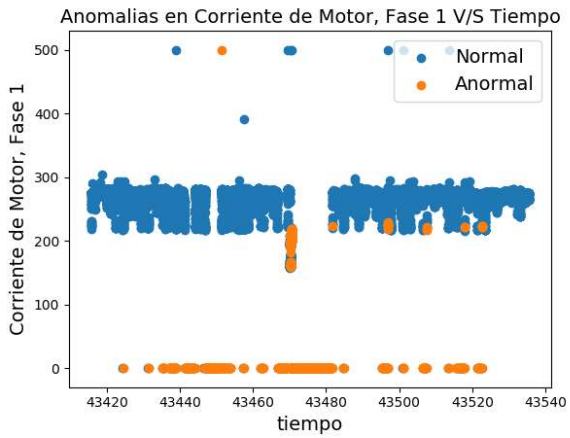


Figura 30: Anomalías en corriente de motor, fase 1 en el tempo

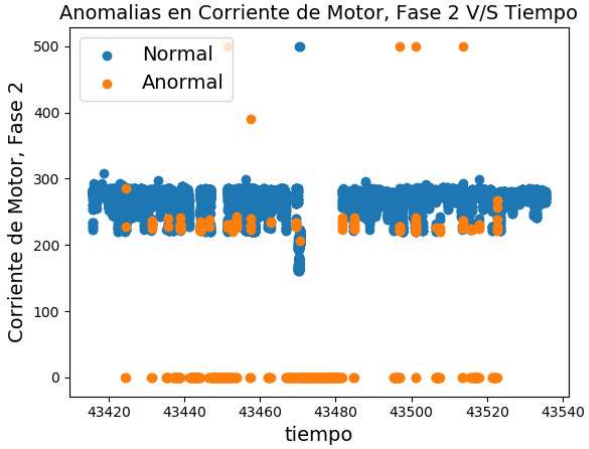


Figura 31: Anomalías en corriente de motor, fase 2 en el tempo

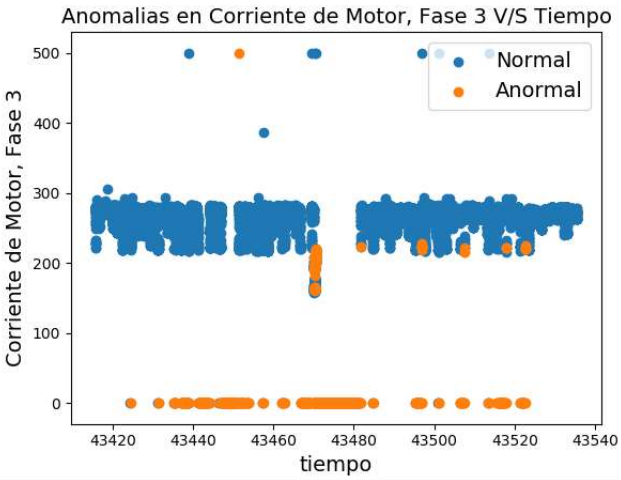


Figura 32: Anomalías en corriente de motor, fase 3 en el tempo

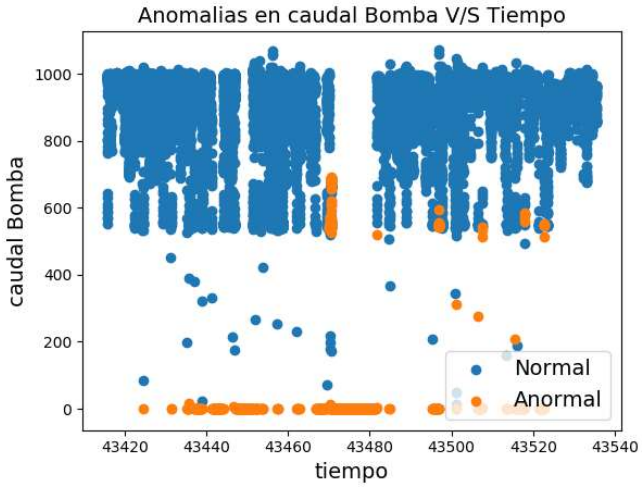


Figura 33: Anomalías en caudal de Bomba, en el tempo

En las gráficas anteriores se observan valores cercanos a "0" lo cuales corresponden a de detención del equipo y su transientes como anómalos y los valores de un estado de equilibrio operacional como normal

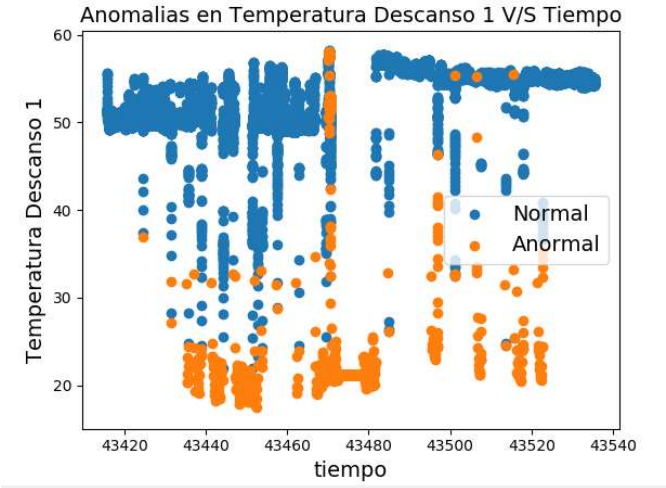


Figura 34: Anomalías en Temperatura Descanso 1 , en el tiempo

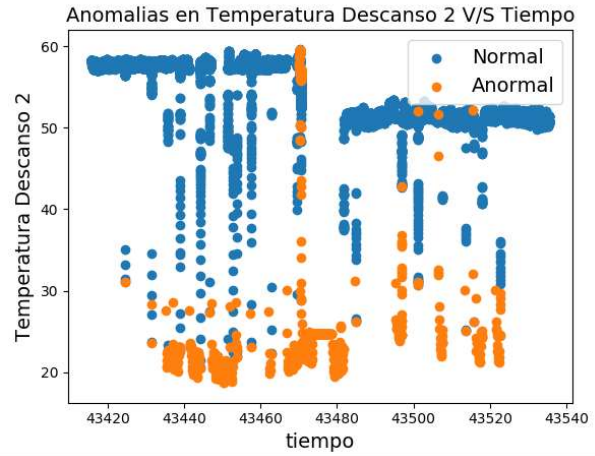


Figura 35: Anomalías en Temperatura Descanso 2, en el tiempo

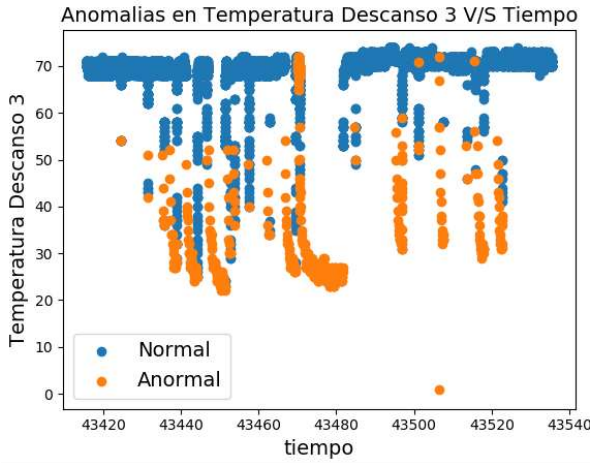


Figura 36: Anomalías en Temperatura Descanso 3, en el tiempo

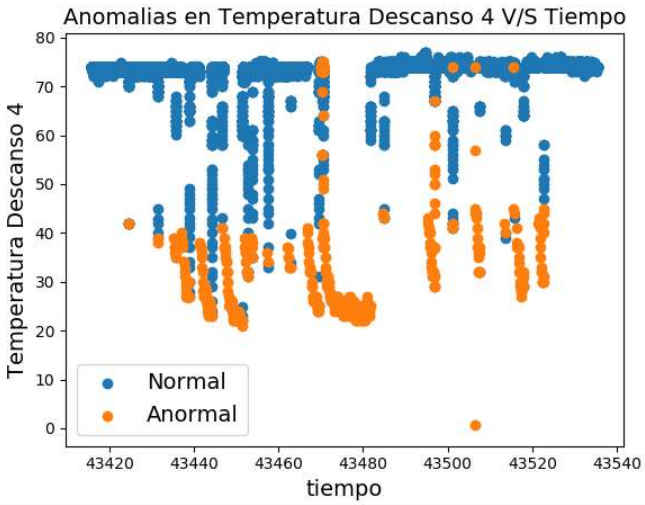


Figura 37: Anomalías en Temperatura Descanso 4, en el tiempo

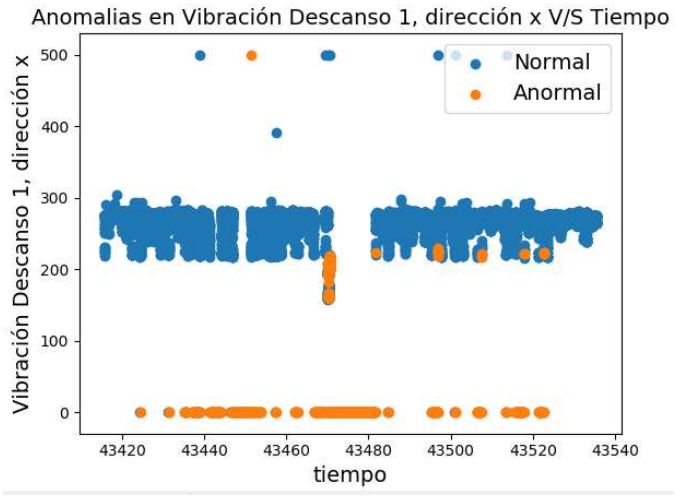


Figura 38: Anomalías en Vibración Descanso 1, Dirección x, en el tiempo

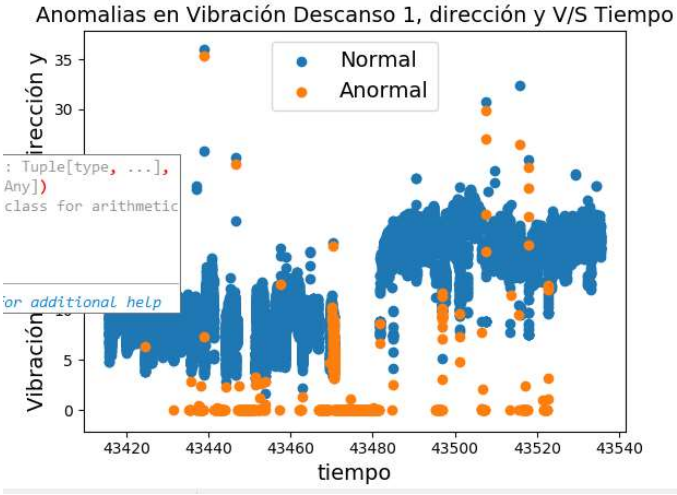


Figura 39: Anomalías en Vibración Descanso 1, Dirección y, en el tiempo

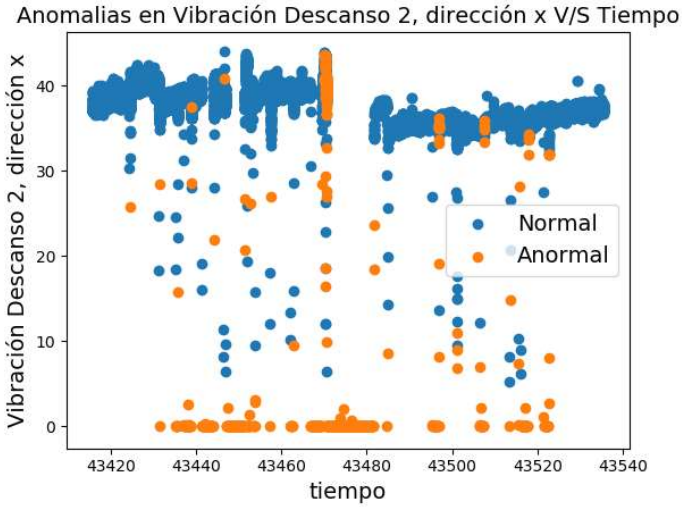


Figura 40: Anomalías en Vibración Descanso 2, Dirección x, en el tiempo

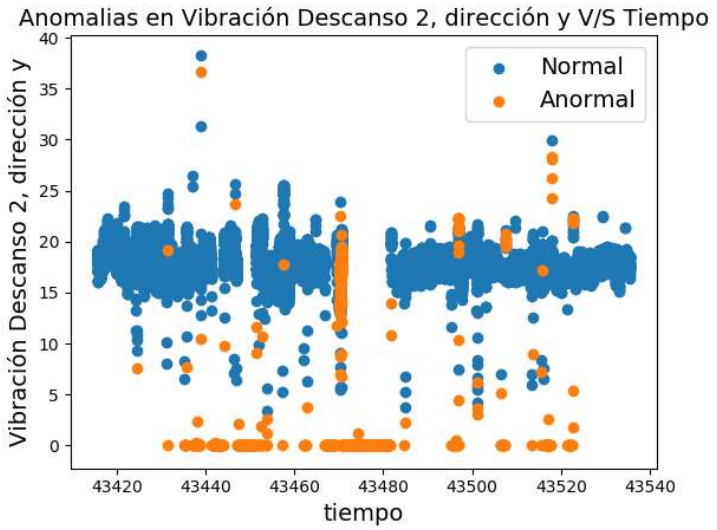


Figura 41: Anomalías en Vibración Descanso 2, Dirección y, en el tiempo

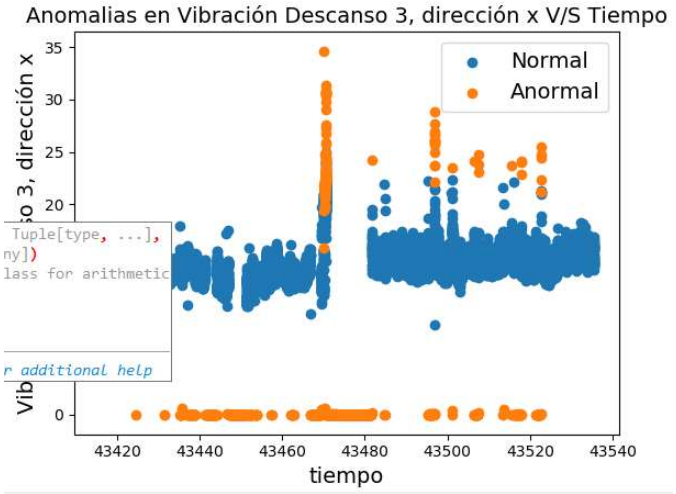


Figura 42: Anomalías en Vibración Descanso 3, Dirección x, en el tiempo

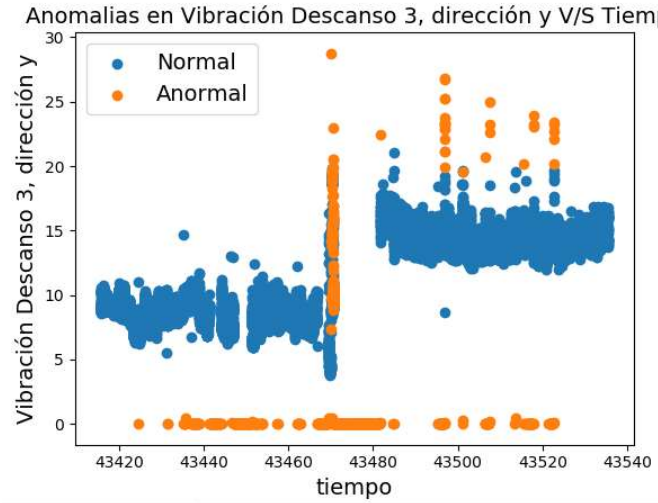
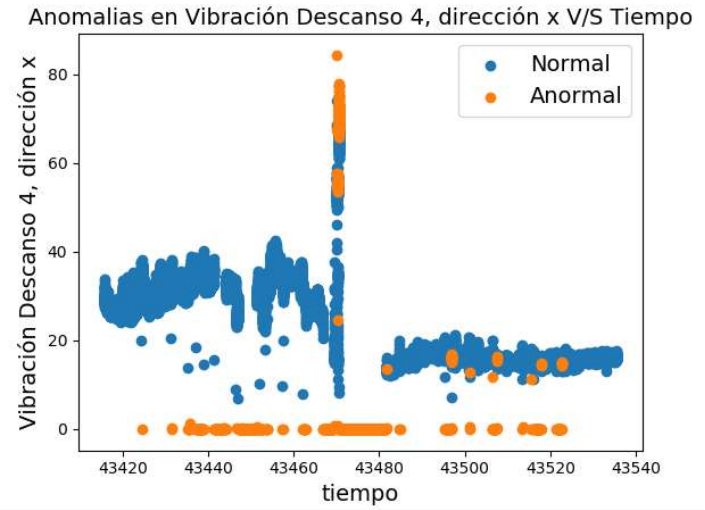


Figura 43: Anomalías en Vibración Descanso 3, Dirección y, en el tiempo



igura 44: Anomalías en Vibración Descanso 4, Dirección x, en el tiempo

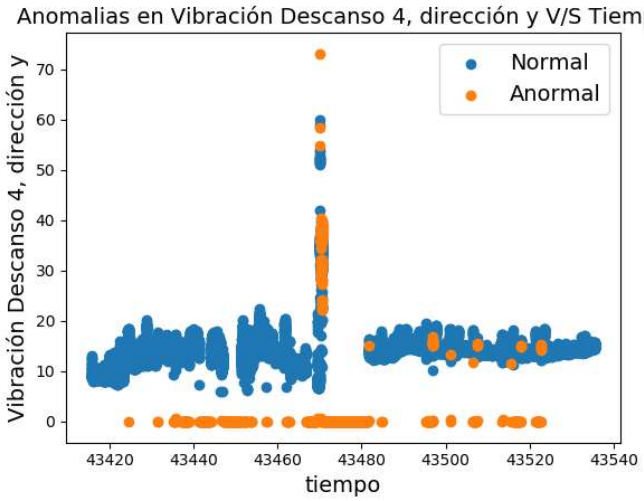


Figura 45: Anomalías en Vibración Descanso 4, Dirección y, en el tiempo

Se observa en las ultimas 4 gráficas una tendencia de aumento de vibración durante un tiempo de 3 horas aproximadamente. Se solicitó datos históricos de falla a la Minera y de la información entregada por la minera, revelan que efectivamente existió una falla en la fecha indicada. Esta falla ocurrió el día 05 de enero de 2019, aproximadamente de las 03:00 horas de la madrugada. La falla consistió en un desplazamiento repentino axial del rotor de la bomba, falla que se desarrolla en un tiempo de 3 horas. Esta situación fue registrada por el sistema de monitoreo on line que posee la Minera, pero que no fue alertada oportunamente, por no haber analistas disponibles en horario nocturno. La siguiente grafica demuestra este aumento de vibraciones desde el sistema de monitoreo on line que posee la Minera.

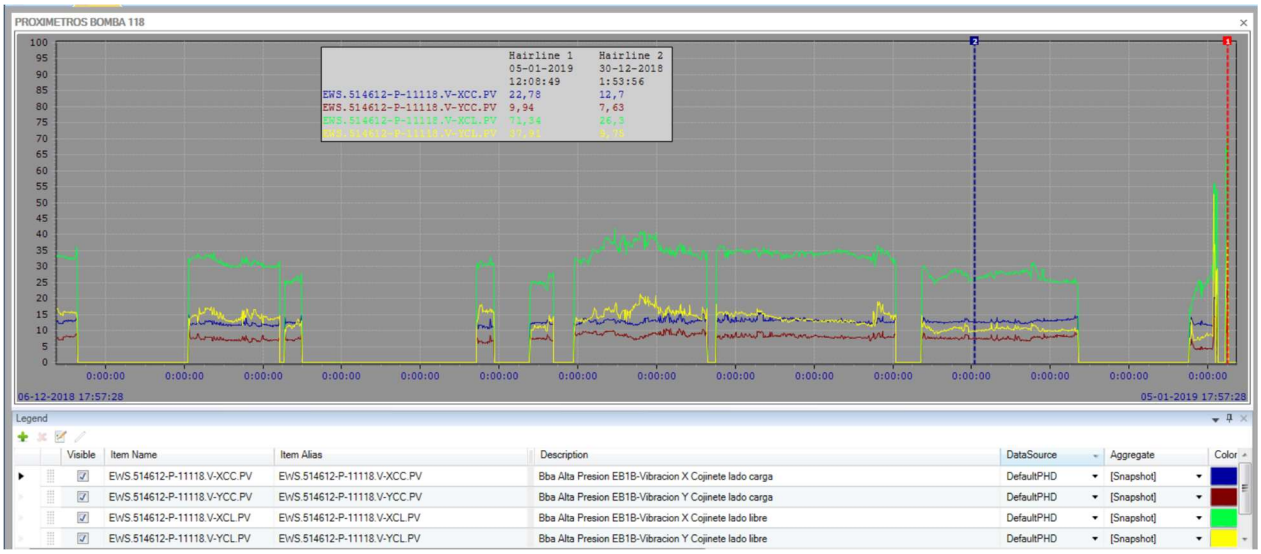


Figura 46: Tendencia de vibración en bomba donde revela la falla mediante el sistema on line de vibración que posee la Minera

la siguiente foto (figura 45) indica el descanso donde se produjo la falla y que genero una detención inesperada del sistema de bombeo.

Figura 47: Descanso de desplazamiento axial que fallo generando una falla de la bomba

Los antecedentes de esta falla permitieron optimizar el modelo ya que cuando se procesan los datos con las 16 variable, no se obtiene datos anormales en torno a la fecha de la falla cuando hay aumento de la tendencia de vibración en las gráficas de vibración de la bomba, tal como se observa en la siguiente gráfica (lado izquierdo) de figura 47.

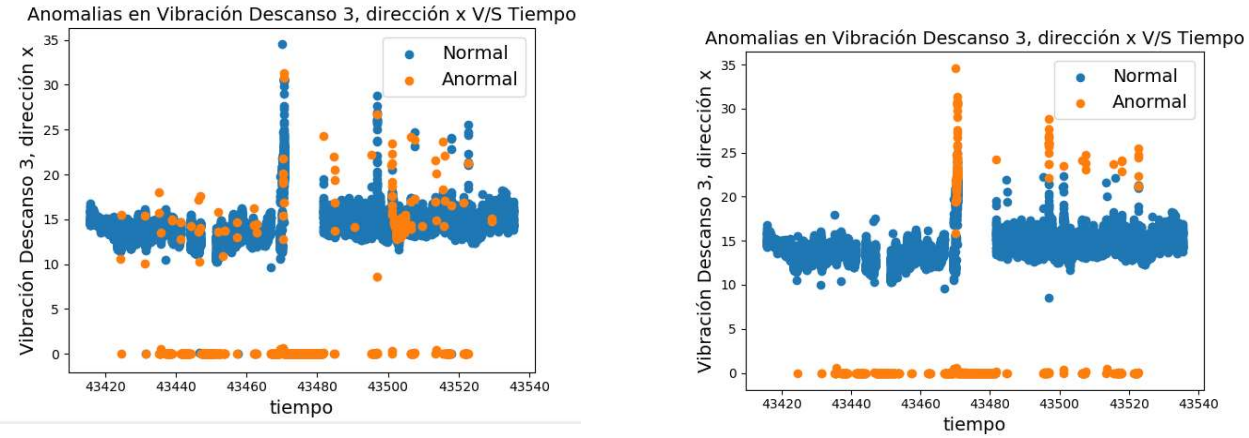


Figura 48: Detección de anomalía Descanso 3 con entrenamiento de 16 variables (izquierda) y entrenamiento con los datos de vibración de la bomba (derecha)

Sin embargo, si se procesan solo con los datos de vibración de la bomba, el aumento de vibraciones si es detectado como una anormalidad, tal como se aprecia en la figura de la derecha de la gráfica 47.

4 CONCLUSIONES:

Del procesamiento de datos realizado en este estudio, se puede concluir lo siguiente:

- Si bien los datos RMS no son de mucha calidad de información frente a un espectro de frecuencia, para determinar el estado en una máquina, sin embargo, si se dispone de una gran cantidad de datos, los valores RMS, aportaran suficiente información útil para detectar anormalidad de operación, pudiendo incluso construir la forma de la órbita en estado normal. Lo que es otro criterio para predecir un estado normal de operacional de uno anormal del equipo.
- Lo anterior es relevante en cuanto a costo de la información ya que el equipamiento para obtener un valor RMS es muy inferior al equipamiento para obtener un valor espectral. Si a lo anterior se suma el hecho que los valores DCS están ya instalados en muchas plantas entonces el costo solo se limita a desarrollar algoritmos de detector de anormalidades, para datos como los usados en esta investigación.
- Si bien se trabajó con datos de vibración y operacionales de temperatura, caudal y amperaje. Estos últimos no son necesarios para entrenar un modelo de anormalidad, siendo suficiente los datos de vibraciones, con lo cual se reduce la cantidad de sensores para estos fines.
- La validación de los resultados respecto de la variación diaria de temperatura es correcta ya que el modelo la considera una normalidad.
- El modelo desarrollado permitirá aplicar un sistema de supervisión que alerte anormalidades y mediante avisos vía email por ejemplo, alertar a operadores día y noche todos los días del año y de esta manera evitar fallas catastróficas, como las ocurridas durante la captura de estos datos, los cuales fueron aportados por la Minera para este estudio al tiempo después.

5 BIBLIOGRAFIA

1.- David Verstraete, Andrés Ferrada, Enrique López Droguett, Viviana Meruane, Mohammad Modarres, “Unsupervised deep generative adversarial based methodology for automatic fault detection,” Entropy, 2018.

2.- https://www.mch.cl/informes-tecnicos/mantenimiento-mineria-miras-mayor-rendimiento/

3.- Jorge Marin “Manual curso de Ultrasonido Transportado nivel I, bajo la norma ISO 18436", 2019.

4.- S.Farokhzad, H.Ahmadi, A.Jaefari "Artificial Neural Network Based Classification of Faults in Centrifugal Water Pump ", 2012

5.- M. Saberi , A. Azadeh , A. Nourmohammadzadeh and P. Pazhoheshfar "Comparing performance and robustness of SVM and ANN for fault diagnosis in a centrifugal pump", 2011

6.- https://sebastianraschka.com/Articles/2014_kernel_pca.html

7.- “Factor Analysis: A Short Introduction, Part 1” Maike Rahn, PhD

8.- http://www.analytictech.com/networks/mds.htm

9.- https://www.deeplearningitalia.com/metodos-para-la-reduccion-de-la-dimensionalidad-manifold-based-el-caso-isomap-2/

10.- Universidad de Ciénega. “Clasificación inteligente de datos”, autor: Alejandro Rodriguez Pelayo, tesis, 2017.

11.- https://es.qwe.wiki/wiki/Nonlinear_dimensionality_reduction

12.- https://en.wikipedia.org/wiki/Linear_discriminant_analysis

13.- https://numpy.org/doc/stable/reference/generated/numpy.interp.html

14.- https://www.redalyc.org/jatsRepo/339/33960285003/html/index.html

15.- https://www.guiaminera.cl/el-perfil-de-los-nuevos-tecnicos-mineros-en-chile-analiticos-sociables-y-autonomos/

16.- https://www.aprendemachinelearning.com/comprende-principal-component-analysis/

ANEXO 1

Datos del computador donde se procesó la información:

Este código operó en un PC cargado con Windows 10 y que tiene las siguientes características:

Procesador:	Intel(R) Core(TM) i5-8300H CPU @ 2.30GHz 2.30 GHz
Memoria instalada (RAM):	8,00 GB (7,86 GB utilizable)
Tipo de sistema:	Sistema operativo de 64 bits, procesador x64
Lápiz y entrada táctil:	La entrada táctil o manuscrita no está disponible para esta pantalla

Código python:

```
@author: Jorge
"""
#%% Import the libraries.
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.model_selection import train_test_split
import collections
from sklearn.cluster import DBSCAN
from scipy.interpolate import InterpolatedUnivariateSpline
import datetime, time
from datetime import datetime
from datetime import timedelta

from sklearn.preprocessing import StandardScaler
```

```
from sklearn.decomposition import PCA, KernelPCA
from sklearn.decomposition import FactorAnalysis
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.manifold import MDS
from sklearn.manifold import Isomap
from sklearn.manifold import LocallyLinearEmbedding
from numpy import savetxt,loadtxt
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA
from sklearn.ensemble import RandomForestClassifier
#from RandomForestClassifierklearn.metrics import confusion_matrix
#from sklearn.metrics import accuracy_score
from sklearn import datasets
from sklearn import preprocessing
from sklearn.neighbors import KNeighborsClassifier
from sklearn.covariance import EllipticEnvelope
from sklearn.ensemble import IsolationForest
from sklearn.neighbors import LocalOutlierFactor
from sklearn import svm
from sklearn.metrics import precision_score, accuracy_score, recall_score, f1_score,
confusion_matrix
from sklearn.cluster import KMeans
from sklearn.cluster import MeanShift
from timeit import timeit
from time import time
from numpy.lib import recfunctions as rfn
import scipy.io as sio
from mpl_toolkits.mplot3d import Axes3D
import math
from sklearn.decomposition import PCA, KernelPCA
from sklearn import datasets
from sklearn import preprocessing
```

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
import statsmodels.api as sm
#import statsmodels.formularmula.api as smf
from sklearn import preprocessing
from sklearn.preprocessing import MinMaxScaler
#%%              LECTURA Y LIMPIEZA DE DATOS
#           CORRIENTE ELECTRICA MOTOR
#            Datos de corriente electrica
#amp_motor=           pd.read_csv(r'D:\Dropbox\1           Jorge\Magister           Uchile\TESIS JM\118\Data118\CORRIENTES_3F1.csv',header=0,delimiter=',',dtype='float64')
amp_motor=           pd.read_excel(r'D:Dropbox\1           Jorge\Magister           Uchile\TESIS JM\118\Data118\CORRIENTES 3F.xlsx')
#%%

amp_f1=amp_motor.iloc[:,0:2] # datos del sensor amperaje fase 1 motor
amp_f2=amp_motor.iloc[:,2:4] # datos del sensor amperaje fase 2 motor
amp_f3=amp_motor.iloc[:,4:6] # datos del sensor amperaje fase 3 motor
#%%
# Eliminando Nam
amp_f1=amp_f1.dropna()
amp_f2=amp_f2.dropna()
amp_f3=amp_f3.dropna()
  eliminando dupliados en el tiempo
amp_f1= amp_f1.drop_duplicates(subset='tiempo_F1',keep='first')
amp_f2= amp_f2.drop_duplicates(subset='tiempo_F2',keep='first')
amp_f3= amp_f3.drop_duplicates(subset='tiempo_F3',keep='first')
#%%
#            separando tiempo y amp por cada fase
amp_f1_t=amp_f1.iloc[:, 0]
amp_f1_amp=amp_f1.iloc[:, 1]
amp_f2_t=amp_f2.iloc[:, 0]
```

```
amp_f2_amp=amp_f2.iloc[:, 1]
amp_f3_t=amp_f3.iloc[:, 0]
amp_f3_amp=amp_f3.iloc[:, 1]
#%%   Grafica de fases
#amp_f1_t=amp_f1_t.strptime('%B %d, %Y, %r')
# >>> t = numpy.datetime64('2012-06-30T20:00:00.000000000-0400')
amp_f1_t2=np.datetime_as_string(amp_f1_t, unit='D')
plt.plot(amp_f1_t2, amp_f1_amp)
#plt.plot(amp_f1_t, amp_f1_amp, 'b.',amp_f2_t, amp_f2_amp, 'rd.', amp_f3_t, amp_f3_amp, 'g.')
#plt.plot(amp_f2_t, amp_f2_amp,'b.',amp_f2_t, amp_f2_amp)
#%%xmax= max(tx)
#xmin= min(tx)
#ymax=max(dx)
#ymin=min(dx)
#plt.xlabel('Tiempo (s)', fontsize=14)
#plt.ylabel('desplazamiento ' "($µm$)", fontsize=14)
plt.title('amperaje fase 1', fontsize=14)
#plt.xlim(xmin,xmax)
#plt.ylim(ymin,ymax)
plt.show()
#%%         TEMPERATURA
#          Temperaturas descansos
temp_motorbba=          pd.read_csv(r'D:Dropbox\1          Jorge\Magister          Uchile\TESIS
JM\118\Data118\T118.csv',header=0,delimiter=',',dtype='float64')
#%%
temp1=temp_motorbba.iloc[:,0:2] # datos de temp descanso motor lado libre
temp2=temp_motorbba.iloc[:,2:4] # datos de temp descanso motor lado acople
temp3=temp_motorbba.iloc[:,4:6] # datos de temp descanso bomba lado acople
temp4=temp_motorbba.iloc[:,6:8] # datos de temp descanso bomba lado libre
# Eliminando Nam
temp1=temp1.dropna()
```

```
temp2=temp2.dropna()
temp3=temp3.dropna()
temp4=temp4.dropna()
#  eliminando dupliados en el tiempo
temp1= temp1.drop_duplicates(subset='tT_1',keep='first')
temp2= temp2.drop_duplicates(subset='tT_2',keep='first')
temp3= temp3.drop_duplicates(subset='tT_3',keep='first')
temp4= temp4.drop_duplicates(subset='tT_4',keep='first')
#%%   separando tiempo y temp por cada descanso
temp1_t=temp1.iloc[:, 0]
temp1_T=temp1.iloc[:, 1]
temp2_t=temp2.iloc[:, 0]
temp2_T=temp2.iloc[:, 1]
temp3_t=temp3.iloc[:, 0]
temp3_T=temp3.iloc[:, 1]
temp4_t=temp4.iloc[:, 0]
temp4_T=temp4.iloc[:, 1]
#%%       CAUDAL
#         caudal de la bomba
caudal_motorbba= pd.read_csv(r'D:\Dropbox\1 Jorge\Magister Uchile\TESIS JM\118\Data118\caudal_118.csv',header=0,delimiter=',',dtype='float64')
#   Eliminando Nam
caudal_motorbba=caudal_motorbba.dropna()

#  eliminando dupliados en el tiempo
caudal_motorbba= caudal_motorbba.drop_duplicates(subset='t_Q',keep='first')
caudal_motorbba_t=caudal_motorbba.iloc[:, 0]
caudal_motorbba_Q=caudal_motorbba.iloc[:, 1]
#%%       VIBRACION
#         Extracción de matriz de vibración motor de bomba 118
```

```
vib_motor= pd.read_csv(r'D:\Dropbox\1 Jorge\Magister Uchile\TESIS JM\118\Data118\Proxi_motor_N.csv',header=0,delimiter=',',dtype='float64')
v1x=vib_motor.iloc[:,0:2] # datos del sensor vibraciones descanso 1 dirección x
v1y=vib_motor.iloc[:,2:4] # datos del sensor vibraciones descanso 1 dirección y
v2x=vib_motor.iloc[:,4:6] # datos del sensor vibraciones descanso 2 dirección x
v2y=vib_motor.iloc[:,6:8] # datos del sensor vibraciones descanso 2 dirección y
#%%  Extracción de matriz de vibración de bomba 118
vib_bba= pd.read_csv(r'D:\Dropbox\1 Jorge\Magister Uchile\TESIS JM\118\Data118\VB118.csv',header=0,delimiter=',',dtype='float64')
v3x=vib_bba.iloc[:,2:4] # datos del sensor vibraciones descanso 3 dirección x
v3y=vib_bba.iloc[:,0:2] # datos del sensor vibraciones descanso 3 dirección y
v4x=vib_bba.iloc[:,6:8] # datos del sensor vibraciones descanso 4 dirección x
v4y=vib_bba.iloc[:,4:6] # datos del sensor vibraciones descanso 4 dirección y
#%%  eliminando datos nam
v1x=v1x.dropna()
v1y=v1y.dropna()
v2x=v2x.dropna()
v2y=v2y.dropna()
v3x=v3x.dropna()
v3y=v3y.dropna()
v4x=v4x.dropna()
v4y=v4y.dropna()



#%%  eliminando datos duplicados en el tiempo


v1x = v1x.drop_duplicates(subset='time_XMC',keep='first')
v1y = v1y.drop_duplicates(subset='time_YMC',keep='first')
v2x = v2x.drop_duplicates(subset='time_XML',keep='first')
v2y = v2y.drop_duplicates(subset='time_YML',keep='first')
v3x = v3x.drop_duplicates(subset='time_XCC',keep='first')
```

```
v3y = v3y.drop_duplicates(subset='time_YCC',keep='first')
v4x = v4x.drop_duplicates(subset='time_XCL',keep='first')
v4y = v4y.drop_duplicates(subset='time_YCL',keep='first')
#%%
v1x_t=v1x.iloc[:, 0]
v1x_V=v1x.iloc[:, 1]
v2x_t=v2x.iloc[:, 0]
v2x_V=v2x.iloc[:, 1]
v3x_t=v3x.iloc[:, 0]
v3x_V=v3x.iloc[:, 1]
v4x_t=v4x.iloc[:, 0]
v4x_V=v4x.iloc[:, 1]
v1y_t=v1y.iloc[:, 0]
v1y_V=v1y.iloc[:, 1]
v2y_t=v2y.iloc[:, 0]
v2y_V=v2y.iloc[:, 1]
v3y_t=v3y.iloc[:, 0]
v3y_V=v3y.iloc[:, 1]
v4y_t=v4y.iloc[:, 0]
v4y_V=v4y.iloc[:, 1]
#%%        ETAPA: SICRONIZANDO DATOS EN EL TIEMPO
#Determinando cual data tiene menor rango de tiempo, para tomar como referencia de sincronización
 max_amp_f1_t, min_amp_f1_t= max(amp_f1_t),min (amp_f1_t)
max_amp_f2_t, min_amp_f2_t= max(amp_f2_t),min (amp_f2_t)
max_amp_f3_t, min_amp_f3_t= max(amp_f3_t),min (amp_f3_t)
max_caudal_motorbba , mincaudal_motorbba= max(caudal_motorbba_t), min(caudal_motorbba_t)
max_temp1_t, min_temp1_t= max(temp1_t),min (temp1_t)
max_temp2_t, min_temp2_t= max(temp2_t),min (temp2_t)
max_temp3_t, min_temp3_t= max(temp3_t),min (temp3_t)
max_temp4_t, min_temp4_t= max(temp4_t),min (temp4_t)
```

```
max_v1x_t , min_v1x_t = max(v1x_t),min (v1x_t)
max_v2x_t , min_v2x_t = max(v2x_t),min (v2x_t)
max_v3x_t , min_v3x_t = max(v3x_t),min (v3x_t)
max_v4x_t , min_v4x_t = max(v4x_t),min (v4x_t)
max_v1y_t , min_v1y_t = max(v1y_t),min (v1y_t)
max_v2y_t , min_v2y_t = max(v2y_t),min (v2y_t)
max_v3y_t , min_v3y_t = max(v3y_t),min (v3y_t)
max_v4y_t , min_v4y_t = max(v4y_t),min (v4y_t)
#%%              TIEMPO REFERENCIA
amp_f2_t=pd.DataFrame(amp_f2_t)
t_ref = amp_f2_t[amp_f2_t['Tiempo_F2'] <= 43535.59027777]
t_ref = amp_f2_t[amp_f2_t['Tiempo_F2']  >= 43415.59647]
#%%
t_ref=t_ref.iloc [:, 0]#  convertir DataFrame a serie
#%%
t_ref_max  = max(t_ref)
t_ref_min = min(t_ref)
#%%          ETAPA: SICRONIZANDO DATOS EN EL TIEMPO
#   Interpolación
int_amp_f1 = np.interp(t_ref,amp_f1_t, amp_f1_amp)
#%%int_amp_f2 = np.interp(t_ref,amp_f2_t, amp_f2_amp)
#int_amp_f2=amp_f2_amp   #  variable correspondiente a la referencia de tiempo
int_amp_f3 = np.interp(t_ref,amp_f3_t, amp_f3_amp)
int_caudal_motorbba = np.interp(t_ref,caudal_motorbba_t, caudal_motorbba_Q)
int_temp1 = np.interp(t_ref,temp1_t, temp1_T)
int_temp2 = np.interp(t_ref,temp2_t, temp2_T)
int_temp3 = np.interp(t_ref,temp3_t, temp3_T)
int_temp4 = np.interp(t_ref,temp4_t, temp4_T)
int_v1x = np.interp(t_ref,v1x_t, v1x_V)
int_v1y = np.interp(t_ref,v1y_t, v1y_V)
int_v2x = np.interp(t_ref,v2x_t, v2x_V)
```

```
int_v2y = np.interp(t_ref,v2y_t, v2y_V)
int_v3x = np.interp(t_ref,v3x_t, v3x_V)
int_v3y = np.interp(t_ref,v3y_t, v3y_V)
int_v4x = np.interp(t_ref,v4x_t, v4x_V)
int_v4y = np.interp(t_ref,v4y_t, v4y_V)
#%% grafica data real vs interpolda
#clf()   # Limpiamos toda la figura
#plt.plot( amp_f1_t, amp_f1_amp, 'b', t_ref, int_amp_f1, 'g.')
#plt.plot( amp_f2_t, amp_f2_amp, 'b', t_ref, int_amp_f2, 'g.')
#plt.plot( amp_f3_t, amp_f3_amp, 'b', t_ref, int_amp_f3, 'g.')
#plt.plot( caudal_motorbba_t, caudal_motorbba_Q, 'b', t_ref, int_caudal_motorbba, 'g.')
#plt.plot( temp1_t, temp1_T, 'b', t_ref, int_temp1, 'g.')
#plt.plot( temp2_t, temp2_T, 'b', t_ref, int_temp2, 'g.')
#plt.plot( temp3_t, temp3_T, 'b', t_ref, int_temp3, 'g.')
#plt.plot( temp4_t, temp4_T, 'b', t_ref, int_temp4, 'g.')
#plt.plot( v1x_t, v1x_V, 'b', t_ref, int_v1x, 'g.')
#plt.plot( v1y_t, v1y_V, 'b', t_ref, int_v1y, 'g.')
#plt.plot( v2x_t, v2x_V, 'b', t_ref, int_v2x, 'g.')
#plt.plot( v2y_t, v2y_V, 'b', t_ref, int_v2y, 'g.')
#plt.plot( v3x_t, v3x_V, 'b', t_ref, int_v3x, 'g.')
#plt.plot( v3y_t, v3y_V, 'b', t_ref, int_v3y, 'g.')
#plt.plot( v4x_t, v4x_V, 'b', t_ref, int_v4x, 'g.')
#plt.plot( v4y_t, v4y_V, 'b', t_ref, int_v4y, 'g.')
plt.legend(('valor real', 'valor_ int'), fontsize=10)
plt.title('Grafica data real vs interpolada', fontsize=14)
#%%                    UNIR DATOS INTERPOLADOS
int_amp_f2 = np.array(amp_f2_amp)
int_amp_f2=np.delete(int_amp_f2, 0)
#%%
bd_sincro=np.column_stack((int_amp_f1,int_amp_f2,int_amp_f3,int_caudal_motorbba,int_temp
1,int_temp2,int_temp3,int_temp4,int_v1x,int_v1y,int_v2x,int_v2y,int_v3x,int_v3y,int_v4x,int_v
4y))
```

```
bd_sincro_time=np.column_stack((t_ref,int_amp_f1,int_amp_f2,int_amp_f3,int_caudal_motorbb
a,int_temp1,int_temp2,int_temp3,int_temp4,int_v1x,int_v1y,int_v2x,int_v2y,int_v3x,int_v3y,int
_v4x,int_v4y))

#  agregar el tiempo de referencia

#%%                    GRABAR DATOS SINCRONIZADOS     archivo csv

savetxt('Dropbox/1 Jorge/Magister Uchile/TESIS JM/118/Data118/bd_sincro.csv', bd_sincro,
delimiter=',')

savetxt('Dropbox/1 Jorge/Magister Uchile/TESIS JM/118/Data118/bd_sincro_time.csv',
bd_sincro_time, delimiter=',')

#%%                    APLICACION MODELO DE APRENDIZAJE

bd_sincro_time = loadtxt('Dropbox/1 Jorge/Magister Uchile/TESIS
JM/118/Data118/bd_sincro_time.csv', delimiter=',')

bd_sincro_time_trval, bd_sincro_time_tr = train_test_split(bd_sincro_time, test_size=0.8,
random_state=0)

#bd_sincro_time_tr=bd_sincro_time            # Dataset si dividir en entrenamiento y validación

t_ref= bd_sincro_time_tr[:,0:1]              # tiempo de ref entrenamiento

t_ref_te= bd_sincro_time_tr[:,0:1]           # tiempo de ref testeo (validación)

#%%

# Datos entrenamiento:

# Xt=bd_sincro_time_tr[:,9:13]               # Datos de vibración de motor

# Xt=bd_sincro_time_tr[:,13:17]              # Datos de vibración de bomba

Xt=bd_sincro_time_tr[:,1:17]                 # Datos totales

# Xt=bd_sincro_time_tr[:,9:17]               # Datos de vibración de motor y bomba

scaler = preprocessing.StandardScaler().fit(Xt)  # def estandarizacion de datos

Xs=scaler.transform(Xt)                      # estandarizacion de datos

Nt=len(Xt)                                   # Cantidad de datos

#%%                    REDUCCION DE VARIABLES

inicio_time = time()

pca = PCA(n_components=2)

Xt = pca.fit_transform(Xs) # proceso para entrenar

tit_graf_RV='PCA 2D'

# tit_graf_RV='PCA 3D'

# Xt = pca.transform(Xs)       # proceso para testear
```

```
# tit_graf_RV='PCA 2D test'
# Kernel PCA  (toma mucho tiempo el proceso)
# kpca = KernelPCA(n_components=3, kernel="rbf") #linear, rbf, poly, sigmmoid, cosine
# Xt = kpca.fit_transform(Xs)
# tit_graf_RV='Kernel PCA 2D, con rbf'
# tit_graf_RV='Kernel PCA 3D, con rbf'
# Factor Analysis
# transformer = FactorAnalysis(n_components=2, random_state=0)
# Xt = transformer.fit_transform(Xs)
# tit_graf_RV='Factor Analysis, 2D'


# #MDS   (con problema, toma mucho tiempo el proceso)
# Xs, X_test = train_test_split(Xs, test_size=0.7, random_state=0)
# embedding = MDS(n_components=3)
# Xt =embedding.fit_transform(Xs)
# tit_graf_RV='MDS 3D 30% de datos'
# #ISOMAP   con problema.  Genera el siguiente error:MemoryError: Unable to allocate array with shape (27860, 27860) and data type float64 Sale ese error
# embedding = Isomap(n_components=3)
# Xt =embedding.fit_transform(Xs)
# tit_graf_RV='ISOMAP 3D'
# #LLE
# embedding = LocallyLinearEmbedding(n_neighbors=5,n_components=2)
# Xt =embedding.fit_transform(Xs)
# tit_graf_RV='LLE 2D'
fin_time = time()
tiempo_proc_red_var=fin_time-inicio_time
print("Tiempo de proceso",tit_graf_RV ,": %.10f segundos." % tiempo_proc_red_var)
#%% Grafica 2D
plt.scatter(Xt[1:Nt, 0], Xt[1:Nt, 1])
plt.scatter(Xt[Nt+1:2*Nt, 0], Xt[Nt+1:2*Nt, 1])
```

```
plt.scatter(Xt[2*Nt+1:3*Nt, 0], Xt[2*Nt+1:3*Nt, 1])
plt.xlabel('Parámetro 1', fontsize=14)
plt.ylabel('Parámetro 2', fontsize=14)
plt.title(tit_graf_RV, fontsize=14)
plt.show
#%% Grafica 3D
fig = plt.figure(figsize=(9,6))
ax = fig.add_subplot(111, projection='3d')
ax.scatter(Xt[1:Nt, 0], Xt[1:Nt, 1],Xt[1:Nt, 2])
ax.scatter(Xt[Nt+1:2*Nt, 0], Xt[Nt+1:2*Nt, 1],Xt[Nt+1:2*Nt, 2])
ax.scatter(Xt[2*Nt+1:3*Nt, 0], Xt[2*Nt+1:3*Nt, 1],Xt[2*Nt+1:3*Nt, 2])
ax.set_xlabel('Parámetro 1', fontsize=14)
ax.set_ylabel('Parámetro 2', fontsize=14)
ax.set_zlabel('Parámetro 3', fontsize=14)
plt.title(tit_graf_RV, fontsize=14)
plt.show()
#%%                    LOS MODELOS DETECCION DE ANOMALIAS
outlier_frac=0.022 #fracción de valores anomalos, valor por defecto es 0.1  =0.022
Model= EllipticEnvelope(contamination=outlier_frac)
tit_graf_DA= tit_graf_RV, ' EllipticEnvelope, outlier_frac=', outlier_frac
# Model=IsolationForest(n_estimators=100, max_samples='auto', contamination=outlier_frac)
# tit_graf_DA=tit_graf_RV, 'Isolation Forest, n_estimators=100', outlier_frac
# Model = LocalOutlierFactor(n_neighbors=20, contamination=outlier_frac)
# tit_graf_DA=tit_graf_RV, ' LOF, n_neighbors=20', outlier_frac
# Model = svm.OneClassSVM(nu=0.008, kernel="rbf", gamma='auto')
# tit_graf_DA=tit_graf_RV, 'SVM, n_neighbors=20, nu=0,008'
Yp=Model.fit_predict(Xt)
Yp=np.where(Yp==1, 0, Yp)   # si el valor es 1 ponga un cero (normal)
Yp=np.where(Yp==-1, 1, Yp)  # si el valor es -1 ponga un uno (anormal)
# Yp_t=np.column_stack((Yp,t_ref_Xte_t))     # datos de test con referencia de tiempo
# Yp_t=np.column_stack((Yp,t_ref_Xtr_t))      # datos de entrenamiento con referencia de tiempo
```

```
Yp_t=np.column_stack((Yp,t_ref))       # datos de entrenamiento con referencia de tiempo
#     Generando datos base de datos normales "Xp0" y anormales "Xp1"
Xp0=Xt[np.where(Yp == 0)[0],:]         # Normales
Xp1=Xt[np.where(Yp == 1)[0],:]         # Anormales
#%% validación del modelo con set de datos
# Datos validación:
# Xt=bd_sincro_time_trval[:,1:17]           # Datos totales
# Xs=scaler.transform(Xt)                   # estandarización datos en proceso para validación
Yp=Model.fit_predict(Xt)
Yp=np.where(Yp==1, 0, Yp)   # si el valor es 1 ponga un cero (normal)
Yp=np.where(Yp==-1, 1, Yp)  # si el valor es -1 ponga un uno (anormal)
Xp0=Xt[np.where(Yp == 0)[0],:]         # Normales
Xp1=Xt[np.where(Yp == 1)[0],:]         # Anormales
#%%   Grafica 2D
plt.scatter(Xp0[:, 0], Xp0[:, 1])
plt.scatter(Xp1[:, 0], Xp1[:, 1])
plt.xlabel('Parámetro 1', fontsize=14)
plt.ylabel('Parámetro 2', fontsize=14)
plt.legend(('Normal','Anormal'), fontsize=14)
#plt.title('Detectado', fontsize=14)
plt.title(tit_graf_DA,  fontsize=14)
plt.show()
#%%   Grafica 3D
plt.scatter(Xp0[:, 0], Xp0[:, 1])
plt.scatter(Xp1[:, 0], Xp1[:, 1])
plt.set_xlabel('Parámetro 1', fontsize=14)
plt.set_ylabel('Parámetro 2', fontsize=14)
plt.set_zlabel('Parámetro 3', fontsize=14)
plt.legend(('Normal','Anormal'), fontsize=14)
plt.title(tit_graf_DA, fontsize=14)
plt.show()
```

```
#%%     Cambiar fecha de nuemro a formato fecha - hora
#Ojo python considera el inicio del contador desde ticks: desde las 12:00 am del 1 de enero de 1970.
#excel considera el inico del contador desde 1/1/1900
Ing_fecha_num=43466.6
d=Ing_fecha_num-25568
#d=43415-25568
seg_1_dia=60*60*24
d_s=d*seg_1_dia
o=datetime.fromtimestamp(d_s)
#%%                    VERIFICACIÓN DE CADA VARIABLE DE DATOS ANOMALOS
bd_normal=bd_sincro_time[np.where(Yp == 0)[0],:]
bd_anormal=bd_sincro_time[np.where(Yp == 1)[0],:]
#%% Grafica 2D
# titulo_var, x="Corriente de Motor, Fase 1", 1
# titulo_var, x="Corriente de Motor, Fase 2", 2
# titulo_var, x="Corriente de Motor, Fase 3", 3
# titulo_var, x="caudal Bomba", 4
# titulo_var, x="Temperatura Descanso 1", 5
# titulo_var, x="Temperatura Descanso 2", 6
# titulo_var, x="Temperatura Descanso 3", 7
# titulo_var, x="Temperatura Descanso 4", 8
# titulo_var, x="Vibración Descanso 1, dirección x", 1
# titulo_var, x="Vibración Descanso 1, dirección y", 10
# titulo_var, x="Vibración Descanso 2, dirección x",11
# titulo_var, x="Vibración Descanso 2, dirección y",12
# titulo_var, x="Vibración Descanso 3, dirección x", 13
# titulo_var, x="Vibración Descanso 3, dirección y",14
# titulo_var, x="Vibración Descanso 4, dirección x",15
titulo_var, x="Vibración Descanso 4, dirección y",16
plt.scatter(bd_normal[:, 0], bd_normal[:, x])
```

```
plt.scatter(bd_anormal[:, 0], bd_anormal[:, x])
#plt.scatter(Xt[Nt+1:2*Nt, 0], Xt[Nt+1:2*Nt, 1])
#plt.scatter(Xt[2*Nt+1:3*Nt, 0], Xt[2*Nt+1:3*Nt, 1])
plt.xlabel('tiempo', fontsize=14)
plt.ylabel(titulo_var, fontsize=14)
plt.legend(('Normal','Anormal'), fontsize=14)
plt.title('Anomalias en '+titulo_var+' V/S Tiempo', fontsize=14)
plt.show
#%%  Grafica de orbita descanso
i=0
f=27800
# descanso 1
titulo_var, x, y="Vibración Descanso 1, dirección x-y", 9,10
# titulo_var, x, y="Vibración Descanso 2, dirección x-y", 11,12
# titulo_var, x, y="Vibración Descanso 3, dirección x-y", 13,14
# titulo_var, x, y="Vibración Descanso 4, dirección x-y", 15,16
plt.scatter(bd_normal[i:f, x], bd_normal[i:f, y])
plt.xlabel('Desplazamiento x', fontsize=14)
plt.ylabel('Desplazamiento y', fontsize=14)
plt.xlim(0,50)
plt.ylim(0,40)
plt.title('Orbita RMS x-y '+titulo_var, fontsize=14)
plt.show
```