**Mantenimiento preventivo con Machine Learning**

1. Problemática

El equipo de dirección de Ayrtac ha puesto foco en la mejora continua de los servicios que ofrece a sus clientes y como se viene viendo a lo largo de este proyecto, dentro del plan de transformación Industrial 4.0, se ha decidido diseñar una estrategia y plan de mantenimiento preventivo de su gama de cerradoras MC. El objetivo es utilizar modelos de machine learning predictivo que sean capaces de evitar averías, sustitución de piezas no fungibles, cortes en la producción e incremento de mano de obra de las fábricas conserveras de sus clientes, donde están funcionando las cerradoras MC. Estos modelos de machine learning predictivo serán capaces de detectar los patrones ocultos en los datos históricos del funcionamiento de las máquinas y con ello poder anticipar los fallos dando un margen al personal de mantenimiento para actuar y sustituir las piezas necesarias.

Ayrtac es una empresa de diseño y fabricación de máquinas cerradoras entre otras cosas y no posee profesionales con conocimientos analíticos para diseñar los modelos de machine learning, de aquí la colaboración con MEIDAT para el diseño de la arquitectura y analítica big data.

Una vez creado el modelo se puede construir un plan de mantenimiento preventivo y predecir cada cuantas horas hay que revisarlas o cambiar piezas susceptibles de avería. De esta manera el departamento de mantenimiento ofrecerá a sus clientes el plan específico según el tipo de máquina/cliente y ofrecer un conjunto de intervenciones u operaciones preventivas a realizar en los equipos y lograr unos objetivos de disponibilidad, fiabilidad y coste, así ampliando la vida útil de las cerradoras MC.

Junto con el servicio técnico de Ayrtac, se ha estimado que al cliente se le ofrecerá un modelo de mantenimiento OPEX con una cuota mensual dependiendo de las características negociadas, con el objetivo de llegar a conseguir una disminución de entre el 50 % o 60 % de las averías. Al reducir estas averías se estima que se puede conseguir un rendimiento medio de un 4% más en la producción de la línea, de esta forma el gasto de mantenimiento preventivo será inmediatamente recuperado por el ahorro de costes en paradas de la producción.

1. Modelo de Machine Learning

En este apartado se va a explicar cómo se ha realizado el modelo de machine learning predictivo, desde la descarga de los datos hasta el scoring de fallo de la máquina.

* 1. Datos

Como para la realización del proyecto no se han podido obtener los datos reales de las máquinas cerradoras, se ha elegido un dataset sintético de kaggle que cumple los requisitos necesarios para poder entrenar un modelo de mantenimiento predictivo y poner a competir varios algoritmos.

* 1. Python

Para la realización del modelo se ha utilizado el lenguaje de programación Python, ya que en caso de ejecutar el proyecto se haría con este lenguaje. La distribución utilizada ha sido Anaconda puesto que es de distribución libre y muy utilizada para la ciencia de datos. Para realizar los notebook se ha utilizado Jupyter y las principales bibliotecas utilizadas han sido Numpy, Pandas y Scikit-learn.

* 1. Set Up del entorno

El primer paso del proyecto ha sido crear un entorno de trabajo en Anaconda Prompt llamado **Mant\_predictivo** en el que se han instalado las bibliotecas necesarias como:

Python, numpy, pandas, matplotlib, seaborn, scikit-learn, scipy, sqlalchemy, xgboost, jupyter, pyjanitor, scikit-plot, yellowbrick, imbalanced-learn, jupyter\_contrib\_nbextensions, cloudpickle, districtdatalabs y yellowbrick.

Una vez creado el entorno e importadas las biblotecas se procede a crear el directorio del proyecto con las carpetas que posteriormente se utilizara para guardar todos documentos y archivos que se vayan generando en el desarrollo.

Se guarda el dataset del proyecto dentro del directorio y con pandas se importa en el notebook, comprobando que se ha importado correctamente y si no es así configurando los parámetros para que tenga la estructura correcta.

Una vez importado el dataset se extrae y se reserva el dataset de validación que corresponde a un 30% de los datos totales, servirá posteriormente al final del proyecto para ejecutar el código de ejecución y comprobar que el código programado funciona correctamente con datos nunca vistos por el algoritmo.

En este proyecto en concreto no será necesario realizar una muestra de los datos para trabajar, ya que con la totalidad de los datos será posible procesarlos en el ordenador.

* 1. Calidad de datos
     + Importación de los datos

Una vez reservado el dataset de validación, se importa el de trabajo. Se cuenta con un dataset de 4919 filas y 27 columnas.

* + - Corrección de nombre de las variables

Se realiza una corrección manual del nombre de las variables, poniendo todas letras en minúsculas y sustituyendo los espacios por guiones bajos.

Texto

Descripción generada automáticamente

Quedando de la siguiente forma:

Texto

Descripción generada automáticamente

* + - Visión general

Tabla

Descripción generada automáticamente

Con esta primera visión general de los datos se observa que no existen nulos aparentes en ninguna variable, casi todas han sido importadas de tipo numérico entero (int64) excepto fallo y operador. Como ya se había comentado hay un total de 27 columnas y 4919 registros usando una memoria del sistema de 1.1 MB.

Se observar variables de temperatura, humedad, operador, 15 medidas de las máquinas, horas desde el último fallo y luego variables de tiempo obtenidas del índice que es Date. Son datos que en la mayoría de los registros son ventanas temporales de una hora del funcionamiento de la máquina.

* + - Tipos de datos

Se va a hacer una corrección de los tipos de datos pasando a categóricas (object) las variables medida\_2, medida\_3, mes, dia\_mes, dia\_semana.

* + - Valores únicos

Las únicas variables que tienen valores únicos son segundo, año y minuto ya que en todos registros el año es 2016 y al ser ventanas temporales de 1 hora no tienen ni segundos ni minutos. Se procede a eliminar estas 3 variables.

* + - Duplicados

Este dataset no tiene registros duplicados por lo que no hay que hacer ninguna corrección.

* + - Separar numéricas y categóricas

Para seguir con el proceso de calidad de datos se van a separar las variables que sean numéricas y las que sean categóricas, esto es debido a que cada tipo de variable necesita ser gestionado de manera diferente.

* + - Gestión de categóricas

En las variables categóricas no existen ni nulos ni atípicos, por lo que no será necesario realizar correcciones.

* + - Gestión de numéricas.

No existen nulos, pero dejamos el código preparado por si en datos nuevos aparecieran que fueran sustituidos por la mediana.

Se detectan algún atípico y para eliminarlo se aplica una winsorización manual en las variables temperatura, humedad y horas desde último fallo, los límites establecidos son los siguientes.

* + - * Temperatura: entre 50 y 80.
      * Humedad: entre 60 y 120
      * Horas ultimo fallo: 0 y 1000
  1. EDA

En el análisis exploratorio de los datos se tienen dos objetivos principales:

* Conocer mejor los datos con los que se están trabajando.
* Encontrar otros problemas de calidad de datos.

Para hacerlo se cuenta con dos herramientas principales:

* Estadísticos básicos.
* Gráficos.

Además, cada una de ellas tiene diferentes técnicas en función si las variables son categóricas o continua. Por ello en la fase anterior se dividió el dataset, en esta parte se sigue trabajando como se venía haciendo.

EDA en Categóricas

* Análisis estadístico:

Se analizan las frecuencias de los valores de cada variable por si fuera interesante agrupar varios por su poca relevancia, en este caso no es necesario realizar ninguna agrupación ni aplicar ningún cambio.

Como en este tipo de proyectos la variable target suele estar muy desbalanceada se procede a analizar la target en el dataset de validación, ya que se hace aleatoriamente.

La variable target en validación indica que hay un 98,5 % de No y un 1,5 % de Si, por lo que es posible que en el modelado sea necesario realizar balanceo.

* Gráficos:

En los gráficos de las variables categóricas se observa que el Operador2 de la variable operador es el que más frecuencia tiene, las demás tienen una frecuencia de los valores muy parecida excepto la target fallo que como anteriormente se ha comentado está muy desbalanceada.

EDA en Numéricas

* Análisis estadístico:

Interfaz de usuario gráfica, Aplicación, Tabla, Excel

Descripción generada automáticamente

En la ilustración anterior se observan los estadísticos básicos de cada variable numérica. No se detecta nada de mayor importancia en los estadísticos básicos pero sí que es importante tener en cuenta el min y max, así como la media y mediana entre los que se encuentran las diferentes variables.

* Gráficos:

Lo principal que se puede observar en los gráficos es que las distribuciones son en todas variables parecidas, bastantes normales pero con la parte superior multimodal, excepto la variable horas desde última avería que se encuentra una distribución más sesgada a la derecha donde se encuentran valores más entre 50 aprox que en valores más altos.

* 1. Transformación de los datos

En este apartado se van a transformar las variables necesarias para trabajar con sklearn ya que es necesario que todas variables categóricas estén como numéricas, también en este apartado se suelen crear variable sintéticas que tengan sentido desde el punto de vista negocio.

Se sigue trabajando con el dataset diferenciado entre variables categóricas y continuas, pero sí que ya se va a separar la variable target.

En variables categóricas

* One Hot Encoding

One Hot Encoding es una técnica que tiene Sklearn para transformar las variables categóricas en numéricas. Este tipo de Encoding se suele usar en variables nominales y consiste en que cada categoría de la variable original pasará a ser una nueva variable de 1/0.

Las variables a las que se ha aplicado OHE han sido a “Operador”, ”medida\_2”, ”medida\_3”, ”fallo”, ”mes”, ”día\_mes”, ”día\_semana”.

En variable numéricas

En este proyecto no se aplica transformaciones a las variables numéricas ya que no es necesario ni discretizar, ni binarizar, ni normalizar.

En dataset unificado

Cuando se ha terminado con las transformaciones de la variable numéricas y categóricas se unifica en un solo dataset y se hace el reescalado de las variables.

* Reescalado con Min-Max

Se ha decidido aplicar esta técnica de reescalado por que en la parte de modelado se pretende poner a competir varios algoritmos como por ejemplo la regresión logística, aun así, siempre es recomendable reescalar.

El reescalado consiste en dejar todas variables entre los mismos máximos y mínimos para que las que tienen más escala o mayor varianza no tengan más peso en el algoritmo.

Con Min-Max Scaling se reduce el min de la variable en 0 y el máximo valor en 1, consiguiente que todas las variables estén entre esos valores.

Las variables en las que hemos hecho OHE no es necesario aplicar la técnica Min-Max ya que sus valores son 0 o 1. Las variable a reescalar con Min-Max Scaling son:

Imagen que contiene Word

Descripción generada automáticamente

* 1. Preselección de variables.

En esta sección del proyecto se van a utilizar técnicas que permiten filtrar el número de variables que van a pasar al “pull” de variables de modelización. Esta técnica se divide en dos fases, métodos supervisados y métodos no supervisados.

Métodos supervisados

Consiste en identificar las variables que son más predictoras a nivel individual.

Hay muchos métodos, pero en este caso se van a utilizar 3:

* Mutual Information.
* Recursive Feature Elimination
* Permutation Importance.

Se observa en ambos métodos que las variables que tienen la mayoría del peso en las predicciones son “horas\_desde\_ultimo\_fallo\_mms”, “temperatura\_mms”, “humedad\_mms”. En Mutual Informations y en RFE también marcaban como predictoras otras variables procedentes de la fecha y alguna otra medida. Visto desde la perspectiva negocio no se ve claro que la máquina pueda fallar dependiento el numero de día que es o mes del año en el que se está, por lo que no se introducen como variables predictoras. Por lo que se eligen las 3 variables antes nombradas como las variables predictoras del dataset y se eliminan las demás que no tenían casi influencia en detectar si la máquina falla o no.

Interfaz de usuario gráfica, Aplicación, Word

Descripción generada automáticamente

* 1. Balanceo

Como ya se ha comentado a lo largo del proceso de modelización es posible que sea necesario realizar balanceo con el objetivo de probar si mejora la predicción en el dataset.

Para ello se aplican diferentes métodos, incluyendo el Sin Balanceo y se comprueba el resultado de la ROC AUC al aplicar un algoritmo como la regresión logística.

* Sin Balanceo:
  + ROC = 0.797
* Undersampling: trata de eliminar aleatoriamente parte de los ceros.
  + ROC = 0.797
* Oversampling: trata de repetir varias veces los unos.
  + ROC = 0.791
* SMOTE-Tomek: técnica que combina el sobremuestreo y el submuestreo eliminando el ruido que se puede generar.
  + ROC: 0.788

Dado que la diferencia en la predicción es tan poco relevante, se ha decidido optar por la técnica del Sin Balanceo, ya que haciéndolo no obtenemos mejora y siempre es más intenso computacionalmente hacer el balanceo.

* 1. Modelización para clasificación.

En este apartado se van a utilizar algoritmos para modelos supervisados de clasificación, ya que lo que se quiere predecir es si la máquina falla o no. Para ello se van a poner a “competir” 3 algoritmos, el primero la regresión logística, en segundo lugar Random Forest y en tercer lugar XGBoost, más adelante se explicaran cada uno de ellos y el que es elegido para ponerlo en producción. También se mostrarán los resultados de la evaluación del modelo y los diferentes reporting que se utilizan.

* **Dataset de validación**

En primer lugar y antes de empezar a modelar se reserva el dataset de validación con el que después se evaluara el modelo.

* **Algoritmos**
  + Regresión Logística:

La regresión logística es uno de los algoritmos más usados en contextos empresariales por su histórico, fiabilidad y capacidad explicativa, solo se puede usar en casos como este, aplicaciones de clasificación y más de target dicotómica. Es un algoritmo rápido de entrenar y usar.

* + Random Forest

Random Forest es un algoritmo muy estable y seguro, se puede utilizar en estos casos que la explicabilidad no es muy importante y se quiere buena capacidad predictiva confiando en que va a seguir funcionando bien en producción. Pertenece a la familia de los arboles de decisión siendo un algoritmo de ensembling, que es la evolución de una técnica conocida como Bagging, que consiste en generar muchos árboles sobre muestras aleatorias y después combinarlos en una única salida con la media si la target es continua o con la moda si la target es categórica.

* + XGBoost:

Este es un algoritmo también de la familia de los arboles de clasificación pero utiliza otra estrategia para mejorar los resultados de los modelos. XGBoost utiliza la técnica llamada boosting que funciona también haciendo muchos modelos pero donde cada uno está destinado a mejorar la predicción en los casos en los que el anterior no consiguió tener buena predicción, intentando capturar los patrones no capturados en los errores hasta que el error parezca aleatorio. Los modelos suelen ser bastante precisos ya que trabajan reduciendo el sesgo, pero pueden tener más riesgo de sobre ajustar.

* **Competición de algoritmos:**

Para saber cual es el algoritmo que mejor funciona en este caso práctico se puede configurar un pipe con un diccionario de algoritmos, parámetros y valores a testar como el que se muestra a continuación:

Texto, Pizarra

Descripción generada automáticamente

* **Resultados de evaluación:**

Los resultados de la “competición” de algoritmos ha sido elegido como mejor predictor XGBoost con una AUC = 92,25 %, en segundo lugar Random Forest con una AUC = 88,84 % y por último la Regresión logística con un AUC = 83,36 %.

Por lo que el algoritmo elegido para preparar el código de entrenamiento y ejecución es el XGBoost con los siguientes parámetros:

Texto

Descripción generada automáticamente

A continuación, se adjuntan los gráficos por si se quiere estudiar el comportamiento de la predicción de forma gráfica.

**Gain Chart**

Gráfico, Gráfico de líneas, Gráfico de dispersión

Descripción generada automáticamente

**Lift Chart**

Gráfico, Gráfico de líneas

Descripción generada automáticamente

**ROC Chart**

Gráfico, Gráfico de líneas, Gráfico de dispersión

Descripción generada automáticamente

* 1. Preparación del código de producción

En este apartado se prepara el código final que después será puesto en producción.

Se ha visto que el dataset tenia distintas variables, pero en este paso es cuando se dejan solo las necesarias que utilizara el algoritmo para predecir. Se introducen todos los cambios que se van a hacer en funciones o se crean el pipe de preprocesamiento.

Un pipeline es una funcionalidad de ScikitLearn que permite encadenar procesos en serie. La salida del proceso actual será la entrada del proceso siguiente. Si se guarda un pipeline guardará todo el proceso que contiene, incluyendo tanto la parte de preprocesamiento como la de modelización. Esto hace más sencillo el reentrenamiento y la ejecución de todo el flujo del proyecto de machine learning.

Por último, una vez que se tiene el pipe final, se guarda con formato .pickle para que mantenga la configuración interna.

En la parte de anexos de adjuntará el código de producción.

* 1. Script código de reentrenamiento y ejecución.

Este apartado es el resultado final de todo el trabajo anteriormente realizado. Se entregan al cliente dos script, uno el de reentrenamiento y otro el de ejecución.

El de reentrenemiento será necesario para entrenar el modelo de vez en cuando para que pueda detectar nuevos patrones que van surgiendo en los datos, en este caso nuevas causas de avería.

El script de ejecución es el código que se ejecutará cuando vienen datos nuevos y dará un scoring de la probabilidad de fallo que tiene la máquina en ese momento. Para ello se ha hecho una simulación con los datos de validación que se reservaron al principio y que el modelo no ha visto en ningún momento.

En la siguiente imagen se muestra el resultado ordenado por de mayor probabilidad de fallo a menor.

Tabla

Descripción generada automáticamente

En los anexos están los códigos de reentrenamiento y producción.