Análisis de series temporales

Práctico de Aprendizaje supervisado y no supervisado

En este práctico vamos a continuar trabajando sobre aprendizaje automático. En este caso específicamente sobre modelos **supervisados** y **no supervisados**.
Se propone diseñar e implementar algunos modelos y definir métricas para ver como performan.

Instalación

Recomiendo instalar un ambiente virtual con las librerías estándar para procesar datos y visualizarlos, para evitar contratiempos.
Las siguientes librerías son más que suficiente para cumplir con los objetivos de este práctico: **jupyterlab, numpy**, **pandas**, **matplotlib**, **scikit-learn** y **xgboost**.

Informe

Se deberá realizar un informe en formato jupyter notebook donde se presente la información solicitada, y se explique las decisiones tomadas a lo largo del análisis. Se espera que el mismo notebook donde se realizan las acciones, tenga un formato de informe con títulos y texto en markdown en cada sección, como se muestra en el informe de ejemplo.
Una vez terminado el informe se deberá **exportar a HTML** para evitar problemas de lectura y facilitar la corrección.
El informe se puede organizar a su mejor criterio, sin omitir la información solicitada.
Se espera que se realicen gráficos cada vez que se pueda, aunque no se lo pida de forma explícita.
Recuerden no comitear el dataset ni los modelos entrenados.

Implementación

En estos prácticos vamos a trabajar con la problemática completa, es decir que nos enfrentamos a un problema de **clasificación multiclase**. Vamos a utilizar el **target** original dado en el dataset, pero tratando de predecir solo **21 clases** para simplificarlo, es decir que vamos a tratar de predecir si los envíos llegan en **0, 1, 2 … o 20 días hábiles**.

También vamos a agregar el concepto de **ventana de predicción**, esto es básicamente una ventana de tiempo formada por dos componentes, un **speed** (cantidad de días hábiles

predichos), más un **offset** (margen de error de la predicción). Nuestras predicciones finales serían de la forma: **(speed, offset)**. Donde por ejemplo **(1, 2)** representa una ventana que abarca desde **1 hasta 3 días hábiles**. También vamos a agregar como restricción fuerte, que un **offset** no puede ser mayor que **3**.
Si un envío cae dentro de una ventana de predicción, diremos que llegó **ontime**, si llega antes de la ventana, diremos que llegó **early**, y por último si llegó después de la ventana de predicción diremos que llegó **delay**.

En cuanto a los features, solo vamos a utilizar los de las rutas, es decir **sender_zipcode**, **receiver_zipcode** y **service**.

Preparación de los features

Diseñar un pipeline con las siguientes transformaciones:

- Recortar el último dígito de los zip codes
- Normalizar los features para que queden en el rango (0, 1)
- Proyectar los features utilizando PCA, manteniendo 3 componentes.

NOTA IMPORTANTE: Estas transformaciones se deben aplicar **sin modificar** el dataframe con los datos originales, pueden usar copias para hacer las pruebas. Es decir que no deben hacer las transformaciones y guardarlas en un dataframe, tal como se hace en el ejemplo.

Preparación del target:

- Limitar el **target** a 20, es decir asignar todo target mayor que 20 a 20.

Preparación del dataset:

- Particionar el dataset en **train** y **test**, teniendo los cuidados necesarios para no romper la temporalidad de los datos. El conjunto de training no puede tener menos del 50% de los datos.
- Si les parece necesario, pueden realizar algún tipo de filtrado o limpieza de los datos, explicando por qué les parece necesario.

Modelo basado en árboles de decisión (supervisado)

- Crear un pipeline con los pasos de “preparación de los features” agregando el clasificador **XGBoostClassifier** como estimador final. Entrenar este modelo, predecir el conjunto de test y calcular las métricas **ontime**, **delay** y **early**, sin ventana (se puede utilizar un array con ceros como en el ejemplo).
- Explicar muy brevemente como funcionan esta clase de modelos.

Modelo basado en vecinos cercanos (no supervisado / semi supervisado)

- Crear un pipeline con los pasos de “preparación de los features” agregando el clasificador **KNeighborsClassifier** como estimador final. Entrenar este modelo, predecir el conjunto de test y calcular las métricas **ontime**, **delay** y **early**, sin ventana.
- Explicar muy brevemente como funcionan esta clase de modelos.

Modelo basado en regresión:

- Crear un pipeline con los pasos de “preparación de los features” agregando un regresor a elección de ustedes como estimador final. Este regresor puede ser tanto **supervisado** como **no supervisado**.
- Entrenar este modelo, predecir el conjunto de test y redondear las predicciones de forma inteligente. Explicar el criterio de redondeo.
- Calcular las métricas **ontime**, **delay** y **early**, sin ventana.
- Justificar muy brevemente la elección del modelo.

Ventanas de predicción:

- Construir un **offset** para mejorar las predicciones de nuestros modelos, de forma que tenga **avg_offset** menor o igual a **1**, recalcular las métricas y explicar cómo se lo construyó.
- Construir un **offset** que mejore las métricas de los modelo y que además tenga un **avg_offset** menor o igual que **2.5**, recalcular las métricas y explicar cómo se lo construyó.

NOTA IMPORTANTE: Se espera que se construya el offset de forma inteligente, creando reglas para formar las ventanas y no asignando siempre el valor máximo posible que cumpla con la condición.
También tengan en cuenta que este punto lo deben hacer al final del trabajo, cuando ya tengamos todos los modelos entrenados, y sus predicciones guardadas.

Recuerden que todas las respuestas y decisiones tomadas tienen que estar justificadas con datos o gráficos. Se evaluará la legibilidad del notebook, el detalle a la hora de responder las preguntas y mostrar la información solicitada, y además que los gráficos utilizados sean apropiados y correctos.