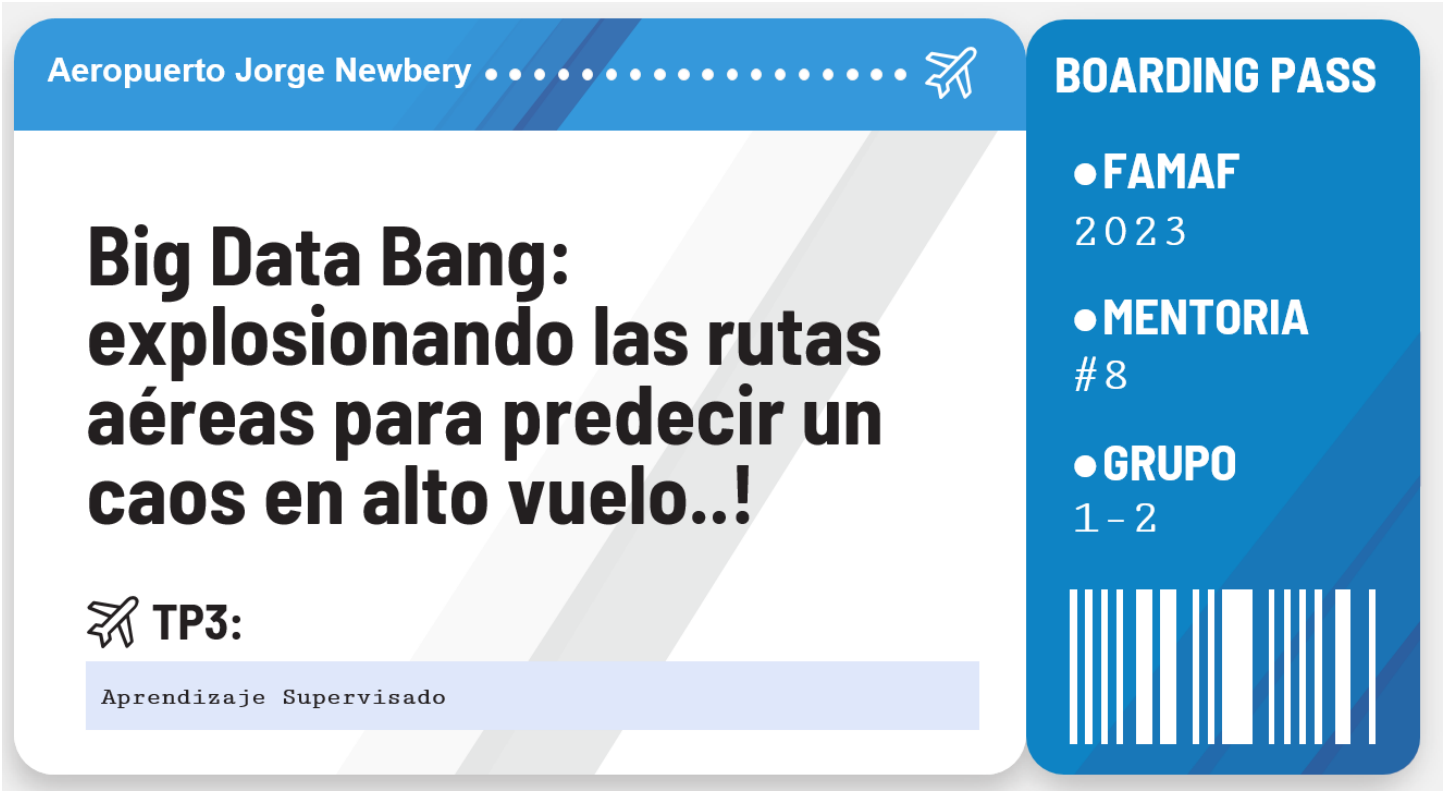

1. Preparación de los datos para aplicar modelos de clasificación:

a. Generar a partir del dataset, los conjuntos de train, test y validation.
b. Analizar las proporciones consideradas para cada conjunto con respecto al total del dataset. Verificar el total de datos de cada conjunto.
c. Para cada conjunto (train, test, val), ¿cuántos datos de cada clase target hay?, se debe buscar mantener la proporción del dataset total.

2. Creación de un modelo baseline:

a. Entrenar un modelo “baseline”, es decir lo más simple posible, para con ello tener un punto de partida con el cual comparar modelos más complejos. Fijar la semilla aleatoria para hacer repetible el experimento
(Hint: se puede usar por ejemplo la clase “DummyClassifier” de scikit-learn)

Documentación:

https://scikit-learn.org/stable/modules/generated/sklearn.dummy.DummyClassifier.html

b. Evaluar sobre el conjunto de entrenamiento, validación y test reportando:
   i. Accuracy
   ii. Precision
   iii. Recall
   iv. F1
   v. Matriz de confusión

c. Pensar cuál métrica es conveniente optimizar en éste problema de clasificación donde se predice el retraso de los vuelos. ¿Sería el Accuracy una buena métrica?

3. Predicción de modelos lineales:

a. Entrenar modelos lineales de clasificación para predecir la variable objetivo. Para ello, deberán utilizar “LogisticRegression” de scikit-learn. Fijar la semilla aleatoria para hacer repetible el experimento y elegir al menos uno de los siguientes otros modelos:
   i. La clase SGDClassifier de scikit-learn.
   ii. La clase LinearSVC de scikit-learn.

Documentación:

https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Logistic Regression.html
https://scikit-learn.org/stable/modules/sgd.html
https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDClassifier.html
https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html

b. Evaluar sobre el conjunto de entrenamiento, validación y test reportando:
i. Accuracy
ii. Precision
iii. Recall
iv. F1
v. Matriz de confusión

c. Elaborar conclusiones en base a la métrica a optimizar y comparar con el modelo baseline.

4. Predicción de modelos basados en árboles de decisión

a. Para ello, deberán fijar la semilla aleatoria y elegir al menos dos de los siguientes modelos:
   i. La clase DecisionTreeClassifier de scikit-learn.
   ii. La clase RandomForestClassifier de scikit-learn.
   iii. La clase GradientBoostingClassifier de scikit-learn.

Documentación:

https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html
https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html

b. Evaluar sobre el conjunto de entrenamiento, validación y test reportando:
   i. Accuracy
   ii. Precision
   iii. Recall
   iv. F1
   v. Matriz de confusión

c. Elabore conclusiones en base a la métrica a optimizar, comparando los modelos elegidos y contrastando con el modelo lineal elegido.

5. Ajuste por hiperparámetros:

a. Para los dos “mejores modelos” obtenidos en los puntos anteriores, seleccionar valores para los hiperparámetros principales de dichos modelos (ajustar con por lo menos 3 parámetros). Utilizar grid-search y k-fold cross-validation.
b. Mencionar el mejor modelo obtenido de la Optimización de Hiperparámetros y con cuáles parámetros se obtuvo ese resultado.
c. Con el mejor modelo obtenido realizar las predicciones sobre test y val.
d. Reportar las métricas del mejor modelo, incluyendo las matrices de confusión.

Comparar el mejor modelo obtenido, con el modelo con parámetros por defecto y con el modelo baseline. Elaborar conclusiones al respecto pensando en la resolución de nuestro problema de clasificación.

TP 3 Aprendizaje Supervisado

Características que debe cumplir el entregable:

- Generar un dataset “limpio”, con todos los pasos aplicados, ya que será el que utilizaremos en los siguientes TP para los modelos de clasificación. Concluir luego de ésta “limpieza” cuántos registros hemos mantenido/eliminado, con el fin de no quedarnos con muy pocos registros para avanzar más adelante con algún modelo de clasificación.
- Se debe ir desarrollando cada punto en la misma notebook donde se escriba el código. Dicho notebook debe contar con un índice, con sus diferentes apartados y el código debe ser fácil de leer, estar probado y comentado (esto último, en función de la necesidad).
- Se debe enviar el link directo del archivo .ipynb ó alternativamente subir el entregable a un repositorio GitHub mediante la integración con Google Colab. Recordar que al compartir el notebook, queden habilitados los permisos de edición, para poder dejar comentarios/correcciones.
- Tener en cuenta que si bien, pueden realizar diversos análisis y visualizaciones, se debe dejar en el entregable sólo aquello que sea relevante.
- Luego de cada análisis es importante poder obtener una conclusión de lo observado y/o breve interpretación de los resultados.

Fecha de Entrega: 11/09