**Universidad de los Andes**

**Facultad de Ingeniería**

**Maestría en Inteligencia Artificial MAIA**

**Camilo Eduardo Matson Hernandez**

**Taller 6– MLFLow**

1. *Parte 1 : Desarrollo de modelos*

El código en general contiene el desarrollo de un modelo de clasificación para detectar la tasa de deserción o churn de clientes de un banco.

El código inicia importando las librerías de manejo de vectores **numpy** y las de manejo de tablas **pandas**. Así mismo se incluyen elementos de sklearn, como los escaladores, el modelo de clasificación para al caso es Random Forest, el componente train\_test\_split para la división del conjunto de entrenamiento y pruebas y finalmente el componente de métricas metrics.

Luego sigue un componente exploratorio de los datos, donde se verifica la proporción de la variable objetivo por medio del comando **df["Attrition\_Flag"].unique()** que nos muestra cuales son los estados de la **misma array(['Existing Customer', 'Attrited Customer'], dtype=object)** y un descriptivo de los campos.

Teniendo los siguientes resultados:

Churn Rate = 16.1 %

Number of Duplicate Entries: 0

Number of Missing Values: 0

Number of Features: 20

Number of Customers: 10127

Data Types and Frequency in Dataset:

int64 9

object 6

float64 5

Posteriormente inicia la fase de preprocesamiento de los datos donde se Utiliza MinMaxScaler para escalar las variables numéricas y se realiza división de datos en conjuntos de entrenamiento y prueba mediante train\_test\_split.

Una vez preprocesados los datos se dividen en train y test, y se hace el entrenamiento usando el modelo random forest, para luego hacer la predicción en los datos no observados y luego construir las métricas del modelo.

Accuracy Score 0.9664296998420221

Precision 0.9304123711340206

Recall 0.8615751789976134

F1-Score 0.8946716232961587

ROC Score 0.9243985691485936

Finalmente se hace validación cruzada del modelo para mirar su capacidad de ajuste, respecto a su capacidad de generalización.

1. *Parte 2 : Empaquetamiento*

Punto 6: Descripción del archivo config.yml es un archivo de configuración en formato YAML que se usa para definir varios parámetros y configuraciones para entrenar y evaluar los modelos de machine learning.

**Inicia con la definición del nombre del paquete:**

package\_name: modelo-abandono

**Posteriormente se incluyen los archivos de datos**

train\_data\_file: bankchurn\_train.csv

test\_data\_file: bankchurn\_test.csv

**Se incluye la variable objetivo:**

target: Attrition\_Flag

**Posteriormente se incluyen las variables con las características y las variables temporales que pueden ser usadas para análisis o transformaciones adicionales.**

features:

- Customer\_Age

- Total\_Amt\_Chng\_Q4\_Q1

- Total\_Relationship\_Count

- Total\_Revolving\_Bal

- Total\_Ct\_Chng\_Q4\_Q1

- Total\_Trans\_Ct

- Total\_Trans\_Amt

- Months\_Inactive\_12\_mon

- Contacts\_Count\_12\_mon

temp\_features:

- Gender

- Dependent\_count

- Education\_Level

- Marital\_Status

- Income\_Category

- Card\_Category

- Months\_on\_book

- Credit\_Limit

- Avg\_Open\_To\_Buy

- Avg\_Utilization\_Ratio

Posteriormente se incluyen las variables con las características y las variables temporales que pueden ser usadas para análisis o transformaciones adicionales. Incluye el nombre del paquete (modelo-abandono), los archivos de datos de entrenamiento y prueba (bankchurn\_train.csv y bankchurn\_test.csv), la variable objetivo (Attrition\_Flag), las características del modelo, tanto principales como temporales, el nombre y archivo de guardado del pipeline, la fracción del conjunto de prueba (0.25), la semilla para reproducibilidad (42), los hiperparámetros del modelo (n\_estimators: 850 y max\_depth: 19), las variables cualitativas y categóricas a mapear, y el mapeo de estas variables cualitativas a valores numéricos. Este archivo facilita la gestión y modificación de parámetros sin necesidad de cambiar el código fuente del modelo.

Punto 10: test\_prediction.py es un código de test unitario para la función make\_prediction del modulo model.predict. El código inicia importando las librerías math, numpy y model.predict de make\_prediction. Luego se define la función de prueba test\_make\_prediction la cual recibe datos para las pruebas unitarias, con el valor 0 para el primer valor esperado, donde se esperan 1127 predicciones. Con el se ejecuta la predidcción con el domando make\_predictions . se verifican los resultados con reults.get y se verifica tanto el formato de la salida como los valores con los asserts.