# Impoting general libraries

import warnings

import itertools

import random

# Data Analysis Libraries

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

# Data Preparation, Modeling, and Evaluation (SKLearn Modules)

from sklearn.pipeline import make\_pipeline

from sklearn.preprocessing import LabelBinarizer

from sklearn.preprocessing import MinMaxScaler,RobustScaler, StandardScaler

from sklearn.neighbors import KNeighborsClassifier

from sklearn.model\_selection import train\_test\_split, cross\_val\_score

# Machine Learning Models

from sklearn.svm import LinearSVC, SVC

from sklearn.linear\_model import LogisticRegression

from sklearn.ensemble import RandomForestClassifier

# Machine Learning Model Evaluation Methods

from sklearn.metrics import accuracy\_score, f1\_score, roc\_auc\_score, confusion\_matrix

# Load data from your specific Spark table

df = spark.sql("SELECT \* FROM lh\_data\_analysis.orange\_telecom\_churn\_data LIMIT 1000")

display(df)

data = df.toPandas()

data.head()

data.churned.value\_counts().plot(kind = "bar", figsize = (10,4),color='Maroon')

plt.title("Ratio of Customers Who churned and Who did Not")

# Remove extraneous columns

data.drop(['state', 'area\_code', 'phone\_number'], axis=1, inplace=True)

# Separating the numerical variables from the categorical variables.

num\_var=data.select\_dtypes(include=[np.number])

cat\_var=data.select\_dtypes(include=[object])

# Convert the explanatory variable to a binary from bool

data.churned = data.churned.astype(int)

# Label Binarization

lb = LabelBinarizer()

for col in cat\_var.columns:

    data[col] = lb.fit\_transform(data[col])

# Separate the label from the features

X= [x for x in data.columns if x != 'churned']

y = data['churned']

X = data[X]

print(X.shape,y.shape)

#Train Test Split: Reserve 30% of data for final model testing

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.30, random\_state=20)

print(X\_train.shape, X\_test.shape, y\_train.shape, y\_test.shape)

#Logistic Regression (No Regularisation)

pipe = make\_pipeline(LogisticRegression(max\_iter=10000))

scores = cross\_val\_score(pipe, X\_train, y\_train, cv=5, scoring="accuracy")

print(scores, scores.mean())

# classification - Random Forest

pipe = make\_pipeline(RandomForestClassifier(n\_estimators=1000,max\_depth=10,criterion='gini'))

scores = cross\_val\_score(pipe, X\_train, y\_train, cv=5, scoring="accuracy")

print(scores, scores.mean())

# Investigating whether class imbalance has impacted our models.

model = RandomForestClassifier()

model.fit(X\_train, y\_train)

y\_pred = model.predict(X\_test)

matrix = confusion\_matrix(y\_test, y\_pred)

#Visualise the matrix

plt.figure(figsize=(10,5))

sns.heatmap(matrix, annot=True, cmap='Blues')

plt.xlabel("Predicted Category")

plt.ylabel("Actual Category")

plt.title("Confusion Matrix for Model We have Selected to Tune")

plt.show()

matrix

import mlflow

from synapse.ml.predict import MLFlowTransformer

df = spark.read.format("delta").load(

   'abfss://c3f6eb93-37e5-4a1e-bbc4-23ec49e99623@onelake.dfs.fabric.microsoft.com/69dc2150-e138-4a5d-9088-9bea790f44cd/Tables/orange\_telecom\_churn\_data' # Your input table filepath here

)

model = MLFlowTransformer(

    inputCols=<INPUT\_COLS>, # Your input columns here

    outputCol=<OUTPUT\_COLS>, # Your new column name here

    modelName=<MODEL\_NAME>, # Your ML model name here

    modelVersion=<MODEL\_VERSION> # Your ML model version here

)

df = model.transform(df)

df.write.format('delta').mode("overwrite").save(

    <OUTPUT\_TABLE> # Your output table filepath here

)