**Churn Modeling Analysis with MLflow: From Data to Deployment**

import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

import mlflow

import mlflow.sklearn

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import LabelEncoder, StandardScaler

from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import accuracy\_score, classification\_report

# Load Dataset

df = pd.read\_csv('/mnt/data/WA\_Fn-UseC\_-Telco-Customer-Churn.csv')

# Data Cleaning

# Dropping customerID as it's not a useful feature

df.drop(columns=['customerID'], inplace=True)

# Convert TotalCharges to numeric (some missing values)

df['TotalCharges'] = pd.to\_numeric(df['TotalCharges'], errors='coerce')

df.fillna(df.median(), inplace=True)

# Encode categorical variables

encoder = LabelEncoder()

categorical\_cols = df.select\_dtypes(include=['object']).columns

for col in categorical\_cols:

df[col] = encoder.fit\_transform(df[col])

# Feature Scaling

scaler = StandardScaler()

scaled\_cols = ['tenure', 'MonthlyCharges', 'TotalCharges']

df[scaled\_cols] = scaler.fit\_transform(df[scaled\_cols])

# Splitting Data

X = df.drop(columns=['Churn'])

y = df['Churn']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# MLflow Tracking

mlflow.set\_experiment("Churn Prediction Experiment")

models = {

"RandomForest": RandomForestClassifier(n\_estimators=100, random\_state=42),

"GradientBoosting": GradientBoostingClassifier(n\_estimators=100, random\_state=42),

"LogisticRegression": LogisticRegression(max\_iter=500)

}

for model\_name, model in models.items():

with mlflow.start\_run(run\_name=model\_name):

# Train Model

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

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

# Log Model and Metrics

accuracy = accuracy\_score(y\_test, y\_pred)

mlflow.log\_param("model\_name", model\_name)

mlflow.log\_metric("accuracy", accuracy)

mlflow.sklearn.log\_model(model, model\_name)

print(f"{model\_name} Accuracy: {accuracy}")

print(classification\_report(y\_test, y\_pred))