**Source Code**

# Import necessary libraries

# For data splitting and model evaluation

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

# For feature scaling

from sklearn.preprocessing import StandardScaler

# Random Forest model

from sklearn.ensemble import RandomForestClassifier

# Logistic Regression model

from sklearn.linear\_model import LogisticRegression

# For model evaluation metrics

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

# For handling class imbalance

from imblearn.over\_sampling import SMOTE

# For handling missing values

from sklearn.impute import SimpleImputer

# For data manipulation

import pandas as pd

# Step 1: Load the dataset

df = pd.read\_csv('churndataset.csv', delimiter=';')

# Step 2: Clean data by replacing commas with dots and converting to numeric where needed

for col in df.columns:

if df[col].dtype == 'object':

df[col] = df[col].str.replace(',', '.').astype(float, errors='ignore')

# Step 3: Separate the features (X) and target variable (y)

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

y = df['Churn']

# Convert 'Yes'/'No' in y to binary values (1 for 'Yes' and 0 for 'No')

y = y.map({'Yes': 1, 'No': 0})

# Step 4: One-hot encode categorical columns to convert them to numeric

X = pd.get\_dummies(X, drop\_first=True)

# Step 5: Handle missing values using SimpleImputer

imputer = SimpleImputer(strategy='mean')

X = imputer.fit\_transform(X)

# Step 6: Split the data into training and testing sets

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

# Step 7: Apply SMOTE for class balancing on the training set

smote = SMOTE(random\_state=42)

X\_train\_sm, y\_train\_sm = smote.fit\_resample(X\_train, y\_train)

# Step 8: Standardize the features

scaler = StandardScaler()

X\_train\_sm = scaler.fit\_transform(X\_train\_sm)

X\_test = scaler.transform(X\_test)

# Step 9: Initialize models with increased max\_iter for Logistic Regression

log\_reg = LogisticRegression(random\_state=42, max\_iter=1000) # Increased max\_iter

rand\_forest = RandomForestClassifier(random\_state=42)

# Step 10: Train and evaluate models

for model, name in zip([log\_reg, rand\_forest], ['Logistic Regression', 'Random Forest']):

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

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

print(f"\n{name} Model Performance")

print("Accuracy:", accuracy\_score(y\_test, y\_pred))

print("Confusion Matrix:\n", confusion\_matrix(y\_test, y\_pred))

print("Classification Report:\n", classification\_report(y\_test, y\_pred, target\_names=['No Churn', 'Churn']))

cv\_scores = cross\_val\_score(model, X\_train\_sm, y\_train\_sm, cv=10, scoring='f1')

print(f"{name} F1-Score (10-fold CV): {cv\_scores.mean():.3f} ± {cv\_scores.std():.3f}")