**Machine Learning Project Report: Telecom Customer Churn Prediction**

**Introduction**

**Problem Statement**

Customer churn is a serious problem for most industries, including business corporations. In the telecom sector, where business companies must hold their customers to support their revenue, the problem is even more pressing. Churn prediction is utilized to identify the customers who will probably terminate their service or subscription. Telecom businesses may use their expectations of customer churn to implement proactively retentive approaches. Specifically, this project aims to construct a machine learning model using a dataset of telecom users to predict customer churn.

**Solution Overview**

Machine learning is a good method to solve the churn prediction problem, as it looks at the past data and searching for the patterns that accompany the churn. There are several crucial steps to solve this project:

1. Data cleaning and preprocessing
2. Exploratory data analysis
3. Feature selection and engineering
4. Model selection, training, and evaluation
5. Hyperparameter tuning to get the best model performance.

**Dataset Sourcing and Evaluation**

**Dataset Description**

The “Telecom Churn Dataset” was used by the project to extract a set of records including demographic and account information and service usage data of the customers. The used dataset is characterized by the target variable and the following features: Churn, gender, senior citizenship, partner, and dependent status, tenure, phone and internet service details, contract type, and payment method was used by the project.

**Dataset Evaluation**

• Relevance: The dataset is directly relevant to the problem as it contains features that influence customer behaviour and churn.

• Size: The dataset includes sufficient data to train a robust model.

**Data Loading and Cleaning**

**Data Loading**

The dataset is loaded into a Pandas DataFrame for easy manipulation and analysis.

import pandas as pd

df = pd.read\_csv("telecom\_churn\_dataset.csv")

**Data Cleaning**

The 'TotalCharges' column's missing values are filled in with the median after the column is converted to numeric format.

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

df.fillna(df['TotalCharges'].median(), inplace=True)

print(df.isnull().sum())

**Data Exploration**

The distribution of the target variable and feature correlations are visualised as part of exploratory data analysis.

import seaborn as sns

import matplotlib.pyplot as plt

sns.countplot(x='Churn', data=df)

plt.title('Target value distribution')

plt.show()

sns.pairplot(df, hue='Churn')

plt.show()

**Model Selection and Implementation**

**Features and Target Variable**

The 'customerID' column is removed, and LabelEncoder is used to encode categorical variables.

from sklearn.preprocessing import LabelEncoder

df.drop(['customerID'], axis=1, inplace=True)

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

for col in categorical:

if col != 'Churn':

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

df['Churn'] = df['Churn'].apply(lambda x: 1 if x == 'Yes' else 0)

features = df.drop('Churn', axis=1)

target = df['Churn']

**Feature Scaling**

Scaling the features is done using StandardScaler.

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

features = scaler.fit\_transform(features)

**Data Splitting**

The dataset is split into training and testing sets.

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(features, target, test\_size=0.2, random\_state=20)

**Handling Class Imbalance**

To address the class imbalance in the training set, SMOTE is used.

from imblearn.over\_sampling import SMOTE

smote = SMOTE(random\_state=20)

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

**Model Implementation**

Two models are implemented: Logistic Regression and Random Forest.

from sklearn.linear\_model import LogisticRegression

from sklearn.ensemble import RandomForestClassifier

# Logistic Regression

logreg = LogisticRegression(class\_weight='balanced', random\_state=42)

logreg.fit(X\_train\_resampled, y\_train\_resampled)

y\_pred\_logreg = logreg.predict(X\_test)

# Random Forest

rf = RandomForestClassifier(class\_weight='balanced', random\_state=42)

rf.fit(X\_train\_resampled, y\_train\_resampled)

y\_pred\_rf = rf.predict(X\_test)

**Model Evaluation**

**Evaluation Function**

Accuracy, precision, recall, F1 score, and confusion matrix are all calculated by an evaluation function.

from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score, confusion\_matrix

def evaluate\_model(y\_true, y\_pred):

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

precision = precision\_score(y\_true, y\_pred, pos\_label=1)

recall = recall\_score(y\_true, y\_pred, pos\_label=1)

f1 = f1\_score(y\_true, y\_pred, pos\_label=1)

conf\_matrix = confusion\_matrix(y\_true, y\_pred)

return accuracy, precision, recall, f1, conf\_matrix

**Logistic Regression Evaluation**

accuracy\_logreg, precision\_logreg, recall\_logreg, f1\_logreg, conf\_matrix\_logreg = evaluate\_model(y\_test, y\_pred\_logreg)

print("Logistic Regression Evaluation:")

print("Accuracy:", accuracy\_logreg)

print("Precision:", precision\_logreg)

print("Recall:", recall\_logreg)

print("F1 Score:", f1\_logreg)

print("Confusion Matrix:\n", conf\_matrix\_logreg)

**Results:**

Accuracy: 0.7587

Precision: 0.5053

Recall: 0.8324

F1 Score: 0.6288

Confusion Matrix:

[[781, 282],

[ 58, 288]]

**Random Forest Evaluation**

accuracy\_rf, precision\_rf, recall\_rf, f1\_rf, conf\_matrix\_rf = evaluate\_model(y\_test, y\_pred\_rf)

print("\nRandom Forest Evaluation:")

print("Accuracy:", accuracy\_rf)

print("Precision:", precision\_rf)

print("Recall:", recall\_rf)

print("F1 Score:", f1\_rf)

print("Confusion Matrix:\n", conf\_matrix\_rf)

**Results:**

Accuracy: 0.7942

Precision: 0.5725

Recall: 0.6387

F1 Score: 0.6038

Confusion Matrix:

[[898, 165],

[125, 221]]

**Hyperparameter Tuning**

**Grid Search for Random Forest**

The Random Forest model's optimal hyperparameters are found using Grid Search.

from sklearn.model\_selection import GridSearchCV

param\_grid = {'n\_estimators': [100, 200, 300],

'max\_depth': [5, 10, 15],

'min\_samples\_split': [2, 5, 10]}

rf\_grid = RandomForestClassifier()

grid\_search = GridSearchCV(estimator=rf\_grid, param\_grid=param\_grid, cv=5, scoring='accuracy')

grid\_search.fit(X\_train, y\_train)

**Final Model with Best Parameters**

The optimal parameters are used to retrain the Random Forest model.

best\_rf = RandomForestClassifier(\*\*grid\_search.best\_params\_)

best\_rf.fit(X\_train\_resampled, y\_train\_resampled)

y\_pred\_best\_rf = best\_rf.predict(X\_test)

**Final Model Evaluation**

accuracy\_best\_rf, precision\_best\_rf, recall\_best\_rf, f1\_best\_rf, conf\_matrix\_best\_rf = evaluate\_model(y\_test, y\_pred\_best\_rf)

print("\nFinal Model Evaluation (Random Forest with Best Parameters):")

print("Accuracy:", accuracy\_best\_rf)

print("Precision:", precision\_best\_rf)

print("Recall:", recall\_best\_rf)

print("F1 Score:", f1\_best\_rf)

print("Confusion Matrix:\n", conf\_matrix\_best\_rf)

**Results:**

Accuracy: 0.7935

Precision: 0.5656

Recall: 0.6850

F1 Score: 0.6196

Confusion Matrix:

[[881, 182],

[109, 237]]

**Conclusion**

**Summary and Future Work**

In this project, successful implementation has been carried out to evaluate several models for predicting telecom customer churn using machine learning. Random Forest model has achieved better performance than Logistic Regression model through hyperparameter tunning; Also, the performance metrics of accuracy and other metrics have been improved using several hyperparameters that can be further developed in future work. More complex models, such as Neural Networks or the Gradient Boosting algorithm, could be considered for experimentation in the future, along with the addition of feature engineering to improve prediction. By applying the machine learning model to the environment and tracking the model’s recall and F1 score monthly, it continues to get improved.