A

Report

on

"Classify Customer Churn: Identify which customers are likely to leave a telecom company based on usage patterns."

submitted as partial fulfillment for the award of

BACHELOR OF TECHNOLOGY

degree

session 2024-25

in

CSE(AIML)

by

Name-Rinku Biswas

Roll-202401100400155

Under the supervision of

"Abhishek Shukla"

KIET GROUP OF INSTITUTIONS,GHAZIABAD

INTRODUCTION

Customer churn is a major concern in the telecom industry, where retaining users is crucial for business growth. This project aims to predict which customers are likely to leave the service based on their usage patterns and demographic details. Using machine learning—specifically an optimized XGBoost model—we analyze the data to help telecom companies take proactive steps to reduce churn and improve customer retention.

METHODOLOGY

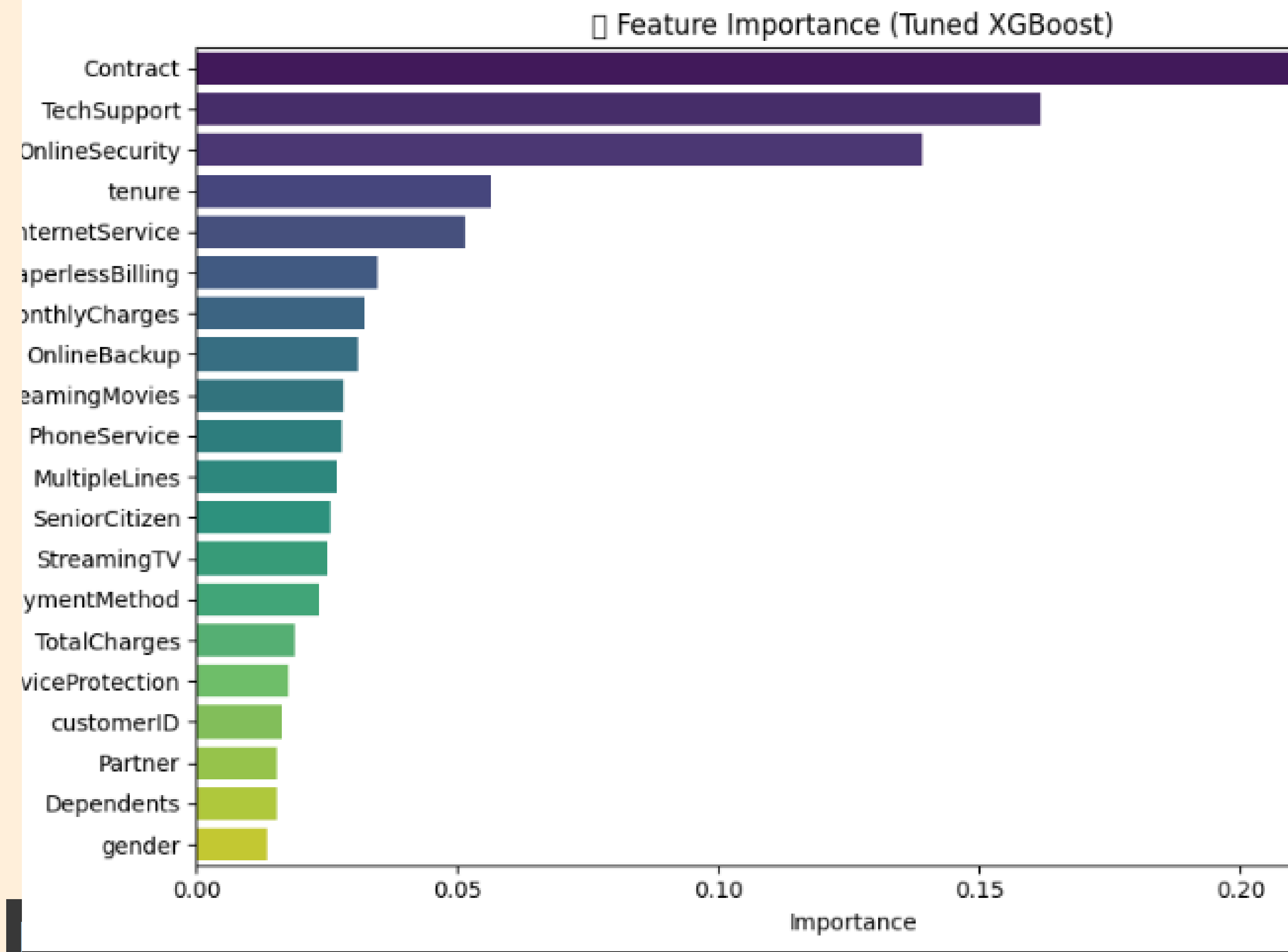
- DATA LOADING: IMPORTED THE CUSTOMER CHURN DATASET IN CSV FORMAT.
- DATA CLEANING: CHECKED AND HANDLED MISSING VALUES.
- ENCODING: USED LABEL ENCODING TO CONVERT CATEGORICAL COLUMNS INTO NUMERIC FORMAT.
- FEATURE SCALING: APPLIED STANDARDSCALER TO NORMALIZE THE FEATURES.
- TRAIN-TEST SPLIT: SPLIT DATA INTO 80% TRAINING AND 20% TESTING SETS.
- MODEL SELECTION: CHOSE XGBOOST CLASSIFIER FOR ITS ACCURACY AND PERFORMANCE.
- HYPERPARAMETER TUNING: USED RANDOMIZEDSEARCHCV TO OPTIMIZE MODEL PARAMETERS.
- EVALUATION: MEASURED MODEL PERFORMANCE USING ACCURACY, CONFUSION MATRIX, AND CLASSIFICATION REPORT.
- FEATURE IMPORTANCE: VISUALIZED IMPORTANT FEATURES INFLUENCING CHURN USING A BAR PLOT.

CODE

```
# 🔁 Train/Test split
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# 📦 Install XGBoost
!pip install xgboost

# 📥 Upload dataset
from google.colab import files
uploaded = files.upload()

# 📄 Load and preview dataset
import pandas as pd
import numpy as np

filename = list(uploaded.keys())[0]
df = pd.read_csv(filename)

print("First 5 rows:")
print(df.head())
print("\nDataset info:")
print(df.info())
print("\nMissing values:")
print(df.isnull().sum())

# 🧹 Preprocessing
from sklearn.preprocessing import LabelEncoder, StandardScaler

# Encode categorical columns
label_encoders = {}
for col in df.select_dtypes(include='object').columns:
le = LabelEncoder()
df[col] = le.fit_transform(df[col])
label_encoders[col] = le

# Define features and target
X = df.drop('Churn', axis=1) # Replace 'Churn' if your target has a different name
y = df['Churn']
# 🔄 Feature scaling
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# 🧠 XGBoost with Hyperparameter Tuning
from xgboost import XGBClassifier
from sklearn.model_selection import RandomizedSearchCV
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

xgb = XGBClassifier(use_label_encoder=False, eval_metric='logloss', random_state=42)
```

```
# Define parameter grid
param_grid = {
'n_estimators': [100, 200, 300],
'max_depth': [3, 4, 5, 6, 7],
'learning_rate': [0.01, 0.05, 0.1, 0.2],
'subsample': [0.6, 0.8, 1.0],
'colsample_bytree': [0.6, 0.8, 1.0]
}

# Randomized search
random_search = RandomizedSearchCV(
estimator=xgb,
param_distributions=param_grid,
n_iter=30,
scoring='accuracy',
cv=3,
verbose=1,
n_jobs=-1
)

# Fit the model
random_search.fit(X_train, y_train)
best_model = random_search.best_estimator_

# 🎯 Make predictions
y_pred = best_model.predict(X_test)

# 📊 Evaluation
print("\n[📊 Tuned XGBoost Evaluation]")
print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))
print("\nClassification Report:")
print(classification_report(y_test, y_pred))
print(f"✅ Accuracy Score: {accuracy_score(y_test, y_pred):.2f}")

# 📈 Feature importance plot
import matplotlib.pyplot as plt
import seaborn as sns

importances = best_model.feature_importances_
feature_names = X.columns
feat_df = pd.DataFrame({'Feature': feature_names, 'Importance': importances})
feat_df = feat_df.sort_values(by='Importance', ascending=False)

plt.figure(figsize=(10, 6))
sns.barplot(data=feat_df, x='Importance', y='Feature', palette='viridis')
plt.title("🔍 Feature Importance (Tuned XGBoost)")
plt.tight_layout()
plt.show()
```

CODE OUTPUT




```
omerID  gender  SeniorCitizen Partner Dependents  tenure P
-VHVEG  Female              0     Yes         No       1
-GNVDE    Male              0      No         No      34
-QPYBK    Male              0      No         No       2
-CFOCW    Male              0      No         No      45
-HQITU  Female              0      No         No       2

ultipleLines InternetService OnlineSecurity  ... DevicePro
hone service             DSL             No  ...
          No             DSL            Yes  ...
          No             DSL            Yes  ...
hone service             DSL            Yes  ...
          No     Fiber optic             No  ...

pport StreamingTV StreamingMovies        Contract Paperle
   No          No              No  Month-to-month
   No          No              No        One year
   No          No              No  Month-to-month
  Yes          No              No        One year
   No          No              No  Month-to-month

       PaymentMethod MonthlyCharges  TotalCharges Churn
    Electronic check          29.85         29.85    No
        Mailed check          56.95        1889.5    No
        Mailed check          53.85        108.15   Yes
transfer (automatic)          42.30       1840.75    No
    Electronic check          70.70        151.65   Yes
```

```
taset info:
lass 'pandas.core.frame.DataFrame'>
ngeIndex: 7043 entries, 0 to 7042
ta columns (total 21 columns):
   Column            Non-Null Count  Dtype
-  ------            --------------  -----
   customerID        7043 non-null   object
   gender            7043 non-null   object
   SeniorCitizen     7043 non-null   int64
   Partner           7043 non-null   object
   Dependents        7043 non-null   object
   tenure            7043 non-null   int64
   PhoneService      7043 non-null   object
   MultipleLines     7043 non-null   object
   InternetService   7043 non-null   object
   OnlineSecurity    7043 non-null   object
0  OnlineBackup      7043 non-null   object
1  DeviceProtection  7043 non-null   object
2  TechSupport       7043 non-null   object
3  StreamingTV       7043 non-null   object
4  StreamingMovies   7043 non-null   object
5  Contract          7043 non-null   object
6  PaperlessBilling  7043 non-null   object
7  PaymentMethod     7043 non-null   object
8  MonthlyCharges    7043 non-null   float64
9  TotalCharges      7043 non-null   object
0  Churn             7043 non-null   object
ypes: float64(1), int64(2), object(18)
```

```
Confusion Matrix:
[[950  86]
 [175 198]]

Classification Report:
              precision    recall  f1-score

           0       0.84      0.92      0.88
           1       0.70      0.53      0.60

    accuracy                           0.81
   macro avg       0.77      0.72      0.74
```

REFRENCES

- The dataset used for this project was provided by Mr. Shivanch Prasad, under whose guidance this project was completed.
- Various Python libraries and tools such as Pandas, Scikit-learn, XGBoost, Matplotlib, and Seaborn were used for data preprocessing, model building, and visualization.
- Concepts and techniques applied in this project are based on standard practices in machine learning and data science.