KIET
GROUP OF INSTITUTIONS
Connecting Life with Learning

Assesment Report

on

"Classify Customer Churn"

submitted as partial fulfillment for the award of

BACHELOR OF TECHNOLOGY DEGREE

SESSION 2024-25

in

CSE AIML

By

Vaishnavi Rai (202401100400205)

Under the supervision of

"Abhishek Shukla"

KIET Group of Institutions, Ghaziabad

Affiliated to

Dr. A.P.J. Abdul Kalam Technical University, Lucknow

(Formerly UPTU)

May, 2025

Introduction:

Customer churn is a key metric for businesses in sectors like telecommunications, where retaining customers is critical. This project focuses on predicting whether a customer is likely to discontinue their service using machine learning techniques. By analyzing historical data, we aim to identify the most significant factors that lead to customer churn.

Methodology:

1. **Data Collection:** The dataset titled "Classify Customer Churn" was loaded using Pandas.
2. **Data Cleaning:** Handled non-numeric 'TotalCharges' values by coercing to numeric and filling missing values with the median.
3. **Feature Selection:** Removed 'customerID' as it had no predictive value.
4. **Encoding:** Used Label Encoding to convert categorical features to numeric.
5. **Splitting:** Divided the dataset into 80% training and 20% testing sets.
6. **Scaling:** Applied StandardScaler to normalize the feature values.
7. **Modeling:** Trained a Random Forest Classifier.
8. **Evaluation:** Measured performance using Accuracy, Classification Report, and Confusion Matrix.

Code:

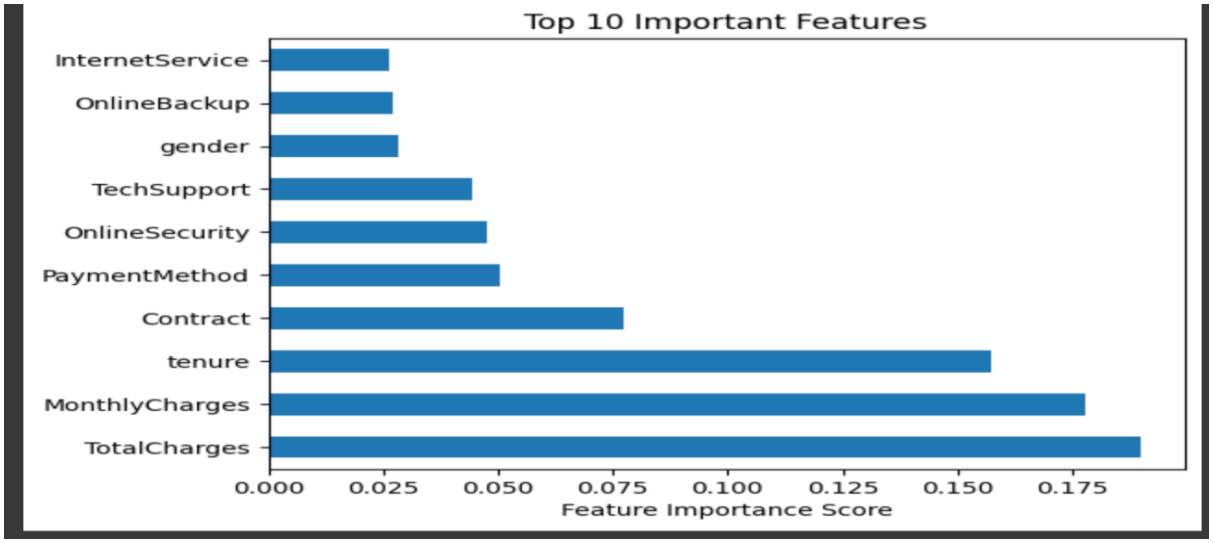
```
# Step 1: Import Required Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

# Step 2: Load the Dataset
# Make sure the file is uploaded in Colab or update the path
file_path = '/content/5. Classify Customer Churn.csv'
df = pd.read_csv(file_path)

# Step 3: Clean the Data
# 'TotalCharges' might have non-numeric values; convert to numeric and handle errors
df['TotalCharges'] = pd.to_numeric(df['TotalCharges'], errors='coerce')

# Replace missing values in 'TotalCharges' with the median of that column
df['TotalCharges'].fillna(df['TotalCharges'].median(), inplace=True)

# 'customerID' is a unique identifier and doesn't help with prediction — remove it
df.drop('customerID', axis=1, inplace=True)

# Step 4: Encode Categorical Variables
```

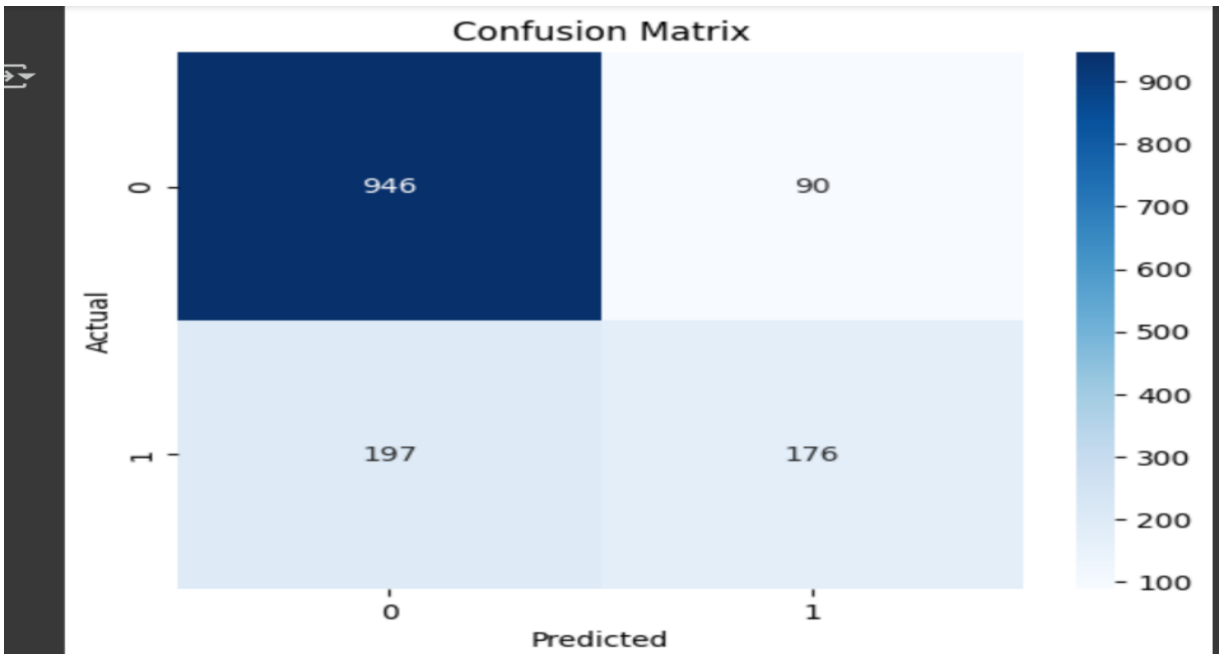
```
# Convert all categorical text columns to numeric using Label Encoding
le = LabelEncoder()
for col in df.select_dtypes(include=['object']).columns:
    df[col] = le.fit_transform(df[col])

# Step 5: Define Features (X) and Target (y)
# The target variable is 'Churn', which indicates if a customer left
X = df.drop('Churn', axis=1)  # Feature set
y = df['Churn']               # Target variable

# Step 6: Split the Dataset
# Split the data into training (80%) and testing (20%) sets
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

# Step 7: Normalize the Features
# Standardize the features to have mean=0 and std=1 (important for many ML models)
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Step 8: Train a Random Forest Classifier
# Random Forest is a robust and commonly used classifier
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train_scaled, y_train)

# Step 9: Evaluate the Model
```

```
# Predict the target for the test data
y_pred = model.predict(X_test_scaled)

# Print accuracy
print("✅ Accuracy:", accuracy_score(y_test, y_pred))

# Print precision, recall, F1-score
print("\n📋 Classification Report:")
print(classification_report(y_test, y_pred))

# Display the confusion matrix using a heatmap
print("\n🔍 Confusion Matrix:")
sns.heatmap(confusion_matrix(y_test, y_pred), annot=True, fmt='d', cmap='Blues')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
# Step 10: Show Feature Importances (Top 10)
# Display the most important features that the model used to make decisions
feature_importances = pd.Series(model.feature_importances_, index=X.columns)
feature_importances.nlargest(10).plot(kind='barh')
plt.title("Top 10 Important Features")
plt.xlabel("Feature Importance Score")
plt.show()
```

OUTPUT:

```
<ipython-input-1-fcd0d8fd936d>:29: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform t


  df['TotalCharges'].fillna(df['TotalCharges'].median(), inplace=True)
✅ Accuracy: 0.7963094393186657

📋 Classification Report:
              precision    recall  f1-score   support

           0       0.83      0.91      0.87      1036
           1       0.66      0.47      0.55       373

    accuracy                           0.80      1409
   macro avg       0.74      0.69      0.71      1409
weighted avg       0.78      0.80      0.78      1409
```





References/Credits:

- Dataset: [Kaggle or data source if applicable]
- Python Libraries: pandas, numpy, matplotlib, seaborn.