Assesement Report

on

"Classify Customer Churn:"

submitted as partial fulfillment for the award of

BACHELOR OF TECHNOLOGY DEGREE

SESSION 2024-25

in

CSE-AIML

By

Sunny Kumar (202401100400195)

Under the supervision of

"Mr. Abhishek Shukla Sir"

KIET Group of Institutions, Ghaziabad

Affiliated to

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

(Formerly UPTU) **May, 2025**

Customer Churn Classification Report

Title Page

Title: Predicting Customer Churn in the Telecom Industry using Machine Learning
Name: ***Sunny Kumar***
Roll Number: ***[202401100400195]***
Course: Artificial Intelligence
Date: ***[18-04-2025]***
Instructor: ***[Mr. Abhishek Shukla Sir]***

b. Introduction

Customer retention is one of the most critical challenges faced by companies in highly competitive industries like telecommunications. Losing customers (referred to as **customer churn**) negatively impacts business revenue, reputation, and operational efficiency.

To combat this, telecom companies leverage **Machine Learning** to proactively identify customers at risk of leaving. This project aims to develop a classification model that predicts whether a customer will churn based on their account and service usage data.

The primary objective is to:

- Understand the factors contributing to churn.
- Build a predictive model using real-world customer data.
- Evaluate its performance using appropriate metrics.

The analysis is based on a provided CSV dataset which contains anonymized customer data, including demographics, subscription details, and churn labels.

c. Methodology

To solve the churn classification problem, the following methodology was followed:

1. Data Loading

The dataset was uploaded and loaded into Google Colab using pandas. A preliminary exploration was conducted using df.head(), df.info(), and df.describe().

2. Data Preprocessing

- **Handling Missing Values:** Missing values were checked using df.isnull().sum(). If any were present, appropriate techniques like imputation or removal were applied.
- **Categorical Encoding:** Since many columns were in text (e.g., "Yes"/"No",

"Male"/"Female"), these were converted to numeric values using LabelEncoder.

- **Feature Scaling:** Numerical features were standardized using StandardScaler to ensure uniformity in feature ranges.

3. Feature and Target Separation

- **Features (X):** All columns except the target column.
- **Target (y):** The column labeled Churn, which was encoded as 0 (No) and 1 (Yes).

4. Train-Test Split

The data was split into training (80%) and testing (20%) sets using train_test_split. This allowed us to train the model and then evaluate its generalization performance.

5. Model Selection and Training

- A **Random Forest Classifier** was selected for training due to its robustness and ability to handle both categorical and numerical data.
- The model was trained using the training set and optimized with default hyperparameters.

6. Evaluation

- The trained model was evaluated using the test set.
- Evaluation metrics used: **Accuracy**, **Precision**, **Recall**, **F1-Score**, and **Confusion Matrix**.
- Additionally, **feature importance** was plotted to understand which variables most affected churn.

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 Dataset
# ------------------------------------------
df = pd.read_csv("5. Classify Customer Churn.csv")
print("First 5 rows of dataset:\n", df.head())

# ------------------------------------------
# Step 3: Explore and Clean the Data
# ------------------------------------------


# Check dataset structure
print("\nDataset Info:\n") print(df.info())


# Check missing values
print("\nMissing Values:\n", df.isnull().sum())
```

```
# Encode categorical variables le =
LabelEncoder() for column in
df.select_dtypes(include=['object']).columns:
df[column] = le.fit_transform(df[column]) # -
------------------------------------------

# Step 4: Prepare Data for Training
# -------------------------------------------


# Define features (X) and target (y)
X = df.drop('Churn', axis=1)  # Target column y
= df['Churn']


# Train-test split (80/20)
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
```

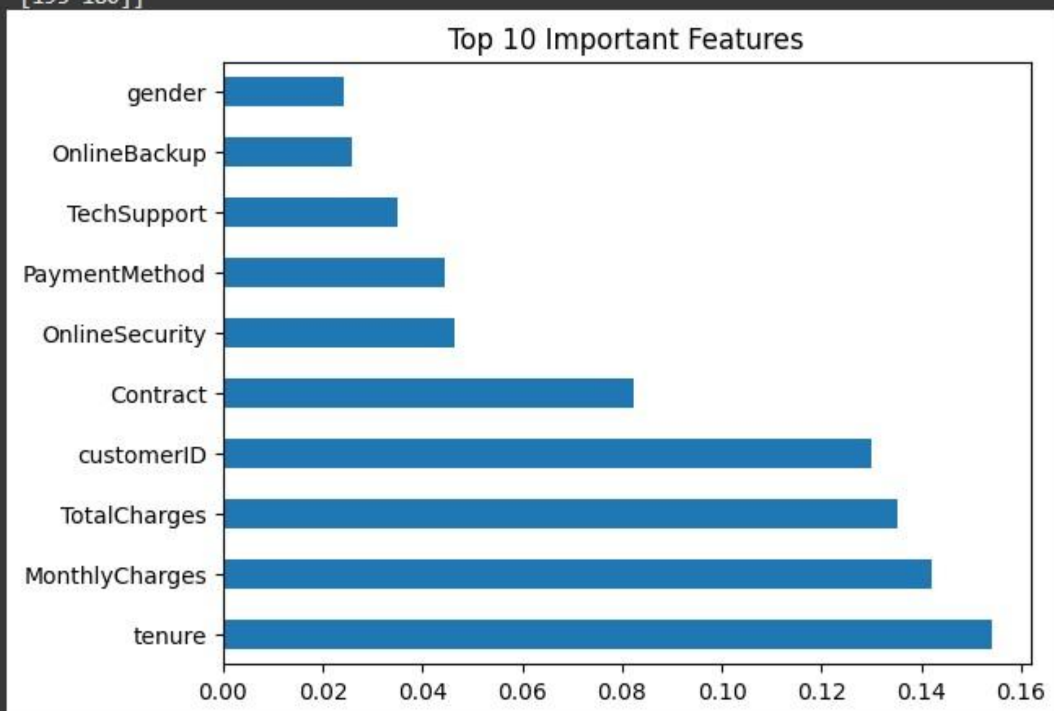
```
# Standardize the features scaler
= StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# -----------------------------------------
# Step 5: Train the Model
# -----------------------------------------

model = RandomForestClassifier(random_state=42)
model.fit(X_train, y_train)

# -----------------------------------------
# Step 6: Evaluate the Model
# -----------------------------------------
```

```
# Predictions
y_pred = model.predict(X_test)

# Evaluation Metrics
print("\n  Accuracy Score:", accuracy_score(y_test, y_pred))
print("\n  Classification Report:\n", classification_report(y_test, y_pred))
print("\n  Confusion Matrix:\n", confusion_matrix(y_test, y_pred))


# ------------------------------------------
# Step 7: Feature Importance Plot
# ------------------------------------------


# Plot top 10 most important features
feat_importances =
```

```
pd.Series(model.feature_importances_,
index=X.columns)  plt.figure(figsize=(10,
6))
feat_importances.nlargest(10).plot(kind
='barh', color='teal')
plt.title("Top 10 Important Features Influencing
Churn")
plt.xlabel("Importance Score")
plt.ylabel("Feature")
plt.grid(True)
plt.tight_layout() plt.show()
```

Output/Result

```
dtype: int64

Accuracy Score: 0.7970191625266146

Classification Report:
              precision    recall  f1-score   support

           0       0.83      0.91      0.87      1036
           1       0.66      0.48      0.56       373

    accuracy                           0.80      1409
   macro avg       0.74      0.70      0.71      1409
weighted avg       0.78      0.80      0.79      1409


Confusion Matrix:
 [[943  93]
 [193 180]]
```



```
Dataset Info:

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 21 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   customerID        7043 non-null   object 
 1   gender            7043 non-null   object 
 2   SeniorCitizen     7043 non-null   int64  
 3   Partner           7043 non-null   object 
 4   Dependents        7043 non-null   object 
 5   tenure            7043 non-null   int64  
 6   PhoneService      7043 non-null   object 
 7   MultipleLines     7043 non-null   object 
 8   InternetService   7043 non-null   object 
 9   OnlineSecurity    7043 non-null   object 
 10  OnlineBackup      7043 non-null   object 
 11  DeviceProtection  7043 non-null   object 
 12  TechSupport       7043 non-null   object 
 13  StreamingTV       7043 non-null   object 
 14  StreamingMovies   7043 non-null   object 
 15  Contract          7043 non-null   object 
 16  PaperlessBilling  7043 non-null   object 
 17  PaymentMethod     7043 non-null   object 
 18  MonthlyCharges    7043 non-null   float64
 19  TotalCharges      7043 non-null   object 
 20  Churn             7043 non-null   object 
dtypes: float64(1), int64(2), object(18)
memory usage: 1.1+ MB
None

Missing Values:
 customerID         0
gender            0
SeniorCitizen     0
Partner           0
Dependents        0
tenure            0
PhoneService      0
MultipleLines     0
InternetService   0
OnlineSecurity    0
OnlineBackup      0
DeviceProtection  0
TechSupport       0
```

f. References/Credits

- Dataset provided by instructor.
- Python libraries used: Pandas, NumPy, Scikitlearn, Matplotlib, Seaborn.
- Google Colab for execution and testing

THANK YOU