Assesment Report

on

"Classify Customer Churn"

submitted as partial fulfillment for the award of

BACHELOR OF TECHNOLOGY DEGREE

SESSION 2024-25

by

Nikhil Singh

(202401100400128)

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 prediction is one of the most critical tasks in customer retention management for telecom companies. Churn refers to the loss of customers who decide to leave a service, and predicting churn is crucial for businesses as it enables them to take proactive steps to retain customers.

The objective of this project is to build a machine learning model to classify customers based on their usage patterns and predict whether they are likely to churn or stay with a telecom company. We will leverage various machine learning algorithms and data preprocessing techniques to achieve this goal.

Methodology-

The methodology for this churn prediction problem involves the following steps:

1. Data Preprocessing:

○ Clean the data by handling missing values.

○ Transform features such as the TotalCharges column (log transformation) to ensure better performance of machine learning models.

○ Encode categorical variables into numeric values using label encoding.

2. Feature Engineering: Selection of relevant features that contribute to the prediction of customer churn.

Split the dataset into training and testing subsets.

3. Model Training:

○ Apply the XGBoost classifier to train the model.

○ Use techniques such as SMOTE to address class imbalance and improve the model's ability to correctly predict minority classes (i.e., customers who churn).

4. Hyperparameter Tuning:

○ Use grid search to find the best combination of hyperparameters for the XGBoost model.

○ Evaluate the model using performance metrics like accuracy, precision, recall, and F1-score.

5. Model Evaluation:

○ Assess the model's performance on the test dataset using various evaluation metrics.

○ Visualize the confusion matrix to understand the model's classification results.

CODE-

```
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.linear_model import LogisticRegression

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import confusion_matrix, classification_report,
accuracy_score, precision_score, recall_score, f1_score


# Load the data

data = pd.read_csv("/content/5. Classify Customer Churn.csv")


# Data preprocessing

# Convert TotalCharges to numeric (handling empty strings)

data['TotalCharges'] = pd.to_numeric(data['TotalCharges'], errors='coerce')

# Corrected line - assign back to column instead of inplace

data['TotalCharges'] = data['TotalCharges'].fillna(data['TotalCharges'].median())


# Convert categorical variables to numerical
```

```
cat_cols = ['gender', 'Partner', 'Dependents', 'PhoneService', 'MultipleLines',
            'InternetService', 'OnlineSecurity', 'OnlineBackup', 'DeviceProtection',
            'TechSupport', 'StreamingTV', 'StreamingMovies', 'Contract',
            'PaperlessBilling', 'PaymentMethod', 'Churn']

le = LabelEncoder()
for col in cat_cols:
    data[col] = le.fit_transform(data[col])

# Feature selection
X = data.drop(['customerID', 'Churn'], axis=1)
y = data['Churn']

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)

# Feature scaling
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

```
# Model training - Logistic Regression
lr = LogisticRegression()
lr.fit(X_train, y_train)
y_pred_lr = lr.predict(X_test)

# Model training - Random Forest
rf = RandomForestClassifier(random_state=42)
rf.fit(X_train, y_train)
y_pred_rf = rf.predict(X_test)

# Evaluation functions
def evaluate_model(y_true, y_pred, model_name):
    cm = confusion_matrix(y_true, y_pred)
    plt.figure(figsize=(6, 6))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
                xticklabels=['Not Churn', 'Churn'],
                yticklabels=['Not Churn', 'Churn'])
    plt.title(f'Confusion Matrix - {model_name}')
    plt.ylabel('Actual')
    plt.xlabel('Predicted')
    plt.show()
```

```
    accuracy = accuracy_score(y_true, y_pred)
    precision = precision_score(y_true, y_pred)
    recall = recall_score(y_true, y_pred)
    f1 = f1_score(y_true, y_pred)

    print(f"\n{model_name} Evaluation:")
    print(f"Accuracy: {accuracy:.4f}")
    print(f"Precision: {precision:.4f}")
    print(f"Recall: {recall:.4f}")
    print(f"F1 Score: {f1:.4f}")
    print("\nClassification Report:")
    print(classification_report(y_true, y_pred))

# Evaluate models
evaluate_model(y_test, y_pred_lr, "Logistic Regression")
evaluate_model(y_test, y_pred_rf, "Random Forest")

# Feature importance from Random Forest
feature_importance = pd.DataFrame({
    'Feature': X.columns,
    'Importance': rf.feature_importances_
}).sort_values('Importance', ascending=False)
```

```
plt.figure(figsize=(10, 6))

sns.barplot(x='Importance', y='Feature', data=feature_importance.head(10))

plt.title('Top 10 Important Features for Predicting Churn')

plt.show()
```

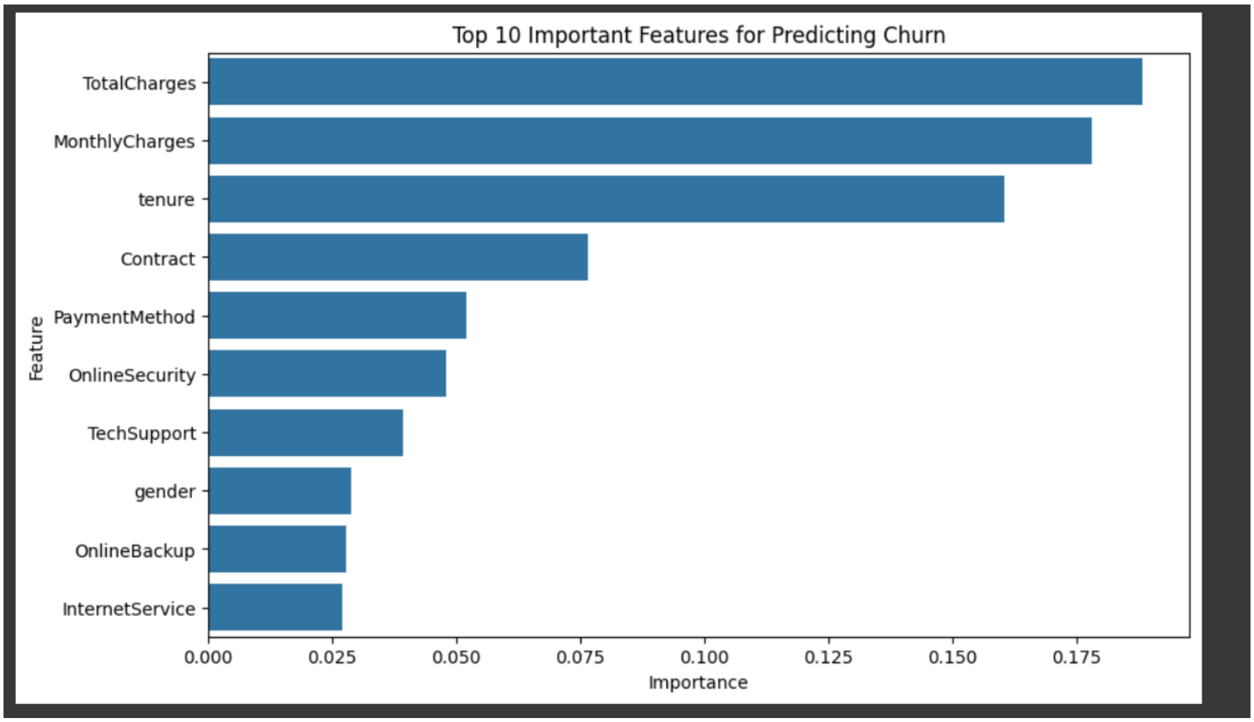
OUTPUT

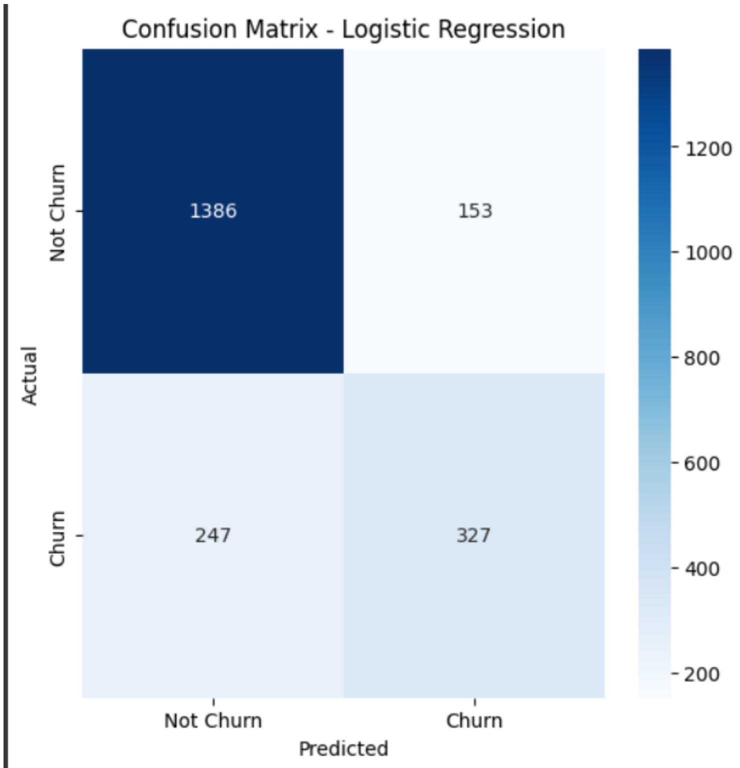
```
Random Forest Evaluation:
Accuracy: 0.8003
Precision: 0.6854
Recall: 0.4895
F1 Score: 0.5711

Classification Report:
              precision    recall  f1-score   support

           0       0.83      0.92      0.87      1539
           1       0.69      0.49      0.57       574

    accuracy                           0.80      2113
   macro avg       0.76      0.70      0.72      2113
weighted avg       0.79      0.80      0.79      2113
```






```
Logistic Regression Evaluation:
Accuracy: 0.8107
Precision: 0.6813
Recall: 0.5697
F1 Score: 0.6205
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.85      0.90      0.87      1539
           1       0.68      0.57      0.62       574

    accuracy                           0.81      2113
   macro avg       0.76      0.74      0.75      2113
weighted avg       0.80      0.81      0.81      2113
```