A

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

Shiksha Agrawal (202401100400175)

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 critical challenge faced by companies in industries with high competition, particularly in the telecom sector. Churn refers to the phenomenon where existing customers decide to discontinue their services, which can have a significant impact on a company's revenue and market share. For businesses, retaining customers is not only more cost-effective than acquiring new ones, but it also ensures long-term sustainability. As a result, predicting customer churn accurately has become a key objective for businesses seeking to improve customer retention and optimize their strategies for customer relationship management.

In the telecom industry, various factors influence customer churn, such as dissatisfaction with service quality, pricing, customer support, and the availability of better alternatives in the market. Identifying the customers most likely to churn based on these factors allows companies to take proactive actions—like offering tailored discounts, improving service quality, or providing personalized customer support—to reduce the likelihood of churn.

The problem of predicting customer churn is a classic example of a classification problem, where the goal is to classify customers into two categories: those who will stay with the company (non-churn) and those who will leave (churn). Given that customer churn is influenced by a variety of factors, the challenge lies in building an accurate predictive model that can process diverse customer information, such as demographic details, service usage patterns, account attributes, and payment history, and make reliable predictions.

In this report, we focus on solving the customer churn prediction problem using a machine learning approach. We will utilize a dataset that includes multiple customer-related attributes and apply data preprocessing techniques such as handling missing data, encoding categorical variables, and feature scaling. The goal is to develop a machine learning model using the Random Forest Classifier, which can effectively predict whether a customer is likely to churn, based on their historical data.

Through this analysis, we aim to not only predict churn accurately but also provide valuable insights to businesses in the telecom sector, enabling them to implement timely interventions to reduce churn rates and enhance customer satisfaction.

Methodology

The methodology for solving the customer churn prediction problem involves several key steps, ranging from data preprocessing to model training and evaluation. Below is a detailed explanation of each step involved in building and evaluating the churn prediction model:

1. Data Collection and Understanding

The dataset used in this project contains information about customers, including both demographic and service-related features such as age, tenure with the company The dataset is loaded from a CSV file.

2. Data Preprocessing

Data preprocessing is a crucial step in preparing the data for machine learning. The following steps are performed to clean and transform the dataset:

- **Removing the ID Column**: If an identifier column (e.g., ' customer ID ') exists, it is dropped from the dataset since it does not contribute to the prediction of churn.
- **Handling Missing Values**: Missing values in the dataset are detected and handled. In this case, any rows with missing values are dropped, ensuring that the dataset is complete and free from missing data points.
- **Converting Data Types**: Some columns, such as 'Total Charges', are initially stored as strings. These columns are converted into numeric data types to ensure proper processing in machine learning algorithms.
- **Encoding Categorical Variables**: Since machine learning algorithms require numerical input, categorical features (e.g., 'gender', 'payment method', etc.) are transformed using **Label Encoding**, which assigns a unique numerical value to each category.

3. Feature Selection and Target Variable

After preprocessing the data, we separate the features (independent variables) and the target variable (dependent variable). The target variable is the 'Churn' column, which indicates whether a customer has churned (1) or not (0). The remaining columns are considered as the features used to predict customer churn.

4. Data Splitting

The dataset is split into training and testing sets using an **80-20 split**, meaning that 80% of the data is used for training the model and the remaining 20% is used to test its performance. This allows the model to learn from the training data and then be evaluated on unseen test data to assess its generalizability.

5. Feature Scaling

Since the features in the dataset have different units and scales (e.g., 'tenure' in months versus 'monthly charges' in dollars), it is essential to standardize the features. **Standard Scaling** is

applied to ensure that all features have a mean of 0 and a standard deviation of 1. This step helps improve the performance of many machine learning algorithms by ensuring that no single feature dominates due to its scale.

6. Model Selection and Training

For this classification problem, we use the **Random Forest Classifier**, a powerful and widely-used ensemble learning algorithm. Random Forest builds multiple decision trees during training and aggregates their results to make a final prediction. The algorithm is chosen due to its ability to handle high-dimensional datasets and its robustness in preventing overfitting.

7. Model Evaluation

Once the model is trained, it is evaluated using the test set to assess its performance. The following evaluation metrics are used:

- **Accuracy**
- **Confusion Matrix**
- **Classification Report**: This report includes precision, recall, F1-score, and support for both the churn and non-churn classes. Precision measures the proportion of positive predictions that were actually correct. **8. Model Tuning and Optimization**

9. Conclusion

The performance of the trained model is analysed based on the evaluation metrics, and insights are drawn regarding the model's ability to predict customer churn.

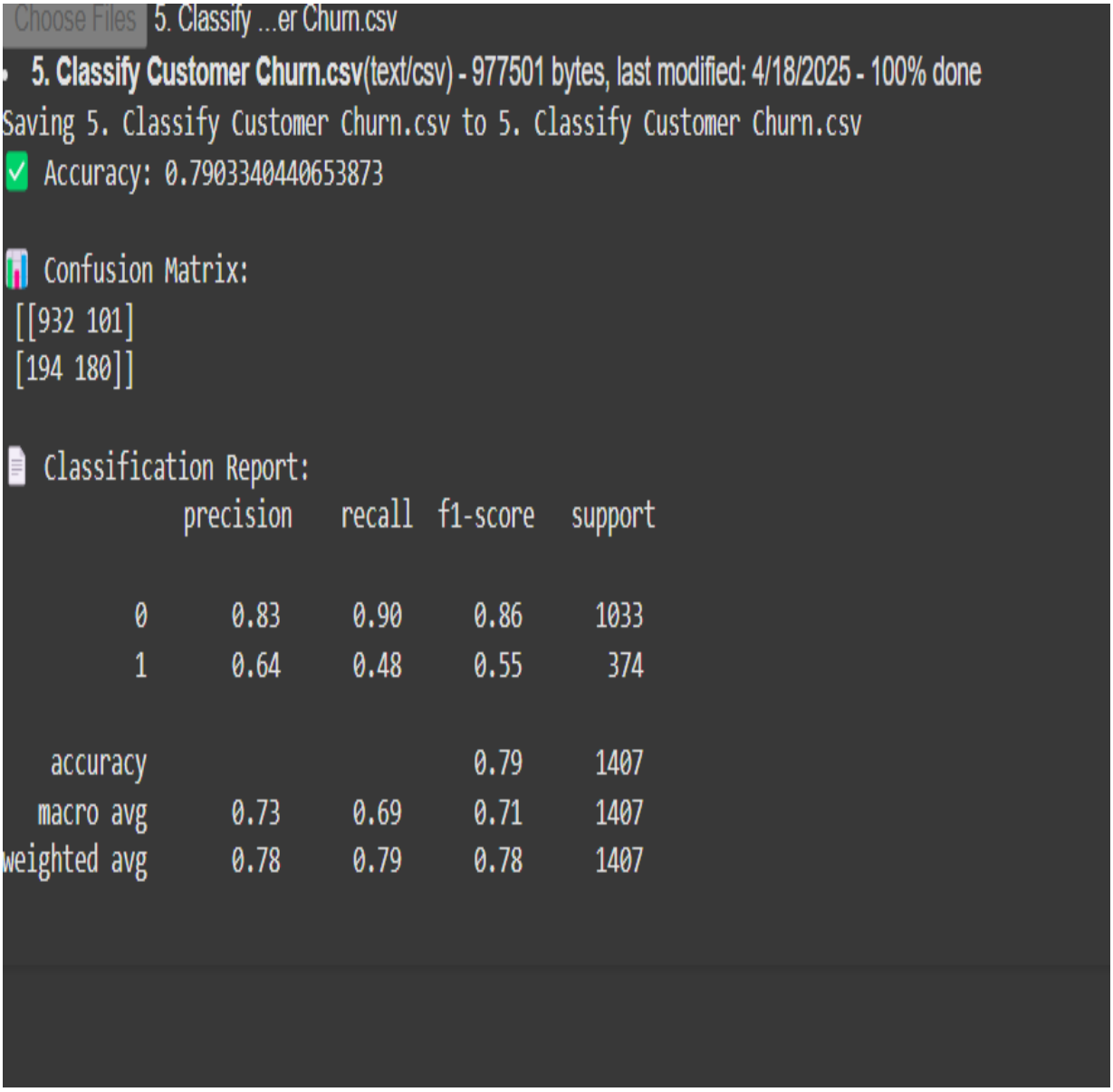
Code

```
# Step 1: Upload the CSV file
from google.colab import files
uploaded = files.upload()

# Step 2: Load the data
import pandas as pd
import io

df = pd.read_csv(io.BytesIO(uploaded['5. Classify Customer Churn.csv']))

# Step 3: Drop ID column if exists
if 'customerID' in df.columns:
    df = df.drop('customerID', axis=1)

# Step 4: Convert TotalCharges to numeric
if 'TotalCharges' in df.columns:
    df['TotalCharges'] = pd.to_numeric(df['TotalCharges'], errors='coerce')

# Step 5: Drop missing values
df = df.dropna()

# Step 6: Encode categorical variables
from sklearn.preprocessing import LabelEncoder

label_encoders = {}
for col in df.select_dtypes(include='object').columns:
    le = LabelEncoder()
```

```
    df[col] = le.fit_transform(df[col])
    label_encoders[col] = le

# Step 7: Split into features and target
X = df.drop('Churn', axis=1)
y = df['Churn']

# Step 8: 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)

# Step 9: Scale features
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Step 10: Train the model
from sklearn.ensemble import RandomForestClassifier

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

# Step 11: Predict and evaluate
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```

```
y_pred = model.predict(X_test_scaled)


print("✅ Accuracy:", accuracy_score(y_test, y_pred))
print("\n📊 Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\n📄 Classification Report:\n", classification_report(y_test, y_pred))
```

Output/Result



References

Kaggle: https://www.kaggle.com/datasets