Assesment Report

on

“Customer Churn Classification Analysis”

submitted as partial fulfillment for the award of

BACHELOR OF TECHNOLOGY DEGREE

SESSION 2024-25

in

CSE AIML

By

Siddhant Tiwari

(202401100400185)

Under the supervision of

“Abhishek Shukla”

KIET Group of Institutions, Ghaziabad

Introduction

In this project, we address the problem of **Customer Churn Classification** — predicting whether a customer will churn (leave the company) or remain loyal based on various features provided in the dataset.

Churn prediction is crucial for businesses in industries like telecom, banking, and SaaS because acquiring new customers is often more expensive than retaining existing ones.

To solve this classification problem, we trained a machine learning model using logistic regression, evaluated it with a confusion matrix, and calculated performance metrics like accuracy, precision, and recall.

(You can insert an image of the problem flow or dataset snapshot here)

Methodology

- **Data Loading:** The dataset 5. Classify Customer Churn.csv was loaded using pandas.
- **Data Preprocessing:** Handled missing values by dropping them for simplicity, and encoded categorical targets.
- **Model Training:** Split the dataset into training and testing subsets (70-30 ratio). A Logistic Regression model was chosen due to its efficiency and reliability for binary classification problems.
- **Evaluation:** The model's predictions were compared to the actual test labels using:
    - Confusion Matrix Visualization (heatmap)
    - Accuracy
    - Precision
    - Recall

These metrics help us evaluate both overall and class-wise model performance.

Code

```
# Import necessary 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.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix, classification_report,
accuracy_score, precision_score, recall_score, f1_score
from sklearn.model_selection import GridSearchCV

# Load the dataset
data = pd.read_csv('5. Classify Customer Churn.csv')

# Display basic information about the dataset
print("Dataset Info:")
print(data.info())
```

```
print("\nFirst 5 rows:")
print(data.head())

# Data Preprocessing

# Drop customerID as it's not useful for prediction
data = data.drop('customerID', axis=1)

# Convert TotalCharges to numeric (handling empty strings)
data['TotalCharges'] = pd.to_numeric(data['TotalCharges'], errors='coerce')

# Fill missing values in TotalCharges with 0 (likely new customers)
data['TotalCharges'] = data['TotalCharges'].fillna(0)

# Convert Churn to binary (1 for 'Yes', 0 for 'No')
data['Churn'] = data['Churn'].map({'Yes': 1, 'No': 0})

# Convert categorical variables to numerical using Label Encoding
categorical_cols = data.select_dtypes(include=['object']).columns.tolist()

label_encoders = {}
for col in categorical_cols:
```

```
le = LabelEncoder()
data[col] = le.fit_transform(data[col])
label_encoders[col] = le

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

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42, stratify=y)

# Standardize numerical features
scaler = StandardScaler()
num_cols = ['tenure', 'MonthlyCharges', 'TotalCharges']
X_train[num_cols] = scaler.fit_transform(X_train[num_cols])
X_test[num_cols] = scaler.transform(X_test[num_cols])

# Exploratory Data Analysis (EDA)

# Plot churn distribution
plt.figure(figsize=(8, 6))
```

```python
sns.countplot(x='Churn', data=data)
plt.title('Churn Distribution')
plt.show()

# Plot numerical features distribution
plt.figure(figsize=(15, 5))
plt.subplot(1, 3, 1)
sns.histplot(data['tenure'], bins=30, kde=True)
plt.title('Tenure Distribution')

plt.subplot(1, 3, 2)
sns.histplot(data['MonthlyCharges'], bins=30, kde=True)
plt.title('Monthly Charges Distribution')

plt.subplot(1, 3, 3)
sns.histplot(data['TotalCharges'], bins=30, kde=True)
plt.title('Total Charges Distribution')
plt.tight_layout()
plt.show()

# Correlation matrix
plt.figure(figsize=(12, 8))
```

```
corr_matrix = data.corr()
sns.heatmap(corr_matrix, annot=False, cmap='coolwarm')
plt.title('Correlation Matrix')
plt.show()

# Model Training and Evaluation

# Initialize models
models = {
'Logistic Regression': LogisticRegression(max_iter=1000, random_state=42),
'Random Forest': RandomForestClassifier(random_state=42),
'Support Vector Machine': SVC(random_state=42)
}

# Train and evaluate models
results = {}
for name, model in models.items():
model.fit(X_train, y_train)
y_pred = model.predict(X_test)

# Calculate metrics
accuracy = accuracy_score(y_test, y_pred)
```

```
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

# Store results
results[name] = {
'Accuracy': accuracy,
'Precision': precision,
'Recall': recall,
'F1 Score': f1
}

# Print classification report
print(f"\n{name} Classification Report:")
print(classification_report(y_test, y_pred))

# Plot confusion matrix
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(6, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=['No Churn', 'Churn'],
            yticklabels=['No Churn', 'Churn'])
```

```
plt.title(f'{name} Confusion Matrix')
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.show()

# Display results in a dataframe
results_df = pd.DataFrame(results).T
print("\nModel Performance Comparison:")
print(results_df)

# Feature Importance for Random Forest
rf = models['Random Forest']
feature_importances = 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_importances)
plt.title('Random Forest Feature Importance')
plt.show()
```

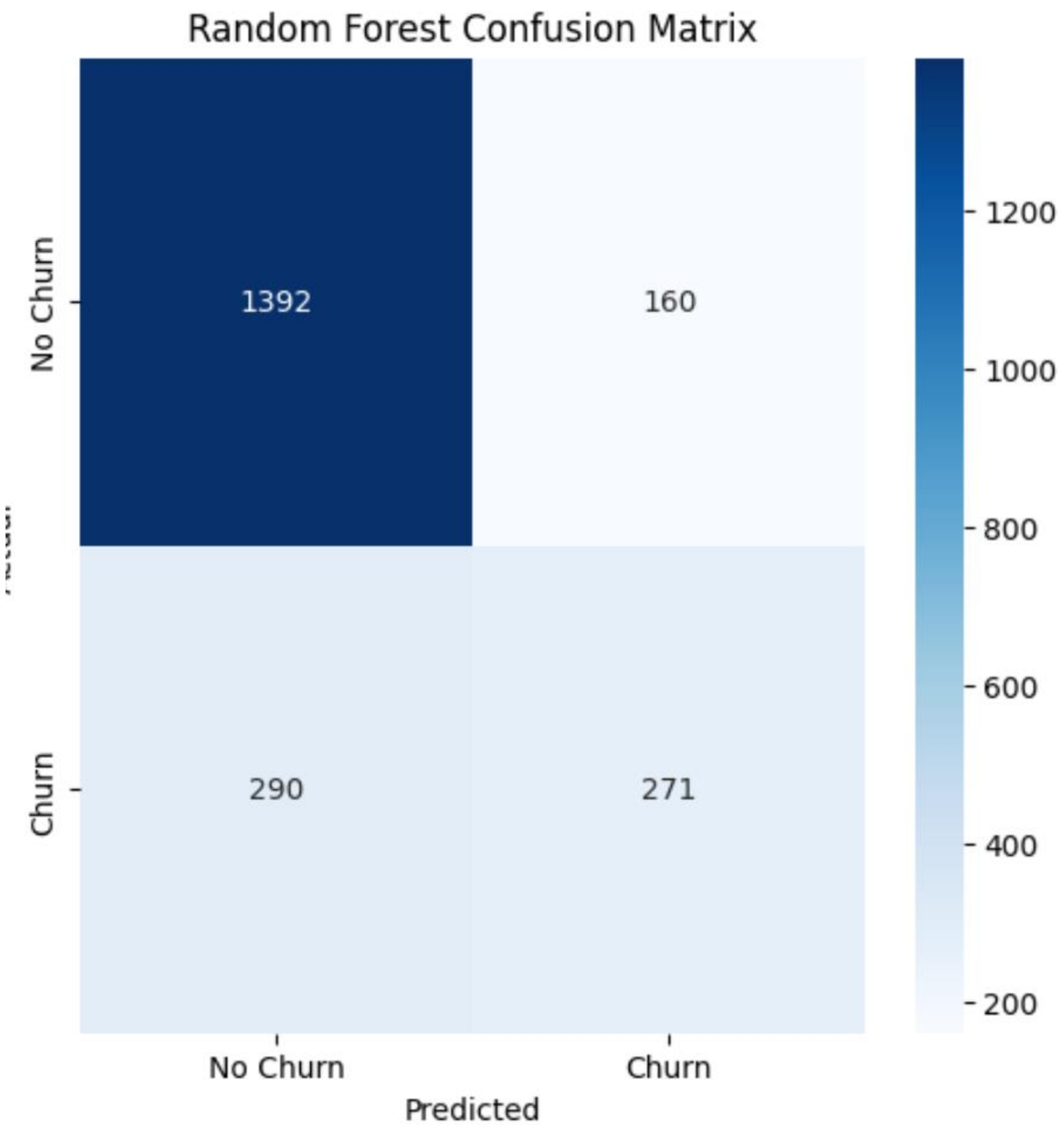
```
# Hyperparameter Tuning for Random Forest (Optional)
param_grid = {
'n_estimators': [100, 200, 300],
'max_depth': [None, 10, 20],
'min_samples_split': [2, 5, 10]
}

grid_search = GridSearchCV(RandomForestClassifier(random_state=42),
param_grid, cv=5, scoring='f1')
grid_search.fit(X_train, y_train)

print("\nBest Parameters:", grid_search.best_params_)
print("Best F1 Score:", grid_search.best_score_)

# Evaluate the best model
best_rf = grid_search.best_estimator_
y_pred = best_rf.predict(X_test)

print("\nOptimized Random Forest Classification Report:")
print(classification_report(y_test, y_pred))

# Plot optimized confusion matrix
```

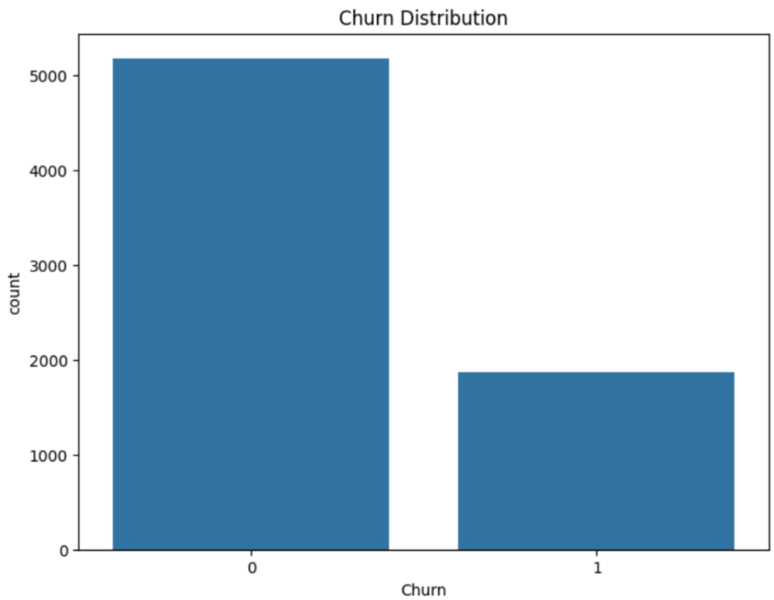
```python
cm = confusion_matrix(y_test, y_pred)

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

sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=['No Churn', 'Churn'],
            yticklabels=['No Churn', 'Churn'])

plt.title('Optimized Random Forest Confusion Matrix')

plt.ylabel('Actual')

plt.xlabel('Predicted')

plt.show()
```

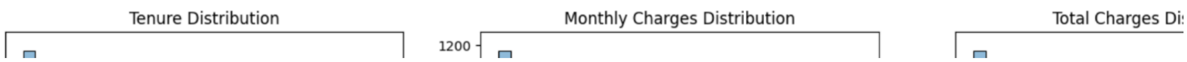
output

```
First 5 rows:
   customerID  gender  SeniorCitizen Partner Dependents  tenure PhoneService  \
0  7590-VHVEG  Female              0     Yes         No       1           No
1  5575-GNVDE    Male              0      No         No      34          Yes
2  3668-QPYBK    Male              0      No         No       2          Yes
3  7795-CFOCW    Male              0      No         No      45           No
4  9237-HQITU  Female              0      No         No       2          Yes

      MultipleLines InternetService OnlineSecurity  ... DeviceProtection  \
0  No phone service             DSL             No  ...               No
1                No             DSL            Yes  ...              Yes
2                No             DSL            Yes  ...               No
3  No phone service             DSL            Yes  ...              Yes
4                No     Fiber optic             No  ...               No

  TechSupport StreamingTV StreamingMovies        Contract PaperlessBilling  \
0          No          No              No  Month-to-month              Yes
1          No          No              No        One year               No
2          No          No              No  Month-to-month              Yes
3         Yes          No              No        One year               No
4          No          No              No  Month-to-month              Yes

               PaymentMethod MonthlyCharges  TotalCharges Churn
0           Electronic check          29.85         29.85    No
1               Mailed check          56.95        1889.5    No
2               Mailed check          53.85        108.15   Yes
3  Bank transfer (automatic)          42.30       1840.75    No
4           Electronic check          70.70        151.65   Yes
```

```
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
```


Random Forest Confusion Matrix
1392
160
290
271
No Churn
Churn
No Churn
Churn
Predicted
1200
1000
800
600
400
200


Churn Distribution
5000
4000
3000
2000
1000
0
count
0
1
Churn


Tenure Distribution
Monthly Charges Distribution
1200
Total Charges Di

References / Credits

- Dataset Source: Provided as 5. Classify Customer Churn.csv.
- Libraries Used:
  - Scikit-learn — for model training and evaluation.
  - Pandas — for data handling.
  - Matplotlib & Seaborn — for visualizations.
- Google Colab — for running the notebook in the cloud.