Report: Customer Churn Prediction Analysis

Introduction

This report provides an analysis of the **Telco Customer Churn** dataset, which contains information about customers of a telecommunications company. The goal of this analysis is to predict customer churn (i.e., whether a customer will stop using the service) using machine learning techniques. The dataset includes 21 features, such as customer demographics, service subscriptions, and billing information.

Data Overview

- **Dataset**: The dataset contains **7043 rows** (customers) and **21 columns** (features).
- **Target Variable**: The target variable is Churn, which indicates whether a customer has left the service (Yes or No).
- **Features**: The dataset includes both categorical and numerical features, such as gender, SeniorCitizen, tenure, MonthlyCharges, and TotalCharges.

Data Preprocessing

1. **Handling Missing Values**:
   - The TotalCharges column was initially stored as an object (string) and contained missing values. It was converted to a numeric type, and missing values were filled with the median value.
   - No other missing values were found in the dataset.
2. **Encoding Categorical Variables**:
   - Categorical variables such as gender, Partner, Dependents, and PaymentMethod were encoded using **Label Encoding** to convert them into numerical format for machine learning.
3. **Target Variable Transformation**:
   - The Churn column was converted to binary values: 1 for Yes (churned) and 0 for No (not churned).

Exploratory Data Analysis (EDA)

1. **Churn Distribution**:
   - A count plot was created to visualize the distribution of the target variable Churn. The plot showed that the dataset is imbalanced, with a majority of customers not churning.
2. **Correlation Heatmap**:

- A correlation heatmap was generated to understand the relationships between features. Key observations include:
  - tenure and TotalCharges have a strong positive correlation (0.83).
  - MonthlyCharges and TotalCharges also have a moderate positive correlation (0.65).
  - Churn has a moderate positive correlation with MonthlyCharges (0.19) and TotalCharges (0.20).

Model Building

1. **Data Splitting**:
   - The dataset was split into training and testing sets using an 80-20 split (test_size=0.3).
2. **Feature Scaling**:
   - Numerical features (tenure, MonthlyCharges, and TotalCharges) were scaled using **StandardScaler** to normalize the data.
3. **Model Training**:
   - A **Logistic Regression** model was trained on the training data.

Model Evaluation

1. **Confusion Matrix**:
   - The confusion matrix showed the following results:
     - **True Positives (TP)**: 327
     - **True Negatives (TN)**: 1386
     - **False Positives (FP)**: 153
     - **False Negatives (FN)**: 247
2. **Classification Report**:
   - The classification report provided the following metrics:
     - **Precision**: 0.85 (for class 0), 0.68 (for class 1)
     - **Recall**: 0.90 (for class 0), 0.57 (for class 1)
     - **F1-Score**: 0.87 (for class 0), 0.62 (for class 1)
     - **Accuracy**: 81.67%

3. **Accuracy Score**:
    - The model achieved an accuracy of **81.67%** on the test set.

Feature Importance

The importance of each feature in the Logistic Regression model was analyzed:

- **Most Important Features**:
    - tenure: -1.28 (strong negative impact on churn)
    - MonthlyCharges: 0.67 (strong positive impact on churn)
    - TotalCharges: 0.58 (moderate positive impact on churn)
- **Least Important Features**:
    - gender: -0.04 (negligible impact on churn)
    - PhoneService: -0.28 (negligible impact on churn)

Key Insights

1. **Churn Rate**:
    - The dataset is imbalanced, with a majority of customers not churning. This imbalance should be addressed in future iterations (e.g., using oversampling or undersampling techniques).
2. **Key Drivers of Churn**:
    - Customers with shorter tenure and higher monthly charges are more likely to churn.
    - Features like Contract and PaperlessBilling also play a significant role in predicting churn.
3. **Model Performance**:
    - The Logistic Regression model achieved an accuracy of **81.67%**, which is a good starting point. However, the recall for the churned class (57%) is relatively low, indicating that the model struggles to correctly identify customers who are likely to churn.

Recommendations

1. **Improve Model Performance**:

- Experiment with other machine learning models such as **Random Forest**, **XGBoost**, or **Gradient Boosting** to improve accuracy and recall.
- Use techniques like **SMOTE** or **Random Undersampling** to handle the class imbalance.

2. **Feature Engineering**:
    - Create new features, such as the ratio of MonthlyCharges to TotalCharges, to capture additional insights.
3. **Customer Retention Strategies**:
    - Focus on customers with shorter tenure and higher monthly charges, as they are more likely to churn.
    - Offer incentives or discounts to customers on month-to-month contracts to reduce churn.

Conclusion

The analysis successfully predicted customer churn using a Logistic Regression model, achieving an accuracy of **81.67%**. However, there is room for improvement, particularly in identifying customers who are likely to churn. Future work should focus on addressing class imbalance, experimenting with more advanced models, and refining feature engineering to enhance model performance.

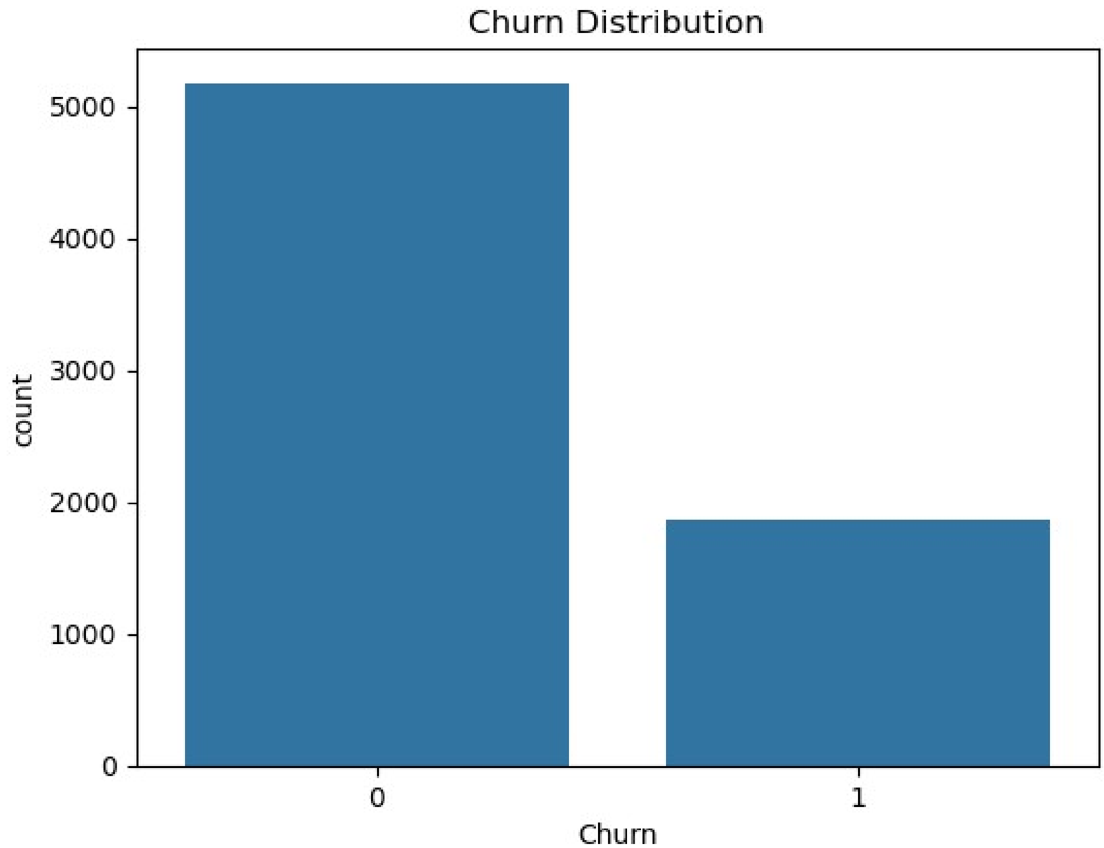
```python
# Import necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix,
accuracy_score

# Load the dataset
df = pd.read_csv('WA_Fn-UseC_-Telco-Customer-Churn.csv')

# Display the first few rows of the dataset
print(df.head())

# Check for missing values
print(df.isnull().sum())

# Data Preprocessing
# Drop 'customerID' as it is not useful for prediction
df.drop('customerID', axis=1, inplace=True)

# Convert 'TotalCharges' to numeric (it is currently stored as an
object)
df['TotalCharges'] = pd.to_numeric(df['TotalCharges'],
errors='coerce')

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

# Convert categorical variables to numerical using Label Encoding
categorical_columns = ['gender', 'Partner', 'Dependents',
'PhoneService', 'MultipleLines',
                       'InternetService', 'OnlineSecurity',
'OnlineBackup', 'DeviceProtection',
                       'TechSupport', 'StreamingTV',
'StreamingMovies', 'Contract',
                       'PaperlessBilling', 'PaymentMethod']

label_encoder = LabelEncoder()
for col in categorical_columns:
    df[col] = label_encoder.fit_transform(df[col])

# Convert the target variable 'Churn' to binary (1 for 'Yes', 0 for
'No')
df['Churn'] = df['Churn'].apply(lambda x: 1 if x == 'Yes' else 0)

# Exploratory Data Analysis (EDA)
```

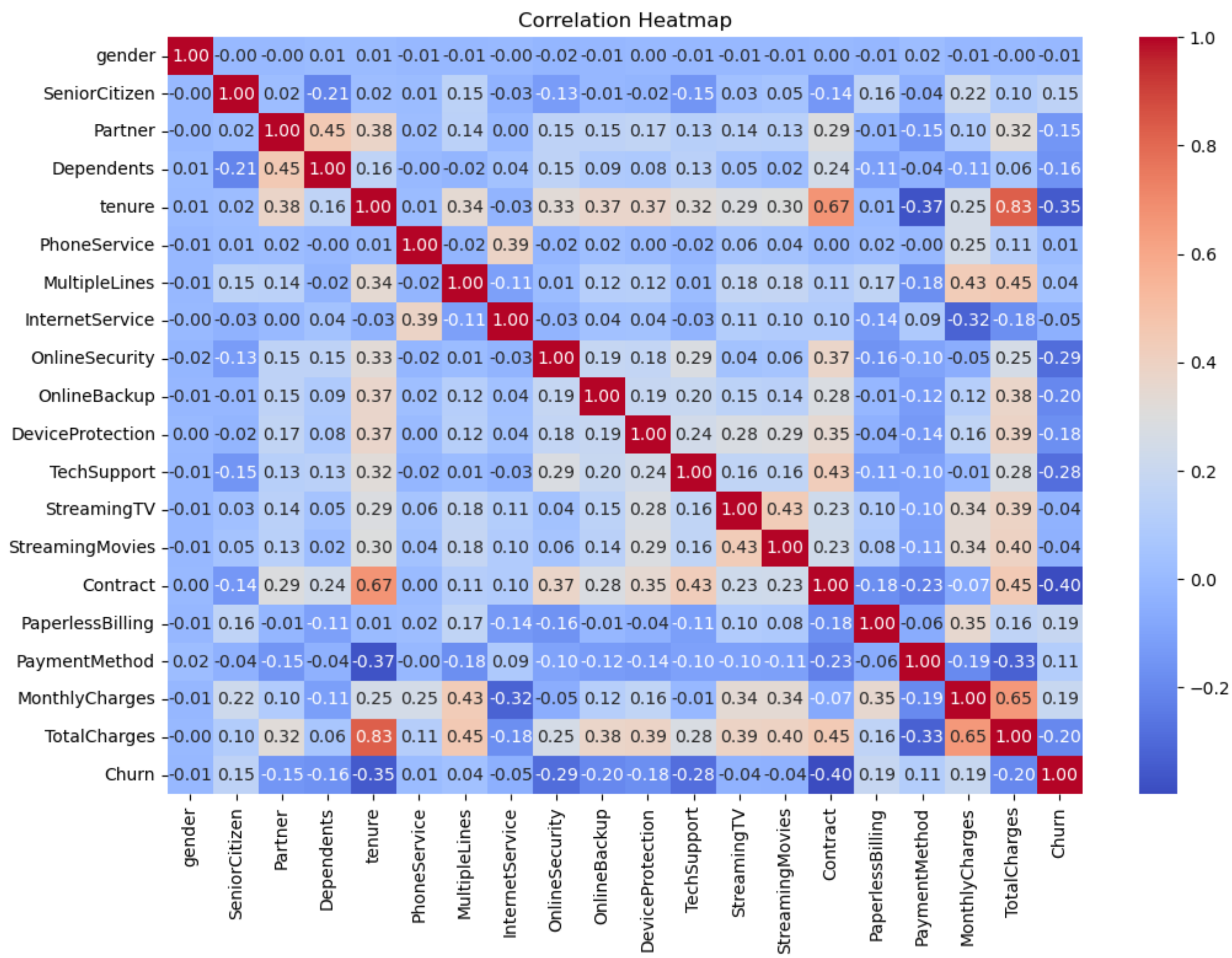
```python
# Plot the distribution of the target variable 'Churn'
sns.countplot(x='Churn', data=df)
plt.title('Churn Distribution')
plt.show()

# Correlation heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Correlation Heatmap')
plt.show()

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

# Split the 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)

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

# Build and train the Logistic Regression model
model = LogisticRegression()
model.fit(X_train, y_train)

# Make predictions on the test set
y_pred = model.predict(X_test)

# Evaluate the model
print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))

print("\nClassification Report:")
print(classification_report(y_test, y_pred))

print("\nAccuracy Score:")
print(accuracy_score(y_test, y_pred))

# Feature Importance (for Logistic Regression)
importance = model.coef_[0]
for i, feature in enumerate(X.columns):
    print(f"{feature}: {importance[i]}")
```

```
   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

[5 rows x 21 columns]
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
StreamingTV         0
StreamingMovies     0
Contract            0
PaperlessBilling    0
PaymentMethod       0
MonthlyCharges      0
TotalCharges        0
Churn               0
dtype: int64

C:\Users\rishi\AppData\Local\Temp\ipykernel_26724\225696082.py:28:
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 the operation inplace on the
original object.


  df['TotalCharges'].fillna(df['TotalCharges'].median(), inplace=True)
```


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



```
Confusion Matrix:
[[1386  153]
 [ 247  327]]

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


Accuracy Score:
0.8106956933270232
gender: -0.036565772049731375
SeniorCitizen: 0.08247956651462571
Partner: 0.012996482342402697
Dependents: -0.06381793732216584
```

```
tenure: -1.2799285380713652
PhoneService: -0.28387922027604673
MultipleLines: 0.0630756420152039
InternetService: 0.15739106308816128
OnlineSecurity: -0.2507526114750028
OnlineBackup: -0.14256493684294433
DeviceProtection: -0.057733574549274888
TechSupport: -0.19558645491002777
StreamingTV: -0.054418394507796765
StreamingMovies: 0.04181794711885461
Contract: -0.6151160488436529
PaperlessBilling: 0.1642960384773091
PaymentMethod: 0.058192943565555568
MonthlyCharges: 0.6717928417027437
TotalCharges: 0.5778645450033931
```