Predicting Customer Churn in a Telecommunications Company

Introduction:

The Telco Customer Churn dataset offers a rich resource for understanding customer behavior in the telecommunication industry. This dataset, encompassing 7,043 customers and 21 features, provides valuable insights into factors that can influence customer loyalty and ultimately lead to churn.

The dataset delves into various customer demographics, service subscriptions, account details, and churn status. Here's a breakdown of some key features:

- **Customer Demographics:** Information such as gender, senior citizen status, presence of a partner, and dependents helps paint a picture of the customer base.

- **Service Subscriptions:** Features like phone service, multiple lines, internet service, and subscriptions to additional services like online security, backup, device protection, tech support, streaming TV, and movies reveal the services customers utilize.

- **Account Details:** The dataset includes details on customer tenure Features like contract type (month-to-month, one-year, two-year), billing preferences (paperless vs. paper), and payment method can shed light on customer commitment and potential points of friction.

- **Financial Information:** Monthly charges and total charges show a picture of customer spending habits and their potential value to the company.

- **Churn Status:** This crucial feature indicates whether a customer churned (left the company) within the last month.

Overall, the Telco Customer Churn dataset serves as a powerful tool for telecommunication companies to understand their customer base, predict churn, and ultimately improve customer retention. By leveraging this data, companies can make data-driven decisions to reduce churn, maintain a loyal customer base, and drive business growth.

Data preprocessing :

The Telco Customer Churn dataset, stored in CSV format, holds valuable customer information.

1. **Importing Libraries and Loading Data:**

```python
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
import seaborn as sns

from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier

from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
```

2. Exploring the Data:

```
[2]: df = pd.read_csv('WA_Fn-UseC_-Telco-Customer-Churn.csv')

df.head()
```

Output:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceProtection	TechSu
0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service	DSL	No	...	No	
1	5575-GNVDE	Male	0	No	No	34	Yes	No	DSL	Yes	...	Yes	
2	3668-QPYBK	Male	0	No	No	2	Yes	No	DSL	Yes	...	No	
3	7795-CFOCW	Male	0	No	No	45	No	No phone service	DSL	Yes	...	Yes	
4	9237-HQITU	Female	0	No	No	2	Yes	No	Fiber optic	No	...	No	

5 rows × 21 columns

3. Handling Missing Values:

```
[4]: df.info()
```

Output:

```
<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
```

```
[6]: df.eq(" ").sum()
```

Output:

```
[6]:  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       11
      Churn               0
      dtype: int64
```

using the Impute method to fill the missing values:

```
total_charge_mean = df['TotalCharges'].mean()
print(total_charge_mean)
df['TotalCharges'].fillna(total_charge_mean, inplace=True)
df.isnull().sum()
```

Output:

```
2281.9169281556156
C:\Users\Harish23\AppDat
The behavior will change

For example, when doing


  df['TotalCharges'].fil
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
```

4. **Encoding Categorical Features:**
   The Categorical Features are the LabelEncoder library used to change the object data type to Numeric data type

```
from sklearn.preprocessing import LabelEncoder

for column in dummy.columns:
    if dummy[column].dtype == 'object':
        le = LabelEncoder()
        dummy[column] = le.fit_transform(dummy[column])
```

```
dummy
```

Output:

	gender	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	DeviceProtection	TechSupport	Strea
0	0	1	0	1	0	1	0	0	2	0	0	
1	1	0	0	34	1	0	0	2	0	2	0	
2	1	0	0	2	1	0	0	2	2	0	0	
3	1	0	0	45	0	1	0	2	0	2	2	
4	0	0	0	2	1	0	1	0	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	
7038	1	1	1	24	1	2	0	2	0	2	2	
7039	0	1	1	72	1	2	1	0	2	2	0	
7040	0	1	1	11	0	1	0	2	0	0	0	
7041	1	1	0	4	1	2	1	0	0	0	0	
7042	1	0	0	66	1	0	1	2	0	2	2	

7043 rows × 19 columns

5. **Data visualization:**

After preprocessing the Telco Customer Churn dataset and converting categorical features using LabelEncoder, we can leverage Matplotlib and Seaborn for data visualization to gain insights into customer churn.

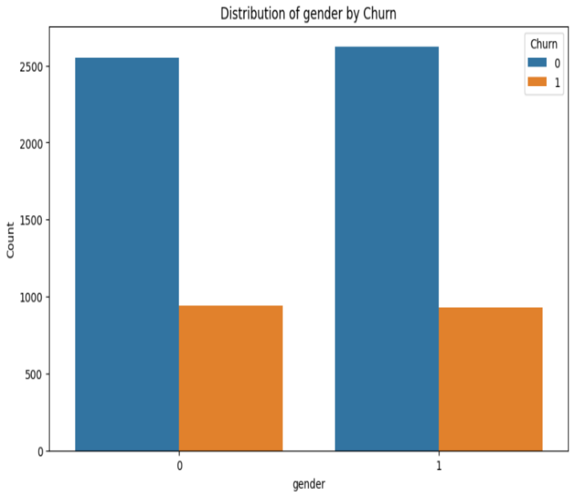
Visualizing Customer Distribution:

- **Distribution of Churn:** Using a bar chart (Matplotlib) or a count plot (Seaborn) to visualize the distribution of churn (Yes vs. No). This reveals the overall churn rate and any potential imbalance in the data.

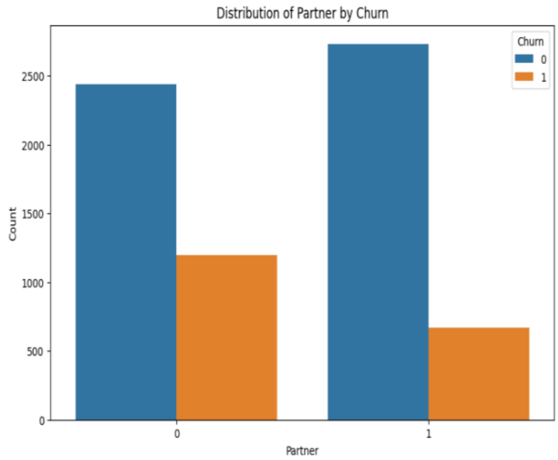
I) Categorical feature:

```
for column in dummy.columns:
    if column not in ['customerID','tenure','MonthlyCharges','TotalCharges','Churn']:
        plt.figure(figsize=(10, 6))
        sns.countplot(data=dummy, x=column, hue='Churn')
        plt.title(f'Distribution of {column} by Churn')
        plt.xlabel(column)
        plt.ylabel('Count')
        plt.legend(title='Churn')
        plt.show()
```

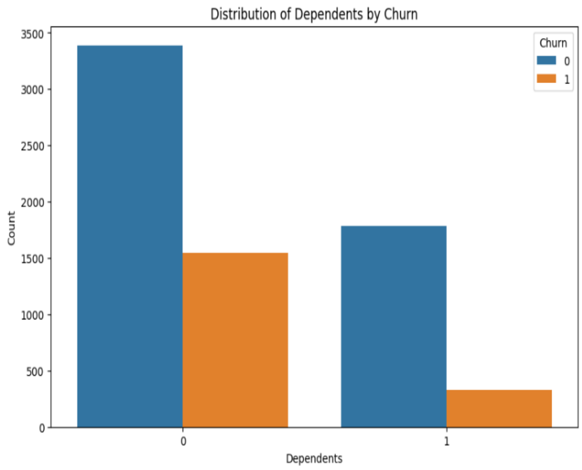
Output:



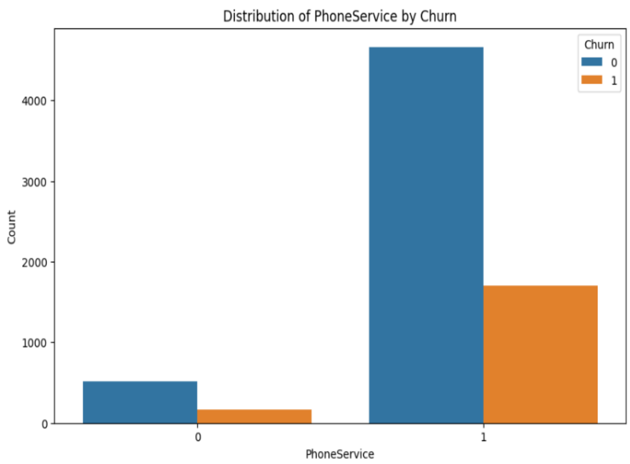
Gender



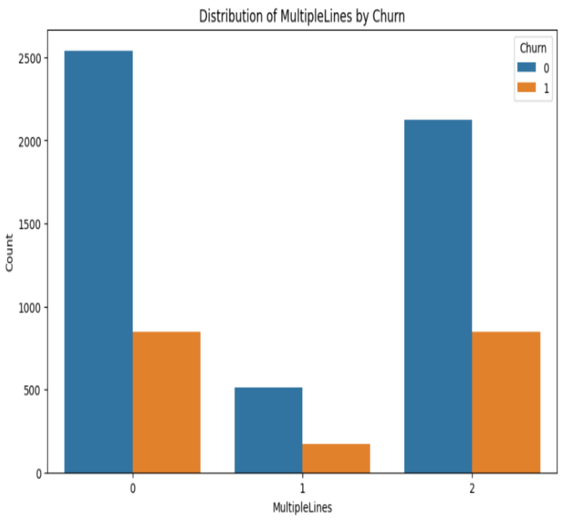
Partner



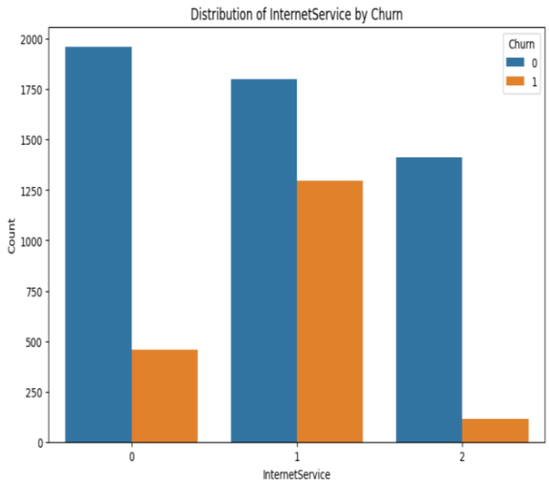
Dependents



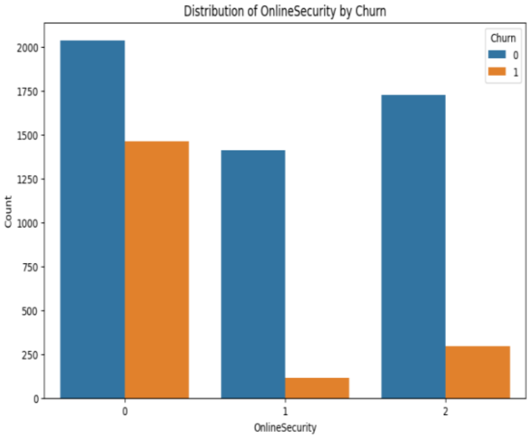
PhoneService



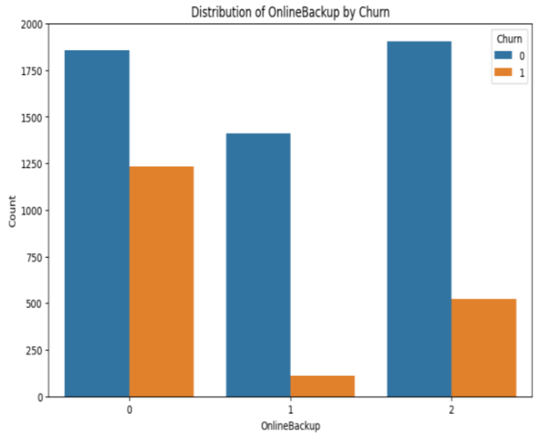
MultipleLines



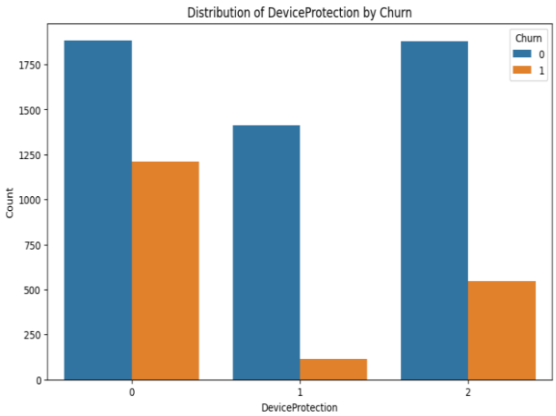
InternetService



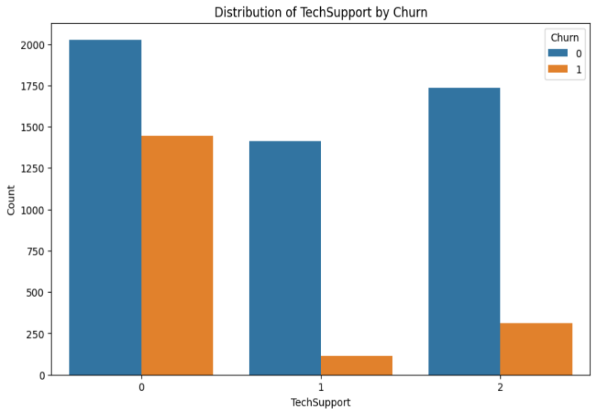
OnlineSecurity



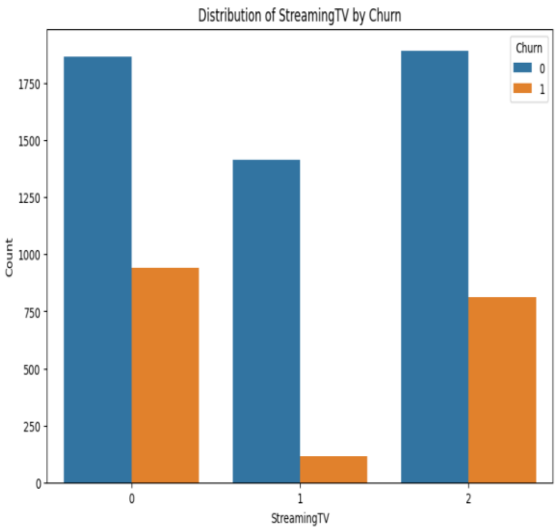
OnlineBackup



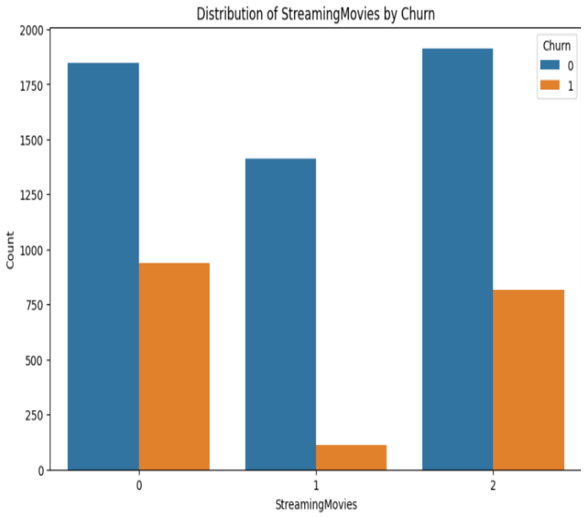
DeviceProtection



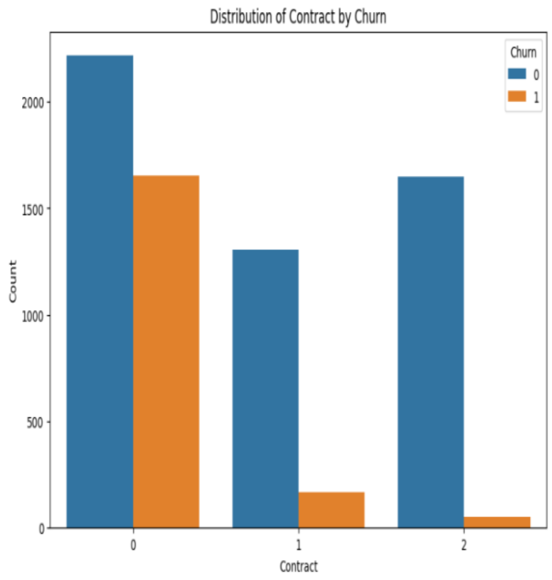
TechSupport



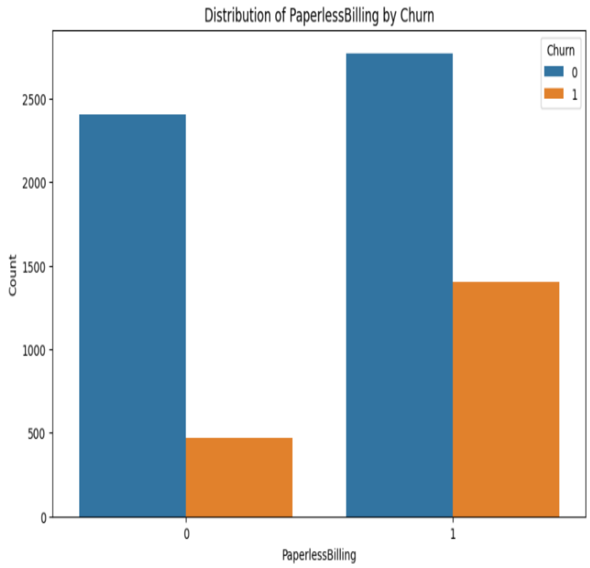
StreamingTV



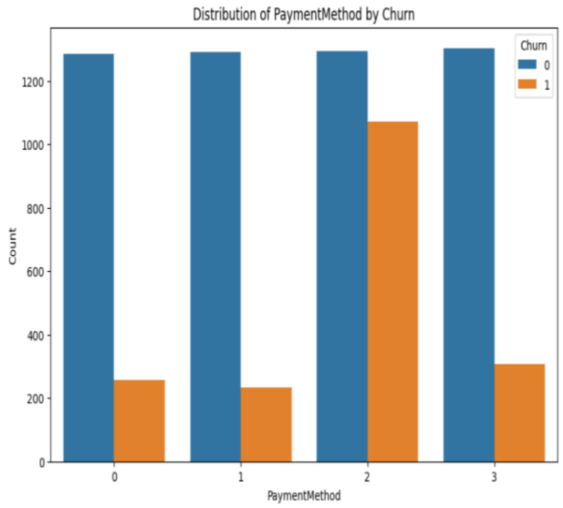
StreamingMovies



Contract



PaperlessBilling



PayementMethod

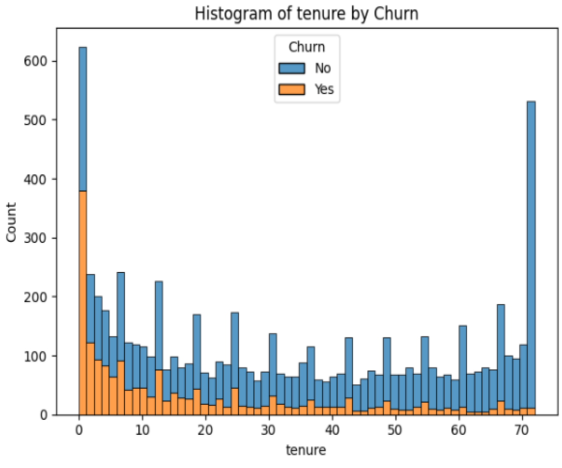
Figure. Feature of all categorical Features in the dataset.

I) Numeric Features:

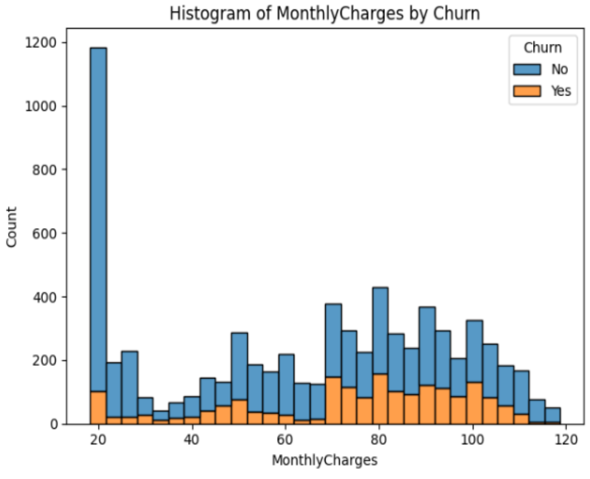
The distribution of numerical features within different churn categories (Yes vs. No) is crucial role to analyzing customer churn using a histogram graph

```
sns.histplot(data=df, x='tenure', hue='Churn', multiple='stack', bins=60)
plt.title(f'Histogram of tenure by Churn')
plt.xlabel("tenure")
plt.ylabel('Count')
plt.tight_layout()
plt.show()

sns.histplot(data=df, x='MonthlyCharges', hue='Churn', multiple='stack', bins=30)
plt.title(f'Histogram of MonthlyCharges by Churn')
plt.xlabel("MonthlyCharges")
plt.ylabel('Count')
plt.tight_layout()
plt.show()

sns.histplot(data=df, x='TotalCharges', hue='Churn', multiple='stack', bins=30)
plt.title(f'Histogram of TotalCharges by Churn')
plt.xlabel("TotalCharges")
plt.ylabel('Count')
plt.tight_layout()
plt.show()
```

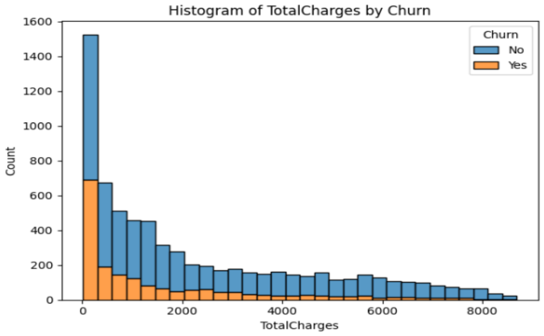
Output:



Tenure



MonthlyCharges



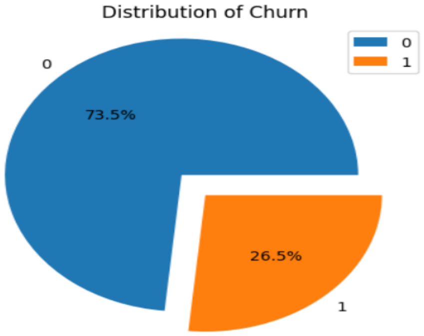
TotalCharges

Figure. Feature of all Numerical Features in the dataset.

II) Overall Distribution of Target:

```
temp = []
for i in list(dummy['Churn'].unique()):
    temp.append(dummy[dummy['Churn'] == i]['Churn'].count())
explode = [0.2,0]
plt.pie(temp,labels = list(dummy['Churn'].unique()), autopct='%1.1f%%',explode=explode)
plt.title('Distribution of Churn')
plt.legend()
plt.show()
```

Output:



6. Feature Selection:

Mutual Information for feature selection is a powerful technique to identify and retain the most informative features for classification tasks. It helps in reducing dimensionality and improving model performance

```
from sklearn.feature_selection import mutual_info_classif

x=dummy.drop(columns=['Churn'])
y=dummy['Churn']
mutual_info_scores = mutual_info_classif(x, y)

feature_scores = pd.DataFrame({'Feature': x.columns, 'Mutual Information': mutual_info_scores})
feature_scores = feature_scores.sort_values(by='Mutual Information', ascending=False)
```

```
print(feature_scores)
```

Output:

```
             Feature  Mutual Information
13          Contract            0.097404
3             tenure            0.070749
10       TechSupport            0.067147
7     OnlineSecurity            0.064741
8       OnlineBackup            0.050816
6    InternetService            0.049052
9   DeviceProtection            0.046724
16    MonthlyCharges            0.045213
12   StreamingMovies            0.043520
17      TotalCharges            0.041860
15     PaymentMethod            0.040559
11       StreamingTV            0.037909
14  PaperlessBilling            0.020427
1            Partner            0.017030
2         Dependents            0.009944
4       PhoneService            0.008208
5      MultipleLines            0.007733
0             gender            0.000000
```

I) Top 15 features to get:

```
sorted_feature_names = feature_scores.head(15)['Feature'].tolist()
```

```
dummy2=dummy[sorted_feature_names]
dummy2
```

Output:

	Contract	tenure	TechSupport	OnlineSecurity	InternetService	OnlineBackup	PaymentMethod	MonthlyCharges	DeviceProtection	TotalCharges	StreamingMovies	StreamingTV	Partner	Dependents	PaperlessBilling
0	0	1	0	0	0	2	2	29.85	0	29.85	0	0	1	0	1
1	1	34	0	2	0	0	3	56.95	2	1889.50	0	0	0	0	0
2	0	2	0	2	0	2	3	53.85	0	108.15	0	0	0	0	1
3	1	45	2	2	0	0	0	42.30	2	1840.75	0	0	0	0	0
4	0	2	0	0	1	0	2	70.70	0	151.65	0	0	0	0	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
7038	1	24	2	2	0	0	3	84.80	2	1990.50	2	2	1	1	1
7039	1	72	0	0	1	2	1	103.20	2	7362.90	2	2	1	1	1
7040	0	11	0	2	0	0	2	29.60	0	346.45	0	0	1	1	1
7041	0	4	0	0	1	0	3	74.40	0	306.60	0	0	1	0	1
7042	2	66	2	2	1	0	0	105.65	2	6844.50	2	2	0	0	1

7043 rows × 15 columns

7. Splitting Data into Training and Testing Sets:

```
X_train_le, X_test_le, y_train_le, y_test_le = train_test_split(dummy2, dummy['Churn'], test_size=0.3)
```

Methodology:

The goal of this study is to predict customer churn in a telecommunications company using three machine learning algorithms: Logistic Regression, Random Forest Classifier, and XGBoost Classifier. The methodology involves data preprocessing, feature selection, model training, evaluation, and comparison.

Model Training

a. Logistic Regression:

- Train a logistic regression model on the selected features. Logistic regression is a linear model suitable for binary classification tasks and provides interpretability of feature coefficients.

b. Random Forest Classifier:

- Train a random forest classifier, an ensemble learning method that combines multiple decision trees to improve prediction accuracy and control overfitting.

c. XGBoost Classifier:

- Train an XGBoost classifier, a powerful gradient boosting algorithm known for its efficiency and performance in various machine learning competitions.

Model Evaluation

a. Performance Metrics:

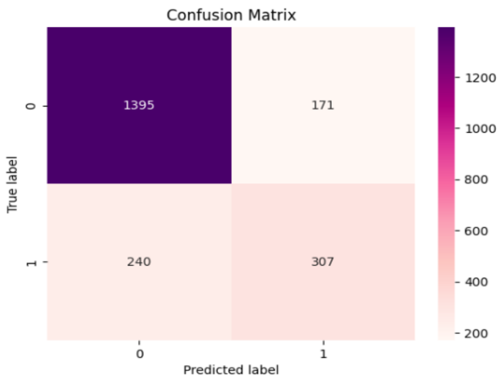
- Evaluate the performance of each model using the following metrics:
  - **Accuracy**: The ratio of correctly predicted instances to the total instances.
  - **Precision**: The ratio of true positive predictions to the total predicted positives.
  - **Recall**: The ratio of true positive predictions to the total actual positives.
  - **F1 Score**: The harmonic mean of precision and recall.

1) Logistic regression:

```
logis_Reg = LogisticRegression(max_iter=1500)
logis_Reg.fit(X_train_le,y_train_le)
y_pred_le = logis_Reg.predict(X_test_le)
print("Logistic Regression Accuracy:", accuracy_score(y_test_le, y_pred_le))
```

Output:

```
Logistic Regression Accuracy: 0.8054898248935163
```



	precision	recall	f1-score	support
0	0.85	0.89	0.87	1566
1	0.64	0.56	0.60	547
accuracy			0.81	2113
macro avg	0.75	0.73	0.74	2113
weighted avg	0.80	0.81	0.80	2113

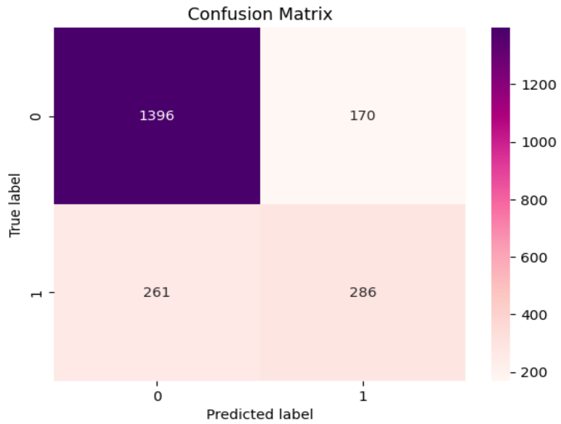
Confusion Matrix **Classification Report**

2) Random Forest Classifier

```
Rando_Forest = RandomForestClassifier(n_estimators=80,random_state=42)
Rando_Forest.fit(X_train_le,y_train_le)
y_pred_le = Rando_Forest.predict(X_test_le)
print("Random Forest Accuracy:", accuracy_score(y_test_le, y_pred_le))
```

Output:

```
Random Forest Accuracy: 0.7960246095598675
```



Confusion Matrix

	precision	recall	f1-score	support
0	0.84	0.89	0.87	1566
1	0.63	0.52	0.57	547
accuracy			0.80	2113
macro avg	0.73	0.71	0.72	2113
weighted avg	0.79	0.80	0.79	2113

Classification Report

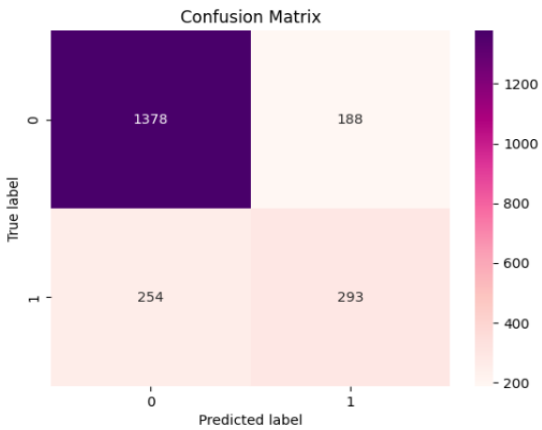
3) XgBoost Classifier:

```
import xgboost as xgb
```

```
XGB_c = xgb.XGBClassifier(use_label_encoder=False, eval_metric='logloss')
XGB_c.fit(X_train_le,y_train_le)
y_pred_le = XGB_c.predict(X_test_le)
print("XGBoost Accuracy:", accuracy_score(y_test_le, y_pred_le))
```

Output:

XGBoost Accuracy: 0.7908187411263606



Confusion Matrix

	precision	recall	f1-score	support
0	0.84	0.88	0.86	1566
1	0.61	0.54	0.57	547
accuracy			0.79	2113
macro avg	0.73	0.71	0.72	2113
weighted avg	0.78	0.79	0.79	2113

Classification Report

Result:

In this project, I have passed the telco customer churn dataset to the three different machine learning models and analyzed them with various metrics. The accuracy of these three different models was Logistic Regression 81%, Random Forest Classifier 79.6%, and XG Boost Classifier 79.08%. Among these models, the Logistic Regression model has produced higher accuracy with the80.5%. And also determined it with the confusion matrix and classification report. In future work, the dataset should be increased because the distribution of the target with the “NO” class is higher with a percentage of 75%. This may introduce biases in the data. So, the distribution of the dataset is equalized nearly to make the model more accurate.