Customer Churn Prediction

Customer Churn Prediction

[4]:
```
#Importing Liberies
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, GridSearchCV
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix, classification_report
```

[8]:
```
data=pd.read_csv('/Users/yashmantri/Downloads/Customer_data - customer_data.csv')
```

[10]:
```
data.head()
```

[10]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceProtection	TechSupp
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	

[12]:
```
data.describe()
```

[12]:

	SeniorCitizen	tenure	MonthlyCharges	TotalCharges
count	7043.000000	7043.000000	7043.000000	7032.000000
mean	0.162147	32.371149	64.761692	2283.300441
std	0.368612	24.559481	30.090047	2266.771362
min	0.000000	0.000000	18.250000	18.800000
25%	0.000000	9.000000	35.500000	401.450000
50%	0.000000	29.000000	70.350000	1397.475000
75%	0.000000	55.000000	89.850000	3794.737500
max	1.000000	72.000000	118.750000	8684.800000

[14]:
```
#fining the number of null values from each column
data.isnull().sum()
```

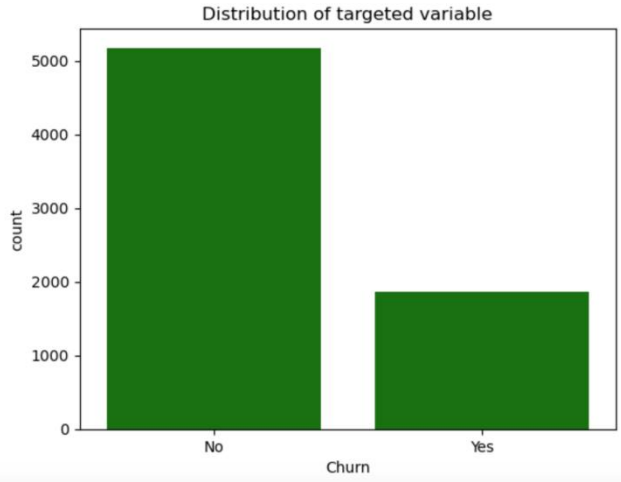
[14]:
```
customerID          0
gender              0
SeniorCitizen       0
Partner             0
Dependents          0
tenure              0
PhoneService        0
MultipleLines       0
InternetService     0
OnlineSecurity      0
OnlineBackup        0
DeviceProtection    0
TechSupport         0
StreamingTV         0
StreamingMovies     0
Contract            0
PaperlessBilling    0
PaymentMethod       0
MonthlyCharges      0
TotalCharges       11
Churn               0
dtype: int64
```

[18]:
```
#Replacinfg Null Values with 0
data['TotalCharges']=data['TotalCharges'].fillna(0)
```

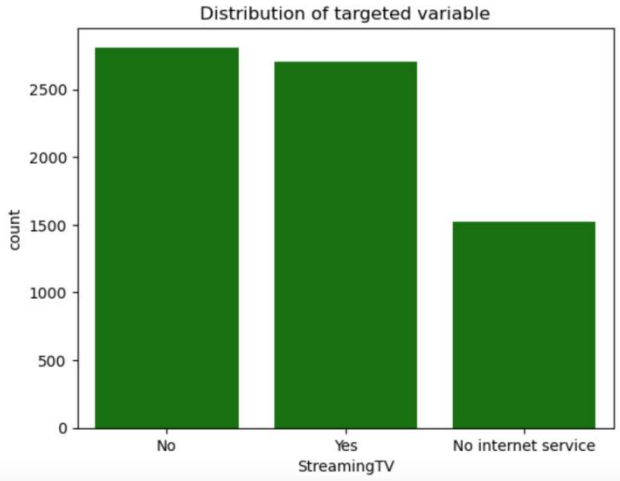
[21]:
```
#fininf the number of null values from each column
data.isnull().sum()
```

[21]:
```
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
```

[27]:
```
#Visulizing the distribution of targeted variable
sns.countplot(x='Churn',data=data, color='green')
plt.title('Distribution of targeted variable')
plt.show()
```



[39]:
```
#Visulizing the distribution of targeted variable
sns.countplot(x='StreamingTV',data=data, color='green')
plt.title('Distribution of targeted variable')
plt.show()
```

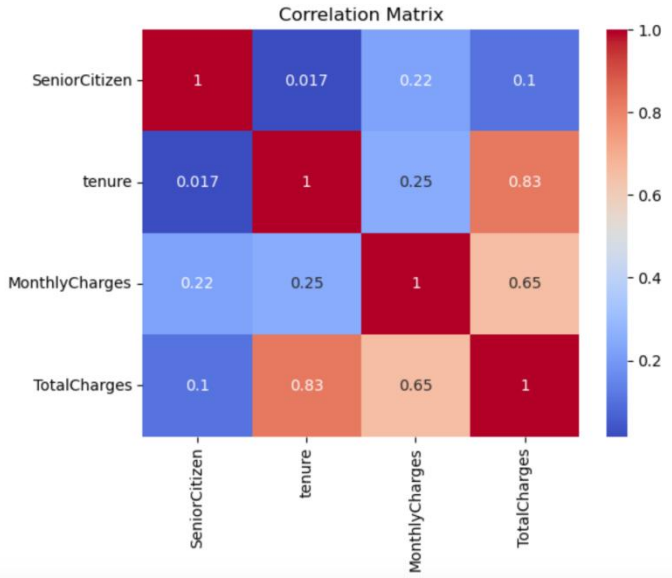


[37]:
```
#Correlation Matrix

num_list=list()
for columns in data.columns:
    if data[columns].dtype != object:
        num_list.append(columns)

correlation_matrix = data [num_list].corr()

#visulising the Matrix
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Matrix')
plt.show()
```



[45]:
```
bookmark=data.copy()
```

[53]:
```
# Encode categorical variables
label_encoders = {}
for column in ['gender','Partner', 'Dependents','PhoneService',
 'MultipleLines','InternetService', 'OnlineSecurity',
 'OnlineBackup','DeviceProtection', 'TechSupport','StreamingTV','StreamingMovies','Contract',
 'PaperlessBilling','PaymentMethod','Churn']:
    le =LabelEncoder()
    data[column] = le.fit_transform(data[column])
    label_encoders[column] = le
```

[55]:
```
data
```

[55]:

eSecurity	...	DeviceProtection	TechSupport	StreamingTV	StreamingMovies	Contract	PaperlessBilling	PaymentMethod	MonthlyCharges	TotalCharges	Churn
0	...	0	0	0	0	0	1	2	29.85	29.85	0
2	...	2	0	0	0	1	0	3	56.95	1889.50	0
2	...	0	0	0	0	0	1	3	53.85	108.15	1
2	...	2	2	0	0	1	0	0	42.30	1840.75	0
0	...	0	0	0	0	0	1	2	70.70	151.65	1
...	...	...	...	...	...	...	...	...	...	...	...
2	...	2	2	2	2	1	1	3	84.80	1990.50	0
0	...	2	0	2	2	1	1	1	103.20	7362.90	0
2	...	0	0	0	0	0	1	2	29.60	346.45	0
0	...	0	0	0	0	0	1	3	74.40	306.60	1
2	...	2	2	2	2	2	1	0	105.65	6844.50	0

[59]:
```
# Features and target
X = data.drop(['customerID', 'Churn'], axis=1)
y = data['Churn']
```

[63]:
```
# 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=0)
```

[67]:
```
# Feature scaling
scaler= StandardScaler()
X_train=scaler.fit_transform(X_train)
X_test=scaler.transform(X_test)
```

[69]:
```
# Logistic Regression

logreg= LogisticRegression()
logreg.fit(X_train, y_train)

logreg_pred= logreg.predict(X_test)
```

[71]:
```
logreg_accuracy= accuracy_score (y_test, logreg_pred)
logreg_precision= precision_score (y_test, logreg_pred)
logreg_recall= recall_score (y_test, logreg_pred)
logreg_f1score=f1_score(y_test,logreg_pred)
```

[75]:
```
# Display accuracy and performance metrics
print(f'Logistic Regression Accuracy: {(logreg_accuracy) *100:.2f}%')
print(f'Logistic Regression Precision: {(logreg_precision) *100:.2f}%')
print(f'Logistic Regression Recall: {(logreg_recall) *100:.2f}%')
print(f'Logistic Regression F1score: {(logreg_f1score) *100:.2f}%')
```

```
Logistic Regression Accuracy: 79.79%
Logistic Regression Precision: 63.35%
Logistic Regression Recall: 54.07%
Logistic Regression F1score: 58.34%
```