CONNECTTEL CUSTOMER CHURN PREDICTION

PROJECT REPORT AND SUMMARY

PROJECT TITLE: ConnectTel Customer Churn Prediction using Supervised Machine Learning

AUTHOR: Adewale Odetara

DATE: 14th November, 2023

Introduction:

In the dynamic landscape of telecommunications, customer churn poses a significant challenge for companies like ConnectTel. The ability to predict and understand customer churn is crucial for business sustainability and growth. In this project, I delve into the realm of churn prediction, leveraging machine learning techniques to develop models capable of identifying customers at risk of leaving the service.

Project Background

ConnectTel is facing a client retention difficulty that threatens the company's long-term viability and growth. Customer churn prediction predicts potential customers to leave a company's service, requiring effective marketing strategies to increase their likelihood of staying.

Project Objective

The primary goal is to develop an accurate and reliable predictive model using machine learning to predict which customers are likely to churn and implement proactive measures.

In [1]:
```
# Import necessary Libraries

# For data analysis
import pandas as pd
import numpy as np
```

In [2]:
```
# For data visualization
import matplotlib.pyplot as plt
import seaborn as sns
```

In [3]:
```
# Data pre-processing
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import LabelEncoder
```

In [4]:
```
#Classifier Libraries
from sklearn.linear_model import SGDClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
```

In [5]:
```
# Ipip install xgboost
from xgboost import XGBClassifier
from sklearn.svm import LinearSVC, SVC
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
```

In [6]:
```
# Evaluation metrics
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_sco
from sklearn.metrics import confusion_matrix
```

In [7]:
```
import warnings
warnings.filterwarnings("ignore")
```

Load the data

In [8]:
```
# Load the dataset
df = pd.read_csv(r"C:\Users\ADMIN\Desktop\Resources\10Alytics Data Science\Capstone Project\Cust
```

In [9]:
```
df.head()
```

Out[9]:

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

5 rows × 21 columns

In [10]:
```
df.shape
```

Out[10]:
```
(7043, 21)
```

The data has 7043 rows and 21 columns

In [11]:
```
# Data verification - Data type, number of features and rows, missing data, e.t.c
df.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
```

There is 21 columns in the dataset and only columns are numeric data type while the remaining are categoricals data type.

In [12]:
```
# Statistical Analysis of the data
df.describe()
```

Out[12]:

	SeniorCitizen	tenure	MonthlyCharges
count	7043.000000	7043.000000	7043.000000
mean	0.162147	32.371149	64.761692
std	0.368612	24.559481	30.090047
min	0.000000	0.000000	18.250000
25%	0.000000	9.000000	35.500000
50%	0.000000	29.000000	70.350000
75%	0.000000	55.000000	89.850000
max	1.000000	72.000000	118.750000

To understand the distribution of variables and the relationship between them.

In [13]:
```
df.describe(exclude=["int64", "float64"]).T
```

Out[13]:

	count	unique	top	freq
customerID	7043	7043	7590-VHVEG	1
gender	7043	2	Male	3555
Partner	7043	2	No	3641
Dependents	7043	2	No	4933
PhoneService	7043	2	Yes	6361
MultipleLines	7043	3	No	3390
InternetService	7043	3	Fiber optic	3096
OnlineSecurity	7043	3	No	3498
OnlineBackup	7043	3	No	3088
DeviceProtection	7043	3	No	3095
TechSupport	7043	3	No	3473
StreamingTV	7043	3	No	2810
StreamingMovies	7043	3	No	2785
Contract	7043	3	Month-to-month	3875
PaperlessBilling	7043	2	Yes	4171
PaymentMethod	7043	4	Electronic check	2365
TotalCharges	7043	6531		11
Churn	7043	2	No	5174

In [14]:

```
# Check for duplicates
df.duplicated().sum()
```

Out[14]:

```
0
```

Data Visualization

Investigate the data to discover any patterns.

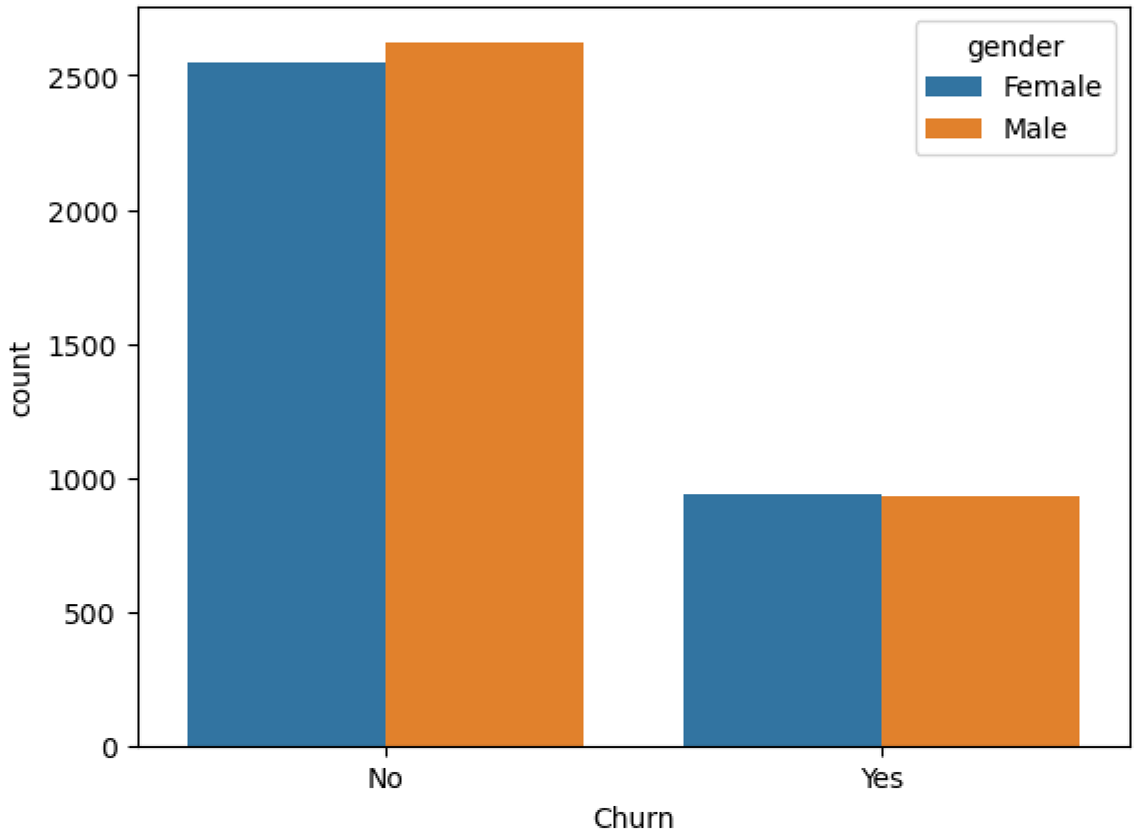
I used seaborn countplot to plot a graph against churn column for the categorical data

In [15]:

```
sns.countplot(x='Churn',data=df,hue='gender')
```

Out[15]:

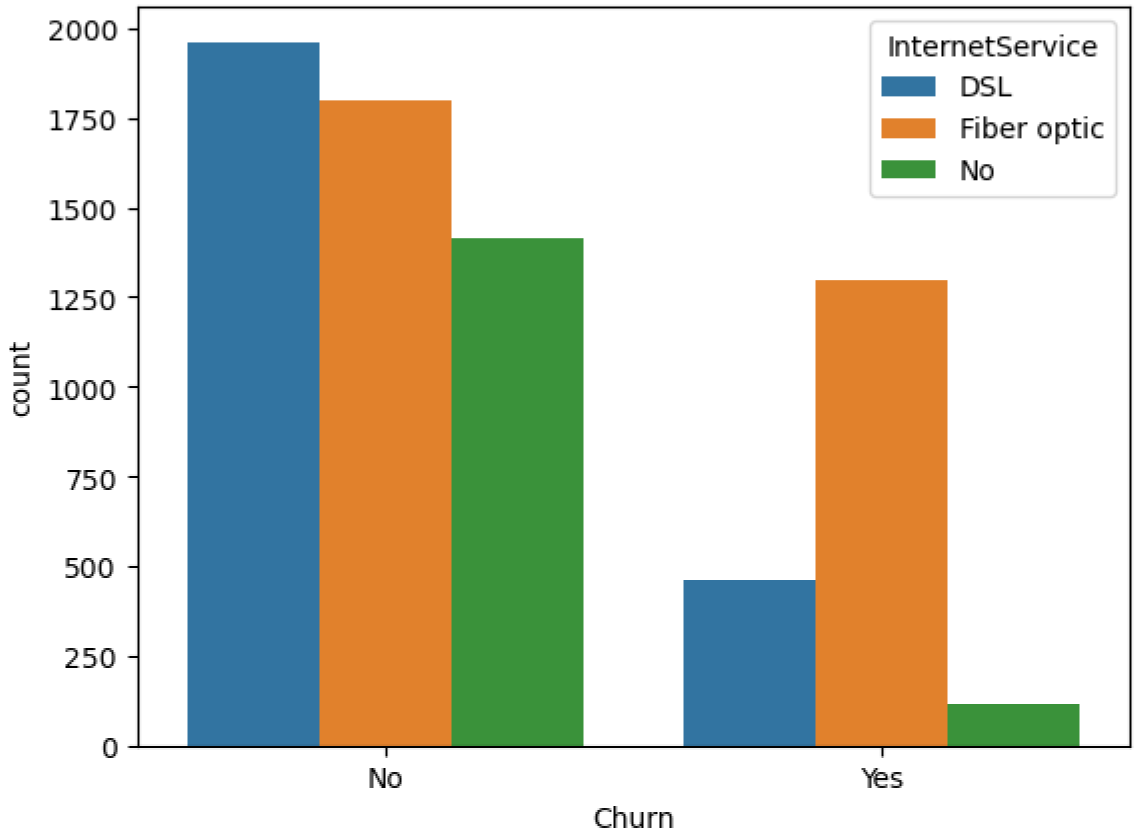
```
<Axes: xlabel='Churn', ylabel='count'>
```



The preceding plot shows that gender is not an important factor in customer churn in this data set because the numbers of both genders who have or have not churned are almost equal.

In [16]:
```
sns.countplot(x='Churn',data=df, hue='InternetService')
```

Out[16]:
```
<Axes: xlabel='Churn', ylabel='count'>
```



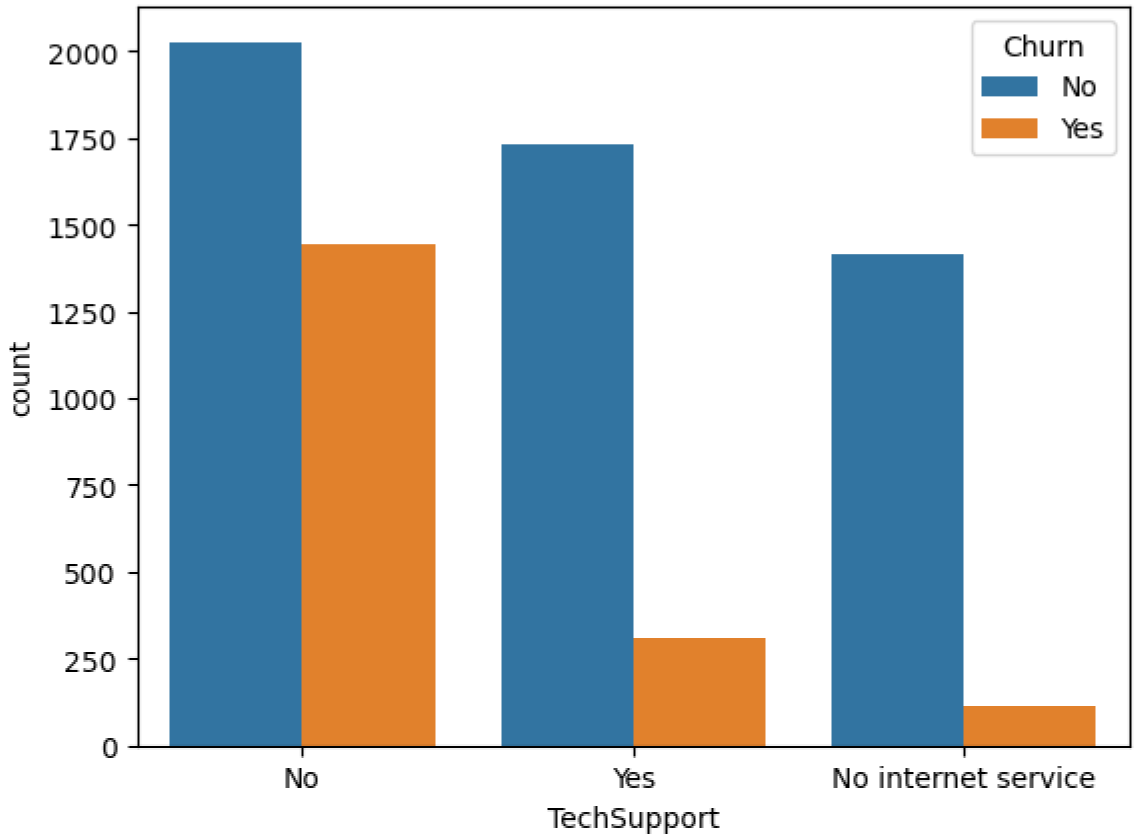
We can see that those who use fiber-optic services have a greater churn rate. This demonstrates that the company's Fiber-optic service has to be improved.

In [17]:

```
sns.countplot(x='TechSupport',data=df, hue='Churn')
```

Out[17]:

```
<Axes: xlabel='TechSupport', ylabel='count'>
```

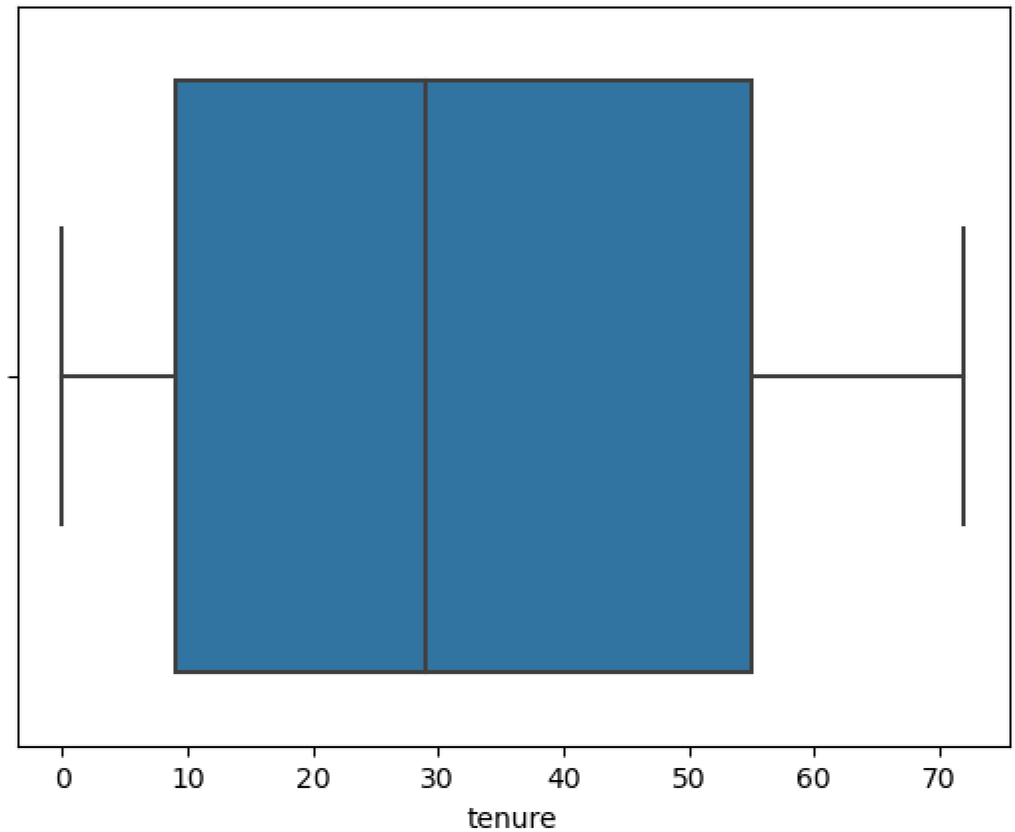


Customers that do not have tech assistance have a higher turnover rate, which is obvious. This also demonstrates that the company's technical help is of high quality.

Check for outliers

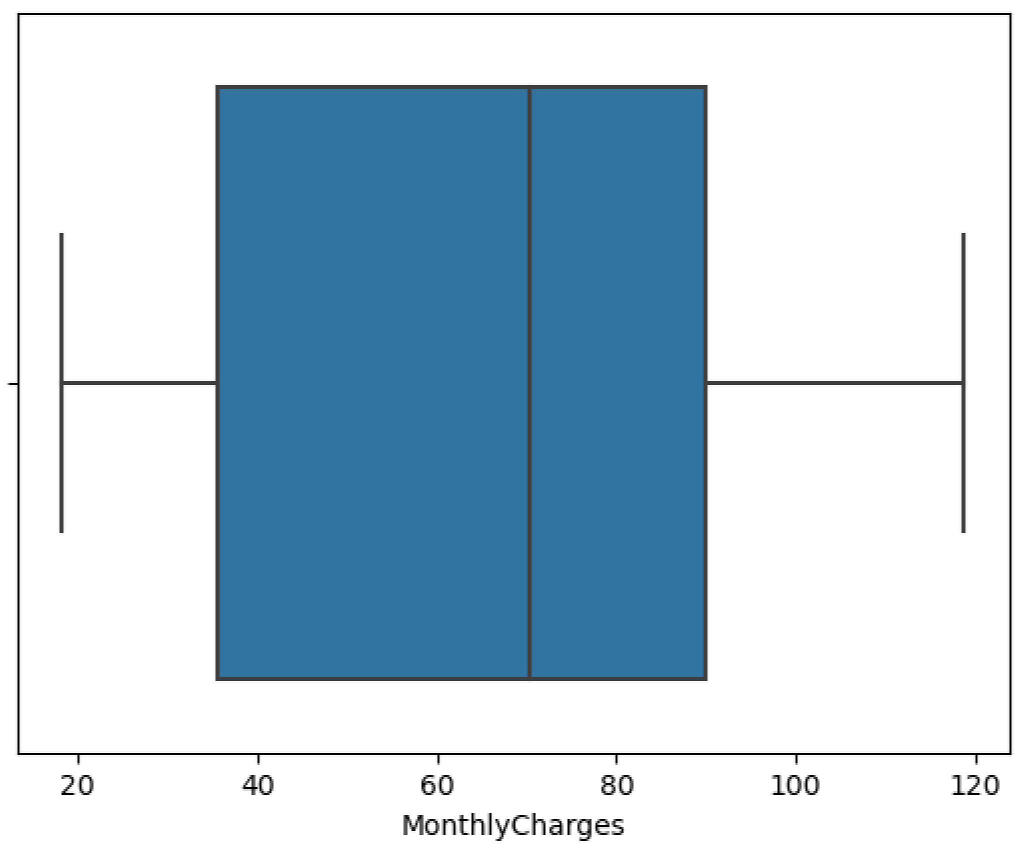
In [18]:
```
sns.boxplot(x=df["tenure"])
```

Out[18]:
```
<Axes: xlabel='tenure'>
```



In [19]:
```
sns.boxplot(x=df["MonthlyCharges"])
```

Out[19]:
```
<Axes: xlabel='MonthlyCharges'>
```



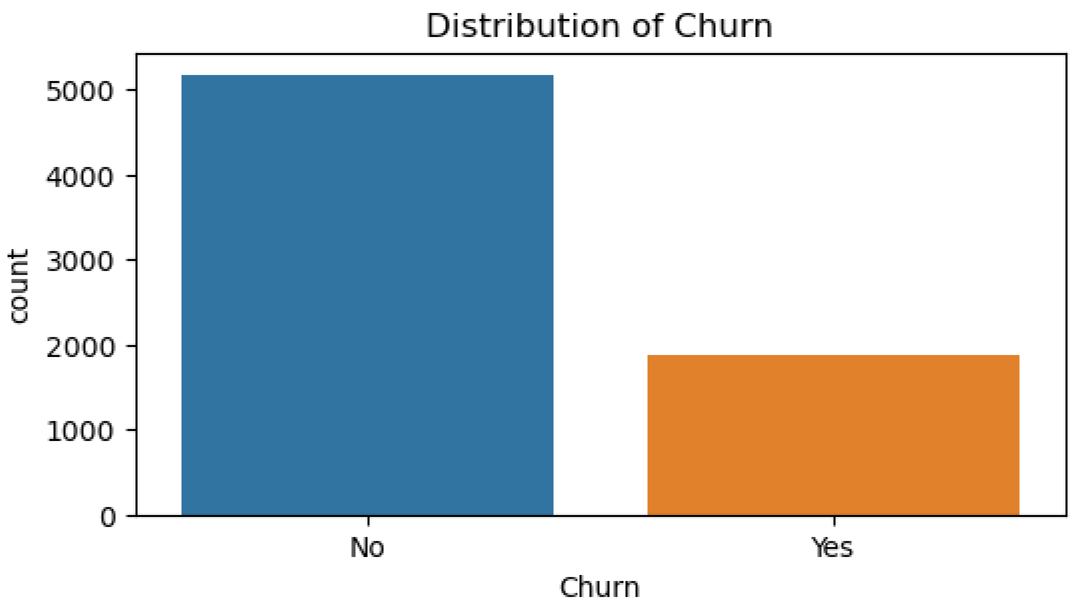
In [ ]:

Exploratory Data Analysis

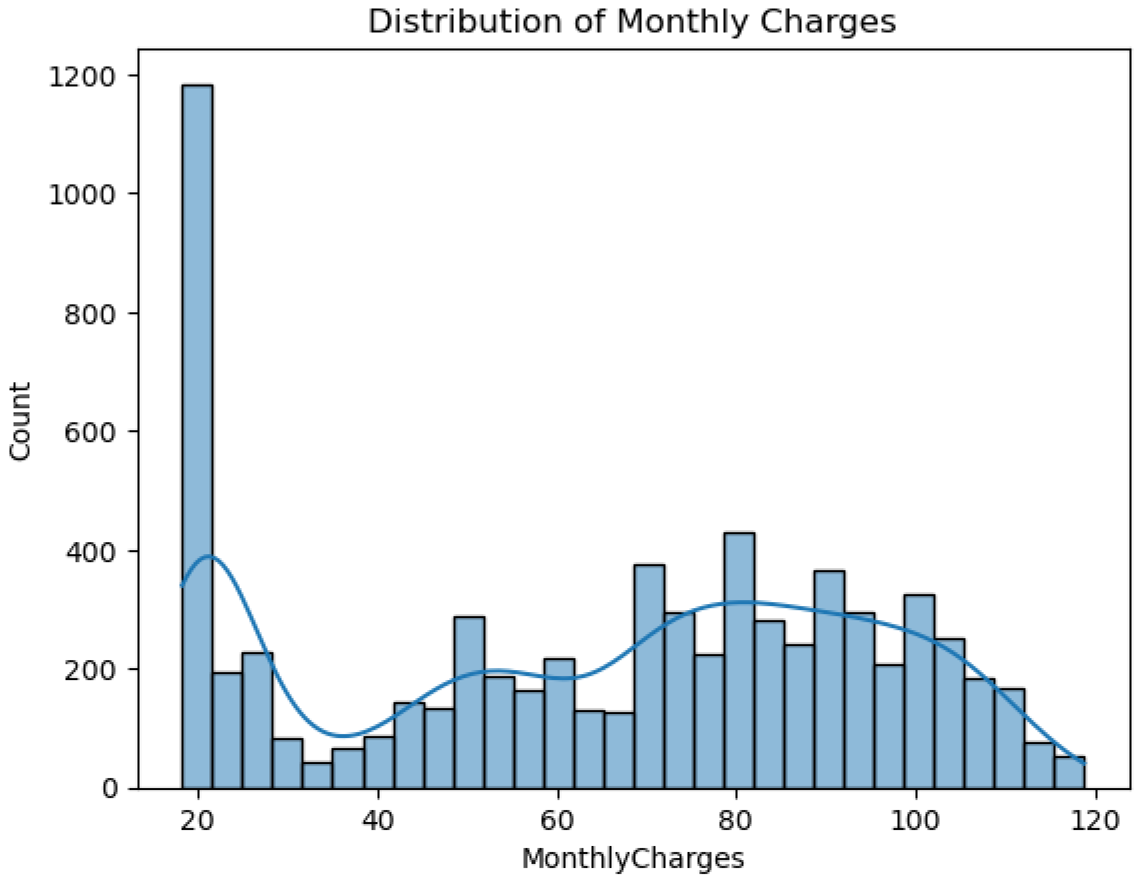
EDA is a comprehensive data analysis process that uses visual methods to uncover trends, patterns, and assumptions, while removing irregularities and unnecessary values from the data.

Univariate Analysis

In [64]:
```
plt.figure(figsize=(6,3))
sns.countplot(x='Churn', data=df)
plt.title('Distribution of Churn')
plt.show()
```

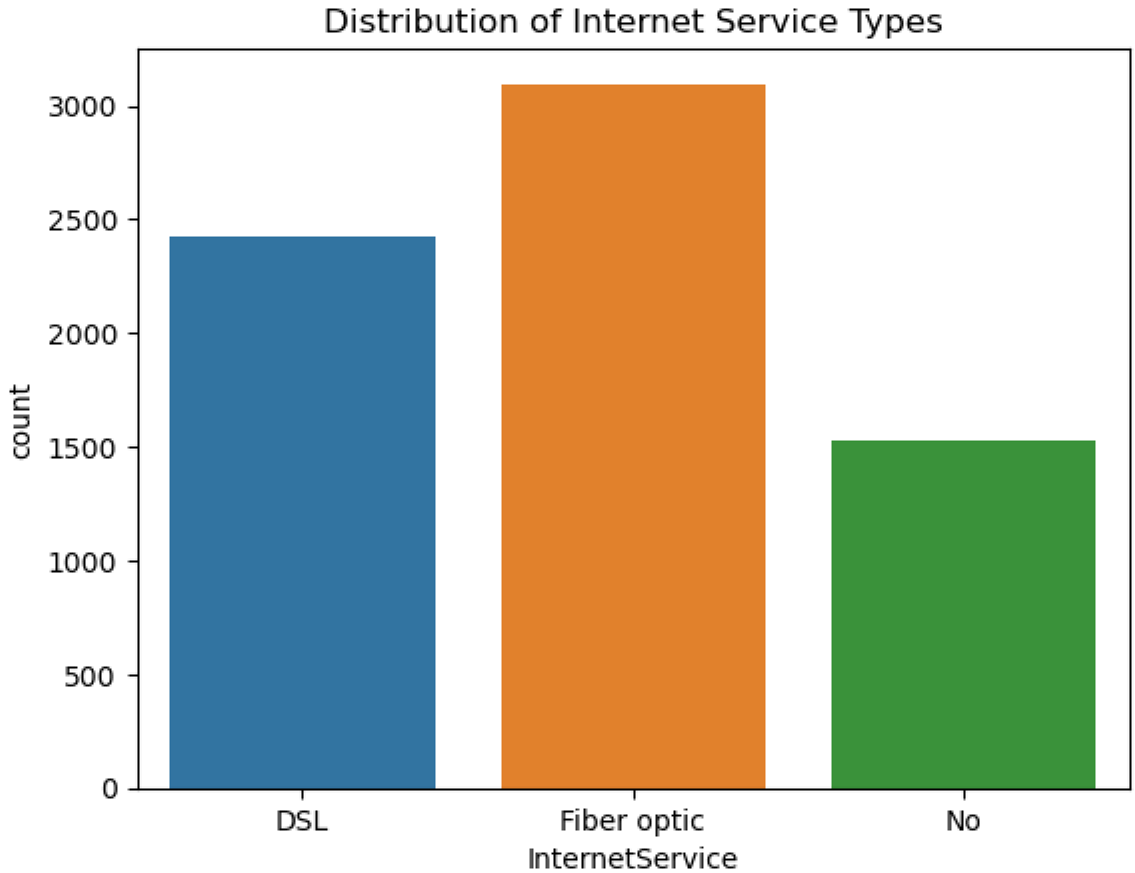


In [21]:
```
sns.histplot(x='MonthlyCharges', data=df, bins=30, kde=True)
plt.title('Distribution of Monthly Charges')
plt.show()
```



The company's success in retaining high-paying clients, even with monthly fees as high as $100, is evident from the lack of clear patterns observed.

In [22]:
```
sns.countplot(x='InternetService', data=df)
plt.title('Distribution of Internet Service Types')
plt.show()
```

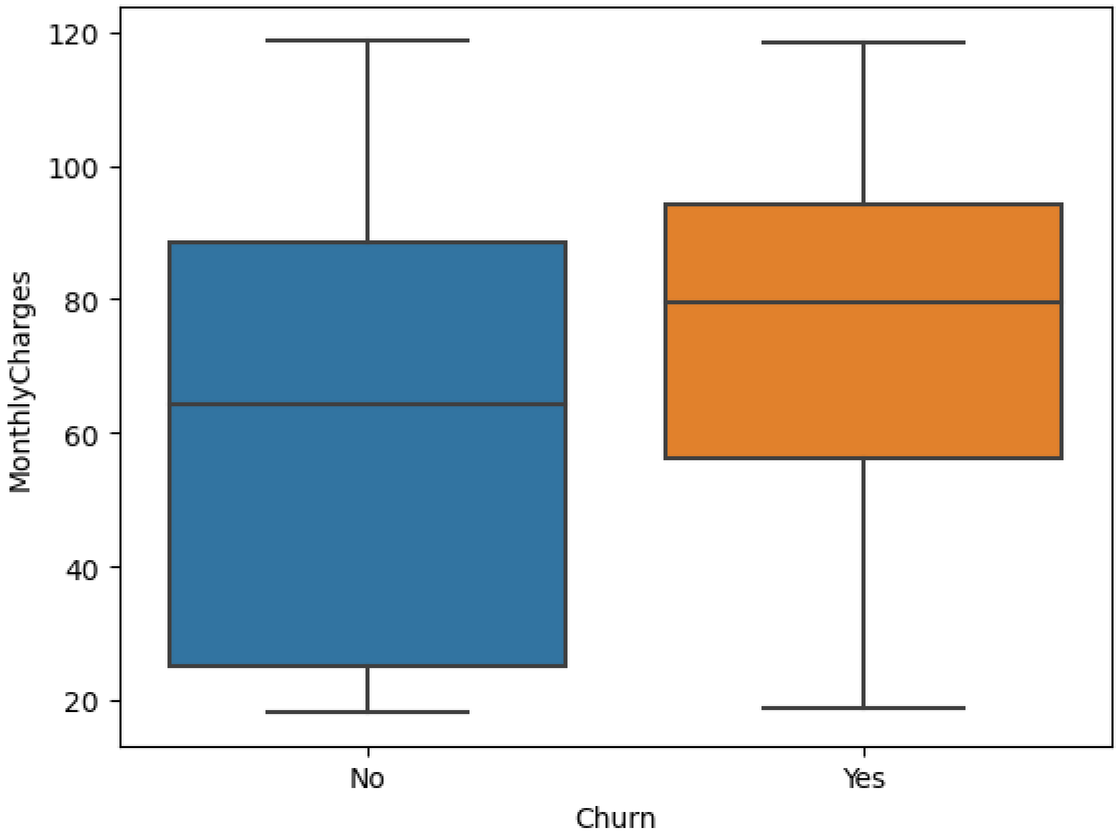


In [ ]:

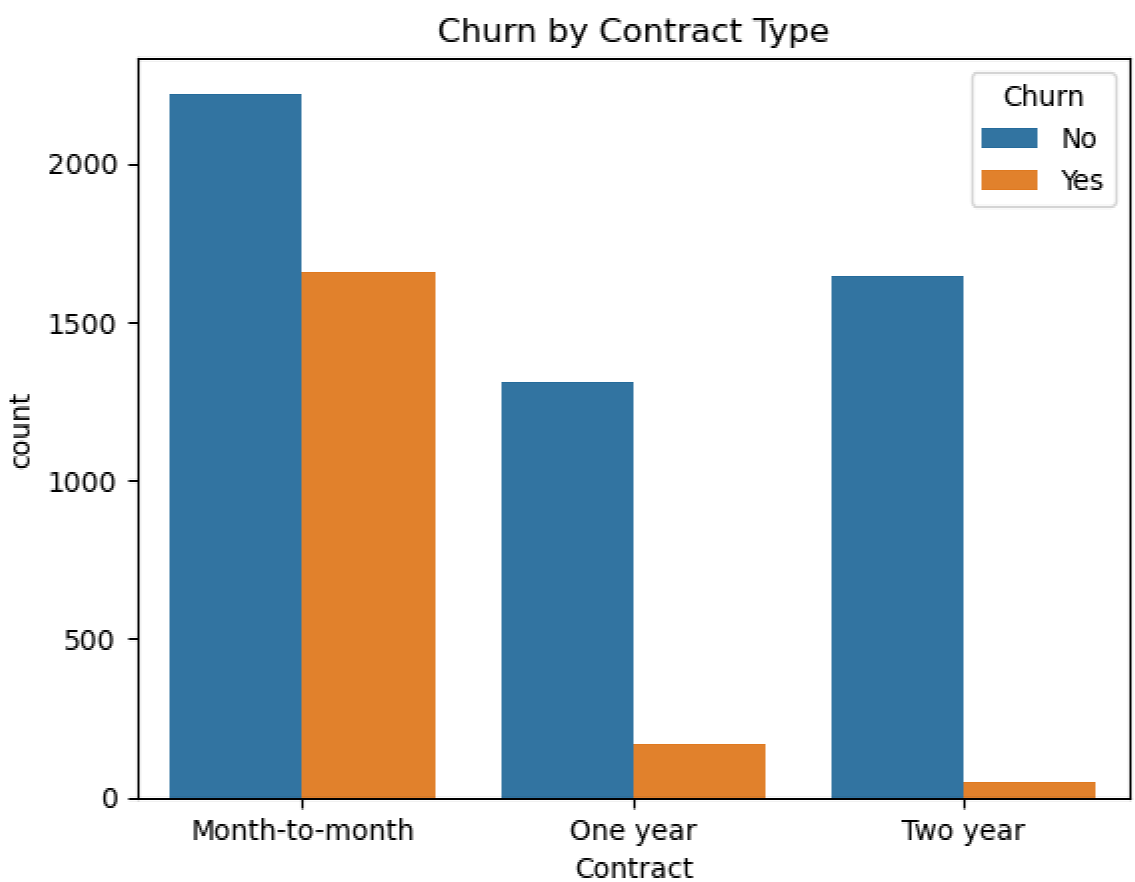
In [ ]:

Bivariate Analysis

In [23]:
```
# Bivariate analysis between MonthlyCharges and Churn
sns.boxplot(x='Churn', y='MonthlyCharges', data=df)
plt.show()
```



In [24]:
```
sns.countplot(x='Contract', data=df, hue='Churn')
plt.title('Churn by Contract Type')
plt.show()
```

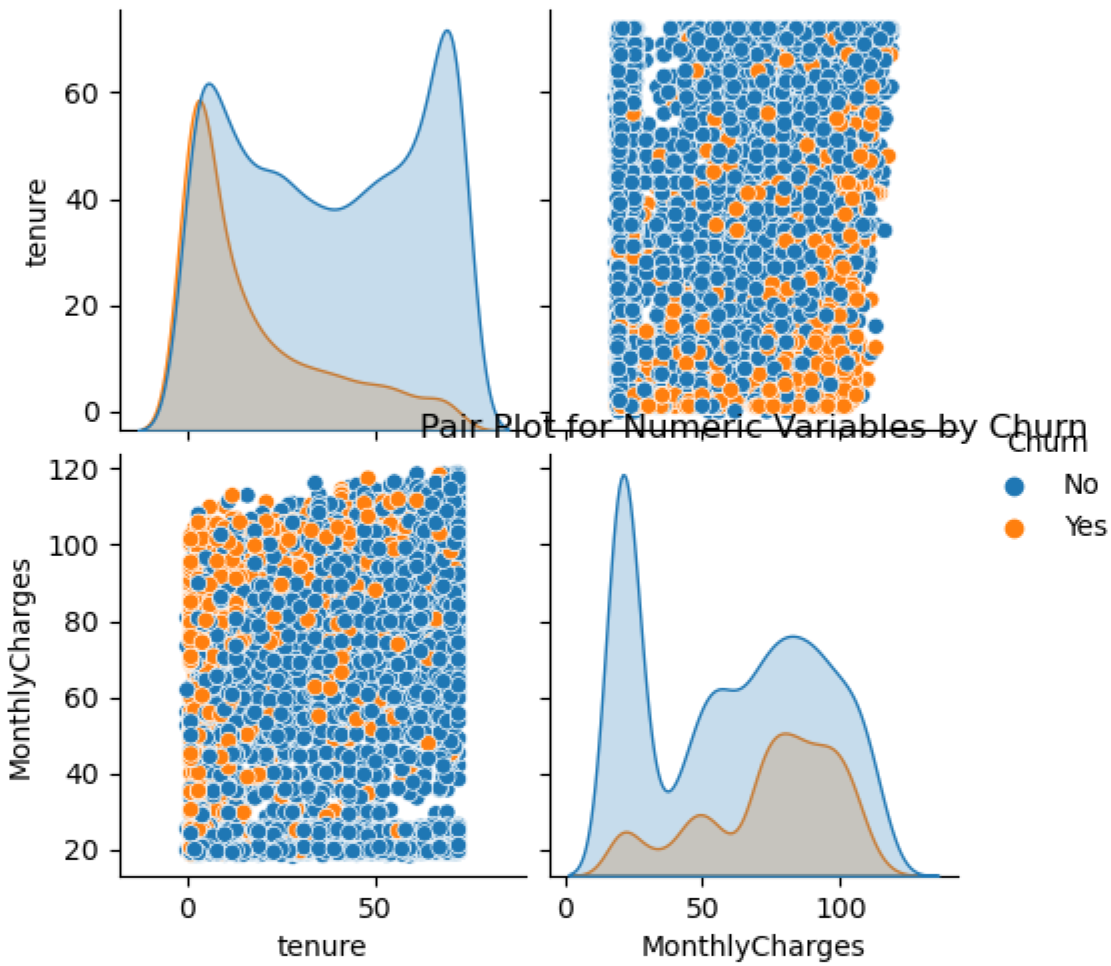


The churn rate is higher in the month-to-month, when new customers try out the service and decide whether to stay or terminate. This can be attributable primarily to the customer's uncertainty.

In [ ]:

Multivariate Analysis

In [25]:
```
sns.pairplot(df[['tenure', 'MonthlyCharges', 'TotalCharges', 'Churn']], hue='Churn')
plt.title('Pair Plot for Numeric Variables by Churn')
plt.show()
```



In [26]:
```
corr_matrix = df.corr()
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Matrix')
plt.show()
```

In [ ]:

Feature Engineering/Data Preprocessing

- Data Cleaning
- Encoding Categorical Variable
- Data Normalization

In [27]:
```
# Create a copy of the data
df1 = df.copy()
```

In [28]:
```
df1.shape
```

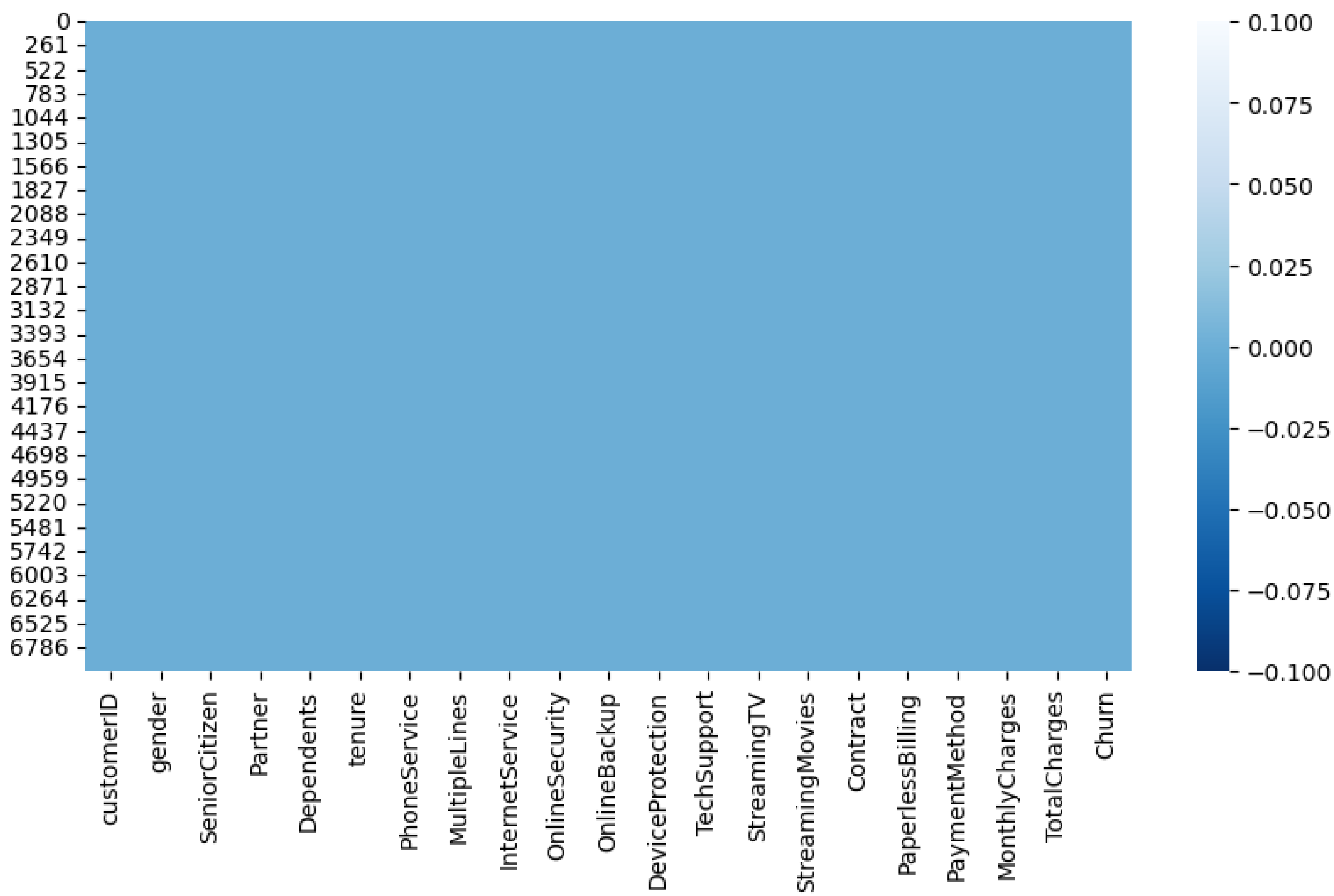
Out[28]:
```
(7043, 21)
```

In [29]:
```
# Check for missing value
df1.isnull().sum()
```

```
Out[29]: 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
```

```
In [30]: # Visualizing the missing data
plt.figure(figsize=(10,5))
sns.heatmap(df1.isnull(), cbar=True, cmap="Blues_r")
```

```
Out[30]: <Axes: >
```



Encoding Categorical Variables

In [31]:
```
# Encoding Categorical Variables

cat_feat = (df1.dtypes == "object")
cat_feat = list(cat_feat[cat_feat].index)

encoder = LabelEncoder()
for i in cat_feat:
    df1[i] = df1[[i]].apply(encoder.fit_transform)
```

In [32]:
```
df1.head()
```

Out[32]:

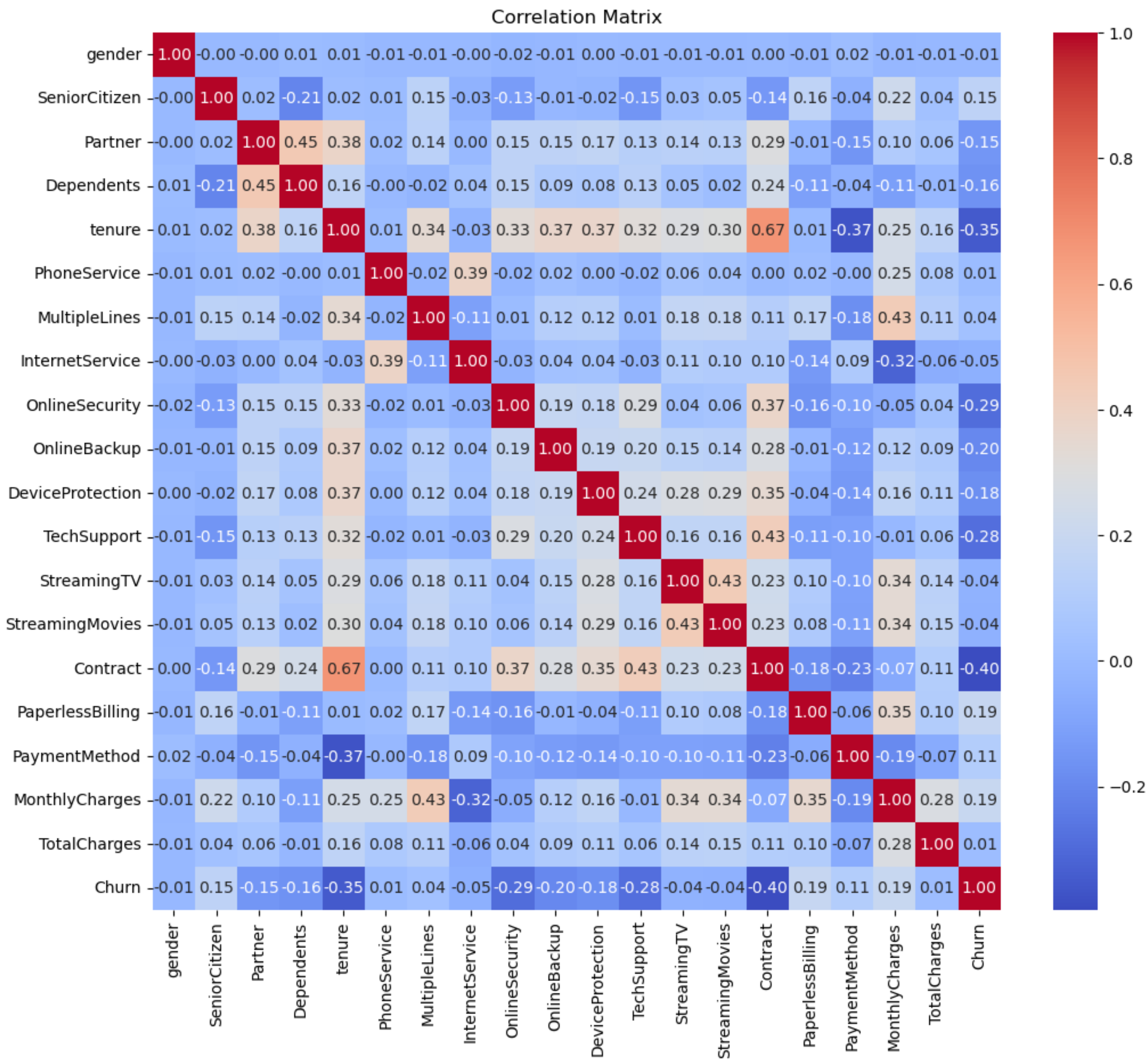
	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService
0	5375	0	0	1	0	1	0	1	0
1	3962	1	0	0	0	34	1	0	0
2	2564	1	0	0	0	2	1	0	0
3	5535	1	0	0	0	45	0	1	0
4	6511	0	0	0	0	2	1	0	1

5 rows × 21 columns

In [33]:
```
# Drop CustomerID column
df1.drop('customerID', axis=1, inplace=True)
```

In [34]:
```
# Explore Correlations

correlation_matrix = df1.corr()
plt.figure(figsize=(12, 10))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Matrix')
plt.show()
```



The correlation matrix shows how the variables are related to each other. a value close to 1 or -1 indicates a strong positive or negative correlation respectively. It can be seen that variable 'MonthlyCharges' and 'Paperlessbill' have a moderate positive correlation with outcome while 'tenue' and contract have a moderate negative relationship with the outcome, which indicating that they could be important factors in predicting customer churn.

In [35]:
```
y = df1.pop('Churn')
```

In [36]:
```
#df1.head()
```

Create new feature from the dataset

In [37]:
```
# Create a 'TotalTenureCharges' feature
df1['TotalTenureCharges'] = df1['tenure'] * df1['MonthlyCharges']
```

In [38]:
```python
df1.head()
```

Out[38]:

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecuri
0	0	0	1	0	1	0	1	0	
1	1	0	0	0	34	1	0	0	
2	1	0	0	0	2	1	0	0	
3	1	0	0	0	45	0	1	0	
4	0	0	0	0	2	1	0	1	

In [39]:
```python
# Normalize/Scaling Dataset

scaler = MinMaxScaler()
scaled_df = scaler.fit_transform(df1)
scaled_df = pd.DataFrame(scaled_df,columns=df1.columns)
```

In [40]:
```python
scaled_df
```

Out[40]:

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineS
0	0.0	0.0	1.0	0.0	0.013889	0.0	0.5	0.0	
1	1.0	0.0	0.0	0.0	0.472222	1.0	0.0	0.0	
2	1.0	0.0	0.0	0.0	0.027778	1.0	0.0	0.0	
3	1.0	0.0	0.0	0.0	0.625000	0.0	0.5	0.0	
4	0.0	0.0	0.0	0.0	0.027778	1.0	0.0	0.5	
...	...	...	...	...	...	...	...	...	
7038	1.0	0.0	1.0	1.0	0.333333	1.0	1.0	0.0	
7039	0.0	0.0	1.0	1.0	1.000000	1.0	1.0	0.5	
7040	0.0	0.0	1.0	1.0	0.152778	0.0	0.5	0.0	
7041	1.0	1.0	1.0	0.0	0.055556	1.0	1.0	0.5	
7042	1.0	0.0	0.0	0.0	0.916667	1.0	0.0	0.5	

7043 rows × 20 columns

In [ ]:

Build Machine Learning Model

In [41]:
```python
# Split the dataset into training and testing sets - x = questions while y = answers

X_train, X_test, y_train, y_test = train_test_split(scaled_df, y, test_size=0.2, random_state=42
```

In [42]:
```
# Model Building
# Logistic Regression

logreg = LogisticRegression()

logreg.fit(X_train, y_train)

ly_pred = logreg.predict(X_test)

print("Logistic Regression")
print("Accuracy:", accuracy_score(y_test, ly_pred))
print("Precision:", precision_score(y_test, ly_pred))
print("Recall:", recall_score(y_test, ly_pred))
print("F1-score:", f1_score(y_test, ly_pred))
print("AUC-ROC:", roc_auc_score(y_test, ly_pred))
```

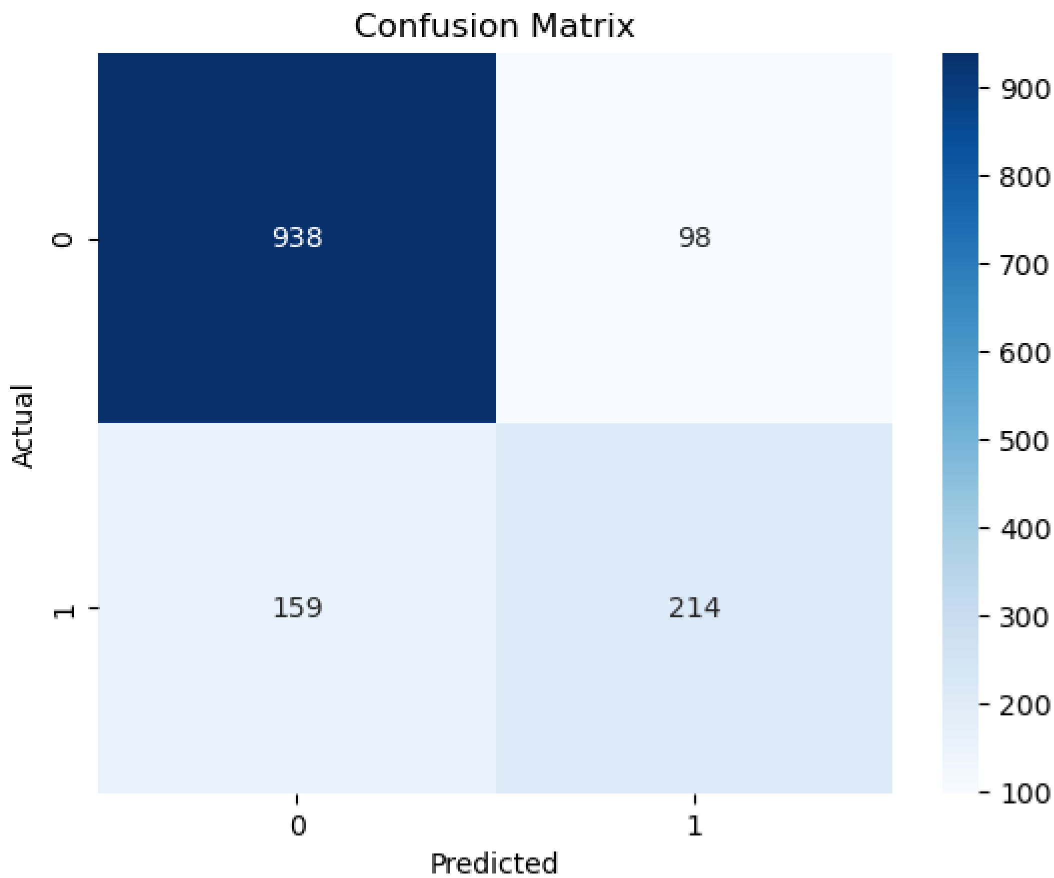
```
Logistic Regression
Accuracy: 0.8176011355571328
Precision: 0.6858974358974359
Recall: 0.5737265415549598
F1-score: 0.6248175182481752
AUC-ROC: 0.7395659734801826
```

In [43]:
```
# Create a confusion matrix

lcm = confusion_matrix(y_test, ly_pred)

# Visualize the confusion matrix

sns.heatmap(lcm, annot=True, cmap="Blues", fmt="g")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.show()
```



In [44]:
```
from sklearn.metrics import classification_report
```

In [45]:
```
# Print the classification report - Logistic Regression

print(classification_report(y_test, ly_pred))
```

```
              precision    recall  f1-score   support

           0       0.86      0.91      0.88      1036
           1       0.69      0.57      0.62       373

    accuracy                           0.82      1409
   macro avg       0.77      0.74      0.75      1409
weighted avg       0.81      0.82      0.81      1409
```

In [46]:
```
# Model Building

# Random Forest Classifier

rfc = RandomForestClassifier()
rfc.fit(X_train, y_train)
rfy_pred = rfc.predict(X_test)
print("Random Forest")
print("Accuracy:", accuracy_score(y_test, rfy_pred))
print("Precision:", precision_score(y_test, rfy_pred))
print("Recall:", recall_score(y_test, rfy_pred))
print("F1-score:", f1_score(y_test, rfy_pred))
print("AUC-ROC:", roc_auc_score(y_test, rfy_pred))
```

```
Random Forest
Accuracy: 0.7906316536550745
Precision: 0.6413043478260869
Recall: 0.4745308310991957
F1-score: 0.5454545454545453
AUC-ROC: 0.6894854927696752
```

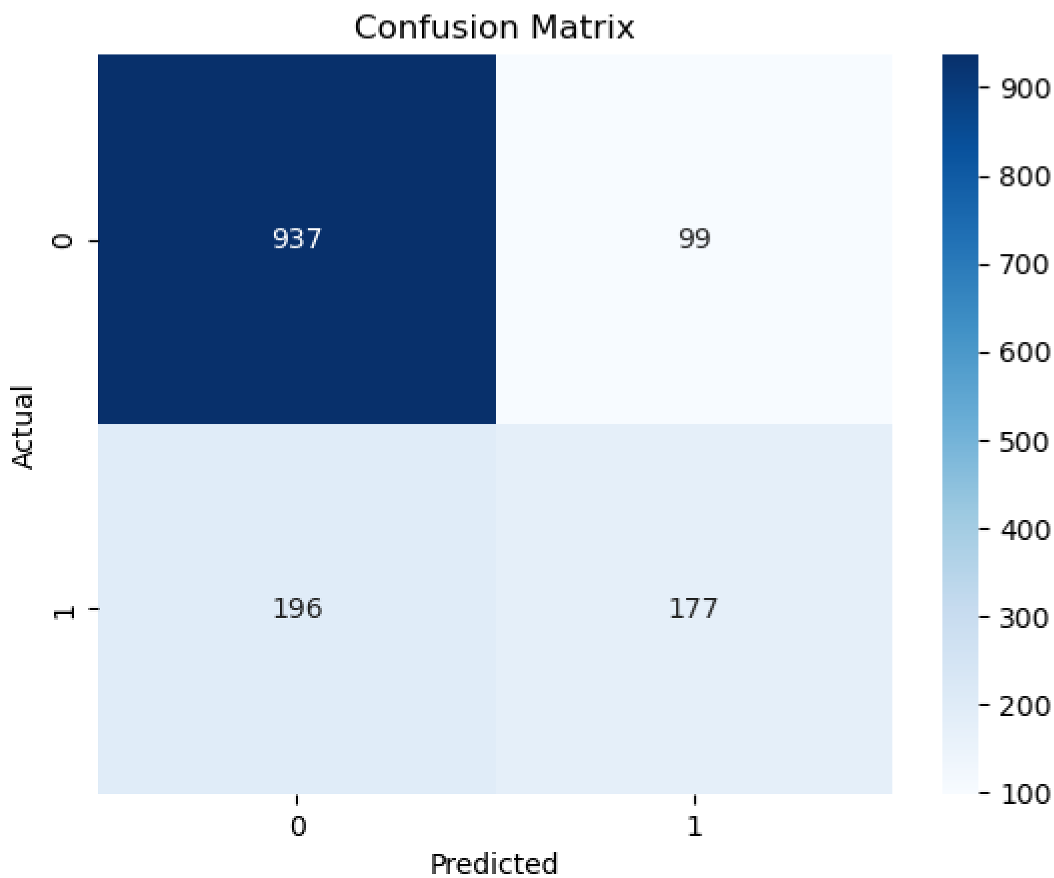
In [47]:

```
# Create a confusion matrix

rcm = confusion_matrix(y_test, rfy_pred)

# Visualize the confusion matrix

sns.heatmap(rcm, annot=True, cmap="Blues", fmt="g")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.show()
```



In [48]:

```
# Print the classification report - Random Forest
print(classification_report(y_test, rfy_pred))
```

```
              precision    recall  f1-score   support

           0       0.83      0.90      0.86      1036
           1       0.64      0.47      0.55       373

    accuracy                           0.79      1409
   macro avg       0.73      0.69      0.70      1409
weighted avg       0.78      0.79      0.78      1409
```

In [49]:

```
# Model Building
```

```python
# Decision Tree Classifier

dtc = DecisionTreeClassifier()
dtc.fit(X_train, y_train)
dty_pred = dtc.predict(X_test)
print("Decision Tree")
print("Accuracy:", accuracy_score(y_test, dty_pred))
print("Precision:", precision_score(y_test, dty_pred))
print("Recall:", recall_score(y_test, dty_pred))
print("F1-score:", f1_score(y_test, dty_pred))
print("AUC-ROC:", roc_auc_score(y_test, dty_pred))
```

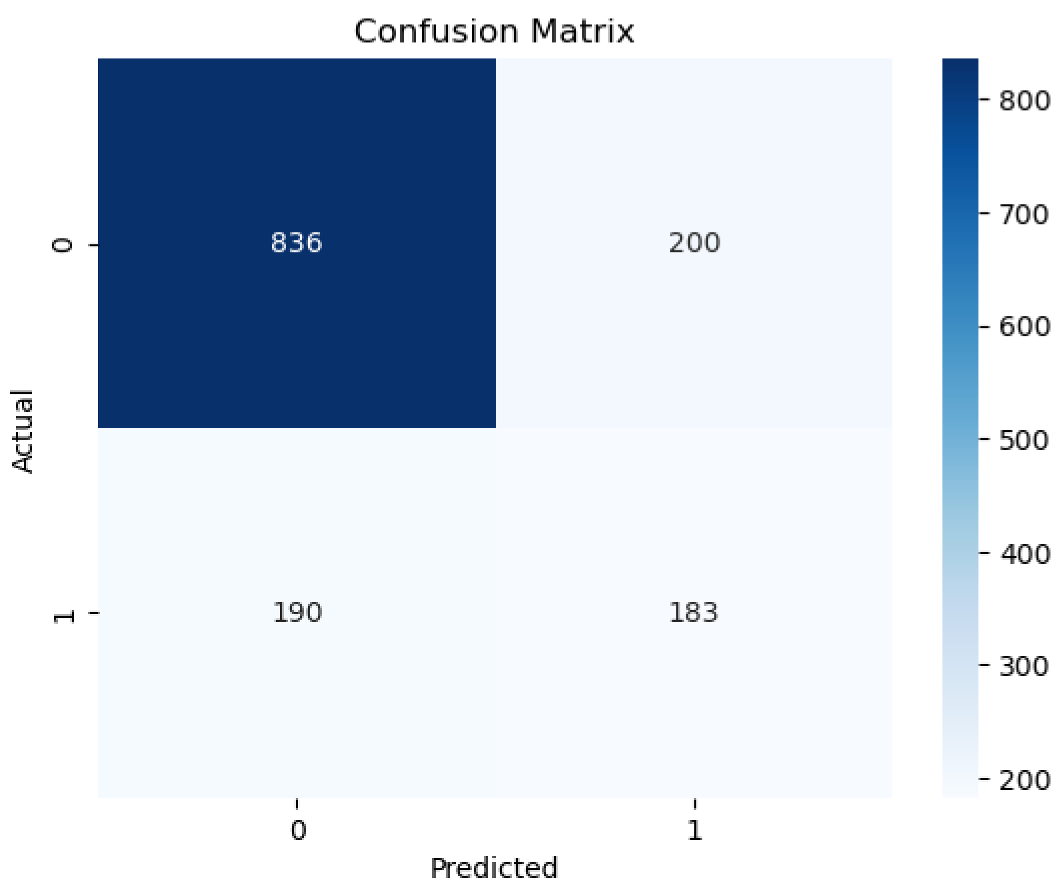
```
Decision Tree
Accuracy: 0.723207948899929
Precision: 0.47780678851174935
Recall: 0.4906166219839142
F1-score: 0.4841269841269842
AUC-ROC: 0.6487832144668606
```

In [50]:
```python
# Create a confusion matrix

dcm = confusion_matrix(y_test, dty_pred)

# Visualize the confusion matrix

sns.heatmap(dcm, annot=True, cmap="Blues", fmt="g")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.show()
```



In [51]:
```python
# Print the classification report
print(classification_report(y_test, dty_pred))
```

```
              precision    recall  f1-score   support

           0       0.81      0.81      0.81      1036
           1       0.48      0.49      0.48       373

    accuracy                           0.72      1409
   macro avg       0.65      0.65      0.65      1409
weighted avg       0.73      0.72      0.72      1409
```

Key Insights:

- Logistic Regression outperforms other models in terms of accuracy, precision, and recall.
- Decision Tree shows lower precision and recall compared to other models

In [52]:
```python
# 7 Machine Learning Algorithms will be applied to the dataset

classifiers = [[XGBClassifier(), 'XGB Classifier'],
               [RandomForestClassifier(), 'Random forest'],
               [SGDClassifier(), 'SGD Classifier'],
               [SVC(), 'SVC'],
               [GaussianNB(), "Naive Bayes"],
               [DecisionTreeClassifier(random_state = 42), "Decision tree"],
               [LogisticRegression(), 'Logistics Regression']
              ]
```

In [53]:
```python
classifiers
```

Out[53]:
```
[[XGBClassifier(base_score=None, booster=None, callbacks=None,
                colsample_bylevel=None, colsample_bynode=None,
                colsample_bytree=None, device=None, early_stopping_rounds=None,
                enable_categorical=False, eval_metric=None, feature_types=None,
                gamma=None, grow_policy=None, importance_type=None,
                interaction_constraints=None, learning_rate=None, max_bin=None,
                max_cat_threshold=None, max_cat_to_onehot=None,
                max_delta_step=None, max_depth=None, max_leaves=None,
                min_child_weight=None, missing=nan, monotone_constraints=None,
                multi_strategy=None, n_estimators=None, n_jobs=None,
                num_parallel_tree=None, random_state=None, ...),
  'XGB Classifier'],
 [RandomForestClassifier(), 'Random forest'],
 [SGDClassifier(), 'SGD Classifier'],
 [SVC(), 'SVC'],
 [GaussianNB(), 'Naive Bayes'],
 [DecisionTreeClassifier(random_state=42), 'Decision tree'],
 [LogisticRegression(), 'Logistics Regression']]
```

In [54]:
```python
acc_list = {}
precision_list = {}
recall_list = {}
roc_list = {}

for classifier in classifiers:
    model = classifier[0]
    model.fit(X_train, y_train)
    model_name = classifier[1]

    pred = model.predict(X_test)

    a_score = accuracy_score(y_test, pred)
```

```
    p_score = precision_score(y_test, pred)
    r_score = recall_score(y_test, pred)
    roc_score = roc_auc_score(y_test, pred)

    acc_list[model_name] = [str(round(a_score * 100, 2)) + '%']
    precision_list[model_name] = [str(round(p_score * 100, 2)) + '%']
    recall_list[model_name] = [str(round(r_score * 100, 2)) + '%']
    roc_list[model_name] = [str(round(roc_score * 100, 2)) + '%']

    if model_name != classifiers[-1][1]:
        print('')
```

In [55]:
```
print("Accuracy Score")
s1 = pd.DataFrame(acc_list)
s1.head()
```

```
Accuracy Score
```

Out[55]:

	XGB Classifier	Random forest	SGD Classifier	SVC	Naive Bayes	Decision tree	Logistics Regression
0	77.86%	79.42%	79.49%	80.84%	75.94%	71.68%	81.76%

In [ ]:

In [56]:
```
print("Precision Score")
s2 = pd.DataFrame(precision_list)
s2.head()
```

```
Precision Score
```

Out[56]:

	XGB Classifier	Random forest	SGD Classifier	SVC	Naive Bayes	Decision tree	Logistics Regression
0	59.44%	64.98%	73.08%	69.29%	53.11%	46.67%	68.59%

In [ ]:

In [57]:
```
print("Recall Score")
s3 = pd.DataFrame(recall_list)
s3.head()
```

```
Recall Score
```

Out[57]:

	XGB Classifier	Random forest	SGD Classifier	SVC	Naive Bayes	Decision tree	Logistics Regression
0	51.47%	48.26%	35.66%	49.6%	77.75%	48.79%	57.37%

In [ ]:

In [58]:
```
print("ROC Score")
s4 = pd.DataFrame(roc_list)
s4.head()
```

```
ROC Score
```

Out[58]:

	XGB Classifier	Random forest	SGD Classifier	SVC	Naive Bayes	Decision tree	Logistics Regression
0	69.41%	69.45%	65.46%	70.84%	76.52%	64.36%	73.96%

Primary Metrics for Churn Prediction:

The two primary metrics for churn prediction are:

- Precision: Focuses on minimizing false positives, ensuring that customers predicted to churn are likely to do so.
- Recall: Emphasizes minimizing false negatives, ensuring that actual churners are correctly identified.

In [ ]:

```

```

PROJECT REPORT AND SUMMARY

PROJECT TITLE: ConnectTel Customer Churn Prediction using Supervised Machine Learning

AUTHOR: Adewale Odetara

DATE: 14th November, 2023

Introduction:

In the dynamic landscape of telecommunications, customer churn poses a significant challenge for companies like ConnectTel. The ability to predict and understand customer churn is crucial for business sustainability and growth. In this project, I delve into the realm of churn prediction, leveraging machine learning techniques to develop models capable of identifying customers at risk of leaving the service.

Project Background

ConnectTel is facing a client retention difficulty that threatens the company's long-term viability and growth. Customer churn prediction predicts potential customers to leave a company's service, requiring effective marketing strategies to increase their likelihood of staying.

Project Objective

The primary goal is to develop an accurate and reliable predictive model using machine learning to predict which customers are likely to churn and implement proactive measures.

Data Loading and Cleaning

The project commenced with the crucial phase of data loading and cleaning, which involved: • Preview the dataset to familiarize myself with its structure and contents. • Identified and standardized data types for consistency. • Detected and eliminated duplicate entries to ensure data integrity.

Exploratory Data Analysis (EDA) To uncover trends or patterns, obtain insights, and remove unnecessary values from the data, I conducted univariate, bivariate, and multivariate analyses to gain an in-depth knowledge of the data and learn about its various features.

Data Preprocessing:

1. Feature Engineering: • Identified and created relevant features that can contribute to the prediction of churn. • Handled missing values and outliers appropriately.

2. Encoding: • I used label encoding to convert categorical variables into a format suitable for machine learning models.

3. Scaling: • I normalized numerical features to ensure a level playing field for machine learning algorithms.

Model Building:

Split the dataset into a training set and a testing set. A common split is 80% for training and 20% for testing.

• Dataset Split: The dataset was divided into 80% for training and 20% for testing. • Models Implemented:

- Logistic Regression
- Random Forest
- Decision Tree

Model Evaluation:

The performance of each model was evaluated using key metrics: • Accuracy: overall correctness of the model predictions. • Precision: proportion of true positives among instances predicted as positive. • Recall: proportion of true positives among actual positive instances. • AUC (Area Under the Curve): the area under the ROC curve, measuring the model's ability to distinguish between classes.

Key Insights:

• Logistic Regression stands out with the highest accuracy and a balanced precision-recall trade-off. • Decision Tree shows lower precision and recall compared to other models. • Random Forest provides a decent accuracy but has lower precision and recall compared to Logistic Regression. It may benefit from tuning hyperparameters to improve performance. • Naïve Bayes is notable for high recall, making it suitable for scenarios where capturing all churn instances is crucial. • Consider business priorities and the cost associated with false positives and false negatives when choosing a model.

Confusion Matrix Analysis

Interpretation:

• True Positives (TP): The number of customers correctly predicted as churners. • True Negatives (TN): The number of customers correctly predicted as non-churners. • False Positives (FP): The number of customers incorrectly predicted as churners. • False Negatives (FN): The number of customers incorrectly predicted as non-churners.

Model Comparison:

• Logistic Regression has the highest True Positives (214) and the fewest False Positives (159), indicating better performance in identifying actual churners. • Random Forest has slightly fewer True Positives but also fewer False Positives compared to Decision Tree. • Decision Tree has the highest False Positives and False Negatives, suggesting lower precision and recall compared to the other models.

Recommendation:

• Logistic Regression appears to be performing better in this scenario, but the choice of the model depends on the specific goals and requirements of the business. Consider the trade-off between false positives and false negatives based on the business impact of predicting churn incorrectly.

The Logistic Regression model has a balanced distribution of false positives and false negatives. It demonstrates a higher ability to correctly identify non-churn instances (True Negatives) compared to correctly identifying churn instances (True Positives). The model's precision and recall can be further optimized by adjusting the classification threshold.

Similar to Logistic Regression, Random Forest exhibits a balanced distribution of false positives and false negatives. It performs slightly worse in correctly identifying both churn and non-churn instances compared to Logistic Regression. Random Forest's strength lies in ensemble learning, providing robustness against overfitting.

The Decision Tree model demonstrates a higher rate of false positives compared to both Logistic Regression and Random Forest. It shows comparable performance in correctly identifying churn instances (True Positives) but struggles with precision due to a higher false positive count. Decision Trees may benefit from pruning or tuning hyperparameters to improve overall performance.

General Observations:

The models generally perform better at identifying non-churn instances (True Negatives) than identifying churn instances (True Positives). The choice between models depends on the specific business requirements and the importance of precision and recall in the context of customer churn.

Conclusion and Recommendations:

In conclusion, the project highlights the potential of machine learning in predicting customer churn. ConnectTel can benefit from deploying the Logistic Regression model for its superior performance. Explore ensemble methods or model stacking to combine the strengths of different models.

Recommendations include continuous monitoring, periodic model updates, and leveraging insights from misclassifications to enhance model robustness and business strategies. Fine-tune model parameters and thresholds to balance precision and recall based on business goals. By prioritizing precision and recall, ConnectTel can strategically address customer churn, fostering long-term customer relationships and business success.

In [ ]:

In [ ]: