Forecasting Telecom Customer Churn for Proactive Retention and Business Success.

Team Members:

Revathi Gollapudi	vgollapu@buffalo.edu
Srikar Chintha	chintha@buffalo.edu
Sumana Madhireddy	sumanama@buffalo.edu

Problem Statement:

The telecommunications industry is dealing with a prevalent issue of customer churn, where subscribers discontinue services, leading to revenue loss and market share erosion. The problem at hand is to develop an effective predictive model for telecom customer churn and subsequently implement strategic retention measures.

This project aims to address the following questions:

What are the main factors that contribute to customer churn in the telecom sector?

Can predictive modeling effectively predict individuals likely to churn?

How can a telecom company leverage predictive insights to implement targeted retention strategies?

Are there any observable patterns or trends in customer churn that can be linked to gender of the individual?

Background of the Problem:

The telecommunications industry operates within a highly competitive environment where fostering customer loyalty is crucial for sustained growth. Churn, the phenomenon of customers switching to rival providers, poses a significant challenge, influenced by factors such as service quality, pricing, and emerging technologies. The rising costs associated with acquiring new customers underscore the importance for telecom companies to effectively address churn. It is imperative to comprehend the predictors and patterns of customer churn to formulate targeted retention initiatives.

Objectives:

Factors contributing to Customer Churn: Examine historical customer data to discern patterns and indicators that precede instances of churn.

Develop a Predictive Model: Create a robust predictive analytics model using machine learning algorithms to forecast potential customer churn.

Implement Retention Strategies: Translate insights from the model into actionable strategies for customer retention, including personalized offerings and targeted communication.

Detect any patterns: From the analysis of data using EDA, any patterns that are leading to customer churn can be detected.

Significance of the Problem:

Churn not only results in an immediate loss of revenue but also incurs additional expenses related to marketing and customer acquisition. Moreover, the impact extends beyond financial implications; in an era where customer experience is a significant differentiator, high churn rates can tarnish a telecom company's reputation, potentially affecting long-term brand equity.

By addressing churn proactively, telecom providers can not only mitigate financial losses but also enhance customer satisfaction, which is crucial in retaining a loyal customer base. Proactive churn management enables companies to optimize resource allocation, directing efforts towards retaining valuable customers rather than solely focusing on acquisition.

In the competitive landscape of the telecommunications industry, where technological advancements and pricing strategies are dynamic, maintaining a low churn rate becomes a strategic imperative.

Furthermore, beyond the quantitative aspects of churn analysis, qualitative insights from customer feedback and surveys play a crucial role. This holistic approach ensures that retention initiatives are not only data-driven but also align with the evolving needs and expectations of the customer base.

In conclusion, a comprehensive strategy that combines data analytics, customer feedback, and proactive initiatives is essential for telecom companies looking to navigate the challenges posed by churn. This approach not only helps in immediate financial gains but also useful in long-erm customer loyalty.

Potential Contribution of the Project:

This project has the potential to transform the way telecom companies handle customer churn, which is when customers switch to other providers. By using data and predictions, the goal is to provide valuable insights based on past customer behavior. This helps companies predict who might leave and allows them to take specific actions to prevent it.

The key is that the model being accurate and efficient can help telecom providers use their resources better. They can tailor their interactions with customers, making them more personal and addressing specific concerns. The ultimate aim is to reduce the number of customers leaving, which is crucial for the success of any telecom business.

It might show telecom companies how to use data analytics in a similar way, influencing how to make important decisions and improving their relationships with customers over the long term. The real value lies in changing the approach of telecom companies from reacting to customer losses to being proactive and preventing them in the first place.

Why above Contribution is Crucial?

This project contribution is crucial to a Telecom Company for the below reasons:

Impact of Revenue:

Churn prediction mitigates immediate revenue loss by retaining valuable subscribers, offering a proactive strategy against revenue decline.

Increased Efficiency:

Identifying potential churners optimizes resource allocation, focusing efforts on high-risk customers and making interventions more cost-effective.

Enhancing Customer Experience:

Predictive analytics enables personalized retention strategies, preventing churn and elevating overall customer experience, leading to increased loyalty.

Savings in Costs:

Investing in customer retention strategies offers significant long-term cost savings compared to the expenses associated with acquiring new customers.

Reputation Management:

Proactively managing churn preserves revenue and maintains a positive brand image, showcasing a company's attentiveness to customer needs.

Long-term Sustainability:

A proactive approach to churn management builds stronger customer relationships, ensuring the company's long-term sustainability and increased customer lifetime value.

Dataset Sources:

Dataset has been taken from the sample dataset from IBM sample dataset for Telecom Customer Churn.

https://www.ibm.com/docs/en/cognos-analytics/11.1.0?topic=samples-telco-customer-churn

The dataset has 7113 rows and 22 columns.

Below are the column names which are features for further analysis and their datatypes from the above dataset. Out of which 4 column are float type, 1 columns is of integer type and remaining 17 are Object type.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7113 entries, 0 to 7112
Data columns (total 22 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   customerID        7102 non-null   object
 1   gender            7073 non-null   object
 2   SeniorCitizen     7102 non-null   float64
 3   Partner           7102 non-null   object
 4   Dependents        7070 non-null   object
 5   tenure            7102 non-null   float64
 6   PhoneService      7102 non-null   object
 7   MultipleLines     7102 non-null   object
 8   InternetService   7102 non-null   object
 9   OnlineSecurity    7102 non-null   object
 10  OnlineBackup      7102 non-null   object
 11  DeviceProtection  7102 non-null   object
 12  TechSupport       7102 non-null   object
 13  StreamingTV       7102 non-null   object
 14  StreamingMovies   7102 non-null   object
 15  Contract          7058 non-null   object
 16  PaperlessBilling  7056 non-null   object
 17  PaymentMethod     7102 non-null   object
 18  MonthlyCharges    7053 non-null   float64
 19  TotalCharges      7102 non-null   float64
 20  Churn             7102 non-null   object
 21  Count             7113 non-null   int64
dtypes: float64(4), int64(1), object(17)
memory usage: 1.2+ MB
```

Data Cleaning:

1.Dropping the duplicates

Dropped the rows having duplicate values to ensure the data is unique.

In [29]:
```
# Checking Duplicate values

Duplic_rows=df[df.duplicated()]
Duplic_rows
```

Out[29]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	TechSupport	St
7043	7554-NEWDD	Male	0.0	No	No	10.0	Yes	Yes	No	No internet service	...	No internet service	
7044	9069-LGEUL	MALE	0.0	Yes	No	23.0	Yes	No	DSL	Yes	...	No	
7045	1964-SVLEA	Male	0.0	No	No	20.0	Yes	No	No	No internet service	...	No internet service	
7046	5696-QURRL	Male	0.0	No	No	1.0	Yes	No	DSL	No	...	No	
7047	7673-BQGKU	Female	0.0	Yes	Yes	69.0	Yes	No	No	No internet service	...	No internet service	
...	...	...	...	...	...	...	...	...	...	...	...	...	
7108	5766-ZJYBB	male	0.0	No	No	1.0	Yes	No	No	No internet service	...	No internet service	
	9798-												

In [6]:
```
#1.Drop the Duplicate rows

df=df.drop_duplicates()
df
```

2. Dropping unwanted columns

Dropped the Count column which doesn't have any significance in predicting whether a customer can continue or discontinue their service.

In [7]:
```
#Drop unwanted columns

df=df.drop('Count',axis=1)
df
```

Out[7]:

curity	...	DeviceProtection	TechSupport	StreamingTV	StreamingMovies	Contract	PaperlessBilling	PaymentMethod	MonthlyCharges	TotalCharges	Churn
No	...	No	No	No	No	Month-to-month	Yes	Electronic check	29.85	29.85	No
Yes	...	Yes	No	No	No	One year	No	Mailed check	56.95	1889.50	No
Yes	...	No	No	No	No	Month-to-month	Yes	Mailed check	53.85	108.15	Yes
Yes	...	Yes	Yes	No	No	One year	No	Bank transfer (automatic)	42.30	1840.75	No
No	...	No	No	No	No	Month-to-month	Yes	Electronic check	70.70	151.65	Yes
...	...	...	...	...	...	...	...	...	...	...	...
Yes	...	Yes	Yes	Yes	Yes	One year	Yes	Mailed check	84.80	1990.50	No
No	...	Yes	No	Yes	Yes	One year	Yes	Credit card (automatic)	103.20	7362.90	No
Yes	...	No	No	No	No	Month-to-month	Yes	Electronic check	29.60	346.45	No
No	...	No	No	No	No	Month-to-month	Yes	Mailed check	74.40	306.60	Yes
Yes	...	Yes	Yes	Yes	Yes	Two year	Yes	Bank transfer (automatic)	105.65	6844.50	No

3. Dropping the rows which have Null values in it.

Dropped the rows which have null values in any of its columns to ensure the data is uniform for prediction across all the rows

In [30]:
```
# Null value row count

row_null = df.isnull().sum()
row_null
```

Out[30]:
```
customerID          11
gender              40
SeniorCitizen       11
Partner             11
Dependents          43
tenure              11
PhoneService        11
MultipleLines       11
InternetService     11
OnlineSecurity      11
OnlineBackup        11
DeviceProtection    11
TechSupport         11
StreamingTV         11
StreamingMovies     11
Contract            55
PaperlessBilling    57
PaymentMethod       11
MonthlyCharges      60
TotalCharges        11
Churn               11
Count                0
dtype: int64
```

In [9]:
```
#Dropping null value rows

df=df.dropna()
df
```

4. Removing the Special Characters except '– '.

Removed the special characters in CustomerID column to ensure entire customerID column is uniform with the rows.

In [45]:
```
#Removing the special characters except '-'
special_chars = ['*','.', '&', '\\', '\/', ';', '\(','\)','_']

for i in special_chars:
    df["customerID"] = df["customerID"].str.strip(i)

df
```

Out[45]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceProtection
0	7590-VHVEG	Female	0.0	Yes	No	1.0	No	No phone service	DSL	No	...	No
2	3668-QPYBK	Male	0.0	No	No	2.0	Yes	No	DSL	Yes	...	No
3	7795-CFOCW	Male	0.0	No	No	45.0	No	No phone service	DSL	Yes	...	Yes
4	9237-HQITU	FEMALE	0.0	No	No	2.0	Yes	No	Fiber optic	No	...	No
5	9305-CDSKC	Female	0.0	No	No	8.0	Yes	Yes	Fiber optic	No	...	Yes
...	...	...	...	...	...	...	...	...	...	...	...	...
7038	6840-RESVB	Male	0.0	Yes	Yes	24.0	Yes	Yes	DSL	Yes	...	Yes
7039	2234-XADUH	female	0.0	Yes	Yes	72.0	Yes	Yes	Fiber optic	No	...	Yes
7040	4801-JZAZL	Female	0.0	Yes	Yes	11.0	No	No phone service	DSL	Yes	...	No
7041	8361-LTMKD	Male	1.0	Yes	No	4.0	Yes	Yes	Fiber optic	No	...	No
7042	3186-AJIEK	Male	0.0	No	No	66.0	Yes	No	Fiber optic	Yes	...	Yes

6831 rows × 21 columns

5. Converting the gender column to Pascal's case for uniformity.

There are some upper case, some lower case for male and female which could lead to confusion while doing categorical encoding. So, converted the entire gender column to Pascal case for uniformity.

In [46]:
```
#Converting to Pascal case

df['gender']=df['gender'].str.title()
df
```

Out[46]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceProtection	T
0	7590-VHVEG	Female	0.0	Yes	No	1.0	No	No phone service	DSL	No	...	No	
2	3668-QPYBK	Male	0.0	No	No	2.0	Yes	No	DSL	Yes	...	No	
3	7795-CFOCW	Male	0.0	No	No	45.0	No	No phone service	DSL	Yes	...	Yes	
4	9237-HQITU	Female	0.0	No	No	2.0	Yes	No	Fiber optic	No	...	No	
5	9305-CDSKC	Female	0.0	No	No	8.0	Yes	Yes	Fiber optic	No	...	Yes	
...	...	...	...	...	...	...	...	...	...	...	...	...	
7038	6840-RESVB	Male	0.0	Yes	Yes	24.0	Yes	Yes	DSL	Yes	...	Yes	
7039	2234-XADUH	Female	0.0	Yes	Yes	72.0	Yes	Yes	Fiber optic	No	...	Yes	
7040	4801-JZAZL	Female	0.0	Yes	Yes	11.0	No	No phone service	DSL	Yes	...	No	
7041	8361-LTMKD	Male	1.0	Yes	No	4.0	Yes	Yes	Fiber optic	No	...	No	
7042	3186-AJIEK	Male	0.0	No	No	66.0	Yes	No	Fiber optic	Yes	...	Yes	

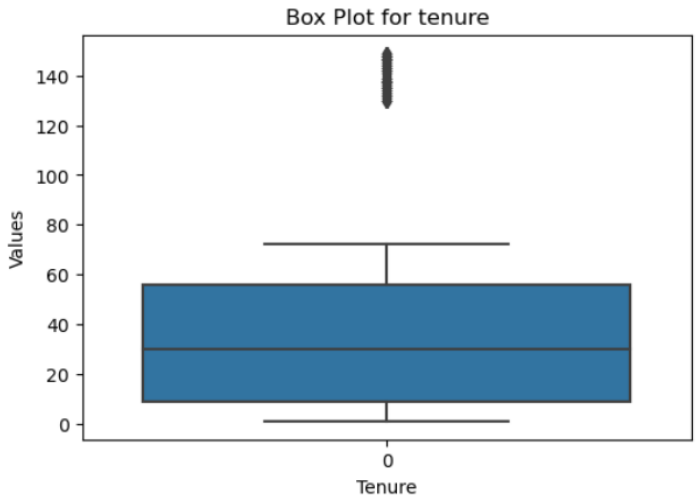
6831 rows × 21 columns

6. Box plot for Tenure column to find the outliers

Have drawn a box plot to visualize outliers in Tenure column to fix if outliers are found

In [34]:
```
#Visualization for outliers in tenure

plt.figure(figsize=(6, 4))
sns.boxplot(df['tenure'])
plt.xlabel('Tenure')
plt.ylabel('Values')
plt.title('Box Plot for tenure')
plt.show()
```



7. Filling Outliers with Mean

Filled the outliers by finding the mean of the tenure column and replacing outliers with mean.

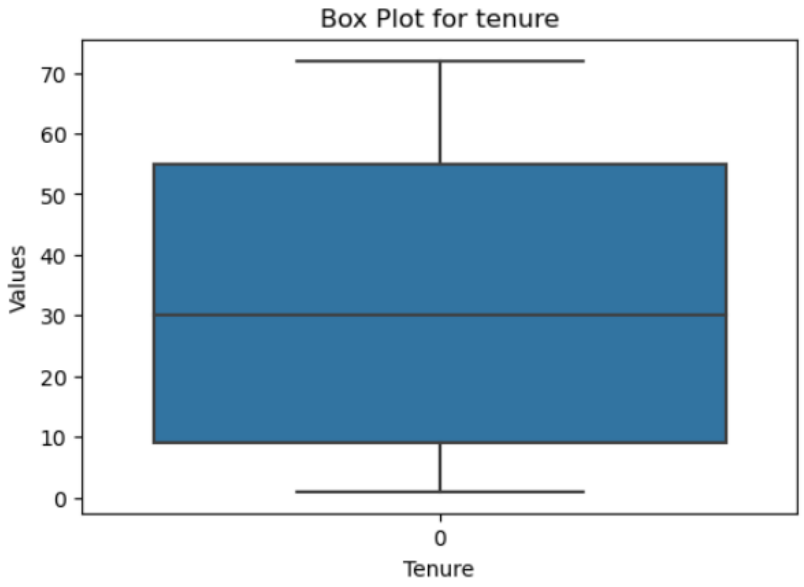
In [13]:
```
#6. outlier fix for tenure

mean_tenure = (df[(df['tenure'] <= 75)]['tenure']).mean()
print(int(mean_tenure))
df.loc[(df['tenure'] >= 75), 'tenure'] = int(mean_tenure)
```

```
32
```

In [14]:
```
#After the outlier treatment

plt.figure(figsize=(6, 4))
sns.boxplot(df['tenure'])
plt.xlabel('Tenure')
plt.ylabel('Values')
plt.title('Box Plot for tenure')
plt.show()
```



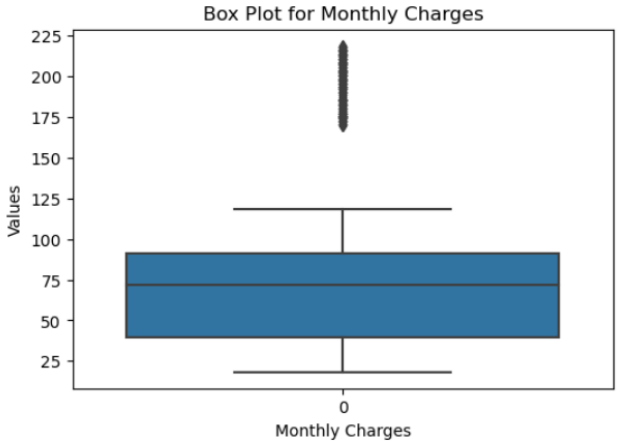
8. Checking Outliers for Monthly charges and filling it with Mean.

Drawn a box plot to detect outliers for Monthly charges column and filled the outliers with mean of the column.

tenure

In [15]:
```
#Checking outliers for Monthly charges

plt.figure(figsize=(6, 4))
sns.boxplot(df['MonthlyCharges'])
plt.xlabel('Monthly Charges')
plt.ylabel('Values')
plt.title('Box Plot for Monthly Charges')
plt.show()
```



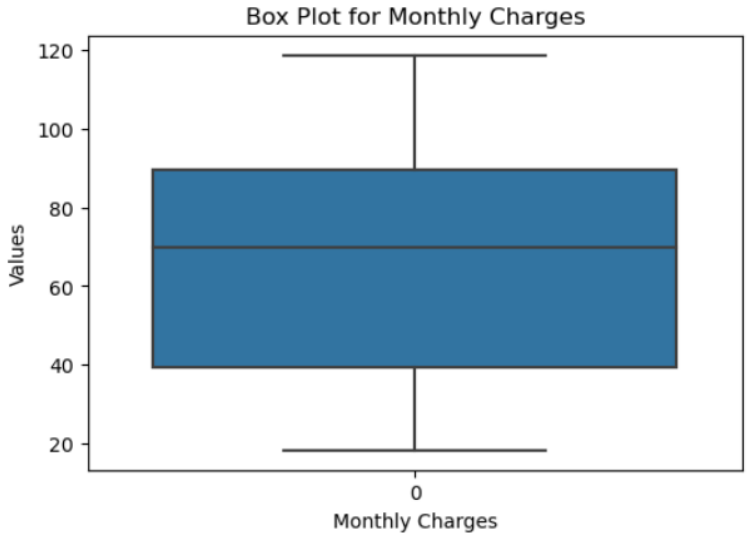
In [51]:
```
#Outlier fix for MonthlyCharges

mean_MonthlyCharges = (df[(df['MonthlyCharges'] <= 120)]['MonthlyCharges']).mean()
print(int(mean_MonthlyCharges))
df.loc[(df['MonthlyCharges'] >= 120), 'MonthlyCharges'] = int(mean_MonthlyCharges)
```

```
64
```

In [52]:
```
#After the outlier treatment

plt.figure(figsize=(6, 4))
sns.boxplot(df['MonthlyCharges'])
plt.xlabel('Monthly Charges')
plt.ylabel('Values')
plt.title('Box Plot for Monthly Charges')
plt.show()
```



9. Normalization of the numerical column to get the values between 0 and 1

Normalized the entire numerical columns (tenure, MonthlyCharges, TotalCharges) to have values between 0 and 1 for ease of comaprison

In [53]:
```
#Normalization for columns with numirical values

df['tenure']=(df['tenure']-min(df['tenure']))/(max(df['tenure'])-min(df['tenure']))

df['MonthlyCharges']=(df['MonthlyCharges']-min(df['MonthlyCharges']))/(max(df['MonthlyCharges'])-min(df['MonthlyCharges']))

df['TotalCharges']=(df['TotalCharges']-min(df['TotalCharges']))/(max(df['TotalCharges'])-min(df['TotalCharges']))
```

In [19]:
```
df
```

Out[19]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceProtection
0	7590-VHVEG	Female	0.0	Yes	No	0.000000	No	No phone service	DSL	No	...	No
2	3668-QPYBK	Male	0.0	No	No	0.014085	Yes	No	DSL	Yes	...	No
3	7795-CFOCW	Male	0.0	No	No	0.619718	No	No phone service	DSL	Yes	...	Yes
4	9237-HQITU	Female	0.0	No	No	0.014085	Yes	No	Fiber optic	No	...	No
5	9305-CDSKC	Female	0.0	No	No	0.098592	Yes	Yes	Fiber optic	No	...	Yes
...	...	...	...	...	...	...	...	...	...	...	...	...
7038	6840-RESVB	Male	0.0	Yes	Yes	0.323944	Yes	Yes	DSL	Yes	...	Yes
7039	2234-XADUH	Female	0.0	Yes	Yes	1.000000	Yes	Yes	Fiber optic	No	...	Yes
7040	4801-JZAZL	Female	0.0	Yes	Yes	0.140845	No	No phone service	DSL	Yes	...	No
7041	8361-LTMKD	Male	1.0	Yes	No	0.042254	Yes	Yes	Fiber optic	No	...	No
7042	3186-AJIEK	Male	0.0	No	No	0.915493	Yes	No	Fiber optic	Yes	...	Yes

6831 rows × 21 columns

10. One hot encoding for String (Categorical) columns

Performed one hot encoding to categorical columns for using them for further analysis of training and in predicting the churn.

In [54]:

```
#Encoding for string columns

df['customerID'] = df['customerID'].astype('category').cat.codes
df['gender'] = df['gender'].astype('category').cat.codes
df['Partner'] = df['Partner'].astype('category').cat.codes
df['Dependents'] = df['Dependents'].astype('category').cat.codes
df['PhoneService'] = df['PhoneService'].astype('category').cat.codes
df['MultipleLines'] = df['MultipleLines'].astype('category').cat.codes
df['InternetService'] = df['InternetService'].astype('category').cat.codes
df['OnlineSecurity'] = df['OnlineSecurity'].astype('category').cat.codes
df['DeviceProtection'] = df['DeviceProtection'].astype('category').cat.codes
df['TechSupport'] = df['TechSupport'].astype('category').cat.codes
df['StreamingTV'] = df['StreamingTV'].astype('category').cat.codes
df['StreamingMovies'] = df['StreamingMovies'].astype('category').cat.codes
df['OnlineBackup'] = df['OnlineBackup'].astype('category').cat.codes
df['Contract'] = df['Contract'].astype('category').cat.codes
df['PaperlessBilling'] = df['PaperlessBilling'].astype('category').cat.codes
df['PaymentMethod'] = df['PaymentMethod'].astype('category').cat.codes
df['Churn'] = df['Churn'].astype('category').cat.codes
```

In [22]:

Out[22]:

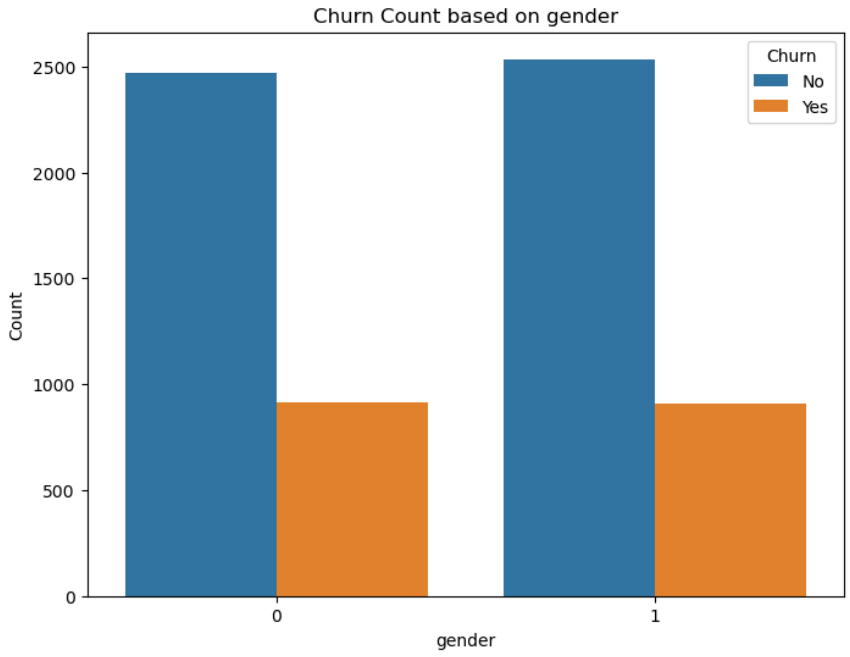
	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceProtection
0	5198	0	0.0	1	0	0.000000	0	1	0	0	...	0
2	2480	1	0.0	0	0	0.014085	1	0	0	2	...	0
3	5355	1	0.0	0	0	0.619718	0	1	0	2	...	2
4	6304	0	0.0	0	0	0.014085	1	0	1	0	...	0
5	6344	0	0.0	0	0	0.098592	1	2	1	0	...	2
...	...	...	...	...	...	...	...	...	...	...	...	...
7038	4690	1	0.0	1	1	0.323944	1	2	0	2	...	2
7039	1477	0	0.0	1	1	1.000000	1	2	1	0	...	2
7040	3249	0	0.0	1	1	0.140845	0	1	0	2	...	0
7041	5740	1	1.0	1	0	0.042254	1	2	1	0	...	0
7042	2151	1	0.0	0	0	0.915493	1	0	1	2	...	2

6831 rows × 21 columns

Exploratory Data Analysis (EDA):

1. Plotting Churn count based on Gender to know how many males and females have discontinued or continued their subscription.

0 - Female, 1- Male

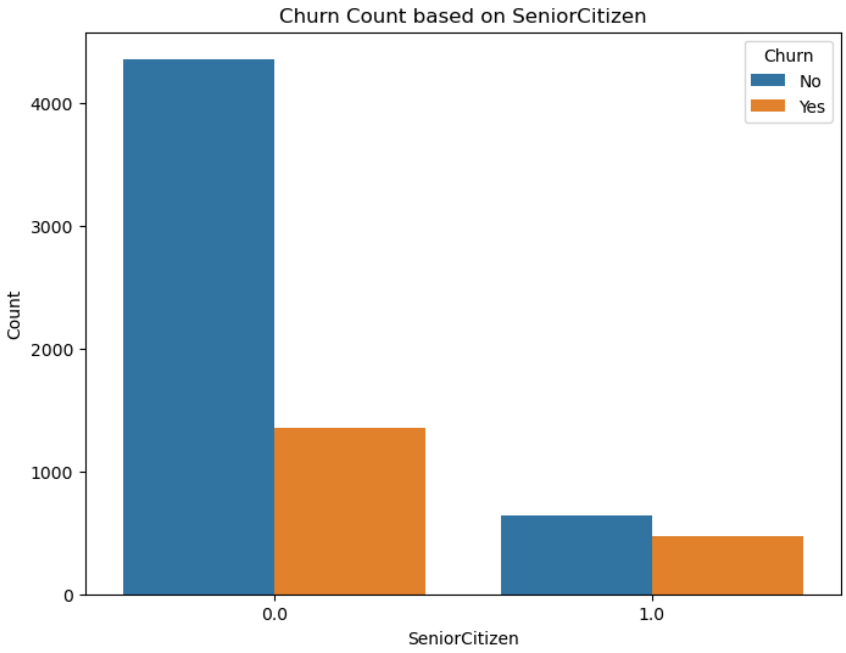



```
print('Female count for churn = Yes:', df[(df['gender'] == 0) & (df['Churn'] == 1)].shape[0])
print('Female count for churn = No:', df[(df['gender'] == 0) & (df['Churn'] == 0)].shape[0])
print('Male count for churn = Yes:', df[(df['gender'] == 1) & (df['Churn'] == 1)].shape[0])
print('Male count for churn = No:', df[(df['gender'] == 1) & (df['Churn'] == 0)].shape[0])
```

```
Female count for churn = Yes: 916
Female count for churn = No: 2469
Male count for churn = Yes: 911
Male count for churn = No: 2535
```

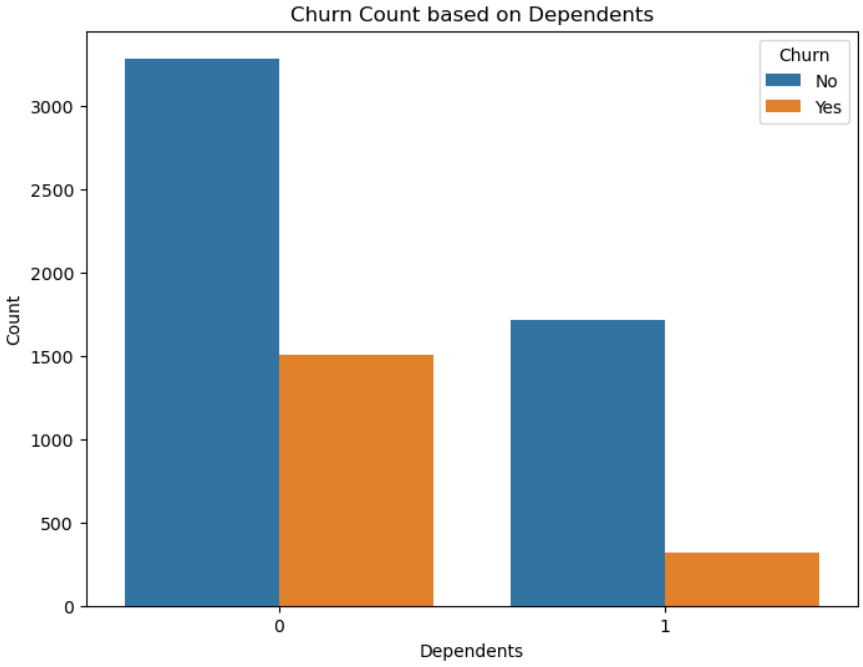
2. Plotting Churn count based on SeniorCitizen to know how many SeniorCitizens have discontinued or continued their subscription.

0.0 - Not SeniorCitizen, 1.0 – SeniorCitizen



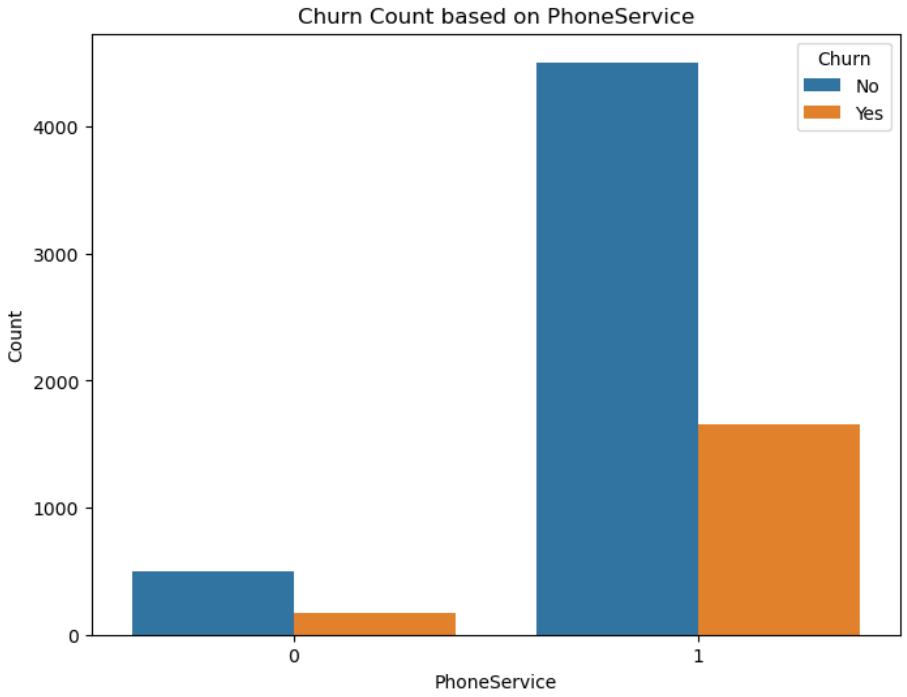
3. Plotting Churn count based on Dependents to know how many customers with Dependents have discontinued or continued their subscription.

0 - Not having Dependents, 1 - having Dependents



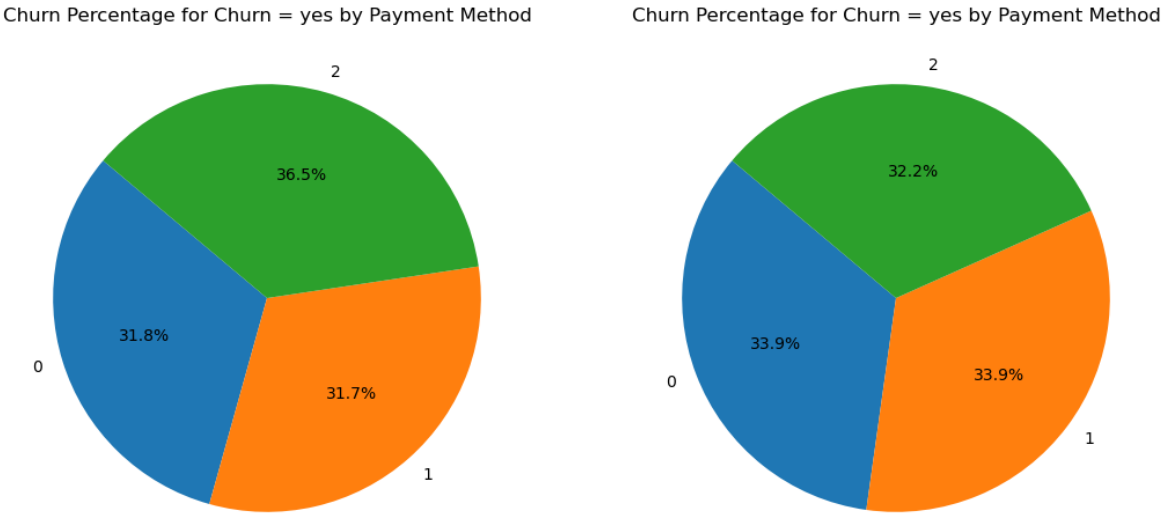
4. Plotting Churn count based on PhoneService to know how many customers choosen PhoneService have discontinued or continued their subscription.

0 - Not having PhoneService, 1 - having PhoneService



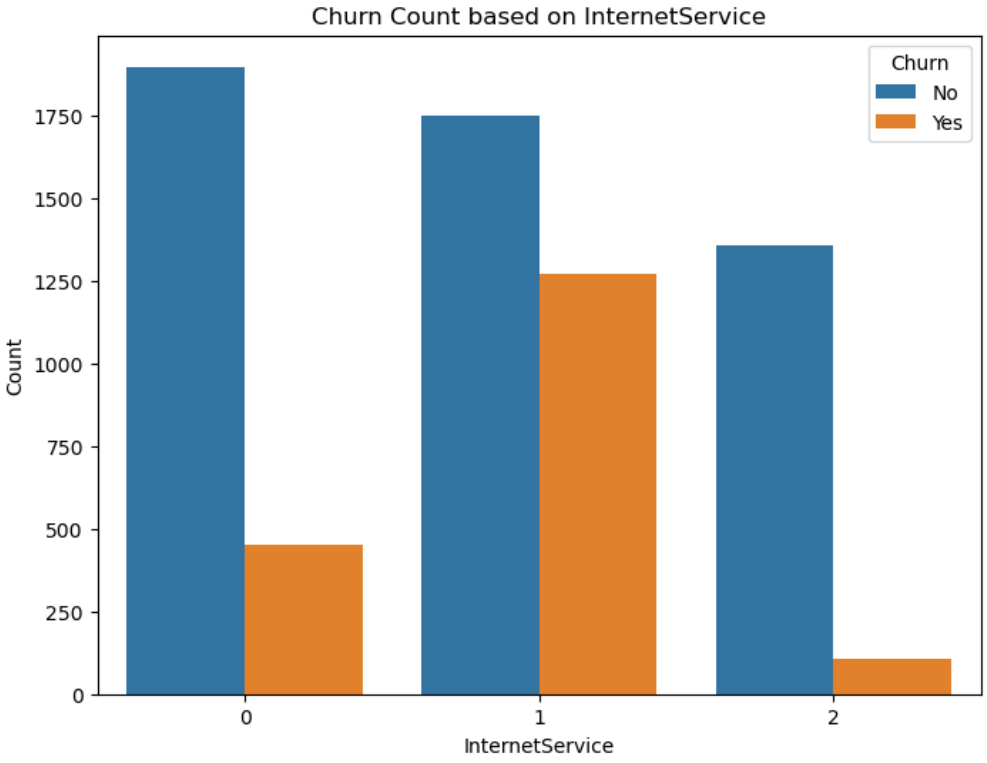
5. Plotting Churn count based on MultipleLines to know how many customers have MultipleLines have discontinued or continued their subscription.

0 - Not having MultipleLines, 1 - No phone service 2 - having MultipleLines



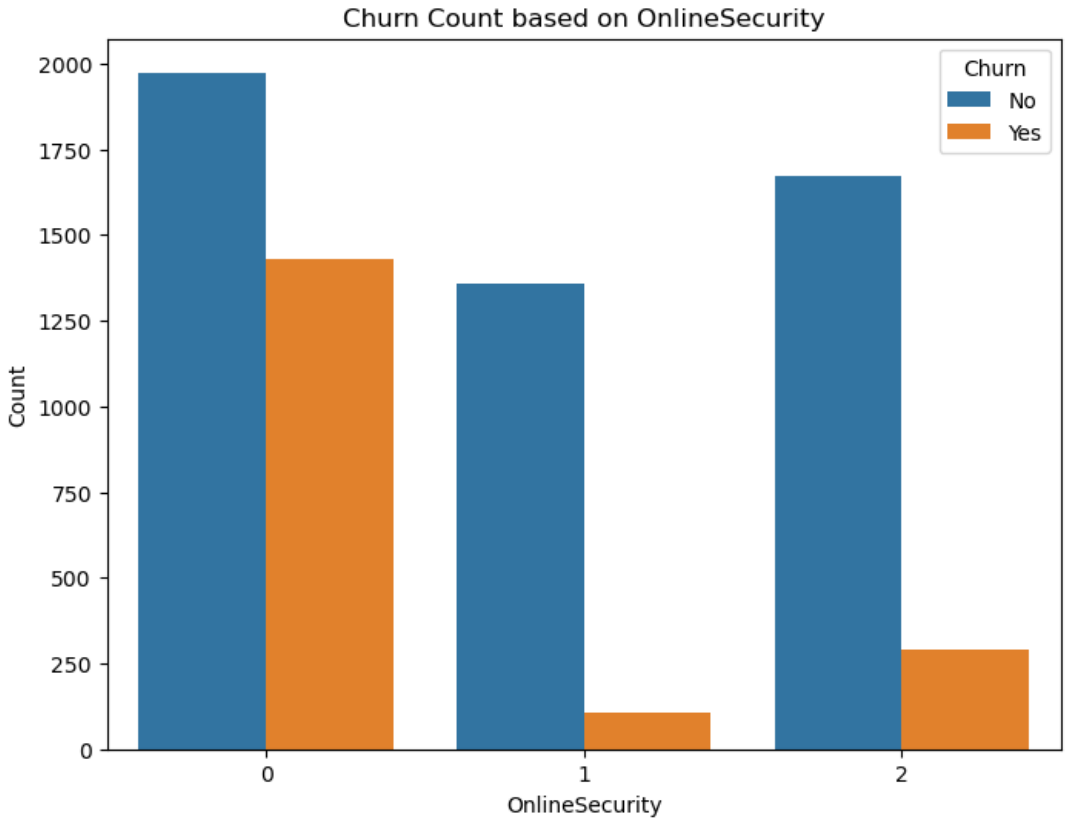
6. Plotting Churn count based on InternetService to know how many customers have InternetService have discontinued or continued their subscription.

0 - DSL, 1 - Fiber optic, 2 – No



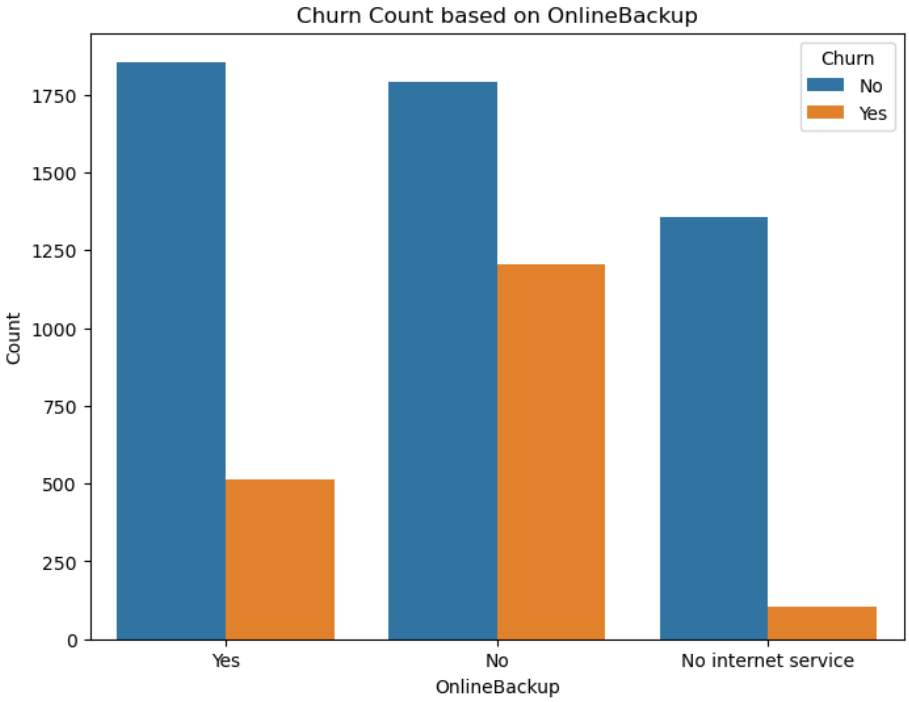
7. Plotting Churn count based on OnlineSecurity to know how many customers have OnlineSecurity have discontinued or continued their subscription.

0 - No, 1 - No internet service, 2 – Yes



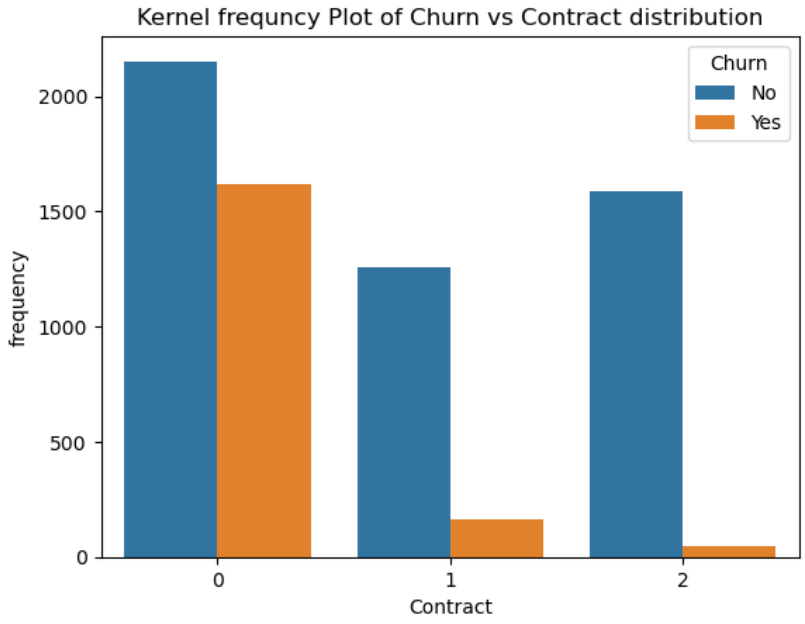
8. Plotting Churn count based on OnlineBackup to know how many customers have OnlineBackup have discontinued or continued their subscription.

0 - No, 1 - No internet service, 2 – Yes



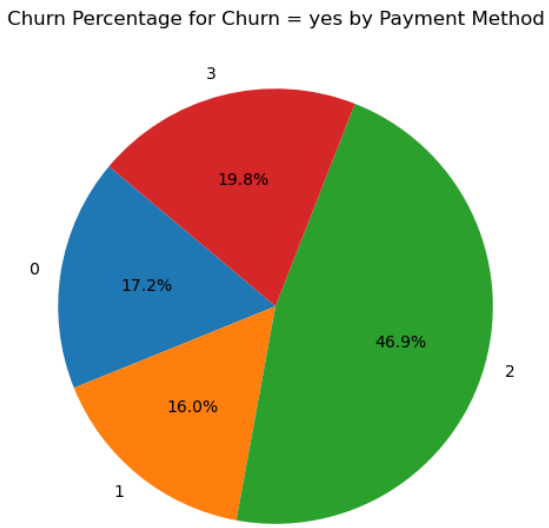
9. Plotting Churn count based on Contract to know how many customers have Contract have discontinued or continued their subscription.

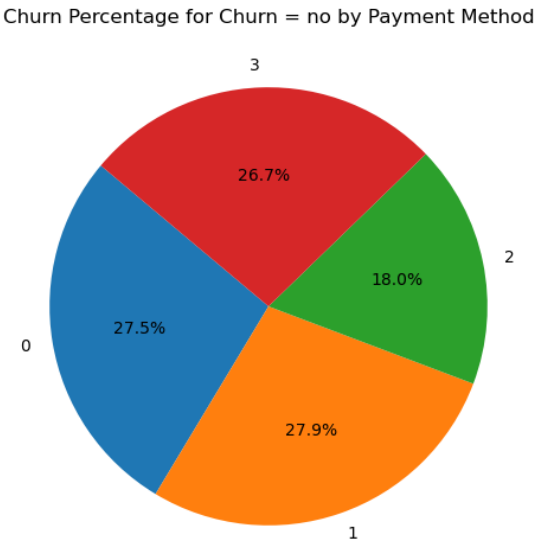
0 - Month-to-month, 1 - One year, 2 - Two year



10. Plotting Churn count based on PaymentMethod to know how many customers have PaymentMethod have discontinued or continued their subscription.

0- Bank transfer (automatic) ,1- Credit card (automatic), 2 - Electronic check, 3- Mailed check





From the first pie chart,

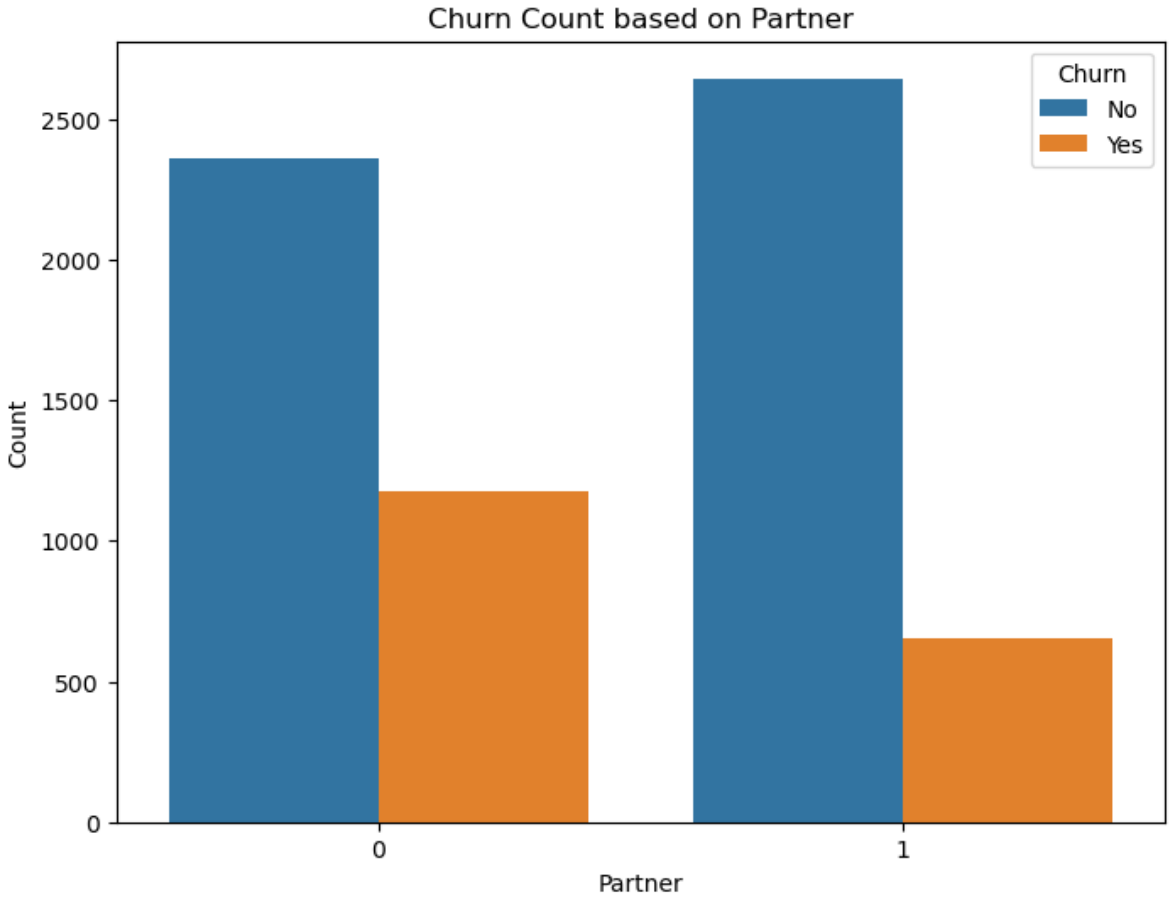
→we can identify that Electronic Check payment method has a higher percentage of churn ('Yes') relative to others. And, Credit Card payment method has a lower percentage of churn ('Yes') compared to others.

Similarly, from the second pie chart

→we can identify that Credit Card payment method has a higher percentage of churn ('No') relative to others. And, Electronic Check payment method has a lower percentage of churn ('No') compared to others.

11. Plotting Churn count based on Partner to know how many customers who are Partner have discontinued or continued their subscription.

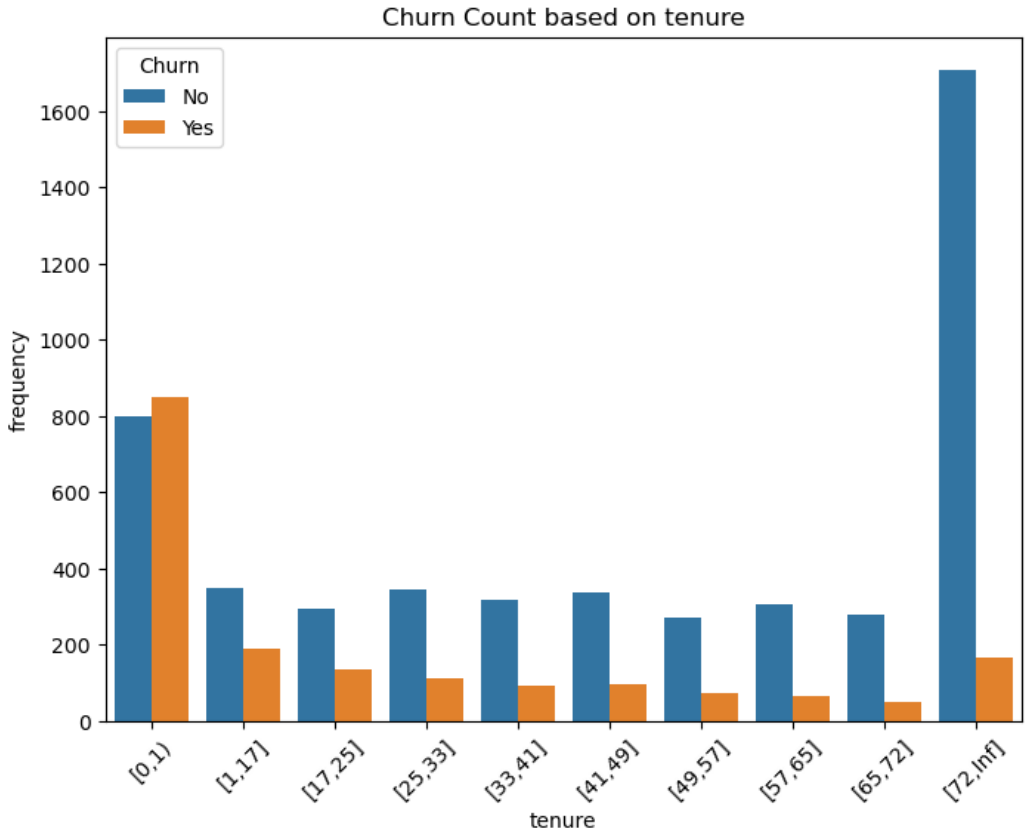
0- No ,1- Yes



12. Plotting Churn count based on tenure to know how many customers have discontinued or continued their subscription.

Categorizing the tenure into groups like [0,1), [1,17], [17,25], [25,33],… [65,72]

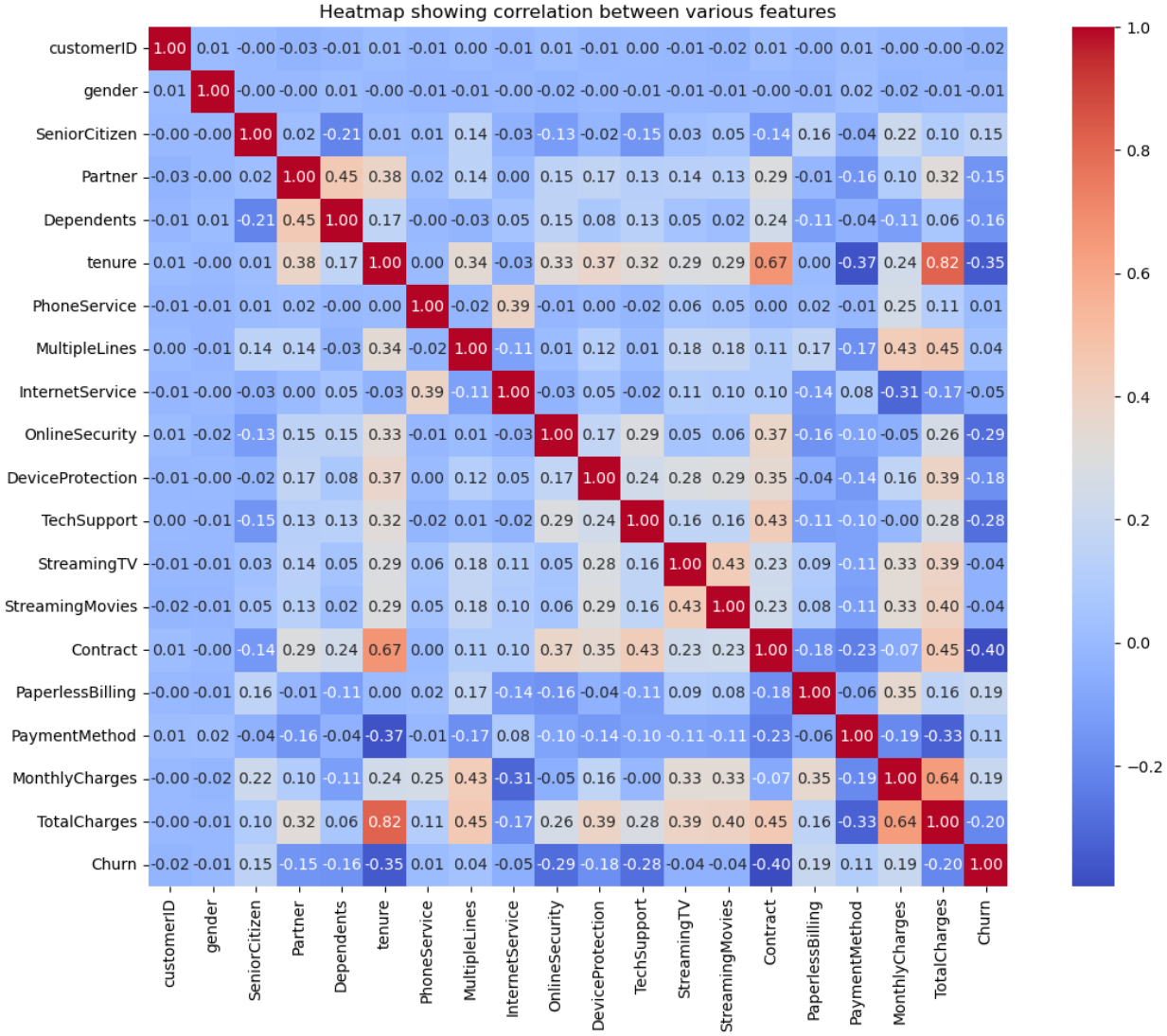
```
bins = [0, 0.1, 0.17, 0.25, 0.33, 0.41, 0.49, 0.57, 0.65, 0.72, 1]
labels = ['[0,1)', '[1,17]', '[17,25]', '[25,33]', '[33,41]', '[41,49]', '[49,57]', '[57,65]', '[65,72]', '[72,Inf]']
df['tenure_cat'] = pd.cut(df['tenure'], bins=bins, labels=labels, include_lowest=True)
```



13. The heatmap depicts the correlation matrix of the dataset.

From the heatmap we can find the Key observations:

1. Total Charges and Tenure has positive correlation (0.82), indicating that Total charges and Tenure are correlated and as Tenure increases total charges also increases.
2. Contract and Tenure has positive correlation (0.67), indicating that Contract and Tenure are correlated and as Contract increases also increases.
3. TechSupport and Contract has positive correlation (0.43), indicating that Contract and TechSupport are correlated.


Heatmap showing correlation between various features
customerID 1.00 0.01 -0.00 -0.03 -0.01 0.01 -0.01 0.00 -0.01 0.01 -0.01 0.00 -0.01 -0.02 0.01 -0.00 0.01 -0.00 -0.00 -0.02
gender 0.01 1.00 -0.00 -0.00 0.01 -0.00 -0.01 -0.01 -0.00 -0.02 -0.00 -0.01 -0.01 -0.01 -0.00 -0.01 0.02 -0.02 -0.01 -0.01
SeniorCitizen -0.00 -0.00 1.00 0.02 -0.21 0.01 0.01 0.14 -0.03 -0.13 -0.02 -0.15 0.03 0.05 -0.14 0.16 -0.04 0.22 0.10 0.15
Partner -0.03 -0.00 0.02 1.00 0.45 0.38 0.02 0.14 0.00 0.15 0.17 0.13 0.14 0.13 0.29 -0.01 -0.16 0.10 0.32 -0.15
Dependents -0.01 0.01 -0.21 0.45 1.00 0.17 -0.00 -0.03 0.05 0.15 0.08 0.13 0.05 0.02 0.24 -0.11 -0.04 -0.11 0.06 -0.16
tenure 0.01 -0.00 0.01 0.38 0.17 1.00 0.00 0.34 -0.03 0.33 0.37 0.32 0.29 0.29 0.67 0.00 -0.37 0.24 0.82 -0.35
PhoneService -0.01 -0.01 0.01 0.02 -0.00 0.00 1.00 -0.02 0.39 -0.01 0.00 -0.02 0.06 0.05 0.00 0.02 -0.01 0.25 0.11 0.01
MultipleLines 0.00 -0.01 0.14 0.14 -0.03 0.34 -0.02 1.00 -0.11 0.01 0.12 0.01 0.18 0.18 0.11 0.17 -0.17 0.43 0.45 0.04
InternetService -0.01 -0.00 -0.03 0.00 0.05 -0.03 0.39 -0.11 1.00 -0.03 0.05 -0.02 0.11 0.10 0.10 -0.14 0.08 -0.31 -0.17 -0.05
OnlineSecurity 0.01 -0.02 -0.13 0.15 0.15 0.33 -0.01 0.01 -0.03 1.00 0.17 0.29 0.05 0.06 0.37 -0.16 -0.10 -0.05 0.26 -0.29
DeviceProtection -0.01 -0.00 -0.02 0.17 0.08 0.37 0.00 0.12 0.05 0.17 1.00 0.24 0.28 0.29 0.35 -0.04 -0.14 0.16 0.39 -0.18
TechSupport 0.00 -0.01 -0.15 0.13 0.13 0.32 -0.02 0.01 -0.02 0.29 0.24 1.00 0.16 0.16 0.43 -0.11 -0.10 -0.00 0.28 -0.28
StreamingTV -0.01 -0.01 0.03 0.14 0.05 0.29 0.06 0.18 0.11 0.05 0.28 0.16 1.00 0.43 0.23 0.09 -0.11 0.33 0.39 -0.04
StreamingMovies -0.02 -0.01 0.05 0.13 0.02 0.29 0.05 0.18 0.10 0.06 0.29 0.16 0.43 1.00 0.23 0.08 -0.11 0.33 0.40 -0.04
Contract 0.01 -0.00 -0.14 0.29 0.24 0.67 0.00 0.11 0.10 0.37 0.35 0.43 0.23 0.23 1.00 -0.18 -0.23 -0.07 0.45 -0.40
PaperlessBilling -0.00 -0.01 0.16 -0.01 -0.11 0.00 0.02 0.17 -0.14 -0.16 -0.04 -0.11 0.09 0.08 -0.18 1.00 -0.06 0.35 0.16 0.19
PaymentMethod 0.01 0.02 -0.04 -0.16 -0.04 -0.37 -0.01 -0.17 0.08 -0.10 -0.14 -0.10 -0.11 -0.11 -0.23 -0.06 1.00 -0.19 -0.33 0.11
MonthlyCharges -0.00 -0.02 0.22 0.10 -0.11 0.24 0.25 0.43 -0.31 -0.05 0.16 -0.00 0.33 0.33 -0.07 0.35 -0.19 1.00 0.64 0.19
TotalCharges -0.00 -0.01 0.10 0.32 0.06 0.82 0.11 0.45 -0.17 0.26 0.39 0.28 0.39 0.40 0.45 0.16 -0.33 0.64 1.00 -0.20
Churn -0.02 -0.01 0.15 -0.15 -0.16 -0.35 0.01 0.04 -0.05 -0.29 -0.18 -0.28 -0.04 -0.04 -0.40 0.19 0.11 0.19 -0.20 1.00
customerID
gender
SeniorCitizen
Partner
Dependents
tenure
PhoneService
MultipleLines
InternetService
OnlineSecurity
DeviceProtection
TechSupport
StreamingTV
StreamingMovies
Contract
PaperlessBilling
PaymentMethod
MonthlyCharges
TotalCharges
Churn
1.0
0.8
0.6
0.4
0.2
0.0
−0.2

PROJECT PHASE 2 REPORT

Section : CSE 587 B
Revathi Gollapudi -- vgollapu@buffalo.edu
Srikar Chintha -- chintha@buffalo.edu
Sumana Madhireddy – sumanama@buffalo.edu

Forecasting Telecom Customer Churn for Proactive Retention and Business Success.

PROBLEM STATEMENT:

The telecommunications industry is dealing with a prevalent issue of customer churn, where subscribers discontinue services, leading to revenue loss and market share erosion. The problem at hand is to develop an effective predictive model for telecom customer churn and subsequently implement strategic retention measures.

Dataset Sources:

Dataset has been taken from the sample dataset from IBM sample dataset for Telecom Customer Churn.
https://www.ibm.com/docs/en/cognosanalytics/11.1.0?topic=samples-telcocustomer-churn .

The dataset has 7113 rows and 22 columns. Below are the column names which are features for further analysis and their datatypes from the above dataset. Out of which 4 column are float type, 1 columns is of integer type and remaining 17 are Object type

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7113 entries, 0 to 7112
Data columns (total 22 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   customerID        7102 non-null   object 
 1   gender            7073 non-null   object 
 2   SeniorCitizen     7102 non-null   float64
 3   Partner           7102 non-null   object 
 4   Dependents        7070 non-null   object 
 5   tenure            7102 non-null   float64
 6   PhoneService      7102 non-null   object 
 7   MultipleLines     7102 non-null   object 
 8   InternetService   7102 non-null   object 
 9   OnlineSecurity    7102 non-null   object 
 10  OnlineBackup      7102 non-null   object 
 11  DeviceProtection  7102 non-null   object 
 12  TechSupport       7102 non-null   object 
 13  StreamingTV       7102 non-null   object 
 14  StreamingMovies   7102 non-null   object 
 15  Contract          7058 non-null   object 
 16  PaperlessBilling  7056 non-null   object 
 17  PaymentMethod     7102 non-null   object 
 18  MonthlyCharges    7053 non-null   float64
 19  TotalCharges      7102 non-null   float64
 20  Churn             7102 non-null   object 
 21  Count             7113 non-null   int64  
dtypes: float64(4), int64(1), object(17)
memory usage: 1.2+ MB
```

The task we are performing is a binary classification task where We predict the Churn . So the target column here is Churn.

Have taken all other columns as input and output prediction column as Churn as shown below:

Splitting the dataset into 70 percent training and 30 percent testing.

choosing Churn column as target column for prediction and all other

columns as input columns for prediction

```
X=df.drop('Churn', axis=1).to_numpy()
Y=df[['Churn']].to_numpy()
split_ratio = 0.7
split= int(split_ratio * len(X))
X_train, X_test = X[:split], X[split:]
Y_train, Y_test = Y[:split], Y[split:]
```

Below are the methods written for calculating Accuracy, Recall, Precision and F1 Score.

```
:
# Method for calculating accuracy
def Accuracy(y_true,y_pred):
    return accuracy_score(y_true,y_pred)*100

# Method for calculating Recall
def recall(y_true,y_pred):
    return recall_score(y_true, y_pred)

# Method for calculating Precision
def precision(y_true,y_pred):
    return precision_score(y_true, y_pred)

# Method for calculating F1_Score
def F1_Score(y_true,y_pred):
    return f1_score(y_true, y_pred)
```

Below are the methods written for visualization of predicted output and actual outcomes of all the below 6 algorithms using Confusion Matrix and ROC(Receiver Operating Characteristics) curve.

```
# Method for visualization of confusion matrix
def ConfusionMatrix(y_true,y_pred):
    conf_matrix = confusion_matrix(y_true, y_pred)
    plt.figure(figsize=(8, 6))
    sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues",cbar=True)
    plt.title("Confusion Matrix Visulization")
    plt.xlabel('Predicted Values')
    plt.ylabel('True Values')
    plt.show()



# Method for ROC(Receiver Operator Characteristics) Curve
def plot_roc_curve(y_true, y_pred):
    false_pos_rate, true_pos_rate, thresholds = roc_curve(y_true, y_pred)
    roc_area_under_curve = auc(false_pos_rate, true_pos_rate)

    # Plotting the ROC curve
    plt.figure()
    plt.plot(false_pos_rate, true_pos_rate, color='blue', lw=2, label='ROC curve (area = %0.2f)' % roc_area_under_curve)
    plt.plot([0, 1], [0, 1], color='red', lw=1, linestyle='--')
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver Operating Characteristic Curve')
    plt.legend(loc="upper left")
    plt.show()
```

Algorithm 1:
K-nearest Neighbors.

Why KNN for chosen dataset?

Target Column here is Churn which has two labels 0 and 1. So, a binary classifier is required. KNN can be used as a binary classifier.

Our chosen data set is of medium-size with 7113 rows and 22 columns.

KNN works effectively with medium-sized datasets since it does not demand a lot of computer power during training.

KNN creates predictions based on a data point's local neighborhood.

When the decision border is established by the behavior of surrounding data points, this is very much useful.

Work done to Tune the model and Implementation:

Below is the implementation of KNN algorithm on chosen dataset using scikit-learn library.

1. K Nearest Neighbours Classification Algorithm

```
[7]:
#Created an instance of KNN classifier
knearest_model = KNeighborsClassifier(n_neighbors=10)

#Trained the model
knearest_model.fit(X_train, Y_train.ravel())

#Made predictions using trained model
knearest_predictions = knearest_model.predict(X_test)


#Calculated Accuracy, Recall, Precision, F1_score for the above predictions

print("Accuracy=",Accuracy(Y_test,knearest_predictions))
print("Recall =",recall(Y_test,knearest_predictions))
print("Precision =",precision(Y_test,knearest_predictions))
print("F1 Score =",F1_Score(Y_test,knearest_predictions))

#Drawn Visualizations using Confusion Matrix, ROC Curve
ConfusionMatrix(Y_test,knearest_predictions)
plot_roc_curve(Y_test,knearest_predictions)
```

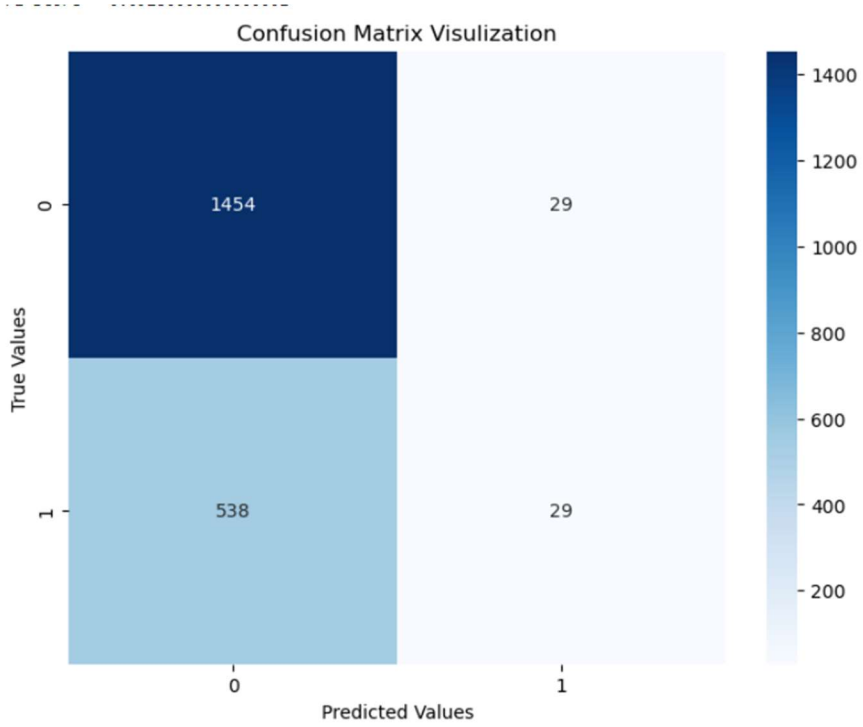
```
Accuracy= 72.34146341463415
Recall = 0.05114638447971781
Precision = 0.5
F1 Score = 0.09280000000000001
```

Here the important factor is choosing number of nearest neighbors(K) value. Most of the work done here is changing the k value and seeing the accuracy and Parameters. the k value of 3 gave the accuracy of 67 and when k value is increased to 7, the accuracy increased to 70.
For the chosen dataset value of k=10 gave highest accuracy.

Effectives of the Algorithm on chosen dataset:

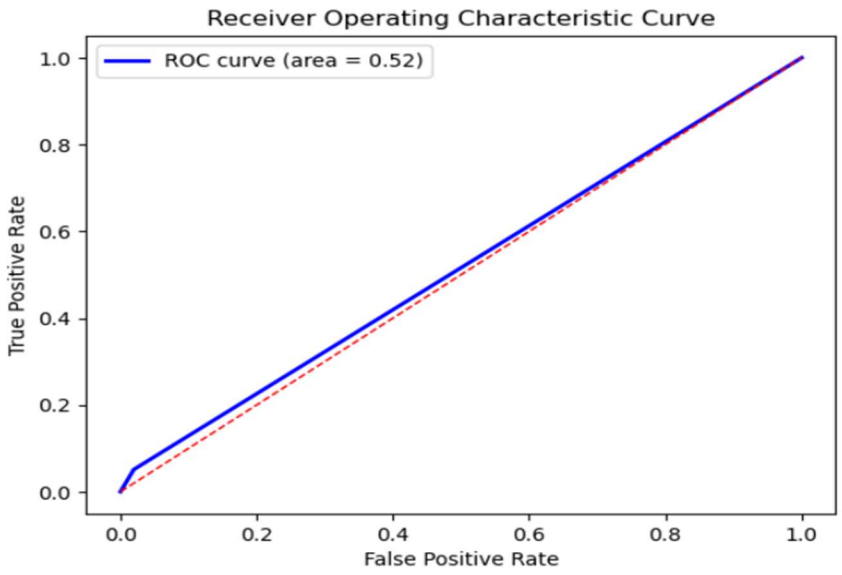
```
Accuracy= 72.34146341463415
Recall = 0.05114638447971781
Precision = 0.5
F1 Score = 0.09280000000000001
```

The model demonstrates moderate accuracy at 72.34%, indicating it correctly predicts about three-quarters of cases.
However, its recall is notably low at 5.11%, suggesting it identifies a small fraction of relevant instances.
With a precision of 50%, half of its predictions are correct.



The confusion matrix shows the model has a high number of true negatives (1454) but also a high number of false negatives (538), indicating it often incorrectly predicts the negative class. There are equal numbers of true positives and false positives (29 each), suggesting the model rarely predicts the positive class, which could indicate a bias towards predicting the negative class or an imbalanced dataset.

The ROC curve with an area of 0.52 suggests that model barely out performs random guessing in distinguishing between classes.
Its effectiveness is marginal, as indicated by the curve's proximity to the diagonal line.



Learnings from above:

These above analysis tells that while the model is fairly good at identifying the negative class, it needs improvement in correctly classifying the positive class, which is essential for a balanced performance across different classes.
KNN works well for low dimensionality and for the
Chosen data set , a k value of 10 gave the highest accuracy.

Algorithm 2:
Logistic Regression

Why Logistic Regression for chosen dataset?

Target Column here is Churn which has two labels 0 and 1. So, a binary classifier is required. Logarithmic regression can be used as a binary classifier as target column has 0's and 1's. It makes it simple to grasp how each feature affects the prediction by modeling the relation between the features and the possibility of the binary classification.

Work done to Tune the model and Implementation:

Below is the implementation of the Logarithmic Regression using scikit-learn library.

2. Logistic Regression Classification Algorithm

8]:

```
#Created an instance of logistic regression model
logisticreg_model = LogisticRegression(penalty='l2', C=1.0, solver='newton-cg')

#Trained the model
logisticreg_model.fit(X_train, Y_train.ravel())

#Made prediction using trained model
logisticreg_predictions = logisticreg_model.predict(X_test)

#Calculated Accuracy, Recall, Precision, F1_score for the above predictions
print("Accuracy =",Accuracy(Y_test,logisticreg_predictions))
print("Recall =",recall(Y_test,logisticreg_predictions))
print("Precision =",precision(Y_test,logisticreg_predictions))
print("F1 Score =",F1_Score(Y_test,knearest_predictions))

#Drawn Visualizations using Confusion Matrix, ROC Curve
ConfusionMatrix(Y_test,logisticreg_predictions)
plot_roc_curve(Y_test,logisticreg_predictions)
```

```
Accuracy = 80.09756097560977
Recall = 0.5361552028218695
Precision = 0.6770601336302895
F1 Score = 0.09280000000000001
```

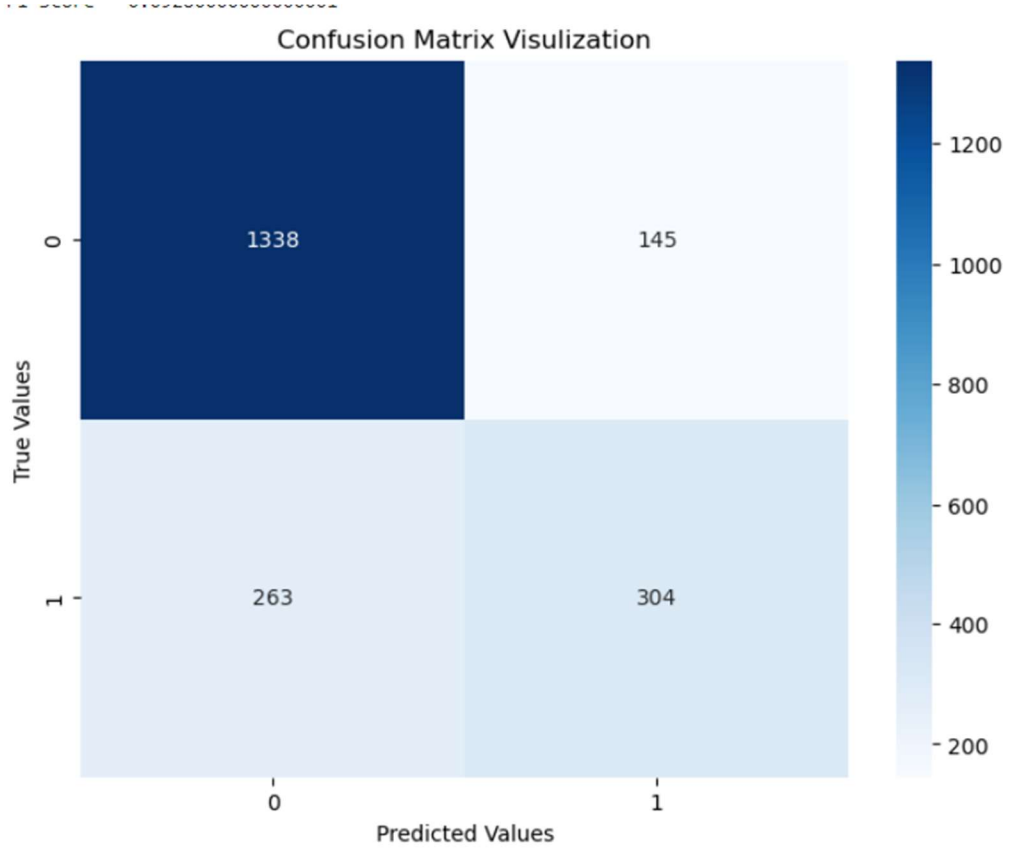
The penalty parameter indicates the regularization used for the model.
It adds a penalty term into the loss function to prevent the model from giving any given characteristic an abnormally high weight.
The penalty we are using is l2 regularization.
We tried different solver' to choose the one with best accuracy.
Here, the 'liblinear' solver gave the accuracy of 79 and the accuracy given by 'newton-cg' is 81, when the solver 'lbfgs' is used, the accuracy obtained in 78.

Effectives of the Algorithm on chosen dataset:

```
Accuracy = 80.09756097560977
Recall = 0.5361552028218695
Precision = 0.6770601336302895
F1 Score = 0.09280000000000001
```

The model's accuracy stands at about 80.1%, indicating a decent overall rate of correct predictions, while the recall is low at about 53.6%, suggesting it fails to identify a good portion of actual positives. Precision is fair at about 67.7%. the F1 score is very low at about 0.093, indicating a less balance between precision and recall.

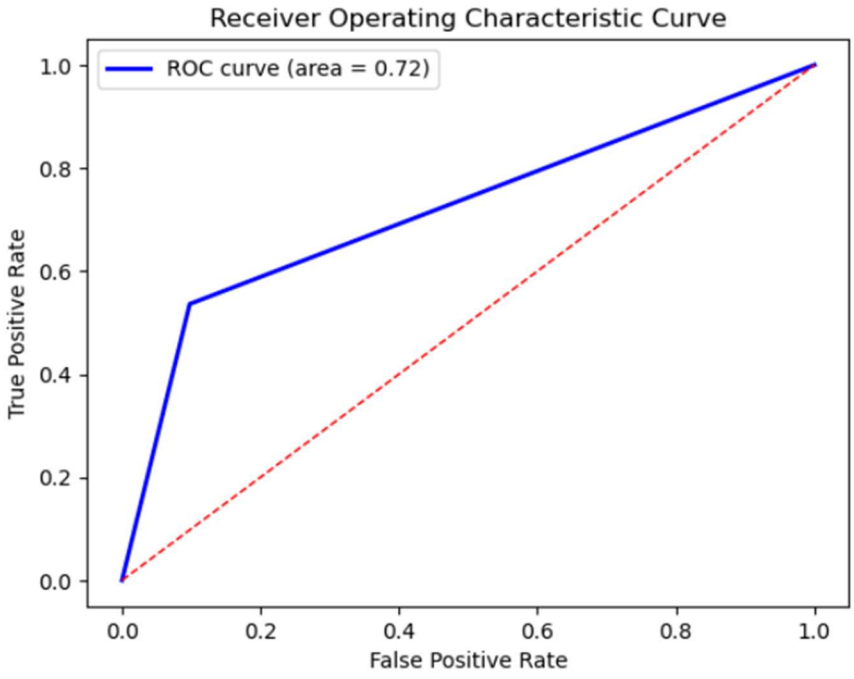
The confusion matrix visualizes the performance of a classification model.



The above matrix shows:

True negatives (TN): 1338 instances correctly predicted as class 0.
False positives (FP): 145 instances incorrectly predicted as class 1.
False negatives (FN): 263 instances incorrectly predicted as class 0.
True positives (TP): 304 instances correctly predicted as class 1.
The model seems to have a higher accuracy for predicting class 0 than class 1, considering the number of true negatives and positives.

An ROC area of 0.72 in logarithmic regression suggests that the model has moderate discrimination ability in distinguishing between the two classes, with a 72% probability of correctly classifying samples.



Learnings from above:

One of the most important factors in determining how the optimization problem of selecting the best-fitting logistic regression model is addressed is the solver selection. Logistic regression does not require a linear connection between features and the output and since the features are mostly independent in the dataset used, this will give a good accuracy. newton-cg solver with l2 regularization gave the highest accuracy.

Algorithm 3:
Support Vector Machine Classifier.

Why for chosen dataset?

Target Column here is Churn which has two labels 0 and 1. So, a binary classifier is required. SVM Classifier can be used as a binary classifier as target column has 0's and 1's. SVM is appropriate for datasets with a lot of features since it can handle high-dimensional feature spaces well and the dataset used here has 20 features(excluded the target and one removed column) and SVM will be able to handle many features well.

Work done to Tune the model and Implementation:

Below is the implementation of SVM classifier using scikit-learn.

3.Support Vector Machine Classification Algorithm

159]:

```
#created a svm classifier with linear kernel
sv_machine_model = SVC(kernel='linear', C=0.01)

#trained the model
sv_machine_model.fit(X_train, Y_train.ravel())

#Made predictions using the trained model
sv_machine_predictions = sv_machine_model.predict(X_test)


#Calculated Accuracy, Recall, Precision, F1_score for the above predictions
print("Accuracy =",Accuracy(Y_test,sv_machine_predictions))
print("Recall =",recall(Y_test,sv_machine_predictions))
print("Precision =",precision(Y_test,sv_machine_predictions))
print("F1 Score =",F1_Score(Y_test,knearest_predictions))

#Drawn Visualizations using Confusion Matrix, ROC Curve
ConfusionMatrix(Y_test,sv_machine_predictions)
plot_roc_curve(Y_test,sv_machine_predictions)
```

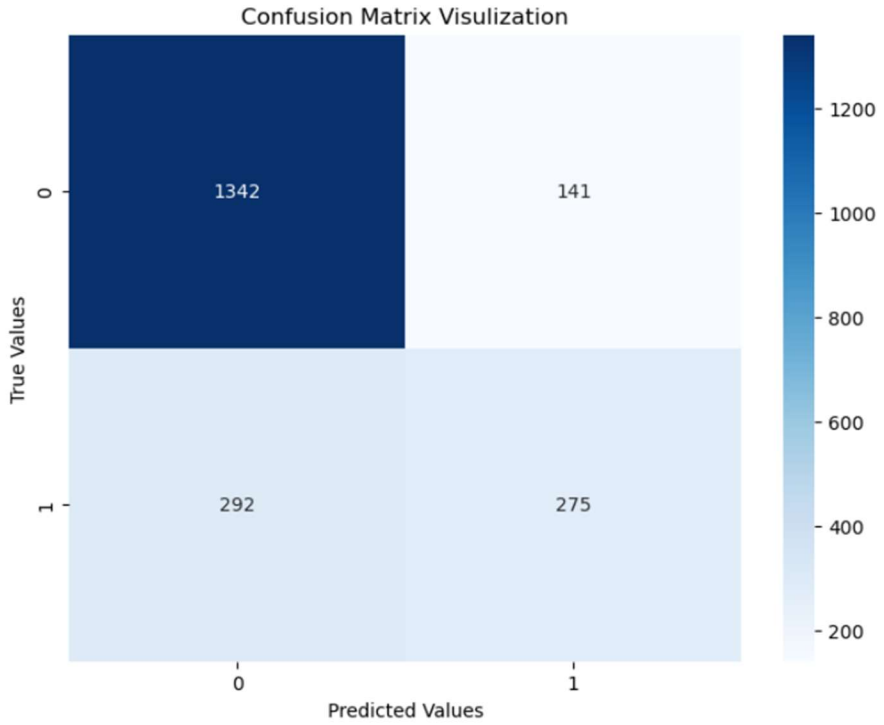
```
Accuracy = 78.8780487804878
Recall = 0.48500881834215165
Precision = 0.6610576923076923
F1 Score = 0.09280000000000001
```

In this model, we will try different kernel and different values for hyperparameter ‘C’ to decide on the best value for both for highest accuracy.
Data points are mapped from the two-dimensional feature space to a high-dimension feature space, where they could be linearly separable, using the kernel function.
For instance, we tried different kernels like ‘sigmoid’ which gave the accuracy of 61 and ‘poly’ which increased the accuracy to 71 and then by using the liner kernel the accuracy increased to 73.
It is a parameter that manages the balance between reducing classification mistakes on the training data and optimizing the border line.
When the value of ‘C’ is set to larger value it prioritizes on maximizing the margin than precisely fitting the training set which prevents it from overfitting.
Here, we will try with different ‘C’ values to choose one with highest accuracy.
When the ‘C’ value is set to 0.2 the accuracy is 73.7 and when ‘C’ is increased to 1, the accuracy is 73.4 and when ‘C’ value changed to 0.01, then accuracy increased to 78.8.

Effectives of the Algorithm on chosen dataset:

```
Accuracy = 78.8780487804878
Recall = 0.48500881834215165
Precision = 0.6610576923076923
F1 Score = 0.09280000000000001
```

The model has an accuracy of roughly 78.9%, reflecting an adequate level of correct predictions, while its recall is lower at 48.5%, showing it often fails to identify true positives. The precision is relatively high at 66.1%, but there is low F1 score of 0.0928.

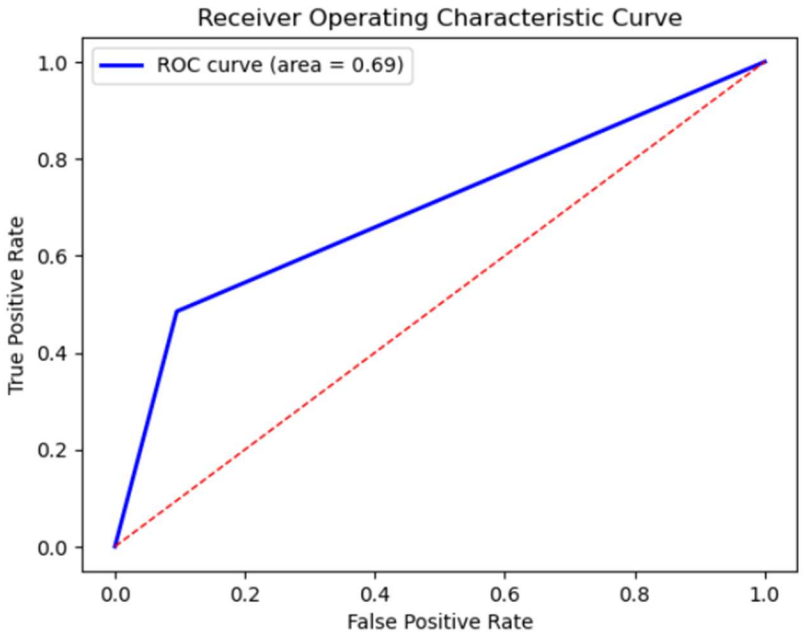


The confusion matrix for the classification model shows below:

- True Negatives (TN): 1342 instances were correctly predicted as class 0.
- False Positives (FP): 141 instances were incorrectly labeled as class 1.
- False Negatives (FN): 292 instances were incorrectly labeled as class 0.
- True Positives (TP): 275 instances were correctly predicted as class 1.

This model shows a slightly better performance in predicting true negatives compared to true positives, suggesting possible class imbalance handling or predictive performance issues.

An ROC AUC of 0.69 for the SVM classifier denotes that the model can correctly differentiate between the classes with fair accuracy compared to random chance but not by a wide margin. Comparatively, it suggests that the model's predictive power is above average but not excellent.



Learnings from above:

SVM is less prone to overfitting. By adjusting hyperparameters like the regularization parameter (C) and the kernel selection, we can increase the accuracy of predictions and we can also capture different kinds of data correlations by using different kernel functions. SVM with linear kernel and hyperparameter C=0.01 gave the highest accuracy.

Algorithm 4:
Naïve Bayes

Why Naïve Bayes for chosen dataset?

Naive Bayes is an easy and quick option for binary classification tasks since it is a simple and computationally efficient technique.
In the given features, Naive Bayes assumes that features are conditionally independent. And since our most features lack interdependency, this model works accurately and Multinomial Naive Bayes model, in particular, works well with discrete features.
The feature dimensionality has less of an impact on Naive Bayes than some of the other algorithms. Even with high-dimensional features, it can function rather effectively, and this suits our dataset well because the dataset has 20 features.

Work done to Tune the model and Implementation:

4. Naive Baye's Classification Algorithm

[160]:
```
# created an instance of Naive Baye's classifier using Multinomial NB
nb_classifier = MultinomialNB(alpha=1.0)

# Fit the classifier on the training data
nb_classifier.fit(X_train, Y_train.ravel())

# Make predictions on the test data
y_pred_nb = nb_classifier.predict(X_test)


#Calculated Accuracy, Recall, Precision, F1_score for the above predictions
print("Accuracy =",Accuracy(Y_test,y_pred_nb))
print("Recall =",recall(Y_test,y_pred_nb))
print("Precision =",precision_score(Y_test, y_pred_nb, zero_division='warn'))
print("F1 Score =",F1_Score(Y_test,knearest_predictions))

#Drawn Visualizations using Confusion Matrix, ROC Curve
ConfusionMatrix(Y_test,y_pred_nb)
plot_roc_curve(Y_test,y_pred_nb)
```

```
Accuracy = 74.29268292682927
Recall = 0.6243386243386243
Precision = 0.5299401197604791
F1 Score = 0.09280000000000001
```

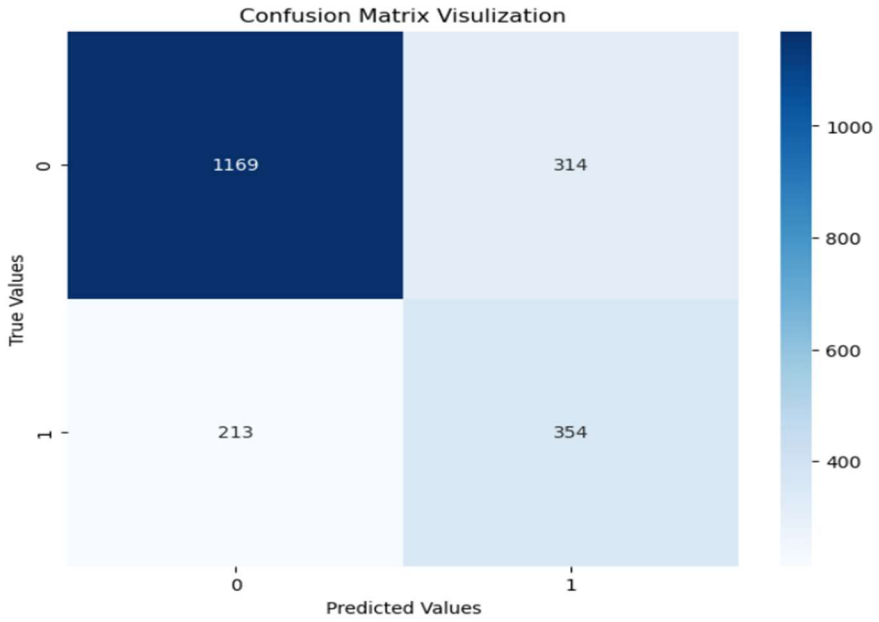
Above is the implementation of Naïve Bayes classifier using scikit-learn.

A hyperparameter that represents the Laplace smoothing is the 'alpha' parameter. A method for dealing with zero probabilities in a Naive Bayes model is Laplace smoothing.
You may control the amount of smoothing that is applied to the probabilities by adjusting the alpha parameter. In other words, you smooth the probabilities to make sure that no feature has a chance of zero by adding alpha value to the count of each feature for each class.
We will tune this parameter by trying different alpha values and the value of 1.0 gave the highest accuracy of 74.

Effectives of the Algorithm on chosen dataset:

```
Accuracy = 74.29268292682927
Recall = 0.6243386243386243
Precision = 0.5299401197604791
F1 Score = 0.09280000000000001
```

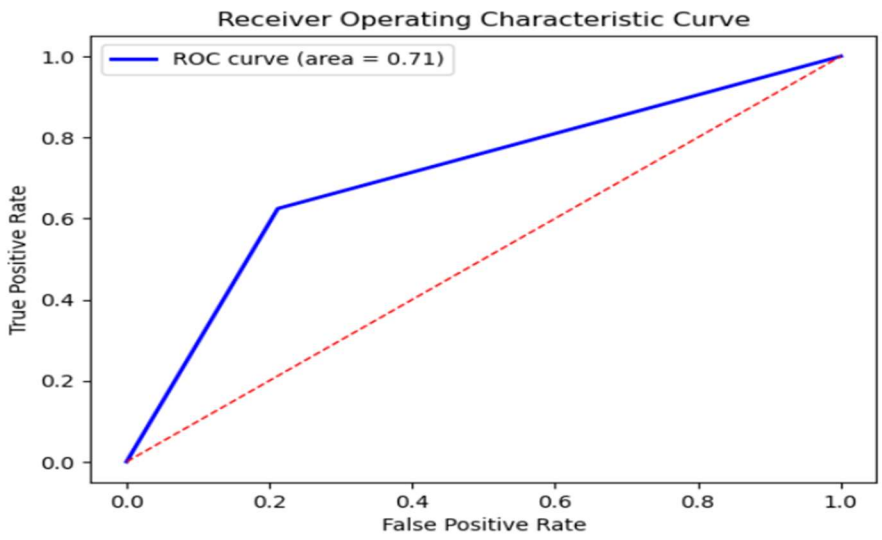
The model's accuracy is 74.3%, showing it correctly predicts three-quarters of outcomes. Its recall of 62.4% indicates a reasonable ability to detect true positives, but a precision of 52.9% suggests it also mislabels a significant number of negatives as positives. The F1 score should be recalculated for accuracy.



The above confusion matrix shows below:

The matrix shows a total of 1169 true negatives, indicating accurate predictions of the negative class. There are 354 true positives, showing fewer, but still substantial, correct predictions of the positive class.

The model has made a considerable number of false predictions with 213 false negatives and 314 false positives, suggesting challenges in accurately classifying positive instances and a tendency to incorrectly classify negative instances as positive.



An ROC curve area, or AUC, of 0.71 for a Naive Bayes classifier suggests that the model has a good ability to distinguish between the positive and negative classes. It is better than a random guess (which would have an AUC of 0.5) but not excellent.
This level of AUC indicates that there is a 71% chance that the classifier will rank a randomly chosen positive instance higher than a randomly chosen negative one.

Learnings from above:

Naive Bayes is an easy and quick option for binary classification tasks since it is a simple and computationally efficient technique.
And since our most features lack interdependency, this model works accurately and Multinomial Naive Bayes model, in particular, works well with discrete features.
The feature dimensionality has less of an impact on Naive Bayes than some of the other algorithms. Even with high-dimensional features, it can function rather effectively, and this suits our dataset well because the dataset has 20 features.
Alpha value of 1.0 gave the highest accuracy.

Algorithm 5:
Random Forest

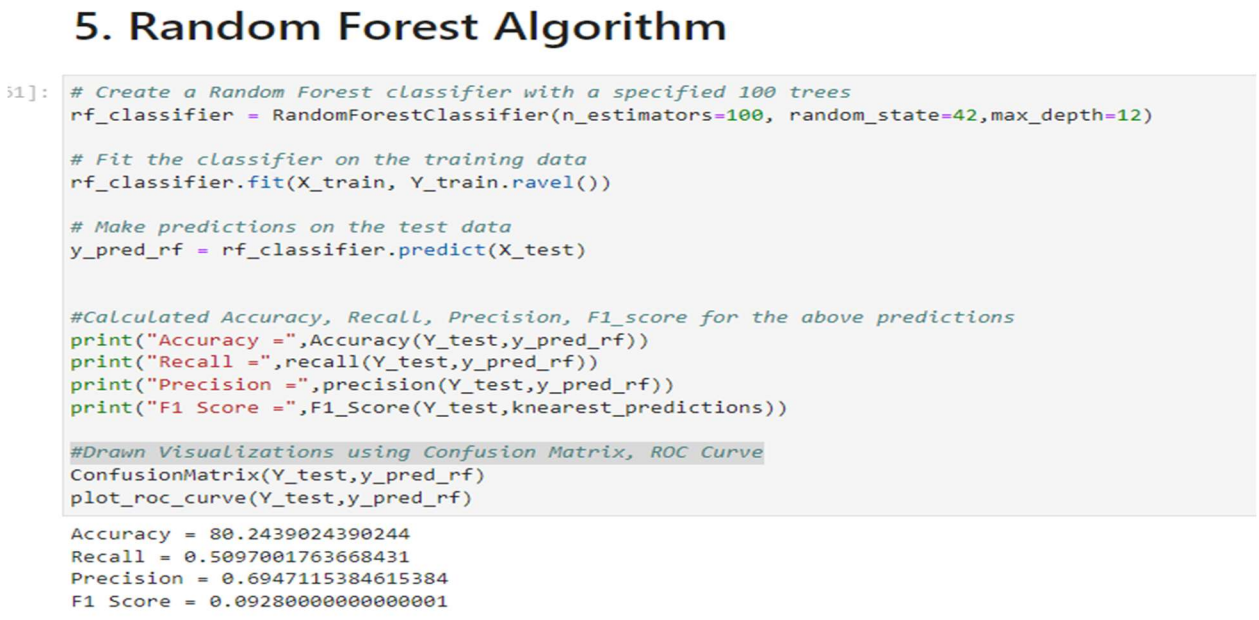
Random Forest is also well known for its great predicting accuracy. It is also an ensemble learning technique that yields predictions by combining many decision trees. Several trees work together to increase generalization performance and minimize overfitting. Random Forest is also an ensemble model which combines the predictions of base models which in case of random forests are decision trees.

Why Random Forest for chosen dataset?

Random Forest lowers the chance of overfitting by combining predictions from several decision trees with above medium size datset of 7113 rows and 22 columns Each tree's depth and complexity are adjustable, with this we can customize our model to get the best accuracy by testing on different values.

Work done to Tune the model and Implementation:

Below is the implementation of Random Forest classifier
Using scikit-learn



5. Random Forest Algorithm

```
51]: # Create a Random Forest classifier with a specified 100 trees
rf_classifier = RandomForestClassifier(n_estimators=100, random_state=42,max_depth=12)

# Fit the classifier on the training data
rf_classifier.fit(X_train, Y_train.ravel())

# Make predictions on the test data
y_pred_rf = rf_classifier.predict(X_test)


#Calculated Accuracy, Recall, Precision, F1_score for the above predictions
print("Accuracy =",Accuracy(Y_test,y_pred_rf))
print("Recall =",recall(Y_test,y_pred_rf))
print("Precision =",precision(Y_test,y_pred_rf))
print("F1 Score =",F1_Score(Y_test,knearest_predictions))

#Drawn Visualizations using Confusion Matrix, ROC Curve
ConfusionMatrix(Y_test,y_pred_rf)
plot_roc_curve(Y_test,y_pred_rf)
```

```
Accuracy = 80.2439024390244
Recall = 0.5097001763668431
Precision = 0.6947115384615384
F1 Score = 0.09280000000000001
```

In this model, to make it efficient, we are tuning two hyperparameters namely, ‘n_estimators’ and ‘max_depth’.
The number of decision trees that are separately developed and then merged to provide predictions in a Random Forest is determined by the value of ‘n _estimators’.

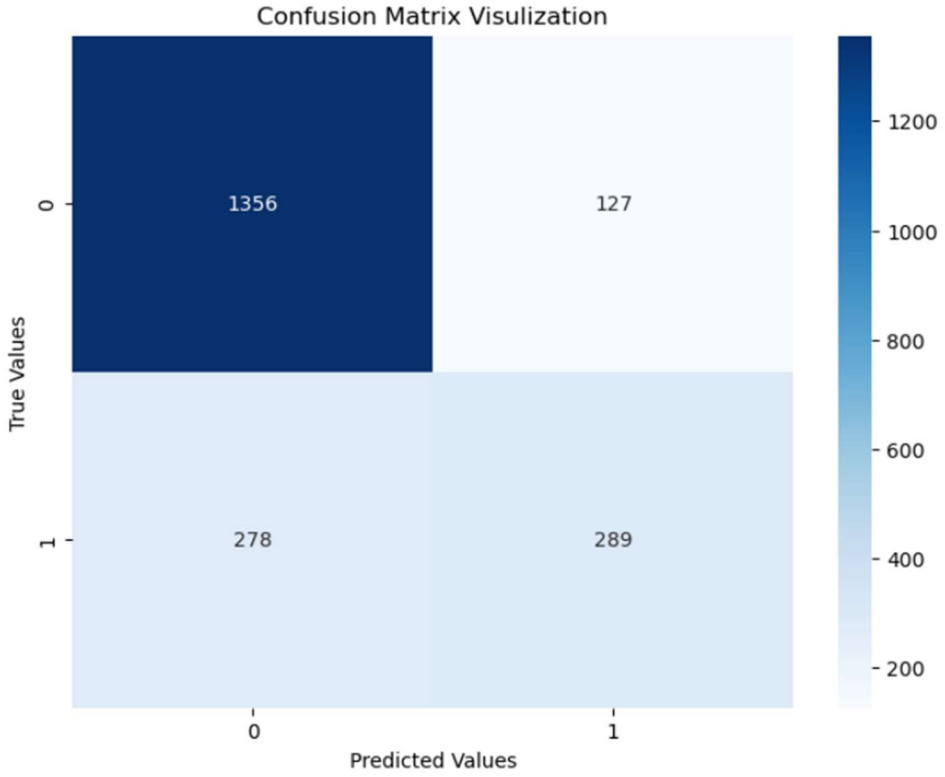
A hyperparameter called max_depth in the Random Forest regulates each decision tree's maximum depth. While each decision tree in a Random Forest can grow to a different depth, the ensemble's maximum depth for each tree is limited by max_depth.
Here, we tried for different ‘max_depth’ while training the model and choose the one with highest accuracy.
Here, the accuracy obtained with ‘max_depth’ value 3 is 79 and with the value being changed to 20 the accuracy is 79.4 and with 12 the accuracy increases to 80.2.

Effectives of the Algorithm on chosen dataset:

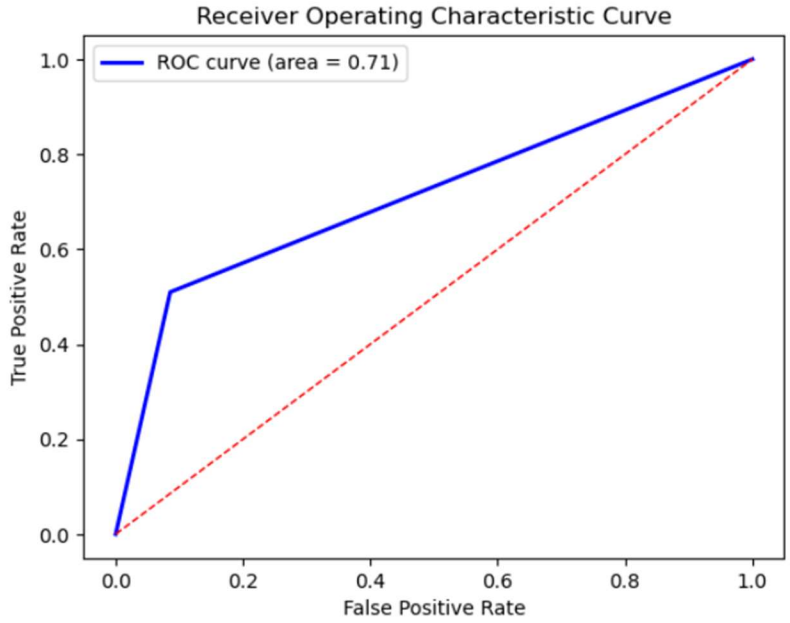
```
Accuracy = 80.2439024390244
Recall = 0.5097001763668431
Precision = 0.6947115384615384
F1 Score = 0.09280000000000001
```

The model exhibits an accuracy of about 80.2%, indicating a relatively high rate of overall correct classifications. However, the recall is just over 50.9%, suggesting it correctly identifies only half of the positive cases, while the precision is higher at 69.47%, indicating a better rate of predicting true positives out of all positive predictions.

The confusion matrix below indicates the model's strong ability to correctly predict the negative class with 1356 true negatives, but it also shows a significant number of false negatives, with 278 instances where the positive class was incorrectly predicted as negative. Conversely, the model predicted 289 true positives, indicating moderate effectiveness in identifying the positive class, and made 127 false positives, where the negative class was incorrectly predicted as positive.



An AUC of 0.71 for a Random Forest classifier tells that it can differentiate between positive and negative classes with good accuracy, yet there's potential for enhancing its predictive performance.



Learnings from above:

Unbalanced datasets may be handled by Random Forest by modifying class weights. By doing this, the model is kept from becoming skewed in favor of the majority class.

Max depth value of 12 gave the highest accuracy of 79.4%.

Algorithm 6:
Gradient Boosting

Gradient Boosting is an ensemble learning technique that builds a strong, reliable model by combining several poor learners.
Gradient Boosting is well-known for its accurate predictions, and it frequently achieves high accuracy
By using an ensemble technique and built-in anti-overfitting measures is reduced and the model's ability to generalize effectively to new data is improved.

Why Gradient Boosting for chosen dataset?

In the chosen dataset with 7113 rows and 22 columns, the interpretability is less. Gradient boosting can manage complicated relations in the data since there is some dependency in the features like ‘internet service’ and ‘online security’ and ‘online backup’ columns in chosen dataset.
Gradient boosting is well at handling such complex interdependence between features.

Work done to Tune the model and Implementation:

6. Gradient boosting Classification Algorithm

```
[62]: # Create a Gradient Boosting classifier with a specified number of estimators (n_estimators)
gb_classifier = GradientBoostingClassifier(n_estimators=100, random_state=42)

# Fit the classifier on the training data
gb_classifier.fit(X_train, Y_train.ravel())

# Make predictions on the test data
y_pred_gb = gb_classifier.predict(X_test)

#Calculated Accuracy, Recall, Precision, F1_score for the above predictions
print("Accuracy =",Accuracy(Y_test,y_pred_gb))
print("Recall =",recall(Y_test,y_pred_gb))
print("Precision =",precision(Y_test,y_pred_gb))
print("F1 Score =",F1_Score(Y_test,knearest_predictions))

#Drawn Visualizations using Confusion Matrix, ROC Curve
ConfusionMatrix(Y_test,y_pred_gb)
plot_roc_curve(Y_test,y_pred_gb)

Accuracy = 79.5609756097561
Recall = 0.49559082892416223
Precision = 0.678743961352657
F1 Score = 0.09280000000000001
```

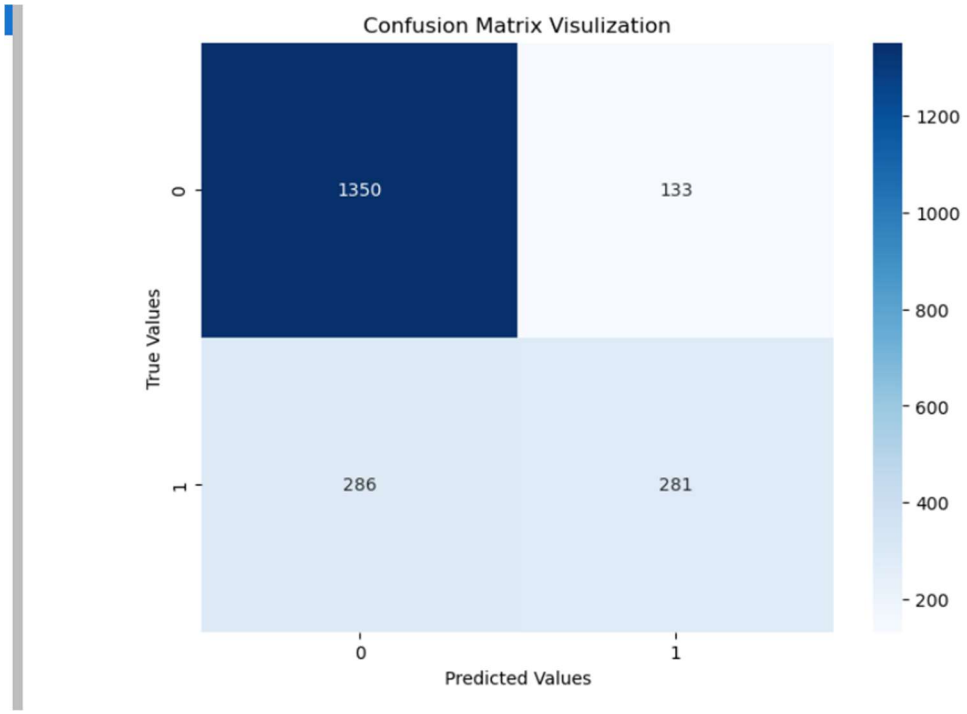
Above is the implementation of Gradient boostin classifier using scikit-learn. Since gradient boosting is an ensemble model which combines the prediction of all the base models, we can increase the accuracy and make our model work more efficiently by tuning the ‘n_estimators’ hyperparameter.

The number of iterations is specified by 'n_estimators'. A new decision tree gets added to the ensemble at each iteration in order to fix the mistakes caused by the earlier models. customizing the number of rounds may improve the performance of the model.
Tried different values for 'n_estimators' to confirm on one value where the accuracy obtained is highest among all other values for 'n_estimators'.
Here, when the 'n_estimators' is 10, the accuracy is 77.4, and when the value is changed to 500 , 78.9 and similarly for 100 the accuracy increased to 79.5.

Effectives of the Algorithm on chosen dataset:

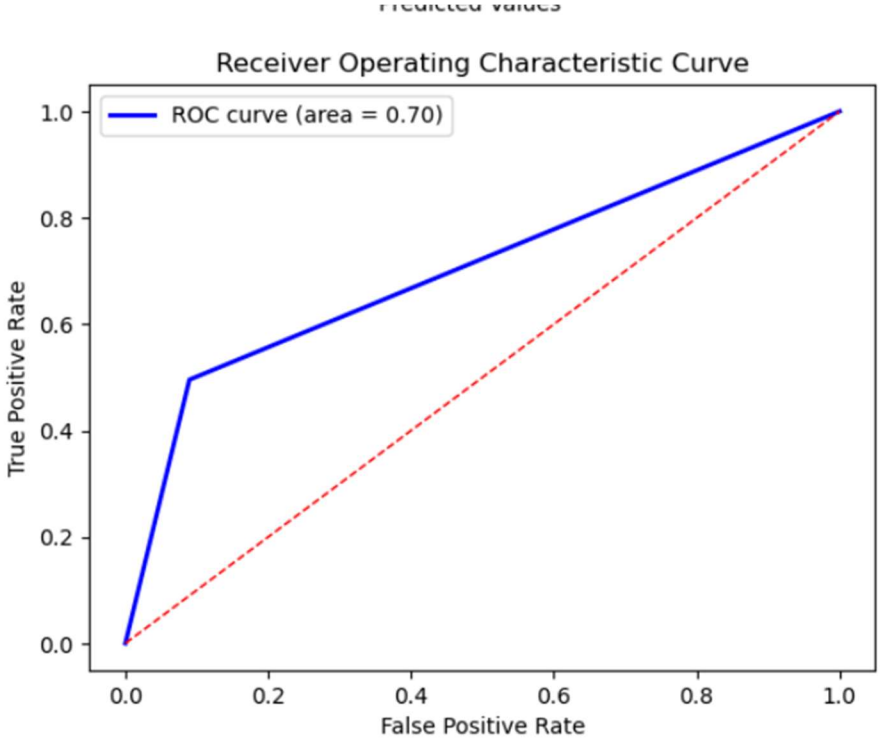
```
Accuracy = 79.5609756097561
Recall = 0.49559082892416223
Precision = 0.678743961352657
F1 Score = 0.09280000000000001
```

The model achieves an accuracy of roughly 79.6%, effectively predicting correct outcomes most of the time, but exhibits a lower recall of about 49.6%, indicating it misses many true positives. Precision is moderately high at approximately 67.9%, but the reported F1 score is less.



The confusion matrix for the Gradient Boosting model shows a high number of true negatives (1350), suggesting effective prediction of the negative class. However, there are 286 false negatives, indicating the model often misclassifies the positive class as negative. True positives are lower at 281, reflecting moderate success in identifying positive cases.

The model has a small number of false positives (133), indicating it is good in predicting the positive class.



The above area of 0.71 for a Gradient Boosting model indicates a satisfactory level of differentiation ability, where the model is capable of distinguishing between the positive and negative classes with a fair degree of accuracy.
It reflects that the model is significantly better than random guessing.

Learnings from the above:

we can increase the accuracy and make our model work more efficiently by tuning the ‘n_estimators’ hyperparameter.
By using the ensemble technique, model’s accuracy is improved and overfitting is reduced. n_estimators value of 100 gave the highest accuracy of 79.5.

References:

1.KNN
2.Logarithmic Regression
3. SVM Classifier
4.Naive Baye's classifier
The above 4 classifiers are taught in class and knowledge about them is obtained from class recordings and slides, implementation reference of above 4 is https://scikit-learn.org/ .

5. Random Forest
6.Gradient Boosting

The above 2 classifers are not taught in class and knowledge about 5th one is obtained from the text book "Data Science from Scratch First Principles with Python By Joel Grus Apr 2015 Oreilly", knowledge about 6th one is from https://www.geeksforgeeks.org/ml-gradient-boosting/
and implementation of both 5 and 6 from https://scikit-learn.org/ and https://scikit-learn.org/stable/modules/classes.html#module-sklearn.ensemble respectively.

PROJECT PHASE 3 REPORT

Section: CSE 587 B

Revathi Gollapudi -- vgollapu@buffalo.edu

Srikar Chintha -- chintha@buffalo.edu

Sumana Madhireddy – sumanama@buffalo.edu

Forecasting Telecom Customer Churn for Proactive Retention and Business Success.

Problem Statement

The telecommunications industry is dealing with a prevalent issue of customer churn, where subscribers discontinue services, leading to revenue loss and market share erosion. The problem at hand is to develop an effective predictive model for telecom customer churn and subsequently implement strategic retention measures.

Abstract

Customer churn prediction is a method that forecasts when consumers are likely to leave a service or subscription for a variety of reasons in industries such as finance, telecommunications, and so on. Customer retention has become increasingly important for businesses due to their constantly changing needs. In order to solve this problem, we developed a Customer Churn Prediction model with a Random Forest classifier and integrated it into "Customer Churn Predictor" app that we developed using Flask. Based on a few variables, the predictor app forecasts whether a consumer in the telecom sector will cancel their service or not.

Dataset

Dataset has been taken from the sample dataset from IBM sample dataset for Telecom Customer Churn.

https://www.ibm.com/docs/en/cognos-analytics/11.1.0?topic=samples-telcocustomer-churn

The dataset has 7113 rows and 22 columns.

Below are the column names which are features for further analysis and their datatypes from the above dataset. Out of which 4 columns are float type, 1 column is of integer type and remaining 17 are Object type.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7113 entries, 0 to 7112
Data columns (total 22 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   customerID        7102 non-null   object
 1   gender            7073 non-null   object
 2   SeniorCitizen     7102 non-null   float64
 3   Partner           7102 non-null   object
 4   Dependents        7070 non-null   object
 5   tenure            7102 non-null   float64
 6   PhoneService      7102 non-null   object
 7   MultipleLines     7102 non-null   object
 8   InternetService   7102 non-null   object
 9   OnlineSecurity    7102 non-null   object
 10  OnlineBackup      7102 non-null   object
 11  DeviceProtection  7102 non-null   object
 12  TechSupport       7102 non-null   object
 13  StreamingTV       7102 non-null   object
 14  StreamingMovies   7102 non-null   object
 15  Contract          7058 non-null   object
 16  PaperlessBilling  7056 non-null   object
 17  PaymentMethod     7102 non-null   object
 18  MonthlyCharges    7053 non-null   float64
 19  TotalCharges      7102 non-null   float64
 20  Churn             7102 non-null   object
 21  Count             7113 non-null   int64
dtypes: float64(4), int64(1), object(17)
memory usage: 1.2+ MB
```

Model

In our project we have implemented K-nearest Neighbors, Logistic Regression, Support Vector Machine Classifier, Naïve Bayes, Random Forest, and Gradient Boosting. Among all of them we choose Random Forest classifier.

Random Forest had the highest accuracy (79.22%), followed by Logistic Regression (78.05%) and Gradient Boosting (78.73%). Additionally, Random Forest had the strongest recall (49.03%), demonstrating how well it captures real positive experiences. Since SVM and Naive Bayes had zero recall, it is possible that they were unable to detect positive examples. The best precision

was achieved using logistic regression (66.03%), which was closely followed by gradient boosting (66.67%) and random forest (66.99%).

Random Forest is also well known for its great predicting accuracy. It is also an ensemble learning technique that yields predictions by combining many decision trees. Several trees work together to increase generalization performance and minimize overfitting.

Random Forest is also an ensemble model which combines the predictions of base models which in case of random forests are decision trees.

Random Forest lowers the chance of overfitting by combining predictions from several decision trees with above medium size dataset of 7113 rows and 22 columns. Each tree's depth and complexity are adjustable, with this we can customize our model to get the best accuracy by testing on different values.

Below is the implementation of Random Forest classifier Using scikit-learn

5. Random Forest Algorithm

```
import pickle
# Create a Random Forest classifier with a specified 100 trees (n_estimators)
rf_classifier = RandomForestClassifier(n_estimators=100, random_state=42,max_depth=12)

# Fit the classifier on the training data
rf_classifier.fit(X_train, Y_train.ravel())

# Make predictions on the test data
y_pred_rf = rf_classifier.predict(X_test)


#Calculated Accuracy, Recall, Precision, F1_score for the above predictions
print("Accuracy =",Accuracy(Y_test,y_pred_rf))
print("Recall =",recall(Y_test,y_pred_rf))
print("Precision =",precision(Y_test,y_pred_rf))
print("F1 Score =",F1_Score(Y_test,knearest_predictions))

#Drawn Visualizations using Confusion Matrix, ROC Curve
ConfusionMatrix(Y_test,y_pred_rf)
plot_roc_curve(Y_test,y_pred_rf)
```

Effectives of the Algorithm on chosen dataset:

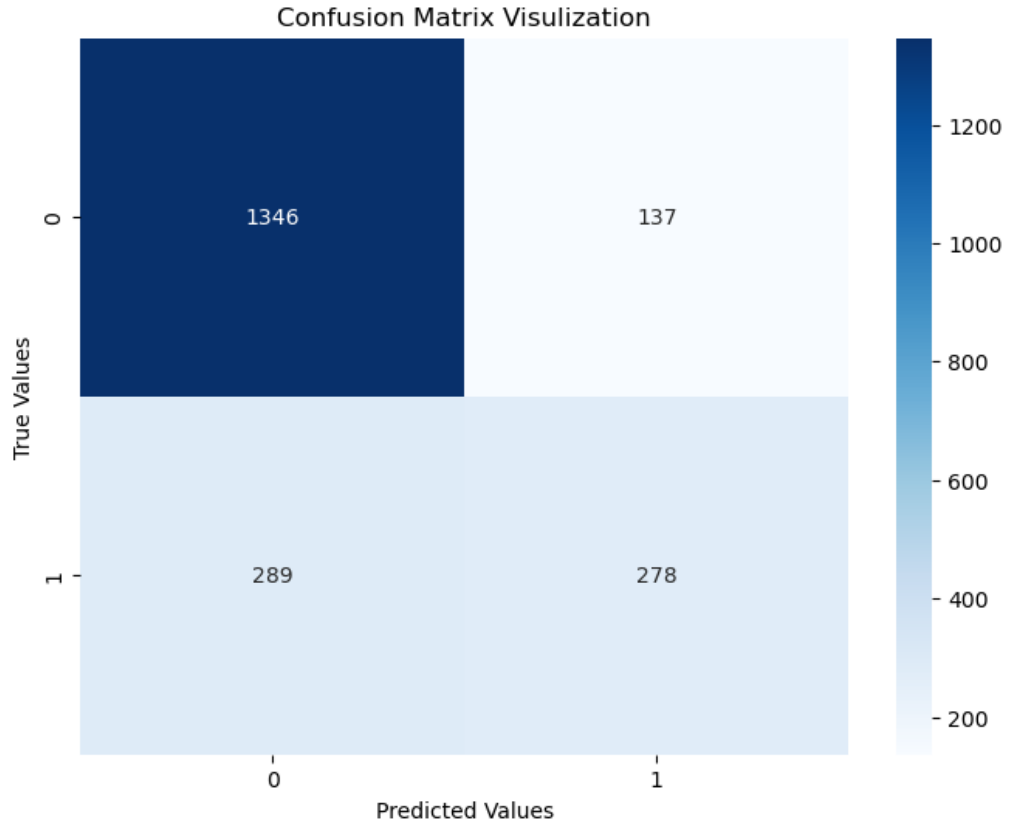
The model exhibits an accuracy of about 79.2%, indicating a relatively high rate of overall correct classifications. However, the recall is just over 49.02%, suggesting it correctly identifies only half of the positive cases, while the precision is higher at 67%, indicating a better rate of predicting true positives out of all positive predictions.

```
Accuracy = 79.21951219512195
Recall = 0.490299823633156 94
Precision = 0.6698795180722892
F1 Score = 0.4884488448844885
```

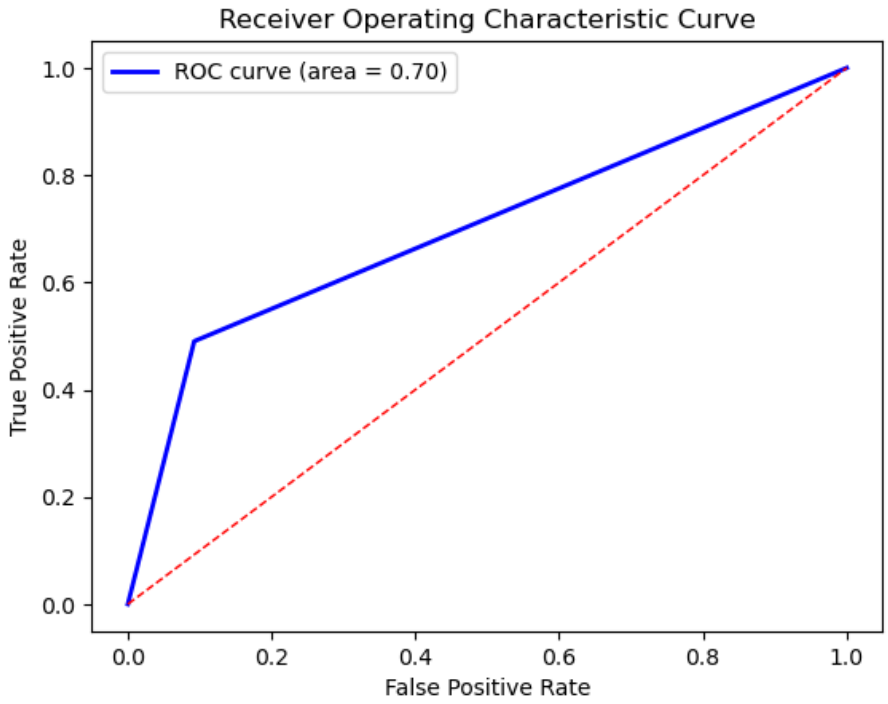
We have saved our model as “best_model_rf” as a pickle file which we have used in our flask program.

```
pickle.dump(rf_classifier, open( "best_model_rf.p", 'wb' ))
```

The confusion matrix below indicates the model's strong ability to correctly predict the negative class with 1346 true negatives, but it also shows a significant number of false negatives, with 289 instances where the positive class was incorrectly predicted as negative. Conversely, the model predicted 278 true positives, indicating moderate effectiveness in identifying the positive class, and made 137 false positives, where the negative class was incorrectly predicted as positive.



An AUC of 0.70 for a Random Forest classifier tells that it can differentiate between positive and negative classes with good accuracy, yet there's potential for enhancing its predictive performance.



Deployment:

In our HTML script, we have used a form which takes input and predicts if customer is going to stay or leave as churn Yes or No. We also used a “predict churn” button to output the prediction.

Input parameters the form includes are Gender, Monthly Charges, Payment Method, Paperless Billing, Senior Citizen, Phone Service, Multiple Lines, Total Charges.

The action property on the form instructs the POST method to be used to send the data to the '/predict' site. The projected churn value is shown in the 'predicted_value' div after the user's input has been analyzed by the server. The code also uses Flask's 'url_for' function to insert an image for a particular photo. Input fields, dropdown menus, and labels are used in the form's layout to capture key information needed to anticipate customer churn.

When the user clicks the "View Visualization Analysis" button, the JavaScript code within the script element manages the dynamic loading of images, enriching the user experience with extra visual insights connected to the dataset.

The script is as shown below. HTML script is placed in templates folder and CSS script is placed in static folder respectively.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Customer Churn Predictor</title>
    <link rel="stylesheet" href="{{ url_for('static', filename = 'css/styles.css')}}">
</head>
```

```
<body>
  <div class = 'main_line'>
    <div class="header">
      <img src="{{ url_for('static',filename = 'css/churn_photo.PNG')}}" alt="Churn Photo" class="churn-photo">
      <h1>Enter below details to predict Customer Churn:</h1>
    </div>
    <form  action="/predict" method="post">
        <label for="gender">Gender:</label>
        <select id="gender" name="gender" required>
            <option value="" disabled selected>Select Gender</option>
            <option value="male">Male</option>
            <option value="female">Female</option>
        </select><br>
        <label for="MonthlyCharges">Monthly Charges:</label>
        <input type="text" id="MonthlyCharges" name="MonthlyCharges" class="centered-input "required><br>
        <label for="PaymentMethod">Payment Method:</label>
        <select id="PaymentMethod" name="PaymentMethod" required>
            <option value="" disabled selected>Select Payment Method</option>
            <option value="Bank transfer">Bank Transfer</option>
            <option value="Credit card">Credit Card</option>
            <option value="Electronic check">Electronic Check</option>
            <option value="Mailed check">Mailed Check</option>
        </select><br>
        <label for="PaperlessBilling">Paperless Billing:</label>
        <select id="PaperlessBilling" name="PaperlessBilling" required>
            <option value="" disabled selected>Select Paperless Billing Option</option>
            <option value="yes">Yes</option>
            <option value="no">No</option>
        </select><br>
        <label for="SeniorCitizen">Senior Citizen:</label>
        <select id="SeniorCitizen" name="SeniorCitizen" required>
            <option value="" disabled selected>Select Senior Citizen Option</option>
            <option value="yes">Yes</option>
            <option value="no">No</option>
        </select><br>
```

```
        <label for="PhoneService">Phone Service:</label>
        <select id="PhoneService" name="PhoneService" required>
            <option value="" disabled selected>Select Phone Service Option</option>
            <option value="yes">Yes</option>
            <option value="no">No</option>
        </select><br>
        <label for="MultipleLines">Multiple Lines:</label>
        <select id="MultipleLines" name="MultipleLines" required>
            <option value="" disabled selected>Select Multiple Lines Option</option>
            <option value="No">No</option>
            <option value="Yes">Yes</option>
        </select><br>
        <label for="TotalCharges">Total Charges:</label>
        <input type="text" id="TotalCharges" name="TotalCharges" class="centered-input" required><br><br>
        <button type="submit">Predict Churn</button>
        <button type="button" id="viewImagesBtn">View Visualization Analysis of the Churn Prediction Dataset</button>
    </form>
    <div id="imageContainer"></div>
    <script>
        document.getElementById('viewImagesBtn').addEventListener('click', function(event) {
            var imageUrls = ['{{ url_for("static", filename="css/senior_citizen.PNG") }}',
                             '{{ url_for("static", filename="css/gender.PNG") }}',
                             '{{ url_for("static", filename="css/payment_method.PNG") }}'];
            imageUrls.forEach(function(url) {
                var img = document.createElement('img');
                img.src = url;
                img.style.width = '500px';
                img.style.height = 'auto';
                document.getElementById('imageContainer').appendChild(img);
            });
        });
    </script>
    <br>
    <div id='predicted_value'>{{ predicted_value }}</div>
  </div>
</body>
</html>
```

Additionally, to further format our front-end application for deployment, we have employed a CSS script. In addition to improving the appearance, we have added an image.

```css
body {
    margin: 0;
    padding: 0;
    font-family: 'Arial', sans-serif;
}
.main_line {
    max-width: 400px;
    margin: 50px auto;
    padding: 20px;
    background-color: lightyellow;
    border-radius: 8px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
}
.header {
    text-align: center;
    margin-bottom: 20px;
}
.churn-photo {
    width: 20vw;
    height: auto;
    border-radius: 0px;
    object-fit: fill;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
    background-color: lightyellow;
}
.centered-input {
    width: 10%;
    padding: 8px;
    margin: 10px auto;
    box-sizing: border-box;
    border: 1px solid #ddd;
    border-radius: 4px;
    display: inline-block;
}
@keyframes backgroundSlide {
  0% { background-position: 0% 50%; }
  50% { background-position: 100% 50%; }
  100% { background-position: 0% 50%; }
}
form {
  text-align: center;
  background-color: #f0f0f0;
  background-image: linear-gradient(45deg, #6D9898, #E0CA3C, #E37B48, #6D9898);
  background-size: 400% 400%;
  animation: backgroundSlide 15s ease infinite;
}
form:hover {
  background-color: #e0e0e0;
  transition: background-color 0.3s ease;
}
label {
  color: #000;
  padding: 5px 10px;
  border-radius: 5px;
  font-weight: bold;
  display: inline-block;
  margin-bottom: 5px;
}
.display-image {
    width: 100%;
    height: auto;
    margin: 10px 0;
}
input {
    width: calc(100% - 16px);
    padding: 8px;
    margin-bottom: 16px;
    box-sizing: border-box;
    border: 1px solid #ddd;
    border-radius: 4px;
}
button {
    background-color: #4CAF50;
    color: white;
    padding: 10px 20px;
    border: none;
    border-radius: 4px;
    cursor: pointer;
}
button:hover {
    background-color: #45a049;
}
#predicted_value {
    margin-top: 20px;
    text-align: center;
    font-weight: bold;
    color: #333;
}
```

This Python script creates a Flask web application that uses a machine learning model that has been built to forecast the customer churn.

The application uses a pickled file to load a Random Forest model that has already been trained, and it integrates the Flask web framework to generate an interface.

Through an HTML form, users provide information about Gender, Monthly Charges, Payment Method, Paperless Billing, Senior Citizen, Phone Service, Multiple Lines, Total Charges.

Subsequently, the preprocessed input data is incorporated into the model to forecast the likelihood of a customer's retention.

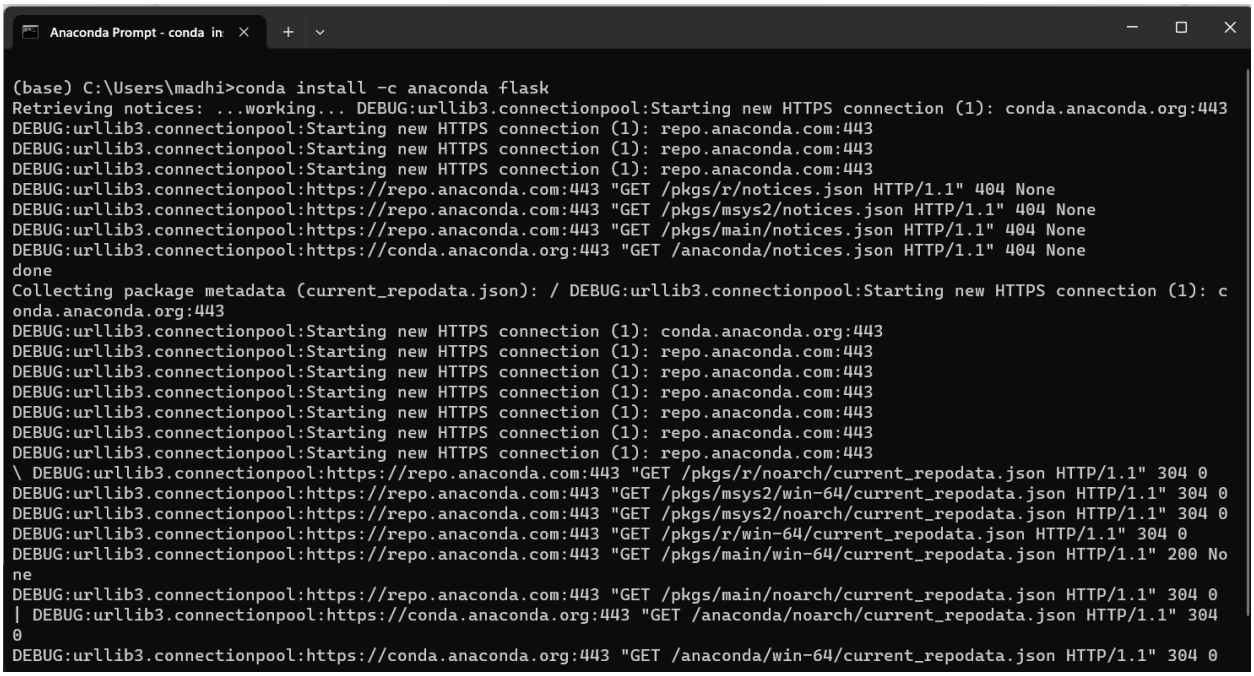
When the application is started as the primary program, it operates in debug mode and displays the forecast on the webpage.

All things considered, this script uses Flask and a trained machine learning model to operate as a functional web application for predicting client turnover.

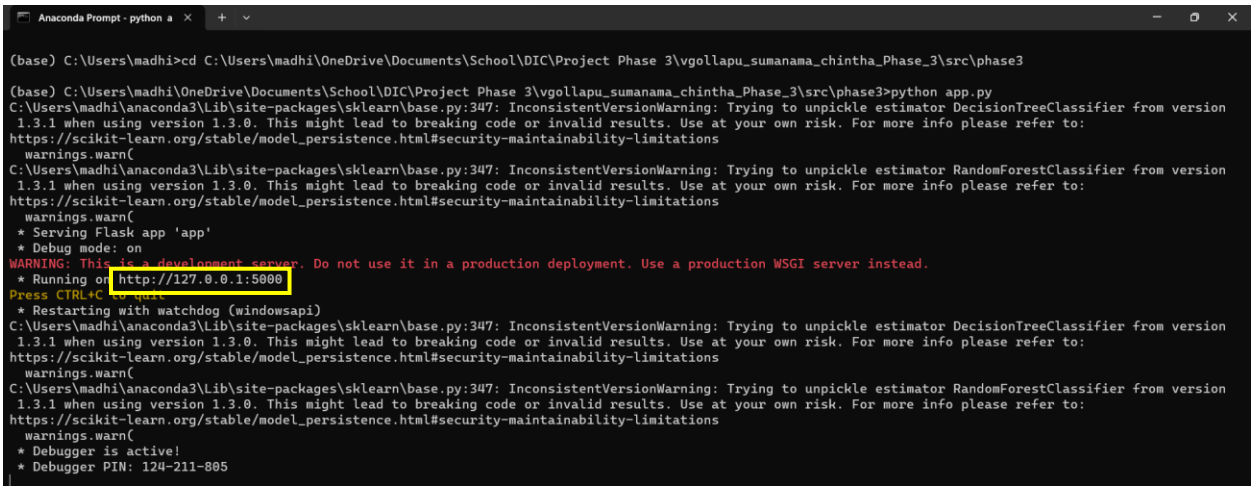
```python
from flask import Flask, render_template, request
import numpy as np
import pickle
app = Flask(__name__)
best_model_rf = pickle.load(open('best_model_rf.p', 'rb'))
monthly_charges_min, monthly_charges_max = 18.25, 219.0
total_charges_min, total_charges_max = 8684.8, 18.8
gender_map = {'male': 1, 'female': 0}
payment_method_map = {'Bank transfer': 0, 'Credit card': 1, 'Electronic check': 2, 'Mailed check': 3}
yes_no_map = {'yes': 1, 'no': 0}
multiple_lines_map = {'No': 0, 'Yes': 2}
@app.route('/')
def index():
    return render_template('input.html')
@app.route('/predict', methods=['POST'])
def predict():
    gender = gender_map[request.form['gender'].lower()]
    monthly_charges = (float(request.form['MonthlyCharges']) - monthly_charges_min) / (monthly_charges_max - monthly_charges_min)
    payment_method = payment_method_map[request.form['PaymentMethod']]
    paperless_billing = yes_no_map[request.form['PaperlessBilling'].lower()]
    senior_citizen = yes_no_map[request.form['SeniorCitizen'].lower()]
    phone_service = yes_no_map[request.form['PhoneService'].lower()]
    multiple_lines = multiple_lines_map[request.form.get('MultipleLines', 'No')]
    total_charges = (float(request.form['TotalCharges']) - total_charges_min) / (total_charges_max - total_charges_min)
    input_data = [
        gender,
        monthly_charges,
        payment_method,
        paperless_billing,
        senior_citizen,
        phone_service,
        multiple_lines,
        total_charges
    ]
    input_data = np.array([input_data])
    prediction = best_model_rf.predict(input_data)
    if prediction:
        value = 'Yes'
    else:
        value = 'No'
    return render_template('input.html', predicted_value=f'Customer Churn prediction is: {value} ')
if __name__ == '__main__':
    app.run(debug=True)
```

Instructions on how our App works:

Install flask in anaconda prompt



Run the app.py file in anaconda prompt, and copy the url link that we get after running the app.py and open the link in browser to run our predictor app.



http://127.0.0.1:5000

Our Customer Churn Predictor APP

Enter below details to predict Customer Churn:

Gender: Select Gender

Monthly Charges:

Payment Method: Select Payment Method

Paperless Billing: Select Paperless Billing Option

Senior Citizen: Select Senior Citizen Option

Phone Service: Select Phone Service Option

Multiple Lines: Select Multiple Lines Option

Total Charges:

Predict Churn | View Visualization Analysis of the Churn Prediction Dataset

After filling the details in the input fields of customer

Enter below details to predict Customer Churn:

Gender: Male

Monthly Charges: 300

Payment Method: Bank Transfer

Paperless Billing: Yes

Senior Citizen: Yes

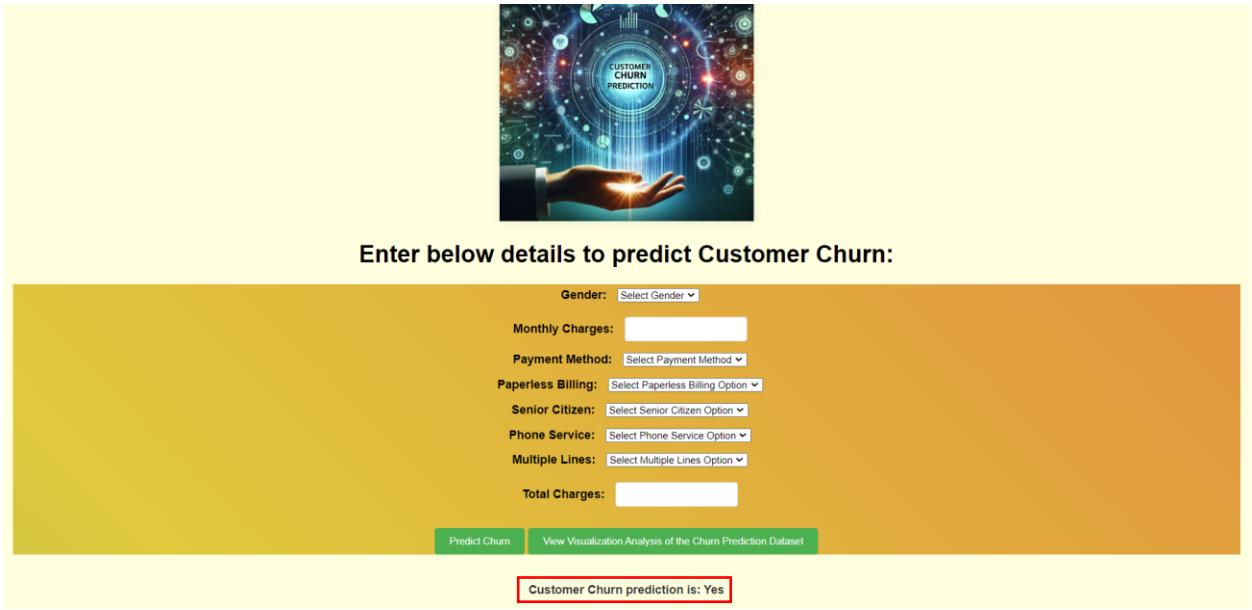
Phone Service: Yes

Multiple Lines: No

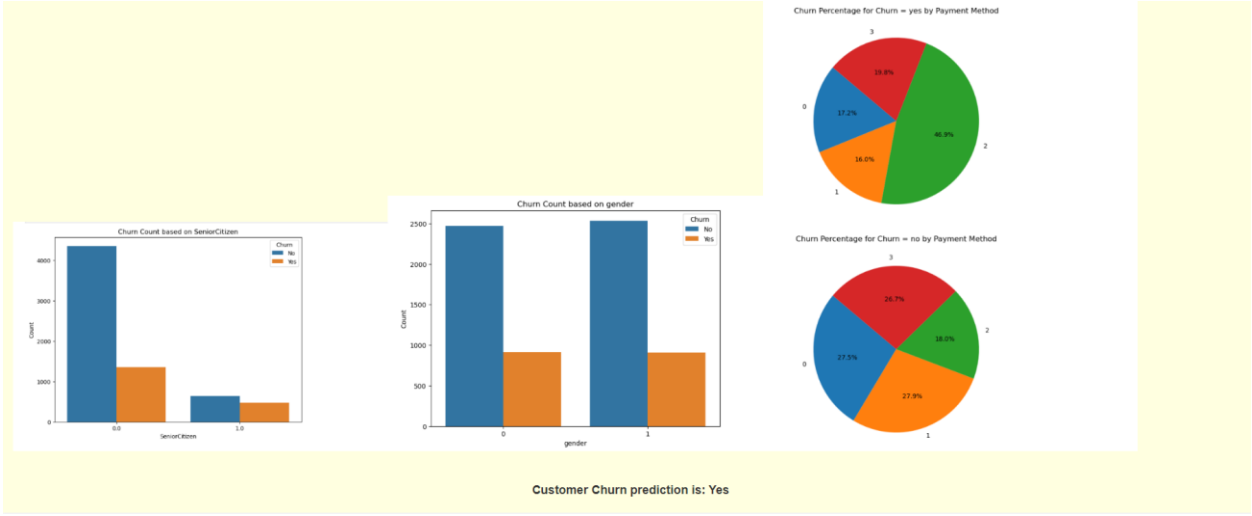
Total Charges: 7000

Predict Churn | View Visualization Analysis of the Churn Prediction Dataset

After clicking on predict button, we will get the churn prediction for the given details of the customer.



When we click on “View Visualization Analysis of the Churn Prediction Dataset” button, we can view the visualizations of our dataset.



Results

A comprehensive strategy that combines data analytics, customer feedback, and proactive initiatives is essential for telecom companies looking to navigate the challenges posed by churn. This approach not only helps in immediate financial gains but also useful in long-erm customer loyalty.

The way telecom firms handle customer churn, the phenomenon of customers switching providers could be revolutionized by this study. The objective is to offer insightful information by utilizing data and forecasts derived from historical client behavior. Businesses can use this information to anticipate potential employees and take proactive measures to keep them from leaving.

Importantly, telecom providers may make better use of their resources thanks to the model's accuracy and efficiency. They can customize their contacts with clients, addressing certain issues and making them feel more personal. Reducing the number of departing clients is the goal, as it is essential to the success of any telecom company.

It may demonstrate to telecom firms how to apply data analytics in a comparable manner, impacting critical decision-making and enhancing long-term customer connections. The true benefit resides in telecom companies adopting a proactive approach that prevents customer losses rather than one that reacts to them after they happen.