Telecom_Churn_Final

October 14, 2024

0.1 Telecom Churn Case Study

With 21 predictor variables we need to predict whether a particular customer will switch to another telecom provider or not. In telecom terminology, this is referred to as churning and not churning, respectively.

[205]:
```
# Suppressing Warnings
import warnings
warnings.filterwarnings('ignore')
```

0.1.1 Step 1: Importing and Merging Data

[206]:
```
# Importing Pandas and NumPy
import pandas as pd, numpy as np
```

[207]:
```
# Importing all datasets
churn_data = pd.read_csv("churn_data.csv")
churn_data.head()
```

[207]:
```
   customerID  tenure PhoneService        Contract PaperlessBilling  \
0  7590-VHVEG       1           No  Month-to-month              Yes
1  5575-GNVDE      34          Yes        One year               No
2  3668-QPYBK       2          Yes  Month-to-month              Yes
3  7795-CFOCW      45           No        One year               No
4  9237-HQITU       2          Yes  Month-to-month              Yes

               PaymentMethod  MonthlyCharges TotalCharges Churn
0           Electronic check           29.85        29.85    No
1               Mailed check           56.95       1889.5    No
2               Mailed check           53.85       108.15   Yes
3  Bank transfer (automatic)           42.30      1840.75    No
4           Electronic check           70.70       151.65   Yes
```

[208]:
```
customer_data = pd.read_csv("customer_data.csv")
customer_data.head()
```

[208]:
```
   customerID  gender  SeniorCitizen Partner Dependents
0  7590-VHVEG  Female              0     Yes         No
```

```
1  5575-GNVDE    Male              0      No          No
2  3668-QPYBK    Male              0      No          No
3  7795-CFOCW    Male              0      No          No
4  9237-HQITU  Female              0      No          No
```

[209]:
```python
internet_data = pd.read_csv("internet_data.csv")
internet_data.head()
```

[209]:
```
   customerID     MultipleLines InternetService OnlineSecurity OnlineBackup  \
0  7590-VHVEG  No phone service             DSL             No          Yes
1  5575-GNVDE                No             DSL            Yes           No
2  3668-QPYBK                No             DSL            Yes          Yes
3  7795-CFOCW  No phone service             DSL            Yes           No
4  9237-HQITU                No     Fiber optic             No           No

  DeviceProtection TechSupport StreamingTV StreamingMovies
0               No          No          No              No
1              Yes          No          No              No
2               No          No          No              No
3              Yes         Yes          No              No
4               No          No          No              No
```

Combining all data files into one consolidated dataframe

[210]:
```python
# Merging on 'customerID'
df_1 = pd.merge(churn_data, customer_data, how='inner', on='customerID')
```

[211]:
```python
# Final dataframe with all predictor variables
telecom = pd.merge(df_1, internet_data, how='inner', on='customerID')
```

0.1.2 Step 2: Inspecting the Dataframe

[212]:
```python
# Let's see the head of our master dataset
telecom.head()
```

[212]:
```
   customerID  tenure PhoneService        Contract PaperlessBilling  \
0  7590-VHVEG       1           No  Month-to-month              Yes
1  5575-GNVDE      34          Yes        One year               No
2  3668-QPYBK       2          Yes  Month-to-month              Yes
3  7795-CFOCW      45           No        One year               No
4  9237-HQITU       2          Yes  Month-to-month              Yes

               PaymentMethod  MonthlyCharges TotalCharges Churn  gender  …  \
0           Electronic check           29.85        29.85    No  Female  …
1               Mailed check           56.95       1889.5    No    Male  …
2               Mailed check           53.85       108.15   Yes    Male  …
3  Bank transfer (automatic)           42.30      1840.75    No    Male  …
```

```
4          Electronic check          70.70       151.65   Yes  Female  …

   Partner Dependents     MultipleLines InternetService OnlineSecurity  \
0      Yes         No  No phone service             DSL             No
1       No         No                No             DSL            Yes
2       No         No                No             DSL            Yes
3       No         No  No phone service             DSL            Yes
4       No         No                No     Fiber optic             No

  OnlineBackup DeviceProtection TechSupport StreamingTV StreamingMovies
0          Yes               No          No          No              No
1           No              Yes          No          No              No
2          Yes               No          No          No              No
3           No              Yes         Yes          No              No
4           No               No          No          No              No

[5 rows x 21 columns]
```

[213]:
```python
# Let's check the dimensions of the dataframe
telecom.shape
```

[213]:
```
(7043, 21)
```

[214]:
```python
# let's look at the statistical aspects of the dataframe
telecom.describe()
```

[214]:

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

[215]:
```python
# Let's see the type of each column
telecom.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   tenure            7043 non-null   int64
 2   PhoneService      7043 non-null   object
```

```
 3   Contract          7043 non-null   object
 4   PaperlessBilling  7043 non-null   object
 5   PaymentMethod     7043 non-null   object
 6   MonthlyCharges    7043 non-null   float64
 7   TotalCharges      7043 non-null   object
 8   Churn             7043 non-null   object
 9   gender            7043 non-null   object
 10  SeniorCitizen     7043 non-null   int64
 11  Partner           7043 non-null   object
 12  Dependents        7043 non-null   object
 13  MultipleLines     7043 non-null   object
 14  InternetService   7043 non-null   object
 15  OnlineSecurity    7043 non-null   object
 16  OnlineBackup      7043 non-null   object
 17  DeviceProtection  7043 non-null   object
 18  TechSupport       7043 non-null   object
 19  StreamingTV       7043 non-null   object
 20  StreamingMovies   7043 non-null   object
dtypes: float64(1), int64(2), object(18)
memory usage: 1.1+ MB
```

0.1.3 Step 3: Data Preparation

Converting some binary variables (Yes/No) to 0/1

[216]:
```python
# List of variables to map

varlist =  ['PhoneService', 'PaperlessBilling', 'Churn', 'Partner',␣
 ↪'Dependents']

# Defining the map function
def binary_map(x):
    return x.map({'Yes': 1, "No": 0})

# Applying the function to the telecom list
telecom[varlist] = telecom[varlist].apply(binary_map)
```

[217]:
```python
telecom.head()
```

[217]:
```
   customerID  tenure  PhoneService        Contract  PaperlessBilling  \
0  7590-VHVEG       1             0  Month-to-month                 1
1  5575-GNVDE      34             1        One year                 0
2  3668-QPYBK       2             1  Month-to-month                 1
3  7795-CFOCW      45             0        One year                 0
4  9237-HQITU       2             1  Month-to-month                 1

              PaymentMethod  MonthlyCharges TotalCharges  Churn  gender  …  \
0          Electronic check           29.85        29.85      0  Female  …
```

```
1              Mailed check           56.95       1889.5      0    Male  ...
2              Mailed check           53.85       108.15      1    Male  ...
3  Bank transfer (automatic)          42.30      1840.75      0    Male  ...
4          Electronic check           70.70       151.65      1  Female  ...

  Partner  Dependents     MultipleLines InternetService OnlineSecurity  \
0       1           0  No phone service             DSL             No
1       0           0                No             DSL            Yes
2       0           0                No             DSL            Yes
3       0           0  No phone service             DSL            Yes
4       0           0                No     Fiber optic             No

  OnlineBackup DeviceProtection TechSupport StreamingTV StreamingMovies
0          Yes               No          No          No              No
1           No              Yes          No          No              No
2          Yes               No          No          No              No
3           No              Yes         Yes          No              No
4           No               No          No          No              No

[5 rows x 21 columns]
```

For categorical variables with multiple levels, create dummy features (one-hot encoded)

[218]:
```python
# Creating a dummy variable for some of the categorical variables and dropping
 ↪the first one.
dummy1 = pd.get_dummies(telecom[['Contract', 'PaymentMethod', 'gender',
 ↪'InternetService']], drop_first=True)

# Adding the results to the master dataframe
telecom = pd.concat([telecom, dummy1], axis=1)
```

[219]:
```python
telecom.head()
```

[219]:
```
   customerID  tenure  PhoneService        Contract  PaperlessBilling  \
0  7590-VHVEG       1             0  Month-to-month                 1
1  5575-GNVDE      34             1        One year                 0
2  3668-QPYBK       2             1  Month-to-month                 1
3  7795-CFOCW      45             0        One year                 0
4  9237-HQITU       2             1  Month-to-month                 1

             PaymentMethod  MonthlyCharges TotalCharges  Churn  gender  ...  \
0         Electronic check           29.85        29.85      0  Female  ...
1             Mailed check           56.95       1889.5      0    Male  ...
2             Mailed check           53.85       108.15      1    Male  ...
3  Bank transfer (automatic)         42.30      1840.75      0    Male  ...
4         Electronic check           70.70       151.65      1  Female  ...
```

```
   StreamingTV  StreamingMovies  Contract_One year Contract_Two year  \
0           No               No              False             False
1           No               No               True             False
2           No               No              False             False
3           No               No               True             False
4           No               No              False             False

  PaymentMethod_Credit card (automatic) PaymentMethod_Electronic check  \
0                                 False                           True
1                                 False                          False
2                                 False                          False
3                                 False                          False
4                                 False                           True

  PaymentMethod_Mailed check gender_Male InternetService_Fiber optic  \
0                      False       False                       False
1                       True        True                       False
2                       True        True                       False
3                      False        True                       False
4                      False       False                        True

  InternetService_No
0              False
1              False
2              False
3              False
4              False

[5 rows x 29 columns]
```

[220]:
```
telecom.MultipleLines.value_counts()
```

[220]:
```
MultipleLines
No                  3390
Yes                 2971
No phone service     682
Name: count, dtype: int64
```

1 Creating dummy variables for the remaining categorical variables and dropping the level with big names.

[221]:
```
# Creating dummy variables for the variable 'MultipleLines'
ml = pd.get_dummies(telecom['MultipleLines'], prefix='MultipleLines')
# Dropping MultipleLines_No phone service column
ml1 = ml.drop(['MultipleLines_No phone service'], axis=1)
```

```python
#Adding the results to the master dataframe
telecom = pd.concat([telecom,ml1], axis=1)

# Creating dummy variables for the variable 'OnlineSecurity'.
os = pd.get_dummies(telecom['OnlineSecurity'], prefix='OnlineSecurity')
os1 = os.drop(['OnlineSecurity_No internet service'], axis = 1)
# Adding the results to the master dataframe
telecom = pd.concat([telecom,os1], axis=1)

# Creating dummy variables for the variable 'OnlineBackup'.
ob = pd.get_dummies(telecom['OnlineBackup'], prefix='OnlineBackup')
ob1 = ob.drop(['OnlineBackup_No internet service'], axis = 1)
# Adding the results to the master dataframe
telecom = pd.concat([telecom,ob1], axis=1)

# Creating dummy variables for the variable 'DeviceProtection'.
dp = pd.get_dummies(telecom['DeviceProtection'], prefix='DeviceProtection')
dp1 = dp.drop(['DeviceProtection_No internet service'], axis = 1)
# Adding the results to the master dataframe
telecom = pd.concat([telecom,dp1], axis=1)

# Creating dummy variables for the variable 'TechSupport'.
ts = pd.get_dummies(telecom['TechSupport'], prefix='TechSupport')
ts1 = ts.drop(['TechSupport_No internet service'], axis =1)
# Adding the results to the master dataframe
telecom = pd.concat([telecom,ts1], axis=1)

# Creating dummy variables for the variable 'StreamingTV'.
st =pd.get_dummies(telecom['StreamingTV'], prefix='StreamingTV')
st1 = st.drop(['StreamingTV_No internet service'], axis =1)
# Adding the results to the master dataframe
telecom = pd.concat([telecom,st1], axis=1)

# Creating dummy variables for the variable 'StreamingMovies'.
sm = pd.get_dummies(telecom['StreamingMovies'], prefix='StreamingMovies')
sm1 = sm.drop(['StreamingMovies_No internet service'], axis =1)
# Adding the results to the master dataframe
telecom = pd.concat([telecom,sm1], axis=1)
```

[222]:
```python
telecom.head()
```

[222]:
```
  customerID  tenure  PhoneService        Contract  PaperlessBilling  \
0  7590-VHVEG       1             0  Month-to-month                 1
1  5575-GNVDE      34             1        One year                 0
2  3668-QPYBK       2             1  Month-to-month                 1
3  7795-CFOCW      45             0        One year                 0
4  9237-HQITU       2             1  Month-to-month                 1
```

```
              PaymentMethod  MonthlyCharges TotalCharges  Churn  gender  …  \
0          Electronic check           29.85        29.85      0  Female  …
1              Mailed check           56.95       1889.5      0    Male  …
2              Mailed check           53.85       108.15      1    Male  …
3 Bank transfer (automatic)           42.30      1840.75      0    Male  …
4          Electronic check           70.70       151.65      1  Female  …

   OnlineBackup_No  OnlineBackup_Yes  DeviceProtection_No  \
0            False              True                 True
1             True             False                False
2            False              True                 True
3             True             False                False
4             True             False                 True

  DeviceProtection_Yes TechSupport_No TechSupport_Yes StreamingTV_No  \
0                False           True           False           True
1                 True           True           False           True
2                False           True           False           True
3                 True          False            True           True
4                False           True           False           True

  StreamingTV_Yes StreamingMovies_No StreamingMovies_Yes
0           False               True               False
1           False               True               False
2           False               True               False
3           False               True               False
4           False               True               False

[5 rows x 43 columns]
```

Dropping the repeated variables

[223]:
```
# We have created dummies for the below variables, so we can drop them
telecom = telecom.
 ↪drop(['Contract','PaymentMethod','gender','MultipleLines','InternetService',␣
 ↪'OnlineSecurity', 'OnlineBackup', 'DeviceProtection',
       'TechSupport', 'StreamingTV', 'StreamingMovies'], axis = 1)
```

[224]:
```
#The varaible was imported as a string we need to convert it to float
telecom['TotalCharges'] = pd.to_numeric(telecom['TotalCharges'],␣
 ↪errors='coerce')
#telecom['TotalCharges'] = telecom['TotalCharges'].
 ↪convert_objects(convert_numeric=True)
```

[225]:
```
telecom.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 32 columns):
 #   Column                                 Non-Null Count  Dtype
---  ------                                 --------------  -----
 0   customerID                             7043 non-null   object
 1   tenure                                 7043 non-null   int64
 2   PhoneService                           7043 non-null   int64
 3   PaperlessBilling                       7043 non-null   int64
 4   MonthlyCharges                         7043 non-null   float64
 5   TotalCharges                           7032 non-null   float64
 6   Churn                                  7043 non-null   int64
 7   SeniorCitizen                          7043 non-null   int64
 8   Partner                                7043 non-null   int64
 9   Dependents                             7043 non-null   int64
 10  Contract_One year                      7043 non-null   bool
 11  Contract_Two year                      7043 non-null   bool
 12  PaymentMethod_Credit card (automatic)  7043 non-null   bool
 13  PaymentMethod_Electronic check         7043 non-null   bool
 14  PaymentMethod_Mailed check             7043 non-null   bool
 15  gender_Male                            7043 non-null   bool
 16  InternetService_Fiber optic            7043 non-null   bool
 17  InternetService_No                     7043 non-null   bool
 18  MultipleLines_No                       7043 non-null   bool
 19  MultipleLines_Yes                      7043 non-null   bool
 20  OnlineSecurity_No                      7043 non-null   bool
 21  OnlineSecurity_Yes                     7043 non-null   bool
 22  OnlineBackup_No                        7043 non-null   bool
 23  OnlineBackup_Yes                       7043 non-null   bool
 24  DeviceProtection_No                    7043 non-null   bool
 25  DeviceProtection_Yes                   7043 non-null   bool
 26  TechSupport_No                         7043 non-null   bool
 27  TechSupport_Yes                        7043 non-null   bool
 28  StreamingTV_No                         7043 non-null   bool
 29  StreamingTV_Yes                        7043 non-null   bool
 30  StreamingMovies_No                     7043 non-null   bool
 31  StreamingMovies_Yes                    7043 non-null   bool
dtypes: bool(22), float64(2), int64(7), object(1)
memory usage: 701.7+ KB
```

Checking for Outliers

[226]:
```python
# Checking for outliers in the continuous variables
num_telecom =
 ↪telecom[['tenure','MonthlyCharges','SeniorCitizen','TotalCharges']]
```

[227]:
```python
# Checking outliers at 25%, 50%, 75%, 90%, 95% and 99%
num_telecom.describe(percentiles=[.25, .5, .75, .90, .95, .99])
```

[227]:

	tenure	MonthlyCharges	SeniorCitizen	TotalCharges
count	7043.000000	7043.000000	7043.000000	7032.000000
mean	32.371149	64.761692	0.162147	2283.300441
std	24.559481	30.090047	0.368612	2266.771362
min	0.000000	18.250000	0.000000	18.800000
25%	9.000000	35.500000	0.000000	401.450000
50%	29.000000	70.350000	0.000000	1397.475000
75%	55.000000	89.850000	0.000000	3794.737500
90%	69.000000	102.600000	1.000000	5976.640000
95%	72.000000	107.400000	1.000000	6923.590000
99%	72.000000	114.729000	1.000000	8039.883000
max	72.000000	118.750000	1.000000	8684.800000

Checking for Missing Values and Inputing Them

[228]:
```
# Adding up the missing values (column-wise)
telecom.isnull().sum()
```

[228]:
```
customerID                                0
tenure                                    0
PhoneService                              0
PaperlessBilling                          0
MonthlyCharges                            0
TotalCharges                             11
Churn                                     0
SeniorCitizen                             0
Partner                                   0
Dependents                                0
Contract_One year                         0
Contract_Two year                         0
PaymentMethod_Credit card (automatic)     0
PaymentMethod_Electronic check            0
PaymentMethod_Mailed check                0
gender_Male                               0
InternetService_Fiber optic               0
InternetService_No                        0
MultipleLines_No                          0
MultipleLines_Yes                         0
OnlineSecurity_No                         0
OnlineSecurity_Yes                        0
OnlineBackup_No                           0
OnlineBackup_Yes                          0
DeviceProtection_No                       0
DeviceProtection_Yes                      0
TechSupport_No                            0
TechSupport_Yes                           0
StreamingTV_No                            0
```

```
StreamingTV_Yes                          0
StreamingMovies_No                       0
StreamingMovies_Yes                      0
dtype: int64
```

It means that 11/7043 = 0.001561834 i.e 0.1%, best is to remove these observations from the analysis

[229]:
```
# Checking the percentage of missing values
round(100*(telecom.isnull().sum()/len(telecom.index)), 2)
```

[229]:
```
customerID                                0.00
tenure                                    0.00
PhoneService                              0.00
PaperlessBilling                          0.00
MonthlyCharges                            0.00
TotalCharges                              0.16
Churn                                     0.00
SeniorCitizen                             0.00
Partner                                   0.00
Dependents                                0.00
Contract_One year                         0.00
Contract_Two year                         0.00
PaymentMethod_Credit card (automatic)     0.00
PaymentMethod_Electronic check            0.00
PaymentMethod_Mailed check                0.00
gender_Male                               0.00
InternetService_Fiber optic               0.00
InternetService_No                        0.00
MultipleLines_No                          0.00
MultipleLines_Yes                         0.00
OnlineSecurity_No                         0.00
OnlineSecurity_Yes                        0.00
OnlineBackup_No                           0.00
OnlineBackup_Yes                          0.00
DeviceProtection_No                       0.00
DeviceProtection_Yes                      0.00
TechSupport_No                            0.00
TechSupport_Yes                           0.00
StreamingTV_No                            0.00
StreamingTV_Yes                           0.00
StreamingMovies_No                        0.00
StreamingMovies_Yes                       0.00
dtype: float64
```

[230]:
```
# Removing NaN TotalCharges rows
telecom = telecom[~np.isnan(telecom['TotalCharges'])]
```

[231]:
```python
# Checking percentage of missing values after removing the missing values
round(100*(telecom.isnull().sum()/len(telecom.index)), 2)
```

[231]:
```
customerID                               0.0
tenure                                   0.0
PhoneService                             0.0
PaperlessBilling                         0.0
MonthlyCharges                           0.0
TotalCharges                             0.0
Churn                                    0.0
SeniorCitizen                            0.0
Partner                                  0.0
Dependents                               0.0
Contract_One year                        0.0
Contract_Two year                        0.0
PaymentMethod_Credit card (automatic)    0.0
PaymentMethod_Electronic check           0.0
PaymentMethod_Mailed check               0.0
gender_Male                              0.0
InternetService_Fiber optic              0.0
InternetService_No                       0.0
MultipleLines_No                         0.0
MultipleLines_Yes                        0.0
OnlineSecurity_No                        0.0
OnlineSecurity_Yes                       0.0
OnlineBackup_No                          0.0
OnlineBackup_Yes                         0.0
DeviceProtection_No                      0.0
DeviceProtection_Yes                     0.0
TechSupport_No                           0.0
TechSupport_Yes                          0.0
StreamingTV_No                           0.0
StreamingTV_Yes                          0.0
StreamingMovies_No                       0.0
StreamingMovies_Yes                      0.0
dtype: float64
```

1.0.1 Step 4: Test-Train Split

[232]:
```python
from sklearn.model_selection import train_test_split
```

[233]:
```python
# Putting feature variable to X
X = telecom.drop(['Churn', 'customerID'], axis = 1)

# Putting the response variable to y
y = telecom['Churn']
```

[234]:
```
X.head()
```

[234]:
```
   tenure  PhoneService  PaperlessBilling  MonthlyCharges  TotalCharges  \
0       1             0                 1           29.85         29.85
1      34             1                 0           56.95       1889.50
2       2             1                 1           53.85        108.15
3      45             0                 0           42.30       1840.75
4       2             1                 1           70.70        151.65

   SeniorCitizen  Partner  Dependents  Contract_One year  Contract_Two year  \
0              0        1           0              False              False
1              0        0           0               True              False
2              0        0           0              False              False
3              0        0           0               True              False
4              0        0           0              False              False

   …  OnlineBackup_No  OnlineBackup_Yes  DeviceProtection_No  \
0  …            False              True                 True
1  …             True             False                False
2  …            False              True                 True
3  …             True             False                False
4  …             True             False                 True

   DeviceProtection_Yes  TechSupport_No  TechSupport_Yes  StreamingTV_No  \
0                 False            True            False            True
1                  True            True            False            True
2                 False            True            False            True
3                  True           False             True            True
4                 False            True            False            True

   StreamingTV_Yes  StreamingMovies_No  StreamingMovies_Yes
0            False                True                False
1            False                True                False
2            False                True                False
3            False                True                False
4            False                True                False

[5 rows x 30 columns]
```

[235]:
```
y.head()
```

[235]:
```
0    0
1    0
2    1
3    0
4    1
Name: Churn, dtype: int64
```

[236]:
```
# Splitting the data into train and test
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size = 0.7,␣
↪random_state = 100)
```

1.0.2 Step 5: Feature Scaling

[237]:
```
#Feature scaling
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train[['tenure', 'MonthlyCharges', 'TotalCharges']] = scaler.
↪fit_transform(X_train[['tenure', 'MonthlyCharges', 'TotalCharges']])
X_train.head()
```

[237]:
```
        tenure  PhoneService  PaperlessBilling  MonthlyCharges  TotalCharges  \
879   0.019693             1                 1       -0.338074     -0.276449
5790  0.305384             0                 1       -0.464443     -0.112702
6498 -1.286319             1                 1        0.581425     -0.974430
880  -0.919003             1                 1        1.505913     -0.550676
2784 -1.163880             1                 1        1.106854     -0.835971

      SeniorCitizen  Partner  Dependents  Contract_One year  \
879               0        0           0              False
5790              0        1           1              False
6498              0        0           0              False
880               0        0           0              False
2784              0        0           1              False

      Contract_Two year  …  OnlineBackup_No  OnlineBackup_Yes  \
879               False  …            False              True
5790              False  …            False              True
6498              False  …            False              True
880               False  …            False              True
2784              False  …             True             False

      DeviceProtection_No  DeviceProtection_Yes  TechSupport_No  \
879                  True                 False            True
5790                 True                 False            True
6498                False                  True            True
880                 False                  True           False
2784                False                  True           False

      TechSupport_Yes  StreamingTV_No  StreamingTV_Yes  StreamingMovies_No  \
879             False            True            False                True
5790            False           False             True               False
6498            False            True            False                True
880              True           False             True               False
2784             True           False             True               False
```

```
      StreamingMovies_Yes
879                 False
5790                 True
6498                False
880                  True
2784                 True

[5 rows x 30 columns]
```

[238]:

```
### Checking the Churn Rate
churn = (sum(telecom['Churn']) / len(telecom['Churn'].index)) * 100
churn

# We have almost 26.5% of churn rate
```

[238]: 26.578498293515356

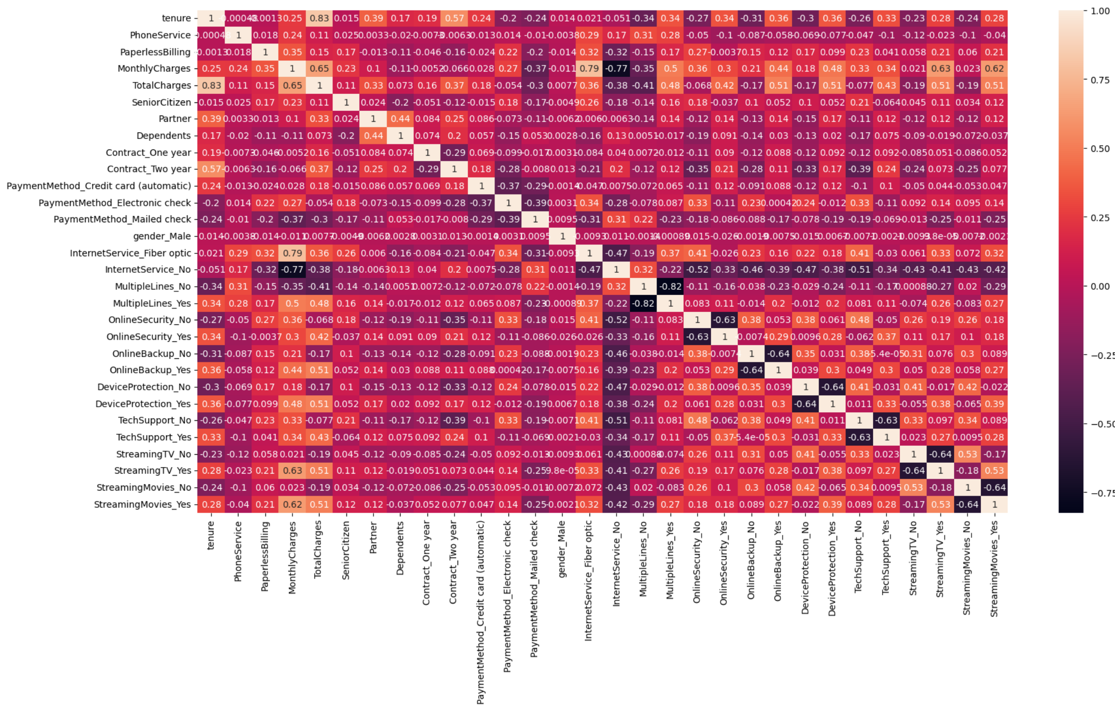
1.0.3 Step 6: Looking at Correlations

[239]:

```
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
```

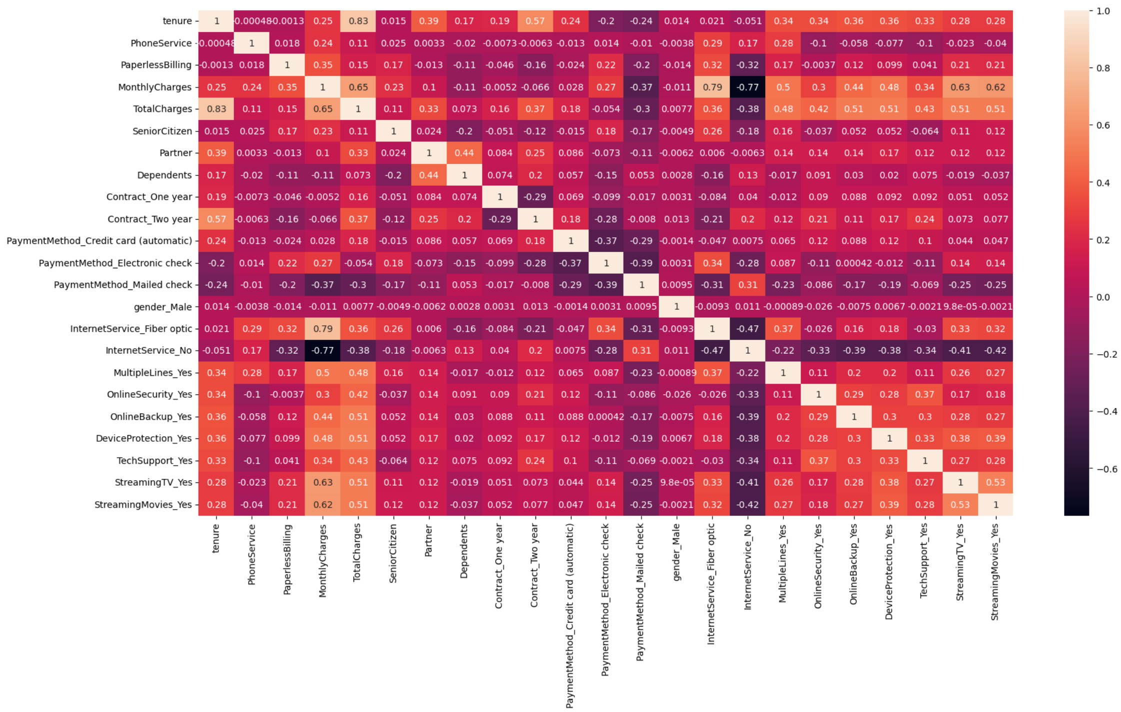
[240]:

```
# let us see the correlation matrix
plt.figure(figsize=(20,10))
sns.heatmap(X_train.corr(), annot = True)
plt.show()
```



```
[241]: #### Dropping highly correlated dummy variables

X_train = X_train.drop(['MultipleLines_No', 'OnlineSecurity_No',␣
 ↪'OnlineBackup_No',␣
 ↪'DeviceProtection_No','TechSupport_No','StreamingTV_No','StreamingMovies_No'],␣
 ↪axis = 1)
X_test = X_test.drop(['MultipleLines_No', 'OnlineSecurity_No',␣
 ↪'OnlineBackup_No',␣
 ↪'DeviceProtection_No','TechSupport_No','StreamingTV_No','StreamingMovies_No'],␣
 ↪axis = 1)
```

```
[242]: # let us see the correlation matrix after dropping highly correlated columns
plt.figure(figsize=(20,10))
sns.heatmap(X_train.corr(), annot = True)
plt.show()
```



[243]:
```
X_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 4922 entries, 879 to 5649
Data columns (total 23 columns):
 #   Column                                 Non-Null Count  Dtype
---  ------                                 --------------  -----
 0   tenure                                 4922 non-null   float64
 1   PhoneService                           4922 non-null   int64
 2   PaperlessBilling                       4922 non-null   int64
 3   MonthlyCharges                         4922 non-null   float64
 4   TotalCharges                           4922 non-null   float64
 5   SeniorCitizen                          4922 non-null   int64
 6   Partner                                4922 non-null   int64
 7   Dependents                             4922 non-null   int64
 8   Contract_One year                      4922 non-null   bool
 9   Contract_Two year                      4922 non-null   bool
 10  PaymentMethod_Credit card (automatic)  4922 non-null   bool
 11  PaymentMethod_Electronic check         4922 non-null   bool
 12  PaymentMethod_Mailed check             4922 non-null   bool
 13  gender_Male                            4922 non-null   bool
 14  InternetService_Fiber optic            4922 non-null   bool
 15  InternetService_No                     4922 non-null   bool
 16  MultipleLines_Yes                      4922 non-null   bool
 17  OnlineSecurity_Yes                     4922 non-null   bool
```

```
 18  OnlineBackup_Yes                       4922 non-null   bool
 19  DeviceProtection_Yes                   4922 non-null   bool
 20  TechSupport_Yes                        4922 non-null   bool
 21  StreamingTV_Yes                        4922 non-null   bool
 22  StreamingMovies_Yes                    4922 non-null   bool
dtypes: bool(15), float64(3), int64(5)
memory usage: 418.2 KB
```

[244]:
```
bool_columns = X_train.select_dtypes(include=['bool']).columns
X_train[bool_columns] = X_train[bool_columns].astype(int)
```

1.0.4 Step 7: Model Building

Let's start by splitting our data into a training set and a test set.

[245]:
```
# Running Your First Training Model
import statsmodels.api as sm

# Logistic regression model
X_train_const = sm.add_constant(X_train)
logm1 = sm.GLM(y_train, X_train_const, family=sm.families.Binomial())
result = logm1.fit()
print(result.summary())
```

```
                 Generalized Linear Model Regression Results
==============================================================================
Dep. Variable:                  Churn   No. Observations:                 4922
Model:                            GLM   Df Residuals:                     4898
Model Family:                Binomial   Df Model:                           23
Link Function:                  Logit   Scale:                          1.0000
Method:                          IRLS   Log-Likelihood:                -2004.7
Date:                Wed, 07 Aug 2024   Deviance:                       4009.4
Time:                        03:42:23   Pearson chi2:                 6.07e+03
No. Iterations:                     7   Pseudo R-squ. (CS):             0.2844
Covariance Type:            nonrobust
================================================================================
=========================
                                            coef    std err          z
P>|z|      [0.025      0.975]
--------------------------------------------------------------------------------
-------------------------
const                                    -3.9382      1.546     -2.547
0.011      -6.969      -0.908
tenure                                   -1.5172      0.189     -8.015
0.000      -1.888      -1.146
PhoneService                              0.9507      0.789      1.205
0.228      -0.595       2.497
PaperlessBilling                          0.3254      0.090      3.614
```

```
0.000       0.149       0.502
MonthlyCharges                           -2.1806      1.160     -1.880
0.060      -4.454       0.092
TotalCharges                              0.7332      0.198      3.705
0.000       0.345       1.121
SeniorCitizen                             0.3984      0.102      3.924
0.000       0.199       0.597
Partner                                   0.0374      0.094      0.399
0.690      -0.146       0.221
Dependents                               -0.1430      0.107     -1.332
0.183      -0.353       0.067
Contract_One year                        -0.6578      0.129     -5.106
0.000      -0.910      -0.405
Contract_Two year                        -1.2455      0.212     -5.874
0.000      -1.661      -0.830
PaymentMethod_Credit card (automatic)    -0.2577      0.137     -1.883
0.060      -0.526       0.011
PaymentMethod_Electronic check            0.1615      0.113      1.434
0.152      -0.059       0.382
PaymentMethod_Mailed check               -0.2536      0.137     -1.845
0.065      -0.523       0.016
gender_Male                              -0.0346      0.078     -0.442
0.658      -0.188       0.119
InternetService_Fiber optic               2.5124      0.967      2.599
0.009       0.618       4.407
InternetService_No                       -2.7792      0.982     -2.831
0.005      -4.703      -0.855
MultipleLines_Yes                         0.5623      0.214      2.628
0.009       0.143       0.982
OnlineSecurity_Yes                       -0.0245      0.216     -0.113
0.910      -0.448       0.399
OnlineBackup_Yes                          0.1740      0.212      0.822
0.411      -0.241       0.589
DeviceProtection_Yes                      0.3229      0.215      1.501
0.133      -0.099       0.744
TechSupport_Yes                          -0.0305      0.216     -0.141
0.888      -0.455       0.394
StreamingTV_Yes                           0.9598      0.396      2.423
0.015       0.183       1.736
StreamingMovies_Yes                       0.8484      0.396      2.143
0.032       0.072       1.624
================================================================================
=========================
```

1.0.5 Step 8: Feature Selection Using RFE

[246]:
```
# Feature selection
from sklearn.linear_model import LogisticRegression
from sklearn.feature_selection import RFE

logreg = LogisticRegression(max_iter=1000)

rfe = RFE(estimator=logreg, n_features_to_select=13)
rfe = rfe.fit(X_train, y_train)
```

[247]:
```
print("Selected features:", X_train.columns[rfe.support_])
```

```
Selected features: Index(['tenure', 'MonthlyCharges', 'TotalCharges',
'SeniorCitizen',
       'Contract_One year', 'Contract_Two year',
       'PaymentMethod_Credit card (automatic)', 'PaymentMethod_Mailed check',
       'InternetService_Fiber optic', 'InternetService_No',
       'MultipleLines_Yes', 'StreamingTV_Yes', 'StreamingMovies_Yes'],
      dtype='object')
```

[248]:
```
rfe.support_
```

[248]:
```
array([ True, False, False,  True,  True,  True, False, False,  True,
        True,  True, False,  True, False,  True,  True,  True, False,
       False, False, False,  True,  True])
```

[249]:
```
list(zip(X_train.columns, rfe.support_, rfe.ranking_))
```

[249]:
```
[('tenure', True, 1),
 ('PhoneService', False, 5),
 ('PaperlessBilling', False, 2),
 ('MonthlyCharges', True, 1),
 ('TotalCharges', True, 1),
 ('SeniorCitizen', True, 1),
 ('Partner', False, 9),
 ('Dependents', False, 8),
 ('Contract_One year', True, 1),
 ('Contract_Two year', True, 1),
 ('PaymentMethod_Credit card (automatic)', True, 1),
 ('PaymentMethod_Electronic check', False, 6),
 ('PaymentMethod_Mailed check', True, 1),
 ('gender_Male', False, 10),
 ('InternetService_Fiber optic', True, 1),
 ('InternetService_No', True, 1),
 ('MultipleLines_Yes', True, 1),
 ('OnlineSecurity_Yes', False, 4),
 ('OnlineBackup_Yes', False, 7),
```

```
 ('DeviceProtection_Yes', False, 11),
 ('TechSupport_Yes', False, 3),
 ('StreamingTV_Yes', True, 1),
 ('StreamingMovies_Yes', True, 1)]
```

[250]:
```
col = X_train.columns[rfe.support_]
X_train.columns[~rfe.support_]
```

[250]:
```
Index(['PhoneService', 'PaperlessBilling', 'Partner', 'Dependents',
       'PaymentMethod_Electronic check', 'gender_Male', 'OnlineSecurity_Yes',
       'OnlineBackup_Yes', 'DeviceProtection_Yes', 'TechSupport_Yes'],
      dtype='object')
```

Assessing the model with StatsModels

[251]:
```
X_train_sm = sm.add_constant(X_train[col])
logm2 = sm.GLM(y_train, X_train_sm, family = sm.families.Binomial())
res = logm2.fit()
print(res.summary())
```

```
                 Generalized Linear Model Regression Results
==============================================================================
Dep. Variable:                  Churn   No. Observations:                 4922
Model:                            GLM   Df Residuals:                     4908
Model Family:                Binomial   Df Model:                           13
Link Function:                  Logit   Scale:                          1.0000
Method:                          IRLS   Log-Likelihood:                -2020.8
Date:                Wed, 07 Aug 2024   Deviance:                       4041.6
Time:                        03:42:24   Pearson chi2:                 6.15e+03
No. Iterations:                     7   Pseudo R-squ. (CS):             0.2797
Covariance Type:            nonrobust
================================================================================
=========================
                                            coef    std err          z
P>|z|      [0.025      0.975]
--------------------------------------------------------------------------------
-------------------------
const                                    -2.2218      0.164    -13.559
0.000      -2.543      -1.901
tenure                                   -1.5388      0.186     -8.257
0.000      -1.904      -1.174
MonthlyCharges                           -1.1395      0.185     -6.173
0.000      -1.501      -0.778
TotalCharges                              0.7223      0.197      3.673
0.000       0.337       1.108
SeniorCitizen                             0.4614      0.099      4.655
0.000       0.267       0.656
Contract_One year                        -0.7326      0.127     -5.769
```

```
0.000      -0.981      -0.484
Contract_Two year                          -1.4007      0.208     -6.722
0.000      -1.809      -0.992
PaymentMethod_Credit card (automatic)      -0.3790      0.112     -3.376
0.001      -0.599      -0.159
PaymentMethod_Mailed check                 -0.4083      0.111     -3.690
0.000      -0.625      -0.191
InternetService_Fiber optic                 1.8340      0.198      9.276
0.000       1.446       2.221
InternetService_No                         -1.8156      0.213     -8.533
0.000      -2.233      -1.399
MultipleLines_Yes                           0.4351      0.102      4.268
0.000       0.235       0.635
StreamingTV_Yes                             0.6440      0.111      5.776
0.000       0.425       0.863
StreamingMovies_Yes                         0.5260      0.109      4.806
0.000       0.311       0.740
================================================================================
=========================
```

[252]:
```
# Getting the predicted values on the train set
y_train_pred = res.predict(X_train_sm)
y_train_pred[:10]
```

[252]:
```
879     0.163642
5790    0.254667
6498    0.556098
880     0.520664
2784    0.670002
3874    0.366496
5387    0.544218
6623    0.792773
4465    0.201923
5364    0.476004
dtype: float64
```

[253]:
```
y_train_pred = y_train_pred.values.reshape(-1)
y_train_pred[:10]
```

[253]:
```
array([0.16364211, 0.25466655, 0.55609771, 0.52066367, 0.67000227,
       0.366496  , 0.54421765, 0.7927729 , 0.2019225 , 0.47600412])
```

[254]:
```
##### Creating a dataframe with the actual churn flag and the predicted␣
 ↪probabilities
y_train_pred_final = pd.DataFrame({'Churn': y_train.values, 'Churn_Prob':␣
 ↪y_train_pred})
y_train_pred_final['CustID'] = y_train.index
```

```
y_train_pred_final.head()
```

[254]:

	Churn	Churn_Prob	CustID
0	0	0.163642	879
1	0	0.254667	5790
2	1	0.556098	6498
3	1	0.520664	880
4	1	0.670002	2784

[255]:
```
##### Creating new column 'predicted' with 1 if Churn_Prob > 0.5 else 0

y_train_pred_final['predicted'] = y_train_pred_final.Churn_Prob.map(lambda x: 1
 ↪if x > 0.5 else 0)
y_train_pred_final.head()
```

[255]:

	Churn	Churn_Prob	CustID	predicted
0	0	0.163642	879	0
1	0	0.254667	5790	0
2	1	0.556098	6498	1
3	1	0.520664	880	1
4	1	0.670002	2784	1

[256]:
```
# Confusion matrix

from sklearn import metrics
confusion = metrics.confusion_matrix(y_train_pred_final.Churn,
 ↪y_train_pred_final.predicted)
print(confusion)
```

```
[[3276  359]
 [ 596  691]]
```

[257]:
```
# Let's check the overall accuracy.

print(metrics.accuracy_score(y_train_pred_final.Churn, y_train_pred_final.
 ↪predicted))
```

```
0.8059731816334823
```

[258]:
```
#### Checking VIFs
from statsmodels.stats.outliers_influence import variance_inflation_factor

vif = pd.DataFrame()
vif['Features'] = X_train_sm[col].columns
vif['VIF'] = [variance_inflation_factor(X_train_sm[col].values, i) for i in
 ↪range (X_train_sm[col].shape[1])]
vif['VIF'] = round(vif['VIF'], 2)
```

```python
vif = vif.sort_values(by = 'VIF', ascending = False)
vif
```

[258]:

	Features	VIF
1	MonthlyCharges	13.74
2	TotalCharges	10.36
0	tenure	7.31
9	InternetService_No	4.90
8	InternetService_Fiber optic	4.57
5	Contract_Two year	2.82
11	StreamingTV_Yes	2.64
12	StreamingMovies_Yes	2.64
10	MultipleLines_Yes	2.28
4	Contract_One year	1.73
7	PaymentMethod_Mailed check	1.67
6	PaymentMethod_Credit card (automatic)	1.43
3	SeniorCitizen	1.32

[259]:

```python
# Dropping column which has high VIFs

col = col.drop(['MonthlyCharges'], 1)
col
```

[259]:

```
Index(['tenure', 'TotalCharges', 'SeniorCitizen', 'Contract_One year',
       'Contract_Two year', 'PaymentMethod_Credit card (automatic)',
       'PaymentMethod_Mailed check', 'InternetService_Fiber optic',
       'InternetService_No', 'MultipleLines_Yes', 'StreamingTV_Yes',
       'StreamingMovies_Yes'],
      dtype='object')
```

[260]:

```python
# Re-run the model using the selected variables
X_train_sm = sm.add_constant(X_train[col])
logm3 = sm.GLM(y_train, X_train_sm, family = sm.families.Binomial())
res = logm3.fit()
print(res.summary())
```

```
                 Generalized Linear Model Regression Results
==============================================================================
Dep. Variable:                  Churn   No. Observations:                 4922
Model:                            GLM   Df Residuals:                     4909
Model Family:                Binomial   Df Model:                           12
Link Function:                  Logit   Scale:                          1.0000
Method:                          IRLS   Log-Likelihood:                -2040.2
Date:                Wed, 07 Aug 2024   Deviance:                       4080.4
Time:                        03:42:24   Pearson chi2:                 5.62e+03
No. Iterations:                     7   Pseudo R-squ. (CS):             0.2740
Covariance Type:            nonrobust
```

```
==============================================================================
========================
                                                 coef    std err          z
P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
------------------------
const                                         -1.5257      0.113    -13.468
0.000      -1.748      -1.304
tenure                                        -1.2295      0.177     -6.931
0.000      -1.577      -0.882
TotalCharges                                   0.3023      0.183      1.651
0.099      -0.057       0.661
SeniorCitizen                                  0.5105      0.099      5.177
0.000       0.317       0.704
Contract_One year                             -0.8131      0.126     -6.464
0.000      -1.060      -0.567
Contract_Two year                             -1.5074      0.207     -7.276
0.000      -1.913      -1.101
PaymentMethod_Credit card (automatic)    -0.4026      0.112     -3.597
0.000      -0.622      -0.183
PaymentMethod_Mailed check                    -0.4061      0.109     -3.711
0.000      -0.621      -0.192
InternetService_Fiber optic                    0.8270      0.106      7.785
0.000       0.619       1.035
InternetService_No                            -0.8864      0.152     -5.823
0.000      -1.185      -0.588
MultipleLines_Yes                              0.1970      0.093      2.118
0.034       0.015       0.379
StreamingTV_Yes                                0.2956      0.095      3.095
0.002       0.108       0.483
StreamingMovies_Yes                            0.1984      0.095      2.087
0.037       0.012       0.385
==============================================================================
========================
```

[261]:
```
y_train_pred = res.predict(X_train_sm).values.reshape(-1)
y_train_pred[:10]
y_train_pred_final['Churn_Prob'] = y_train_pred
```

[262]:
```
# Creating new column 'predicted' with 1 if Churn_Prob > 0.5 else 0

y_train_pred_final['predicted'] = y_train_pred_final.Churn_Prob.map(lambda x: 1␣
↪if x > 0.5 else 0)
```

[263]:
```
# Let's check the overall accuracy.
print(metrics.accuracy_score(y_train_pred_final.Churn, y_train_pred_final.
↪predicted))
```

```
0.8021129622104836
```

The accuracy is still practically the same.

Let's now check the VIFs again

[264]:
```
vif = pd.DataFrame()
vif['Features'] = X_train_sm[col].columns
vif['VIF'] = [variance_inflation_factor(X_train_sm[col].values, i) for i in␣
 ↪range (X_train_sm[col].shape[1])]
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = 'VIF', ascending = False)
vif
```

[264]:

	Features	VIF
1	TotalCharges	7.12
0	tenure	6.79
4	Contract_Two year	2.75
7	InternetService_Fiber optic	2.60
11	StreamingMovies_Yes	2.52
10	StreamingTV_Yes	2.51
8	InternetService_No	2.30
9	MultipleLines_Yes	2.22
3	Contract_One year	1.66
6	PaymentMethod_Mailed check	1.57
5	PaymentMethod_Credit card (automatic)	1.39
2	SeniorCitizen	1.29

[265]:
```
# Re-run the model without TotalCharges

col = col.drop(['TotalCharges'], 1)
col
```

[265]:
```
Index(['tenure', 'SeniorCitizen', 'Contract_One year', 'Contract_Two year',
       'PaymentMethod_Credit card (automatic)', 'PaymentMethod_Mailed check',
       'InternetService_Fiber optic', 'InternetService_No',
       'MultipleLines_Yes', 'StreamingTV_Yes', 'StreamingMovies_Yes'],
      dtype='object')
```

[266]:
```
X_train_sm = sm.add_constant(X_train[col])
logm4 = sm.GLM(y_train, X_train_sm, family = sm.families.Binomial())
res = logm4.fit()
print(res.summary())
```

```
                 Generalized Linear Model Regression Results
==============================================================================
Dep. Variable:                  Churn   No. Observations:                 4922
Model:                            GLM   Df Residuals:                     4910
Model Family:                Binomial   Df Model:                           11
```

```
Link Function:                  Logit   Scale:                          1.0000
Method:                          IRLS   Log-Likelihood:                -2041.6
Date:                Wed, 07 Aug 2024   Deviance:                       4083.2
Time:                        03:42:25   Pearson chi2:                 5.23e+03
No. Iterations:                     7   Pseudo R-squ. (CS):             0.2736
Covariance Type:            nonrobust
================================================================================
=========================
                                           coef    std err          z
P>|z|      [0.025      0.975]
--------------------------------------------------------------------------------
-------------------------
const                                   -1.5642      0.110    -14.183
0.000      -1.780      -1.348
tenure                                  -0.9594      0.065    -14.842
0.000      -1.086      -0.833
SeniorCitizen                            0.5110      0.099      5.170
0.000       0.317       0.705
Contract_One year                       -0.8054      0.125     -6.429
0.000      -1.051      -0.560
Contract_Two year                       -1.4776      0.205     -7.194
0.000      -1.880      -1.075
PaymentMethod_Credit card (automatic)   -0.4018      0.112     -3.586
0.000      -0.621      -0.182
PaymentMethod_Mailed check              -0.3831      0.108     -3.531
0.000      -0.596      -0.170
InternetService_Fiber optic              0.9038      0.095      9.467
0.000       0.717       1.091
InternetService_No                      -0.9031      0.151     -5.968
0.000      -1.200      -0.607
MultipleLines_Yes                        0.2237      0.092      2.444
0.015       0.044       0.403
StreamingTV_Yes                          0.3321      0.093      3.575
0.000       0.150       0.514
StreamingMovies_Yes                      0.2334      0.093      2.519
0.012       0.052       0.415
================================================================================
=========================
```

[267]:
```
y_train_pred = res.predict(X_train_sm).values.reshape(-1)
y_train_pred[:10]
y_train_pred_final['Churn_Prob'] = y_train_pred
```

[268]:
```
y_train_pred_final['predicted'] = y_train_pred_final.Churn_Prob.map(lambda x: 1␣
↪if x > 0.5 else 0)
```

[269]:
```
print(metrics.accuracy_score(y_train_pred_final.Churn, y_train_pred_final.
  ↪predicted))
```

```
0.802519301097115
```

[270]:
```
vif = pd.DataFrame()
vif['Features'] = X_train_sm[col].columns
vif['VIF'] = [variance_inflation_factor(X_train_sm[col].values, i) for i in␣
  ↪range (X_train_sm[col].shape[1])]
vif['VIF'] = round(vif['VIF'], 2)
vif = vif.sort_values(by = 'VIF', ascending = False)
vif
```

[270]:

	Features	VIF
3	Contract_Two year	2.68
10	StreamingMovies_Yes	2.46
9	StreamingTV_Yes	2.44
6	InternetService_Fiber optic	2.42
8	MultipleLines_Yes	2.19
0	tenure	1.97
7	InternetService_No	1.82
2	Contract_One year	1.65
5	PaymentMethod_Mailed check	1.57
4	PaymentMethod_Credit card (automatic)	1.39
1	SeniorCitizen	1.28

All variables have a good value of VIF. So we need not drop any more variables and we can proceed with making predictions using this model only

[271]:
```
# Let's take a look at the confusion matrix again
confusion = metrics.confusion_matrix(y_train_pred_final.Churn,␣
  ↪y_train_pred_final.predicted)
print(confusion)
```

```
[[3277  358]
 [ 614  673]]
```

[272]:
```
# cross checking the accuracy will help us understand the model if its changed␣
  ↪or not
print(metrics.accuracy_score(y_train_pred_final.Churn, y_train_pred_final.
  ↪predicted))
```

```
0.802519301097115
```

2 Actual/Predicted not_churn churn

```
# not_churn        3277       358
# churn             614       673
```

2.1 Metrics beyond simply accuracy

[273]:
```
TP = confusion[1,1] # true positive
TN = confusion[0,0] # true negatives
FP = confusion[0,1] # false positives
FN = confusion[1,0] # false negatives
```

[274]:
```
# Let's see the sensitivity of our logistic regression model
print("sensitivity =",  TP / float(TP+FN))

# Let us calculate specificity
print("specificity = ", TN / float(TN+FP))

# Calculate false postive rate - predicting churn when customer does not have␣
↪churned
print("False positive rate =", FP/float(TN+FP))

# positive predictive value
print("Positive predicted rate =", TP/float(TP+FP))

# Negative predictive value
print ("Negative predicted Rate =", TN / float(TN+ FN))
```

```
sensitivity = 0.5229215229215229
specificity =  0.9015130674002751
False positive rate = 0.0984869325997249
Positive predicted rate = 0.6527643064985451
Negative predicted Rate = 0.8421999485993318
```

2.1.1 Step 9: Plotting the ROC Curve

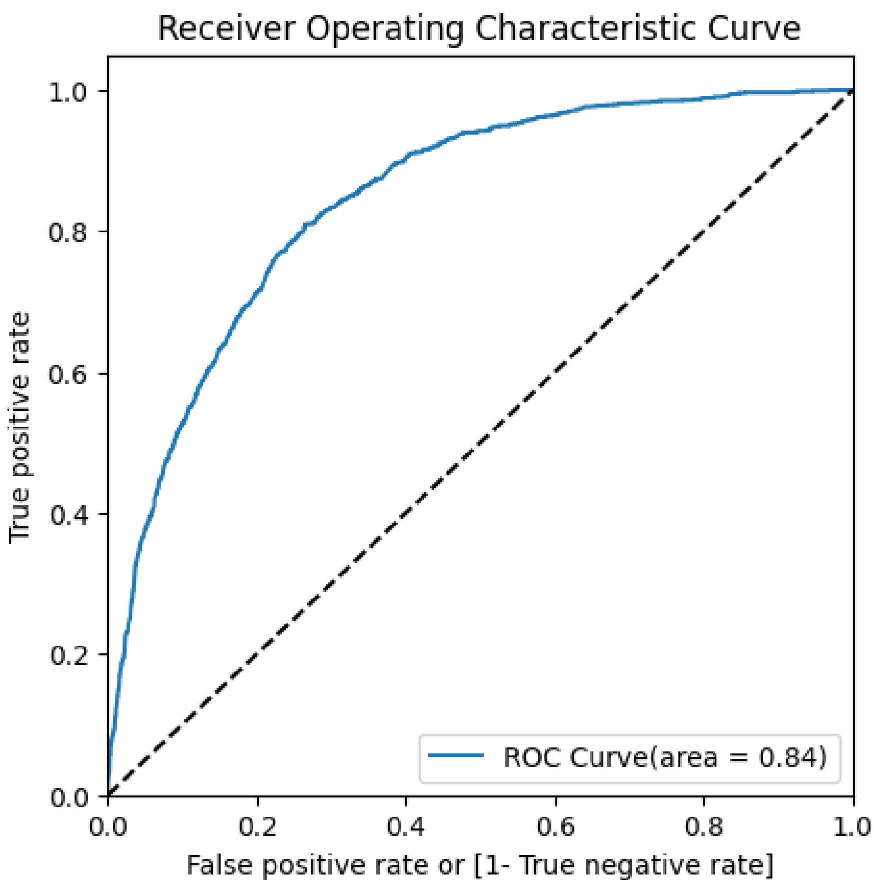
An ROC curve demonstrates several things:

- It shows the tradeoff between sensitivity and specificity (any increase in sensitivity will be accompanied by a decrease in specificity).
- The closer the curve follows the left-hand border and then the top border of the ROC space, the more accurate the test.
- The closer the curve comes to the 45-degree diagonal of the ROC space, the less accurate the test.

[275]:
```
# Plot the ROC Curve
def draw_roc(actual, probs):
  fpr, tpr , thresholds = metrics.roc_curve(actual, probs,␣
↪drop_intermediate=False)
  auc_score = metrics.roc_auc_score(actual, probs)
  plt.figure(figsize = (5,5))
  plt.plot(fpr, tpr, label = "ROC Curve(area = %0.2f)" % auc_score)
  plt.plot([0,1], [0,1], 'k--')
  plt.xlim([0.0, 1.0])
```

```
    plt.ylim([0.0, 1.05])
    plt.xlabel("False positive rate or [1- True negative rate]")
    plt.ylabel("True positive rate")
    plt.title("Receiver Operating Characteristic Curve")
    plt.legend(loc = "lower right")
    plt.show()
    return
```

[276]:
```
fpr, tpr, thresholds = metrics.roc_curve(y_train_pred_final.Churn,␣
 ↪y_train_pred_final.Churn_Prob, drop_intermediate = False)
draw_roc(y_train_pred_final.Churn, y_train_pred_final.Churn_Prob)
```



2.1.2 Step 10: Finding Optimal Cutoff Point

Optimal cutoff probability is that prob where we get balanced sensitivity and specificity

[277]:
```
# finding the optimal cut off point

numbers = [float(x)/10 for x in range(10)]
```

```
for i in numbers:
  y_train_pred_final[i] = y_train_pred_final.Churn_Prob.map(lambda x: 1 if x >␣
↪0.5 else 0)
y_train_pred_final.head()
```

[277]:

```
   Churn  Churn_Prob  CustID  predicted  0.0  0.1  0.2  0.3  0.4  0.5  0.6  \
0      0    0.204339     879          0    0    0    0    0    0    0    0
1      0    0.215562    5790          0    0    0    0    0    0    0    0
2      1    0.639615    6498          1    1    1    1    1    1    1    1
3      1    0.687136     880          1    1    1    1    1    1    1    1
4      1    0.735303    2784          1    1    1    1    1    1    1    1

   0.7  0.8  0.9
0    0    0    0
1    0    0    0
2    1    1    1
3    1    1    1
4    1    1    1
```

[299]:

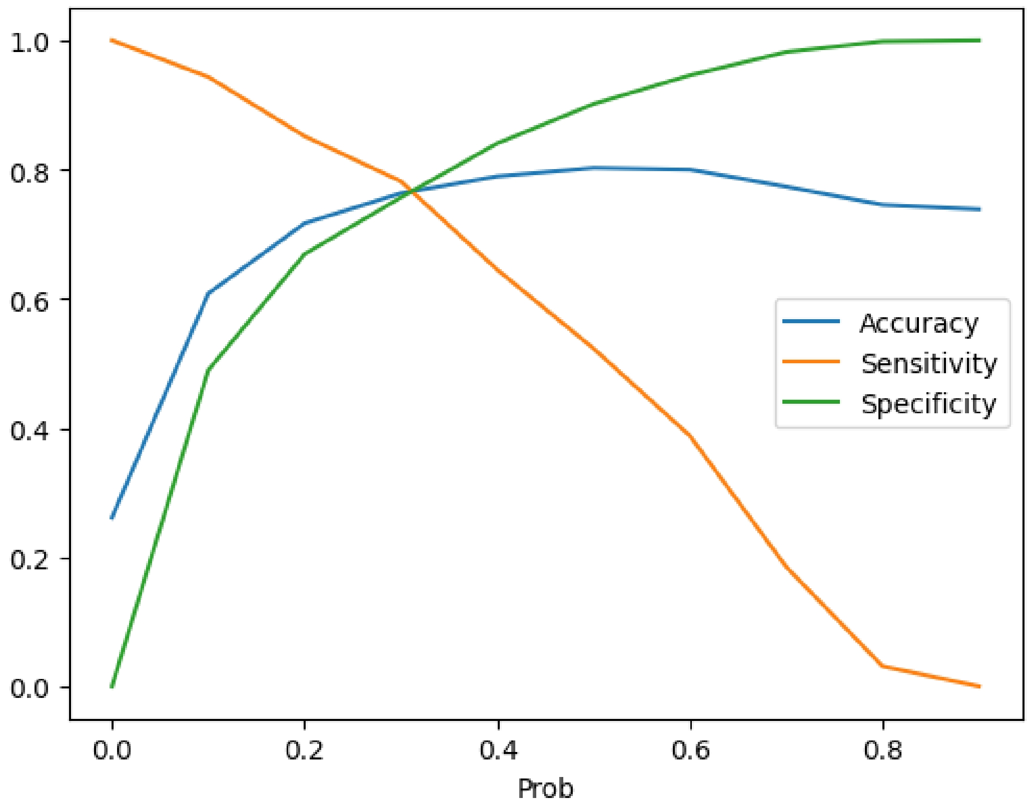
```
# Now let's calculate accuracy sensitivity and specificity for various␣
↪probability cutoffs.

cutoff_df = pd.DataFrame(columns = ['Prob', 'Accuracy','Sensitivity',␣
↪'Specificity'])
from sklearn.metrics import confusion_matrix



# TP = confusion[1,1] # true positive
# TN = confusion[0,0] # true negatives
# FP = confusion[0,1] # false positives
# FN = confusion[1,0] # false negatives

# List of probability cutoffs to evaluate
num = [0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9]

for i in num:
    # Create binary predictions based on the cutoff
    y_train_pred_final['Predicted_Label'] = y_train_pred_final['Churn_Prob'].
↪apply(lambda x: 1 if x >= i else 0)

# Compute the confusion matrix
    cm1 = confusion_matrix(y_train_pred_final['Churn'],␣
↪y_train_pred_final['Predicted_Label'])

# Calculate accuracy, sensitivity, and specificity
    total1 = sum(sum(cm1))
    Accuracy = (cm1[0,0] + cm1[1,1]) / total1
```

```
    Specificity = cm1[0,0] / (cm1[0,0] + cm1[0,1])
    Sensitivity = cm1[1,1] / (cm1[1,0] + cm1[1,1])

# Store the results in the DataFrame
    cutoff_df.loc[i] = [i, Accuracy, Sensitivity, Specificity]

print(cutoff_df)
```

```
     Prob  Accuracy  Sensitivity  Specificity
0.0   0.0  0.261479     1.000000     0.000000
0.1   0.1  0.608086     0.943279     0.489409
0.2   0.2  0.716782     0.851593     0.669051
0.3   0.3  0.763308     0.781663     0.756809
0.4   0.4  0.789313     0.644911     0.840440
0.5   0.5  0.802519     0.522922     0.901513
0.6   0.6  0.799878     0.387723     0.945805
0.7   0.7  0.773466     0.184926     0.981843
0.8   0.8  0.745429     0.031080     0.998349
0.9   0.9  0.738521     0.000000     1.000000
```

[300]:
```
# Let's plot accuracy sensitivity and specificity for various probabilities.

cutoff_df.plot.line(x = 'Prob', y = ['Accuracy', 'Sensitivity', 'Specificity'])
plt.show
```

[300]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



From the curve above, 0.3 is the optimum point to take it as a cutoff probability.

[302]:
```
y_train_pred_final['final_predicted'] = y_train_pred_final.Churn_Prob.map(␣
↪lambda x: 1 if x > 0.3 else 0)

y_train_pred_final.head()
```

[302]:

	Churn	Churn_Prob	CustID	predicted	0.0	0.1	0.2	0.3	0.4	0.5	0.6	\
0	0	0.204339	879	0	0	0	0	0	0	0	0	
1	0	0.215562	5790	0	0	0	0	0	0	0	0	
2	1	0.639615	6498	1	1	1	1	1	1	1	1	
3	1	0.687136	880	1	1	1	1	1	1	1	1	
4	1	0.735303	2784	1	1	1	1	1	1	1	1	

	0.7	0.8	0.9	Predicted_Label	final_predicted
0	0	0	0	0	0
1	0	0	0	0	0
2	1	1	1	0	1
3	1	1	1	0	1
4	1	1	1	0	1

[303]:
```
# Let's check the overall accuracy.
metrics.accuracy_score(y_train_pred_final.Churn, y_train_pred_final.
 ↪final_predicted)
```

[303]: 0.76330759853718

[304]:
```
confusion2 = metrics.confusion_matrix(y_train_pred_final.Churn,␣
 ↪y_train_pred_final.final_predicted )
confusion2
```

[304]:
```
array([[2751,  884],
       [ 281, 1006]])
```

[305]:
```
TP = confusion2[1,1] # true positive
TN = confusion2[0,0] # true negatives
FP = confusion2[0,1] # false positives
FN = confusion2[1,0] # false negatives
```

[306]:
```
# Let's see the sensitivity of our logistic regression model
TP / float(TP+FN)
```

[306]: 0.7816627816627817

[307]:
```
# Let us calculate specificity
TN / float(TN+FP)
```

[307]: 0.756808803301238

[308]:
```
# Calculate false postive rate - predicting churn when customer does not have␣
 ↪churned
print(FP/ float(TN+FP))
```

0.24319119669876205

[309]:
```
# Positive predictive value
print (TP / float(TP+FP))
```

0.5322751322751322

[310]:
```
# Negative predictive value
print (TN / float(TN+ FN))
```

0.9073218997361477

[311]:
```
# Preciosin and Recall
#Looking at the confusion matrix again
```

```
confusion = metrics.confusion_matrix(y_train_pred_final.Churn,␣
↪y_train_pred_final.predicted )
confusion
```

[311]:
```
array([[3277,  358],
       [ 614,  673]])
```

[281]:
```
print("Precision =", confusion[1,1]/(confusion[0,1]+confusion[1,1]))
```

```
Precision = 0.6527643064985451
```

[282]:
```
print("Recall =", confusion[1,1]/(confusion[1,0]+confusion[1,1]))
```

```
Recall = 0.5229215229215229
```

[283]:
```
from sklearn.metrics import precision_score, recall_score
```

[284]:
```
print("Precision =", precision_score(y_train_pred_final.Churn,␣
↪y_train_pred_final.predicted))
print("Recall =", recall_score(y_train_pred_final.Churn, y_train_pred_final.
↪predicted))
```

```
Precision = 0.6527643064985451
Recall = 0.5229215229215229
```

2.1.3 Precision and recall tradeoff

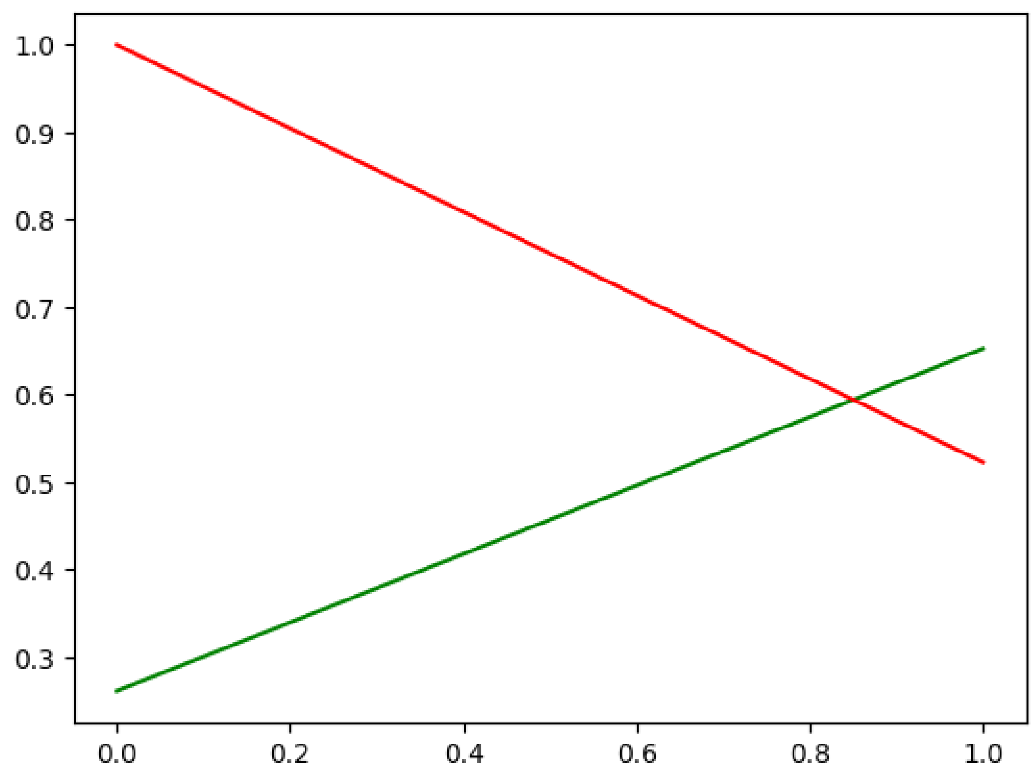
[312]:
```
from sklearn.metrics import precision_recall_curve
y_train_pred_final.Churn, y_train_pred_final.predicted
```

[312]:
```
(0       0
 1       0
 2       1
 3       1
 4       1
        ..
 4917    0
 4918    0
 4919    0
 4920    0
 4921    0
 Name: Churn, Length: 4922, dtype: int64,
 0       0
 1       0
 2       1
 3       1
 4       1
```

```
        ..
4917    0
4918    0
4919    0
4920    0
4921    0
Name: predicted, Length: 4922, dtype: int64)
```

[313]:
```
p , r , thresholds = precision_recall_curve(y_train_pred_final.Churn,␣
↪y_train_pred_final.predicted)
plt.plot(thresholds, p[:-1], "g-")
plt.plot(thresholds, r[:-1], "r-")
```

[313]:
```
[<matplotlib.lines.Line2D at 0x794f512d85e0>]
```



2.1.4 Step 11: Making predictions on the test set

[286]:
```
# test the data with y_test and X_test
X_test[['tenure','MonthlyCharges','TotalCharges']] = scaler.
↪transform(X_test[['tenure','MonthlyCharges','TotalCharges']])
```

[287]:
```
X_test = X_test[col]
X_test.head()
```

[287]:
```
        tenure  SeniorCitizen  Contract_One year  Contract_Two year  \
942  -0.347623              0              False              False
3730  0.999203              0              False              False
1761  1.040015              0              False               True
2283 -1.286319              0              False              False
1872  0.346196              0              False               True

      PaymentMethod_Credit card (automatic)  PaymentMethod_Mailed check  \
942                                    True                       False
3730                                   True                       False
1761                                   True                       False
2283                                  False                        True
1872                                  False                       False

      InternetService_Fiber optic  InternetService_No  MultipleLines_Yes  \
942                          True               False              False
3730                         True               False               True
1761                        False                True               True
2283                         True               False              False
1872                        False                True              False

      StreamingTV_Yes  StreamingMovies_Yes
942             False                 True
3730             True                 True
1761            False                False
2283            False                False
1872            False                False
```

[288]:
```
y_test.head()
```

[288]:
```
942     0
3730    1
1761    0
2283    1
1872    0
Name: Churn, dtype: int64
```

[289]:
```
X_test_sm.head()
```

[289]:
```
      const    tenure  SeniorCitizen  Contract_One year  Contract_Two year  \
942     1.0 -0.347623              0                  0                  0
3730    1.0  0.999203              0                  0                  0
1761    1.0  1.040015              0                  0                  1
2283    1.0 -1.286319              0                  0                  0
```

```
1872    1.0  0.346196             0                 0                 1

      PaymentMethod_Credit card (automatic)  PaymentMethod_Mailed check  \
942                                       1                           0
3730                                      1                           0
1761                                      1                           0
2283                                      0                           1
1872                                      0                           0

      InternetService_Fiber optic  InternetService_No  MultipleLines_Yes  \
942                             1                   0                  0
3730                            1                   0                  1
1761                            0                   1                  1
2283                            1                   0                  0
1872                            0                   1                  0

      StreamingTV_Yes  StreamingMovies_Yes
942                 0                    1
3730                1                    1
1761                0                    0
2283                0                    0
1872                0                    0
```

[290]:
```
X_test_sm.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 2110 entries, 942 to 4987
Data columns (total 12 columns):
 #   Column                                 Non-Null Count  Dtype
---  ------                                 --------------  -----
 0   const                                  2110 non-null   float64
 1   tenure                                 2110 non-null   float64
 2   SeniorCitizen                          2110 non-null   int64
 3   Contract_One year                      2110 non-null   int64
 4   Contract_Two year                      2110 non-null   int64
 5   PaymentMethod_Credit card (automatic)  2110 non-null   int64
 6   PaymentMethod_Mailed check             2110 non-null   int64
 7   InternetService_Fiber optic            2110 non-null   int64
 8   InternetService_No                     2110 non-null   int64
 9   MultipleLines_Yes                      2110 non-null   int64
 10  StreamingTV_Yes                        2110 non-null   int64
 11  StreamingMovies_Yes                    2110 non-null   int64
dtypes: float64(2), int64(10)
memory usage: 214.3 KB
```

[291]:
```
bool_columns = X_test.select_dtypes(include=['bool']).columns
X_test[bool_columns] = X_test[bool_columns].astype(int)
```

[292]:
```python
X_test_sm = sm.add_constant(X_test[col])
```

[293]:
```python
logm_test = sm.GLM(y_test, X_test_sm, family = sm.families.Binomial())
res = logm_test.fit()
print(res.summary())
```

```
                 Generalized Linear Model Regression Results
==============================================================================
Dep. Variable:                  Churn   No. Observations:                 2110
Model:                            GLM   Df Residuals:                     2098
Model Family:                Binomial   Df Model:                           11
Link Function:                  Logit   Scale:                          1.0000
Method:                          IRLS   Log-Likelihood:                -925.02
Date:                Wed, 07 Aug 2024   Deviance:                       1850.0
Time:                        03:42:27   Pearson chi2:                 1.98e+03
No. Iterations:                     7   Pseudo R-squ. (CS):             0.2600
Covariance Type:            nonrobust
================================================================================
=========================
                                           coef    std err          z
P>|z|      [0.025      0.975]
--------------------------------------------------------------------------------
-------------------------
const                                   -1.5693      0.162     -9.709
0.000      -1.886      -1.252
tenure                                  -0.8892      0.095     -9.410
0.000      -1.074      -0.704
SeniorCitizen                           -0.0548      0.149     -0.368
0.713      -0.346       0.237
Contract_One year                       -0.8690      0.190     -4.571
0.000      -1.242      -0.496
Contract_Two year                       -1.9106      0.310     -6.162
0.000      -2.518      -1.303
PaymentMethod_Credit card (automatic)   -0.2058      0.164     -1.252
0.211      -0.528       0.116
PaymentMethod_Mailed check              -0.1075      0.163     -0.661
0.509      -0.427       0.212
InternetService_Fiber optic              0.9304      0.141      6.580
0.000       0.653       1.208
InternetService_No                      -0.5142      0.213     -2.412
0.016      -0.932      -0.096
MultipleLines_Yes                        0.3264      0.136      2.405
0.016       0.060       0.592
StreamingTV_Yes                          0.1900      0.141      1.347
0.178      -0.087       0.467
StreamingMovies_Yes                      0.5022      0.142      3.526
0.000       0.223       0.781
```

```
==============================================================================
========================
```

```
[294]: y_test_pred = res.predict(X_test_sm)
```

```
[295]: y_test_pred[:10]
```

```
[295]: 942     0.491669
3730    0.328609
1761    0.008177
2283    0.598074
1872    0.013362
1970    0.617101
2532    0.235379
1616    0.007807
2485    0.588950
5914    0.214762
dtype: float64
```

```
[314]: # Converting y_pred to a dataframe which is an array
y_pred_1 = pd.DataFrame(y_test_pred)
y_pred_1
```

```
[314]:              0
942   0.491669
3730  0.328609
1761  0.008177
2283  0.598074
1872  0.013362
…          …
1289  0.038491
3508  0.052596
6765  0.006749
3598  0.382200
4987  0.006279

[2110 rows x 1 columns]
```

```
[316]: # Converting y_test to dataframe
y_test_df = pd.DataFrame(y_test)
```

```
[317]: # Putting CustID to index
y_test_df['CustID'] = y_test_df.index
```

```
[318]: # Removing index for both dataframes to append them side by side
y_pred_1.reset_index(drop=True, inplace=True)
y_test_df.reset_index(drop=True, inplace=True)
```

[319]:
```
# Appending y_test_df and y_pred_1
y_pred_final = pd.concat([y_test_df, y_pred_1],axis=1)
```

[320]:
```
y_pred_final.head()
```

[320]:
```
   Churn  CustID         0
0      0     942  0.491669
1      1    3730  0.328609
2      0    1761  0.008177
3      1    2283  0.598074
4      0    1872  0.013362
```

[321]:
```
# Renaming the column
y_pred_final= y_pred_final.rename(columns={ 0 : 'Churn_Prob'})
```

[324]:
```
# Rearranging the columns

y_pred_final = y_pred_final.reindex(columns=['CustID', 'Churn', 'Churn_Prob'])

# Let's see the head of y_pred_final
y_pred_final.head()
```

[324]:
```
   CustID  Churn  Churn_Prob
0     942      0    0.491669
1    3730      1    0.328609
2    1761      0    0.008177
3    2283      1    0.598074
4    1872      0    0.013362
```

[326]:
```
# Base don the precision and Recall trade off graph choosing cutoff as 0.82 ,␣
↪please use accordingly for your data
y_pred_final['final_predicted'] = y_pred_final.Churn_Prob.map(lambda x: 1 if x␣
↪> 0.82 else 0)
```

The choice of a cutoff value, such as 0.82, is often based on the specific context and goals of the classification problem. In binary classification, the probability cutoff determines the threshold above which a prediction is considered positive (e.g., churn) and below which it is considered negative (e.g., no churn). The default threshold is typically 0.5, but there are several reasons why you might choose a different threshold like 0.82:

Class Imbalance: If the classes are imbalanced (e.g., there are many more non-churners than churners), a lower threshold might help in identifying more positive cases, improving sensitivity/recall at the cost of specificity.

Cost-Benefit Analysis: The costs of false positives and false negatives might differ significantly. For example, if false negatives (not identifying a churner) are more costly than false positives (incorrectly identifying a non-churner as a churner), you might lower the threshold to reduce false negatives.

Optimization for Specific Metrics: Depending on the business requirements, you might optimize for metrics such as F1 score, sensitivity (recall), or precision. A threshold like 0.82 might have been found to optimize these metrics during model validation.

Receiver Operating Characteristic (ROC) Curve: The ROC curve plots the true positive rate (sensitivity) against the false positive rate (1-specificity) for various threshold values. The point closest to the top-left corner of the ROC space (where both sensitivity and specificity are high) can be chosen as the optimal threshold. This process might lead to a threshold like 0.82.

Precision-Recall Tradeoff: In some cases, particularly with imbalanced datasets, a Precision-Recall curve might be used to find the threshold that provides the best tradeoff between precision and recall.

[327]:
```
confusion2 = metrics.confusion_matrix(y_pred_final.Churn, y_pred_final.
 ↪final_predicted )
confusion2
```

[327]:
```
array([[1528,    0],
       [ 580,    2]])
```

[328]:
```
TP = confusion2[1,1] # true positive
TN = confusion2[0,0] # true negatives
FP = confusion2[0,1] # false positives
FN = confusion2[1,0] # false negatives
```

[329]:
```
# Let's see the sensitivity of our logistic regression model
TP / float(TP+FN)
```

[329]:
```
0.003436426116838488
```

[330]:
```
# Let us calculate specificity
TN / float(TN+FP)
```

[330]:
```
1.0
```

[ ]:
```

```