Objective:

Objective of this model is to predict whether the customer is prone to churn from the existing network to new network or not.

Project Plan :

Analyzing and Importing data:

Importing data and analyzing it to extract insights that supports for decision making process (like obtaining dependent and independent variables).

Data Cleansing:

-Check nulls (If null: Delete the row or replace with mean, median or mode) -Foreign values -Check voids -Wrong format data

Minimize the Dimension of Dataframe:

-Check correlation between columns -Pearson correlation is used to check correlation between two continuous column. -Highly correlated columns are obtained and one of them are dropped.

Encoding Data:

Categorical data is encoded (using label or one hot encoding based on the requirement) so that they're easily readable by the machines.

Model building:

-Splitting the data into train and test data sets -Train the model using decision tree classifier -Fit the data into the model -Predicting the response of the test dataset -Evaluating accuracy of the model

Optimizing the Model performance:

-Gini and Entropy algorithms are used to optimize the performance of the model developed. -Maximum depth is given for the decision tree to avoid overfitting and underfitting

Importing libraries

```
import pandas  as pd #Data manipulation
import numpy as np #Data manipulation
import matplotlib.pyplot as plt # Visualization
import seaborn as sns #Visualization
import plotly.express as px #Visualization
import plotly.graph_objs as go #Visualization
import os

## for statistical tests
```

```
import scipy
import statsmodels.formula.api as smf
import statsmodels.api as sm

## for machine learning
from sklearn import model_selection, preprocessing, feature_selection,
ensemble, linear_model, metrics, decomposition

## for explainer
#from lime import lime_tabular


pd.options.plotting.backend = "plotly"
plt.rcParams['figure.figsize'] = [8,5]
plt.rcParams['font.size'] =14
plt.rcParams['font.weight']= 'bold'
plt.style.use('seaborn-whitegrid')

df=pd.read_csv('Churn.csv')
df.head()

   c_AreaCode c_InternationalPlan    c_Phone c_State c_VMailPlan  \
0         415                  no   382-4657      KS         yes
1         415                  no   371-7191      OH         yes
2         415                  no   358-1921      NJ          no
3         408                 yes   375-9999      OH          no
4         415                 yes   330-6626      OK          no

   q_AccountLength  q_CustServCalls  q_DayCalls  q_DayCharge
q_DayMins  ...  \
0              128                1         110        45.07
265.1  ...
1              107                1         123        27.47
161.6  ...
2              137                0         114        41.38
243.4  ...
3               84                2          71        50.90
299.4  ...
4               75                3         113        28.34
166.7  ...

   q_EveCharge  q_EveMins  q_InternationalCharge  q_InternationalMins
\
0        16.78      197.4                   2.70                 10.0

1        16.62      195.5                   3.70                 13.7

2        10.30      121.2                   3.29                 12.2

3         5.26       61.9                   1.78                  6.6
```

```
4        12.61      148.3                       2.73                   10.1


   q_Internationalcalls  q_NightCalls  q_NightCharge  q_NightMins  \
0                     3            91          11.01        244.7
1                     3           103          11.45        254.4
2                     5           104           7.32        162.6
3                     7            89           8.86        196.9
4                     3           121           8.41        186.9

   q_VMailMessage  y_Churn
0              25   False.
1              26   False.
2               0   False.
3               0   False.
4               0   False.

[5 rows x 21 columns]
```

Data Cleansing

```
df.isnull().sum()
```

```
c_AreaCode              0
c_InternationalPlan     0
c_Phone                 0
c_State                 0
c_VMailPlan             0
q_AccountLength         0
q_CustServCalls         0
q_DayCalls              0
q_DayCharge             0
q_DayMins               0
q_EveCalls              0
q_EveCharge             0
q_EveMins               0
q_InternationalCharge   0
q_InternationalMins     0
q_Internationalcalls    0
q_NightCalls            0
q_NightCharge           0
q_NightMins             0
q_VMailMessage          0
y_Churn                 0
dtype: int64
```

View summary of data

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4617 entries, 0 to 4616
Data columns (total 21 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   c_AreaCode              4617 non-null   int64  
 1   c_InternationalPlan     4617 non-null   object 
 2   c_Phone                 4617 non-null   object 
 3   c_State                 4617 non-null   object 
 4   c_VMailPlan             4617 non-null   object 
 5   q_AccountLength         4617 non-null   int64  
 6   q_CustServCalls         4617 non-null   int64  
 7   q_DayCalls              4617 non-null   int64  
 8   q_DayCharge             4617 non-null   float64
 9   q_DayMins               4617 non-null   float64
 10  q_EveCalls              4617 non-null   int64  
 11  q_EveCharge             4617 non-null   float64
 12  q_EveMins               4617 non-null   float64
 13  q_InternationalCharge  4617 non-null   float64
 14  q_InternationalMins    4617 non-null   float64
 15  q_Internationalcalls   4617 non-null   int64  
 16  q_NightCalls            4617 non-null   int64  
 17  q_NightCharge           4617 non-null   float64
 18  q_NightMins             4617 non-null   float64
 19  q_VMailMessage          4617 non-null   int64  
 20  y_Churn                 4617 non-null   object 
dtypes: float64(8), int64(8), object(5)
memory usage: 757.6+ KB
```

```
v=df.groupby(['c_InternationalPlan'])['q_InternationalCharge'].mean()
v
```

```
c_InternationalPlan
 no     2.766931
 yes    2.860045
Name: q_InternationalCharge, dtype: float64
```

```
#Visualization
import matplotlib.pyplot as plt
x=['Subscribed','Unsubscribed']
y=[2.77,2.86]
a=plt.bar(x,y,width=0.4)
plt.xlabel('Subscription')
plt.ylabel('Charge/min')
plt.title('Comparision of charges for subscribed and unsubscribed 
customers for International calls')
a
```

```
<BarContainer object of 2 artists>
```

Inference :

From the plot it is observed that the charges for subscribed customer and unsubscribed customer varies for international calls.

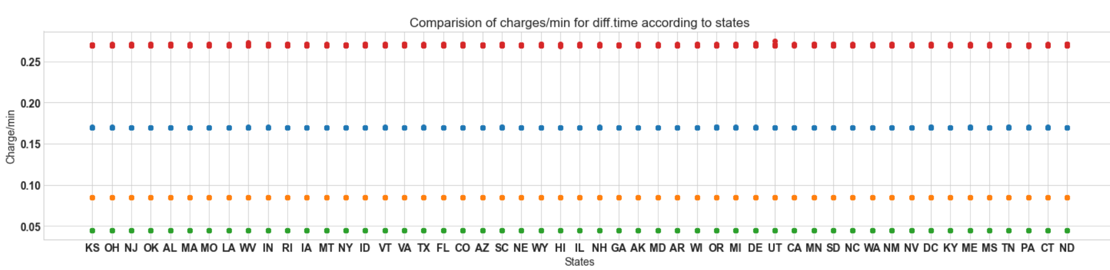
```
df['DC/min']=df['q_DayCharge']/df['q_DayMins']
df['EC/min']=df['q_EveCharge']/df['q_EveMins']
df['NC/min']=df['q_NightCharge']/df['q_NightMins']
df['IC/min']=df['q_InternationalCharge']/df['q_InternationalMins']
df

      c_AreaCode c_InternationalPlan    c_Phone c_State c_VMailPlan  \
0            415                  no   382-4657      KS         yes
1            415                  no   371-7191      OH         yes
2            415                  no   358-1921      NJ          no
3            408                 yes   375-9999      OH          no
4            415                 yes   330-6626      OK          no
...          ...                 ...        ...     ...         ...
4612         510                  no   345-7512      NY         yes
4613         408                  no   343-6820      NM         yes
4614         408                  no   338-4794      VT         yes
4615         415                  no   355-8388      MI         yes
4616         415                  no   409-6884      IN          no

      q_AccountLength  q_CustServCalls  q_DayCalls  q_DayCharge
q_DayMins  \
0                 128                1         110        45.07
265.1
1                 107                1         123        27.47
161.6
2                 137                0         114        41.38
243.4
3                  84                2          71        50.90
```

```
299.4
4                  75                3          113        28.34
166.7
...               ...              ...          ...          ...
...
4612               57                3           81        24.48
144.0
4613              177                3           91        32.13
189.0
4614               67                1          126        21.68
127.5
4615               98                0           98        28.71
168.9
4616              140                2          100        34.80
204.7

      ...  q_Internationalcalls  q_NightCalls  q_NightCharge
q_NightMins  \
0     ...                     3            91          11.01
244.7
1     ...                     3           103          11.45
254.4
2     ...                     5           104           7.32
162.6
3     ...                     7            89           8.86
196.9
4     ...                     3           121           8.41
186.9
...   ...                   ...           ...            ...
...
4612  ...                     6           122           7.14
158.6
4613  ...                     1           116           7.36
163.6
4614  ...                     3            91           9.04
200.9
4615  ...                     3            96           7.45
165.5
4616  ...                     4           115           9.13
202.8

      q_VMailMessage  y_Churn    DC/min    EC/min    NC/min    IC/min

0                 25   False.  0.170011  0.085005  0.044994  0.270000

1                 26   False.  0.169988  0.085013  0.045008  0.270073

2                  0   False.  0.170008  0.084983  0.045018  0.269672

3                  0   False.  0.170007  0.084976  0.044997  0.269697
```

```
4                    0   False.  0.170006  0.085030  0.044997  0.270297
...                ...      ...       ...       ...       ...       ...
4612                25   False.  0.170000  0.084989  0.045019  0.270588
4613                29   False.  0.170000  0.084988  0.044988  0.270064
4614                33   False.  0.170039  0.085005  0.044998  0.270000
4615                23   False.  0.169982  0.085020  0.045015  0.269930
4616                 0   False.  0.170005  0.085016  0.045020  0.270248

[4617 rows x 25 columns]
```

```python
import matplotlib.pyplot as plt
fig=plt.figure()
fig.set_size_inches(25,5)
x=df['c_State']
y1=df['DC/min']
y2=df['EC/min']
y3=df['NC/min']
y4=df['IC/min']
a=plt.scatter(x,y1)
b=plt.scatter(x,y2)
c=plt.scatter(x,y3)
d=plt.scatter(x,y4)
plt.xlabel('States')
plt.ylabel('Charge/min')
plt.title('Comparision of charges/min for diff.time according to states')
a
```

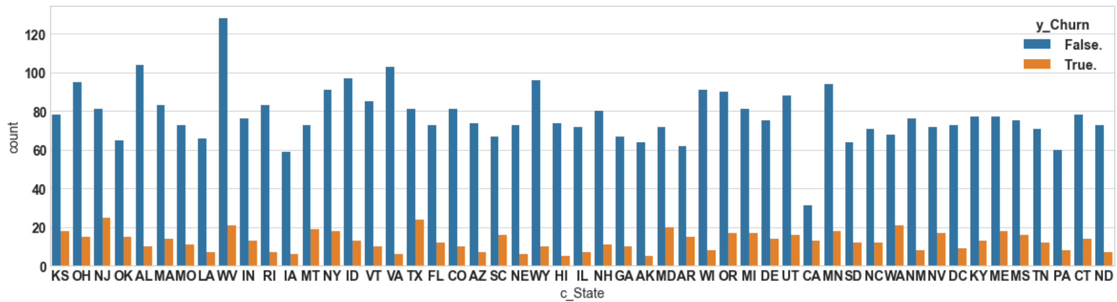
```
<matplotlib.collections.PathCollection at 0x1ffc2aead30>
```



Inference :

From the above plot it is observed that the charges doesn't vary much from state to state. So the c_State column can be dropped

```
plt.figure(figsize=(20,5))
ax = sns.countplot(x='c_State', hue="y_Churn", data=df)
```




```
max_acl=df['q_AccountLength'].max()
min_acl=df['q_AccountLength'].min()
print("The maximum account length :",max_acl)
print("The minimum account length :",min_acl)
```

```
The maximum account length : 243
The minimum account length : 1
```

```
df.iloc[817]
```

```
c_AreaCode                      510
c_InternationalPlan              no
c_Phone                    355-9360
c_State                          UT
c_VMailPlan                      no
q_AccountLength                 243
q_CustServCalls                   2
q_DayCalls                       92
q_DayCharge                   16.24
q_DayMins                      95.5
q_EveCalls                       63
q_EveCharge                   13.91
q_EveMins                     163.7
q_InternationalCharge          1.78
q_InternationalMins             6.6
q_Internationalcalls              6
q_NightCalls                    118
q_NightCharge                 11.89
q_NightMins                   264.2
q_VMailMessage                    0
y_Churn                      False.
DC/min                     0.170052
EC/min                     0.084973
NC/min                     0.045004
```

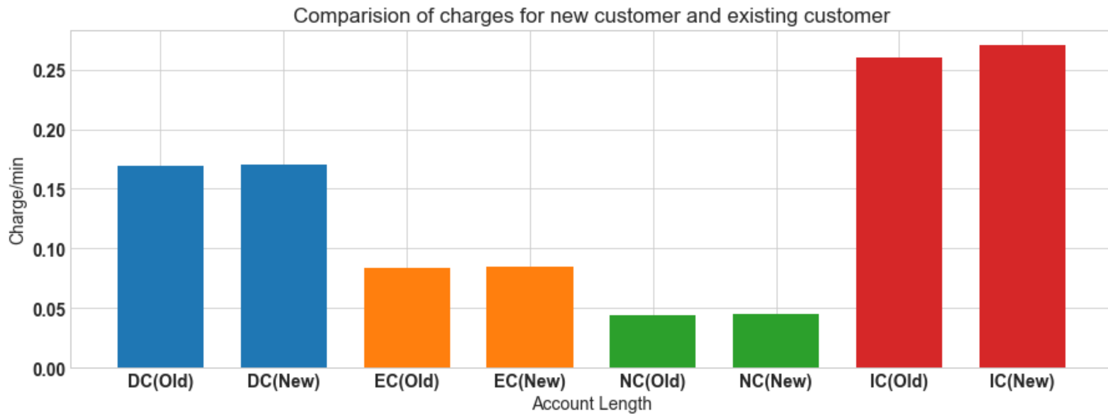
```
IC/min                       0.269697
Name: 817, dtype: object
```

```python
df.iloc[245]
```

```
c_AreaCode                        408
c_InternationalPlan                no
c_Phone                      331-5138
c_State                            SC
c_VMailPlan                        no
q_AccountLength                    22
q_CustServCalls                     0
q_DayCalls                        107
q_DayCharge                     18.75
q_DayMins                       110.3
q_EveCalls                         93
q_EveCharge                     14.15
q_EveMins                       166.5
q_InternationalCharge            2.57
q_InternationalMins               9.5
q_Internationalcalls                5
q_NightCalls                       96
q_NightCharge                     9.1
q_NightMins                     202.3
q_VMailMessage                      0
y_Churn                        False.
DC/min                       0.169991
EC/min                       0.084985
NC/min                       0.044983
IC/min                       0.270526
Name: 245, dtype: object
```

```python
import matplotlib.pyplot as plt
fig=plt.figure()
fig.set_size_inches(15,5)
x1=['DC(Old)','DC(New)']
y1=[0.169,0.170]
x2=['EC(Old)','EC(New)']
y2=[0.084,0.085]
x3=['NC(Old)','NC(New)']
y3=[0.044,0.045]
x4=['IC(Old)','IC(New)']
y4=[0.26,0.27]
plt.bar(x1,y1,width=0.7)
plt.bar(x2,y2,width=0.7)
plt.bar(x3,y3,width=0.7)
plt.bar(x4,y4,width=0.7)
plt.xlabel('Account Length')
plt.ylabel('Charge/min')
plt.title('Comparision of charges for new customer and existing
```

```
customer')
plt.show()
```



Inference:

The charges for low account length customer and high account length customer remains the same. So account length column can be dropped.

```
statechurncount=df.groupby("c_State").y_Churn .value_counts()
statechurncount
```

```
c_State  y_Churn
AK        False.     64
          True.       5
AL        False.    104
          True.      10
AR        False.     62
                   ...
WI        True.       8
WV        False.    128
          True.      21
WY        False.     96
          True.      10
Name: y_Churn, Length: 102, dtype: int64
```

Explore class variable

```
df['y_Churn'].value_counts()
```

```
 False.    3961
 True.      656
Name: y_Churn, dtype: int64
```

The y_Chrunis the target variable which is ordinal in nature.

```
ip_churn=df.groupby("c_InternationalPlan").y_Churn .value_counts()
ip_churn
```

```
c_InternationalPlan  y_Churn
 no                   False.    3701
```

```
                      True.      470
 yes                  False.     260
                      True.      186
Name: y_Churn, dtype: int64
```

Correlation

```
correlation=df.corr().abs()
correlation
```

```
                       c_AreaCode  q_AccountLength  q_CustServCalls  \
c_AreaCode               1.000000         0.020394         0.021046   
q_AccountLength          0.020394         1.000000         0.002620   
q_CustServCalls          0.021046         0.002620         1.000000   
q_DayCalls               0.013179         0.032783         0.008747   
q_DayCharge              0.018903         0.001999         0.008155   
q_DayMins                0.018900         0.002002         0.008149   
q_EveCalls               0.011528         0.015598         0.007730   
q_EveCharge              0.011533         0.006775         0.015611   
q_EveMins                0.011513         0.006778         0.015598   
q_InternationalCharge    0.007386         0.003501         0.016148   
q_InternationalMins      0.007292         0.003483         0.016079   
q_Internationalcalls     0.011531         0.023485         0.016778   
q_NightCalls             0.015316         0.009482         0.010258   
q_NightCharge            0.002782         0.002095         0.013868   
q_NightMins              0.002794         0.002077         0.013871   
q_VMailMessage           0.002597         0.012983         0.006951   
DC/min                   0.014331         0.001995         0.018327   
EC/min                   0.023385         0.014497         0.014045   
NC/min                   0.004466         0.020796         0.001603   
IC/min                   0.014661         0.011580         0.029188   

                       q_DayCalls  q_DayCharge  q_DayMins  q_EveCalls  
\
c_AreaCode               0.013179     0.018903   0.018900    0.011528   

q_AccountLength          0.032783     0.001999   0.002002    0.015598   

q_CustServCalls          0.008747     0.008155   0.008149    0.007730   

q_DayCalls               1.000000     0.002821   0.002823    0.003923   

q_DayCharge              0.002821     1.000000   1.000000    0.012992   

q_DayMins                0.002823     1.000000   1.000000    0.012990   

q_EveCalls               0.003923     0.012992   0.012990    1.000000   

q_EveCharge              0.006429     0.010262   0.010268    0.001151   

q_EveMins                0.006430     0.010255   0.010260    0.001135   
```

q_InternationalCharge	0.013055	0.012262	0.012261	0.002831
q_InternationalMins	0.012951	0.012315	0.012314	0.002798
q_Internationalcalls	0.010889	0.000163	0.000166	0.005198
q_NightCalls	0.013299	0.005164	0.005165	0.015463
q_NightCharge	0.010724	0.009593	0.009591	0.002624
q_NightMins	0.010730	0.009606	0.009604	0.002610
q_VMailMessage	0.003846	0.009025	0.009028	0.006508
DC/min	0.005004	0.045677	0.045904	0.005130
EC/min	0.009264	0.011786	0.011783	0.022399
NC/min	0.006709	0.013106	0.013099	0.010932
IC/min	0.008493	0.015359	0.015362	0.000118

	q_EveCharge	q_EveMins	q_InternationalCharge \
c_AreaCode	0.011533	0.011513	0.007386
q_AccountLength	0.006775	0.006778	0.003501
q_CustServCalls	0.015611	0.015598	0.016148
q_DayCalls	0.006429	0.006430	0.013055
q_DayCharge	0.010262	0.010255	0.012262
q_DayMins	0.010268	0.010260	0.012261
q_EveCalls	0.001151	0.001135	0.002831
q_EveCharge	1.000000	1.000000	0.000170
q_EveMins	1.000000	1.000000	0.000172
q_InternationalCharge	0.000170	0.000172	1.000000
q_InternationalMins	0.000163	0.000165	0.999993

q_Internationalcalls	0.005751	0.005749	0.021548
q_NightCalls	0.014231	0.014226	0.006439
q_NightCharge	0.018846	0.018837	0.004497
q_NightMins	0.018839	0.018830	0.004500
q_VMailMessage	0.017878	0.017872	0.005202
DC/min	0.017246	0.017242	0.008715
EC/min	0.035968	0.036596	0.006301
NC/min	0.000705	0.000701	0.008879
IC/min	0.000293	0.000285	0.087671

	q_InternationalMins	q_Internationalcalls	\
c_AreaCode	0.007292	0.011531	
q_AccountLength	0.003483	0.023485	
q_CustServCalls	0.016079	0.016778	
q_DayCalls	0.012951	0.010889	
q_DayCharge	0.012315	0.000163	
q_DayMins	0.012314	0.000166	
q_EveCalls	0.002798	0.005198	
q_EveCharge	0.000163	0.005751	
q_EveMins	0.000165	0.005749	
q_InternationalCharge	0.999993	0.021548	
q_InternationalMins	1.000000	0.021431	
q_Internationalcalls	0.021431	1.000000	
q_NightCalls	0.006431	0.003294	
q_NightCharge	0.004546	0.014624	
q_NightMins	0.004549	0.014651	
q_VMailMessage	0.005167	0.007423	
DC/min	0.008690	0.011105	
EC/min	0.006301	0.005035	
NC/min	0.008852	0.026916	
IC/min	0.091300	0.027672	

	q_NightCalls	q_NightCharge	q_NightMins	\
c_AreaCode	0.015316	0.002782	0.002794	
q_AccountLength	0.009482	0.002095	0.002077	
q_CustServCalls	0.010258	0.013868	0.013871	
q_DayCalls	0.013299	0.010724	0.010730	
q_DayCharge	0.005164	0.009593	0.009606	
q_DayMins	0.005165	0.009591	0.009604	
q_EveCalls	0.015463	0.002624	0.002610	

q_EveCharge	0.014231	0.018846	0.018839
q_EveMins	0.014226	0.018837	0.018830
q_InternationalCharge	0.006439	0.004497	0.004500
q_InternationalMins	0.006431	0.004546	0.004549
q_Internationalcalls	0.003294	0.014624	0.014651
q_NightCalls	1.000000	0.025722	0.025742
q_NightCharge	0.025722	1.000000	0.999999
q_NightMins	0.025742	0.999999	1.000000
q_VMailMessage	0.000889	0.004665	0.004672
DC/min	0.028335	0.001849	0.001825
EC/min	0.007975	0.017241	0.017251
NC/min	0.004684	0.004907	0.003729
IC/min	0.001188	0.013655	0.013657

	q_VMailMessage	DC/min	EC/min	NC/min	IC/min
c_AreaCode	0.002597	0.014331	0.023385	0.004466	0.014661
q_AccountLength	0.012983	0.001995	0.014497	0.020796	0.011580
q_CustServCalls	0.006951	0.018327	0.014045	0.001603	0.029188
q_DayCalls	0.003846	0.005004	0.009264	0.006709	0.008493
q_DayCharge	0.009025	0.045677	0.011786	0.013106	0.015359
q_DayMins	0.009028	0.045904	0.011783	0.013099	0.015362
q_EveCalls	0.006508	0.005130	0.022399	0.010932	0.000118
q_EveCharge	0.017878	0.017246	0.035968	0.000705	0.000293
q_EveMins	0.017872	0.017242	0.036596	0.000701	0.000285
q_InternationalCharge	0.005202	0.008715	0.006301	0.008879	0.087671
q_InternationalMins	0.005167	0.008690	0.006301	0.008852	0.091300
q_Internationalcalls	0.007423	0.011105	0.005035	0.026916	0.027672
q_NightCalls	0.000889	0.028335	0.007975	0.004684	0.001188
q_NightCharge	0.004665	0.001849	0.017241	0.004907	0.013655
q_NightMins	0.004672	0.001825	0.017251	0.003729	0.013657
q_VMailMessage	1.000000	0.026199	0.010810	0.006164	0.006811
DC/min	0.026199	1.000000	0.000187	0.020774	0.005734

```
EC/min                         0.010810  0.000187  1.000000  0.000557
0.010020
NC/min                         0.006164  0.020774  0.000557  1.000000
0.010549
IC/min                         0.006811  0.005734  0.010020  0.010549
1.000000
```

```
#To find highly correlated continuos columns (Pearson Correlation is
used for cont-cont columns)
hcorr_features=set()
x=0.95
for i in range(len(correlation.columns)):
    for j in range(i):
        if(correlation.iloc[i,j])>x:
            colname=correlation.columns[i]
            hcorr_features.add(colname)
print("Highly co-related continuos columns",hcorr_features)
```

```
Highly co-related continuos columns {'q_DayMins', 'q_EveMins',
'q_InternationalMins', 'q_NightMins'}
```

Droping the highly correleated values from visulaization

```
df.drop('q_NightMins',axis=1,inplace=True)
df.drop('q_EveMins',axis=1,inplace=True)
df.drop('q_DayMins',axis=1,inplace=True)
df.drop('q_InternationalMins',axis=1,inplace=True)
df.drop('c_VMailPlan',axis=1,inplace=True) #This can be dropped
because there are charges for subscribed customer and zero for
unsubscribed customer
df.drop('c_AreaCode',axis=1,inplace=True)
df.drop('c_Phone',axis=1,inplace=True)
df.drop('q_AccountLength',axis=1,inplace=True)
df.drop('c_State',axis=1,inplace=True)
df.drop('q_CustServCalls',axis=1,inplace=True)
df.drop('DC/min',axis=1,inplace=True)
df.drop('EC/min',axis=1,inplace=True)
df.drop('NC/min',axis=1,inplace=True)
df.drop('IC/min',axis=1,inplace=True)
```

```
df
```

```
     c_InternationalPlan  q_DayCalls  q_DayCharge  q_EveCalls
q_EveCharge  \
0                     no         110        45.07          99
16.78
1                     no         123        27.47         103
16.62
2                     no         114        41.38         110
10.30
3                    yes          71        50.90          88
5.26
```

```
4                    yes         113        28.34        122
12.61
...                  ...         ...          ...        ...
...
4612                  no          81        24.48        112
15.91
4613                  no          91        32.13         96
25.76
4614                  no         126        21.68        129
25.17
4615                  no          98        28.71        117
19.24
4616                  no         100        34.80        107
10.78

      q_InternationalCharge  q_Internationalcalls  q_NightCalls  \
0                      2.70                     3            91
1                      3.70                     3           103
2                      3.29                     5           104
3                      1.78                     7            89
4                      2.73                     3           121
...                     ...                   ...           ...
4612                   2.30                     6           122
4613                   4.24                     1           116
4614                   3.51                     3            91
4615                   3.86                     3            96
4616                   3.27                     4           115

      q_NightCharge  q_VMailMessage  y_Churn
0             11.01              25   False.
1             11.45              26   False.
2              7.32               0   False.
3              8.86               0   False.
4              8.41               0   False.
...             ...             ...      ...
4612           7.14              25   False.
4613           7.36              29   False.
4614           9.04              33   False.
4615           7.45              23   False.
4616           9.13               0   False.

[4617 rows x 11 columns]
```

Encoding

```
from sklearn.preprocessing import LabelEncoder
l_enc = LabelEncoder()
for i in (0,1,1):
    df.iloc[:,i] = l_enc.fit_transform(df.iloc[:,i])
df
```

```
      c_InternationalPlan  q_DayCalls  q_DayCharge  q_EveCalls
q_EveCharge  \
0                       0          74        45.07          99
16.78
1                       0          87        27.47         103
16.62
2                       0          78        41.38         110
10.30
3                       1          35        50.90          88
5.26
4                       1          77        28.34         122
12.61
...                   ...         ...          ...         ...
...
4612                    0          45        24.48         112
15.91
4613                    0          55        32.13          96
25.76
4614                    0          90        21.68         129
25.17
4615                    0          62        28.71         117
19.24
4616                    0          64        34.80         107
10.78

      q_InternationalCharge  q_Internationalcalls  q_NightCalls  \
0                      2.70                     3            91
1                      3.70                     3           103
2                      3.29                     5           104
3                      1.78                     7            89
4                      2.73                     3           121
...                     ...                   ...           ...
4612                   2.30                     6           122
4613                   4.24                     1           116
4614                   3.51                     3            91
4615                   3.86                     3            96
4616                   3.27                     4           115

      q_NightCharge  q_VMailMessage  y_Churn
0             11.01              25   False.
1             11.45              26   False.
2              7.32               0   False.
3              8.86               0   False.
4              8.41               0   False.
...             ...             ...      ...
4612           7.14              25   False.
4613           7.36              29   False.
4614           9.04              33   False.
4615           7.45              23   False.
4616           9.13               0   False.
```

```
[4617 rows x 11 columns]
```

```
y = df.y_Churn
X = df.iloc[:,:-1]
```

Splitting datas into train and test datas

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test =
train_test_split(X,y,test_size=0.2,random_state=1)
```

Feature engineering

- checking the data types of the variable after splitting

```
# check data types in X_train

X_train.dtypes
```

```
c_InternationalPlan        int32
q_DayCalls                 int64
q_DayCharge              float64
q_EveCalls                 int64
q_EveCharge              float64
q_InternationalCharge    float64
q_Internationalcalls       int64
q_NightCalls               int64
q_NightCharge            float64
q_VMailMessage             int64
dtype: object
```

Train the model using DecisionTreeClassifier

```
from sklearn.tree import DecisionTreeClassifier
clf = DecisionTreeClassifier()
clf = clf.fit(X_train,y_train)
y_pred = clf.predict(X_test)
```

Evaluating the accuracy of the model

```
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.8560606060606061
```

Optimizing the performance of the decision tree model

- Criteria choosen is "entropy" and "gini"
- max_depth is given as 3 for our model

Decision Tree Classifier with criterion entropy index

```
# Create Decision Tree classifer object
clf_en = DecisionTreeClassifier(criterion="entropy", max_depth=3)

# Train Decision Tree Classifer
```

```
clf_en = clf_en.fit(X_train,y_train)

#Predict the response for test dataset
y_pred = clf_en.predict(X_test)

# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.9036796536796536
```

Here, y_test are the true class labels and y_pred are the predicted class labels in the test-set.

Compare the train-set and test-set accuracy

Now, we are comparing the train-set and test-set accuracy to check for overfitting.

```
y_pred_train_entropy = clf_en.predict(X_train)
```

```
y_pred_train_entropy
```

```
array([' False.', ' False.', ' False.', ..., ' False.', ' False.',
       ' False.'], dtype=object)
```

```
from sklearn.metrics import accuracy_score
print('Training-set accuracy score: {0:0.4f}'.
format(accuracy_score(y_train, y_pred_train_entropy)))
```

```
Training-set accuracy score: 0.8993
```

Check for overfitting and underfitting

```
print('Training set score: {:.4f}'.format(clf_en.score(X_train,
y_train)))

print('Test set score: {:.4f}'.format(clf_en.score(X_test, y_test)))
```

```
Training set score: 0.8993
Test set score: 0.9037
```

Here,the training-set accuracy score is 0.8985 while the test-set accuracy to be 0.9048. These two values are quite comparable. So, there is no sign of overfitting.

Decision Tree Classifier with criterion gini index

```
# Create Decision Tree classifer object
clf = DecisionTreeClassifier(criterion="gini", max_depth=3)

# Train Decision Tree Classifer
clf = clf.fit(X_train,y_train)

#Predict the response for test dataset
y_pred = clf.predict(X_test)
```

```
# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.9047619047619048
```

Compare the train-set and test-set accuracy

Now, we are comparing the train-set and test-set accuracy to check for overfitting.

```
y_pred_train_gini = clf.predict(X_train)
```

```
y_pred_train_gini
```

```
array([' False.', ' False.', ' False.', ..., ' False.', ' False.',
       ' False.'], dtype=object)
```

```
from sklearn.metrics import accuracy_score
print('Training-set accuracy score: {0:0.4f}'.
format(accuracy_score(y_train, y_pred_train_gini)))
```

```
Training-set accuracy score: 0.8985
```

Check for overfitting and underfitting

```
print('Training set score: {:.4f}'.format(clf.score(X_train,
y_train)))

print('Test set score: {:.4f}'.format(clf.score(X_test, y_test)))
```

```
Training set score: 0.8985
Test set score: 0.9048
```

We can see that the training-set score and test-set score is same as above. The training-set accuracy score is 0.8985 while the test-set accuracy to be 0.9048. These two values are quite comparable. So, there is no sign of overfitting.

Now, based on the above analysis we can conclude that our classification model accuracy is very good.

But, it does not give the underlying distribution of values. Also, it does not tell anything about the type of errors our classifer is making.

We have another tool called Confusion matrix that comes to our rescue.

Summary:

-In this project, we have build a Decision-Tree Classifier model to predict the churn rate. -We build two models, one with criterion gini index and another one with criterion entropy. The model yields a very good performance as indicated by the model accuracy in both the cases which was found to be 0.90. -In the model with criterion gini index, the training-set accuracy score is 0.8985 while the test-set accuracy to be 0.9048. These two values are quite comparable. So, there is no sign of overfitting. -Similarly, in the model with criterion entropy, the training-set accuracy score is 0.8993 while the test-set accuracy to be 0.9037.We get the same values as in the case with criterion gini. So, there is no sign of overfitting.