Churn Reduction

Muddassir Mohammed

24 May 2018

CONTENTS

Chapter 1

Introduction

1.1 Problem Statement:

Churn (loss of customers to competition) is a problem for companies because it is more expensive to acquire a new customer than to keep your existing one from leaving. The objective of this case is to predict customer behavior. Given are customer usage pattern and if the customer has moved or not. This is a classification problem in which we have to develop an algorithm to predict the churn score based on usage pattern.

1.2 Data Used:

Here we are using two datasets train and test datasets.
Train_data: It has data to be trained. It has 3333 observations
Test_data: It has data to be tested. It has 1667 observations
Variables:
There are 21 variables.20 dependent and one target variable.
Each variable describes customer usage pattern

Dependent Variables:

State: State of customer
Account length: Number of days he is using company services
Area code: Pin code of his area
Phone number: Phone of customer
International plan: whether he is having international plan or not

Voice mail plan: A voicemail system (also known as voice message or voice bank) is a computer-based system that allows users and subscribers to exchange personal voice. Whether he is applicable to voice mail plan or not
Number vmail messages: Number of voice mails sent
Total day minutes: Number of minutes spent during day
Total day calls: Number of day calls made
Total day charge: Charge for day calls
Total eve minutes: Number of minutes spent in evening
Total eve calls: Number of evening calls made
Total eve charge: Charge for evening calls
Total night minutes: Number of minutes spent at night
Total night calls: Number of night calls made
Total night charge: Charge for night calls
Total intl minutes: Number of minutes spent on international calls
Total intl calls: Number of international calls
Total intl charge: International calls charge
Number customer service calls: No of Customer service calls made

Target Variable:

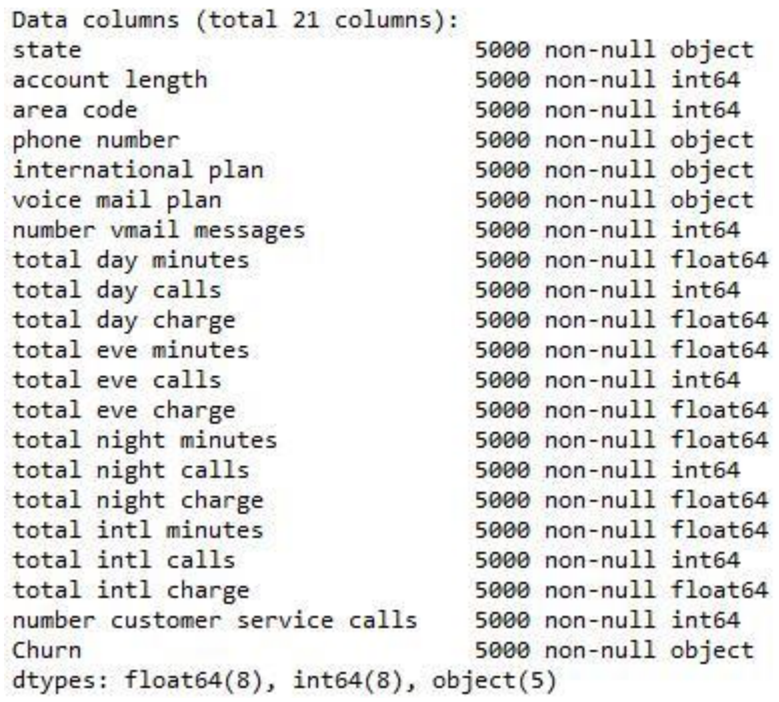
Churn: This is our target variable. Depending upon above dependent variables,

We have to predict whether the customer is going to be churn from the company or not.

Chapter 2

Exploratory Data Analysis

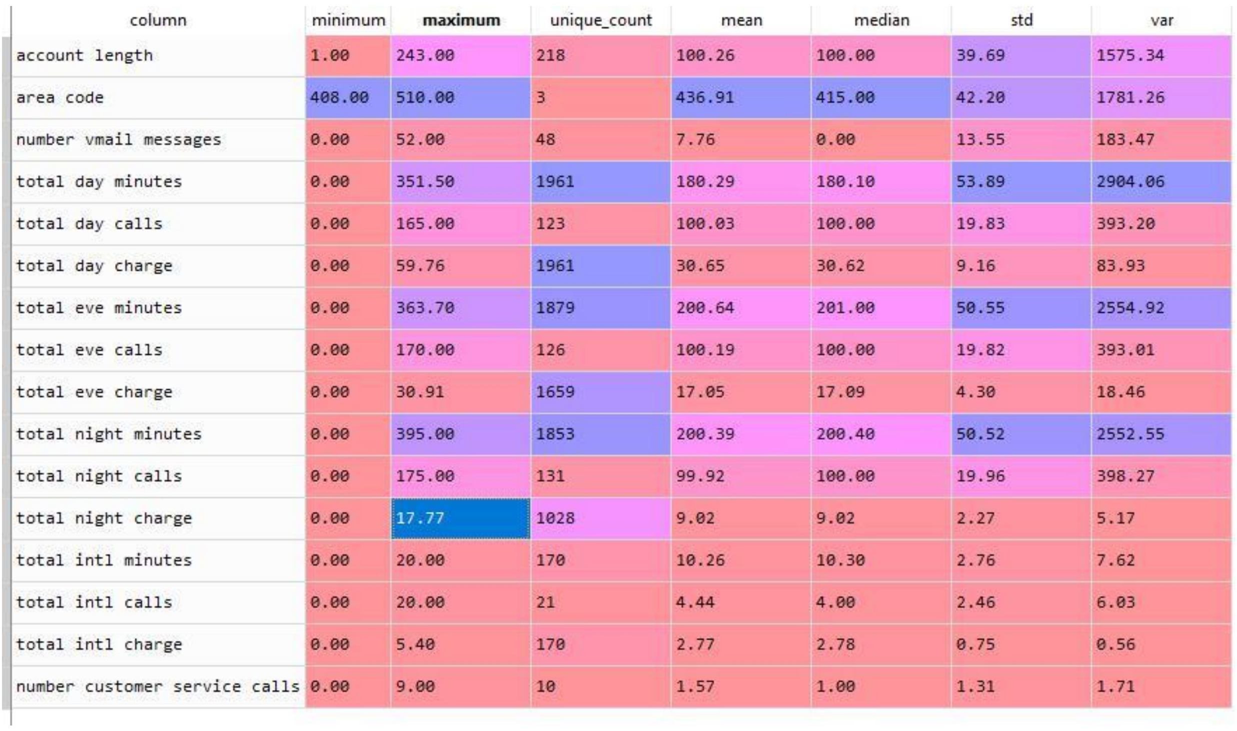
Out of 21 variables. There are 16 numerical variables and 5 categorical variables. Overall test and training dataset consists of 5000 observations.



```
Data columns (total 21 columns):
state                             5000 non-null object
account length                    5000 non-null int64
area code                         5000 non-null int64
phone number                      5000 non-null object
international plan                5000 non-null object
voice mail plan                   5000 non-null object
number vmail messages             5000 non-null int64
total day minutes                 5000 non-null float64
total day calls                   5000 non-null int64
total day charge                  5000 non-null float64
total eve minutes                 5000 non-null float64
total eve calls                   5000 non-null int64
total eve charge                  5000 non-null float64
total night minutes               5000 non-null float64
total night calls                 5000 non-null int64
total night charge                5000 non-null float64
total intl minutes                5000 non-null float64
total intl calls                  5000 non-null int64
total intl charge                 5000 non-null float64
number customer service calls     5000 non-null int64
Churn                             5000 non-null object
dtypes: float64(8), int64(8), object(5)
```

2.1 Exploration of Numerical Variable:

For every column, we have calculated mean, median, standard deviation, maximum, minimum, unique value counts

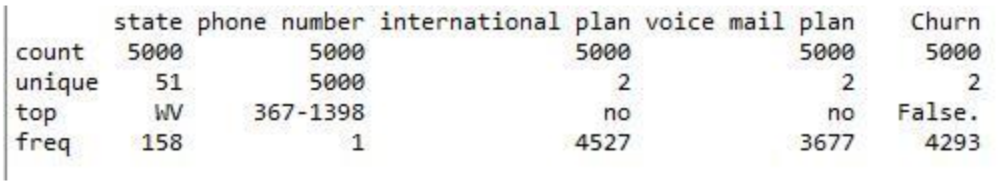


column	minimum	maximum	unique_count	mean	median	std	var
account length	1.00	243.00	218	100.26	100.00	39.69	1575.34
area code	408.00	510.00	3	436.91	415.00	42.20	1781.26
number vmail messages	0.00	52.00	48	7.76	0.00	13.55	183.47
total day minutes	0.00	351.50	1961	180.29	180.10	53.89	2904.06
total day calls	0.00	165.00	123	100.03	100.00	19.83	393.20
total day charge	0.00	59.76	1961	30.65	30.62	9.16	83.93
total eve minutes	0.00	363.70	1879	200.64	201.00	50.55	2554.92
total eve calls	0.00	170.00	126	100.19	100.00	19.82	393.01
total eve charge	0.00	30.91	1659	17.05	17.09	4.30	18.46
total night minutes	0.00	395.00	1853	200.39	200.40	50.52	2552.55
total night calls	0.00	175.00	131	99.92	100.00	19.96	398.27
total night charge	0.00	17.77	1028	9.02	9.02	2.27	5.17
total intl minutes	0.00	20.00	170	10.26	10.30	2.76	7.62
total intl calls	0.00	20.00	21	4.44	4.00	2.46	6.03
total intl charge	0.00	5.40	170	2.77	2.78	0.75	0.56
number customer service calls	0.00	9.00	10	1.57	1.00	1.31	1.71

Important Insights:

- We can see that maximum number of morning, evening and night calls are almost equal, But nights charges<evening charges<morning charges
- We can see that there are not much customer service calls, customers are only from three areas
- We can see that total intl charge column has less variance of 0.56 and total day minutes has high variance of 2904
- Standard deviation of total intl charge is less and for total day minutes it is high.

2.2 Exploration of categorical variable:



```
       state phone number international plan voice mail plan    Churn
count   5000         5000               5000            5000     5000
unique    51         5000                  2               2        2
top       WV     367-1398                 no              no   False.
freq     158            1               4527            3677     4293
```

Important Insights:

- We can see that there are 5 categorical variables.
- Customers are from 51 different states. Most of them are from WV
- There are no duplicates in the data as there are 5000 observations with identified by Phone number.
- Also we can see that 73% of customers has no voice mail plans,90% of customers have no international plans

2.3 Target Class Exploration:

Total observations: 5000

“True”: 707 = 14.14%

“False”: 4293 = 85.86%

Train:

Total: 3333
True: 483 = 14.49%
False: 2850 = 85.50%

Test:

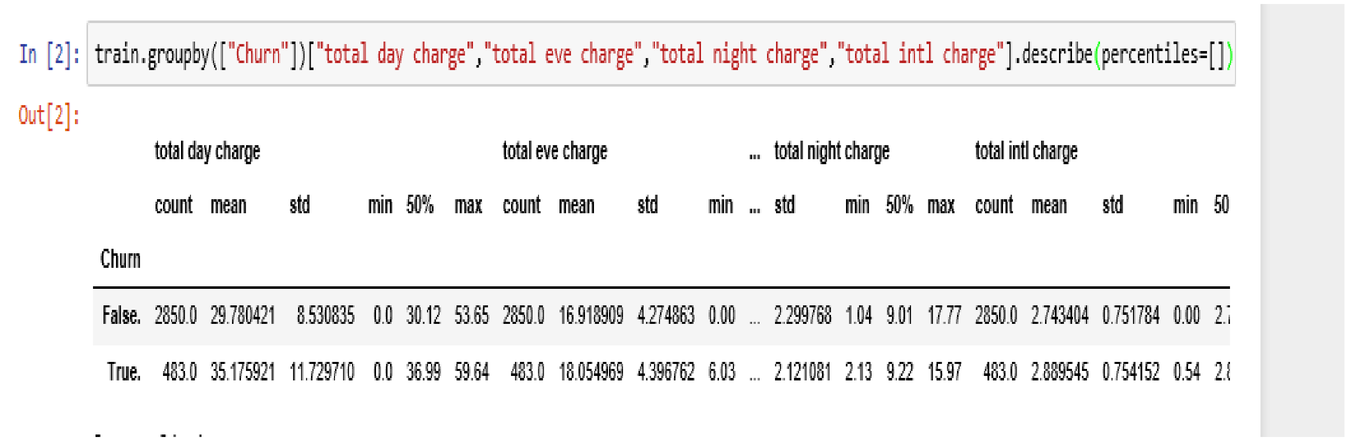
Total: 1667
True: 224 = 13.4
False: 1443 = 86.56%

We can see this dataset has Target class imbalance problem.

85% of customers are not going to churn out which means they are loyal to the company.
We can see that for training dataset churn rate is 14.5%.
14.5% is actually quite bad for a company; such a churn rate can make the company go bankrupt.

Let us answer some basic questions:

Customers may be churning out because of high charges



```
In [2]: train.groupby(["Churn"])["total day charge","total eve charge","total night charge","total intl charge"].describe(percentiles=[])
Out[2]:
```

	total day charge						total eve charge				...	total night charge				total intl charge				
	count	mean	std	min	50%	max	count	mean	std	min	...	std	min	50%	max	count	mean	std	min	50
Churn																				
False.	2850.0	29.780421	8.530835	0.0	30.12	53.65	2850.0	16.918909	4.274863	0.00	...	2.299768	1.04	9.01	17.77	2850.0	2.743404	0.751784	0.00	2.7
True.	483.0	35.175921	11.729710	0.0	36.99	59.64	483.0	18.054969	4.396762	6.03	...	2.121081	2.13	9.22	15.97	483.0	2.889545	0.754152	0.54	2.8

How much (on average) churned users charged on phone during daytime?

```
In [78]: train[train["Churn"]==" True."]["total day minutes"].mean()
Out[78]: 206.91407867494814
```

How much (on average) users (not Churned) charged on phone during daytime?

```
In [79]: train[train["Churn"]==" False."]["total day minutes"].mean()
Out[79]: 175.17575438596492
```

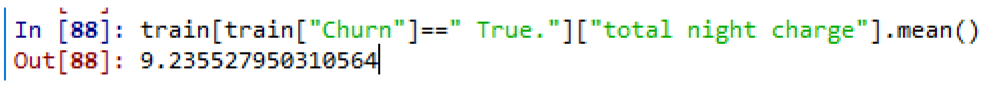
How much (on average) users charged on phone during evening time?
Churned:

```
In [86]: train[train["Churn"]==" True."]["total eve charge"].mean()
Out[86]: 18.054968944099382
```

Not Churned:

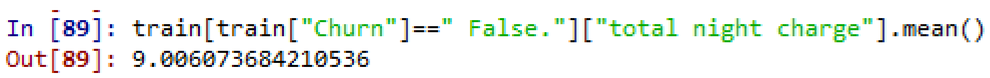
```
In [87]: train[train["Churn"]==" False."]["total eve charge"].mean()
Out[87]: 16.918908771929825
```

How much (on average) users charged on phone during night time?
Churned:



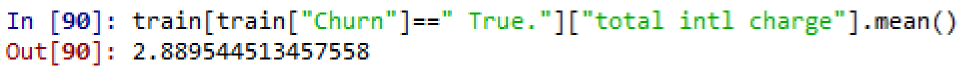
```
In [88]: train[train["Churn"]==" True."]["total night charge"].mean()
Out[88]: 9.235527950310564
```

Not Churned:



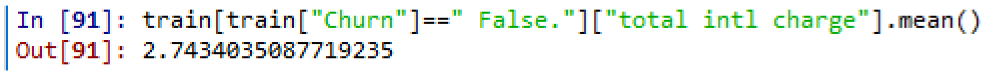
```
In [89]: train[train["Churn"]==" False."]["total night charge"].mean()
Out[89]: 9.006073684210536
```

How much (on average) users charged for international calls?
Churned:



```
In [90]: train[train["Churn"]==" True."]["total intl charge"].mean()
Out[90]: 2.889544513457558
```

Not Churned:



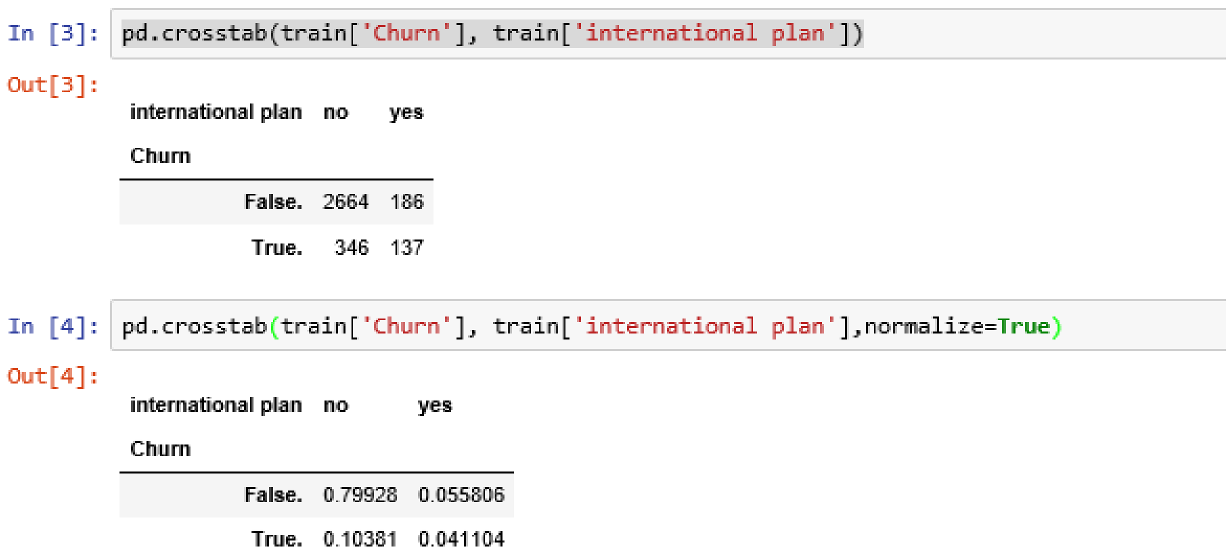
```
In [91]: train[train["Churn"]==" False."]["total intl charge"].mean()
Out[91]: 2.7434035087719235
```

Churn Customers Average cost= day + evening + night + international = 237.07
Loyal (Not Churn) Customers Average Cost= 203.82

We can see that users who are charged more are churning out.
Hence company can introduce some offers like free talktime or less cost during night to retain the customers from churning out

Do Churned Customers use Additional Services:

Let us construct contingency table for churn and international plan



```
In [3]: pd.crosstab(train['Churn'], train['international plan'])
Out[3]:
```

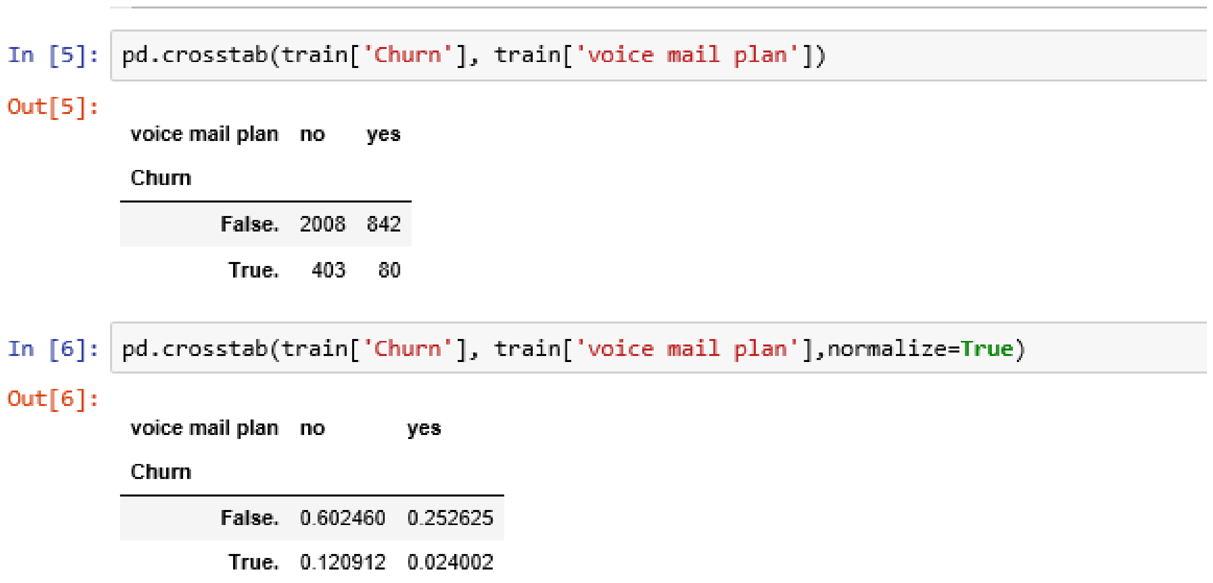
international plan	no	yes
Churn		
False.	2664	186
True.	346	137

```
In [4]: pd.crosstab(train['Churn'], train['international plan'],normalize=True)
Out[4]:
```

international plan	no	yes
Churn		
False.	0.79928	0.055806
True.	0.10381	0.041104

We can see that 2664(79%) persons do not churn out are not using international plan

Let us construct contingency table for churn and voice mail plan



```
In [5]: pd.crosstab(train['Churn'], train['voice mail plan'])
Out[5]:
```

voice mail plan	no	yes
Churn		
False.	2008	842
True.	403	80

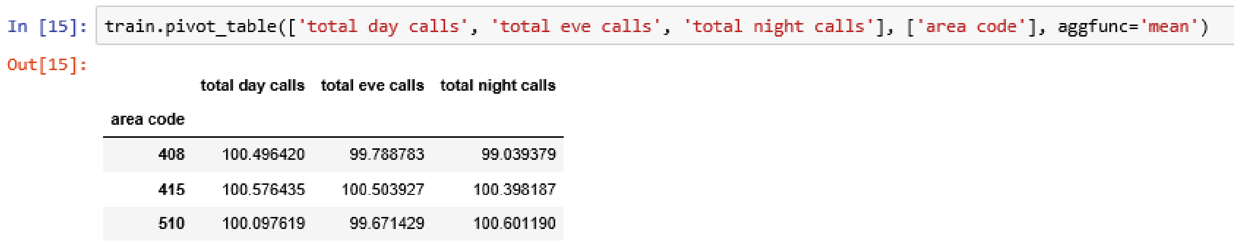
```
In [6]: pd.crosstab(train['Churn'], train['voice mail plan'],normalize=True)
Out[6]:
```

voice mail plan	no	yes
Churn		
False.	0.602460	0.252625
True.	0.120912	0.024002

We can see that 2008 customers(60%) do not churn out,are not using international plan
From this contingency tables,we can say that,most of the loyal customers are not using additional services.

Does Area Code affects customer usage:

Let us construct pivot table to undestand the relation between area code and no of calls made by customer



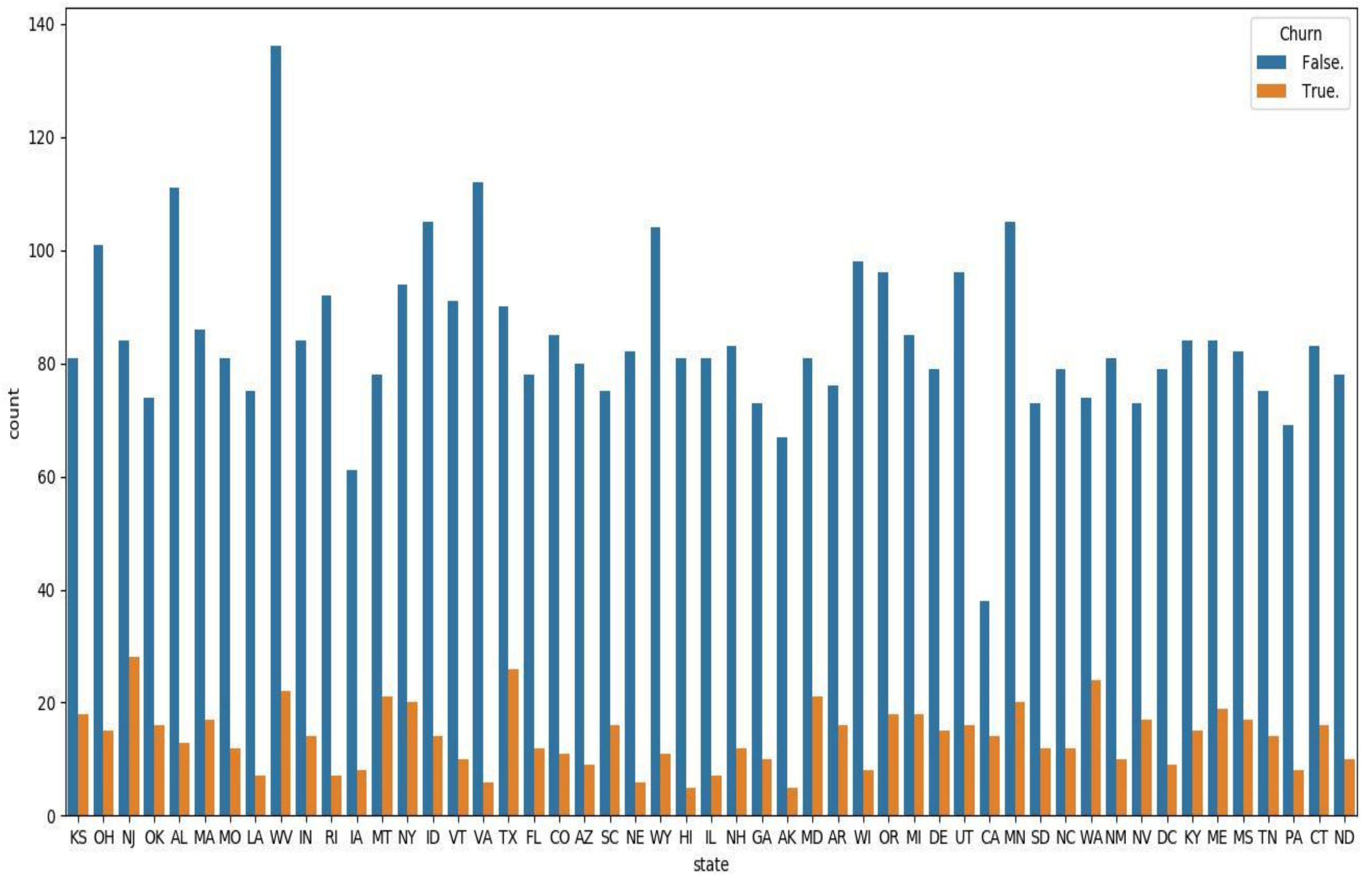
```
In [15]: train.pivot_table(['total day calls', 'total eve calls', 'total night calls'], ['area code'], aggfunc='mean')
Out[15]:
```

area code	total day calls	total eve calls	total night calls
408	100.496420	99.788783	99.039379
415	100.576435	100.503927	100.398187
510	100.097619	99.671429	100.601190

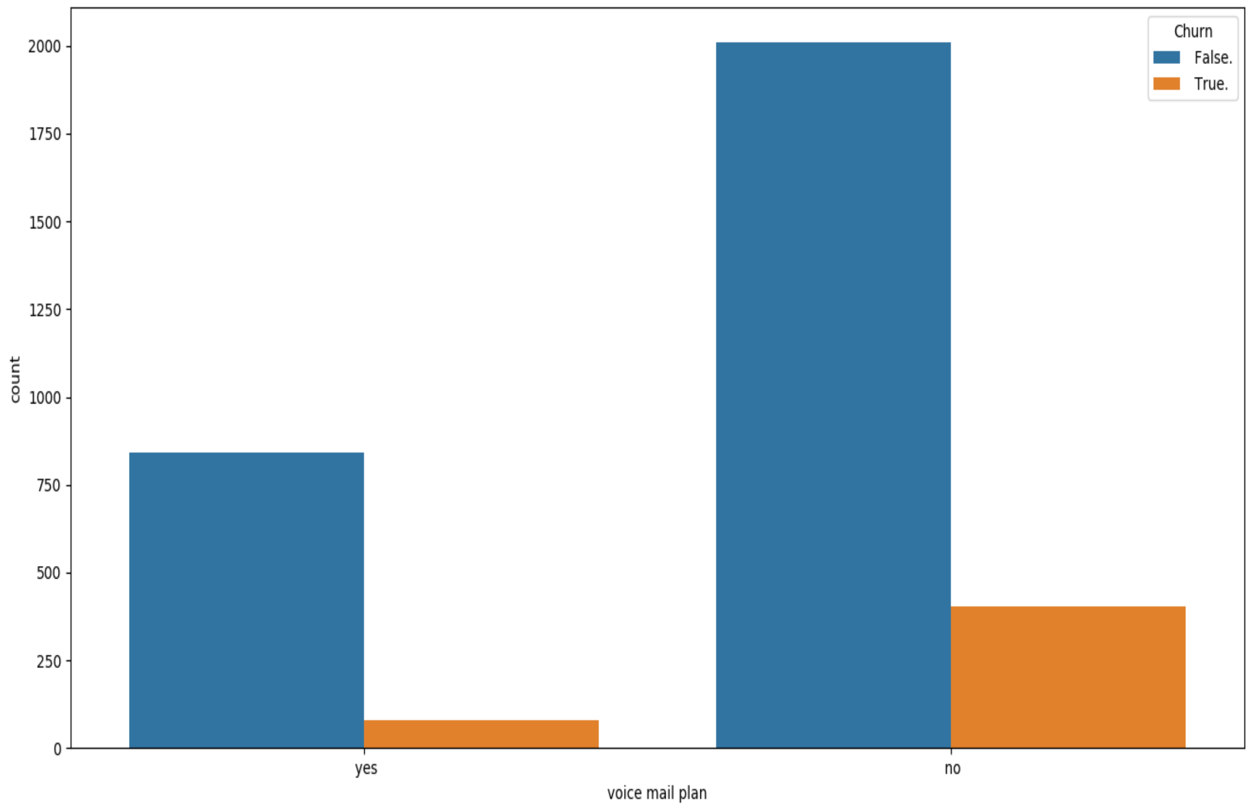
We can see that all usage is almost same in all three areas

Let us visualize the data to understand the data more

Distribution of States and Churn

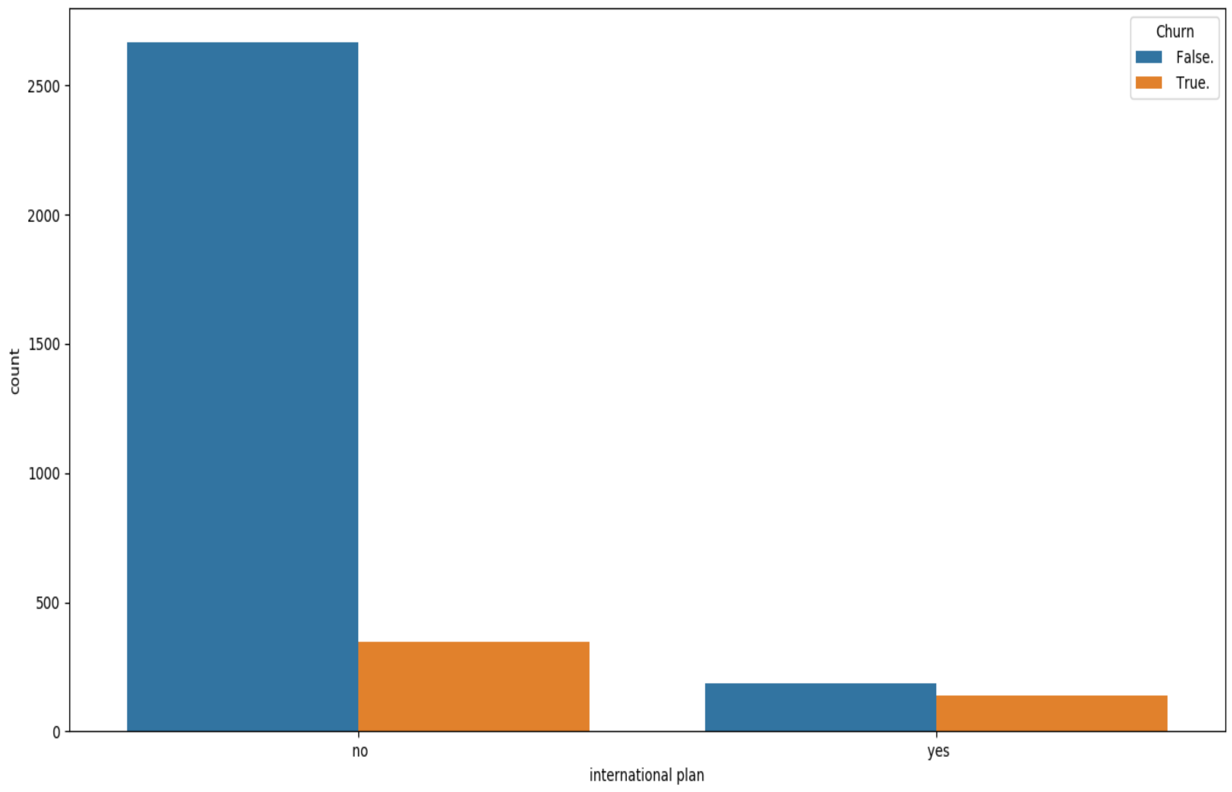


Distribution of Voice mail Plan and Churn:



We can see that there are more customers who doesn't have voice mail plan and they are not going to churn out

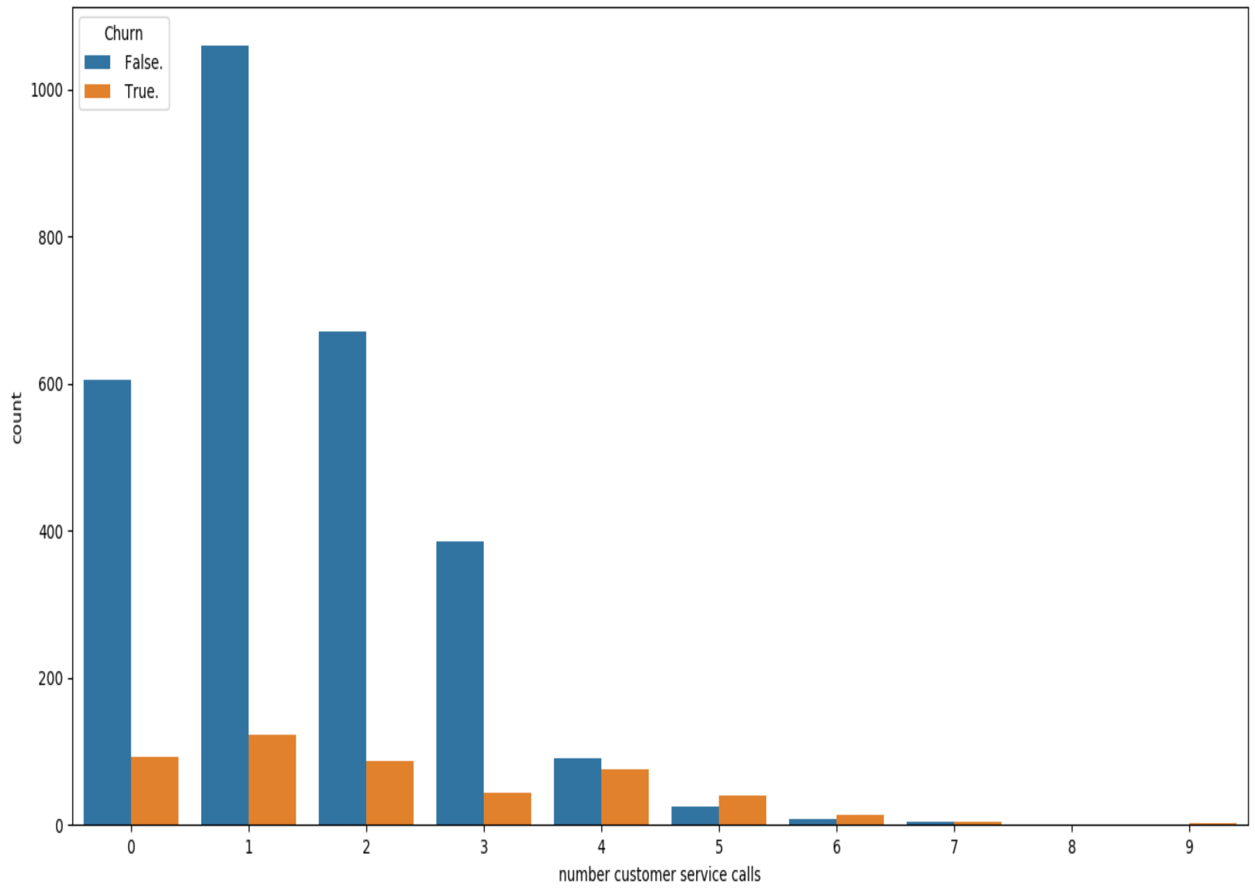
Distribution of International plan and Churn:



There are more customers who don’t have international plan are not going to be churned out.

We see that, with *International Plan*, the churn rate is much higher, which is an interesting observation! Perhaps large and poorly controlled expenses with international calls are very conflict-prone and lead to dissatisfaction among the telecom operator’s customers.

Distribution of customer service calls and churn:



Picture clearly states that the churn rate strongly increases starting from 4 calls to the service center.

Important Insights:

- If (Customer Service calls > 3) & (International plan = True) & (voice mail plan=True), there is high chance that customer is churning out
- Customers who spent more minutes and who are charged more have higher chances of churning out

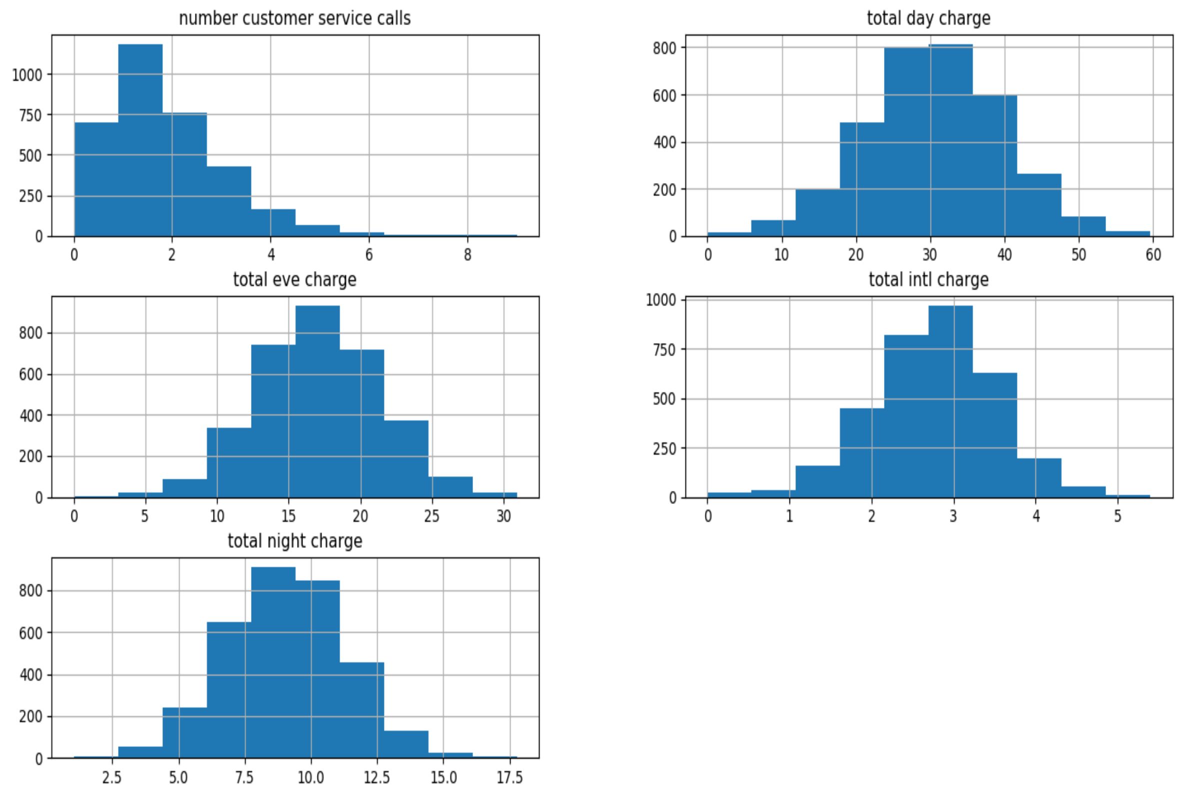
2.4 Plots of important variables:

Histograms:

It is important to look at the distribution of the numerical variables

Code:

```
features=["total day charge","total eve charge","total night charge","total intl charge","number customer service calls"]
train[features].hist()
```

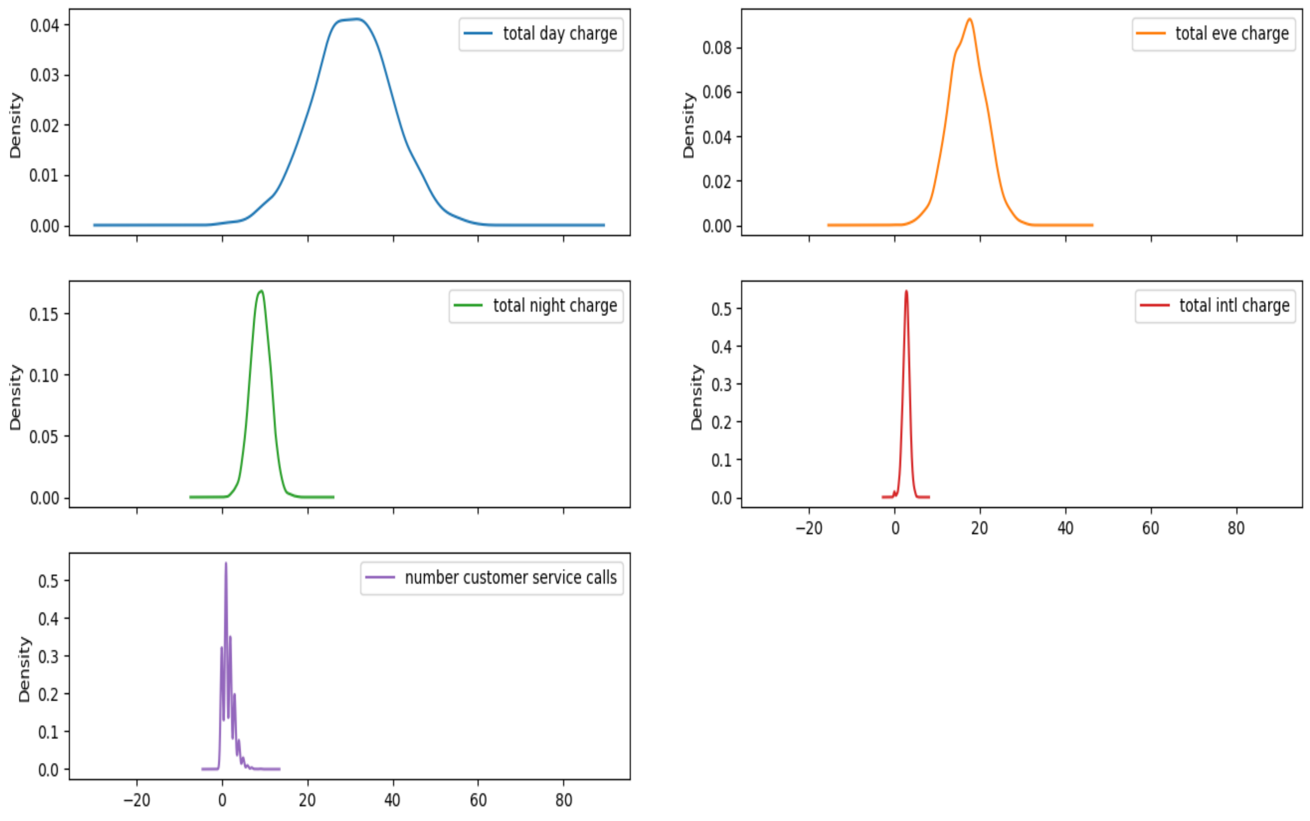


Kernel Density plots:

These are smoothed version of histograms

Code:

```
train[features].plot(kind="density",subplots=True,layout=(3,2))
```



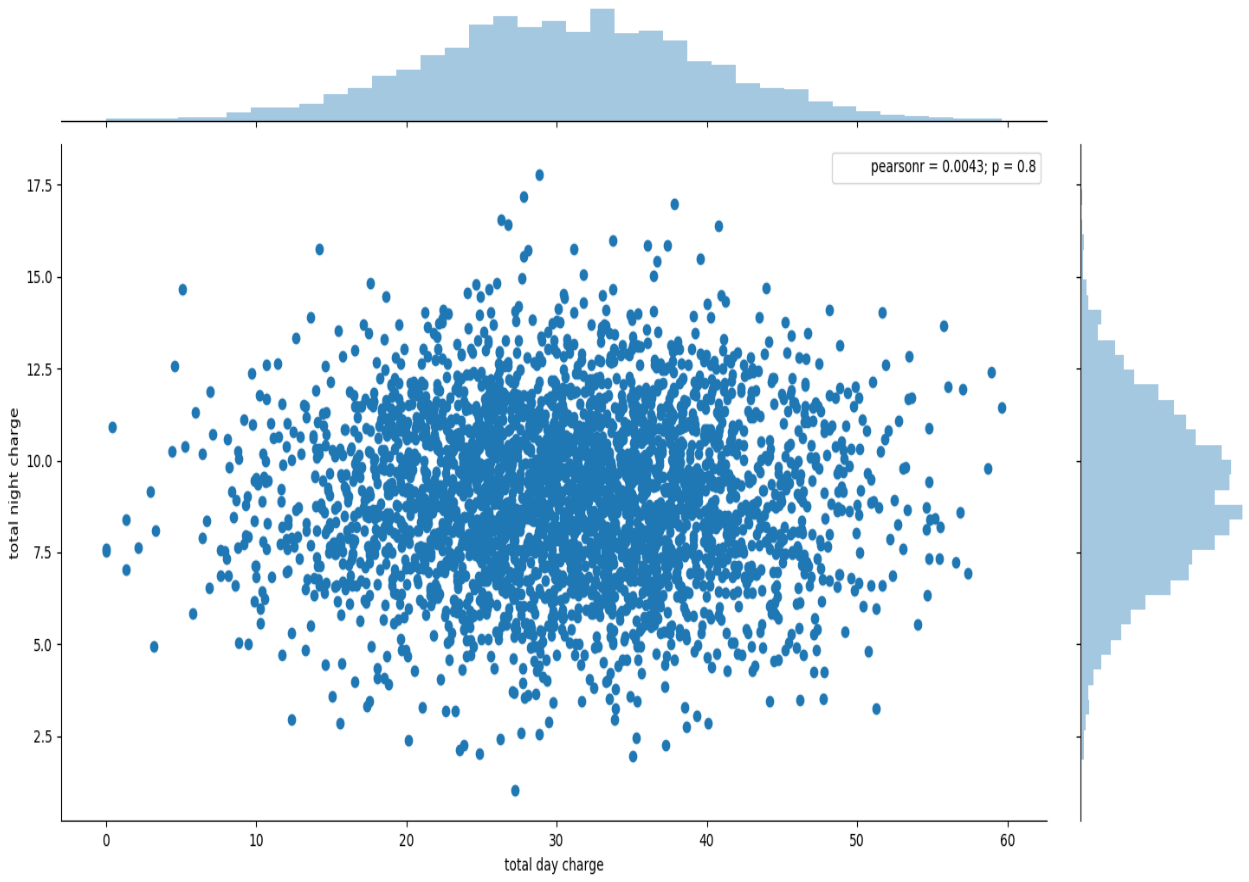
We can clearly see that all are normally distributed except ”number customer service calls” which is slightly right skewed

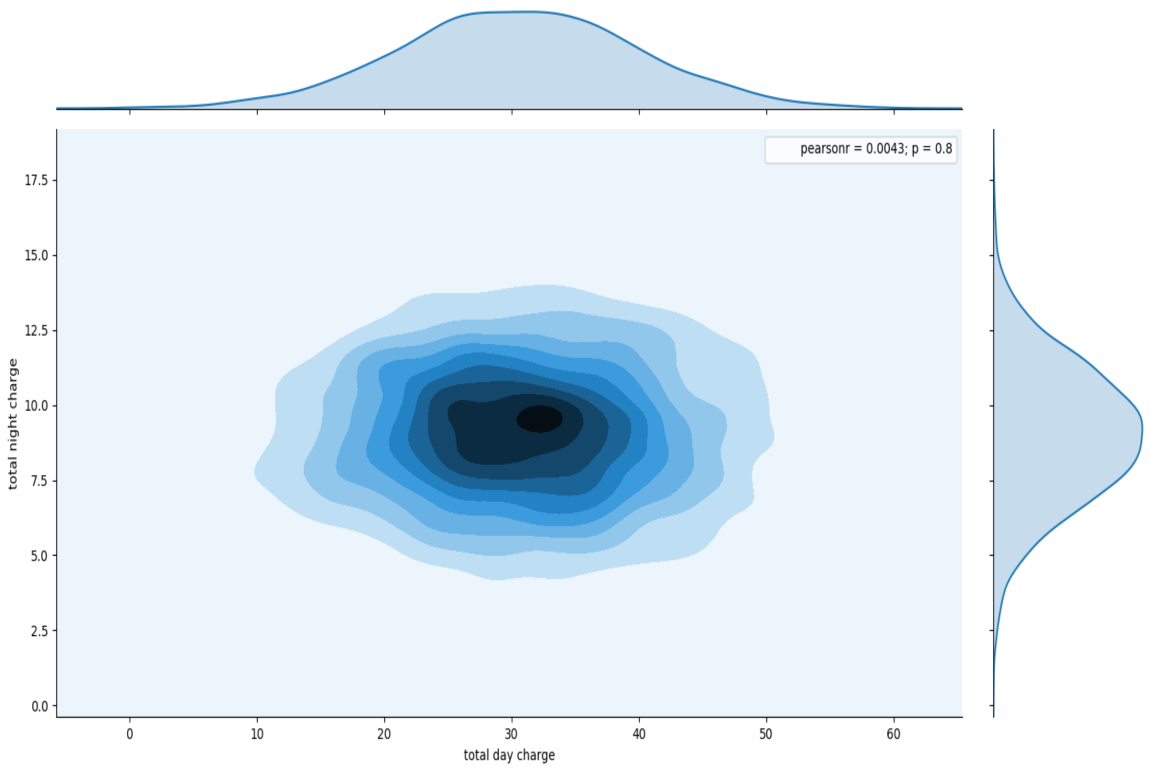
Scatter plots:

Let us check scatter plots of the variables to understand the concentration of values in 2D space

Code:

```
sns.jointplot("total day charge","total night charge",data=train,kind="scatter")
sns.jointplot("total day charge","total night charge",data=train,kind="kde")
```



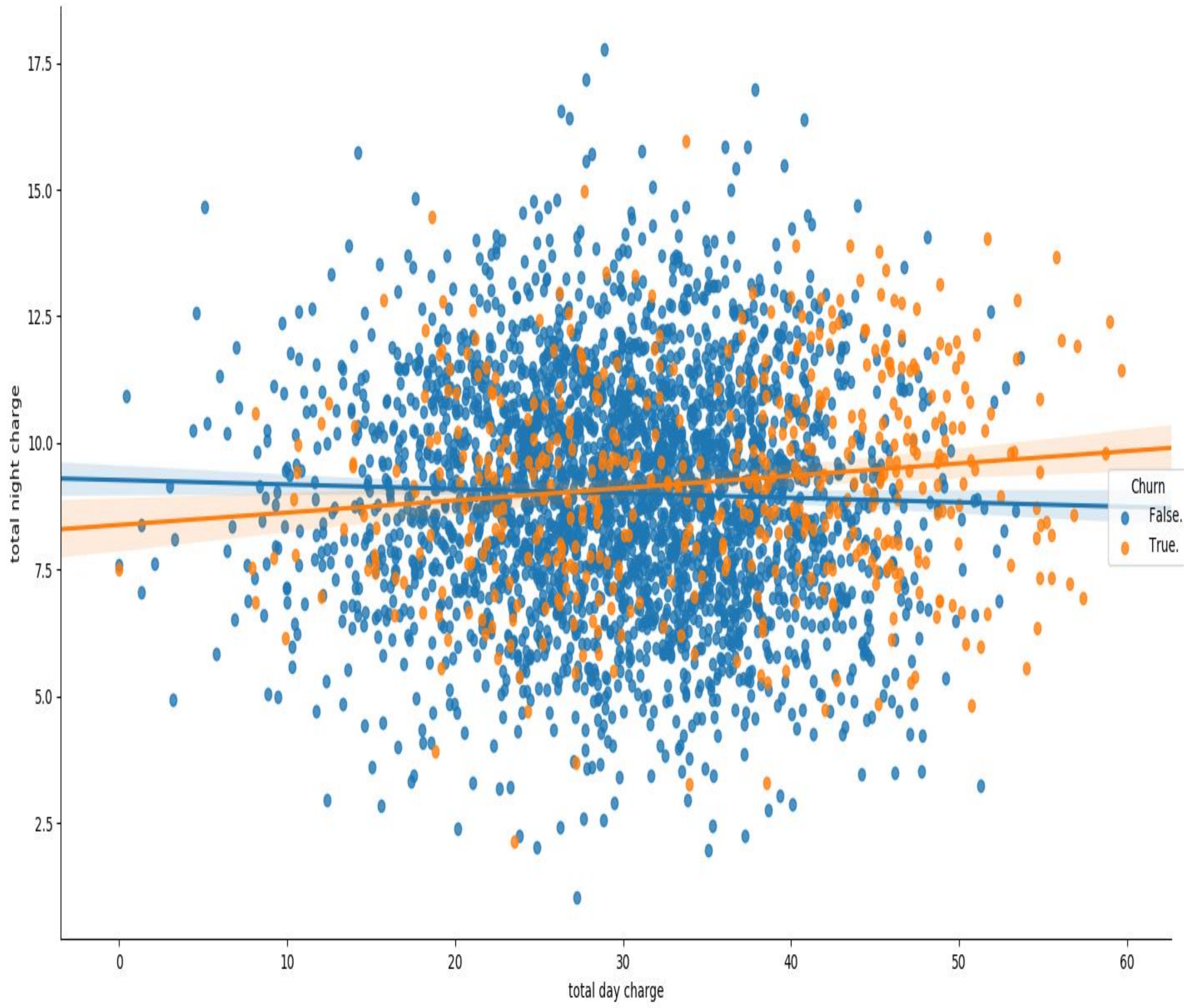


We can see that two variables are normally distributed and we can see that more concentrated area in 2D space and also it seems like both are uncorrelated.

Let’s see how these variables are related to the target variable Churn.

Code:

```
sns.lmplot('total day charge', 'total night charge', data=train,
        hue='Churn')
```

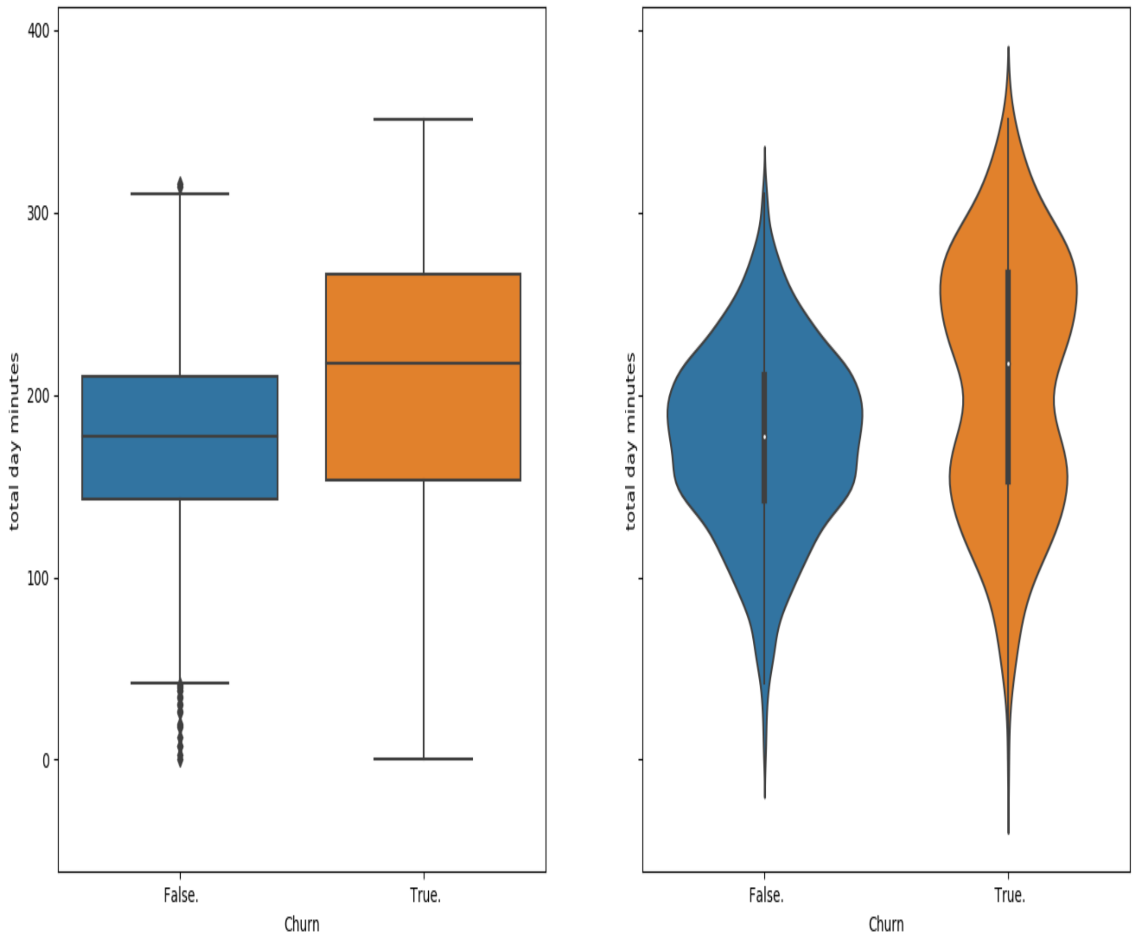


It seems that churned customers lean towards top right corner that is such customers tend to spend more time on phone during day time and night time

Let’s look at the distribution of day minutes spoken for the loyal and disloyal customers separately. We will create box and violin plots for *Total day minutes* grouped by the target variable.

Code:

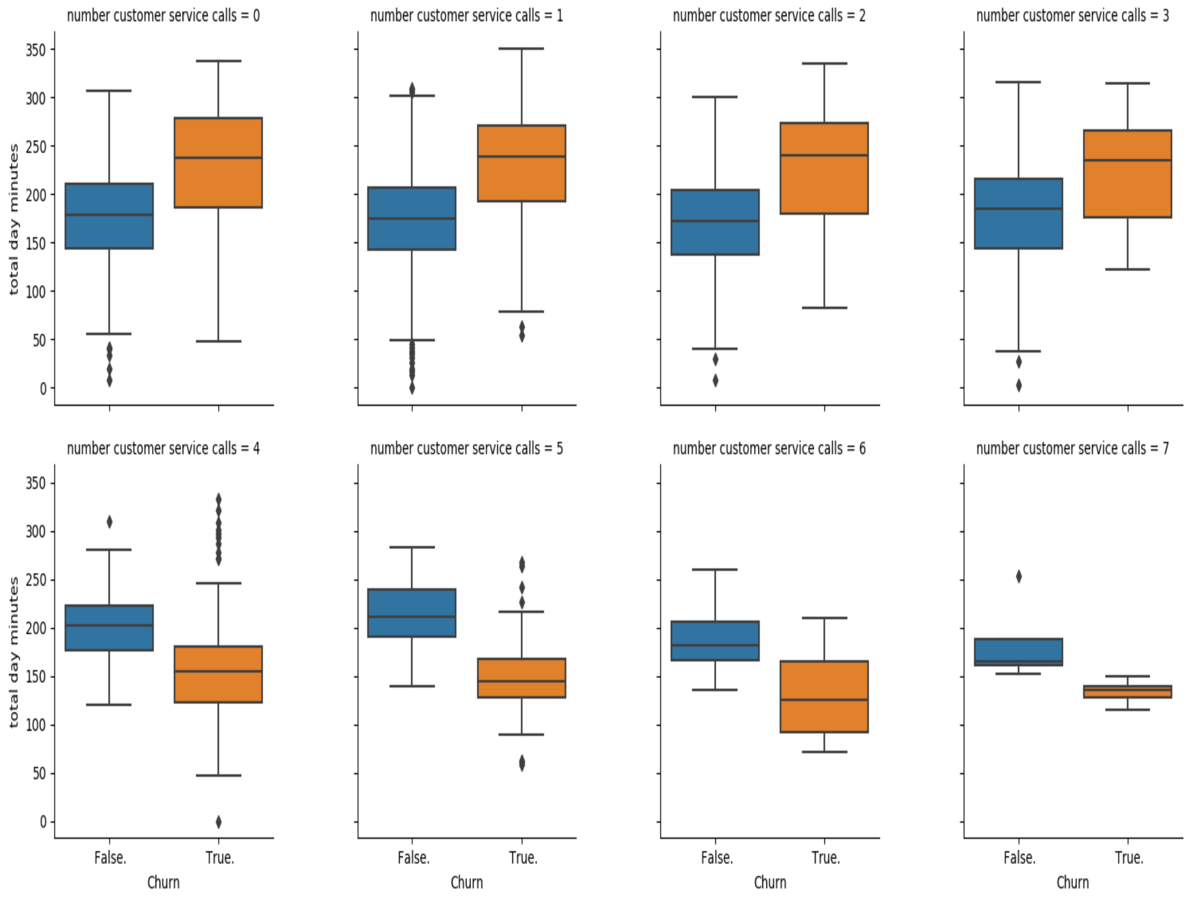
```
_, axes = plt.subplots(1, 2, sharey=True, figsize=(10, 4))

sns.boxplot(x='Churn', y='total day minutes',
        data=train, ax=axes[0])
sns.violinplot(x='Churn', y='total day minutes',
          data=train, ax=axes[1])
```



In this case, the violin plot does not contribute any additional information about our data as everything is clear from the box plot alone: disloyal customers tend to talk on the phone more.

An interesting observation:
On average, customers that discontinue their contracts are more active users of communication services. Perhaps they are unhappy with the tariffs, so a possible measure to prevent churn could be a reduction in call rates. The company will need to undertake additional economic analysis to find out whether such measures would be beneficial.

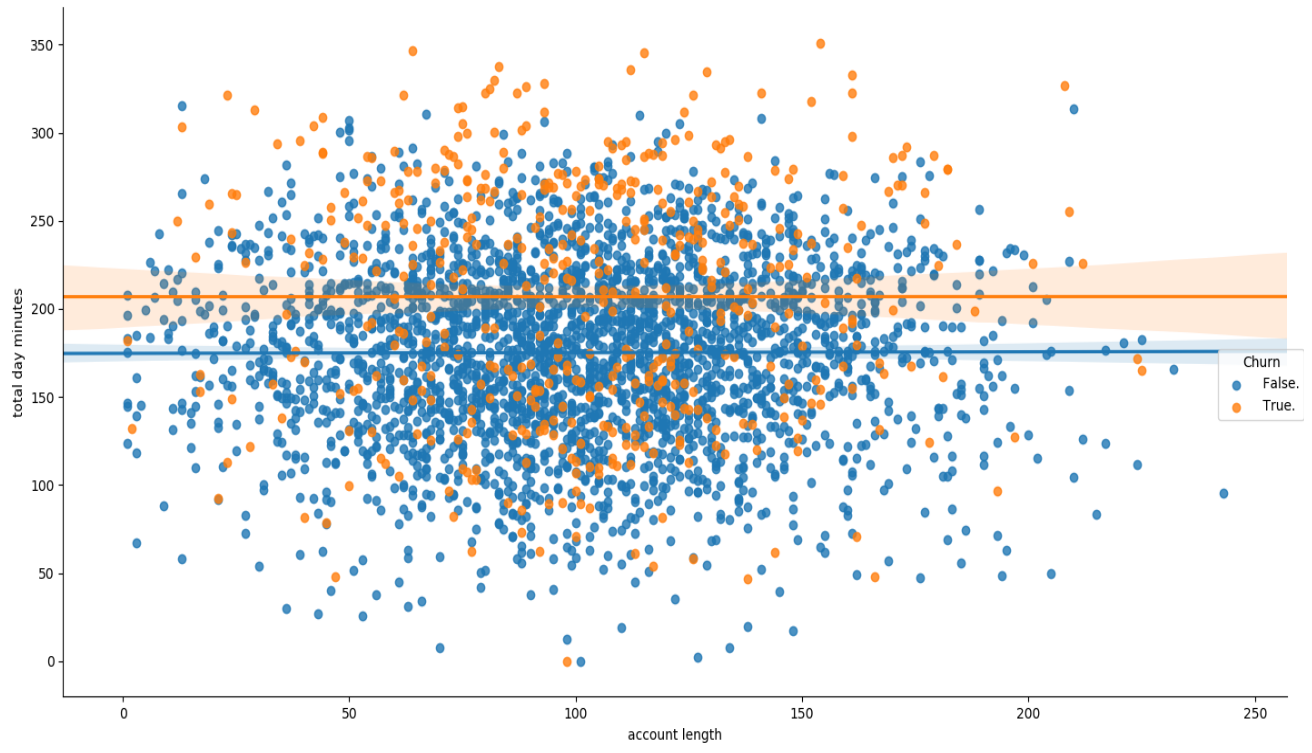
Let's visualize the interaction between *Total day minutes* and two categorical variables in the same plot



From this, we could conclude that, starting with 4 calls, *Total day minutes* may no longer be the main factor for customer churn. Perhaps, in addition to our previous guess about the tariffs, there are customers that are dissatisfied with the service due to other problems, which might lead to fewer number of day minutes spent on calls.

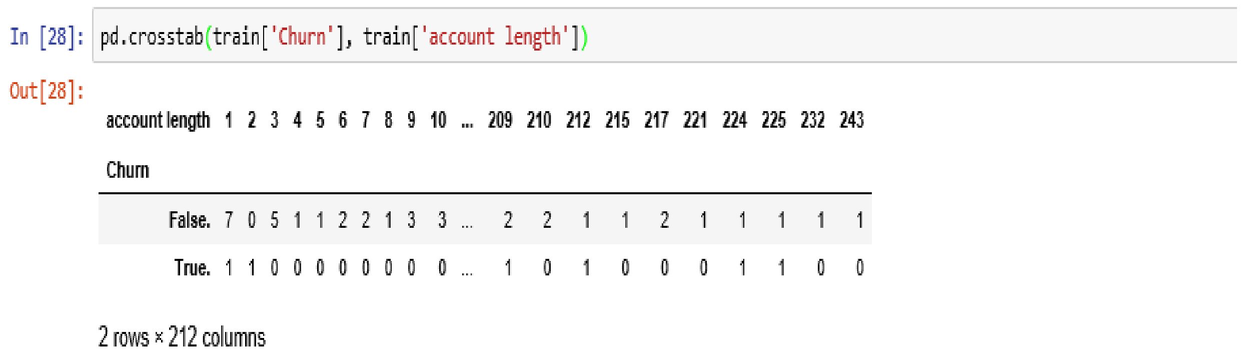
Let us Check if there any relation between customer usage and account length:

Let us draw a scatter plot between customer usage and account length



It clearly states that there is no relation between account length and customer usage

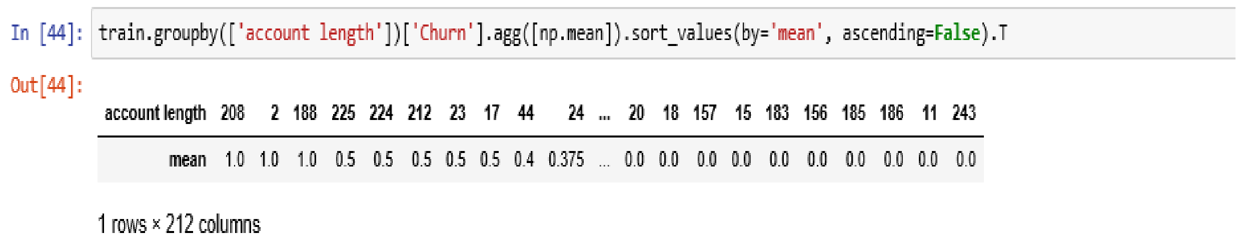
Let us check contingency table for account length and Churn variable



```
In [28]: pd.crosstab(train['Churn'], train['account length'])
```

```
Out[28]:
account length  1  2  3  4  5  6  7  8  9  10  ...  209  210  212  215  217  221  224  225  232  243
Churn
        False.  7  0  5  1  1  2  2  1  3   3  ...    2    2    1    1    2    1    1    1    1    1
         True.  1  1  0  0  0  0  0  0  0   0  ...    1    0    1    0    0    0    1    1    0    0

2 rows × 212 columns
```

Let us check the churn rates for different account lengths



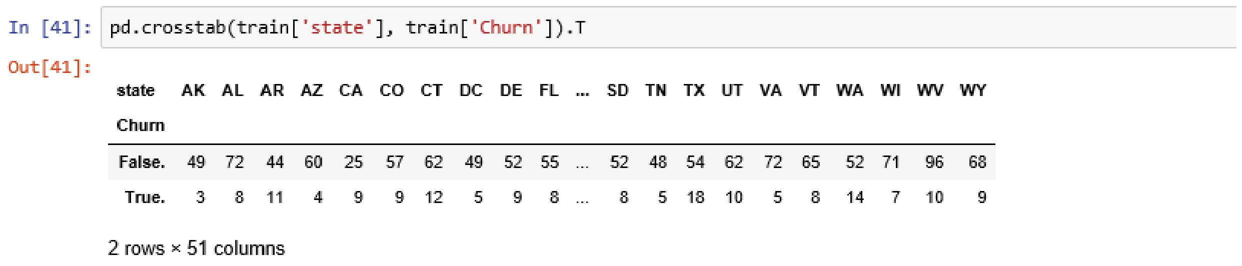
```
In [44]: train.groupby(['account length'])['Churn'].agg([np.mean]).sort_values(by='mean', ascending=False).T
Out[44]:
account length  208   2  188  225  224  212   23   17   44    24 ...  20   18  157   15  183  156  185  186   11  243
          mean  1.0 1.0  1.0  0.5  0.5  0.5  0.5  0.5  0.4 0.375 ... 0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0

1 rows × 212 columns
```

We can see that for account length 208,2,188
Churn Rate=100 percent

And for account length 17,23,44,212,224,225
Churn Rate=50 percent

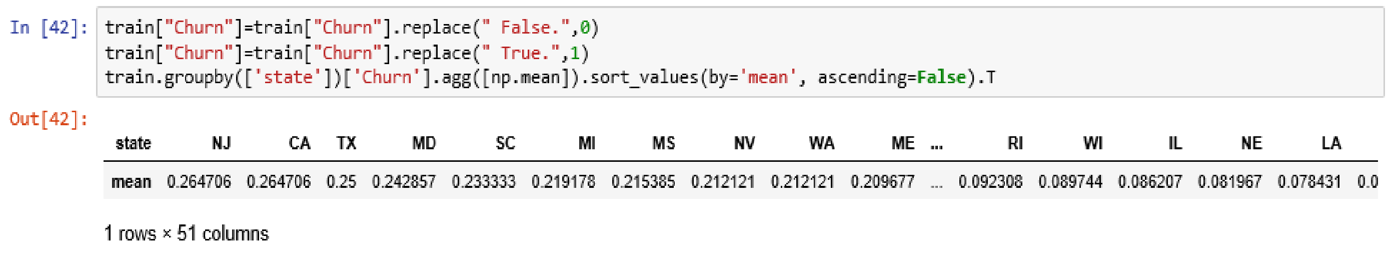
Top 5 States with high Churn Rate:

Let us create contingency table for state and churn variable



```
In [41]: pd.crosstab(train['state'], train['Churn']).T
Out[41]:
  state  AK  AL  AR  AZ  CA  CO  CT  DC  DE  FL ...  SD  TN  TX  UT  VA  VT  WA  WI  WV  WY
  Churn
 False.  49  72  44  60  25  57  62  49  52  55 ...  52  48  54  62  72  65  52  71  96  68
  True.   3   8  11   4   9   9  12   5   9   8 ...   8   5  18  10   5   8  14   7  10   9

2 rows × 51 columns
```

In the case of *State*, the number of distinct values is rather high: 51. We see that there are only a few data points available for each individual state — only 3 to 17 customers in each state abandoned the operator. Let’s ignore that for a second and calculate the churn rate for each state, sorting it from high to low



```
In [42]: train["Churn"]=train["Churn"].replace(" False.",0)
         train["Churn"]=train["Churn"].replace(" True.",1)
         train.groupby(['state'])['Churn'].agg([np.mean]).sort_values(by='mean', ascending=False).T
Out[42]:
state        NJ        CA    TX        MD        SC        MI        MS        NV        WA        ME ...        RI        WI        IL        NE        LA
 mean  0.264706  0.264706  0.25  0.242857  0.233333  0.219178  0.215385  0.212121  0.212121  0.209677 ...  0.092308  0.089744  0.086207  0.081967  0.078431  0.0

1 rows × 51 columns
```

It seems that the churn rate in *New Jersey* and *California* are above 25% and less than 6% for Hawaii and Alaska.
Top 5 states are:

- **NJ(New Jersey)**
- **CA(California)**
- **TX(Texas)**
- **MD(Maryland)**
- **SC(South Carolina)**

Chapter 3

Pre-Processing

Data Preprocessing is a technique that is used to convert the raw data into a clean data set.

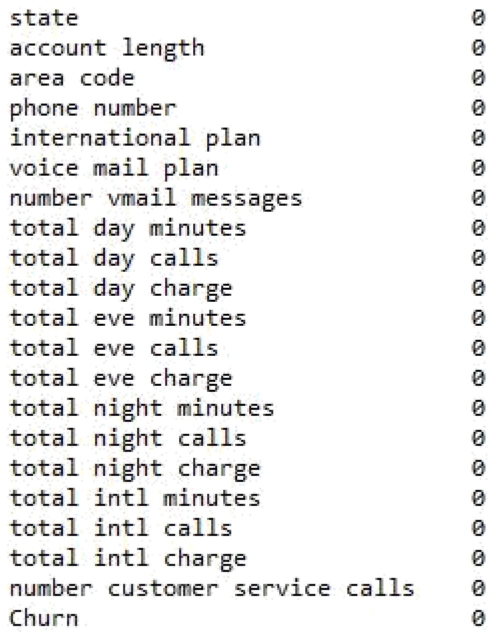
Before feeding the data to machine learning model, it should be cleaned so that the model will perform well.

The first step of pre-processing is

3.1 Missing Value Analysis

Missing data in the training data set can reduce the power / fit of a model or can lead to a biased model. It can lead to wrong prediction or classification.

Let’s check missing values of each column in the train and test data set



```
state                            0
account length                   0
area code                        0
phone number                     0
international plan               0
voice mail plan                  0
number vmail messages            0
total day minutes                0
total day calls                  0
total day charge                 0
total eve minutes                0
total eve calls                  0
total eve charge                 0
total night minutes              0
total night calls                0
total night charge               0
total intl minutes               0
total intl calls                 0
total intl charge                0
number customer service calls    0
Churn                            0
```

We can see that all the columns have zero missing values.

So we can proceed to next step of data pre-processing

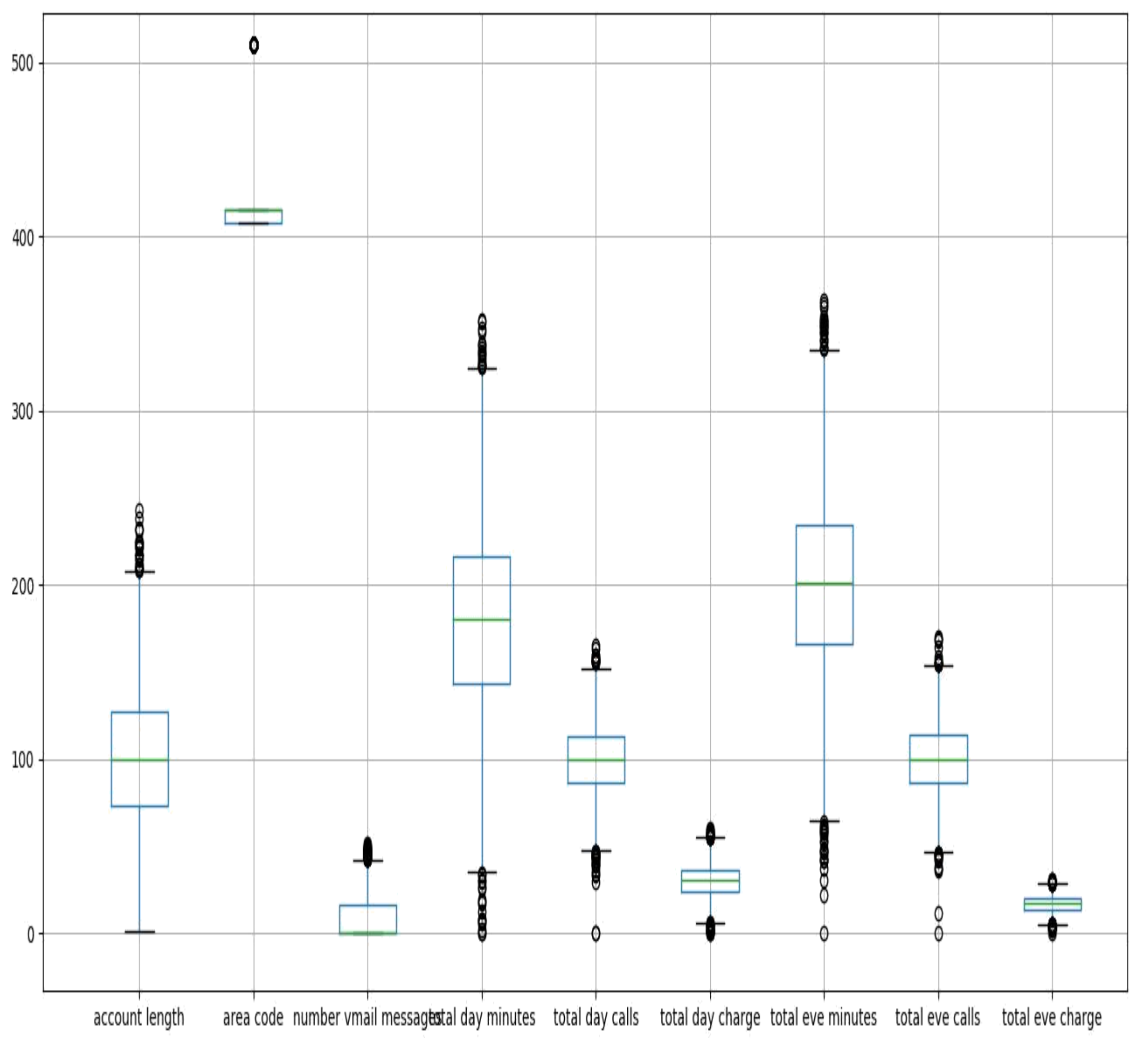
3.2 Outlier Analysis

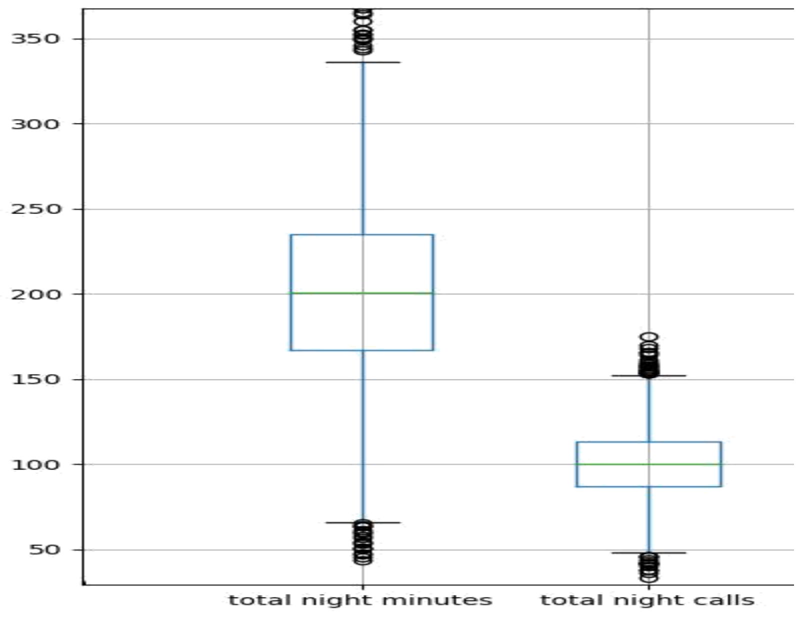
Outlier is an observation that appears far away and diverges from an overall pattern in a sample.

Let’s apply boxplot to have a look at outliers in the dataset

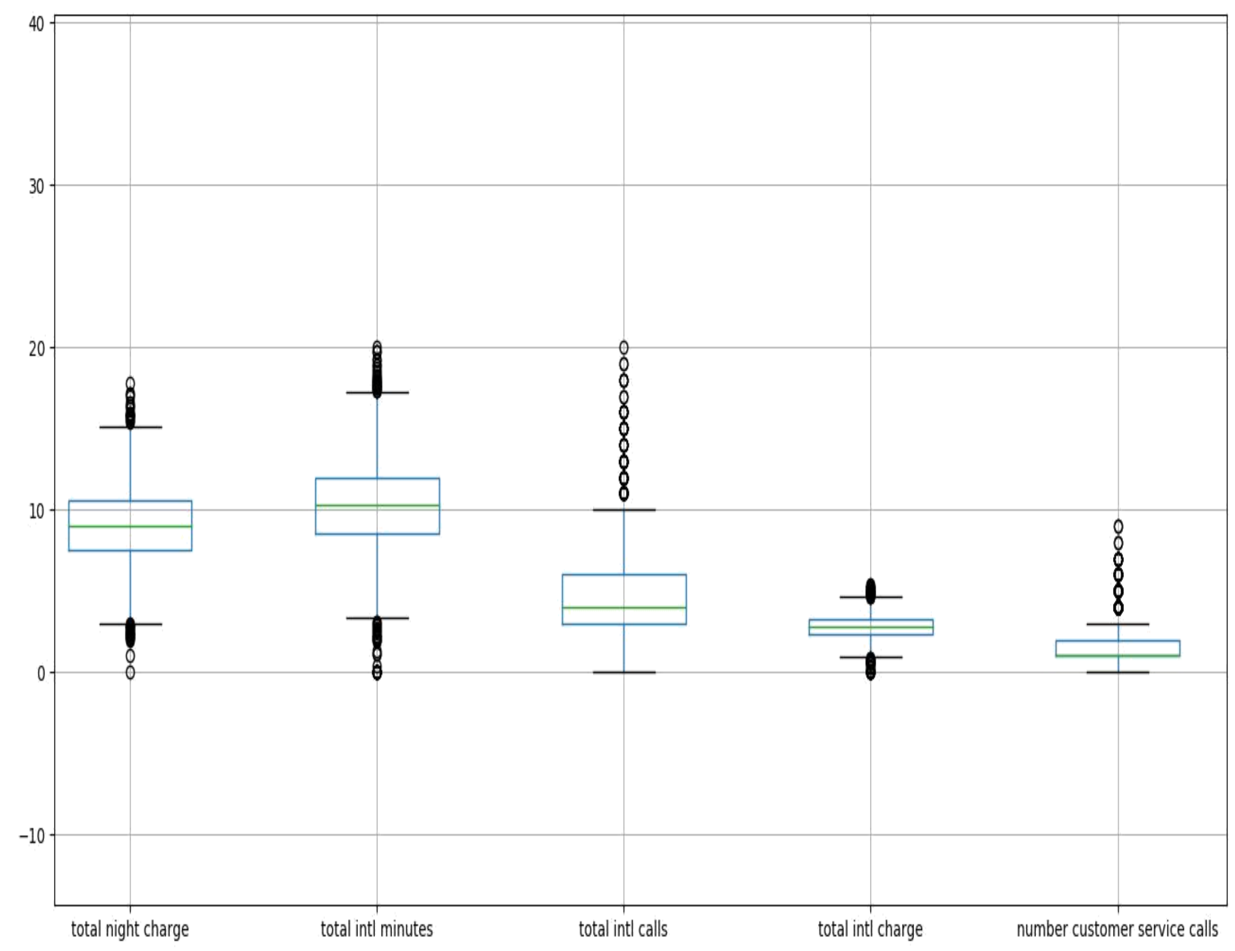
Code:

print(con_data.boxplot(figsize=(4,4)))


500
400
300
200
100
0
account length
area code
number vmail messages
total day minutes
total day calls
total day charge
total eve minutes
total eve calls
total eve charge



350
300
250
200
150
100
50
total night minutes
total night calls

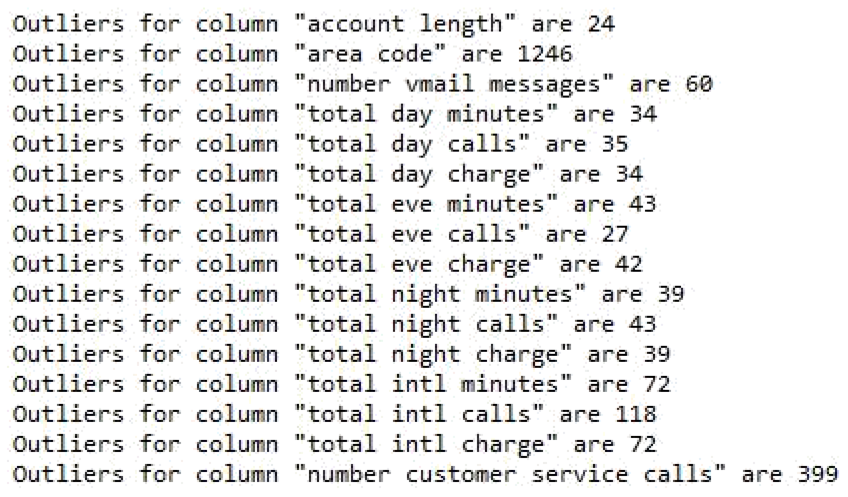


Code:

```
Num_Columns=[]
Cat_Columns=[]
for col in data.columns:
        if data[col].dtype==np.int64 or data[col].dtype==np.float64:
        Num_Columns.append(col)
else:
        Cat_Columns.append(col)

con_data=data[Num_Columns]
 print(con_data.boxplot(figsize=(4,4)))
```

From above boxplots, we can see that there are outliers in the dataset.

Let us look at the number of outliers present in the each column in the dataset.

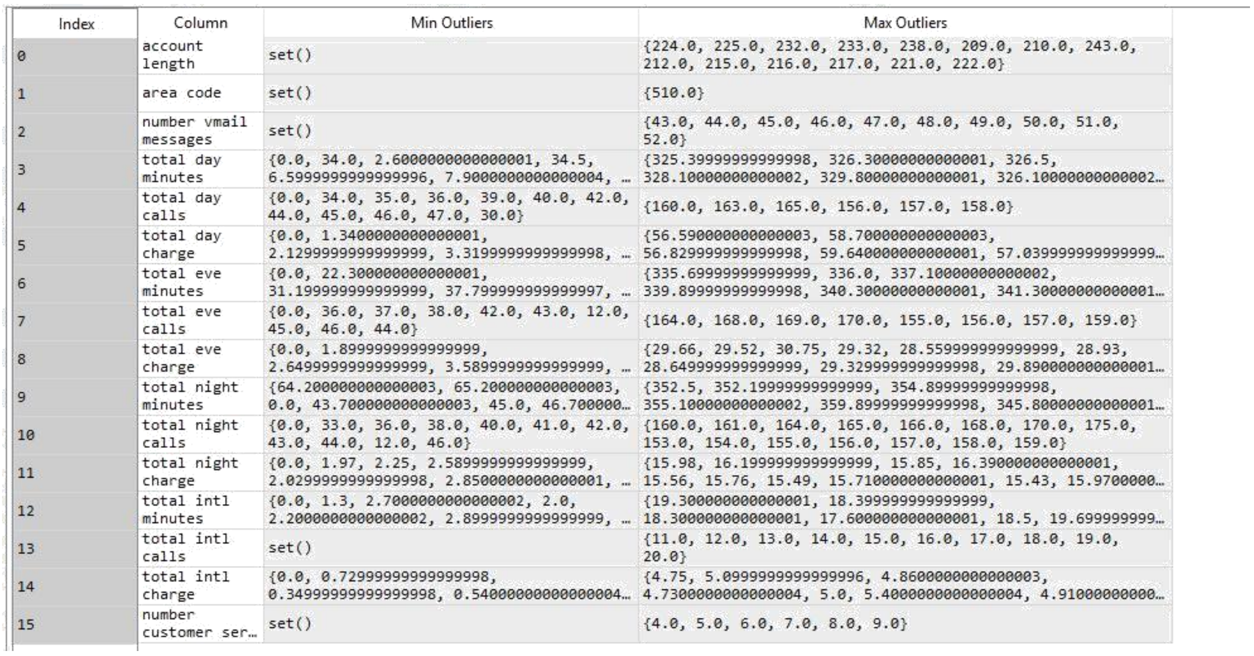


```
Outliers for column "account length" are 24
Outliers for column "area code" are 1246
Outliers for column "number vmail messages" are 60
Outliers for column "total day minutes" are 34
Outliers for column "total day calls" are 35
Outliers for column "total day charge" are 34
Outliers for column "total eve minutes" are 43
Outliers for column "total eve calls" are 27
Outliers for column "total eve charge" are 42
Outliers for column "total night minutes" are 39
Outliers for column "total night calls" are 43
Outliers for column "total night charge" are 39
Outliers for column "total intl minutes" are 72
Outliers for column "total intl calls" are 118
Outliers for column "total intl charge" are 72
Outliers for column "number customer service calls" are 399
```

Code:

```
Outliers_list=[]

for col in con_data.columns:

    q75,q25=np.percentile(data[col],[75,25])

    iqr=q75-q25

    minimum=q25-(iqr*1.5)

    maximum=q75+(iqr*1.5)

    Outliers_list.append((col,set(data.loc[data[col]<minimum,col]),set(data.loc[data[col]>maximum,col])))

    data.loc[data[col]<minimum,col]=np.nan

    data.loc[data[col]>maximum,col]=np.nan

    print("""Outliers for column "{}" are {}""".format(col,pd.isnull(data[col]).sum()))


Outlies_df=pd.DataFrame(Outliers_list,columns=["Column","Min Outliers","Max Outliers"])
```

We can see that “area code” and “number customer service calls” have highest number of outliers.



Index	Column	Min Outliers	Max Outliers
0	account length	set()	{224.0, 225.0, 232.0, 233.0, 238.0, 209.0, 210.0, 243.0, 212.0, 215.0, 216.0, 217.0, 221.0, 222.0}
1	area code	set()	{510.0}
2	number vmail messages	set()	{43.0, 44.0, 45.0, 46.0, 47.0, 48.0, 49.0, 50.0, 51.0, 52.0}
3	total day minutes	{0.0, 34.0, 2.6000000000000001, 34.5, 6.5999999999999996, 7.9000000000000004, …	{325.39999999999998, 326.30000000000001, 326.5, 328.10000000000002, 329.80000000000001, 326.10000000000002…
4	total day calls	{0.0, 34.0, 35.0, 36.0, 39.0, 40.0, 42.0, 44.0, 45.0, 46.0, 47.0, 30.0}	{160.0, 163.0, 165.0, 156.0, 157.0, 158.0}
5	total day charge	{0.0, 1.3400000000000001, 2.1299999999999999, 3.3199999999999998, …	{56.590000000000003, 58.700000000000003, 56.829999999999998, 59.640000000000001, 57.039999999999999…
6	total eve minutes	{0.0, 22.300000000000001, 31.199999999999999, 37.799999999999997, …	{335.69999999999999, 336.0, 337.10000000000002, 339.89999999999998, 340.30000000000001, 341.30000000000001…
7	total eve calls	{0.0, 36.0, 37.0, 38.0, 42.0, 43.0, 12.0, 45.0, 46.0, 44.0}	{164.0, 168.0, 169.0, 170.0, 155.0, 156.0, 157.0, 159.0}
8	total eve charge	{0.0, 1.8999999999999999, 2.6499999999999999, 3.5899999999999999, …	{29.66, 29.52, 30.75, 29.32, 28.559999999999999, 28.93, 28.649999999999999, 29.329999999999998, 29.890000000000001…
9	total night minutes	{64.200000000000003, 65.200000000000003, 0.0, 43.700000000000003, 45.0, 46.700000…	{352.5, 352.19999999999999, 354.89999999999998, 355.10000000000002, 359.89999999999998, 345.80000000000001…
10	total night calls	{0.0, 33.0, 36.0, 38.0, 40.0, 41.0, 42.0, 43.0, 44.0, 12.0, 46.0}	{160.0, 161.0, 164.0, 165.0, 166.0, 168.0, 170.0, 175.0, 153.0, 154.0, 155.0, 156.0, 157.0, 158.0, 159.0}
11	total night charge	{0.0, 1.97, 2.25, 2.5899999999999999, 2.0299999999999998, 2.8500000000000001, …	{15.98, 16.199999999999999, 15.85, 16.390000000000001, 15.56, 15.76, 15.49, 15.710000000000001, 15.43, 15.9700000…
12	total intl minutes	{0.0, 1.3, 2.7000000000000002, 2.0, 2.2000000000000002, 2.8999999999999999, …	{19.300000000000001, 18.399999999999999, 18.300000000000001, 17.600000000000001, 18.5, 19.699999999…
13	total intl calls	set()	{11.0, 12.0, 13.0, 14.0, 15.0, 16.0, 17.0, 18.0, 19.0, 20.0}
14	total intl charge	{0.0, 0.72999999999999998, 0.34999999999999998, 0.54000000000000004…	{4.75, 5.0999999999999996, 4.8600000000000003, 4.7300000000000004, 5.0, 5.4000000000000004, 4.9100000000…
15	number customer ser…	set()	{4.0, 5.0, 6.0, 7.0, 8.0, 9.0}

We can see that area code has

Let us look at the outliers present in those columns.

Area code column has three values 408,415 and 510

510 are outliers which are of count 1246

Number of service calls column has 5 outliers.

We cannot delete these observations which results in loss of data.

So we are going to impute these outliers with extreme values that column.

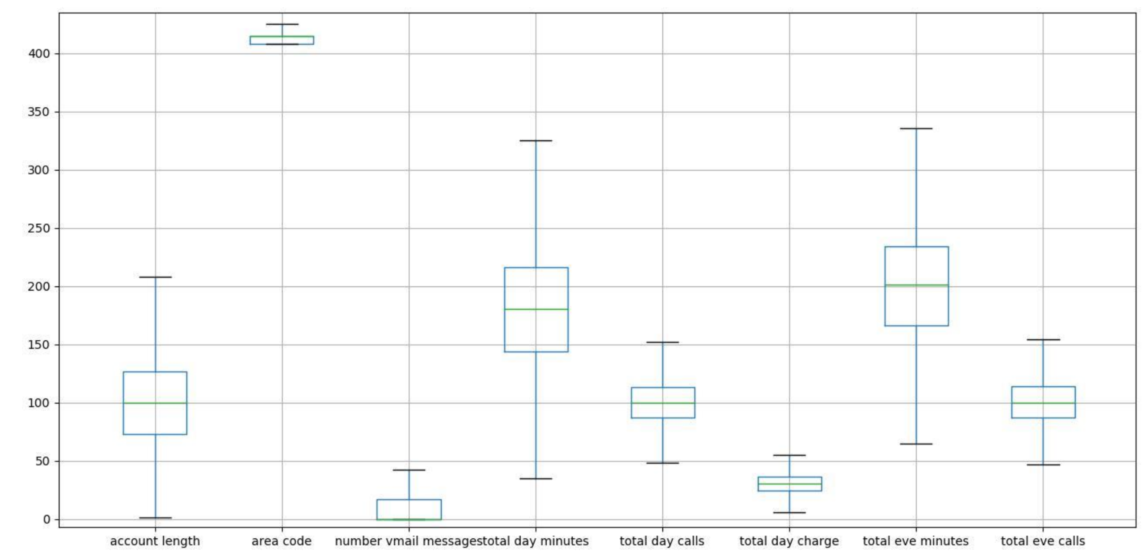
Imputing with extreme values:

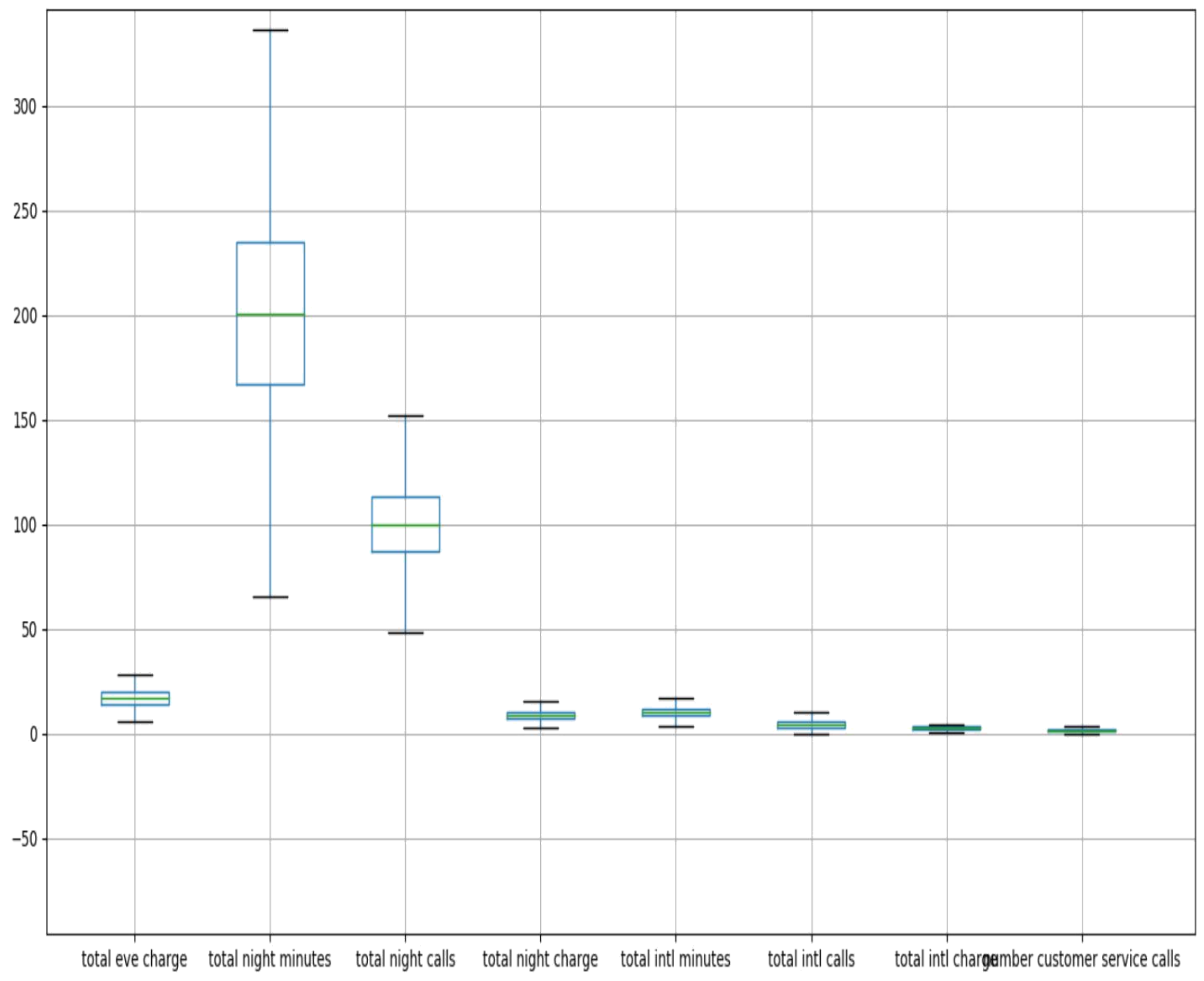
Here extreme values refers to value of 5th and 95th percentiles.

5th percententile is the lower extreme value and 95th percentile refers to upper extreme.

Code:

Let us look at results after imputing with extreme values





```
Outliers for column "account length" are 0
Outliers for column "area code" are 0
Outliers for column "number vmail messages" are 0
Outliers for column "total day minutes" are 0
Outliers for column "total day calls" are 0
Outliers for column "total day charge" are 0
Outliers for column "total eve minutes" are 0
Outliers for column "total eve calls" are 0
Outliers for column "total eve charge" are 0
Outliers for column "total night minutes" are 0
Outliers for column "total night calls" are 0
Outliers for column "total night charge" are 0
Outliers for column "total intl minutes" are 0
Outliers for column "total intl calls" are 0
Outliers for column "total intl charge" are 0
Outliers for column "number customer service calls" are 0
```

We can see that there are no outliers.

So we can proceed to next step

3.3 Feature Selection:

Actually the success of all Machine Learning algorithms depends on how we present the data. The features in your data will directly influence the predictive models we use and the results we can achieve. We can say that, the better the features that we prepare and choose, the better the results you will achieve. Unneeded and irrelevant data to be removed that do not contribute much information. Feature selection is used to reduce the complexity of the model

For Categorical variables, we are using Chi-square test.

For numerical variables we use Correlation method.

Let us apply correlation test of independence on categorical variables.

Chi-Square Test of Independence:

It compares two variables in a contingency table to see if they are related

Hypothesis Testing:

Null Hypothesis: Two variables are independent

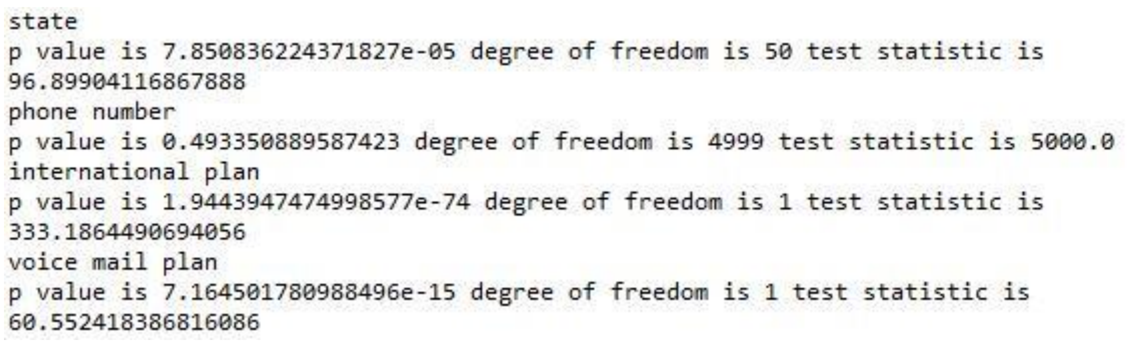
Alternate Hypothesis: Two variables are not independent

If p>0.05 we reject null hypothesis.

Let us write the code to apply chi-square test on target variable churn and remaining categorical variables

Code:

```
for col in Cat_Columns:
    if col!="Churn":
        chi2,p,dof,exp=chi2_contingency(pd.crosstab(data["Churn"],data[col]))
        print(col)
        print("p value is {} degree of freedom is {} test statistic is {}".format(p,dof,chi2))
```



```
state
p value is 7.850836224371827e-05 degree of freedom is 50 test statistic is
96.89904116867888
phone number
p value is 0.493350889587423 degree of freedom is 4999 test statistic is 5000.0
international plan
p value is 1.9443947474998577e-74 degree of freedom is 1 test statistic is
333.1864490694056
voice mail plan
p value is 7.164501780988496e-15 degree of freedom is 1 test statistic is
60.552418386816086
```

We can see that the output of the code

Phone number column has p value greater than 0.05 which says we can accept null hypothesis.

Null hypothesis states that two variables are not dependent.

So we have to delete "phone number column" as it does not contain much information about target variable "Churn".

Correlation Analysis:

Correlation applies only on numerical data.

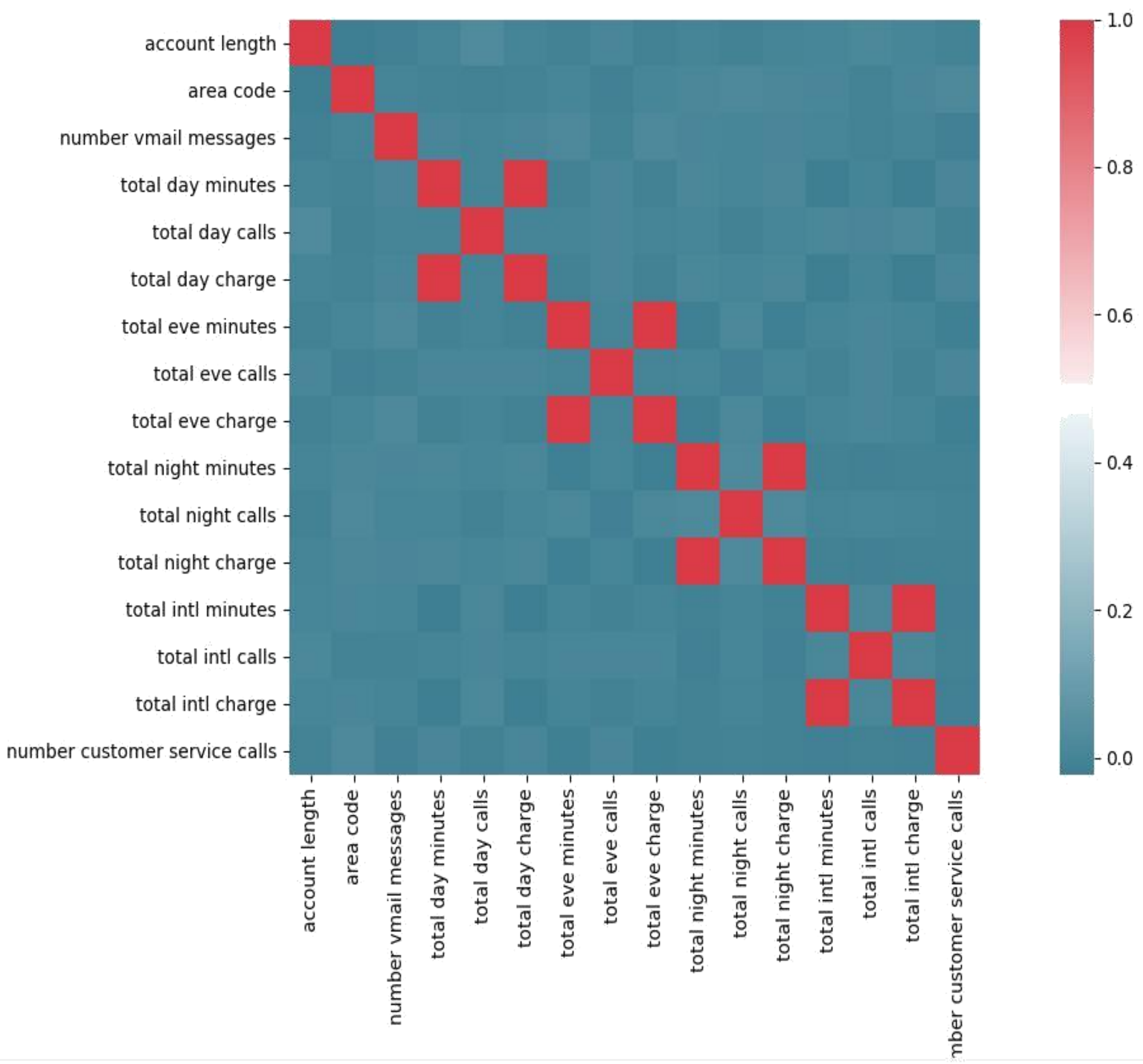
Data has dependent and independent variables.

There must be high correlation between independent variable and dependent variable.(variable is carrying high info about dependent) and low correlation among independents and dependents

Let us visualize the dependency of continuous variables with the help of heatmap

Code:

```
corr_mat=data[Num_Columns].corr() corr_df=pd.DataFrame(corr_mat)
sns.heatmap(corr_mat,mask=np.zeros_like(corr_mat,dtype=np.bool),cmap=sns.divergin
g_palette(220,10,as_cmap=True),
        square=True)
```



Red boxes indicate high correlation and blue refers to high correlation.

From the above heatmap, the following dependencies are observed

total day charge,total day minutes(High Correlation)

total eve charge,total eve minutes(High Correlation)

total night minutes,total night charge(High Correlation)

total intl charge,total intl minutes(High Correlation)

These are dependent variables which have high correlation and contains same information.

If we include those variables it leads to multicollinearity.so we can delete these variables.

From Feature Selection, We conclude that 5 variables can be deleted from the dataset.

Those 5 variables are:

Phone number

total day minutes
total eve minutes
total night minutes
total intl minutes

3.4 Feature Engineering:

Now let us derive new features from existing features.

Total number of morning calls, evening calls, night calls, and international calls are same but different times and different plans.so we can reduce it into total number of calls by adding each of them

Morning call charges, evening charges, night and international charges can be reduced to total charges to reduce complexity

Code:
```
data["total calls"]=data["total day calls"]+data["total eve calls"]+data["total night
calls"]+data["total intl calls"]
data["total call charge"]=data["total day charge"]+data["total eve charge"]+data["total night
charge"]+data["total intl charge"]
data=data.drop(["total day calls","total eve calls","total night calls","total intl calls","total day
charge","total eve charge","total night charge","total intl charge"],axis=1)
```

With the help of feature engineering, we have derived new features like total calls, total call charge and deleted the existing ones

3.5 Feature Scaling:

Feature scaling is a method used to standardize the range of independent variables or features of data.

There are two methods

Normalization
Standardization

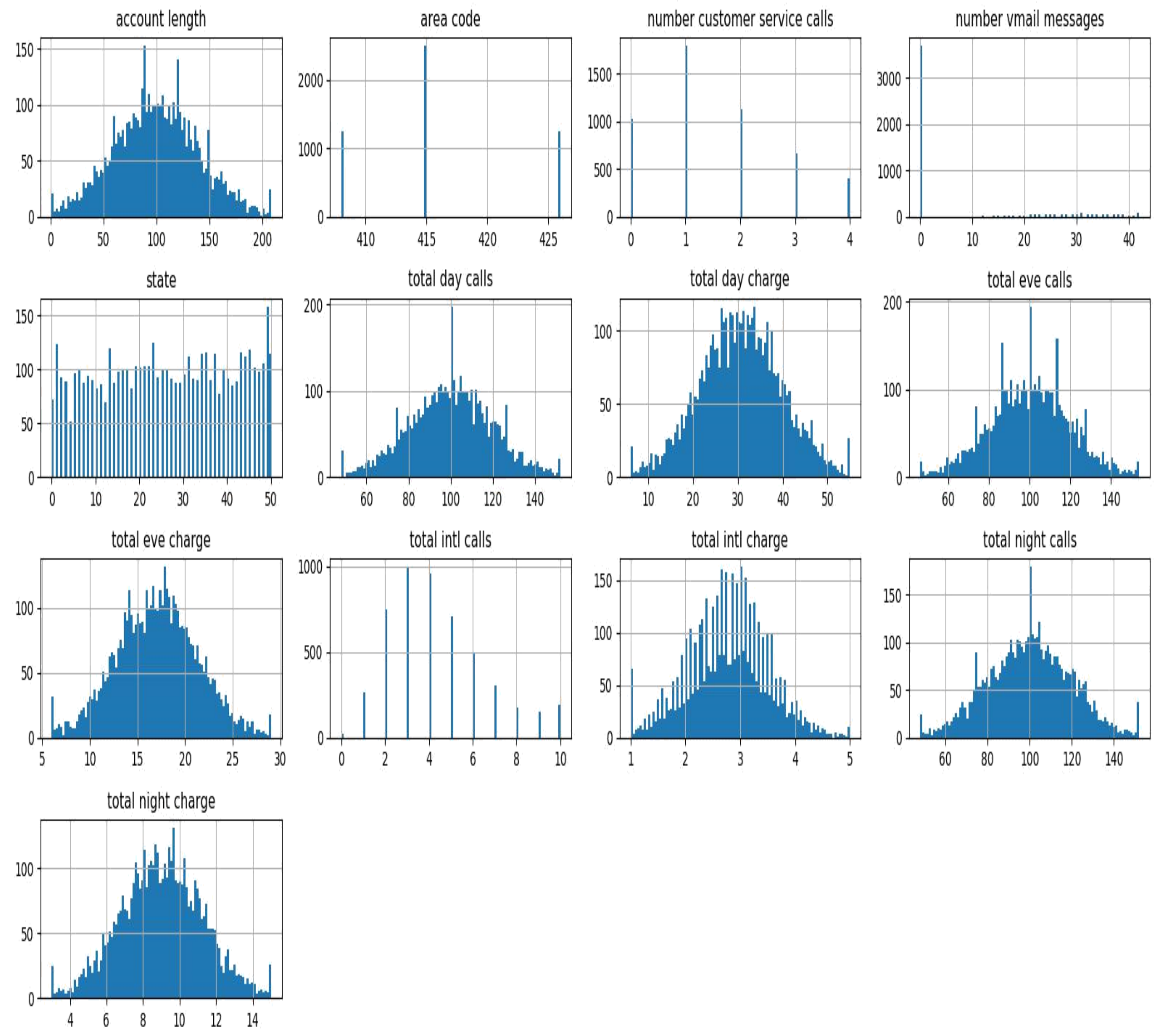
When data is not normally distributed, we go for Normalization

Whereas Standardization works only for normally distributed data.

Let us check the distribution for all the columns in the data

Code:

```
print(data[['state','account length','area code','number vmail messages','total day calls','total day charge','total eve calls','total eve charge','total night calls','total night charge','total intl calls','total intl charge','number customer service calls']].hist(figsize=(4,4),bins=100))
```



We can see that some columns like number vmail messages,state,area code,total intl calls are not normally distributed. Standardization won't work well for columns that are not normally distributed.so it better if we normalization process.

With the help of normalization we are converting entire data in the range of 0 to 1.

Formula:

Value new= (Value-min value)/(max value-min value)

Code:

```
data=pd.DataFrame(MinMaxScaler().fit_transform(data),columns=data.columns)
```

With this, we have completed all steps of preprocessing and successfully cleaned the data

Chapter 4

Modelling

In modelling phase, we apply different models on train data and test the model with test data.

4.1 Logistic regression:

Let us apply Logistic Regression on the training set and test it with testing set

Code:

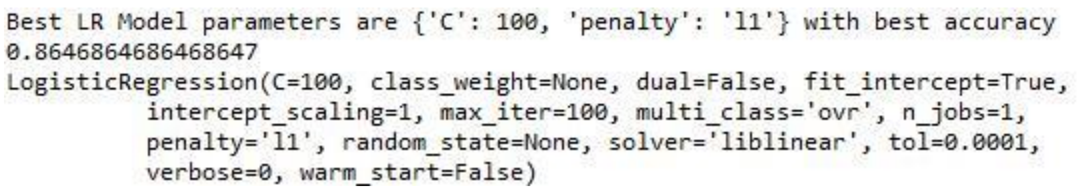
```
#Logistic Regression

lr=LogisticRegression()

lr_gs=GridSearchCV(lr,cv=5,param_grid= {'C':[10**i for i in range(-3,3)],'penalty':['l1','l2']})

lr_gs.fit(x_train, y_train)

lr_est=lr_gs.best_estimator_

print("Best LR Model parameters are {} with best accuracy
{}".format(lr_gs.best_params_,lr_gs.best_score_))

y_pred_lr=lr_gs.predict(x_test)

#Evaluation Metrics

cm_lr=pd.crosstab(y_test,y_pred_lr)

TN=cm_lr.iloc[0,0]

FP=cm_lr.iloc[0,1]

FN=cm_lr.iloc[1,0]

TP=cm_lr.iloc[1,1]

from sklearn.metrics import accuracy_score
```

```
print("*****Logistic Regression Evaluation Metrics********")

print("Accuracy is {}".format(accuracy_score(y_test,y_pred_lr)*100))

#negative cases prediction

print("Specificity or True Negative Rate is {}".format(TN*100/(TN+FP)))

#postive cases prediction

print("Recall or sensitivity or True Postive Rate is {}".format(TP*100/(TP+FN)))

print("False Positive Rate is {}".format(FP*100/(FP+TN)))

print("False Negative Rate is {}".format(FN*100/(FN+TP)))

print("***************************************************")

print("*********cross validation score****************")
from sklearn.cross_validation import cross_val_score
cv_score=np.mean(cross_val_score(lr_est,x_train,y_train,cv=5))
print("cv score for logistic Regression is {:5.2f}".format(cv_score*100))
print("************************************************")
```

In the above code, we have applied logistic regression on training set.

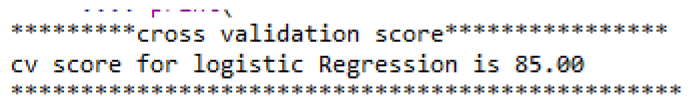
We have tuned all the parameters in Logisitic estimator with help of GridSearchCV and selected best estimator

Best Estimator:

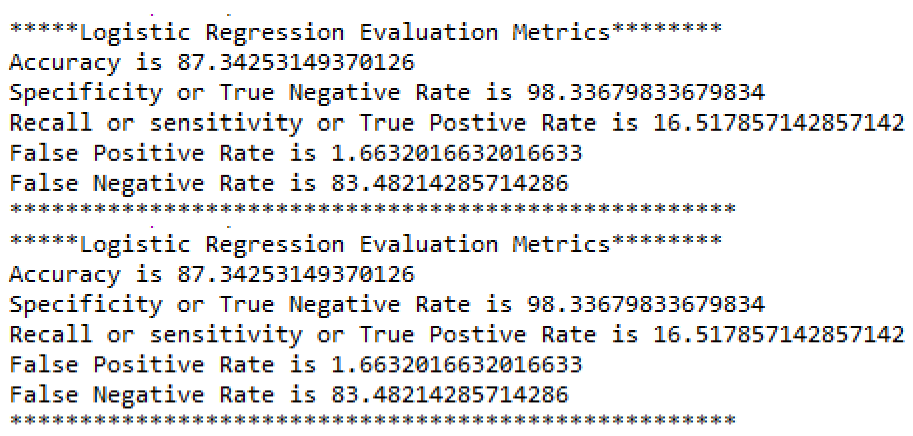


```
Best LR Model parameters are {'C': 100, 'penalty': 'l1'} with best accuracy
0.8646864686468647
LogisticRegression(C=100, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
          penalty='l1', random_state=None, solver='liblinear', tol=0.0001,
          verbose=0, warm_start=False)
```

Cross Validation Score:



```
*********cross validation score****************
cv score for logistic Regression is 85.00
************************************************
```



```
*****Logistic Regression Evaluation Metrics********
Accuracy is 87.34253149370126
Specificity or True Negative Rate is 98.33679833679834
Recall or sensitivity or True Postive Rate is 16.517857142857142
False Positive Rate is 1.6632016632016633
False Negative Rate is 83.48214285714286
**************************************************
*****Logistic Regression Evaluation Metrics********
Accuracy is 87.34253149370126
Specificity or True Negative Rate is 98.33679833679834
Recall or sensitivity or True Postive Rate is 16.517857142857142
False Positive Rate is 1.6632016632016633
False Negative Rate is 83.48214285714286
**************************************************
```

4.2 Decision Tree:

Let us apply Decision Tree on the training set and test it with testing set

Code:

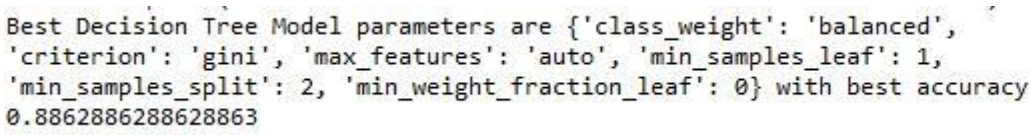
```
#Decision Tree

from sklearn.tree import DecisionTreeClassifier

dt=DecisionTreeClassifier()

param={"criterion":['gini','entropy'],'min_samples_split':[2,5,10],'min_samples_leaf':[1,5,10],"cl
ass_weight":["balanced"],'max_features':['auto','sqrt','log2'],'min_weight_fraction_leaf':[0,0.5]}

gs_dt=GridSearchCV(dt,cv=5,param_grid=param)

gs_dt.fit(x_train,y_train)

dt_est=gs_dt.best_estimator_

print("Best Decision Tree Model parameters are {} with best
accuracy {}".format(gs_dt.best_params_,gs_dt.best_score_))

y_pred_dt=dt_est.predict(x_test)

cm_dt=pd.crosstab(y_test,y_pred_dt)

TN=cm_dt.iloc[0,0]

FP=cm_dt.iloc[0,1]

FN=cm_dt.iloc[1,0]

TP=cm_dt.iloc[1,1]
```

```
from sklearn.metrics import accuracy_score print("*****Decision
Tree Evaluation Metrics********") print("Accuracy is
{}".format(accuracy_score(y_test,y_pred_dt)*100)) #negative cases
prediction
print("Specificity or True Negative Rate is {}".format(TN*100/(TN+FP)))
#postive cases prediction
print("Recall or sensitivity or True Postive Rate is {}".format(TP*100/(TP+FN)))
print("False Positive Rate is {}".format(FP*100/(FP+TN)))
print("False Negative Rate is {}".format(FN*100/(FN+TP)))
print("*****************************************************")
```

In the above code, we have applied Decision Tree on training set.

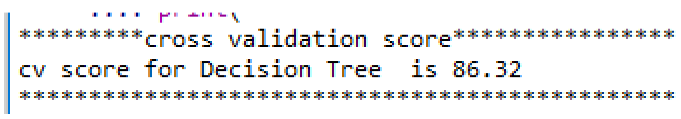
We have tuned all the parameters in Decision Tree estimator with help of GridSearchCV and selected best estimator

Best Estimator:



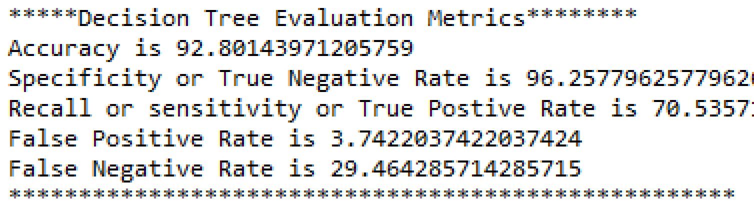
```
Best Decision Tree Model parameters are {'class_weight': 'balanced',
'criterion': 'gini', 'max_features': 'auto', 'min_samples_leaf': 1,
'min_samples_split': 2, 'min_weight_fraction_leaf': 0} with best accuracy
0.8862886288628863
```

Cross Validation Score:



```
*********cross validation score****************
cv score for Decision Tree  is 86.32
***********************************************
```

Output of Decision Tree:



```
*****Decision Tree Evaluation Metrics********
Accuracy is 92.80143971205759
Specificity or True Negative Rate is 96.25779625779626
Recall or sensitivity or True Postive Rate is 70.53571428571429
False Positive Rate is 3.7422037422037424
False Negative Rate is 29.464285714285715
****************************************************
```

4.3 Random Forest:

Let us apply Random Forest on the training set and test it with testing set

Code:

```
#Random Forest

from sklearn.ensemble import RandomForestClassifier

rf=RandomForestClassifier()

param={"n_estimators":[10,100,300,500,700,1000],"criterion":["gini","entropy"],"max_features":["auto","sqrt","log2"],"min_samples_split":[2,5,10],"min_samples_leaf":[1,5,10],"min_weight_fraction_leaf":[0,0.5],"verbose":[0,2,4,5],"class_weight":["balanced"]}
```

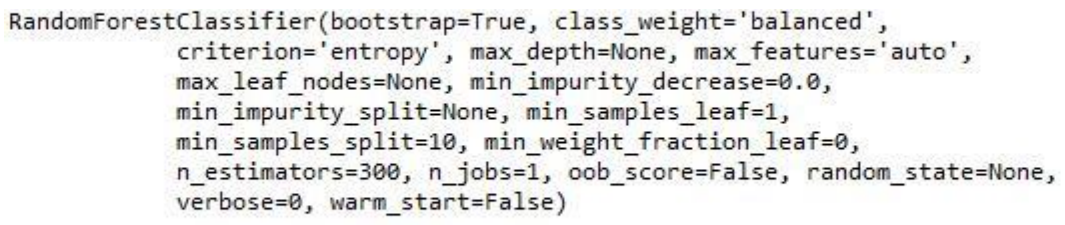
```
gs_rf=GridSearchCV(rf,cv=5,param_grid=param)

gs_rf.fit(x_train,y_train)

rf_est=gs_rf.best_estimator_

print("Best RF Model parameters are {} with best accuracy
{}".format(gs_rf.best_params_,gs_rf.best_score_))

y_pred_rf=rf_est.predict(x_test)

cm_rf=pd.crosstab(y_test,y_pred_rf)

TN=cm_rf.iloc[0,0]

FP=cm_rf.iloc[0,1]

FN=cm_rf.iloc[1,0]

TP=cm_rf.iloc[1,1]
```

```
from sklearn.metrics import accuracy_score print("*****Random

Forest Evaluation Metrics********") print("Accuracy is

{}".format(accuracy_score(y_test,y_pred_rf)*100)) #negative cases

prediction

print("Specificity or True Negative Rate is {}".format(TN*100/(TN+FP)))

#postive cases prediction

print("Recall or sensitivity or True Postive Rate is {}".format(TP*100/(TP+FN)))

print("False Positive Rate is {}".format(FP*100/(FP+TN)))

print("False Negative Rate is {}".format(FN*100/(FN+TP)))

print("**************************************************")
```

In the above code, we have applied Random Forest on training set.

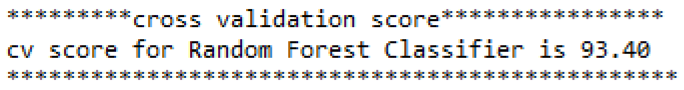
We have tuned all the parameters in Random Forest estimator with help of GridSearchCV and selected best estimator

Best Estimator:



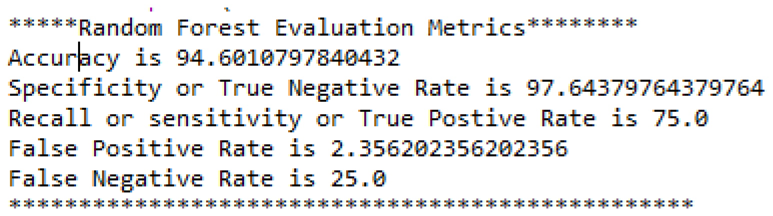
```
RandomForestClassifier(bootstrap=True, class_weight='balanced',
            criterion='entropy', max_depth=None, max_features='auto',
            max_leaf_nodes=None, min_impurity_decrease=0.0,
            min_impurity_split=None, min_samples_leaf=1,
            min_samples_split=10, min_weight_fraction_leaf=0,
            n_estimators=300, n_jobs=1, oob_score=False, random_state=None,
            verbose=0, warm_start=False)
```

Cross Validation Score:



```
*********cross validation score****************
cv score for Random Forest Classifier is 93.40
************************************************
```

Output of Random Forest:



```
*****Random Forest Evaluation Metrics********
Accuracy is 94.6010797840432
Specificity or True Negative Rate is 97.64379764379764
Recall or sensitivity or True Postive Rate is 75.0
False Positive Rate is 2.356202356202356
False Negative Rate is 25.0
*************************************************
```

4.4 KNN Classifier:

Let us apply KNN Classifier on the training set and test it with testing set

Code:

#KNN

from sklearn.neighbors import KNeighborsClassifier

KNN=KNeighborsClassifier()

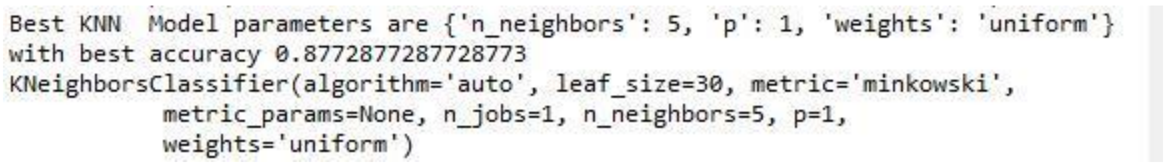
gs_KNN=GridSearchCV(KNN,cv=5,param_grid={'n_neighbors':[3,5,7,9,11,13,15],'weights':['uniform','distance'],'p':[1,2]})

gs_KNN.fit(x_train,y_train)

KNN_est=gs_KNN.best_estimator_

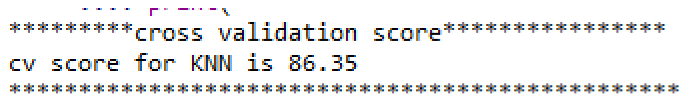
```
print("Best KNN Model parameters are {} with best accuracy
{}".format(gs_KNN.best_params_,gs_KNN.best_score_))

print(KNN_est)

y_pred_KNN=KNN_est.predict(x_test)

cm_KNN=pd.crosstab(y_test,y_pred_KNN)

TN=cm_KNN.iloc[0,0]

FP=cm_KNN.iloc[0,1]

FN=cm_KNN.iloc[1,0]

TP=cm_KNN.iloc[1,1]

from sklearn.metrics import accuracy_score

print("*****KNN Evaluation Metrics********")

print("Accuracy is {}".format(accuracy_score(y_test,y_pred_KNN)*100))

#negative cases prediction

print("Specificity or True Negative Rate is {}".format(TN*100/(TN+FP)))

#postive cases prediction

print("Recall or sensitivity or True Postive Rate is {}".format(TP*100/(TP+FN)))

print("False Positive Rate is {}".format(FP*100/(FP+TN)))

print("False Negative Rate is {}".format(FN*100/(FN+TP)))

print("****************************************************")


print("*********cross validation score***************")
from sklearn.cross_validation import cross_val_score
cv_score=np.mean(cross_val_score(lr_est,x_train,y_train,cv=5))
print("cv score for KNN is {:5.2f}".format(cv_score*100))
print("***********************************************")
```

Best Estimator:



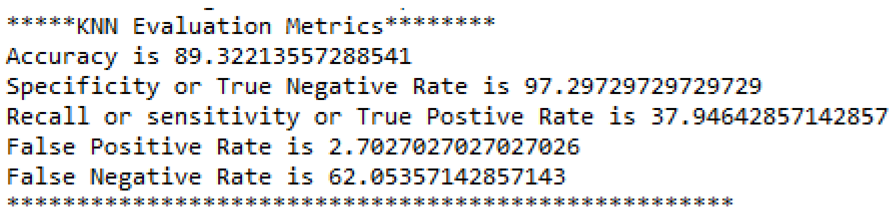
```
Best KNN  Model parameters are {'n_neighbors': 5, 'p': 1, 'weights': 'uniform'}
with best accuracy 0.8772877287728773
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=1, n_neighbors=5, p=1,
           weights='uniform')
```

Cross validation Score:



```
*********cross validation score****************
cv score for KNN is 86.35
***********************************************
```

Output of KNN:



```
*****KNN Evaluation Metrics********
Accuracy is 89.32213557288541
Specificity or True Negative Rate is 97.29729729729729
Recall or sensitivity or True Postive Rate is 37.94642857142857
False Positive Rate is 2.7027027027027026
False Negative Rate is 62.05357142857143
****************************************************
```

4.5 Naïve Bayes Classifier:

Let us apply Naïve Bayes Classifier on the training set and test it with testing set

Code:

```
#Naive Bayes

from sklearn.naive_bayes import GaussianNB

gn=GaussianNB()

gs_gn=GridSearchCV(gn,cv=5,param_grid={})

gs_gn.fit(x_train,y_train)

gn_est=gs_gn.best_estimator_

print("Best Naive Bayes Model parameters are {} with best
accuracy {}".format(gs_gn.best_params_,gs_gn.best_score_))

print(gn_est)

y_pred_gn=gn_est.predict(x_test)

cm_gn=pd.crosstab(y_test,y_pred_gn)

TN=cm_gn.iloc[0,0]

FP=cm_gn.iloc[0,1]

FN=cm_gn.iloc[1,0]
```

TP=cm_gn.iloc[1,1]

from sklearn.metrics import accuracy_score print("*****Naive Bayes

Evaluation Metrics********") print("Accuracy is

{}".format(accuracy_score(y_test,y_pred_gn)*100)) print(gn_est)

#negative cases prediction

print("Specificity or True Negative Rate is {}".format(TN*100/(TN+FP)))

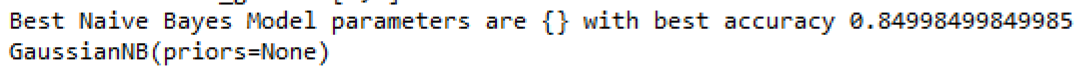
#postive cases prediction

print("Recall or sensitivity or True Postive Rate is {}".format(TP*100/(TP+FN)))
print("False Positive Rate is {}".format(FP*100/(FP+TN))) print("False Negative Rate is {}".format(FN*100/(FN+TP)))
print("***")

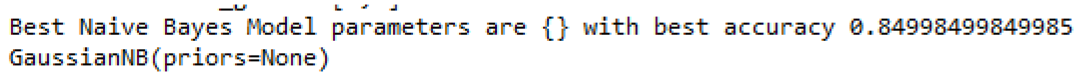
print("*********cross validation score***************")
from sklearn.cross_validation import cross_val_score
cv_score=np.mean(cross_val_score(gn_est,x_train,y_train,cv=5))
print("cv score for Naive Bayes Classifier is {:5.2f}".format(cv_score*100))
print("***")

In the above code, we have applied Naïve Bayes Classifier on training set.

We have tuned all the parameters in Naïve Bayes estimator with help of GridSearchCV and selected best estimator.

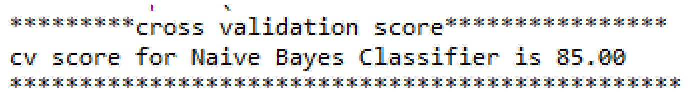


```
Best Naive Bayes Model parameters are {} with best accuracy 0.84998499849985
GaussianNB(priors=None)
```



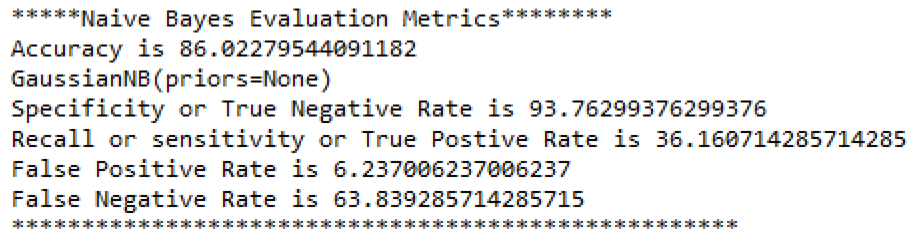
```
Best Naive Bayes Model parameters are {} with best accuracy 0.84998499849985
GaussianNB(priors=None)
```

Cross Validation Score:



```
*********cross validation score***************
cv score for Naive Bayes Classifier is 85.00
***********************************************
```

Output of Naïve Bayes:



```
*****Naive Bayes Evaluation Metrics********
Accuracy is 86.02279544091182
GaussianNB(priors=None)
Specificity or True Negative Rate is 93.76299376299376
Recall or sensitivity or True Postive Rate is 36.160714285714285
False Positive Rate is 6.237006237006237
False Negative Rate is 63.839285714285715
****************************************************
```

Chapter 5

Selecting the best model

To select a best model, we have to compare the performance of the models.

So we need evaluation metrics to evaluate the results of the models and select the best model which gives best results.

5.1 Evaluation Metrics:

To understand the performance of the model, first let us check accuracies of different models.

Python:

Accuracy of Random Forest=94.6

Accuracy of Logistic Regression=87.34

Accuracy of Naive Bayes Classifier=86.022

Accuracy of KNN=89.32

Accuracy of Decision Tree=86.02

Clearly we can see that random forest has highest accuracy.

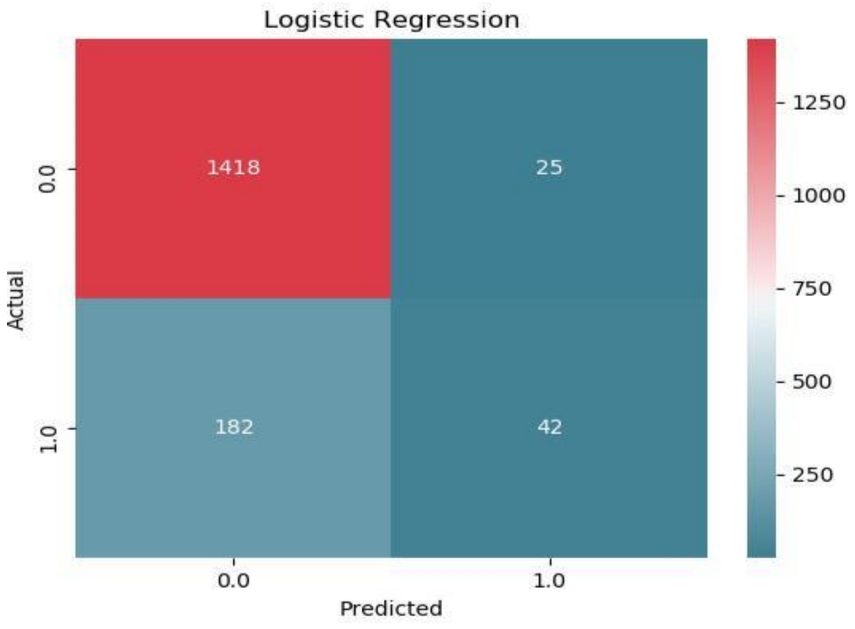
Is Accuracy the right/only metric to evaluate the performance of the model?

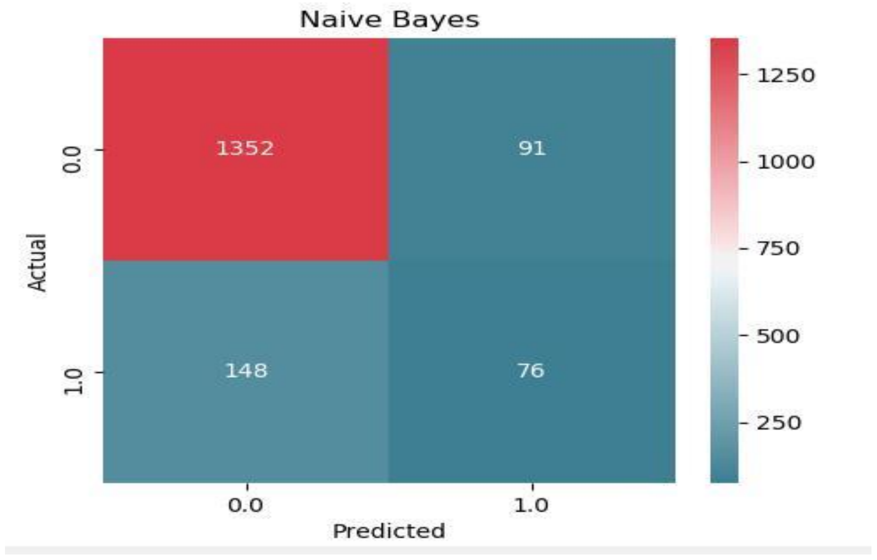
Definitely no.

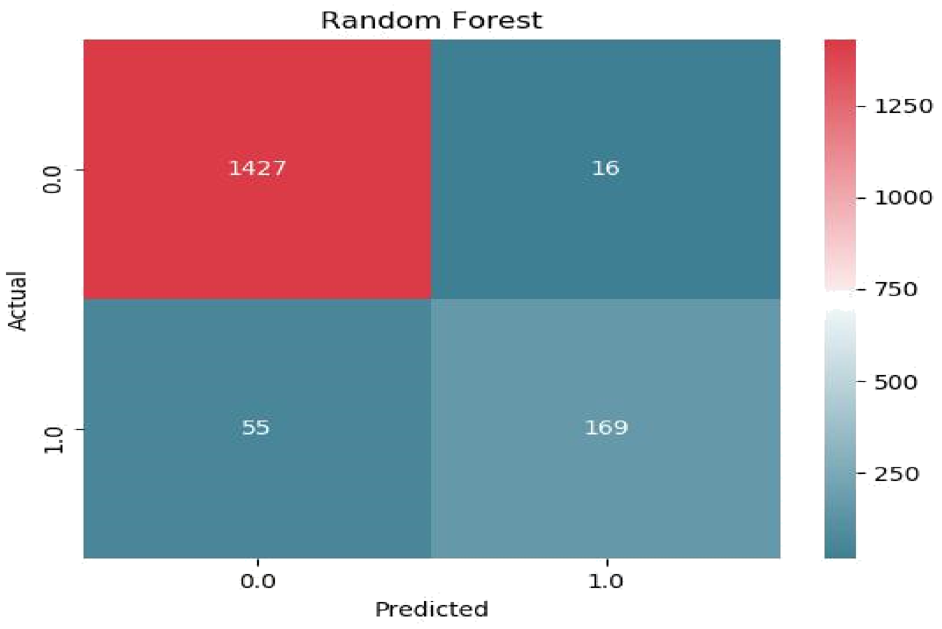
There are several other metrics we have to calculate to evaluate the performance of the model.

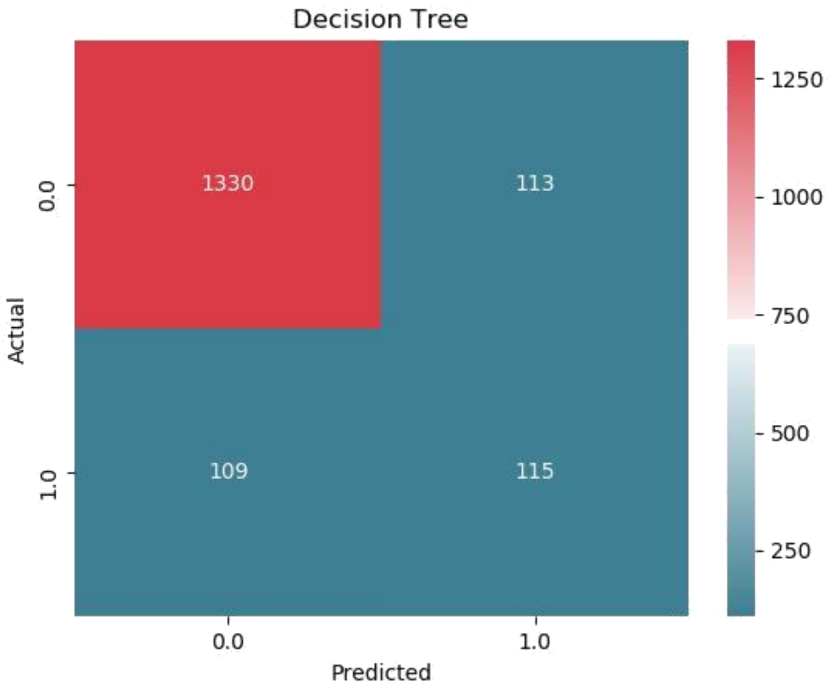
Let us first construct confusion matrix and let us calculate the evaluation metrics.

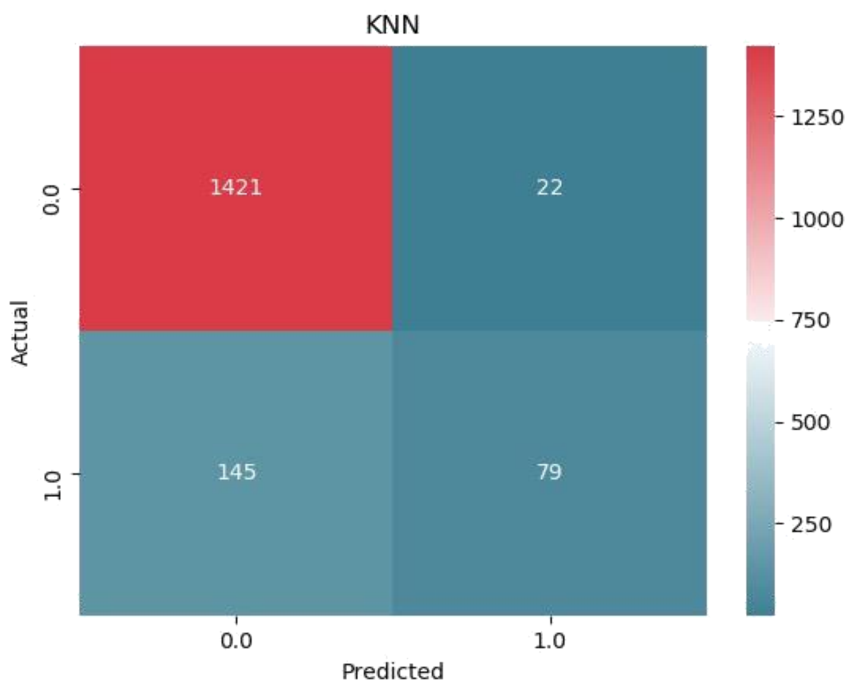
5.2 Confusion Matrix:






Random Forest
1427
16
55
169
0.0
1.0
Actual
0.0
1.0
Predicted
1250
1000
750
500
250


Decision Tree
1330
113
109
115
0.0
1.0
Actual
0.0
1.0
Predicted
1250
1000
750
500
250



We have constructed confusion matrices of all models. Now let us understand evaluation metrics according to problem statement.

True Positive (TP) as a customer that is predicted to churn and actually churns

True Negative (TN) as a customer that is predicted to not churn and that doesn't churn

False Positive (FP) as a customer that is predicted to churn and actually doesn't churn

False Negative (FN) as a customer that is predicted to not churn but actually churns

5.3 Business Understanding:

Let us understand these metrics from business point of view.

Let us consider life time value for customer is 1000/year.83.3/month

If that customer has quit the company it means 1000rs loss for the company.

When the customer is predicted as churn, we offer him 100rs offer to make him stay.

It is better to lose 100 than 1000.Not all the customers who accept offer will stay.

Let us consider there is 50% chance that customer stays after receiving the offer

0.5*1000+0.5*100=550

With the help of Business understanding let us construct cost matrix

TP costs the company 550
TN costs the company 0
FP costs the company 100
FN costs the company 1,000

Cost Matrix:

	F	**T**
F	**TN (0)**	**FP (100)**
T	**FN (1000)**	**TP (550)**

From these cost matrix, we can say that we lose 1000 INR for every 1% increase in False Negative Rate.

We are losing 1000 INR if we predict the customer is not going to churn even if he is actually going to churn out.

So our model should have less False Negative rate.

First Preference-False Negative Rate

Second Preference-Accuracy.

We have calculated Accuracies, cross validation scores and evaluation metrics in modelling phase.

Let us have a look at them

Logistic Regression:
Accuracy: 87.34
Cross Validation Score: 85
False Negative Rate: 84.48

Decision Trees:
Accuracy: 92.80
Cross Validation Score: 86.32
False Negative Rate: 29.46

Random Forest:
Accuracy: 94.6
Cross Validation Score: 93.40
False Negative Rate: 25

KNN:
Accuracy: 89.32
Cross Validation Score: 86.35
False Negative Rate: 62.05

Naive Bayes:
Accuracy: 86.022
Cross Validation Score: 85
False Negative Rate: 63.83

We can see that Random Forest dominates the rest in accuracy (94.6%), False Negative Rate (25%) and cross validation score (93.40%)

So we can freeze Random Forest as our model.

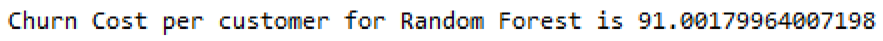
5.4 Churn Cost per Customer:

With the help of the model, we have successfully predicted and we have calculated metrics, confusion matrix and cost matrix

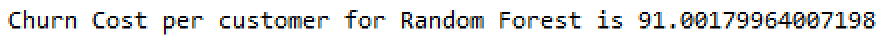
We have defined a function which will calculate Churn Cost per Customer(ccpc)

```
from sklearn.metrics import confusion_matrix

cost_mat=np.array([[0,100],[1000,550]])

def churn_cost_per_customer(y,y1):
        confusion_matrix_cost=confusion_matrix(y,y1)*cost_mat
        return np.sum(confusion_matrix_cost)/float(len(y))

print("Churn Cost per customer for Random Forest is
{}".format(churn_cost_per_customer(y_test,y_pred_rf)))
```

Let us calculate ccpc with the help of this function



```
Churn Cost per customer for Random Forest is 91.00179964007198
```



```
Churn Cost per customer for Random Forest is 91.00179964007198
```

Inference: It means our company has to spend 91 INR for every customer average instead of losing 1000 INR, 550 INR, 100 INR on every customer which is pretty good.

5.5 Improving the model to Reduce loss for the company:

We can still improve the model to reduce total churn cost per customer.

We have already seen that this unbalanced dataset.

Total observations: 5000

“True”: 707 = 14.14%

“False”: 4293 = 85.86%

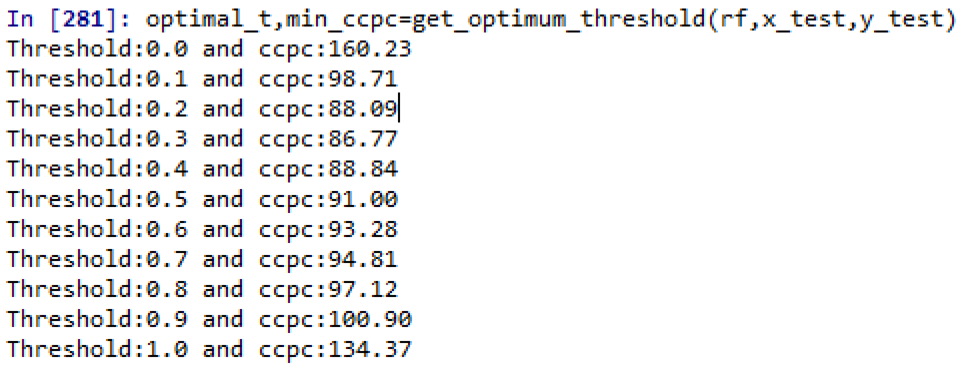
So probability of 0.5 may give less ccpc

So to lower ccpc, we have to vary all probability Thresholds so that we can get low ccpc

Let us define the function to get optimal threshold to get lower ccpc.

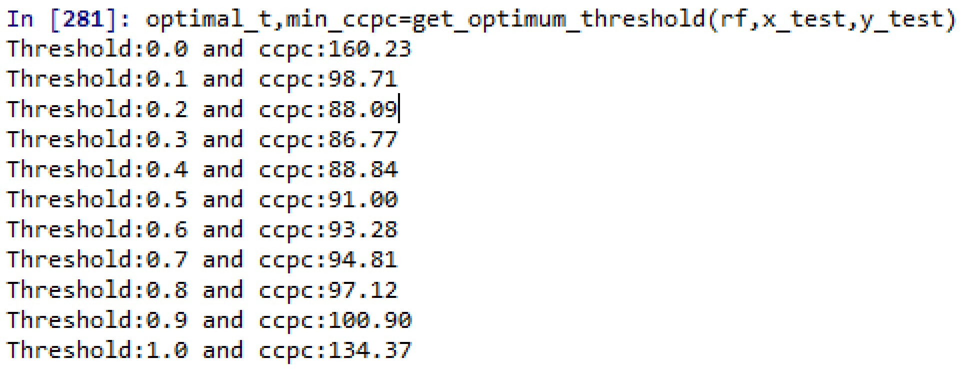
Code:

```
def get_optimum_threshold(est,x,y):
    min_cost,optimal_t=np.max(cost_mat),0
    for t in np.arange(0,1.01,0.1):
        y_predict_threshold=(est.predict_proba(x)[:,1]>t).astype(int)
        ccpc=churn_cost_per_customer(y,y_predict_threshold)
        print("Threshold:{0:3.1f} and ccpc:{1:5.2f}".format(t,ccpc)) if
        ccpc<min_cost:
            min_cost=ccpc
            optimal_t=t
    return optimal_t,min_cost
optimal_t,min_ccpc=get_optimum_threshold(rf,x_test,y_test)
```



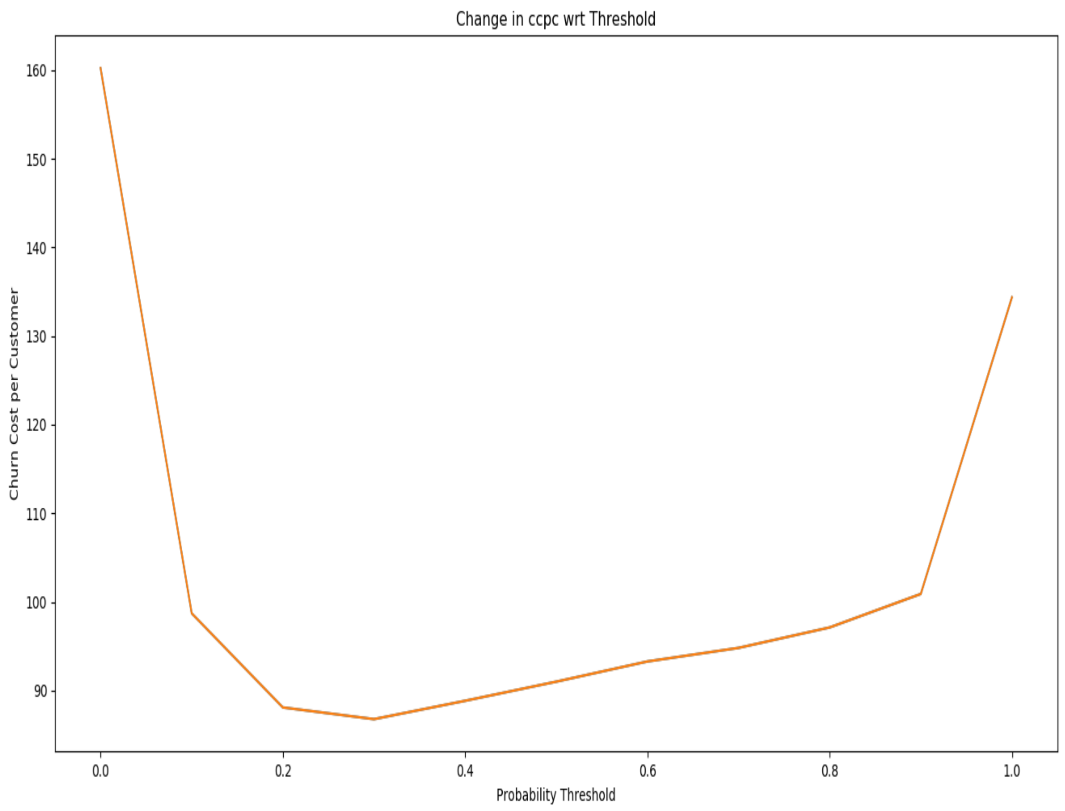
```
In [281]: optimal_t,min_ccpc=get_optimum_threshold(rf,x_test,y_test)
Threshold:0.0 and ccpc:160.23
Threshold:0.1 and ccpc:98.71
Threshold:0.2 and ccpc:88.09
Threshold:0.3 and ccpc:86.77
Threshold:0.4 and ccpc:88.84
Threshold:0.5 and ccpc:91.00
Threshold:0.6 and ccpc:93.28
Threshold:0.7 and ccpc:94.81
Threshold:0.8 and ccpc:97.12
Threshold:0.9 and ccpc:100.90
Threshold:1.0 and ccpc:134.37
```

pc:

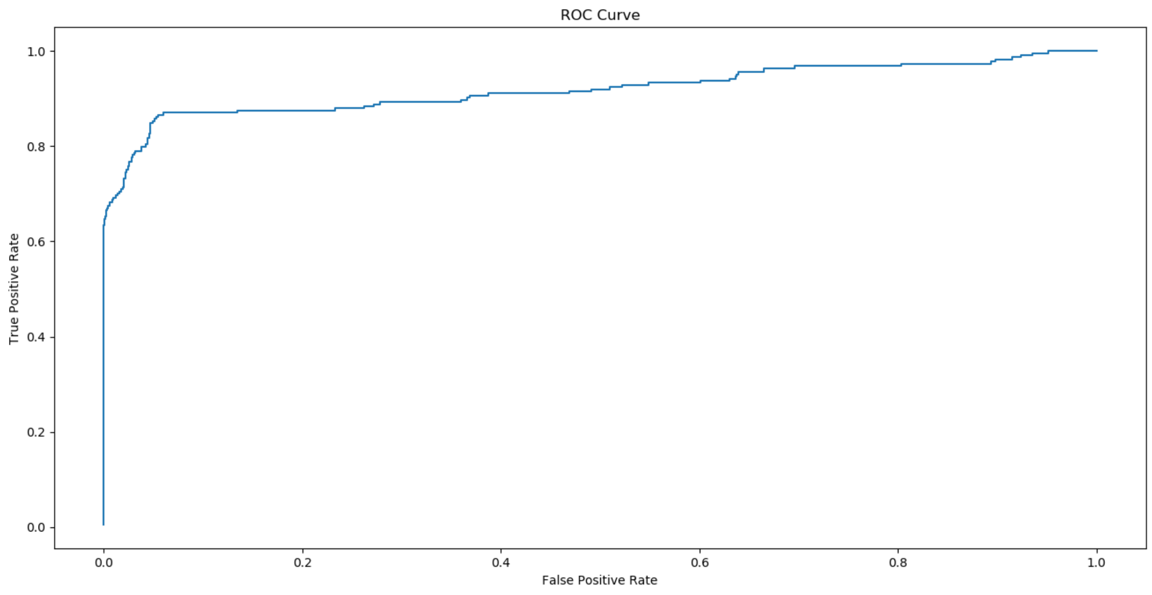


```
In [281]: optimal_t,min_ccpc=get_optimum_threshold(rf,x_test,y_test)
Threshold:0.0 and ccpc:160.23
Threshold:0.1 and ccpc:98.71
Threshold:0.2 and ccpc:88.09
Threshold:0.3 and ccpc:86.77
Threshold:0.4 and ccpc:88.84
Threshold:0.5 and ccpc:91.00
Threshold:0.6 and ccpc:93.28
Threshold:0.7 and ccpc:94.81
Threshold:0.8 and ccpc:97.12
Threshold:0.9 and ccpc:100.90
Threshold:1.0 and ccpc:134.37
```

CCPC vs Probability Threshold:



ROC Curve for the model:



We can see that by changing the threshold values, there is change in estimator which estimates **"lower churn cost per customer"**

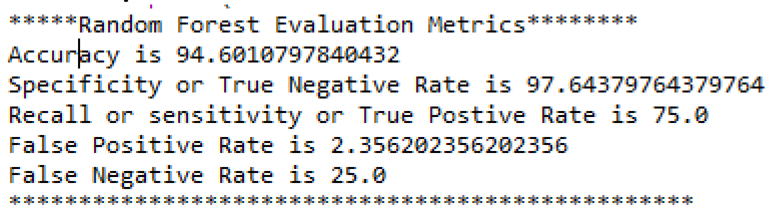
With the above results, we have optimized the model to predict low ccpc (91 to 86 INR) with probability threshold 0.3 and churn cost per customer 86.77

5.6 Predicting with optimized model:

We have successfully optimized the model

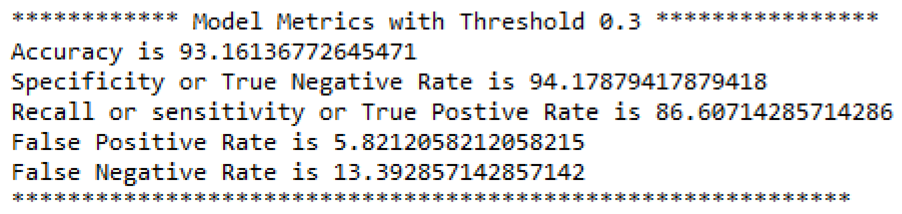
Let us see the evaluation metrics and output of optimized model.

Before Optimization:



```
*****Random Forest Evaluation Metrics********
Accuracy is 94.6010797840432
Specificity or True Negative Rate is 97.64379764379764
Recall or sensitivity or True Postive Rate is 75.0
False Positive Rate is 2.356202356202356
False Negative Rate is 25.0
*************************************************
```

After optimization:

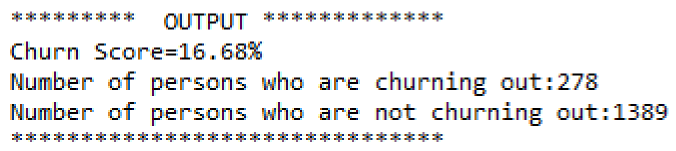


```
************ Model Metrics with Threshold 0.3 ****************
Accuracy is 93.16136772645471
Specificity or True Negative Rate is 94.17879417879418
Recall or sensitivity or True Postive Rate is 86.60714285714286
False Positive Rate is 5.8212058212058215
False Negative Rate is 13.392857142857142
************************************************************
```

We can see that we can be able to reduce false negative rate to 13% and also Churn Cost per Customer.

Chapter 6

Conclusion:



```
*********  OUTPUT *************
Churn Score=16.68%
Number of persons who are churning out:278
Number of persons who are not churning out:1389
*******************************
```

With the help of above estimator, we estimated that churn score is 16.68% which means 16.68% customers are about to churn out from the company.

With this project, It is proven that how important it is to use the cost reduction of customer churns as a performance measure of the estimator, rather than just using the accuracy of the churn estimator. In a business objective point of view, using cost reduction takes more sense.

Instead of losing thousands of rupees on each customer, we spend hundreds on each customer to reduce losses of the company

References:

R Code(Working with different models)

#Modelling

Best Model: Decision tree

False Negative Rate:11.3%

Accuracy: 98%

#Decision Tree

```
library(stats)

library(C50)

#Tuned Decision Tree,ran the model with 100,200,300,500,1000 trails,all are giving
same accuracies

dt=C50::C5.0(Churn ~ .,train,rules=TRUE,trials=100)

y_pred=predict(dt,test[,-16],class=TRUE)


#Model Evaluation

library(base)

library(ModelMetrics)

cm_c50=table(test$Churn,y_pred)

TN=cm_c50[1,1]

FP=cm_c50[1,2]

FN=cm_c50[2,1]
```

```
TP=cm_c50[2,2]
cat("*****Decision Tree Evaluation Metrics********")
cat("Accuracy is:",(TN+TP)/(TN+TP+FP+FN))
#negative cases prediction
cat("Specificity or True Negative Rate is ",(TN*100/(TN+FP)))
#postive cases prediction
cat("Recall or sensitivity or True Postive Rate is ",(TP*100/(TP+FN)))
cat("False Positive Rate is ",(FP*100)/(FP+TN))
cat("False Negative Rate is ",(FN*100)/(FN+TP))
cat("**************************************************")
```

False Negative Rate: 0%(Over fitted model)

Accuracy: 100%

#Random Forest

```
library(randomForest)
rf=randomForest(Churn ~ .,train,ntree=200,importance=TRUE)
y_pred=predict(rf,test[,-16],class=TRUE)

#Model Evaluation
cm_rf=table(test$Churn,y_pred)
TN=cm_rf[1,1]
FP=cm_rf[1,2]
FN=cm_rf[2,1]
TP=cm_rf[2,2]
```

```
cat("*****Random Forest Evaluation Metrics********")

cat("Accuracy is:",(TN+TP)/(TN+TP+FP+FN))

#negative cases prediction

cat("Specificity or True Negative Rate is ",(TN*100/(TN+FP)))

#postive cases prediction

cat("Recall or sensitivity or True Postive Rate is ",(TP*100/(TP+FN)))

cat("False Positive Rate is ",(FP*100)/(FP+TN))

cat("False Negative Rate is ",(FN*100)/(FN+TP))

cat("**************************************************")

summary(rf)

control <- trainControl(method="repeatedcv", number=10, repeats=3, search="grid")

set.seed(seed)

tunegrid <- expand.grid(.mtry=c(1:15))

rf_gridsearch <- train(Churn~., data=train, method="rf", metric=metric,
tuneGrid=tunegrid, trControl=control)

print(rf_gridsearch)

plot(rf_gridsearch)
```

False Negative Rate:0

Accuracy: 100 %

#KNN Classifier (over fitted model)

```
library(class)

knn_pred=knn(train[,-16],test[,-16],train$Churn,k=1)

cm_knn=table(test$Churn,knn_pred)

TN=cm_knn[1,1]

FP=cm_knn[1,2]

FN=cm_knn[2,1]
```

```
TP=cm_knn[2,2]

cat("*****KNN Evaluation Metrics********")

cat("Accuracy is:",(TN+TP)/(TN+TP+FP+FN))

#negative cases prediction

cat("Specificity or True Negative Rate is ",(TN*100/(TN+FP)))

#postive cases prediction

cat("Recall or sensitivity or True Postive Rate is ",(TP*100/(TP+FN)))

cat("False Positive Rate is ",(FP*100)/(FP+TN))

cat("False Negative Rate is ",(FN*100)/(FN+TP))

cat("**************************************************")
```

False Negative Rate: 75%

Accuracy: 87%

#Naive Bayes Classifier

```
library(e1071)

nb_model=naiveBayes(Churn ~ .,train)

nb_pred=predict(nb_model,test[,-16],type='class')

cm_nb=table(test$Churn,nb_pred)

TN=cm_nb[1,1]

FP=cm_nb[1,2]

FN=cm_nb[2,1]

TP=cm_nb[2,2]

cat("*****Naive Bayes Evaluation Metrics********")

cat("Accuracy is:",(TN+TP)/(TN+TP+FP+FN)) #negative

cases prediction

cat("Specificity or True Negative Rate is ",(TN*100/(TN+FP)))

#postive cases prediction
```

```
cat("Recall or sensitivity or True Postive Rate is ",(TP*100/(TP+FN)))

cat("False Positive Rate is ",(FP*100)/(FP+TN))

cat("False Negative Rate is ",(FN*100)/(FN+TP))

cat("**************************************************")
```

False Negative Rate: 85%

Accuracy: 75%

#Logistic Regression

```
library(e1071)

glm_model=glm(Churn ~ .,train,family = 'binomial')

glm_pred=predict(glm_model,test[,-16],type='response')

glm_pred=ifelse(glm_pred>0.4,2,1)

cm_glm=table(test$Churn,glm_pred)

TN=cm_glm[1,1]

FP=cm_glm[1,2]

FN=cm_glm[2,1]

TP=cm_glm[2,2]

cat("*****Logistic Regression Metrics********")

cat("Accuracy is:",(TN+TP)/(TN+TP+FP+FN))

#negative cases prediction

cat("Specificity or True Negative Rate is ",(TN*100/(TN+FP)))

#postive cases prediction

cat("Recall or sensitivity or True Postive Rate is ",(TP*100/(TP+FN)))

cat("False Positive Rate is ",(FP*100)/(FP+TN))

cat("False Negative Rate is ",(FN*100)/(FN+TP))
cat("***************************************************")
```

***************** Thank You *****************************