Project

Churn Reduction

Aditya Tiwari
Adi138250@gmail.com

Contents

1. Introduction

1.1 Problem Statement

The objective of this Case is to predict customer behavior. We are providing you a public dataset that has customer usage pattern and if the customer has moved or not. It is expected to develop an algorithm to predict the churn score based on usage pattern.

1.2 Data

The predictors provided are as follows:

- account length
- international plan
- voicemail plan
- number of voicemail messages
- total day minutes used
- day calls made
- total day charge
- total evening minutes
- total evening calls
- total evening charge
- total night minutes
- total night calls
- total night charge
- total international minutes used
- total international calls made
- total international charge
- number of customer service calls made

Target Variable :

move: if the customer has moved (1=yes; 0 = no)

Table : Churn Reduction Sample Data (Columns: 1-8)

state	account length	area code	phone number	international plan	voice mail plan	number vmail messages	total day minutes
KS	128	415	382-4657	no	yes	25	265.1
OH	107	415	371-7191	no	yes	26	161.6
NJ	137	415	358-1921	no	no	0	243.4

Table : Churn Reduction Sample Data (Columns: 9-17)

total day calls	total day charge	total eve minutes	total eve calls	total eve charge	total night minutes	total night calls	total night charge	total intl minutes
110	45.07	197.4	99	16.78	244.7	91	11.01	10
123	27.47	195.5	103	16.62	254.4	103	11.45	13.7
114	41.38	121.2	110	10.3	162.6	104	7.32	12.2

Table : Churn Reduction Sample Data (Columns: 17-21)

total intl calls	total intl charge	number customer service calls	Churn
3	2.7	1	False.
3	3.7	1	False.
5	3.29	0	False.

Below are the variables present in Churn Reduction dataset

Table: Churn Reduction

S.no	Variables
1	state
2	account length
3	area code
4	phone number
5	international plan
6	voice mail plan
7	number vmail messages
8	total day minutes
9	total day calls
10	total day charge
11	total eve minutes
12	total eve calls
13	total eve charge
14	total night minutes
15	total night calls
16	total night charge
17	total intl minutes
18	total intl calls
19	total intl charge
20	number customer service calls
21	Churn

2. Methodology

2.1 Pre-Processing

Any predictive modeling requires that we look at the data before we start modeling. We decided to simply remove few variables after loading data set but here we have dropped few variables after correlation test for continuous variables and chi square test for categorical variables. Since our target variable is classified i.e. categorical and few independent variables which are also categorical hence we have applied categorical test.

However, in data mining terms looking at data refers to so much more than just looking. Looking at data refers to exploring the data, cleaning the data as well as visualizing the data through graphs and plots. This is often called as Exploratory Data Analysis. To start this process, we will first check the presence of missing values in data set.

2.1.1 Missing Value Analysis

Missing values Analysis is required to be done so that we can check if there is any missing data. In case data is missing at few places we will impute those missing values by different methods in order to generate appropiate results. In our case we have zero missing values hence no imputation is required. Below table illustrate missing values present in variables of the dataset.

S.no	Variables	Number of Missing Values
1	state	0
2	account length	0
3	area code	0
4	phone number	0
5	international plan	0
6	voice mail plan	0
7	number vmail messages	0
8	total day minutes	0
9	total day calls	0
10	total day charge	0
11	total eve minutes	0
12	total eve calls	0
13	total eve charge	0
14	total night minutes	0
15	total night calls	0
16	total night charge	0
17	total intl minutes	0
18	total intl calls	0
19	total intl charge	0

20	number customer service calls	0
21	Churn	0

In above table we can clearly see that we don’t have any missing values present in any of the variables of dataset. So, there is no imputation is required and we will move to outlier analysis.

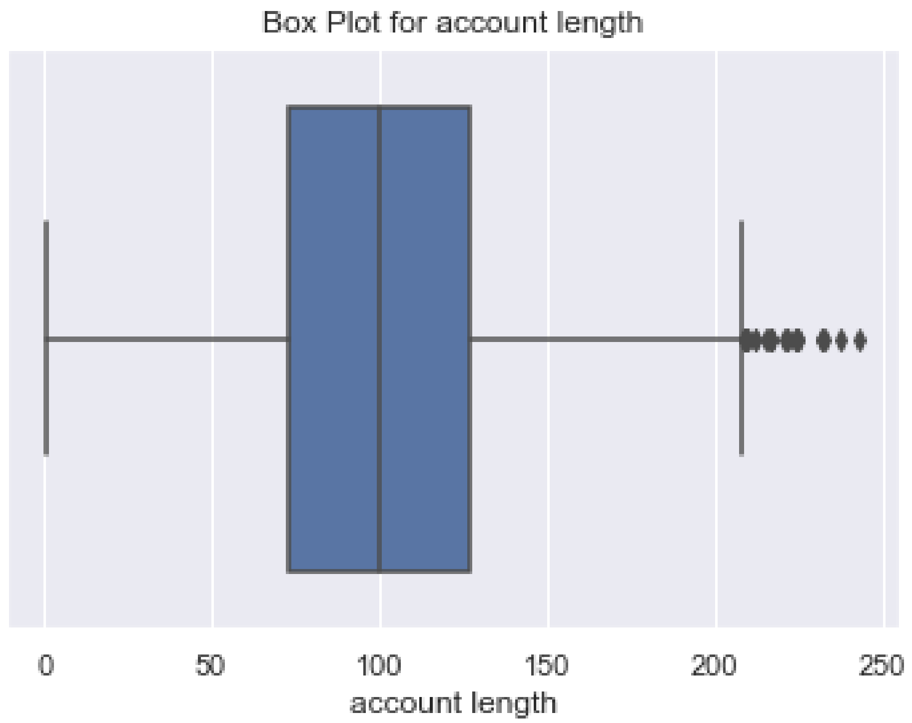
2.1.2 Outlier Analysis

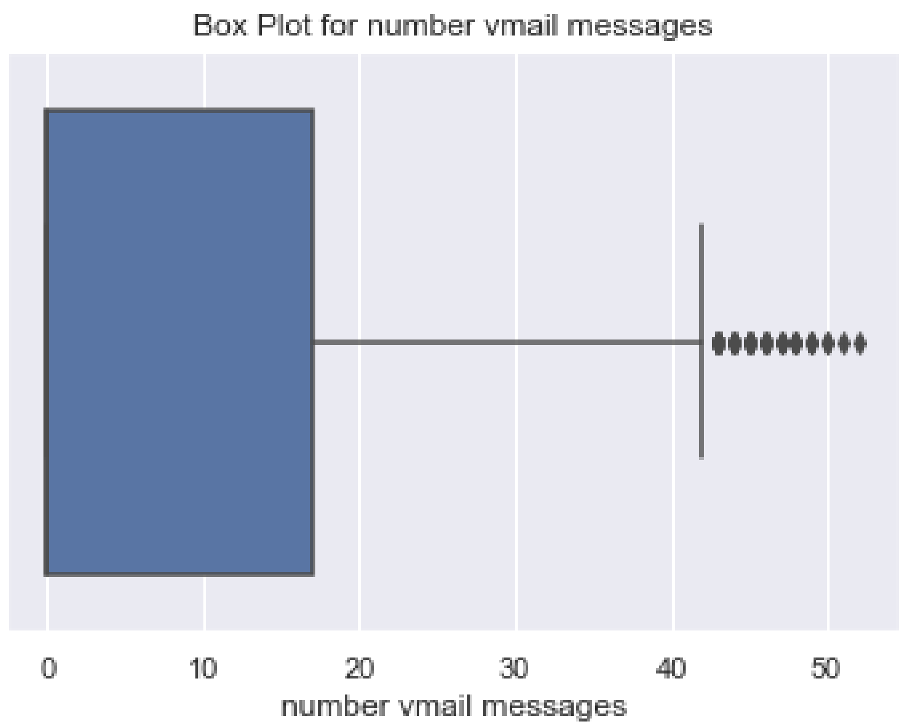
The Other steps of Preprocessing Technique is Outliers analysis, an outlier is an observation point that is distant from other observations. Outliers in data can be good and it can be bad as well. Here in our case we there are outliers present. So we will not remove these outliers instead we will be substituting them with other balancing values (such as mean, median or knn method) because we expect them to be relatively random values and replacing them with set values may cause inaccuracy in analysis later. The outlier analysis is done by plotting the box plot. Boxplot is a method for graphically depicting groups of numerical data through their quartiles. Box plots may also have lines extending vertically from the boxes (whiskers) indicating variability outside the upper and lower quartiles.

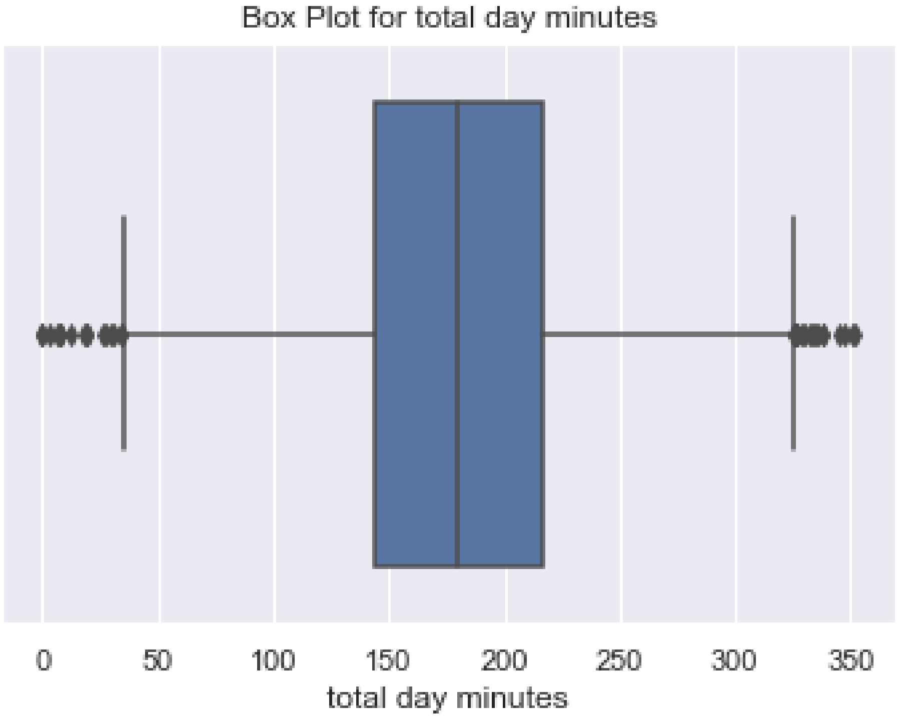
Fig. 1.0

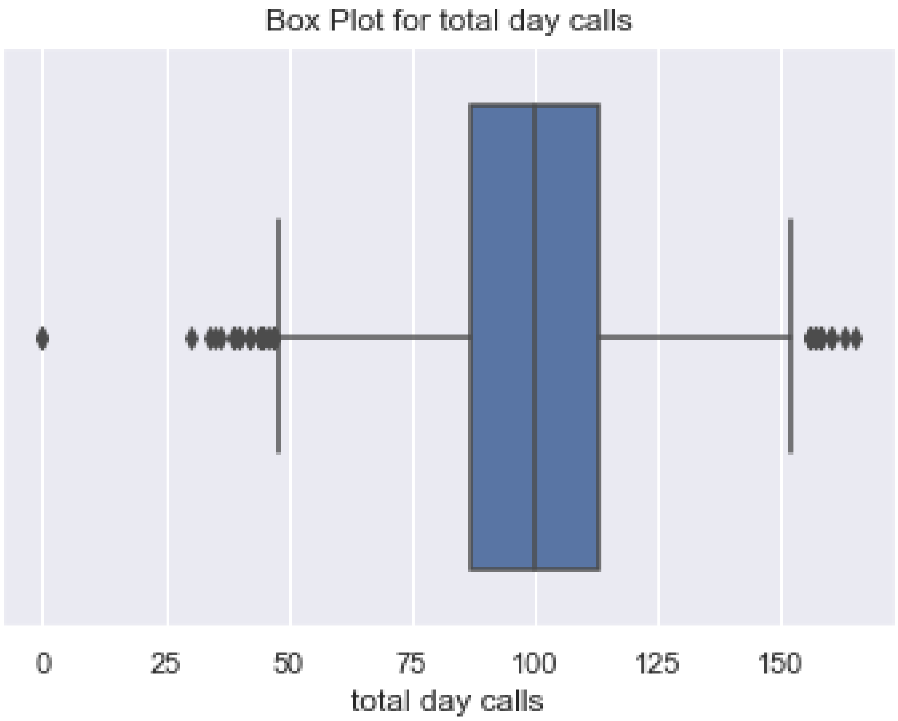
Outlier Analysis

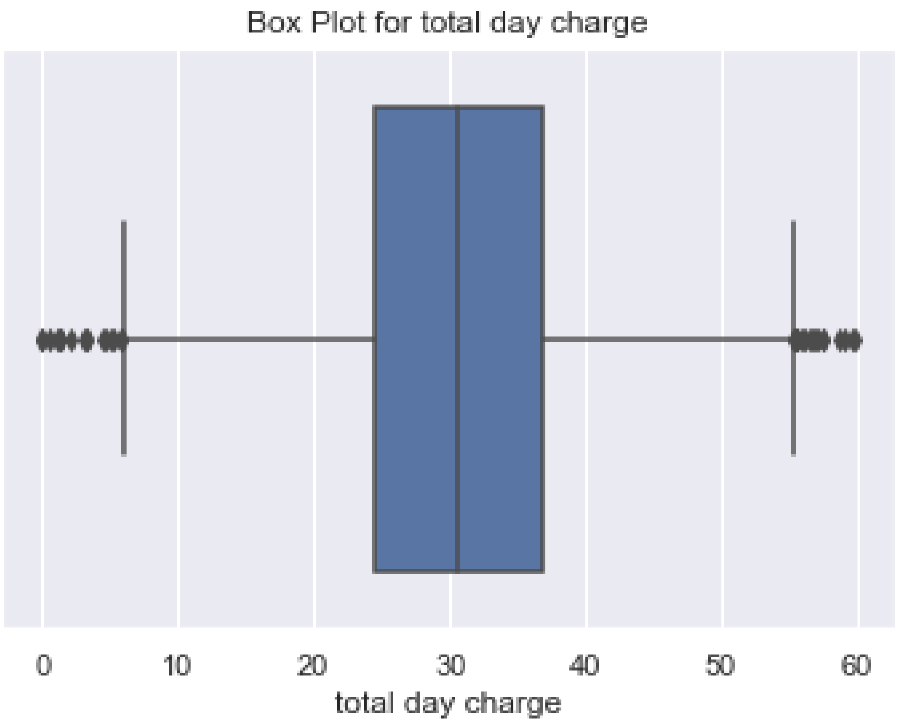
```
#Outlier analysis using box plot method.
sns.set()
for i in variable_num:
    sns.boxplot(churn_red[i])
    plt.title("Box Plot for "+str(i))
    plt.show()
```

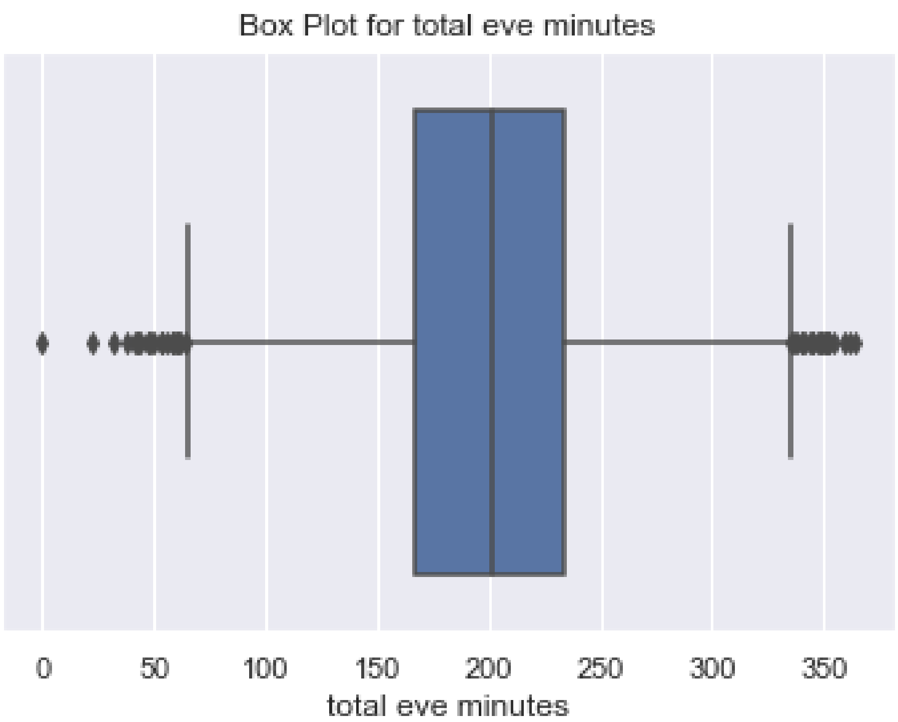

Box Plot for account length
0
50
100
150
200
250
account length

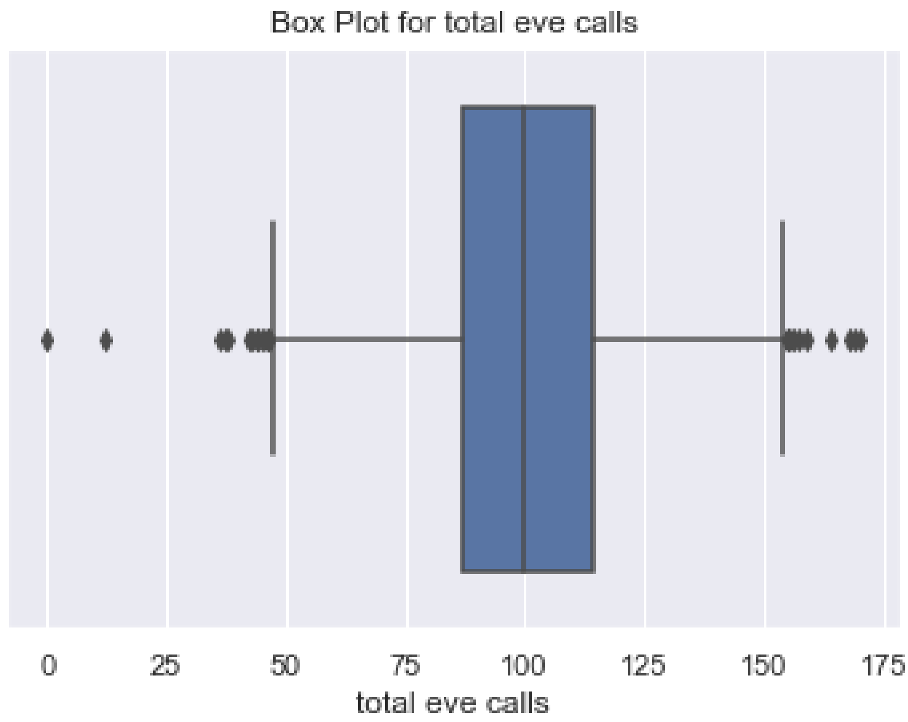

Box Plot for number vmail messages
0
10
20
30
40
50
number vmail messages

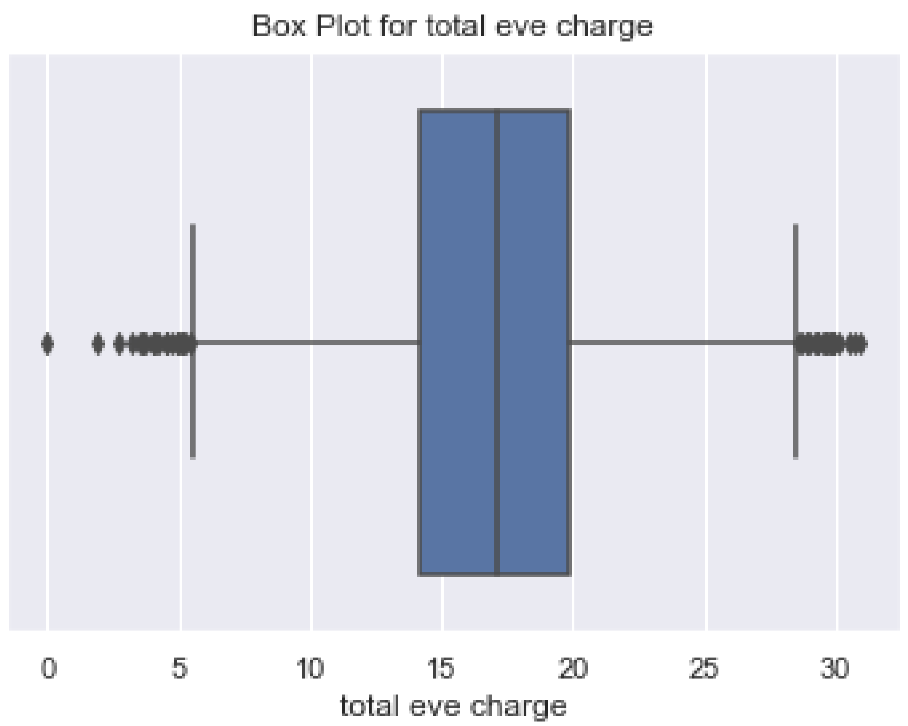

Box Plot for total day minutes
0
50
100
150
200
250
300
350
total day minutes

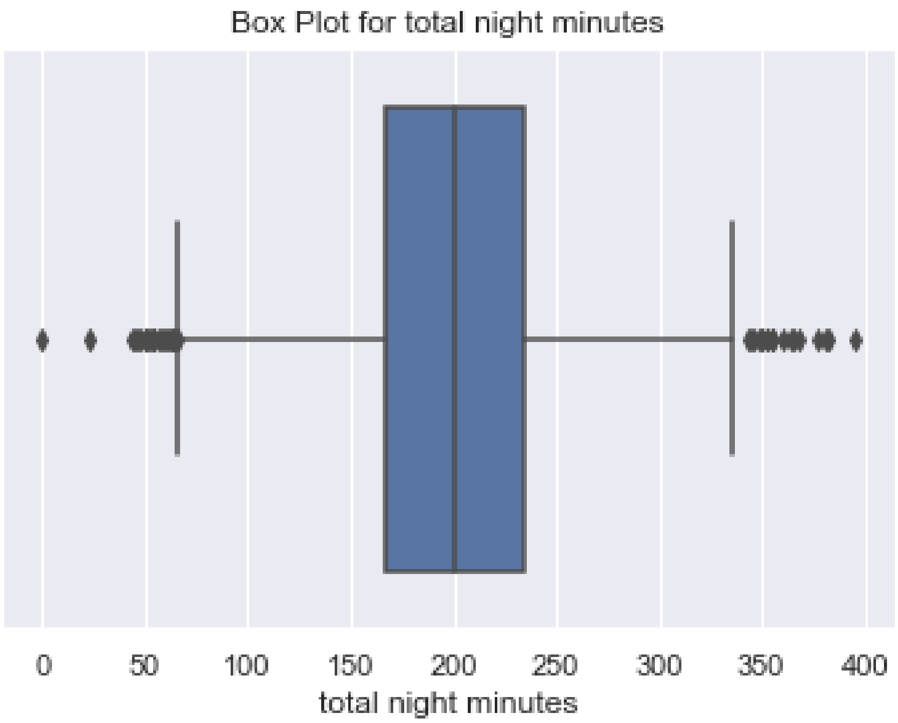

Box Plot for total day calls
0
25
50
75
100
125
150
total day calls

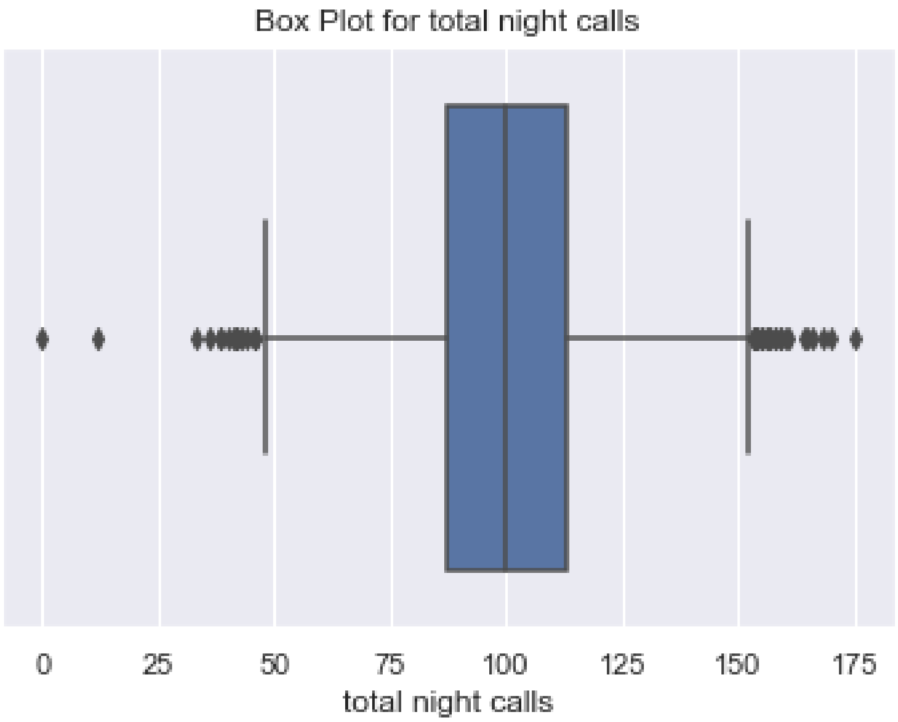

Box Plot for total day charge
0
10
20
30
40
50
60
total day charge

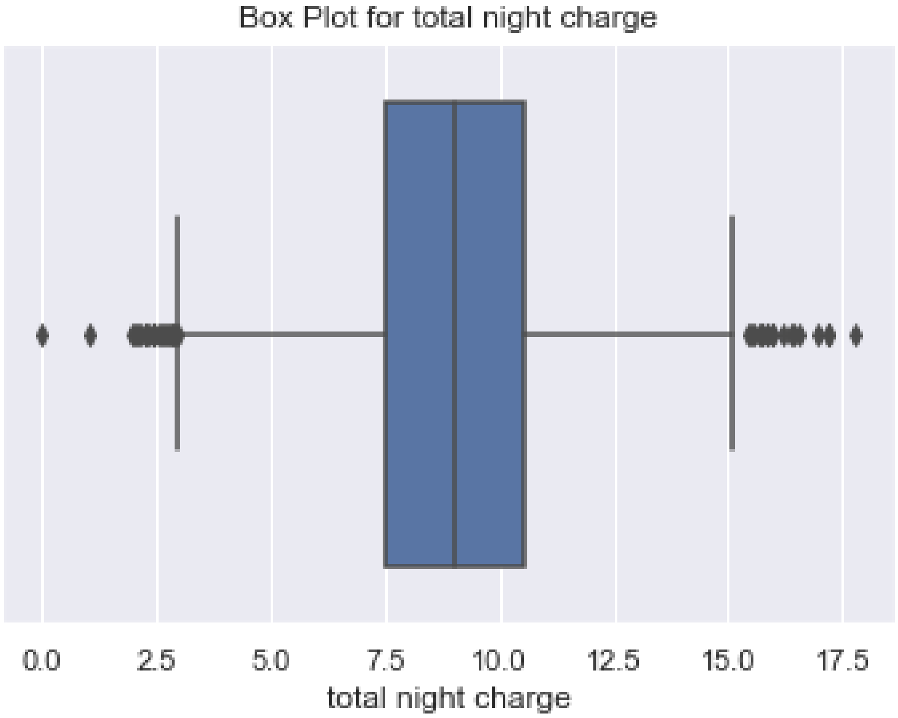

Box Plot for total eve minutes
0
50
100
150
200
250
300
350
total eve minutes

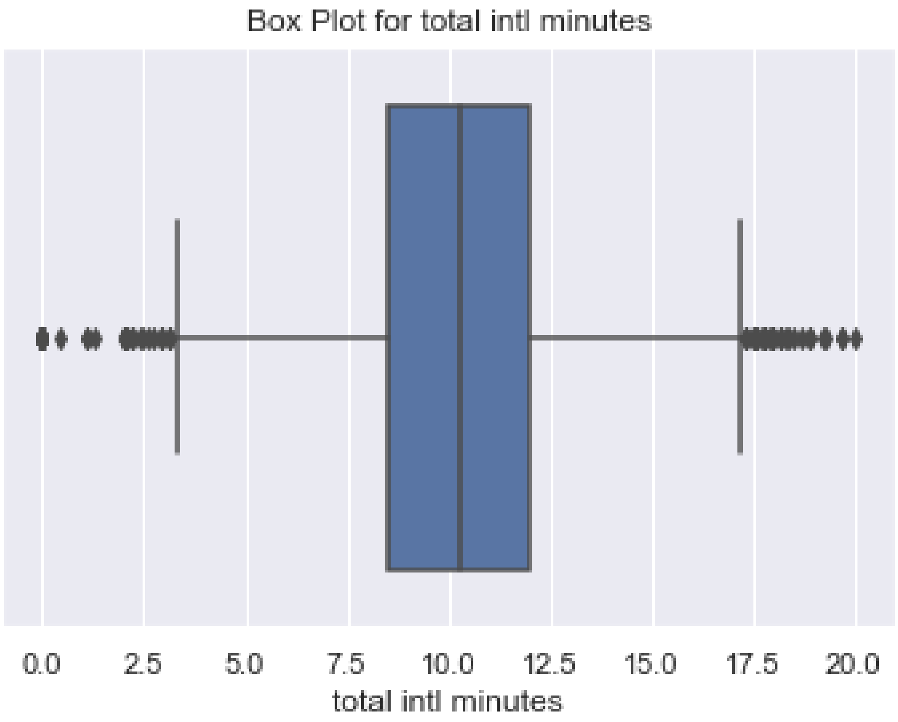

Box Plot for total eve calls
0
25
50
75
100
125
150
175
total eve calls

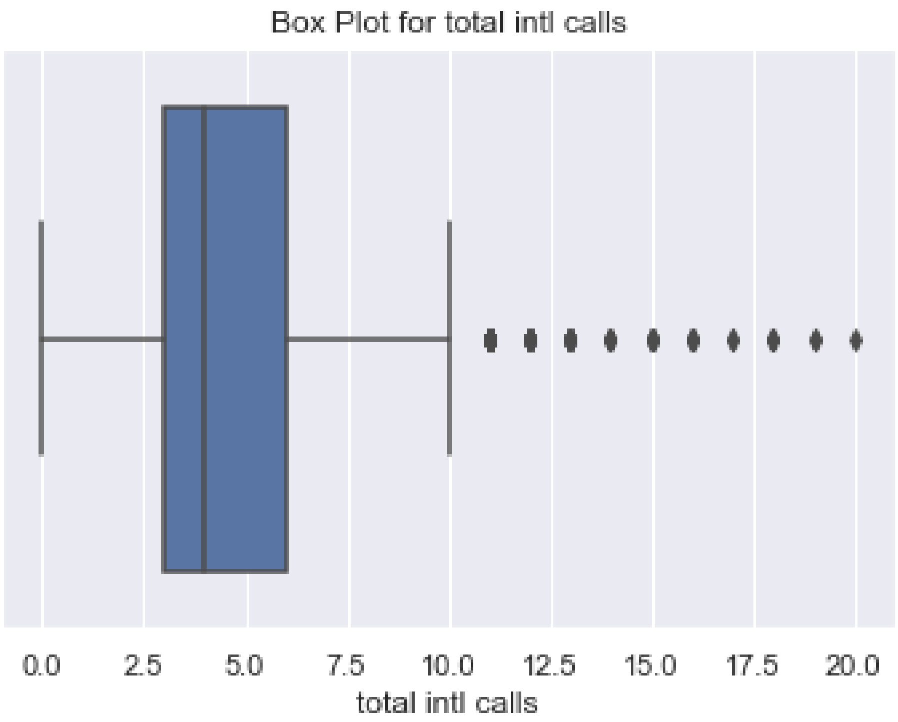

Box Plot for total eve charge
0
5
10
15
20
25
30
total eve charge


Box Plot for total night minutes
0
50
100
150
200
250
300
350
400
total night minutes

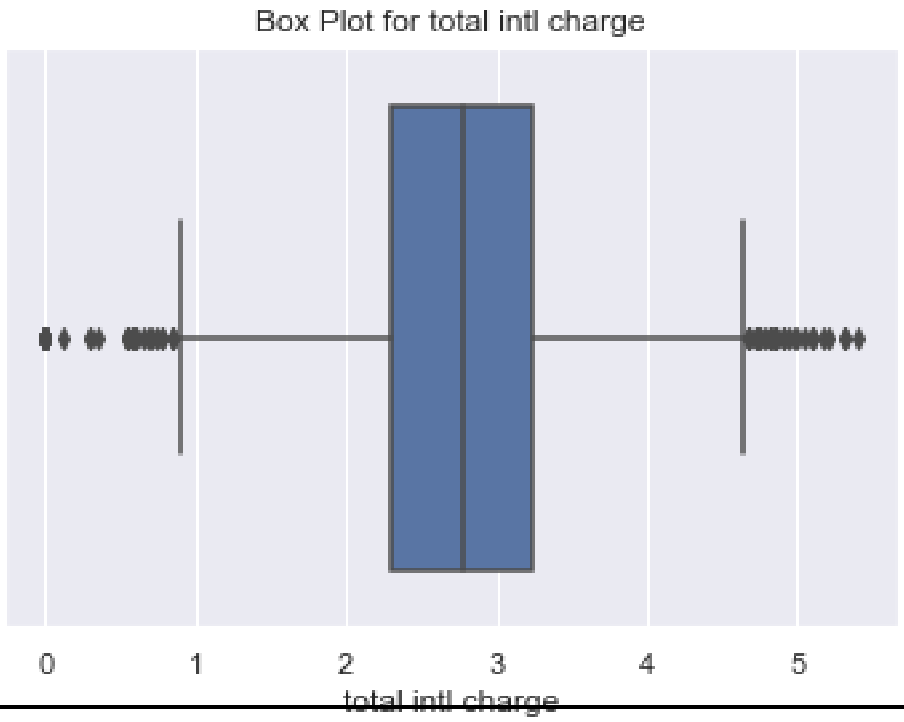

Box Plot for total night calls
0
25
50
75
100
125
150
175
total night calls


Box Plot for total night charge
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
total night charge

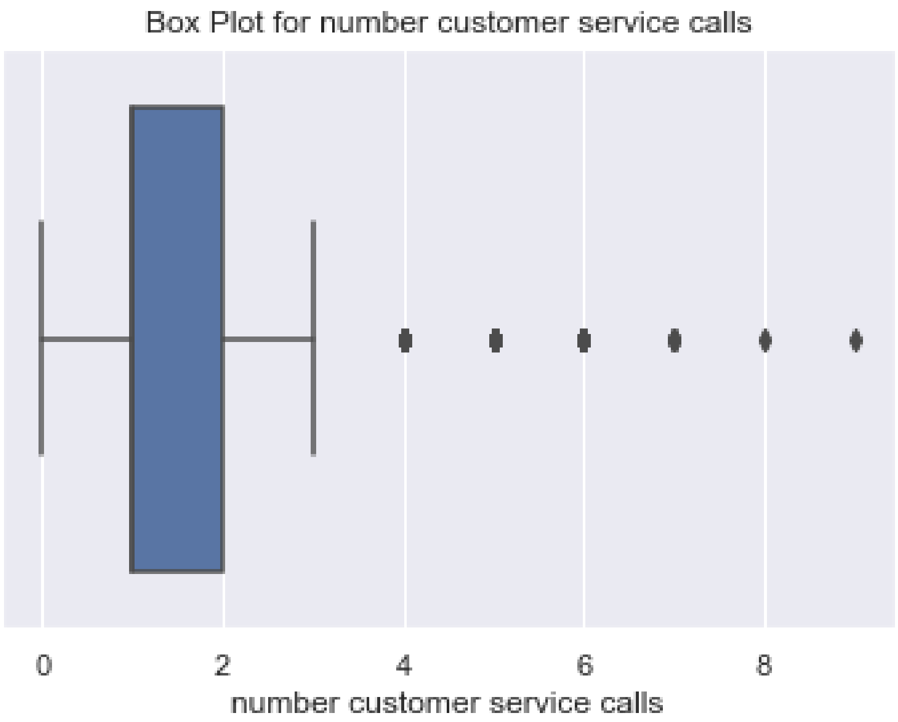

Box Plot for total intl minutes
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
total intl minutes



Box Plot for total intl calls
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
total intl calls



Box Plot for total intl charge
0
1
2
3
4
5
total intl charge



The above boxplots clearly shows the outliers present in the variables. So to treat the outlier we calculated upper and lower quartile then the interquartile range and then we created nan value in place of outliers which are later replaced by knn imputation method.

Outlier treatment

```
#detect and replace outliers with NA
#Extracting quartiles
for i in variable_num:
    q75,q25=np.percentile(churn_red[i],[75,25])



   ##calculating iqr
    iqr=q75-q25

    #calculating inner and outer fence
    minimum= q25-(iqr*1.5)
    maximum= q75+(iqr*1.5)

    #replace with NA
    churn_red.loc[churn_red[i]<minimum,i] = np.nan
    churn_red.loc[churn_red[i]>maximum,i] = np.nan
```

```python
#Checking missing values created by outliers
churn_missing_value = churn_red.isnull().sum()
```

```python
churn_missing_value
```

```
account length                 24
international plan              0
voice mail plan                 0
number vmail messages          60
total day minutes              34
total day calls                35
total day charge               34
total eve minutes              43
total eve calls                27
total eve charge               42
total night minutes            39
total night calls              43
total night charge             39
total intl minutes             72
total intl calls              118
total intl charge              72
number customer service calls 399
Churn                           0
dtype: int64
```

```python
#create data frame with missing values
churn_missing_val = pd.DataFrame(churn_red.isnull().sum())
```

```
churn_missing_val
```

	variables	missing_percentage
0	account length	0.48
1	international plan	0.00
2	voice mail plan	0.00
3	number vmail messages	1.20
4	total day minutes	0.68
5	total day calls	0.70
6	total day charge	0.68
7	total eve minutes	0.86
8	total eve calls	0.54
9	total eve charge	0.84
10	total night minutes	0.78
11	total night calls	0.86
12	total night charge	0.78
13	total intl minutes	1.44
14	total intl calls	2.36
15	total intl charge	1.44
16	number customer service calls	7.98
17	Churn	0.00

```
#impute with knn
#Loading data set again
churn_red = cr1.copy()
```

```
churn_red['account length'].iloc[10] = np.nan
```

```
C:\Users\Aditya\Anaconda3\lib\site-packages\pandas\core\indexing.py:189: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy
  self._setitem_with_indexer(indexer, value)
```

```
#Applying KNN imputation method
churn_red = pd.DataFrame(KNN(k = 3).fit_transform(churn_red), columns = churn_red.columns)
```

```
Imputing row 1/5000 with 0 missing, elapsed time: 4.595
Imputing row 101/5000 with 1 missing, elapsed time: 4.597
Imputing row 201/5000 with 0 missing, elapsed time: 4.599
Imputing row 301/5000 with 0 missing, elapsed time: 4.604
Imputing row 401/5000 with 0 missing, elapsed time: 4.605
Imputing row 501/5000 with 0 missing, elapsed time: 4.606
Imputing row 601/5000 with 0 missing, elapsed time: 4.608
Imputing row 701/5000 with 0 missing, elapsed time: 4.609
Imputing row 801/5000 with 0 missing, elapsed time: 4.610
Imputing row 901/5000 with 0 missing, elapsed time: 4.611
Imputing row 1001/5000 with 0 missing, elapsed time: 4.613
Imputing row 1101/5000 with 0 missing, elapsed time: 4.615
Imputing row 1201/5000 with 1 missing, elapsed time: 4.616
Imputing row 1301/5000 with 0 missing, elapsed time: 4.617
Imputing row 1401/5000 with 2 missing, elapsed time: 4.619
Imputing row 1501/5000 with 0 missing, elapsed time: 4.620
```

```
churn_red['account length'].iloc[10]
```

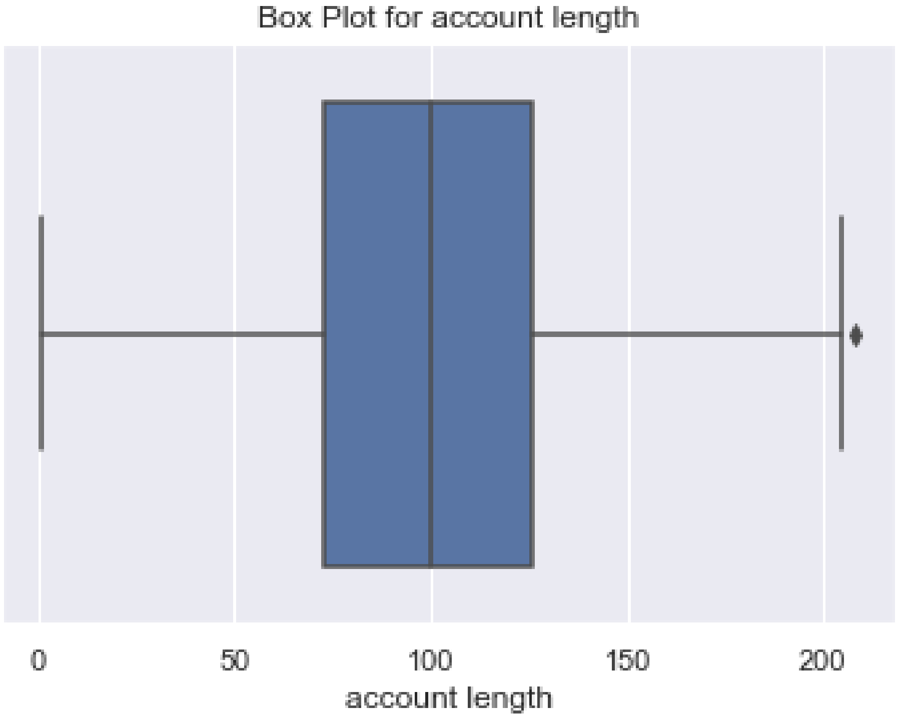
```
79.4621474951254
```

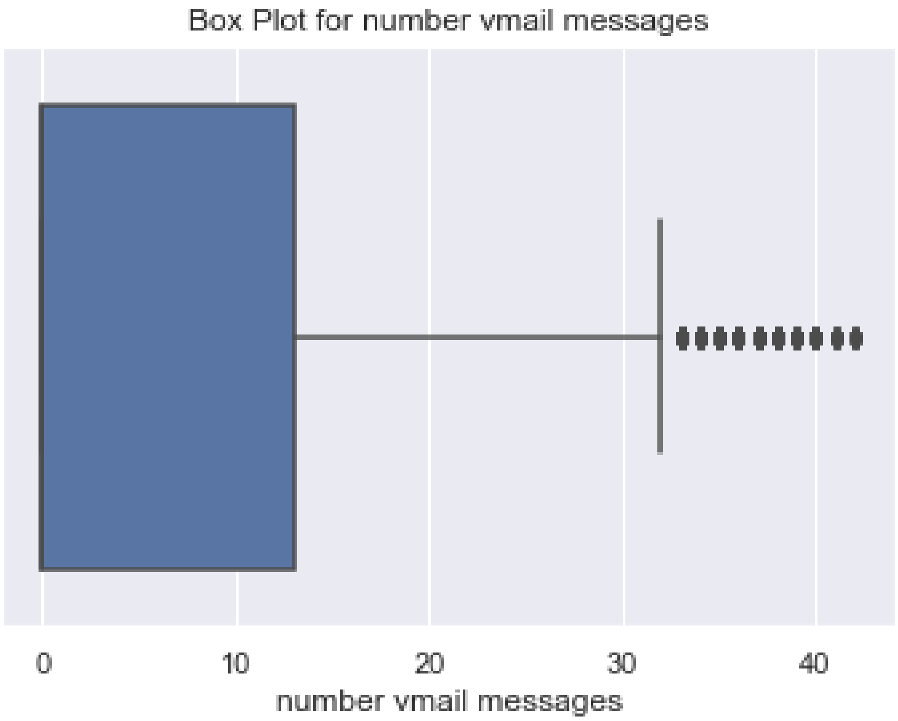
```
#Here we will go with knn imputation method as we have seen earlier that actual value was 65 and knn gave the result as
#79.46 which is close to actual value hence we will impute na with knn.
```

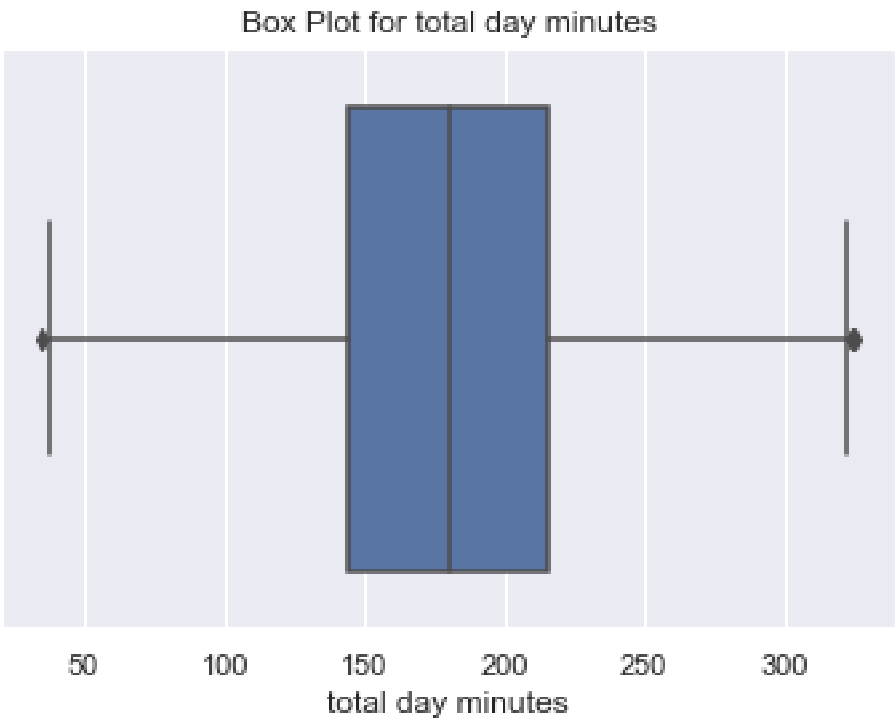
```
churn_red.isnull().sum()
```

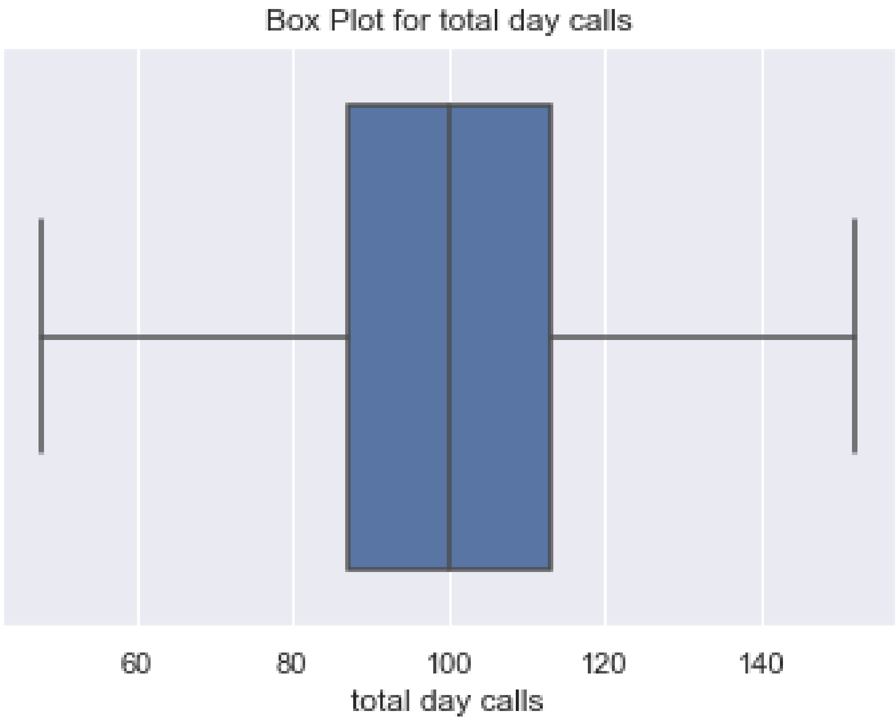
```
account length                   0
international plan               0
voice mail plan                  0
number vmail messages            0
total day minutes                0
total day calls                  0
total day charge                 0
total eve minutes                0
total eve calls                  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
dtype: int64
```

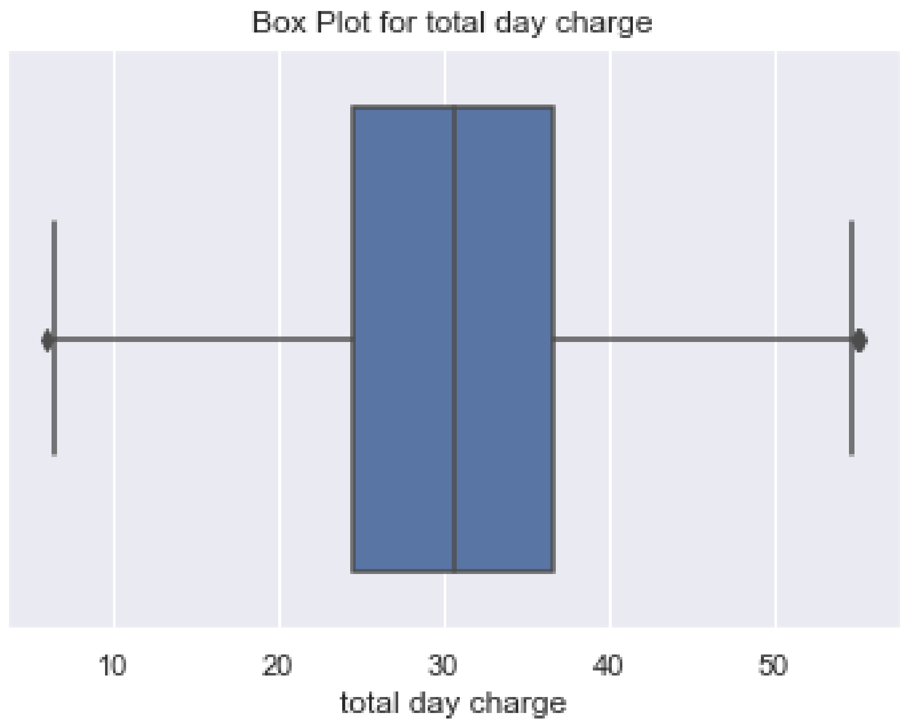
Boxplots after outlier treatment

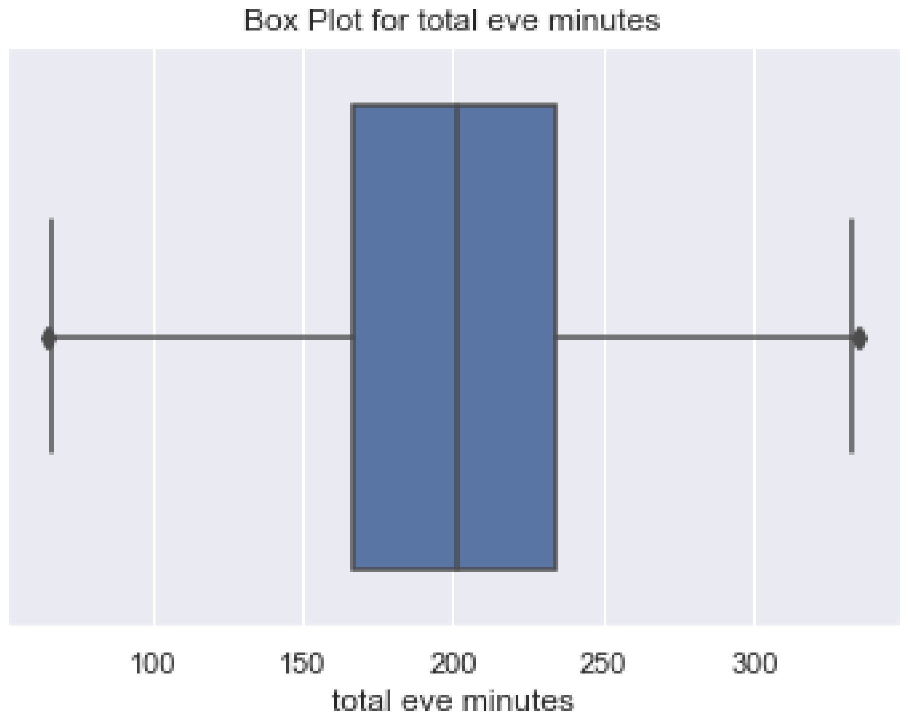


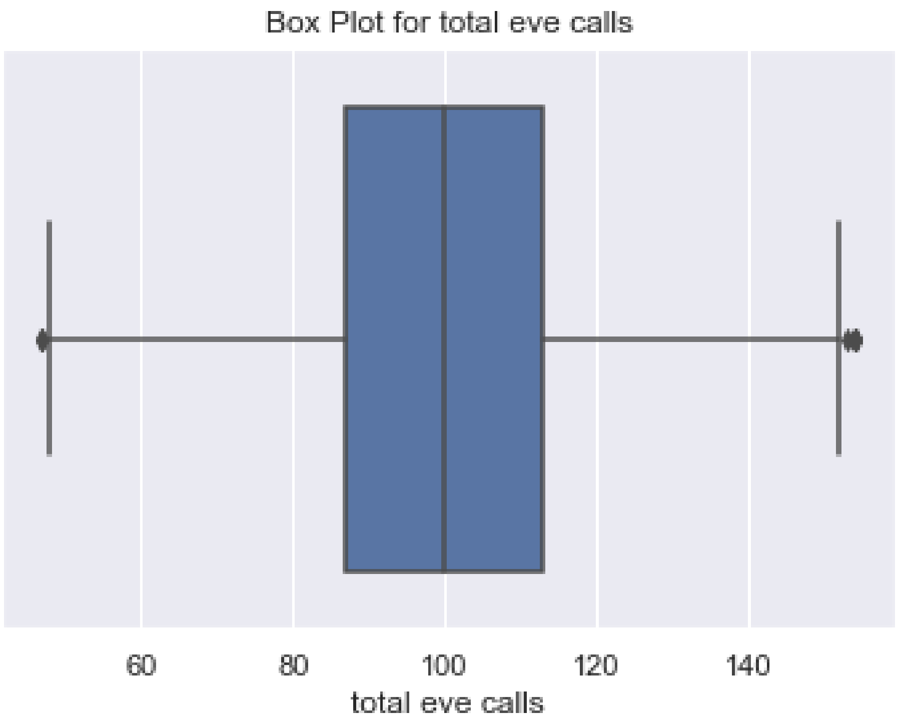


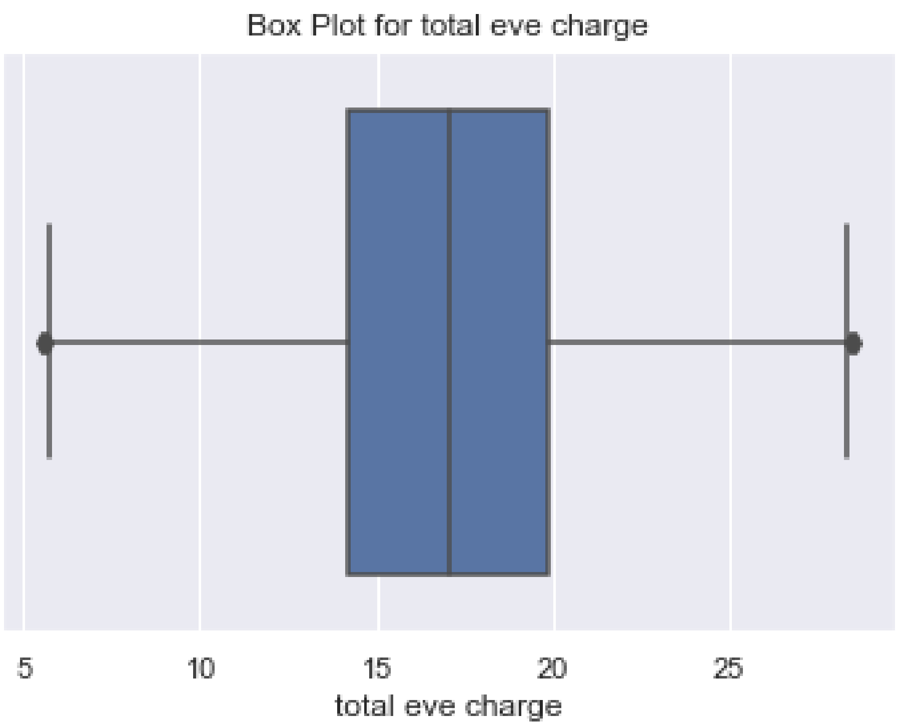

Box Plot for total day minutes
50
100
150
200
250
300
total day minutes

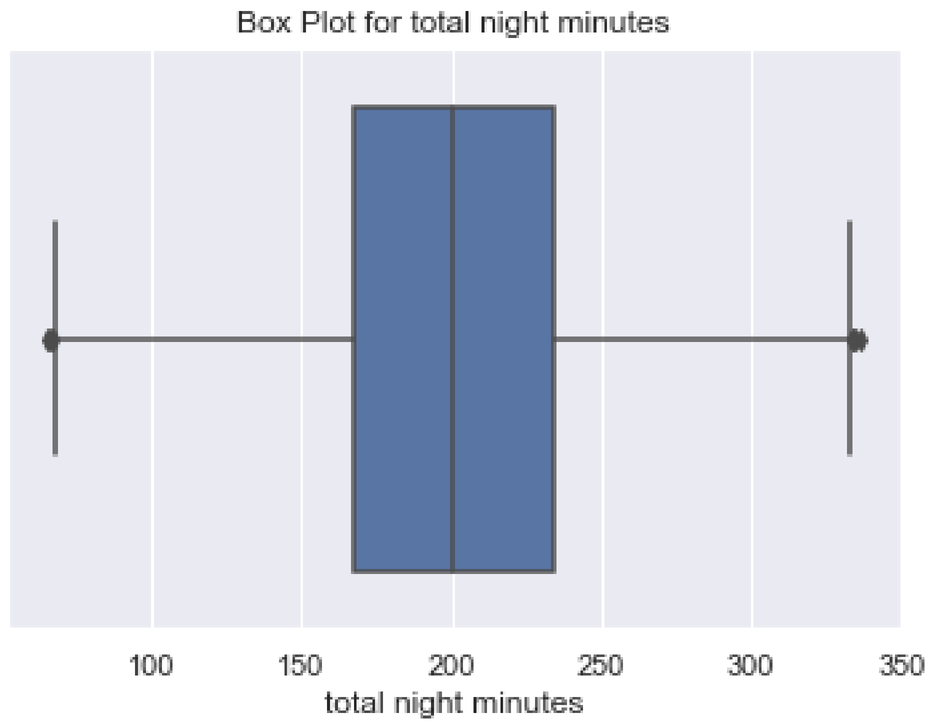

Box Plot for total day calls
60
80
100
120
140
total day calls

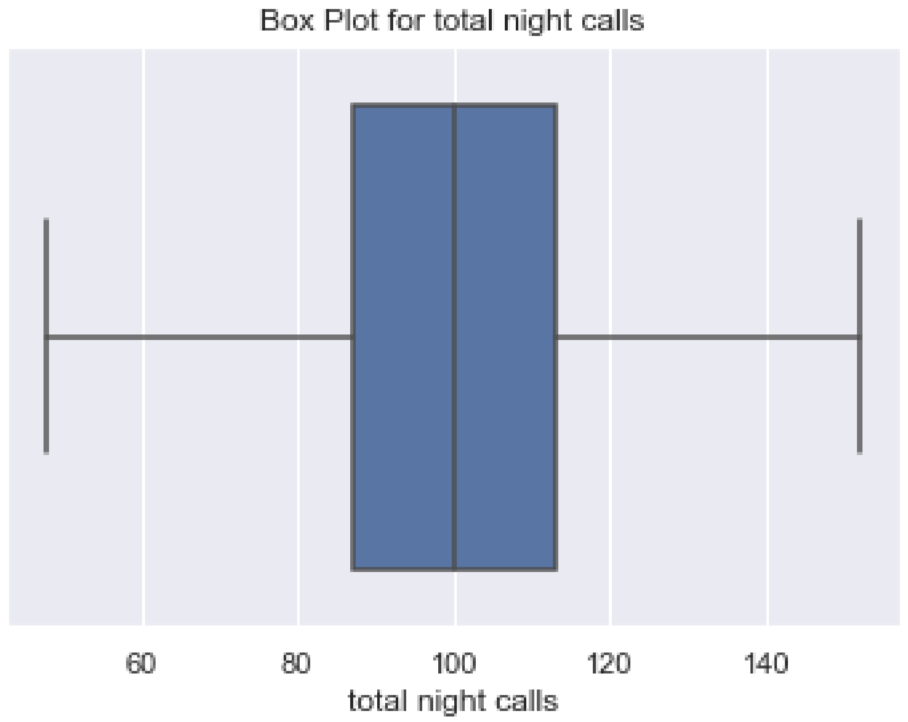

Box Plot for total day charge
10
20
30
40
50
total day charge

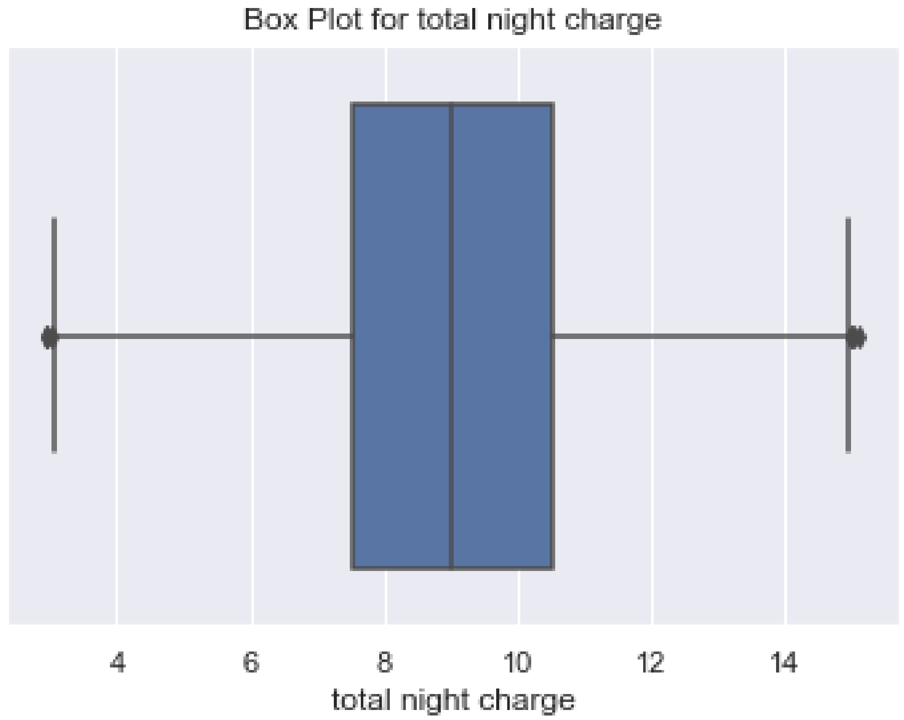

Box Plot for total eve minutes
100
150
200
250
300
total eve minutes

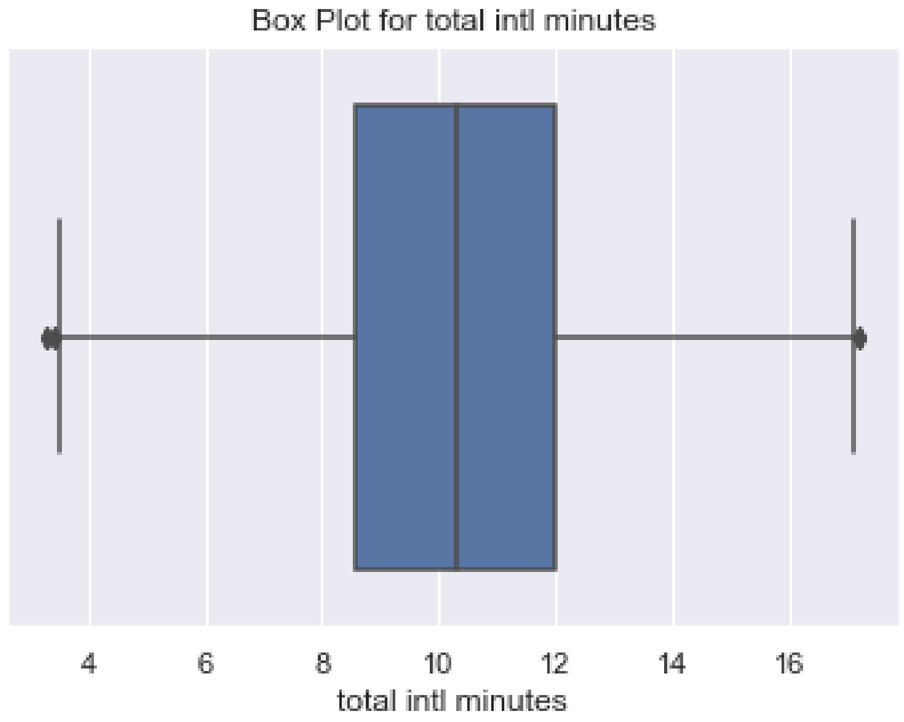

Box Plot for total eve calls
60
80
100
120
140
total eve calls

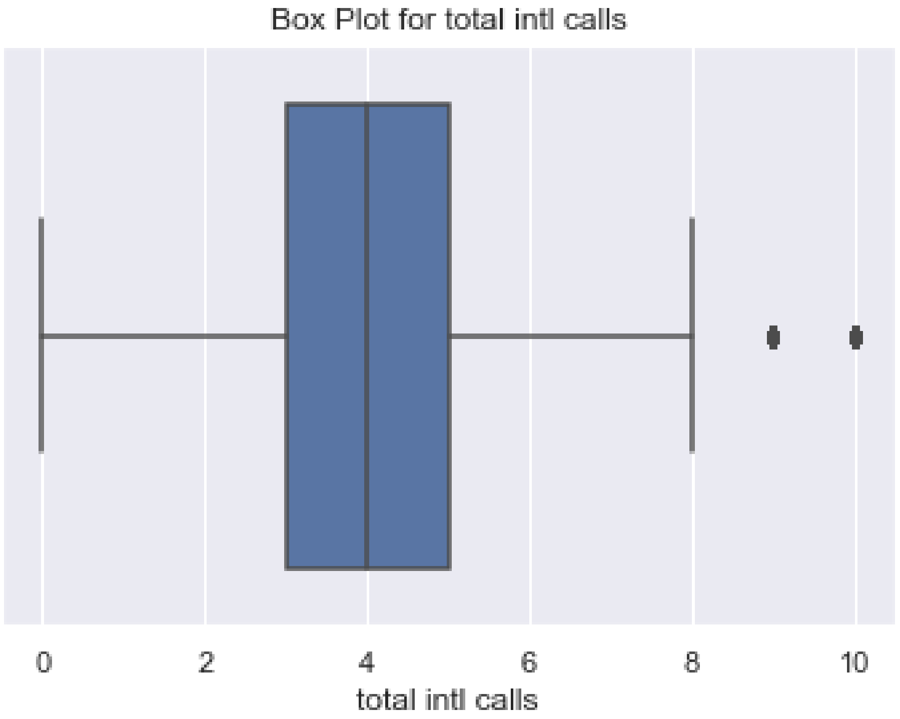

Box Plot for total eve charge
5
10
15
20
25
total eve charge

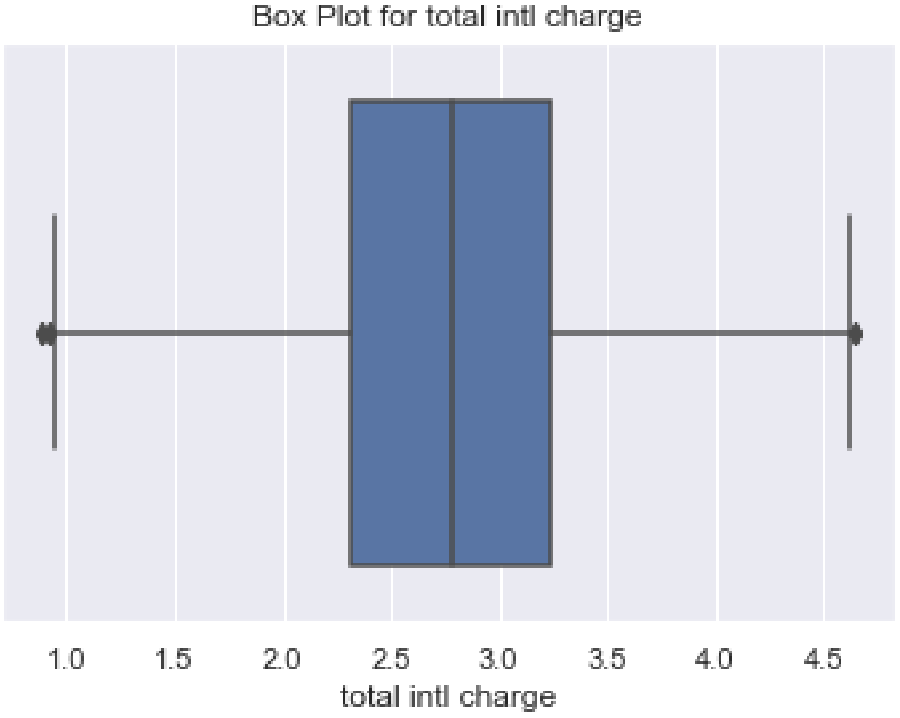

Box Plot for total night minutes
100
150
200
250
300
350
total night minutes

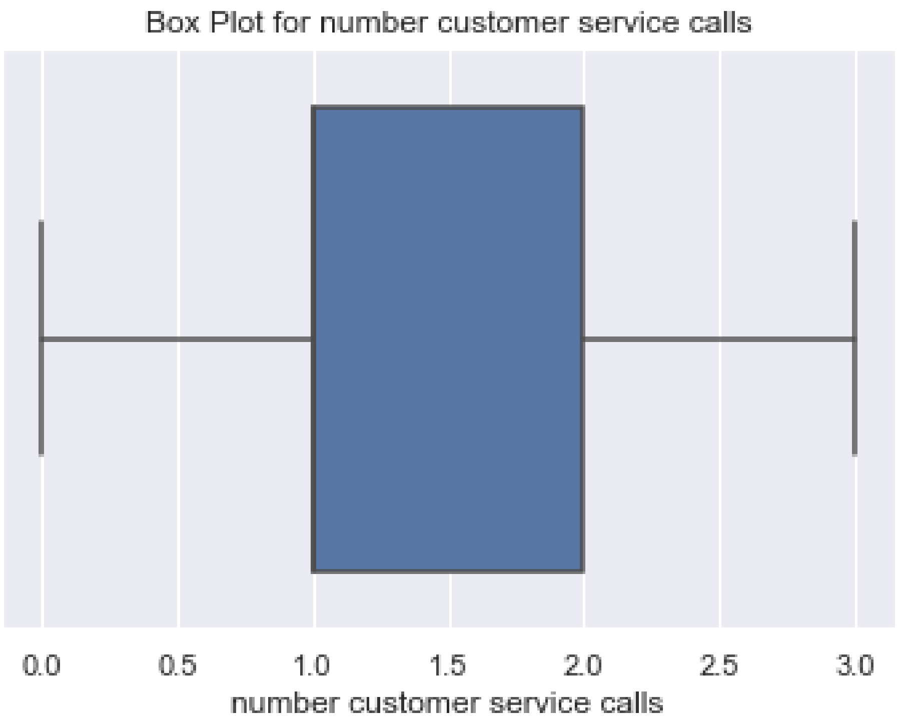

Box Plot for total night calls
60
80
100
120
140
total night calls


Box Plot for total night charge
4
6
8
10
12
14
total night charge


Box Plot for total intl minutes
4
6
8
10
12
14
16
total intl minutes


Box Plot for total intl calls
0
2
4
6
8
10
total intl calls


Box Plot for total intl charge
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
total intl charge



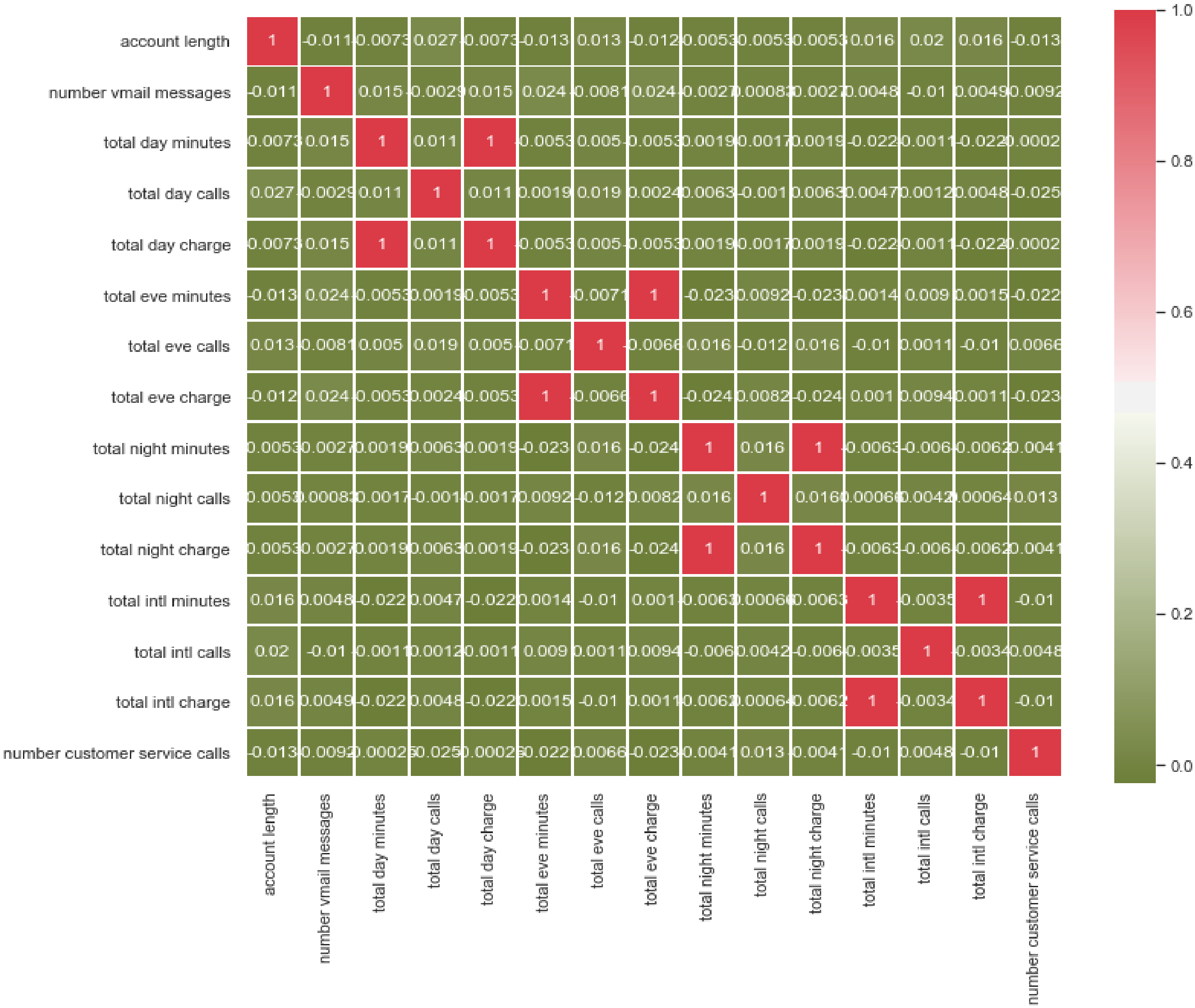
2.1.3 Feature Selection

Machine learning works on a simple rule of GIGO i.e. Garbage In Garbage Out. Here garbage refers to the noise or redundant values.

This becomes even more important when the number of features are very large. We need not use every feature at our disposal for creating an algorithm. We can assist our algorithm by feeding in only those features that are important. Feature subsets gives better results than complete set of features for the same algorithm or “Sometimes, less is better!”.

We should consider the selection of feature for model keeping in mind that there should be low correlation between two independent variables otherwise there will be problem of multicollinearity.

Fig. 3.0



From above correlation plot we can clearly see that variables like total day minutes and total day charge, total eve minutes and total eve charge, total night minutes and total night charge, total intl minutes and total intl charge have high correlation. It means that we must drop one variable out of two having high correlation. So, in our study here we will drop variables 'total day minutes', 'total night minutes', 'total eve minutes', 'total intl minutes'.

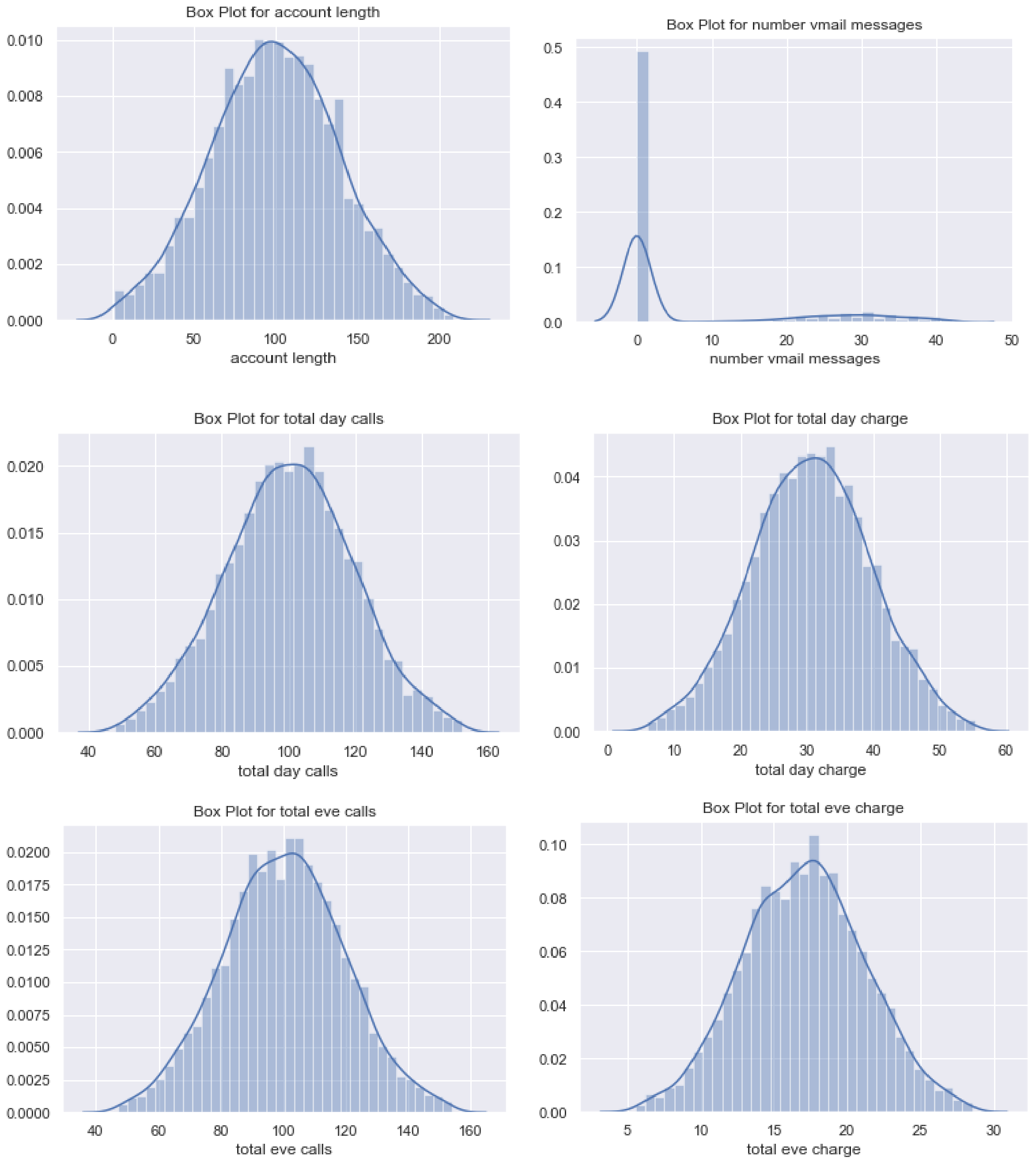
Color dark Red indicates there is strong positive correlation and dark green indicates negative correlation.

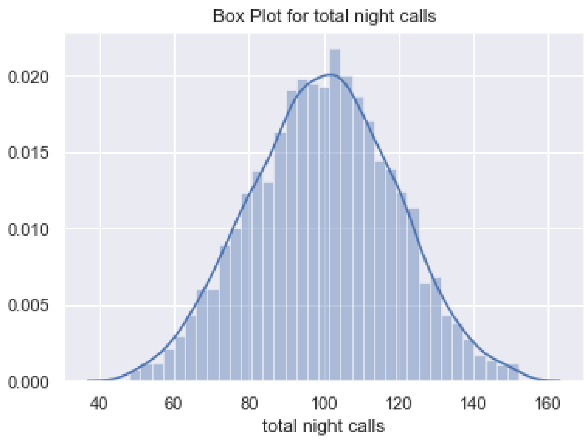
3. Data Distribution

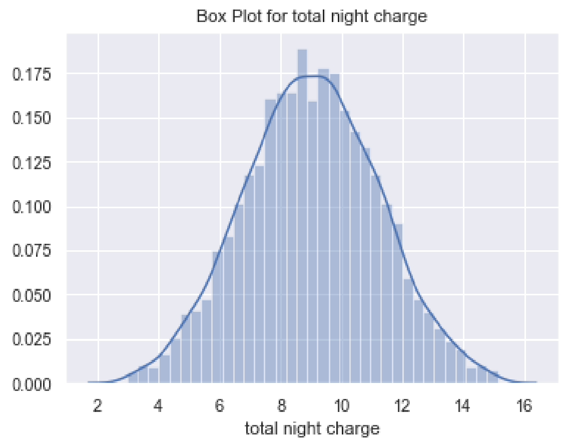
Checking distribution of variables with help of distribution plot

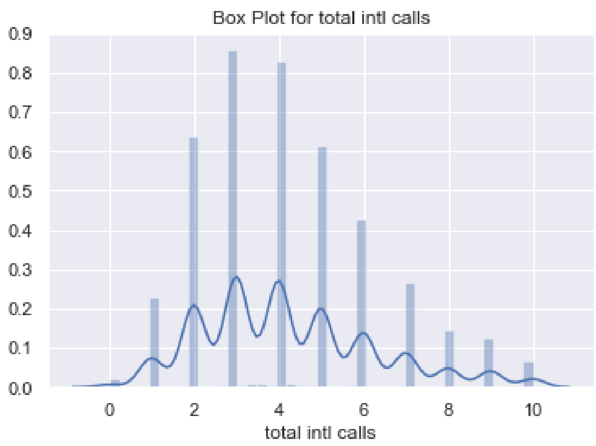
Distribution of continuous predictor variables

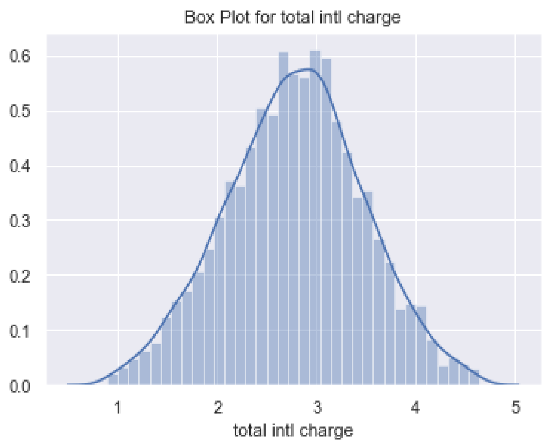
Fig. 4.0

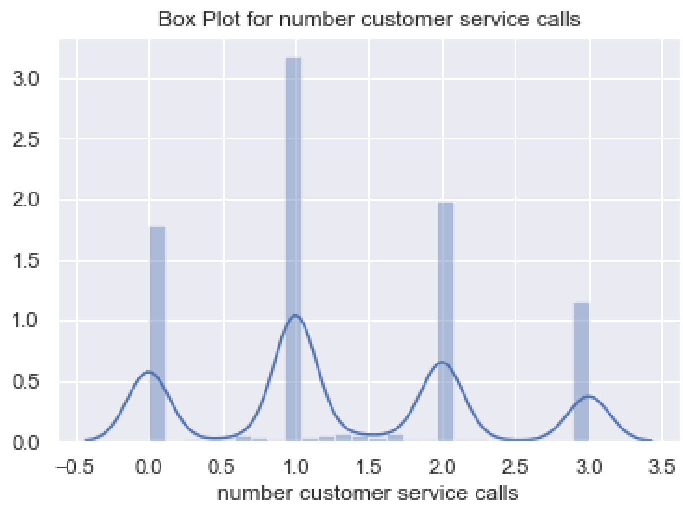












From above distribution plots we can clearly see that data is normaly distributed. Since data is normally distributed hence we will standardise data i.e. making data revolve near about it's mean point giving more appropiate results.

```
#As from above distribution plots we can clearly see that data is normally distributed hence we can apply standardisation.
for i in variable_num_update:
    print(i)
    churn_red[i] = (churn_red[i] - churn_red[i].mean())/churn_red[i].std()
```

```
account length
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
```

```
churn_red.head(5)
```

	account length	international plan	voice mail plan	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	Churn
0	0.729680	0.0	1.0	1.368521	0.510514	1.618908	-0.064526	-0.066337	-0.461033	0.909281	-0.596755	-0.117152	-0.329016	0.0
1	0.188929	0.0	1.0	1.445884	1.188521	-0.358847	0.142540	-0.104992	0.164915	1.110030	-0.596755	1.332756	-0.329016	0.0
2	0.961430	0.0	0.0	-0.565547	0.719132	1.204254	0.504906	-1.631853	0.217077	-0.774266	0.372473	0.738294	-1.383732	0.0
3	-0.403323	1.0	0.0	-0.565547	-1.523505	2.274040	-0.633958	-0.919454	-0.565358	-0.071647	1.341700	-1.451067	0.725700	0.0
4	-0.635073	1.0	0.0	-0.565547	0.666977	-0.261083	1.126105	-1.073776	1.103837	-0.276958	-0.596755	-0.073655	1.780416	0.0

4. Modelling

4.1 Model Selection

Model Selection is a process of selecting the model which have better accuracy and can work on train and test data. We must select a model where algorithm works well and shows low error rate. We have also used error metrics here; error metrics can be defined as an Error Metric a type of Metric used to measure the error of a forecasting model. They can provide a way to forecast and quantitatively compare the performance of competing models. We made Decision Tree Classifier, Random Forest Classifier and Logistic regression model. When we executed all three models the results were decision tree classifier showed much better results compared to other two.

4.1.1 Decision Tree Classifier

A tree has many analogies in real life and turns out that it has influenced a wide area of machine learning. In decision analysis, a decision tree classifier can be used to visually and explicitly represent decisions and decision making. As the name goes, it uses a tree-like model of decisions. Decision Tree Classifier is a simple and widely used classification technique. It applies a straightforward idea to solve the classification problem.

Decision Tree Algorithm

Decision Tree Classifier

]:
```
#Divide data into train and test
X = churn_red.values[:, 0:13]
Y = churn_red.values[:,13]

X_train, X_test, y_train, y_test = train_test_split( X, Y, test_size = 0.2)
```

]:
```
#Decision Tree
C50_model = tree.DecisionTreeClassifier(criterion='entropy').fit(X_train, y_train)
```

]:
```
#predict new test cases
C50_Predictions = C50_model.predict(X_test)
```

]:
```
#Decision tree and classifier. here we are using C5.0 model
clf = tree.DecisionTreeClassifier(criterion = 'entropy').fit(X_train, y_train)
```

]:
```
#predict new test cases
y_pred = clf.predict(X_test)
```

Error Metrics

:
```
cm = confusion_matrix(y_test, y_pred)
```

:
```
cm
```

:
```
array([[800,  50],
       [ 47, 103]], dtype=int64)
```

:
```
cm = pd.crosstab(y_test, y_pred)
```

:
```
cm
```

:

col_0	0.0	1.0
row_0		
0.0	800	50
1.0	47	103

:
```
##Let us save TP, TN, FP, FN
TN = cm.iloc[0,0]
FP = cm.iloc[0,1]
FN = cm.iloc[1,0]
TP = cm.iloc[1,1]
```

```
TN, FP
```

```
(800, 50)
```

```
#Checking accuracy of model
accuracy_score(y_test, y_pred)*100
```

```
90.3
```

```
#Here we have used recall error metrics to see who are customers who have actually churned out. So, it will help our client
#to work on those customers by providing some good deals or by some other marketing mean to retain those customers back.
#Recall rate(True positivve)
(TP*100)/(TP+FN)
```

```
68.66666666666667
```

```
#Results
#Accuracy: 90.3
##Recall rate(True positivve): 68.66
```

4.1.2 Random Forest Classifier

Random forests classifier is an ensemble learning method for classification that operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of overfitting to their training set.

Random forest functions in following way

- Draws a bootstrap sample from training data.
- For each sample grow a decision tree and at each node of the tree

a. Ramdomly draws a subset of variable and p total of features that are available
b. Picks the best variable and best split from the subset of mtry variable
c. Continues until the tree is fully grown.

Random Forest Implementation

Random Forest

```
#Loading copy again
churn_red = CR.copy()
```

```
churn_red.head(5)
```

	account length	international plan	voice mail plan	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	Churn
0	0.729680	0.0	1.0	1.368521	0.510514	1.618908	-0.064526	-0.066337	-0.461033	0.909281	-0.596755	-0.117152	-0.329016	0.0
1	0.188929	0.0	1.0	1.445884	1.188521	-0.358847	0.142540	-0.104992	0.164915	1.110030	-0.596755	1.332756	-0.329016	0.0
2	0.961430	0.0	0.0	-0.565547	0.719132	1.204254	0.504906	-1.631853	0.217077	-0.774266	0.372473	0.738294	-1.383732	0.0
3	-0.403323	1.0	0.0	-0.565547	-1.523505	2.274040	-0.633958	-0.919454	-0.565358	-0.071647	1.341700	-1.451067	0.725700	0.0
4	-0.635073	1.0	0.0	-0.565547	0.666977	-0.261083	1.126105	-1.073776	1.103837	-0.276958	-0.596755	-0.073655	1.780416	0.0

```
#Divide data into train and test
X = churn_red.values[:, 0:13]
Y = churn_red.values[:,13]

X_train, X_test, y_train, y_test = train_test_split( X, Y, test_size = 0.2)
```

```
#Random Forest
RF_model = RandomForestClassifier(n_estimators = 20).fit(X_train, y_train)
```

```
RF_Predictions = RF_model.predict(X_test)
```

```
#build confusion matrix
cm = confusion_matrix(y_test, y_pred)
```

```
cm
```

```
array([[724, 139],
       [123,  14]], dtype=int64)
```

```
cm = pd.crosstab(y_test, RF_Predictions)
```

```
cm
```

col_0	0.0	1.0
row_0		
0.0	857	6
1.0	75	62

```
#Let us save TP, TN, FP, FN
TN = cm.iloc[0,0]
FN = cm.iloc[1,0]
TP = cm.iloc[1,1]
FP = cm.iloc[0,1]
```

```
TN
```

857

```
#Accuracy score
accuracy_score(y_test,y_pred)*100
```

73.8

```
#Recall rate(True positivve)
(TP*100)/(TP+FN)
```

45.25547445255474

```
#Accuracy: 73.8
#Recall rate(True positivve) : 45.25
```

4.1.3 Logistic Regression

The logistic regression is a predictive analysis. Logistic regression is used to describe data and to explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables.

Logistic Regression

```
#Loading copy again
churn_red = CR1.copy()
```

```
#Let us prepare data for logistic regression
#replace target categories with Yes or No
churn_red['Churn'] = churn_red['Churn'].replace('No', 0)
churn_red['Churn'] = churn_red['Churn'].replace('Yes', 1 )
```

```
churn_red.head(5)
```

	account length	international plan	voice mail plan	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	Churn
0	0.729680	0.0	1.0	1.368521	0.510514	1.618908	-0.064526	-0.066337	-0.461033	0.909281	-0.596755	-0.117152	-0.329016	0.0
1	0.188929	0.0	1.0	1.445884	1.188521	-0.358847	0.142540	-0.104992	0.164915	1.110030	-0.596755	1.332756	-0.329016	0.0
2	0.961430	0.0	0.0	-0.565547	0.719132	1.204254	0.504906	-1.631853	0.217077	-0.774266	0.372473	0.738294	-1.383732	0.0
3	-0.403323	1.0	0.0	-0.565547	-1.523505	2.274040	-0.633958	-0.919454	-0.565358	-0.071647	1.341700	-1.451067	0.725700	0.0
4	-0.635073	1.0	0.0	-0.565547	0.666977	-0.261083	1.126105	-1.073776	1.103837	-0.276958	-0.596755	-0.073655	1.780416	0.0

```
churn_red_logit.shape
```

(5000, 1)

```
#Add continous variables
churn_red_logit = churn_red_logit.join(churn_red[variable_num_update])
```

```
churn_red_logit.shape
```

(5000, 12)

```
##Create dummies for categorical variables
variable_cat_update = ['international plan', 'voice mail plan']

for i in variable_cat_update:
    temp = pd.get_dummies(churn_red[i], prefix = i)
    churn_red_logit = churn_red_logit.join(temp)
```

```python
churn_red_logit.head(5)
```

	Churn	account length	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	international plan_0.0	international plan_1.0	p
0	0.0	0.729680	1.368521	0.510514	1.618908	-0.064526	-0.066337	-0.461033	0.909281	-0.596755	-0.117152	-0.329016	1	0	
1	0.0	0.188929	1.445884	1.188521	-0.358847	0.142540	-0.104992	0.164915	1.110030	-0.596755	1.332756	-0.329016	1	0	
2	0.0	0.961430	-0.565547	0.719132	1.204254	0.504906	-1.631853	0.217077	-0.774266	0.372473	0.738294	-1.383732	1	0	
3	0.0	-0.403323	-0.565547	-1.523505	2.274040	-0.633958	-0.919454	-0.565358	-0.071647	1.341700	-1.451067	0.725700	0	1	
4	0.0	-0.635073	-0.565547	0.666977	-0.261083	1.126105	-1.073776	1.103837	-0.276958	-0.596755	-0.073655	1.780416	0	1	

```python
churn_red_logit.shape
```

```
(5000, 16)
```

```python
#Splitting the data
Sample_Index = np.random.rand(len(churn_red_logit)) < 0.8

train = churn_red_logit[Sample_Index]
test = churn_red_logit[~Sample_Index]
```

```python
#select column indexes for independent variables
train_cols = train.columns[1:16]
```

```python
#Built Logistic Regression

logit = sm.Logit(train['Churn'], train[train_cols]).fit()

logit.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.351780
         Iterations 7
```

```
C:\Users\Aditya\Anaconda3\lib\site-packages\statsmodels\base\model.py:1092: RuntimeWarning: invalid value encountered in sqrt
  bse_ = np.sqrt(np.diag(self.cov_params()))
C:\Users\Aditya\Anaconda3\lib\site-packages\scipy\stats\_distn_infrastructure.py:879: RuntimeWarning: invalid value encountere
d in greater
  return (self.a < x) & (x < self.b)
C:\Users\Aditya\Anaconda3\lib\site-packages\scipy\stats\_distn_infrastructure.py:879: RuntimeWarning: invalid value encountere
d in less
  return (self.a < x) & (x < self.b)
C:\Users\Aditya\Anaconda3\lib\site-packages\scipy\stats\_distn_infrastructure.py:1821: RuntimeWarning: invalid value encounter
ed in less_equal
  cond2 = cond0 & (x <= self.a)
```

Logit Regression Results

Dep. Variable:	Churn	**No. Observations:**	4000
Model:	Logit	**Df Residuals:**	3986
Method:	MLE	**Df Model:**	13
Date:	Sat, 16 Feb 2019	**Pseudo R-squ.:**	0.1409
Time:	04:37:42	**Log-Likelihood:**	-1407.1
converged:	True	**LL-Null:**	-1637.9
		LLR p-value:	2.095e-90

	coef	std err	z	P>|z|	[0.025	0.975]
account length	0.0613	0.049	1.261	0.207	-0.034	0.157
number vmail messages	0.1781	0.195	0.915	0.360	-0.204	0.560
total day calls	0.0654	0.048	1.370	0.171	-0.028	0.159
total day charge	0.5765	0.051	11.264	0.000	0.476	0.677
total eve calls	0.0099	0.049	0.202	0.840	-0.086	0.106
total eve charge	0.2488	0.049	5.043	0.000	0.152	0.345
total night calls	-0.0251	0.049	-0.517	0.605	-0.120	0.070
total night charge	0.1777	0.048	3.665	0.000	0.083	0.273
total intl calls	-0.2224	0.051	-4.364	0.000	-0.322	-0.123
total intl charge	0.1357	0.049	2.759	0.006	0.039	0.232
number customer service calls	0.0205	0.049	0.421	0.674	-0.075	0.116

international plan_0.0	-1.7897	nan	nan	nan	nan	nan
international plan_1.0	0.1060	nan	nan	nan	nan	nan
voice mail plan_0.0	-0.1380	nan	nan	nan	nan	nan
voice mail plan_1.0	-1.5456	nan	nan	nan	nan	nan

```
#Predict test data
test['Actual_prob'] = logit.predict(test[train_cols])
```

```
C:\Users\Aditya\Anaconda3\lib\site-packages\ipykernel_launcher.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy
```

```
#Here we are probabilities into classified form of yes or no.
test['ActualVal'] = 1
test.loc[test.Actual_prob < 0.5, 'ActualVal'] = 0
```

```
C:\Users\Aditya\Anaconda3\lib\site-packages\ipykernel_launcher.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy
  """Entry point for launching an IPython kernel.
C:\Users\Aditya\Anaconda3\lib\site-packages\pandas\core\indexing.py:543: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy
  self.obj[item] = s
```

```
#Build confusion matrix
CM = pd.crosstab(test['Churn'], test['ActualVal'])
```

```
CM
```

ActualVal	0	1
Churn		
0.0	846	17
1.0	118	19

```
#let us save TP, TN, FP, FN
TN = CM.iloc[0,0]
FN = CM.iloc[1,0]
TP = CM.iloc[1,1]
FP = CM.iloc[0,1]
```

```
TN
```

846

```
#check accuracy of model
((TP+TN)*100)/(TP+TN+FP+FN)
```

86.5

```
#Recall rate(True positivve)
(TP*100)/(TP+FN)
```

13.86861313868613

```
#Accuracy: 86.5
#Recall rate(True positivve) : 13.86
```

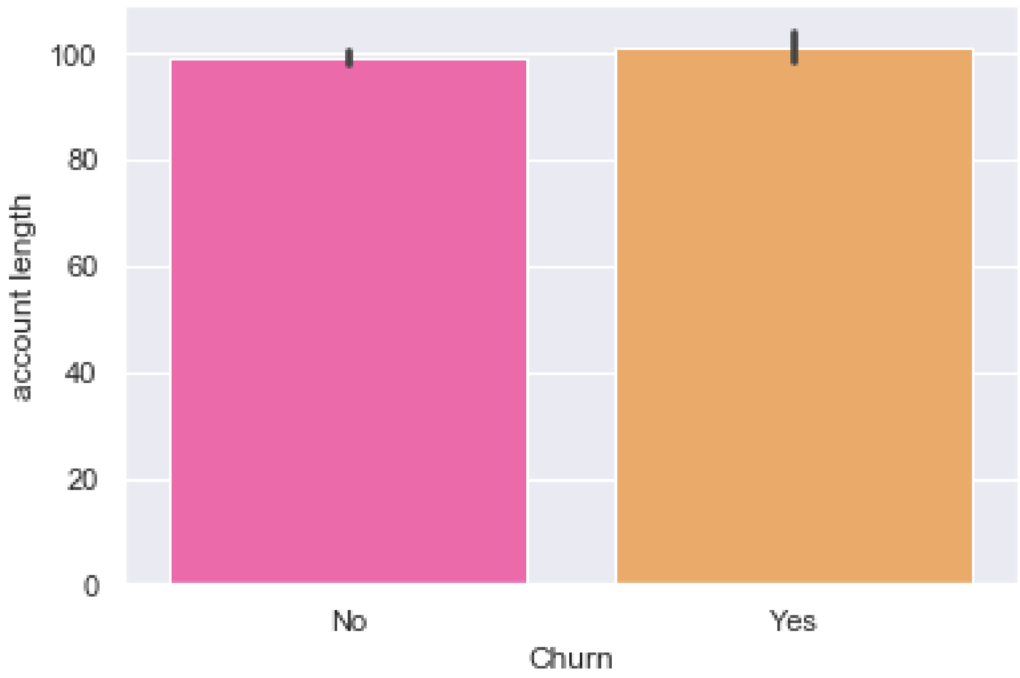
```
#Decision tree classifier model is performing better than random forest and Logistic regression. Hence we will apply decision
#tree classifier.
```

6. Conclusion: - For the Churn reduction decision tree classifier Model is appropriate to predict the churn score based on usage pattern.

Appendix - A

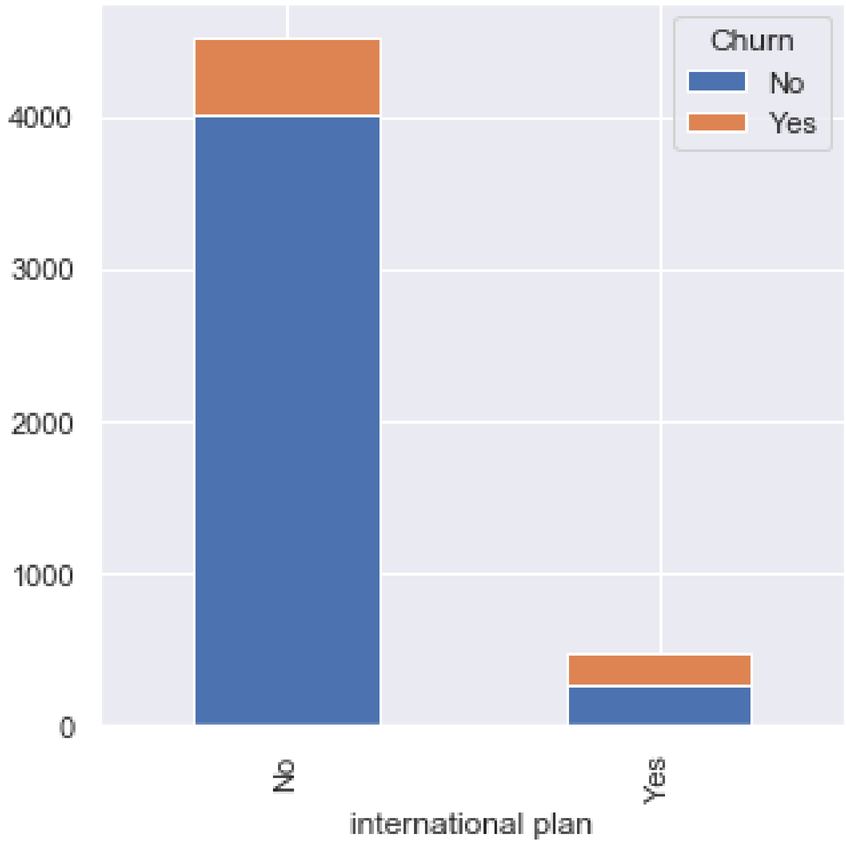
Bar Plots

Bar plot between account length and Churn



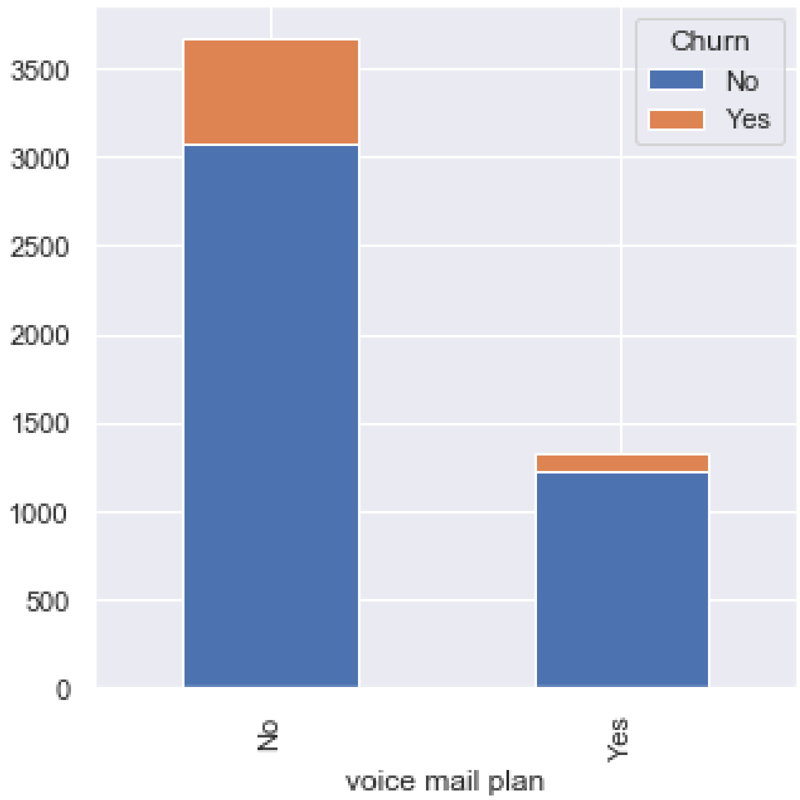
From above bar plot we can see that those customers having more account length have churned out.

Bar plot between international plan and churn



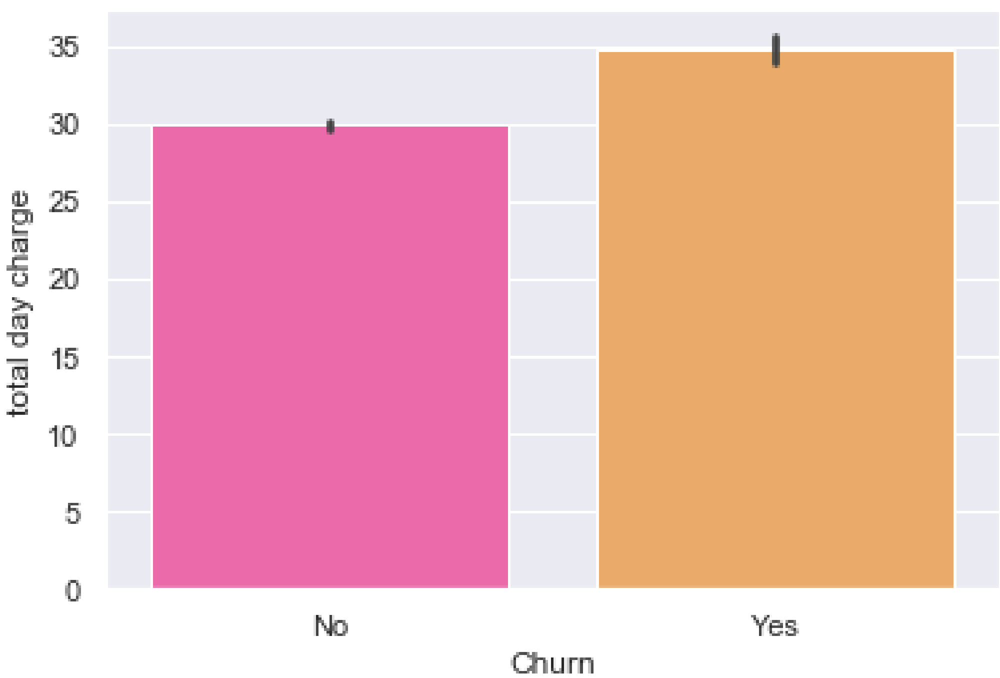
As we can see from above bar plot that whether customers are having international plan or not, still few have churned out. So, we can infer here that there might be some other reasons as well because of which few customers have churned out even they were not having the international plan.

Bar plot between voice mail plan and churn

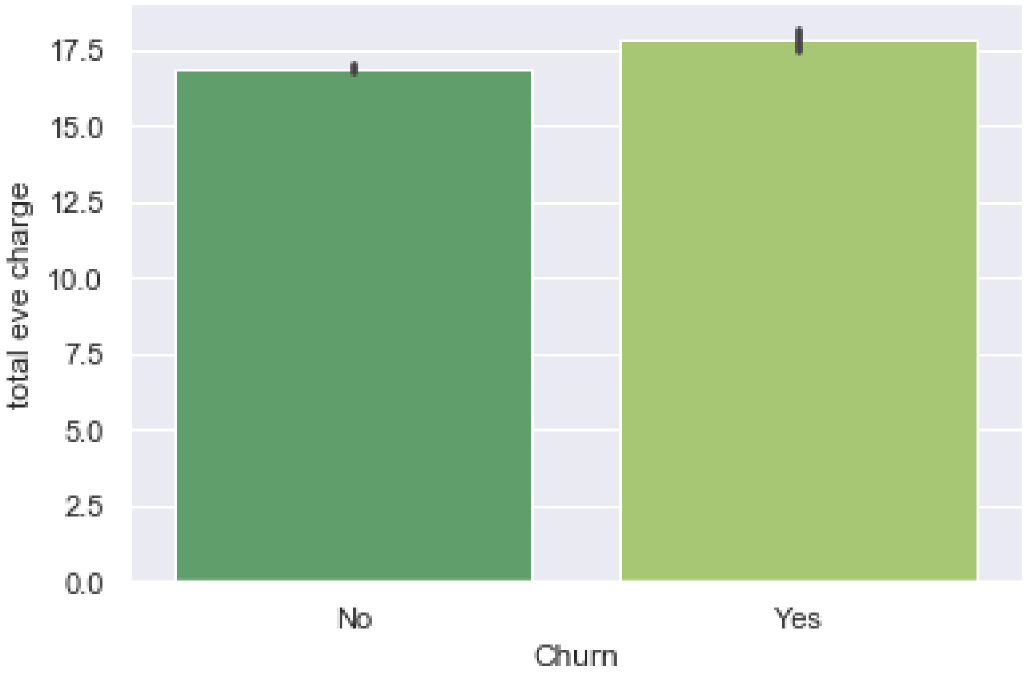


From above plot we can clearly see that customers not having voicemail plan have even churn out. So it's like same as international plan and company should look after it.

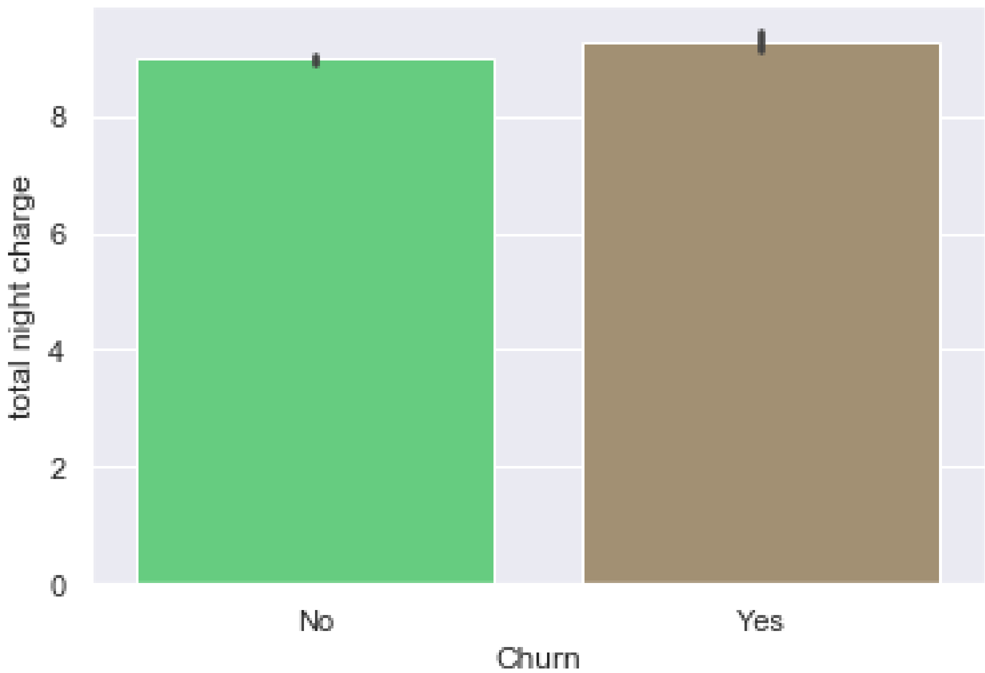
Bar plot between total day charge and churn



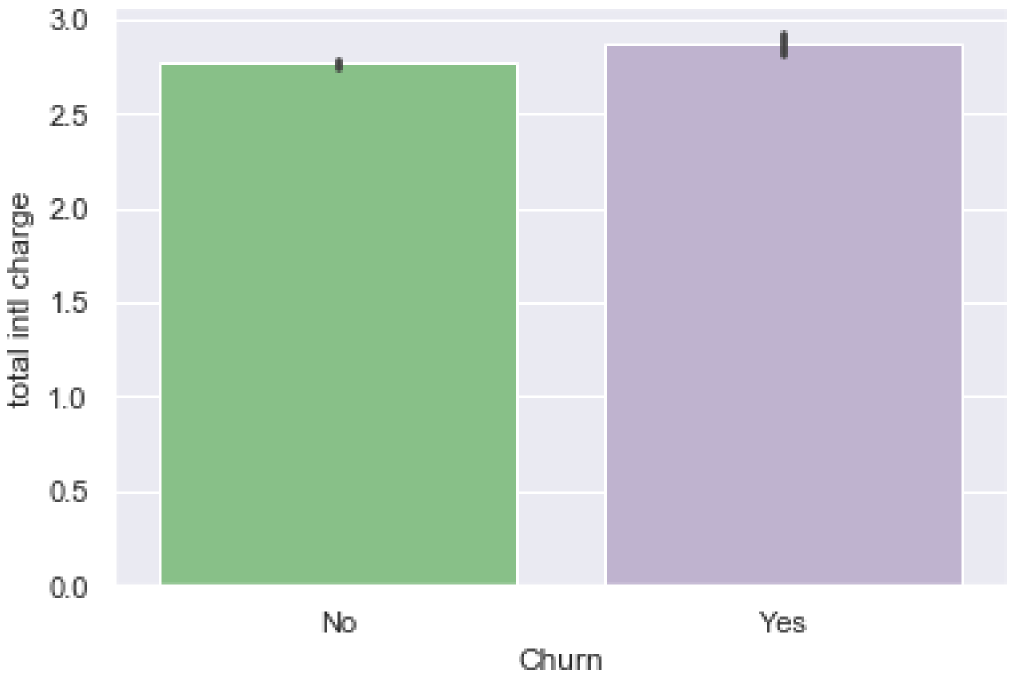
Bar plot between total eve charge and churn



Bar plot between total night charge and churn

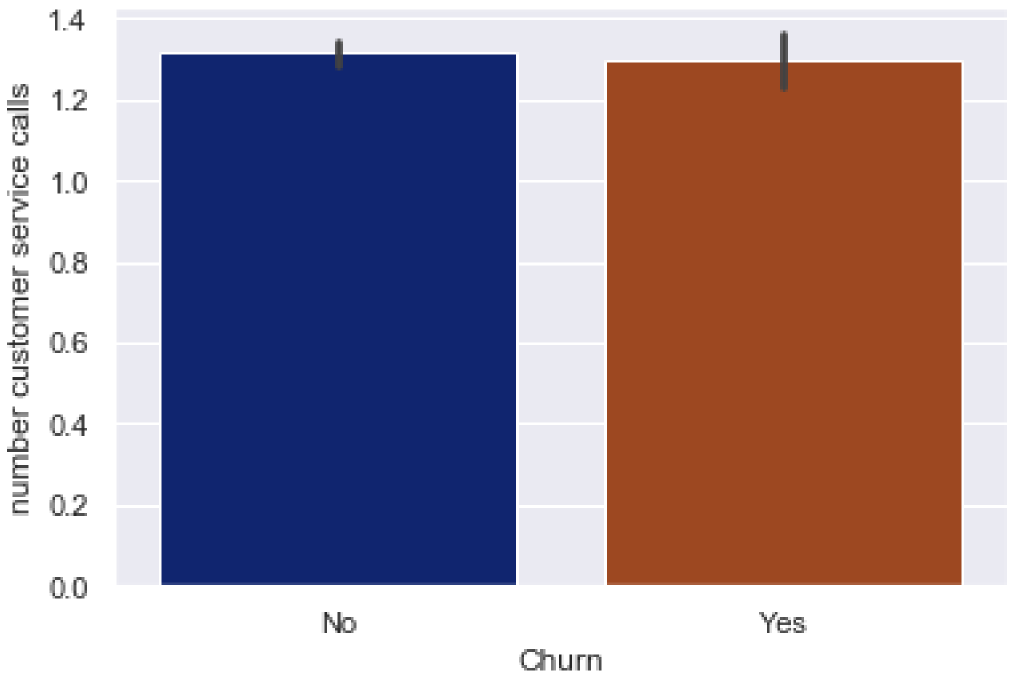


Bar plot between total intl charge and churn



From above plots we can clearly see that there is a churn where charges are more. So, company should do something about it either by introducing some new tariffs or through providing some extra minutes to retain their customers.

Bar plot between number customer service calls and churn



In above bar plot we can clearly see and infer that the number of customer service call has resulted in a no to the customer churn.

Appendix- B - Python Code

Fig 3.0 Python Code

Feature Selection

```
br_corr = churn_red.loc[:,variable_num]
```

```
sns.set()
f, ax = plt.subplots(figsize=(12,10))

corr_matrix = br_corr.corr()

sns.heatmap(corr_matrix, mask = np.zeros_like(corr_matrix, dtype=np.bool), cmap = sns.diverging_palette(27100,10, as_cmap= True),
          annot = True , linewidths = 0.9,square = True, ax=ax)
```

Complete Python File

Loading Libraries

```
import os
import pandas as pd
import numpy as np
import seaborn as sns
from fancyimpute import KNN
%matplotlib inline
from ggplot import *
import matplotlib.pyplot as plt
from scipy import stats
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
from scipy.stats import chi2_contingency
from sklearn.metrics import confusion_matrix
from sklearn.cross_validation import train_test_split
from sklearn import tree
from sklearn.metrics import accuracy_score
from sklearn.ensemble import RandomForestClassifier
import statsmodels.api as sm
```

```
##Setting working directory
os.chdir('E:/Project/Churn reduction')
```

```
os.getcwd()
```

```
df = pd.read_csv('E:/Project/Churn reduction/Train_data.csv')
```

```
df1 = pd.read_csv('E:/Project/Churn reduction/Test_data.csv')
```

```
df3 = pd.concat([df,df1])
```

```
df3.to_csv('churn_reduction.csv', index=False)
```

```
churn_red = pd.read_csv('E:/Project/Churn reduction/churn_reduction.csv')
```

```
churn_red.head(5)
```

	state	account length	area code	phone number	international plan	voice mail plan	number vmail messages	total day minutes	total day calls	total day charge	...	total eve calls	total eve charge	total night minutes	total night calls	total night charge	total intl minutes	total intl calls	total intl charge
0	KS	128	415	382-4657	no	yes	25	265.1	110	45.07	...	99	16.78	244.7	91	11.01	10.0	3	2.70
1	OH	107	415	371-7191	no	yes	26	161.6	123	27.47	...	103	16.62	254.4	103	11.45	13.7	3	3.70
2	NJ	137	415	358-1921	no	no	0	243.4	114	41.38	...	110	10.30	162.6	104	7.32	12.2	5	3.29
3	OH	84	408	375-9999	yes	no	0	299.4	71	50.90	...	88	5.26	196.9	89	8.86	6.6	7	1.78
4	OK	75	415	330-6626	yes	no	0	166.7	113	28.34	...	122	12.61	186.9	121	8.41	10.1	3	2.73

```
churn_red.dtypes
```

```
state                              object
account length                      int64
area code                           int64
phone number                       object
international plan                 object
voice mail plan                    object
number vmail messages               int64
total day minutes                 float64
total day calls                     int64
total day charge                  float64
total eve minutes                 float64
total eve calls                     int64
total eve charge                  float64
total night minutes               float64
total night calls                   int64
total night charge                float64
total intl minutes                float64
total intl calls                    int64
total intl charge                 float64
number customer service calls       int64
Churn                              object
dtype: object
```

```
#Since we do not require few variables because we have to predict the churn score based on usage pattern so variables like
#"area code", "phone number" and "state" are not important so we will drop them.
churn_red.drop(['state', 'area code','phone number'], axis=1, inplace=True)
```

```
#Converting categorical variable into 1 and 0 i.e. for yes = 1 and for no = 0
lis = []
for i in range(0, churn_red.shape[1]):
    #print(i)
    if(churn_red.iloc[:,i].dtypes == 'object'):
        churn_red.iloc[:,i] = pd.Categorical(churn_red.iloc[:,i])
        #print(churn_red[[i]])
        churn_red.iloc[:,i] = churn_red.iloc[:,i].cat.codes
        churn_red.iloc[:,i] = churn_red.iloc[:,i].astype('object')

        lis.append(churn_red.columns[i])
```

```
churn_red.head(5)
```

	account length	international plan	voice mail plan	number vmail messages	total day minutes	total day calls	total day charge	total eve minutes	total eve calls	total eve charge	total night minutes	total night calls	total night charge	total intl minutes	total intl calls	total intl charge	number customer service calls	Churn
0	128	0	1	25	265.1	110	45.07	197.4	99	16.78	244.7	91	11.01	10.0	3	2.70	1	0
1	107	0	1	26	161.6	123	27.47	195.5	103	16.62	254.4	103	11.45	13.7	3	3.70	1	0
2	137	0	0	0	243.4	114	41.38	121.2	110	10.30	162.6	104	7.32	12.2	5	3.29	0	0
3	84	1	0	0	299.4	71	50.90	61.9	88	5.26	196.9	89	8.86	6.6	7	1.78	2	0
4	75	1	0	0	166.7	113	28.34	148.3	122	12.61	186.9	121	8.41	10.1	3	2.73	3	0

```
churn_red.dtypes
```

```
account length                  int64
international plan             object
voice mail plan                object
number vmail messages           int64
total day minutes             float64
total day calls                 int64
total day charge              float64
total eve minutes             float64
total eve calls                 int64
total eve charge              float64
total night minutes           float64
total night calls               int64
total night charge            float64
total intl minutes            float64
total intl calls                int64
total intl charge             float64
number customer service calls   int64
Churn                          object
dtype: object
```

Missing Value Analysis

Missing Value Analysis ¶

```
churn_red.isnull().sum()
```

```
account length                  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
dtype: int64
```

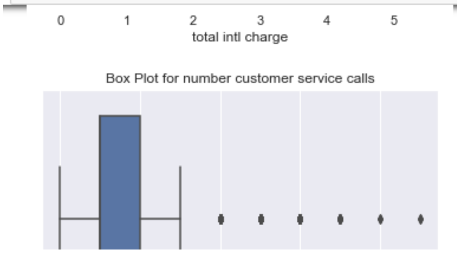
```
#As above we can clearly see that we don't have any missing values present in data set hence we will move to outlier analysis.
```

```
#Storing continuous and categorical variable in different objects
variable_num = ['account length', 'number vmail messages', 'total day minutes', 'total day calls', 'total day charge',
               'total eve minutes', 'total eve calls', 'total eve charge','total night minutes','total night calls',
                'total night charge','total intl minutes', 'total intl calls','total intl charge',
                'number customer service calls']

variable_cat = ['international plan', 'voice mail plan', 'Churn']
```

Outlier Analysis

```
#Outlier analysis using box plot method.
sns.set()
for i in variable_num:
    sns.boxplot(churn_red[i])
    plt.title("Box Plot for "+str(i))
    plt.show()
```



```
churn_red.shape
```

```
(5000, 18)
```

```
#detect and replace outliers with NA
#Extracting quartiles
for i in variable_num:
    q75,q25=np.percentile(churn_red[i],[75,25])



   ##calculating iqr
    iqr=q75-q25

    #calculating inner and outer fence
    minimum= q25-(iqr*1.5)
    maximum= q75+(iqr*1.5)

    #replace with NA
    churn_red.loc[churn_red[i]<minimum,i] = np.nan
    churn_red.loc[churn_red[i]>maximum,i] = np.nan
```

```
#Checking missing values created by outliers
churn_missing_value = churn_red.isnull().sum()
```

```
churn_missing_value
```

```
account length                 24
international plan              0
voice mail plan                 0
number vmail messages          60
total day minutes              34
total day calls                35
total day charge               34
total eve minutes              43
total eve calls                27
total eve charge               42
total night minutes            39
total night calls              43
total night charge             39
total intl minutes             72
total intl calls              118
total intl charge              72
number customer service calls  399
Churn                           0
dtype: int64
```

```
#create data frame with missing values
churn_missing_val = pd.DataFrame(churn_red.isnull().sum())
```

```
#calculating percentage
churn_missing_val['missing_percentage'] = (churn_missing_val['missing_percentage']/len(churn_red))*100
```

```
churn_missing_val
```

	variables	missing_percentage
0	account length	0.48
1	international plan	0.00
2	voice mail plan	0.00
3	number vmail messages	1.20
4	total day minutes	0.68
5	total day calls	0.70
6	total day charge	0.68
7	total eve minutes	0.86
8	total eve calls	0.54
9	total eve charge	0.84
10	total night minutes	0.78
11	total night calls	0.86
12	total night charge	0.78
13	total intl minutes	1.44
14	total intl calls	2.36
15	total intl charge	1.44

16	number customer service calls	7.98
17	Churn	0.00

```
churn_missing_val.to_csv('E:/Project/Churn reduction/missing_value_perc.csv')
```

```
#Creating copy
cr = churn_red.copy()
```

```
cr1=churn_red.copy()
```

```
#Creating missing value
churn_red['account length'].iloc[10]
```

```
65.0
```

```
#Imputation method
#actual value = 65
#Mean = 99.68
#Median = 100
#KNN = 79.46
```

```
churn_red['account length'].iloc[10] = np.nan
```

```
C:\Users\Aditya\Anaconda3\lib\site-packages\pandas\core\indexing.py:189: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy
  self._setitem_with_indexer(indexer, value)
```

```
churn_red['account length'].iloc[10]
```

```
nan
```

```
#Impute with mean
churn_red['account length'] = churn_red['account length'].fillna(churn_red['account length'].mean())
```

```
churn_red['account length'].iloc[10]
```

```
99.68522613065326
```

```
#Loading data set again
churn_red = cr.copy()
```

```
churn_red['account length'].iloc[10] = np.nan
```

```
C:\Users\Aditya\Anaconda3\lib\site-packages\pandas\core\indexing.py:189: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy
  self._setitem_with_indexer(indexer, value)
```

```
#Impute with median
churn_red['account length'] = churn_red['account length'].fillna(churn_red['account length'].median())
```

```
churn_red['account length'].iloc[10]
```

```
100.0
```

```
#impute with knn
#Loading data set again
churn_red = cr1.copy()
```

```
churn_red['account length'].iloc[10] = np.nan
```

```
C:\Users\Aditya\Anaconda3\lib\site-packages\pandas\core\indexing.py:189: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy
  self._setitem_with_indexer(indexer, value)
```

```
#Applying KNN imputation method
churn_red = pd.DataFrame(KNN(k = 3).fit_transform(churn_red), columns = churn_red.columns)
```

```
Imputing row 1/5000 with 0 missing, elapsed time: 4.595
Imputing row 101/5000 with 1 missing, elapsed time: 4.597
Imputing row 201/5000 with 0 missing, elapsed time: 4.599
Imputing row 301/5000 with 0 missing, elapsed time: 4.604
Imputing row 401/5000 with 0 missing, elapsed time: 4.605
Imputing row 501/5000 with 0 missing, elapsed time: 4.606
Imputing row 601/5000 with 0 missing, elapsed time: 4.608
Imputing row 701/5000 with 0 missing, elapsed time: 4.609
Imputing row 801/5000 with 0 missing, elapsed time: 4.610
Imputing row 901/5000 with 0 missing, elapsed time: 4.611
Imputing row 1001/5000 with 0 missing, elapsed time: 4.613
Imputing row 1101/5000 with 0 missing, elapsed time: 4.615
Imputing row 1201/5000 with 1 missing, elapsed time: 4.616
Imputing row 1301/5000 with 0 missing, elapsed time: 4.617
Imputing row 1401/5000 with 2 missing, elapsed time: 4.619
dexing.html#indexing-view-versus-copy 0 missing, elapsed time: 4.620
```

```
churn_red['account length'].iloc[10]
```

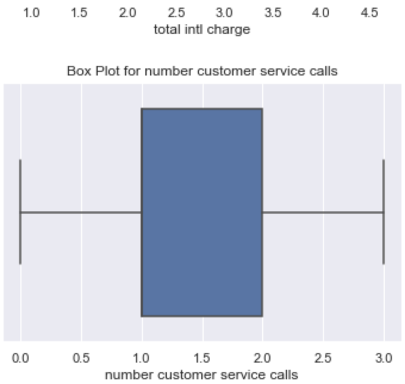
```
79.4621474951254
```

```
#Here we will go with knn imputation method as we have seen earlier that actual value was 65 and knn gave the result as
#79.46 which is close to actual value hence we will impute na with knn.
```

```
churn_red.isnull().sum()
```

```
account length                    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
dtype: int64
```

```
#Drawing box plot after replacement of outliers
sns.set()
for i in variable_num:
    sns.boxplot(churn_red[i])
    plt.title("Box Plot for "+str(i))
    plt.show()
```




```
#creating copy
df = churn_red.copy()
```

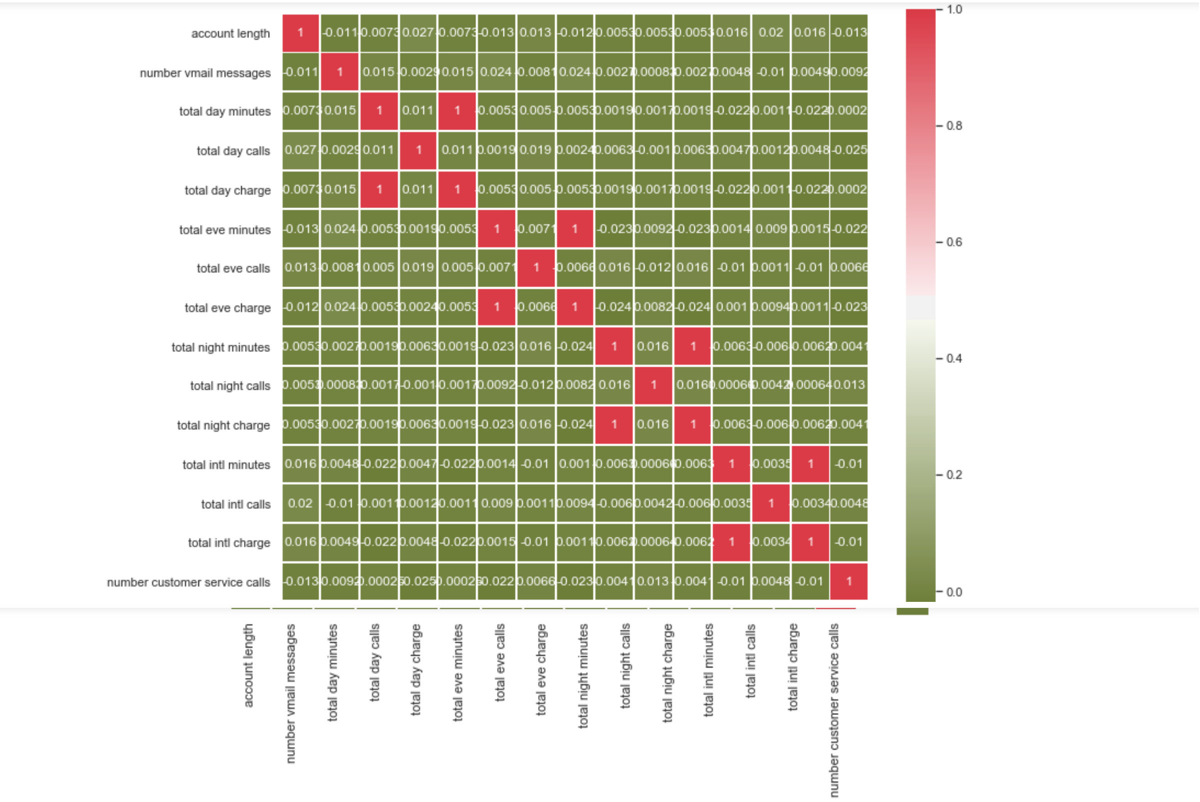
Feature Selection

```
br_corr = churn_red.loc[:,variable_num]
```

```
sns.set()
f, ax = plt.subplots(figsize=(12,10))

corr_matrix = br_corr.corr()

sns.heatmap(corr_matrix, mask = np.zeros_like(corr_matrix, dtype=np.bool), cmap = sns.diverging_palette(27100,10, as_cmap= True),
            annot = True , linewidths = 0.9,square = True, ax=ax)
```




```
#loop for chi square values
for i in variable_cat:
    print(i)
    chi2, p, dof, ex = chi2_contingency(pd.crosstab(churn_red['Churn'], churn_red[i]))
    print(p)
```

```
international plan
1.9443947474998577e-74
voice mail plan
7.164501780988496e-15
Churn
0.0
```

```
#From above correlation chart we can clearly see that there is high correlation between few variables hence we will drop few
#drop those variables. The variable we are going to drop are 'total day minutes' ,'total eve minutes','total night minutes'
#and 'total intl minutes'
```

```
churn_red = churn_red.drop(['total day minutes','total night minutes','total eve minutes', 'total intl minutes'], axis=1)
```

```
churn_red.shape
```

```
(5000, 14)
```

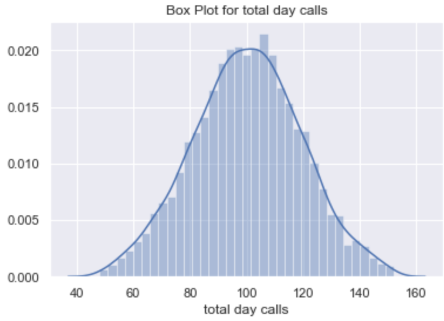
```
#Storing updated continuous and categorical variable in different objects
variable_num_update = ['account length', 'number vmail messages','total day calls', 'total day charge','total eve calls',
                    'total eve charge','total night calls','total night charge','total intl calls','total intl charge',
                    'number customer service calls']


variable_cat_update = ['international plan', 'voice mail plan', 'Churn']
```

Checking Distribution of data

```python
#Drawing histogram to check distribution of numeric varriable
sns.set()
for i in variable_num_update:
    sns.distplot(churn_red[i])
    plt.title("Box Plot for "+str(i))
    plt.show()
```

number vmail messages



Box Plot for total day charge

```python
#As from above distribution plots we can clearly see that data is normally distributed hence we can apply standardisation.
for i in variable_num_update:
    print(i)
    churn_red[i] = (churn_red[i] - churn_red[i].mean())/churn_red[i].std()
```

```
account length
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
```

```python
churn_red.head(5)
```

	account length	international plan	voice mail plan	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	Churn
0	0.729680	0.0	1.0	1.368521	0.510514	1.618908	-0.064526	-0.066337	-0.461033	0.909281	-0.596755	-0.117152	-0.329016	0.0
1	0.188929	0.0	1.0	1.445884	1.188521	-0.358847	0.142540	-0.104992	0.164915	1.110030	-0.596755	1.332756	-0.329016	0.0
2	0.961430	0.0	0.0	-0.565547	0.719132	1.204254	0.504906	-1.631853	0.217077	-0.774266	0.372473	0.738294	-1.383732	0.0
3	-0.403323	1.0	0.0	-0.565547	-1.523505	2.274040	-0.633958	-0.919454	-0.565358	-0.071647	1.341700	-1.451067	0.725700	0.0
4	-0.635073	1.0	0.0	-0.565547	0.666977	-0.261083	1.126105	-1.073776	1.103837	-0.276958	-0.596755	-0.073655	1.780416	0.0

Model Development

```
#Making copy
CR = churn_red.copy()
CR1 = churn_red.copy()
```

```
#replace target categories with Yes or No
#churn_red['Churn'] = churn_red['Churn'].replace(0, 'No')
#churn_red['Churn'] = churn_red['Churn'].replace(1, 'Yes')
```

```
churn_red.head(5)
```

	account length	international plan	voice mail plan	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	Churn
0	0.729680	0.0	1.0	1.368521	0.510514	1.618908	-0.064526	-0.066337	-0.461033	0.909281	-0.596755	-0.117152	-0.329016	0.0
1	0.188929	0.0	1.0	1.445884	1.188521	-0.358847	0.142540	-0.104992	0.164915	1.110030	-0.596755	1.332756	-0.329016	0.0
2	0.961430	0.0	0.0	-0.565547	0.719132	1.204254	0.504906	-1.631853	0.217077	-0.774266	0.372473	0.738294	-1.383732	0.0
3	-0.403323	1.0	0.0	-0.565547	-1.523505	2.274040	-0.633958	-0.919454	-0.565358	-0.071647	1.341700	-1.451067	0.725700	0.0
4	-0.635073	1.0	0.0	-0.565547	0.666977	-0.261083	1.126105	-1.073776	1.103837	-0.276958	-0.596755	-0.073655	1.780416	0.0

Decision Tree Classifier

```
#Divide data into train and test
X = churn_red.values[:, 0:13]
Y = churn_red.values[:,13]

X_train, X_test, y_train, y_test = train_test_split( X, Y, test_size = 0.2)
```

```
#Decision Tree
C50_model = tree.DecisionTreeClassifier(criterion='entropy').fit(X_train, y_train)
```

```
#predict new test cases
C50_Predictions = C50_model.predict(X_test)
```

```
#Decision tree and classifier. here we are using C5.0 model
clf = tree.DecisionTreeClassifier(criterion = 'entropy').fit(X_train, y_train)
```

```
#predict new test cases
y_pred = clf.predict(X_test)
```

Error Metrics

```
cm = confusion_matrix(y_test, y_pred)
```

```
cm
```

```
array([[800,  50],
       [ 47, 103]], dtype=int64)
```

```
cm = pd.crosstab(y_test, y_pred)
```

```
cm
```

col_0	0.0	1.0
row_0		
0.0	800	50
1.0	47	103

```
##Let us save TP, TN, FP, FN
TN = cm.iloc[0,0]
FP = cm.iloc[0,1]
FN = cm.iloc[1,0]
TP = cm.iloc[1,1]
```

```
TN, FP
```

```
(800, 50)
```

```
#Checking accuracy of model
accuracy_score(y_test, y_pred)*100
```

```
90.3
```

```
#Here we have used recall error metrics to see who are customers who have actually churned out. So, it will help our client
#to work on those customers by providing some good deals or by some other marketing mean to retain those customers back.
#Recall rate(True positivve)
(TP*100)/(TP+FN)
```

```
68.66666666666667
```

```
#Results
#Accuracy: 90.3
##Recall rate(True positivve): 68.66
```

Random Forest

```
#Loading copy again
churn_red = CR.copy()
```

```
churn_red.head(5)
```

	account length	international plan	voice mail plan	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	Churn
0	0.729680	0.0	1.0	1.368521	0.510514	1.618908	-0.064526	-0.066337	-0.461033	0.909281	-0.596755	-0.117152	-0.329016	0.0
1	0.188929	0.0	1.0	1.445884	1.188521	-0.358847	0.142540	-0.104992	0.164915	1.110030	-0.596755	1.332756	-0.329016	0.0
2	0.961430	0.0	0.0	-0.565547	0.719132	1.204254	0.504906	-1.631853	0.217077	-0.774266	0.372473	0.738294	-1.383732	0.0
3	-0.403323	1.0	0.0	-0.565547	-1.523505	2.274040	-0.633958	-0.919454	-0.565358	-0.071647	1.341700	-1.451067	0.725700	0.0
4	-0.635073	1.0	0.0	-0.565547	0.666977	-0.261083	1.126105	-1.073776	1.103837	-0.276958	-0.596755	-0.073655	1.780416	0.0

```
#Divide data into train and test
X = churn_red.values[:, 0:13]
Y = churn_red.values[:,13]

X_train, X_test, y_train, y_test = train_test_split( X, Y, test_size = 0.2)
```

```
#Random Forest
RF_model = RandomForestClassifier(n_estimators = 20).fit(X_train, y_train)
```

```
RF_Predictions = RF_model.predict(X_test)
```

```
#build confusion matrix
cm = confusion_matrix(y_test, y_pred)
```

```
cm
```

```
array([[724, 139],
       [123,  14]], dtype=int64)
```

```
cm = pd.crosstab(y_test, RF_Predictions)
```

```
cm
```

col_0	0.0	1.0
row_0		
0.0	857	6
1.0	75	62

```
#Let us save TP, TN, FP, FN
TN = cm.iloc[0,0]
FN = cm.iloc[1,0]
TP = cm.iloc[1,1]
FP = cm.iloc[0,1]
```

```
TN
```

```
857
```

```
#Accuracy score
accuracy_score(y_test,y_pred)*100
```

```
73.8
```

```
#Recall rate(True positivve)
(TP*100)/(TP+FN)
```

```
45.25547445255474
```

```
#Accuracy: 73.8
#Recall rate(True positivve) : 45.25
```

Logistic Regression

```
#Loading copy again
churn_red = CR1.copy()
```

```
#Let us prepare data for logistic regression
#replace target categories with Yes or No
churn_red['Churn'] = churn_red['Churn'].replace('No', 0)
churn_red['Churn'] = churn_red['Churn'].replace('Yes', 1 )
```

```
churn_red.head(5)
```

```
#Create logistic data. Save target variable first
churn_red_logit = pd.DataFrame(churn_red['Churn'])
```

```
churn_red_logit.shape
```

```
(5000, 1)
```

```
#Add continous variables
churn_red_logit = churn_red_logit.join(churn_red[variable_num_update])
```

```
churn_red_logit.shape
```

```
(5000, 12)
```

```
##Create dummies for categorical variables
variable_cat_update = ['international plan', 'voice mail plan']

for i in variable_cat_update:
    temp = pd.get_dummies(churn_red[i], prefix = i)
    churn_red_logit = churn_red_logit.join(temp)
```

```
#Splitting the data
Sample_Index = np.random.rand(len(churn_red_logit)) < 0.8

train = churn_red_logit[Sample_Index]
test = churn_red_logit[~Sample_Index]
```

```
#select column indexes for independent variables
train_cols = train.columns[1:16]
```

```
#Built Logistic Regression

logit = sm.Logit(train['Churn'], train[train_cols]).fit()

logit.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.351780
         Iterations 7
```

Logit Regression Results

Dep. Variable:	Churn	**No. Observations:**	4000
Model:	Logit	**Df Residuals:**	3986
Method:	MLE	**Df Model:**	13
Date:	Sat, 16 Feb 2019	**Pseudo R-squ.:**	0.1409
Time:	04:37:42	**Log-Likelihood:**	-1407.1
converged:	True	**LL-Null:**	-1637.9
		LLR p-value:	2.095e-90

	coef	std err	z	P>|z|	[0.025	0.975]
account length	0.0613	0.049	1.261	0.207	-0.034	0.157
number vmail messages	0.1781	0.195	0.915	0.360	-0.204	0.560
total day calls	0.0654	0.048	1.370	0.171	-0.028	0.159
total day charge	0.5765	0.051	11.264	0.000	0.476	0.677
total eve calls	0.0099	0.049	0.202	0.840	-0.086	0.106
total eve charge	0.2488	0.049	5.043	0.000	0.152	0.345
total night calls	-0.0251	0.049	-0.517	0.605	-0.120	0.070
total night charge	0.1777	0.048	3.665	0.000	0.083	0.273
total intl calls	-0.2224	0.051	-4.364	0.000	-0.322	-0.123
total intl charge	0.1357	0.049	2.759	0.006	0.039	0.232
number customer service calls	0.0205	0.049	0.421	0.674	-0.075	0.116
international plan_0.0	-1.7897	nan	nan	nan	nan	nan
international plan_1.0	0.1060	nan	nan	nan	nan	nan
voice mail plan_0.0	-0.1380	nan	nan	nan	nan	nan
voice mail plan_1.0	-1.5456	nan	nan	nan	nan	nan

```
#Predict test data
#We are creating new variable where we will calculate and store actual probabilty
test['Actual_prob'] = logit.predict(test[train_cols])
```

```
C:\Users\Aditya\Anaconda3\lib\site-packages\ipykernel_launcher.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy
```

```
#Here we are probabilities into classified form of yes or no.
test['ActualVal'] = 1
test.loc[test.Actual_prob < 0.5, 'ActualVal'] = 0
```

```
C:\Users\Aditya\Anaconda3\lib\site-packages\ipykernel_launcher.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy
  """Entry point for launching an IPython kernel.
C:\Users\Aditya\Anaconda3\lib\site-packages\pandas\core\indexing.py:543: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy
  self.obj[item] = s
```

```
#Build confusion matrix
CM = pd.crosstab(test['Churn'], test['ActualVal'])
```

```
CM
```

ActualVal	**0**	**1**
Churn		
0.0	846	17
1.0	118	19

```
#let us save TP, TN, FP, FN
TN = CM.iloc[0,0]
FN = CM.iloc[1,0]
TP = CM.iloc[1,1]
FP = CM.iloc[0,1]
```

```
TN
```

846

```
#check accuracy of model
((TP+TN)*100)/(TP+TN+FP+FN)
```

86.5

```
#check accuracy of model
((TP+TN)*100)/(TP+TN+FP+FN)
```

```
86.5
```

```
#Recall rate(True positivve)
(TP*100)/(TP+FN)
```

```
13.86861313868613
```

```
#Accuracy: 86.5
#Recall rate(True positivve) : 13.86
```

```
#Decision tree classifier model is performing better than random forest and logistic regression. Hence we will apply decision
#tree classifier.
```

```
#Loading copy again
churn_red = df.copy()
```

```
churn_red['Churn'] = churn_red['Churn'].replace(0, 'No')
churn_red['Churn'] = churn_red['Churn'].replace(1, 'Yes')
```

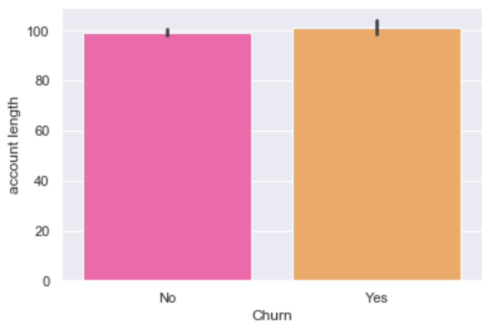
```
churn_red['international plan'] = churn_red['international plan'].replace(0, 'No')
churn_red['international plan'] = churn_red['international plan'].replace(1, 'Yes')
```

```
churn_red['voice mail plan'] = churn_red['voice mail plan'].replace(0, 'No')
churn_red['voice mail plan'] = churn_red['voice mail plan'].replace(1, 'Yes')
```

```
sns.barplot(x='Churn', y='account length', palette = 'spring',  data = churn_red)
```

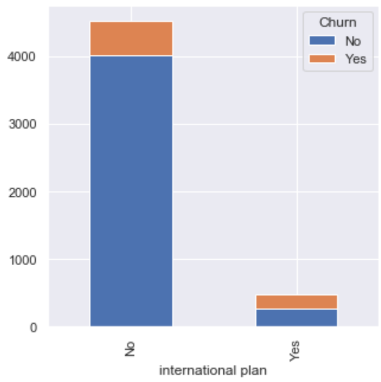
```
C:\Users\Aditya\Anaconda3\lib\site-packages\scipy\stats\stats.py:1713: FutureWarning: Using a non-tuple sequence for multi
nsional indexing is deprecated; use `arr[tuple(seq)]` instead of `arr[seq]`. In the future this will be interpreted as an
y index, `arr[np.array(seq)]`, which will result either in an error or a different result.
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x23d08496a90>
```



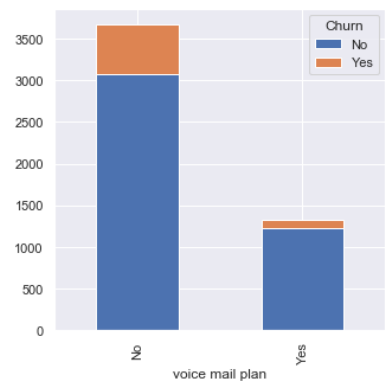

```
churn_red.groupby(["international plan", "Churn"]).size().unstack().plot(kind='bar', stacked=True, figsize=(5,5))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x23d077d24e0>
```



```
churn_red.groupby(["voice mail plan", "Churn"]).size().unstack().plot(kind='bar', stacked=True, figsize=(5,5))
```

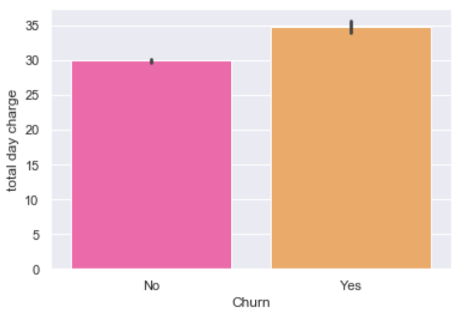
<matplotlib.axes._subplots.AxesSubplot at 0x23d198a4940>




```
sns.barplot(x='Churn', y='total day charge', palette = 'spring',  data = churn_red)
```

```
C:\Users\Aditya\Anaconda3\lib\site-packages\scipy\stats\stats.py:1713: FutureWarning: Using a non-tuple sequence for multidime
nsional indexing is deprecated; use `arr[tuple(seq)]` instead of `arr[seq]`. In the future this will be interpreted as an arra
y index, `arr[np.array(seq)]`, which will result either in an error or a different result.
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```

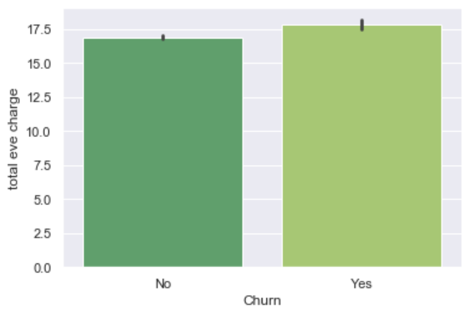
<matplotlib.axes._subplots.AxesSubplot at 0x23d1984a8d0>




```
sns.barplot(x='Churn', y='total eve charge', palette = 'summer',  data = churn_red)
```

```
C:\Users\Aditya\Anaconda3\lib\site-packages\scipy\stats\stats.py:1713: FutureWarning: Using a non-tuple sequence for multidime
nsional indexing is deprecated; use `arr[tuple(seq)]` instead of `arr[seq]`. In the future this will be interpreted as an arra
y index, `arr[np.array(seq)]`, which will result either in an error or a different result.
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```

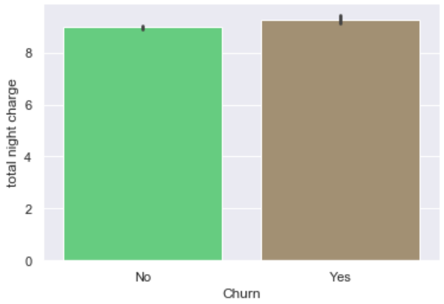
<matplotlib.axes._subplots.AxesSubplot at 0x23d140916a0>



```
sns.barplot(x='Churn', y='total night charge', palette = 'terrain',  data = churn_red)
```

```
C:\Users\Aditya\Anaconda3\lib\site-packages\scipy\stats\stats.py:1713: FutureWarning: Using a non-tuple sequence for multidime
nsional indexing is deprecated; use `arr[tuple(seq)]` instead of `arr[seq]`. In the future this will be interpreted as an arra
y index, `arr[np.array(seq)]`, which will result either in an error or a different result.
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```

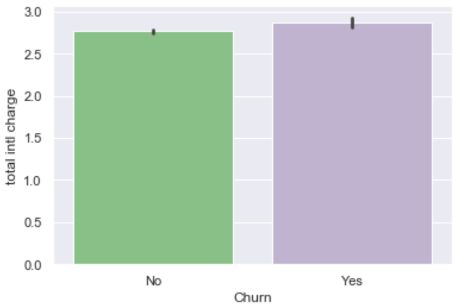
```
<matplotlib.axes._subplots.AxesSubplot at 0x23d18fad470>
```




```
sns.barplot(x='Churn', y='total intl charge', palette = 'Accent',  data = churn_red)
```

```
C:\Users\Aditya\Anaconda3\lib\site-packages\scipy\stats\stats.py:1713: FutureWarning: Using a non-tuple sequence for multidime
nsional indexing is deprecated; use `arr[tuple(seq)]` instead of `arr[seq]`. In the future this will be interpreted as an arra
y index, `arr[np.array(seq)]`, which will result either in an error or a different result.
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x23d18fd4550>
```



Churn

```
sns.barplot(x='Churn', y='number customer service calls', palette = 'dark',  data = churn_red)
```

```
C:\Users\Aditya\Anaconda3\lib\site-packages\scipy\stats\stats.py:1713: FutureWarning: Using a non-tuple sequence for multidime
nsional indexing is deprecated; use `arr[tuple(seq)]` instead of `arr[seq]`. In the future this will be interpreted as an arra
y index, `arr[np.array(seq)]`, which will result either in an error or a different result.
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x23d1a40f828>
```