Churn reduction

October 22, 2018

1 Chapter 1

1.1 Introduction

1.1.1 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. This problem statement is targeted at enabling churn reduction using analytics concepts.

Input Data *Train_data.csv, Test_data.csv*

Output Data *Churn_Predicted (R).csv, Churn_Predicted (Python).csv, Performance of models in the given Test_data (Python).csv, Performance of models in the given Test_data (R).csv*

1.1.2 Data

Our task is to build classification models which will tell us wheather a customer will churn or not. Given below is a sample of the data set that we are using to predict the churn reduction:

Churn Table Sample Data (1-5 columns)

state	account length	area code	phone number	international plan
KS	128	415	382-4657	no
OH	107	415	371-7191	no
NJ	137	415	358-1921	no
OH	84	408	375-9999	yes
OK	75	415	330-6626	yes

Churn Table Sample Data (6-10 columns)

voice mail plan	number vmail messages		total day minutes	total day calls
yes	25	265.1	110	45.07
yes	26	161.6	123	27.47
no	0	243.4	114	41.38
no	0	299.4	71	50.9
no	0	166.7	113	28.34

Churn Table Sample Data (10-15 columns)

total eve minutes	total eve calls	total eve charge	total night minutes	total night calls
197.4	99	16.78	244.7	91
195.5	103	16.62	254.4	103
121.2	110	10.3	162.6	104
61.9	88	5.26	196.9	89
148.3	122	12.61	186.9	121

Churn Table Sample Data (16-21 columns)

total night charge	total intl minutes	total intl calls	total intl charge	number customer service calls	Churn
11.01	10	3	2.7	1	False.
11.45	13.7	3	3.7	1	False.
7.32	12.2	5	3.29	0	False.
8.86	6.6	7	1.78	2	False.
8.41	10.1	3	2.73	3	False.

Hence, we have below 21 predictors which we will use to predict the Churn rate.
Predictors

S. No.	Predictors
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

S. No.	Predictors
20.	number customer service calls
21.	Churn

2 Chapter 2

2.1 Exploratory Data Analysis

2.1.1 Univariate Analysis

Exploratory data analysis is most important step before we can apply any machine learning model. It specifically cleans up the data for the model, so that the model can work as expected from it, and not get any unexplained predictions. It involves various steps which are explained below. But before doing all the below steps, we can do simple analysis too by looking at the data. For example, if we check one of the variable is *phone number*. We can deduce from the business problem that this data is explaining customers of telecom company. Hence, "phone number" will be unique for all the observations. Hence there won't be any pattern in this variable that will explain the result in our dependent variable *Churn*. So we can drop this variable.

Similarly for more complex exploratory data analysis, we perform following steps to filter out variables and make them model ready.

Missing Values Analysis

What are missing values? In real world, the data are not always complete. There are numerous times when we get some observations with one or many variables values as missing. These data are not helpful in creating model and doing analysis. Hence, they need to be taken care of at the start of model making, or in the pre-processing stage.

In our data, we did not find any missing data. It can be observed from below missing values analysis table.

Missing Values count for each variable

Variables	Missing_Values_Count
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

Variables	Missing_Values_Count
total intl minutes	0
total intl calls	0
total intl charge	0
number customer service calls	0
Churn	0

Code used to evaluate missing values in the data:

```
#Create dataframe with missing percentage
missing_val = pd.DataFrame(dataset.isnull().sum())

#Reset index
missing_val = missing_val.reset_index()

#Rename variable
missing_val = missing_val.rename(columns = {'index': 'Variables', 0: 'Missing_percentage'})
```

Hence from our data, we can see that we don't need to apply for any missing values imputation methods. Also we checked for those data which may have total (day, evening, night & international) minutes as 0 and other related attributes like calls or charge as not 0, but we did not find those kind of data too.

Outlier Analysis

What are outliers? Basically when the observations has some inconsitent data with the rest of dataset. It can be caused by number of things like poor data quality or contamination, low quality measurements, malfunction equipment and manual errors. Sometimes they are correct data but some exceptional cases.

It can be detected by lots of ways, some of which are: * Graphical Tools * Box plot * QQ Plot * Scatter Plots * Statistical Techniques * Grubb's Technique (It will only work on those data which are uniformly distributed.)

We are going to use the graphical tools to detect if we have any outliers in our data.

Box Plots

Refer to Appendix 1A (Code and figure)

QQ Plots

Refer to Appendix 1B (Code and figure)

Probability Distribution Curve

Refer to Appendix 1C (Code and figure)

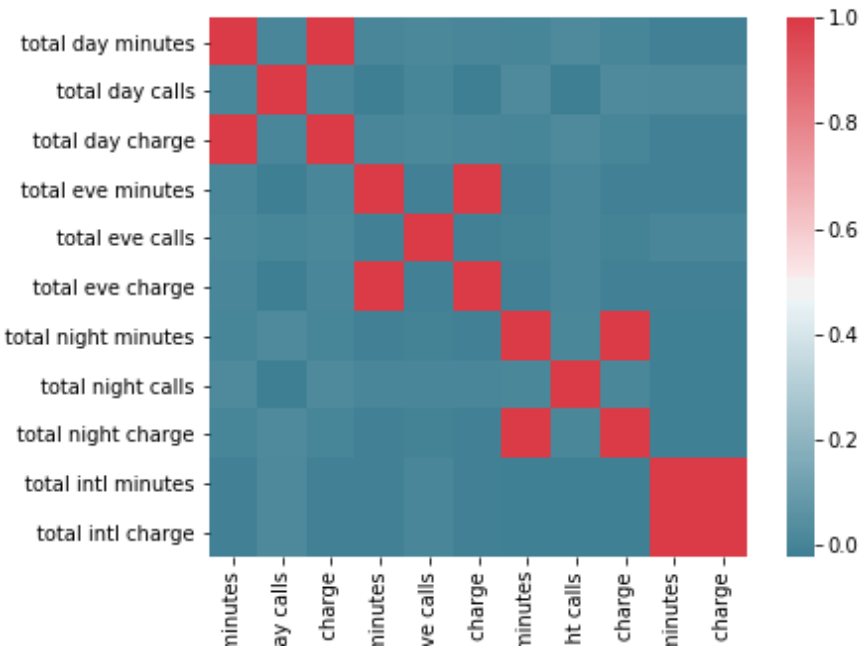
Feature Scaling

What is feature scaling? When the range of data for one variable is in different range than the other variable's data, then the model may work in incorrect manner. This reduces model's efficiency and gives us unexplained and incorrect results.

Since the data which we got, are in the range of 0-200s, we can leave feature scaling.

2.1.2 Multivariate Analysis

Univariate analysis only explains the behaviour of a single variable. But this doesn't necessarily explain the variable if we take a group of variable and get an explaination. For that, we perform



Correlation Analysis

$$r = \frac{1}{n-1} \sum \left(\frac{x - \overline{x}}{s_x} \right) \left(\frac{y - \overline{y}}{s_y} \right)$$

Formula

Multivariate Analysis. This analysis is required to explain if the predictors has any correlation in between the variable. This helps is to reject predictors with high correlation, because that will hamper the performance of our model.

For Normal Data

We can explain the correlation between normalised data by plotting heat maps. This can be seen in the following diagram.

The correlation analysis explains how much correlated to each other. If a data is highly positively correlated or higly negatively correlated, then the model performances degrades. It works like this, if two variables convey same message to a model, then it will be redundant task for the model to cater each variable for basically the same pattern. Hence it's important to remove variable or predictors with high correlation.

The correlation can be calculated by below formula:

Now as we can see in the heatmap, the scale on the right explains the colors with respect to correlation between the two variables. From the diagram, we can deduce that the variables that

are higly correlated are:

Variables	Variables they are higly correlated with
total day minutes	*total day charge*
total eve minutes	*total eve minutes*
total night minutes	*total night minutes*
total intl minutes	*total intl minutes*

Hence we can choose either of the variables from each row, as they convey the same meaning.

For Categorical Data

The categorical variables needs different analysis for correlation tests. One of the techniques used for analysing correlation between correlation is **Chi Square Test**.

We have set the threshold for rejecting null hypothesis as **0.05**. Below code has been used to calculate p-values of each column:

```
from scipy.stats import chi2_contingency

#cat_names include all the categorical variable
for i in cat_names:
    print(i)
    chi2, p, dof, ex = chi2_contingency(pd.crosstab(dataset['Churn'], dataset[i]))
    print(p)
```

Based on that, we rejected some of the categorical variable from the dataset. Now are data is cleaned for splitting operations.

2.1.3 Train Test Split

By looking at the data, we can understand that data is largely skewed, i.e. the ratio of False. and True. are not proportionally distributed. If you look at the below count,

Churn	Count	Percentage
False.	2850	85.50 %
True.	483	14.50 %

Hence we can't randomly split the data because it can give incorrect training data for the model. We have to maintain the same ratio while we do the train test split.

Hence, we will go for **Stratified Sampling**. With stratified sampling, we divide the data into separate groups, called strata. Then, a probability sample (often a simple random sample) is drawn from each group.

Stratified sampling has several advantages over simple random sampling. For example, using stratified sampling, it may be possible to reduce the sample size required to achieve a given precision. Or it may be possible to increase the precision with the same sample size.

It can be applied using train_test_split, and by passing the array to *stratify* parameter, like below.

```
from sklearn.model_selection import train_test_split
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, stratify=y, test_size=0.25)
```

3 Chapter 3

3.1 Model Selection

In exploratory analysis stage, we arrived at the conclusion that the problem requires **Classification Model** to predict our dependent variable, i.e. *Churn*.

We have applied multiple classification model which will be explained in below sections. For choosing a model, we are going to construct **Confusion matrix** which will help us trying to evaluate the model performance using various metrics like Accuracy, Precision, Prevalence, True Positive Rate etc.

3.1.1 Logical Regression

Logistic Regression is one of the basic and popular algorithm to solve a classification problem. It is named as 'Logistic Regression', because it's underlying technique is quite the same as Linear Regression. The term "Logistic" is taken from the Logit function that is used in this method of classification.

After applying the model in our cleaned data, we get the following confusion matrix:

n=834	Predicted False	Predicted True
Actual False	693	20
Actual True	100	21

From the above confusion matrix we can calculate below metrics for the model:

ModelName	Accuracy^	True Positive Rate^	Precision^	Prevalence^
Logistic Regression	0.856115	0.173554	0.512195	0.134373125

^*The meaning of each metrics are explained in Conclusion. Please refer to it for more details*

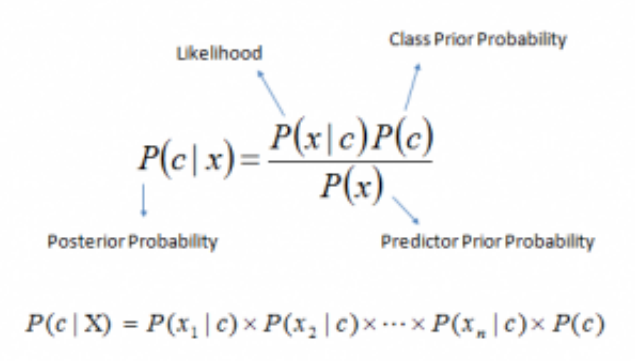
These metrics tell us that model performed pretty well on the data and was able to provide an accuracy of 85%. It's not bad for a start. Let's look into how other models perform on the given data.

3.1.2 K-Nearest Neighbors

K-Nearest Neighbors or KNN algorithm is one of the simplest classification algorithm and it is one of the most used learning algorithms. Its purpose is to use a database in which the data points are separated into several classes to predict the classification of a new sample point.

KNN Algorithm is based on feature similarity: How closely out-of-sample features resemble our training set determines how we classify a given data point. It is used for **classification**—the output is a class membership (predicts a class—a discrete value). An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors.

After applying the model in our cleaned data, we get the following confusion matrix:



Formula

n=834	Predicted False	Predicted True
Actual False	706	7
Actual True	85	36

From the above confusion matrix we can calculate below metrics for the model:

ModelName	Accuracy^	True Positive Rate^	Precision^	Prevalence^
Logistic Regression	0.856115	0.173554	0.512195	0.134373125
KNN	0.889688	0.297521	0.837209	0.134373125

^*The meaning of each metrics are explained in Conclusion. Please refer to it for more details*

3.1.3 Naive Bayes

It is a classification technique based on Bayes' Theorem with an assumption of independence among predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature.

For example, a fruit may be considered to be an apple if it is red, round, and about 3 inches in diameter. Even if these features depend on each other or upon the existence of the other features, all of these properties independently contribute to the probability that this fruit is an apple and that is why it is known as 'Naive'.

It can be explained using below formula also,

where, * P(c | x) is the posterior probability of class (c, target) given predictor (x, attributes). * P(c) is the prior probability of class. * P(x | c) is the likelihood which is the probability of predictor given class. * P(x) is the prior probability of predictor.

After applying the model in our cleaned data, we get the following confusion matrix:

n=834	Predicted False	Predicted True
Actual False	666	47
Actual True	75	46

From the above confusion matrix we can calculate below metrics for the model:

ModelName	Accuracy^	True Positive Rate^	Precision^	Prevalence^
Logistic Regression	0.856115	0.173554	0.512195	0.134373125
KNN	0.889688	0.297521	0.837209	0.134373125
Naive Bayes	0.853717	0.380165	0.494624	0.134373125

The general motive of using Decision Tree is to create a training model which can use to predict class or value of target variables by **learning decision rules** inferred from prior data(training data).

The understanding level of Decision Trees algorithm is so easy compared with other classification algorithms. The decision tree algorithm tries to solve the problem, by using tree representation. Each **internal node** of the tree corresponds to an attribute, and each **leaf node** corresponds to a class label.

After applying the model in our cleaned data, we get the following confusion matrix:

n=834	Predicted False	Predicted True
Actual False	672	41
Actual True	31	90

From the above confusion matrix we can calculate below metrics for the model:

ModelName	Accuracyˆ	True Positive Rateˆ	Precisionˆ	Prevalenceˆ
Logistic Regression	0.856115	0.173554	0.512195	0.134373125
KNN	0.889688	0.297521	0.837209	0.134373125
Naive Bayes	0.853717	0.380165	0.494624	0.134373125
Decision Tree Classifier	0.913669	0.743802	0.687023	0.134373125

ˆ*The meaning of each metrics are explained in Conclusion. Please refer to it for more details*

3.1.5 Random Forest Classifier

Random Forest is the one of the most used algorithms, because of it's simplicity and the fact that it can be used for both classification and regression tasks, just like Decision Trees.

It is a supervised learning algorithm. The *forest* it builds, is an ensemble of Decision Trees, most of the time trained with the *bagging* method. The general idea of the bagging method is that a combination of learning models increases the overall result.

In a nut shell, we can say that Random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction.

After applying the model in our cleaned data, we get the following confusion matrixˆ:

n=834	Predicted False	Predicted True
Actual False	702	11
Actual True	53	68

From the above confusion matrix we can calculate below metrics for the model:

ModelName	Accuracyˆ	True Positive Rateˆ	Precisionˆ	Prevalenceˆ
Logistic Regression	0.856115	0.173554	0.512195	0.134373125
KNN	0.889688	0.297521	0.837209	0.134373125
Naive Bayes	0.853717	0.380165	0.494624	0.134373125
Decision Tree Classifier	0.913669	0.743802	0.687023	0.134373125
Random Forest Classifier	0.961007798	0.727678571	0.976047904	0.134373125

ˆThe meaning of each metrics are explained in Conclusion. Please refer to it for more details

4 Conclusion

4.1 Model Evaluation

As we have seen in previous section, as we tried on different models, we cam across different results. To evaluate a classification model, a confusion matrix is constructed first.

A confusion matrix consists of following values:

n=Total	Predicted False	Predicted True
Actual False	*True Negative (TN)*	*False Positve (FP)*
Actual True	*False Negative (FN)*	*True Positive (TN)*

Based on above matrix, we used below metrics to evaluate a given model:

- Accuracy: Overall, how often is the classifier correct?
  - (TP+TN)/total
- True Positive Rate: When it's actually yes, how often does it predict yes?
  - TP/actual yes
  - Also known as "Sensitivity" or "Recall"
- Precision: When it predicts yes, how often is it correct?
  - TP/predicted yes
- Prevalence: How often does the yes condition actually occur in our sample?
  - actual yes/total

These metrics help us to choose from different models we create. Based on these, we came across below data, regarding the performance of a model.

ModelName	Accuracyˆ	True Positive Rateˆ	Precisionˆ	Prevalenceˆ
Logistic Regression	0.856115	0.173554	0.512195	0.134373125
KNN	0.889688	0.297521	0.837209	0.134373125
Naive Bayes	0.853717	0.380165	0.494624	0.134373125
Decision Tree Classifier	0.913669	0.743802	0.687023	0.134373125
Random Forest Classifier	0.961007798	0.727678571	0.976047904	0.134373125

4.2 Model Selection

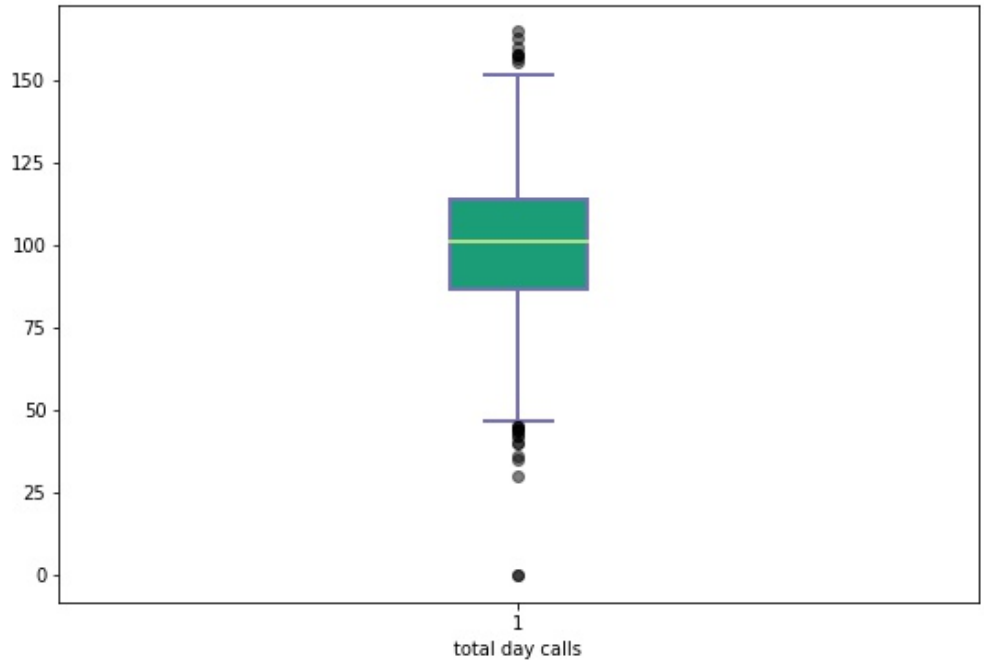
From the above table, we can clearly see that accuracy & precision wise **Random Forest Classifier** beats the rest of the model. Hence we will go with Random Forest to predict the data.

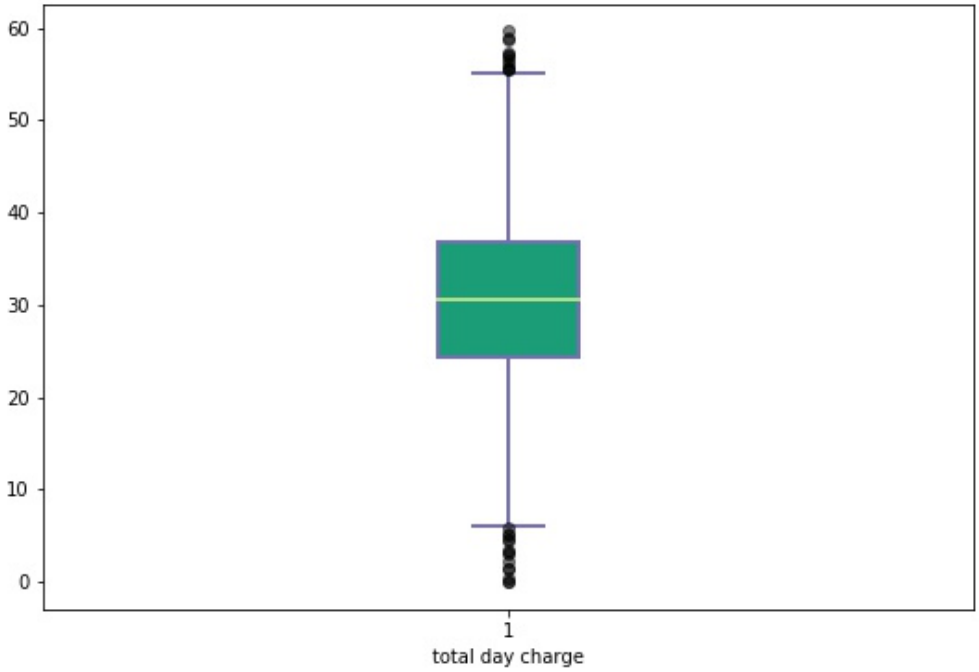
5 Appendix 1 - Univariate Analysis Graphs

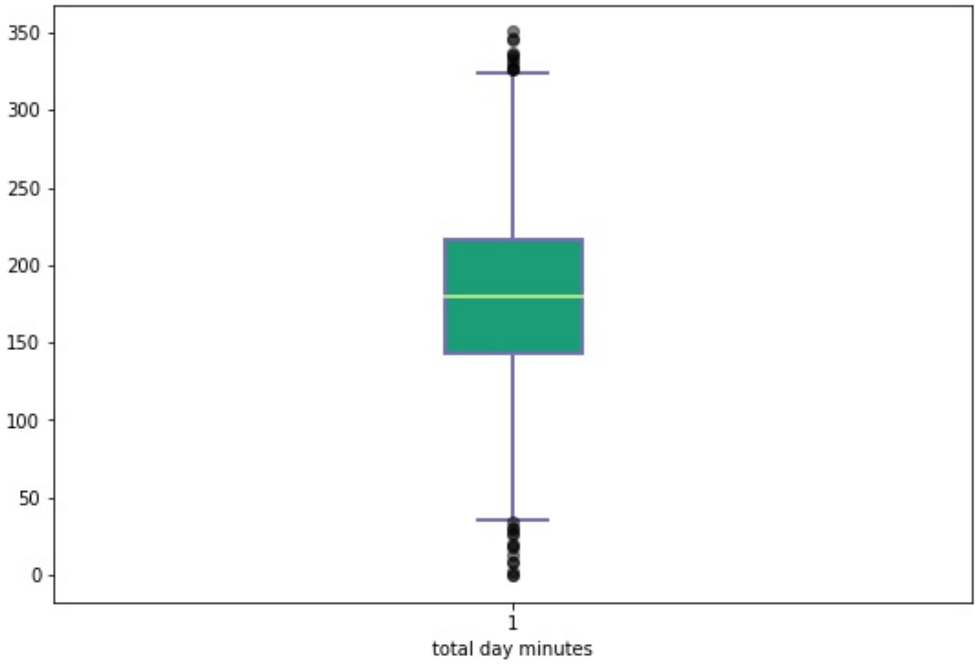
5.1 A) Box Plots

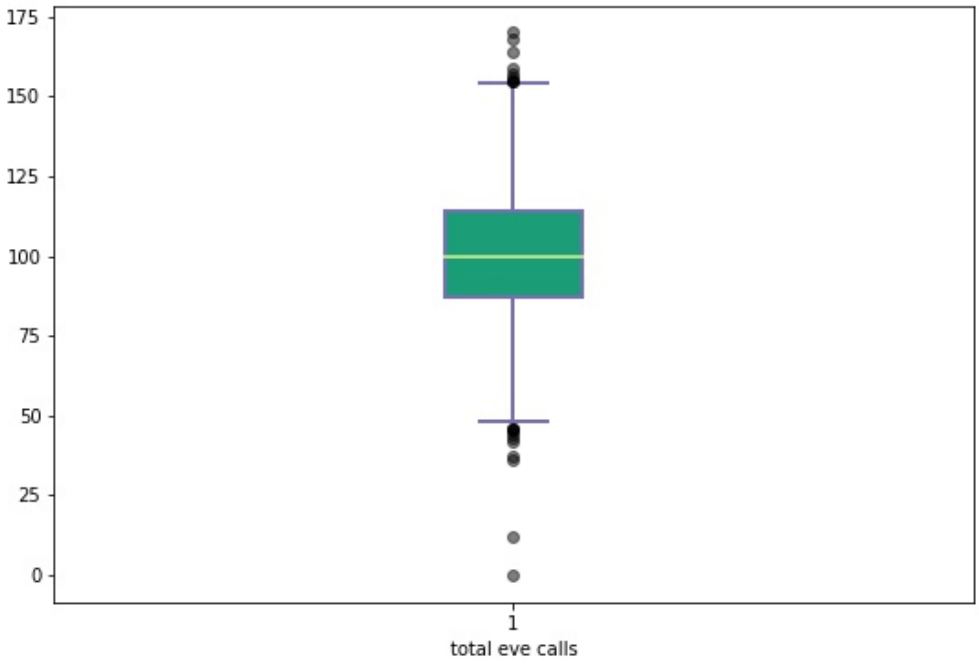
What are Box plots? A box plot is a method for graphically depicting groups of numerical data through their quartiles.

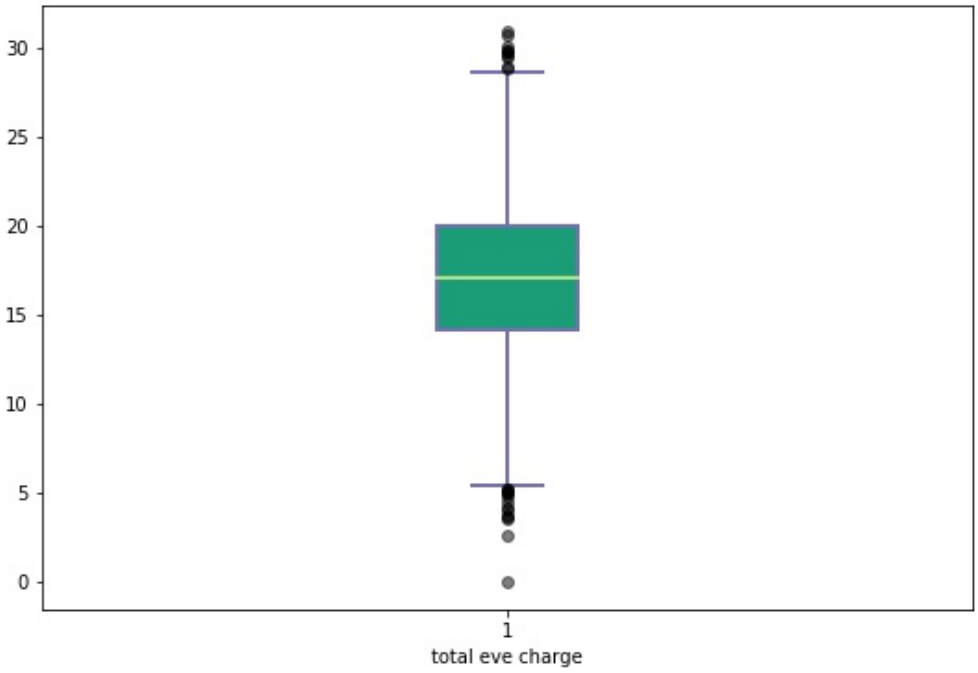
Box plots for each numerical data are shown below:

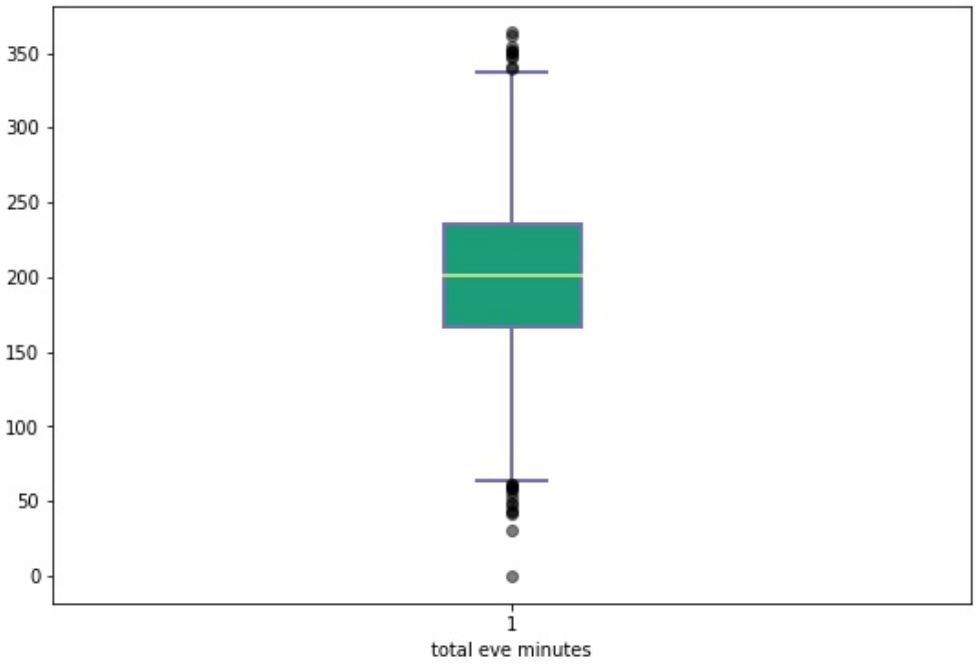


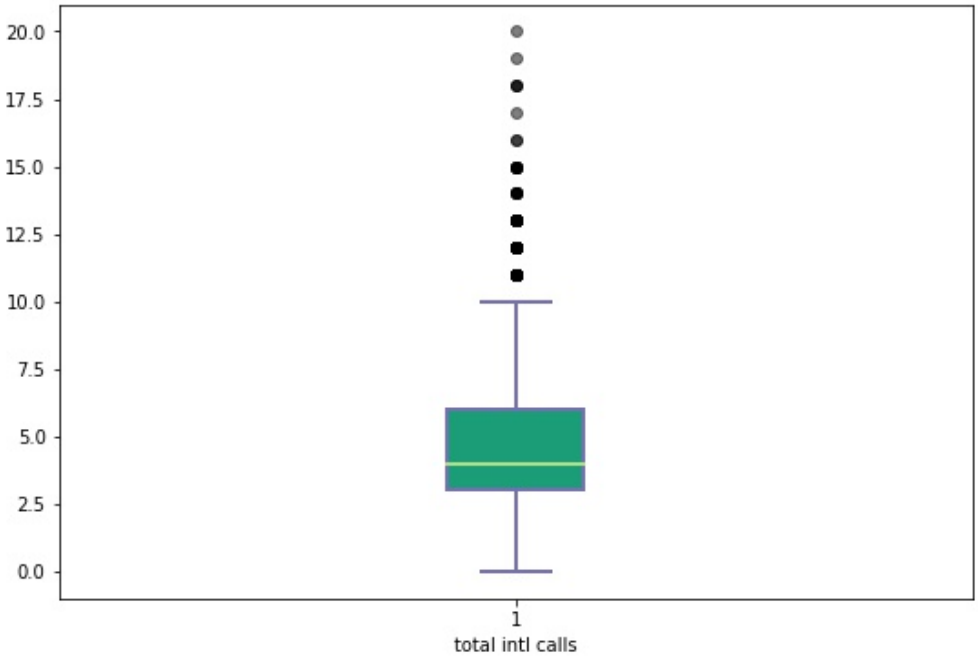


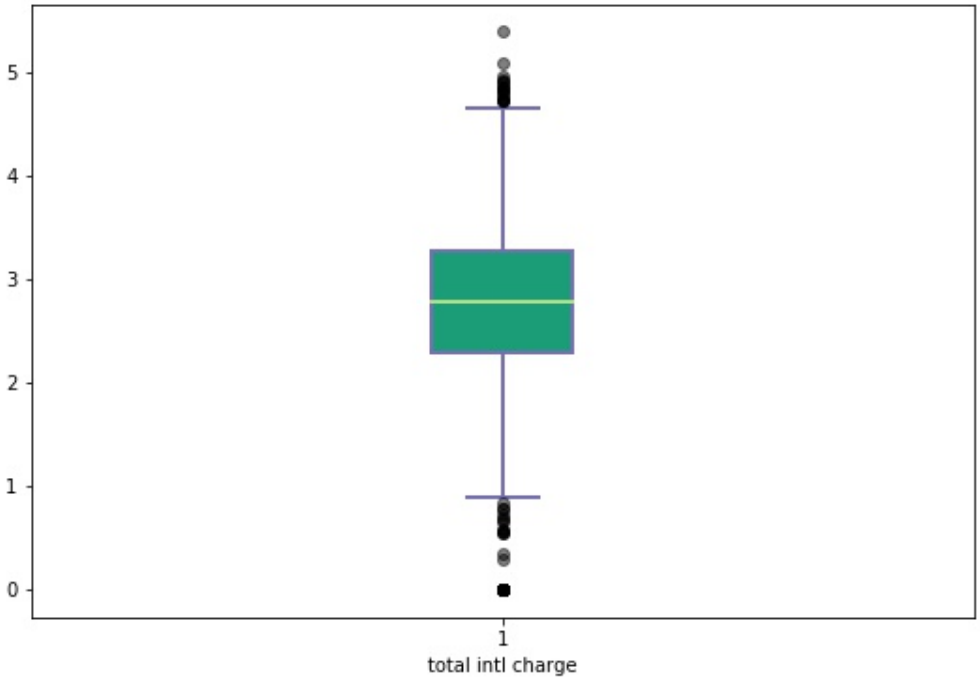

350
300
250
200
150
100
50
0
1
total day minutes

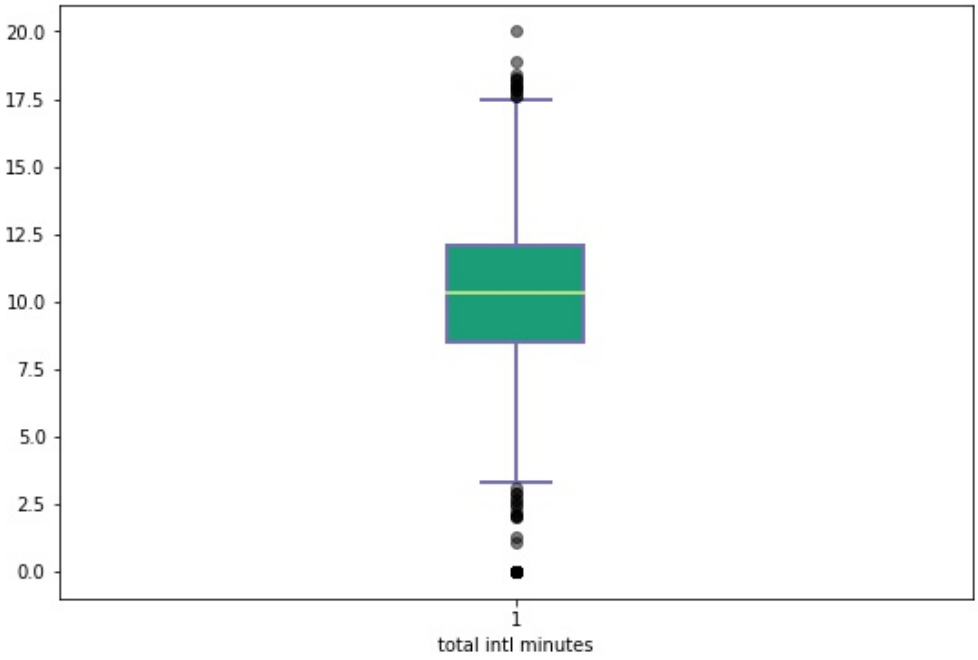

175
150
125
100
75
50
25
0
1
total eve calls

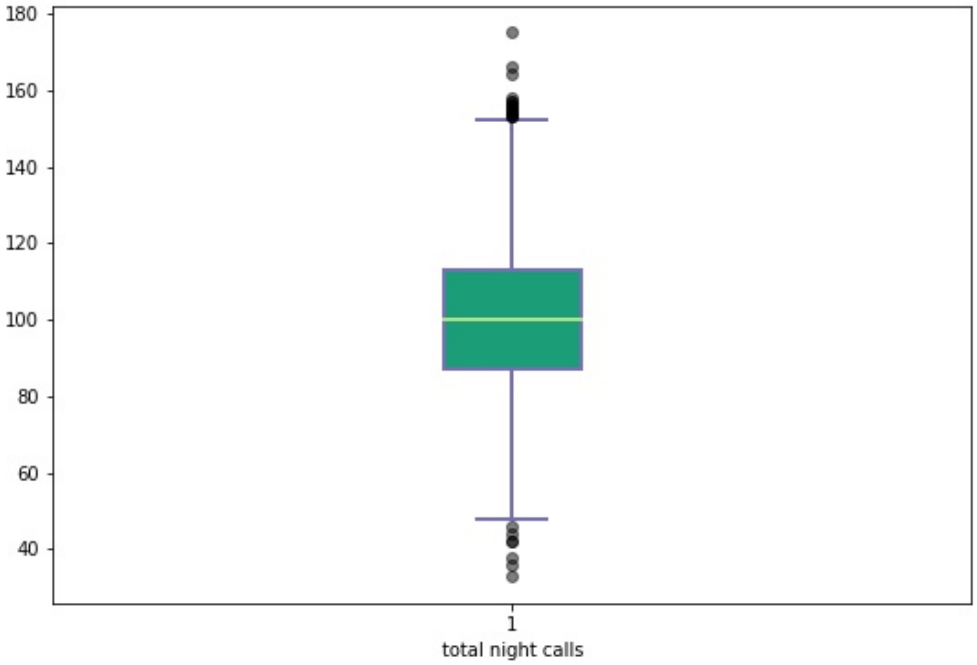

30
25
20
15
10
5
0
1
total eve charge

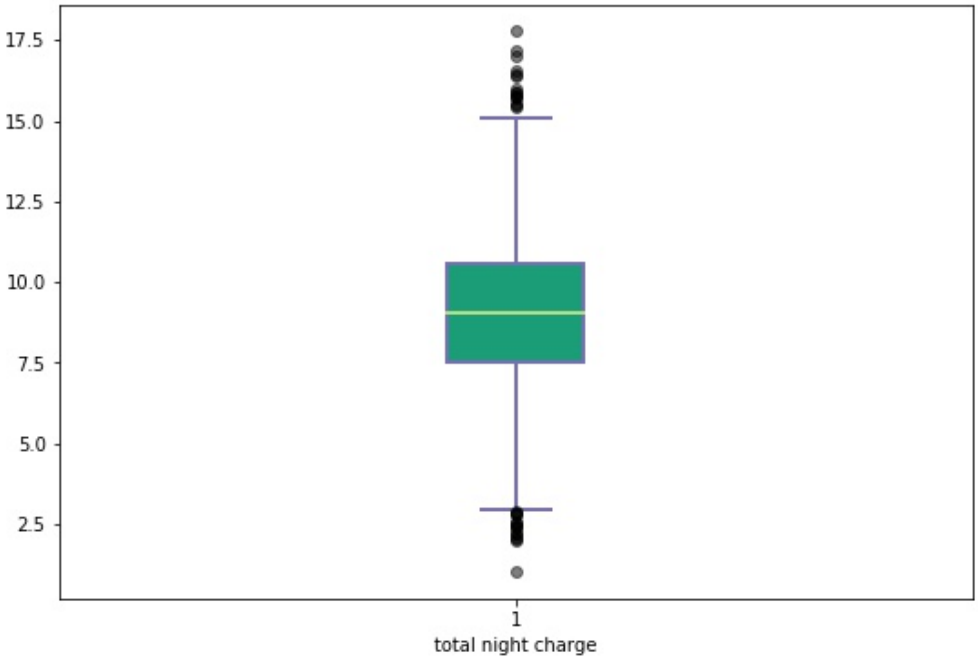

350
300
250
200
150
100
50
0
1
total eve minutes

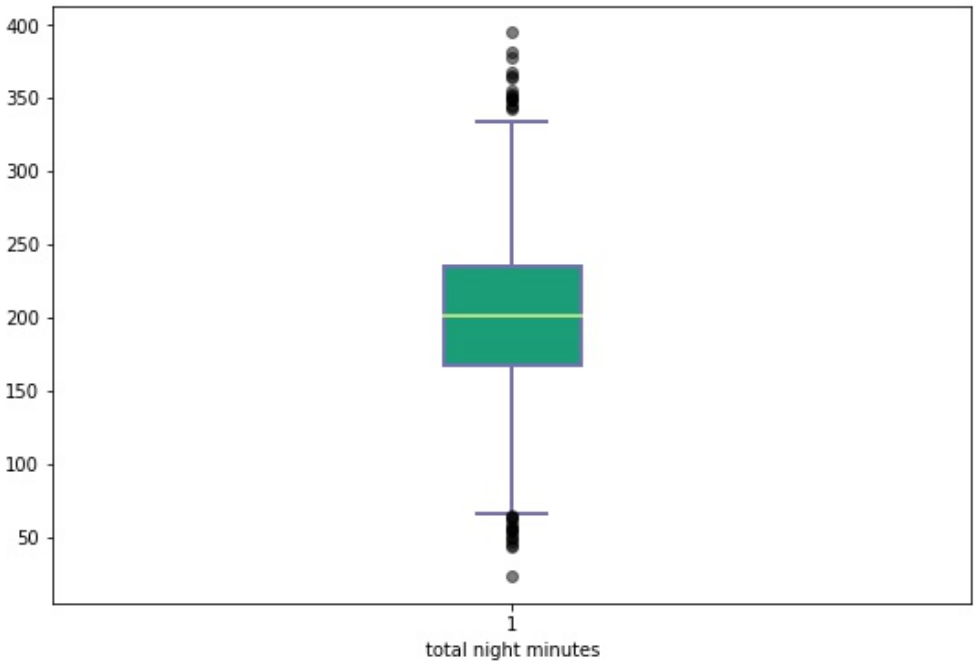

20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
1
total intl calls


5
4
3
2
1
0
1
total intl charge


20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
1
total intl minutes


180
160
140
120
100
80
60
40
1
total night calls





Code used to create boxplots:

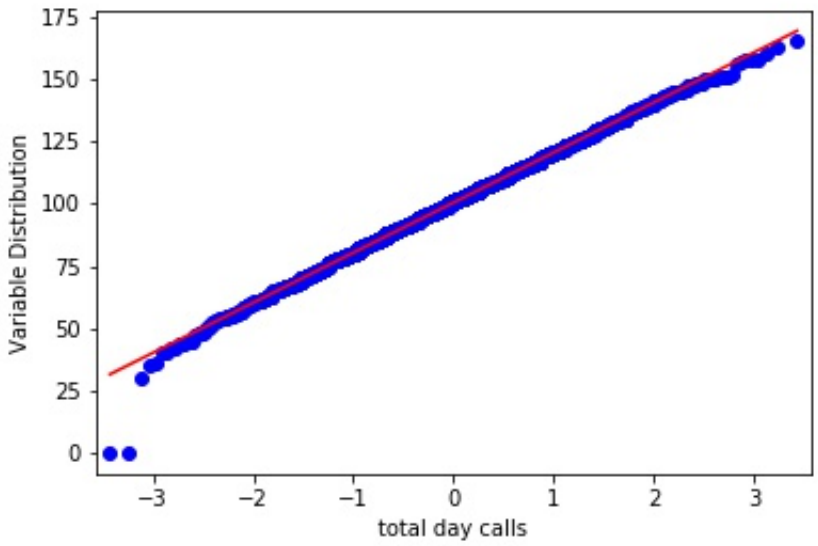
```
def CreateBoxPlot(dataset, columnNames):
    fig = plt.figure(1, figsize=(9, 6))
    #ax = fig.add_subplot(111)
    bp = plt.boxplot(dataset[columnNames], patch_artist=True)

    for box in bp['boxes']:
        # change outline color
        box.set( color='#7570b3', linewidth=2)
```

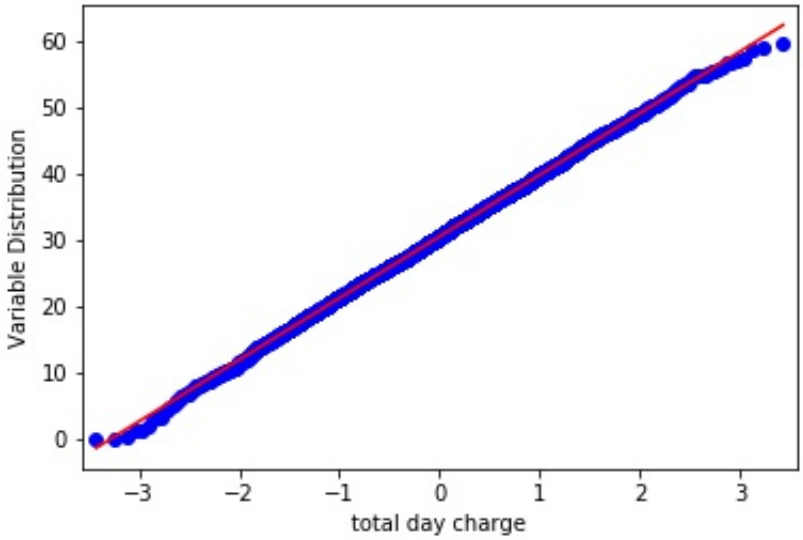
```
    # change fill color
    box.set( facecolor = '#1b9e77' )

for whisker in bp['whiskers']:
    whisker.set(color='#7570b3', linewidth=2)

for cap in bp['caps']:
    cap.set(color='#7570b3', linewidth=2)

for median in bp['medians']:
    median.set(color='#b2df8a', linewidth=2)

for flier in bp['fliers']:
    flier.set(marker='o', color='#e7298a', alpha=0.5)

plt.xlabel(columnNames)

filename = "D:\\edWisor\\Project-I\\Boxplot Figures\\"+columnNames+' boxplot.png'
fig.savefig(filename, bbox_inches='tight')
return filename
```

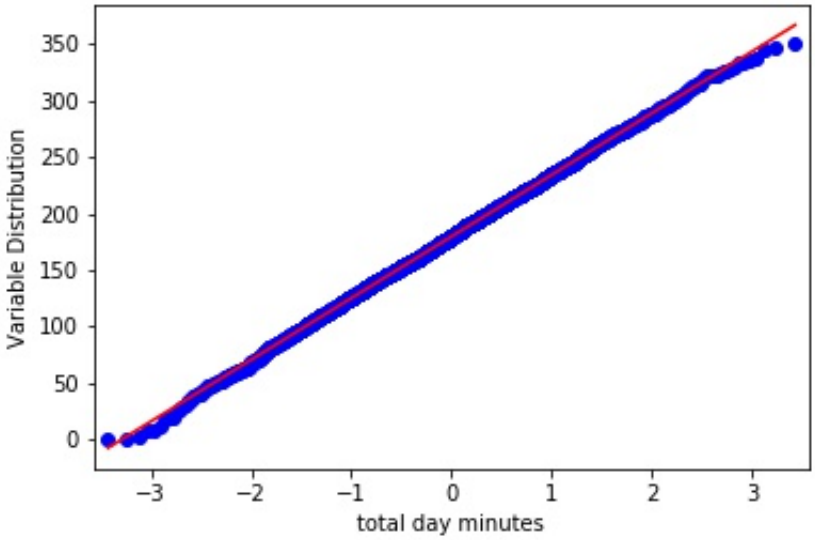
5.2 B) QQ Plots

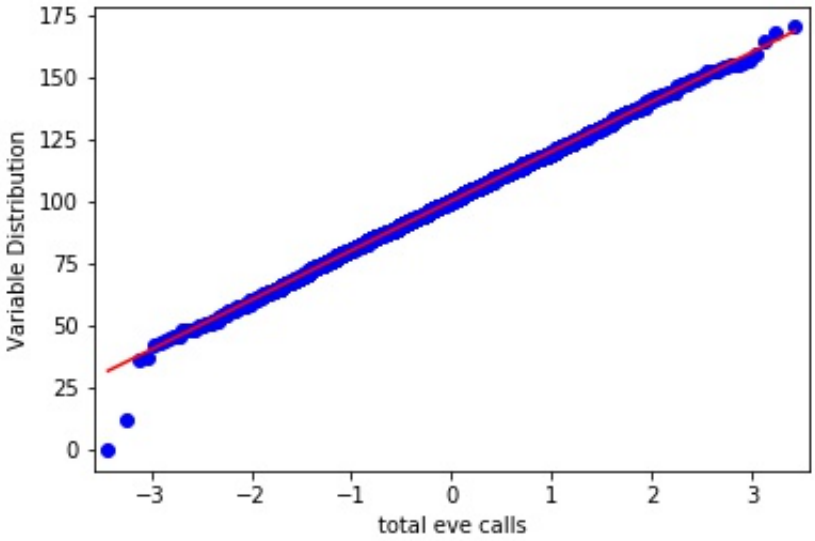
What are QQ plots? The quantile-quantile (q-q) plot is a graphical technique for determining if two data sets come from populations with a common distribution. A 45-degree reference line is plotted. If the two sets come from a population with the same distribution, the points should fall approximately along this reference line. The greater the departure from this reference line, the greater the evidence for the conclusion that the two data sets have come from populations with different distributions.

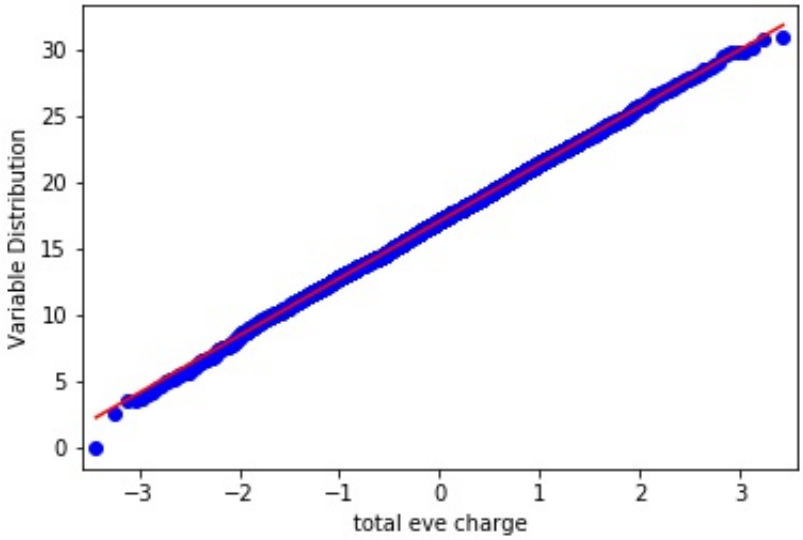
QQ Plots for each numerical data are shown below:

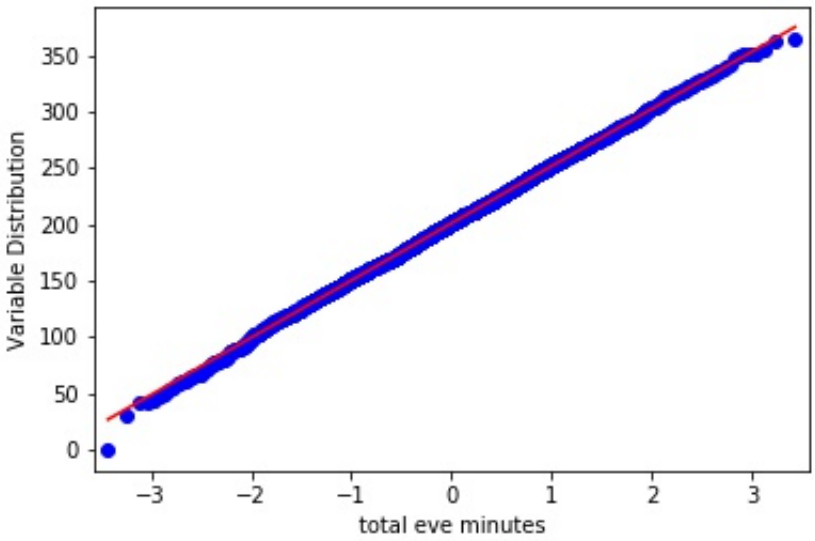


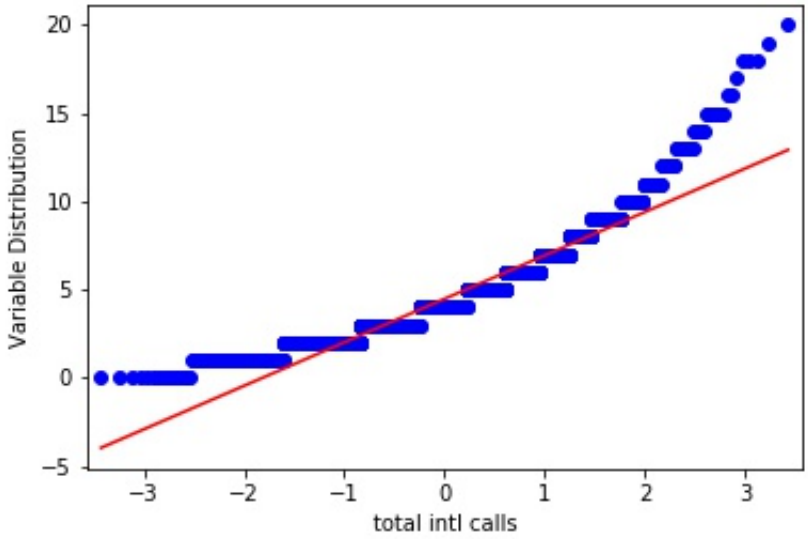

Variable Distribution
total day charge

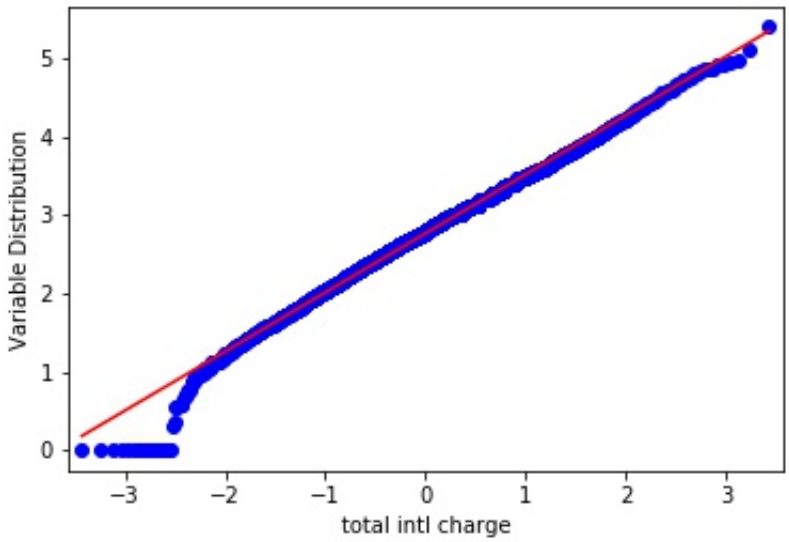

Variable Distribution
total day minutes

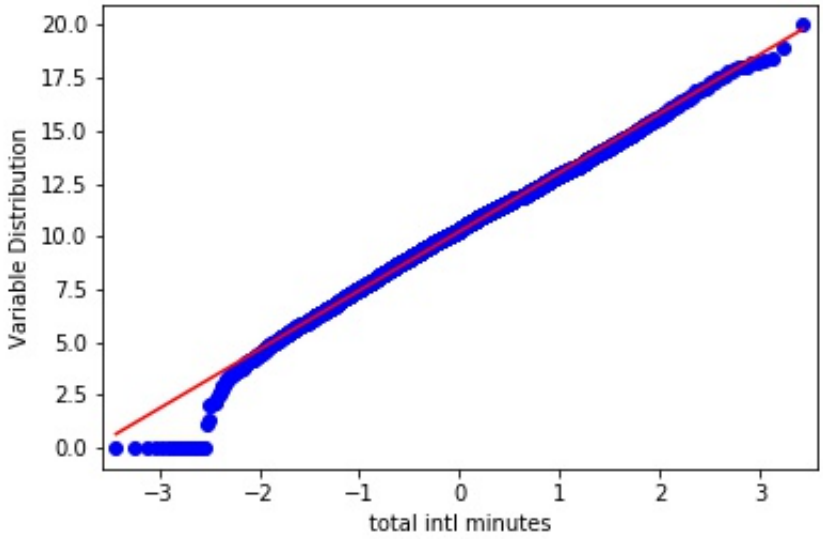

Variable Distribution
total eve calls

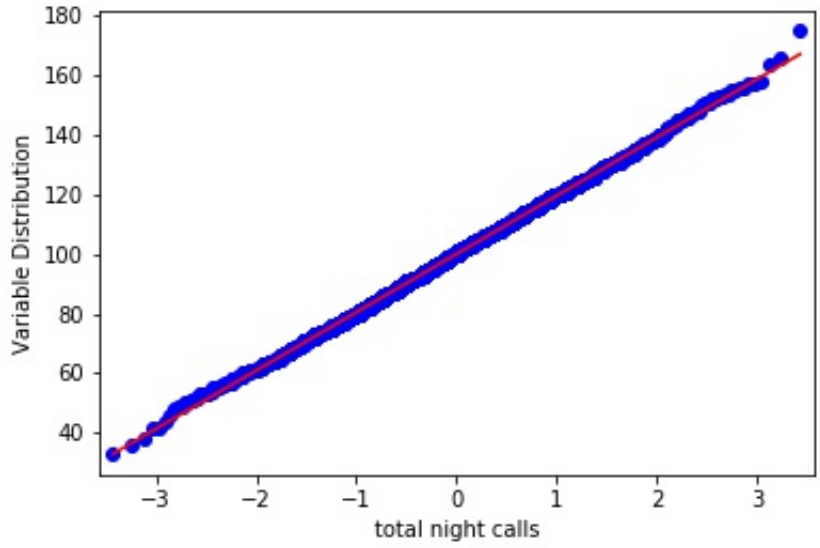

Variable Distribution
total eve charge

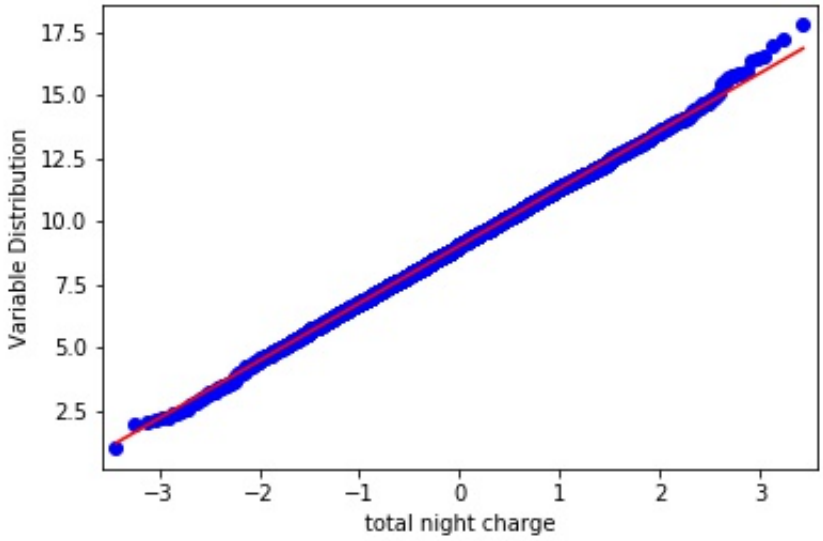

Variable Distribution
total eve minutes

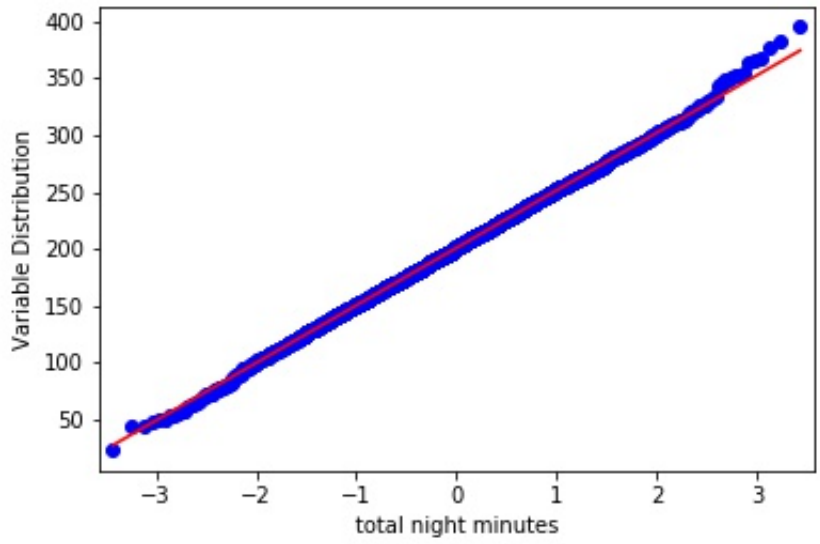

Variable Distribution
total intl calls


Variable Distribution
total intl charge


Variable Distribution
total intl minutes


Variable Distribution
total night calls


Variable Distribution
total night charge

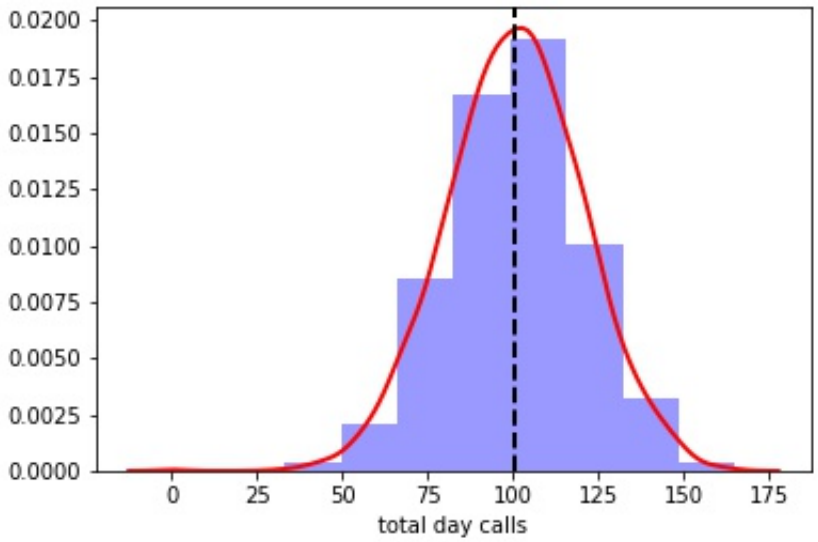


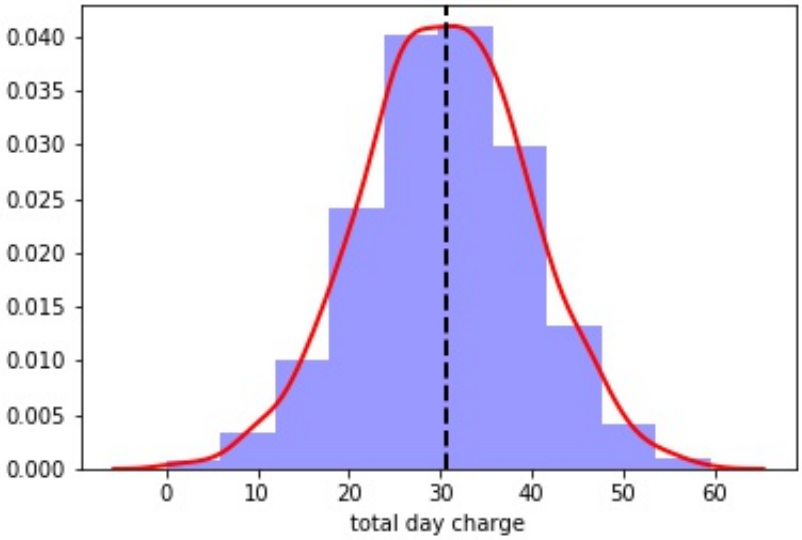
Code used to create QQ plots is as below:

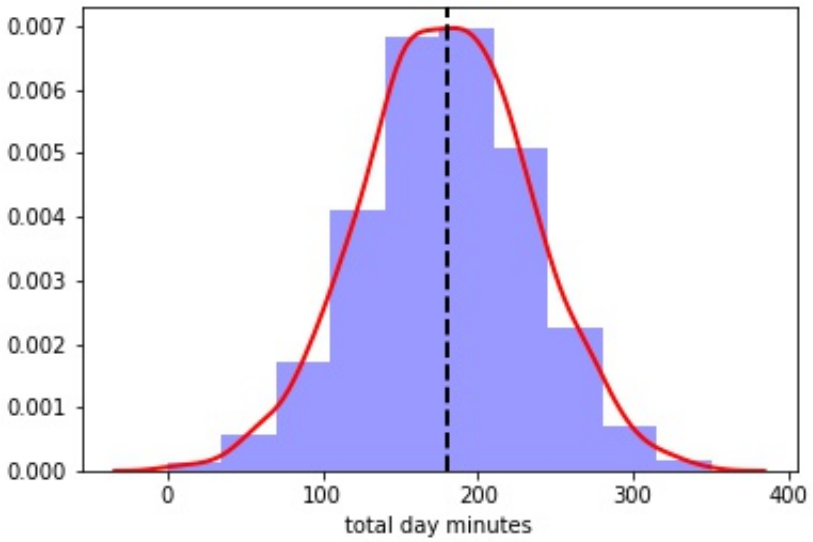
```
from statsmodels.graphics.gofplots import qqplot
from matplotlib import pyplot
def GetQQplot(df,col_name):
    qqplot(df[col_name], line='s')
    pyplot.xlabel(col_name)
    pyplot.ylabel("Variable Distribution")
    filename = "D:\\edWisor\\Project-I\\QQPlot Figures\\"+col_name+' QQplot.jpg'
    pyplot.savefig(filename)
    return filename
```

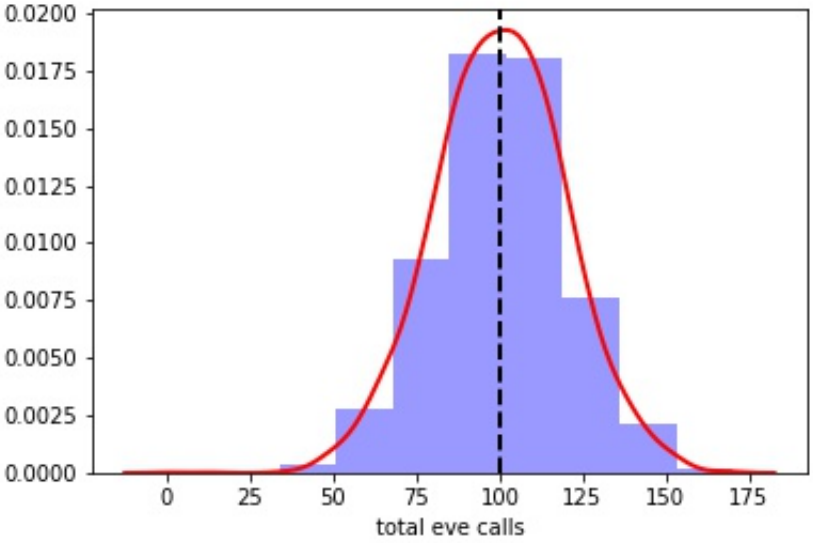
5.3 C) Probability Distribution Curve

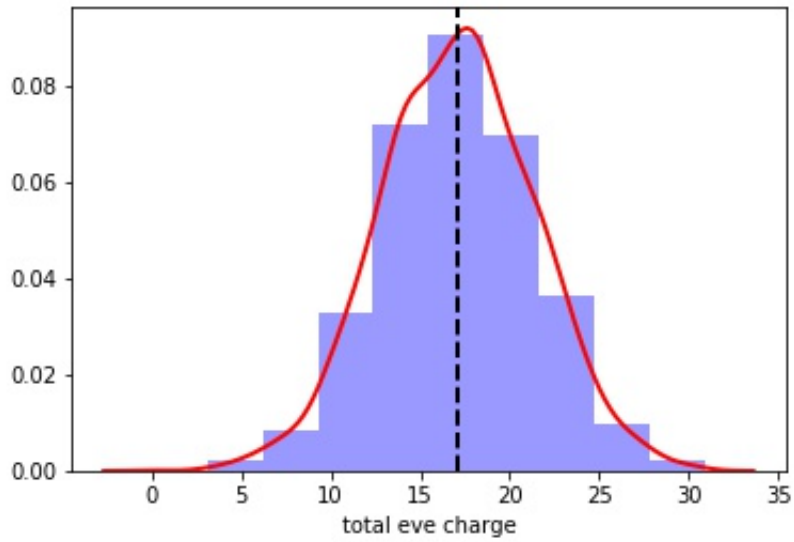
Probability distribution of each column can be seen as below:

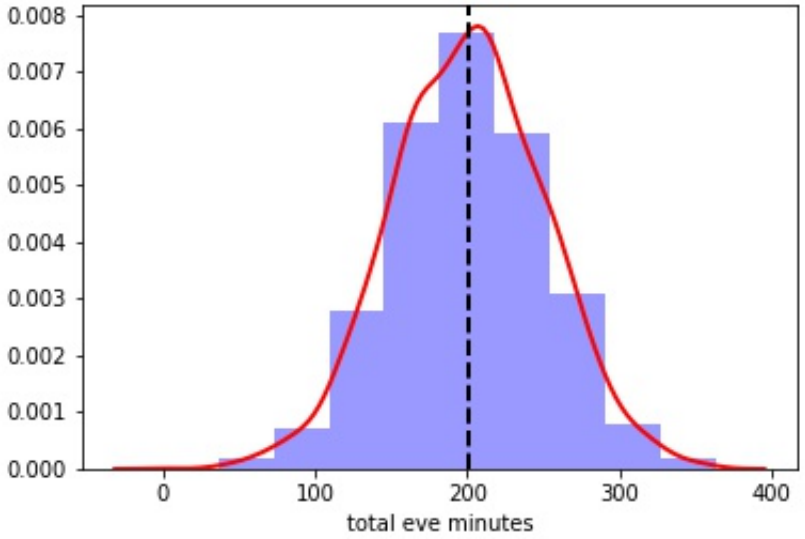

0.0200
0.0175
0.0150
0.0125
0.0100
0.0075
0.0050
0.0025
0.0000
0
25
50
75
100
125
150
175
total day calls

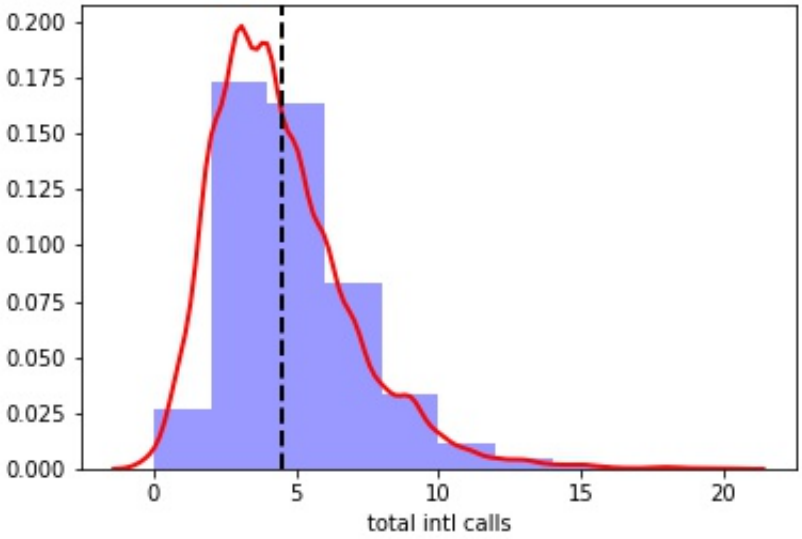

0.040
0.035
0.030
0.025
0.020
0.015
0.010
0.005
0.000
0
10
20
30
40
50
60
total day charge

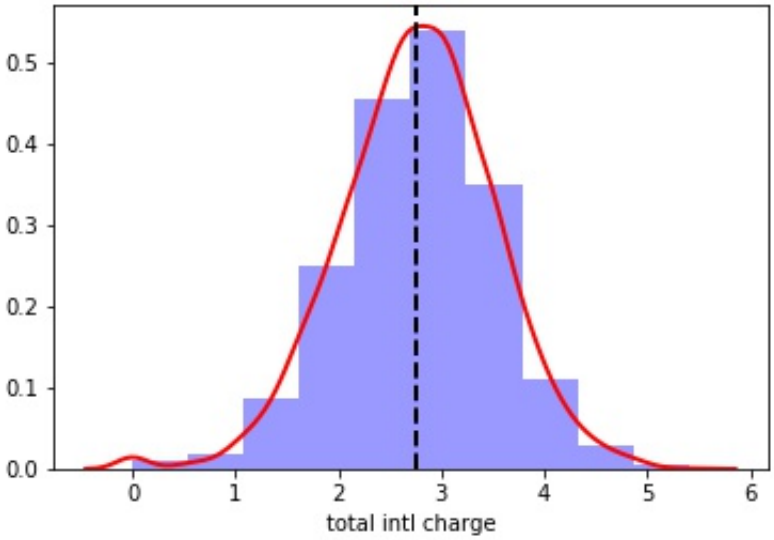

0.007
0.006
0.005
0.004
0.003
0.002
0.001
0.000
0
100
200
300
400
total day minutes

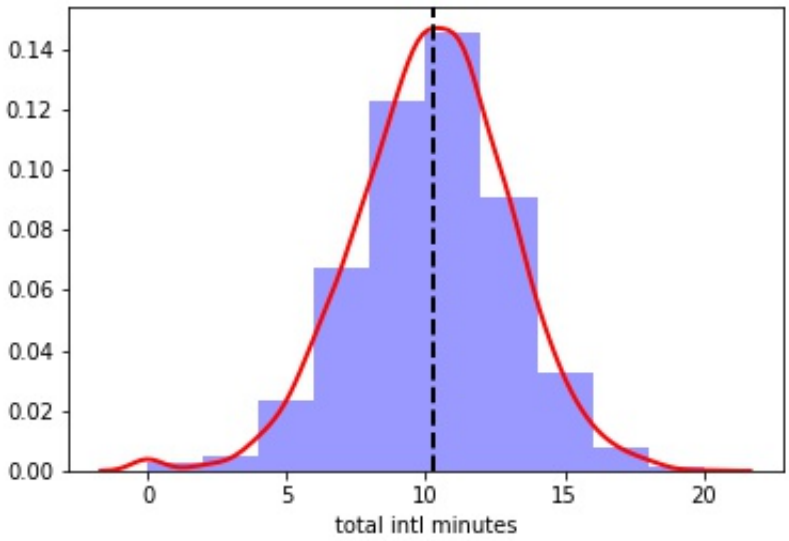

0.0200
0.0175
0.0150
0.0125
0.0100
0.0075
0.0050
0.0025
0.0000
0
25
50
75
100
125
150
175
total eve calls

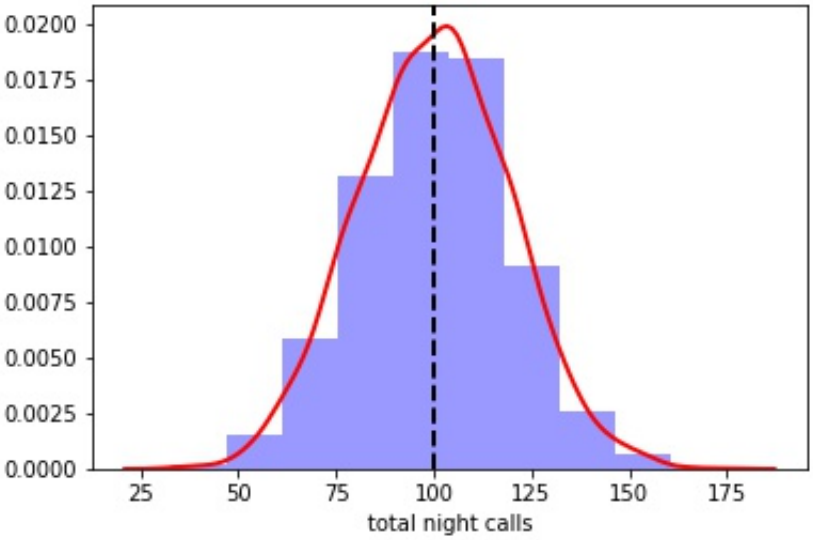

0.08
0.06
0.04
0.02
0.00
0
5
10
15
20
25
30
35
total eve charge

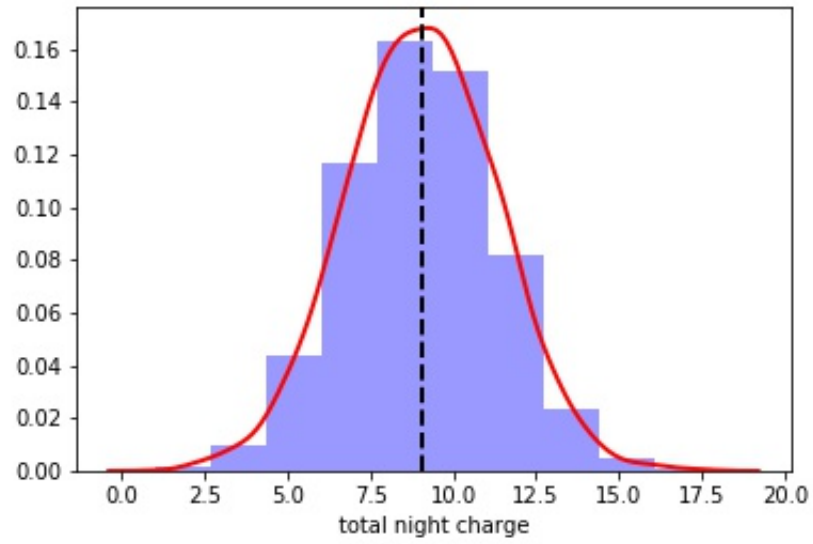

0.008
0.007
0.006
0.005
0.004
0.003
0.002
0.001
0.000
0
100
200
300
400
total eve minutes

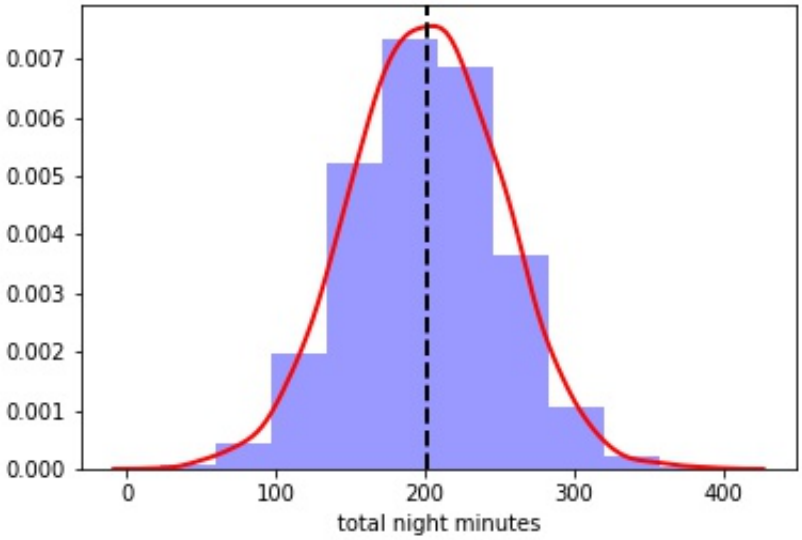

0.200
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
0
5
10
15
20
total intl calls


0.5
0.4
0.3
0.2
0.1
0.0
0
1
2
3
4
5
6
total intl charge


0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
0
5
10
15
20
total intl minutes


0.0200
0.0175
0.0150
0.0125
0.0100
0.0075
0.0050
0.0025
0.0000
25
50
75
100
125
150
175
total night calls





Code used to create Probability Distribution Curve can be seen as below

```
import seaborn as sns
def GetDistributionGraph(df,col_name):

    sns.distplot(df[col_name], hist=True, kde=True,color="black" ,
                 bins=10, hist_kws={"color": "b"},
                 kde_kws={'linewidth': 2,"color": "r"})
    meanvalue=round(df[col_name].mean(),2)

    plt.axvline(meanvalue, color='k', linestyle='dashed', linewidth=2)
    filename = "D:\\edWisor\\Project-I\\Distribution Figures\\"+col_name+" distribution.jpg"
    plt.savefig(filename)
    return filename
```