Churn Data Analysis

Apurv Kumar

22/08/2018

Chapter 1

Introduction

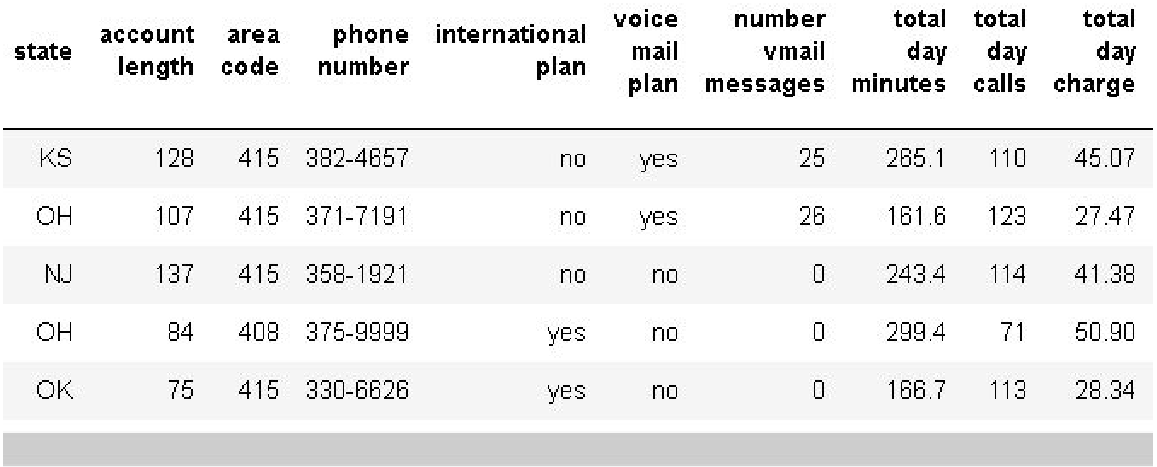
1.1 Problem Statement

Churn rate refers to the percentage of employees that leave the company within a specified time period. Some staff churning is inevitable, but a high rate of churn is costly.
Recruitment and training costs money, and it would be some time before the company would start getting return on investment on the new hirees.
Therefore if a company has a high churn rate, it needs to acquire the reason for the same, so that it could change its policies or rules to better accumulate its employees, and ultimately bring down the churn rate.

In this project, I prepare a model to determine if an employee will leave the company based on the historical date, and come up with an efficient model for the same.

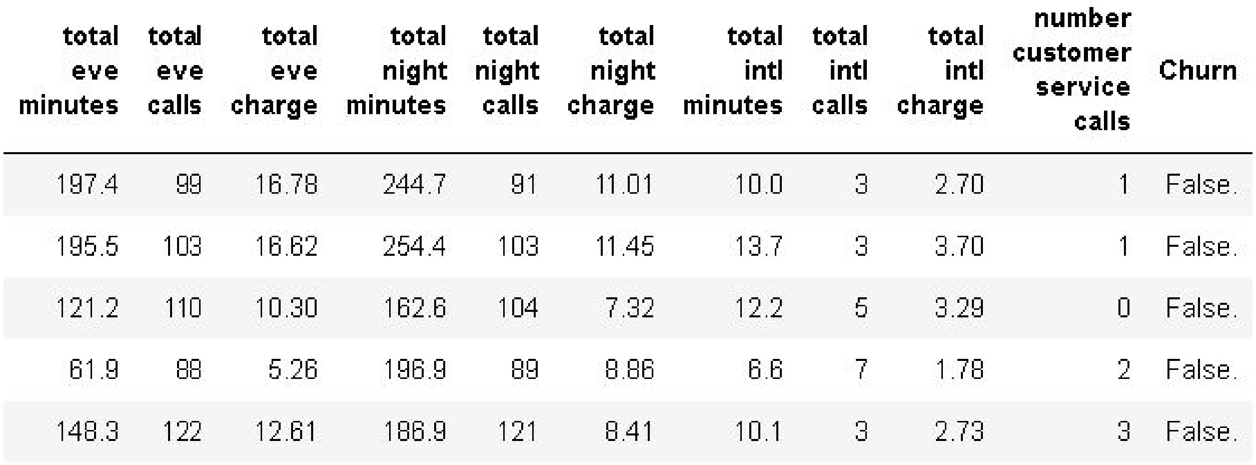
1.2 Data

Our task is to build a classification model to predict the decision that an employee takes : to churn or to not churn.
Given below is a sample of the data that we are using to create our classification model:



state	account length	area code	phone number	international plan	voice mail plan	number vmail messages	total day minutes	total day calls	total day charge
KS	128	415	382-4657	no	yes	25	265.1	110	45.07
OH	107	415	371-7191	no	yes	26	161.6	123	27.47
NJ	137	415	358-1921	no	no	0	243.4	114	41.38
OH	84	408	375-9999	yes	no	0	299.4	71	50.90
OK	75	415	330-6626	yes	no	0	166.7	113	28.34

Table 1.1: Churn Data (Columns: 1-10)



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
197.4	99	16.78	244.7	91	11.01	10.0	3	2.70	1	False.
195.5	103	16.62	254.4	103	11.45	13.7	3	3.70	1	False.
121.2	110	10.30	162.6	104	7.32	12.2	5	3.29	0	False.
61.9	88	5.26	196.9	89	8.86	6.6	7	1.78	2	False.
148.3	122	12.61	186.9	121	8.41	10.1	3	2.73	3	False.

Table 1.2: Absenteeism Data (Columns: 11-21)

As you can see in the table below, we have 20 predictor variables, using which we have to make our analysis:

S. No	Predictor
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

Table 1.3: Predictor variables

Chapter 2

Methodology

2.1 Pre Processing

Pre processing is an essential part of any data analysis process. We explore the data, plot graphs to see its distribution and perform data cleaning and pruning in order to convert the input dataset into a proper format to be able to correctly do our analysis.

A dataset can have various issues like missing values, exceptional data, incorrect columns name, invalid characters, etc. Therefore it is essential that we perform the step of data pre processing. Let us start:

2.1.1 Missing Value Analysis

A dataset in real world would always have some values missing for some of the cells. It is our role to be able to efficiently handle it. A model trained before handling the missing values would yield incorrect predictions.

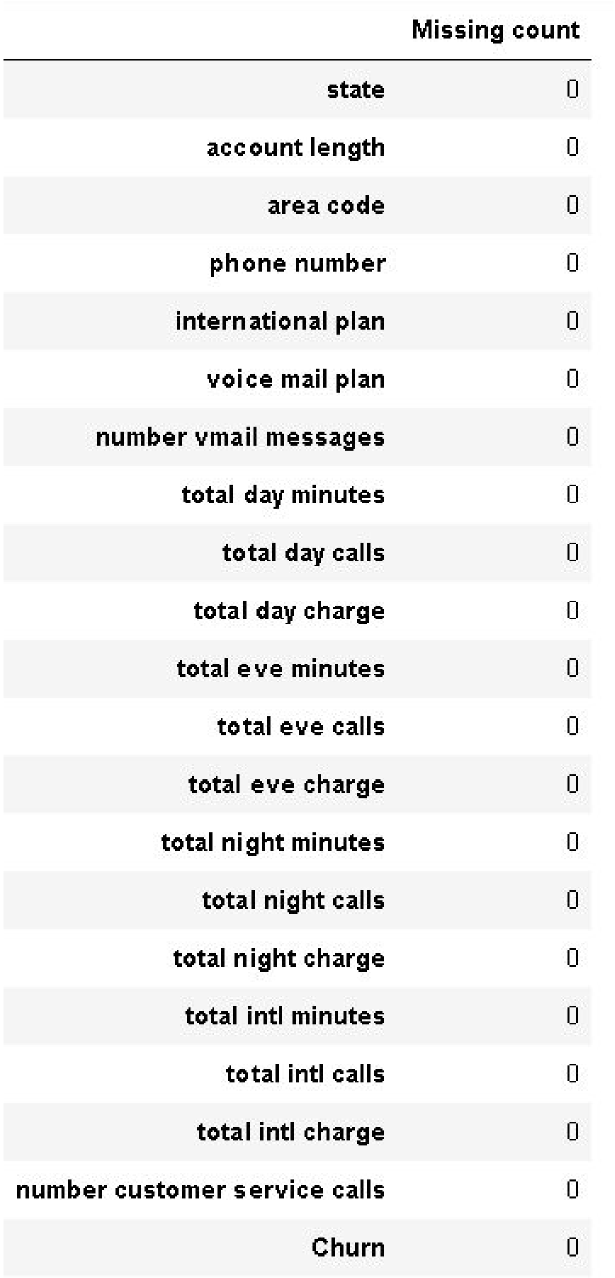
In our analysis I look into 3 ways to handle the missing values:

1. Imputation by mean: We replace every missing value by the mean of the corresponding column.
2. Imputation by median: We replace every missing value by the median of the corresponding column.
3. Imputation by KNN: We use the K Nearest Neighbour algorithm to impute the missing values.
4.

For each of the three ways we perform the following steps to select the best among the three:

1. We replace one non-missing numerical value with NaN.
2. We perform the imputation method (mean, median or KNN).
3. Through step 2, we decide upon which methods imputed the NaN value with a value which is closest to the original value, and then use that method as default.

Let us take a look at our dataset to determine if there are any missing values:



	Missing 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
total intl minutes	0
total intl calls	0
total intl charge	0
number customer service calls	0
Churn	0

Table 2.1: Missing value count in the Churn dataset

An interesting visualization that I came across to visualize missing values in a dataset is to utilize the “missingno” library. Find below the results of the visualization:

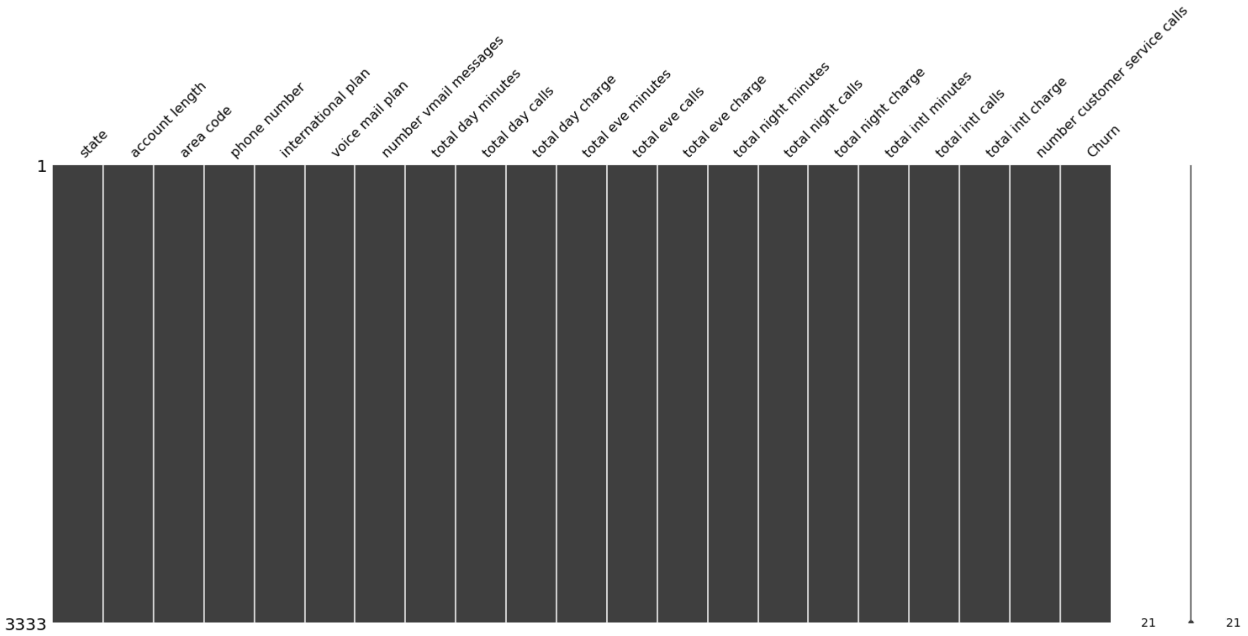


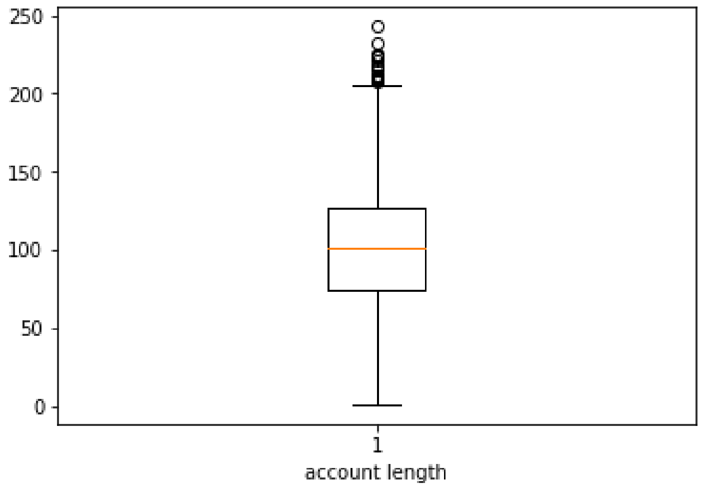
Fig 2.1: Missing value visualization: None in this dataset!!

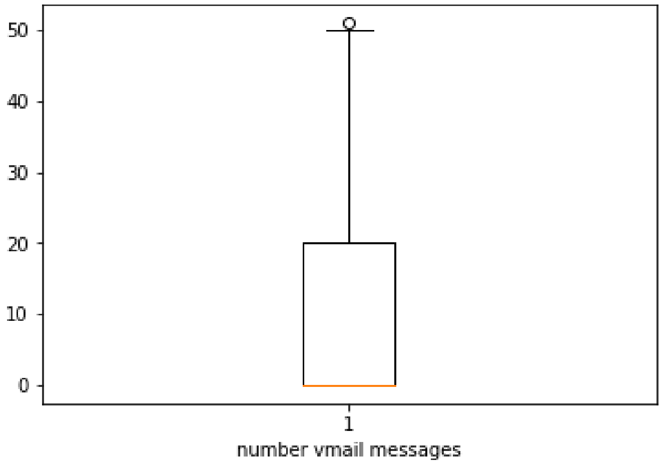
Lucky for us that our dataset does not have any missing values. This case will not usually happen in real world, we would encounter a dataset which would have numerous missing values, and would need to employ imputation methods.

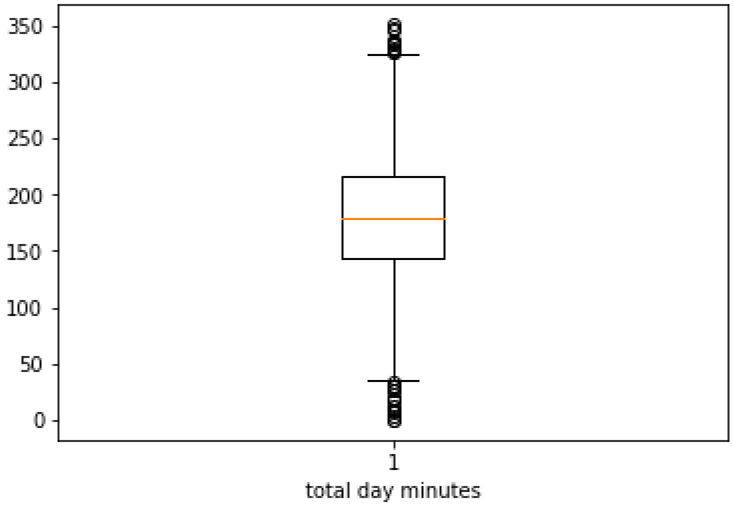
2.1.2 Outlier Analysis

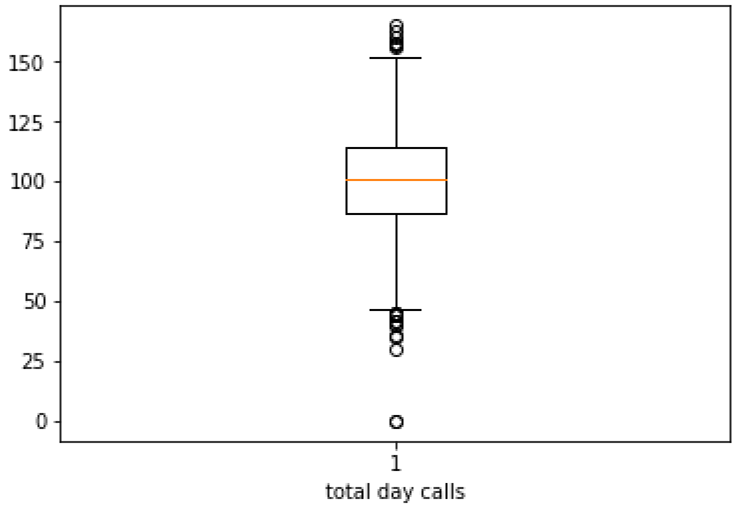
Let us move to the outlier analysis.
Outlier refers to data points which have very extreme values and do not usually denote the general population. Failing to handle the outliers will result in our model producing incorrect and biased predictions.

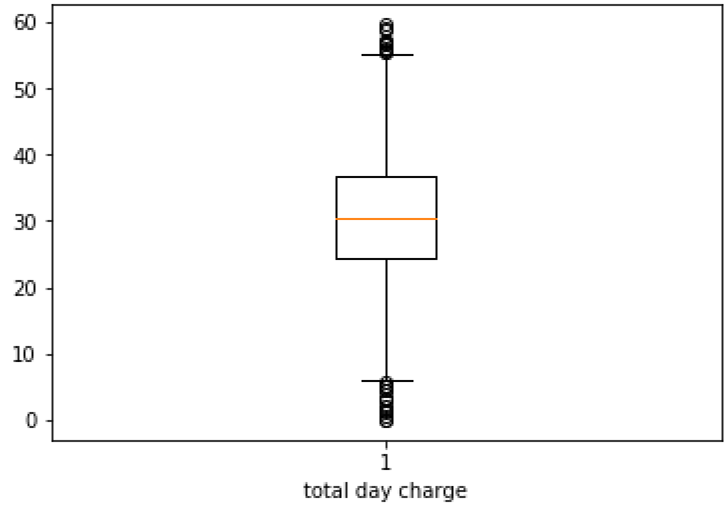
One of the efficient ways to visualise the presence of outliers in any dataset is by using the Boxplot method to plot the outliers. Boxplot works for numerical data points, not categorical. Shown below is the plot for each of the numerical columns:

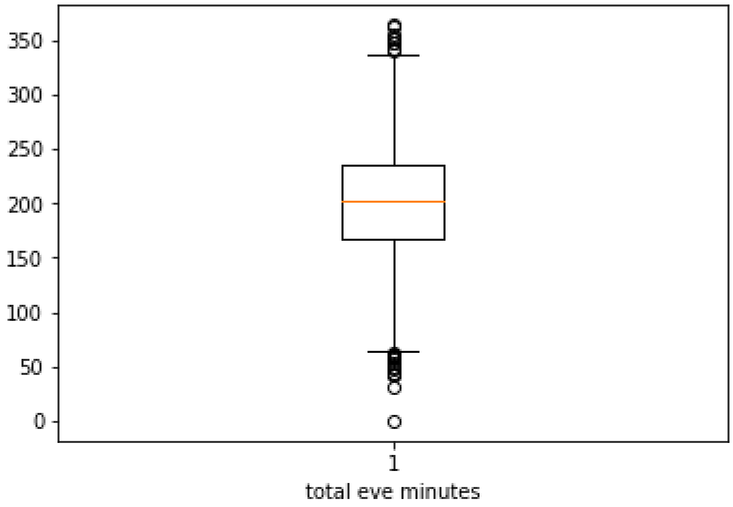

250
200
150
100
50
0
1
account length

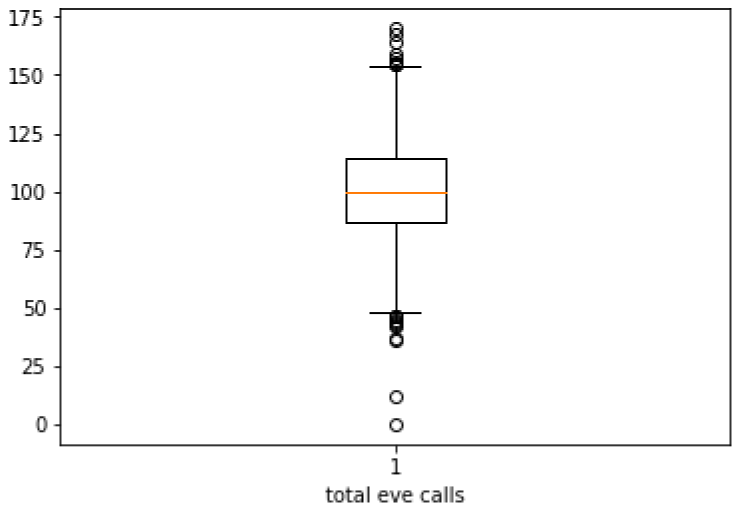

50
40
30
20
10
0
1
number vmail messages

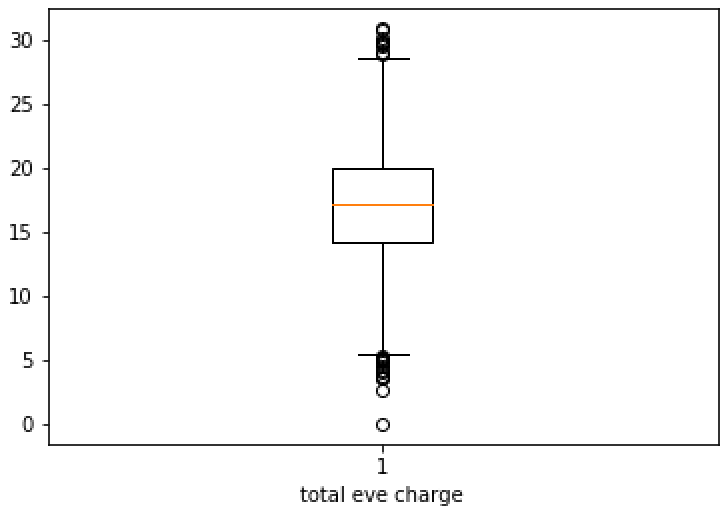

350
300
250
200
150
100
50
0
1
total day minutes

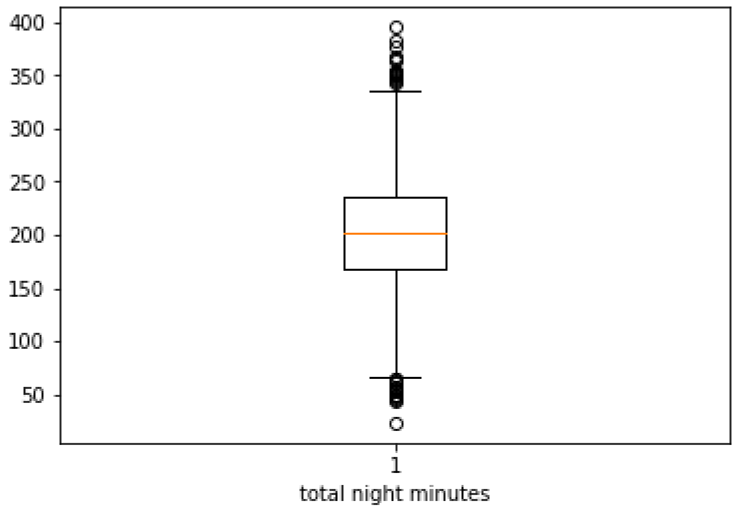

150
125
100
75
50
25
0
1
total day calls

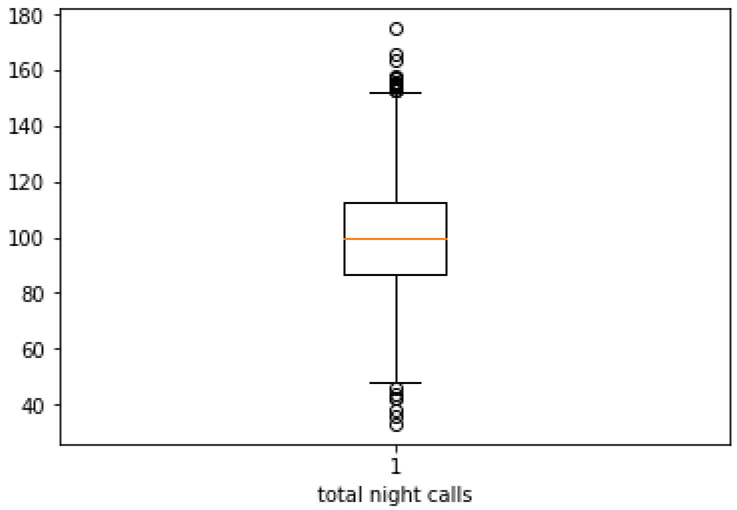

60
50
40
30
20
10
0
1
total day charge

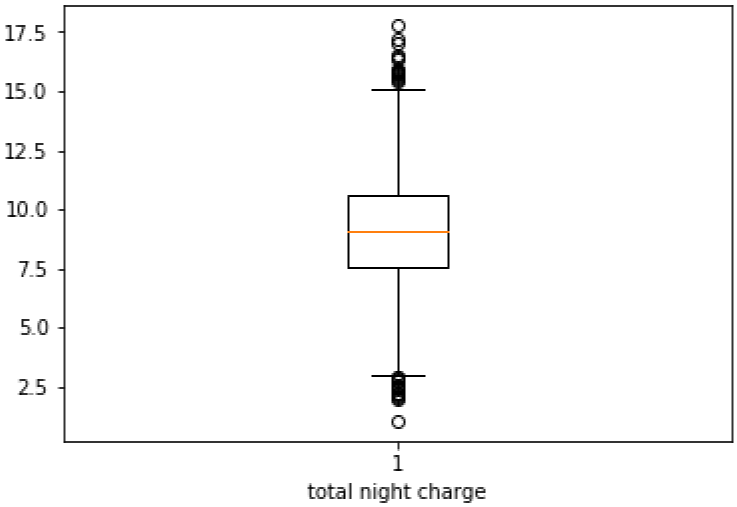

350
300
250
200
150
100
50
0
1
total eve minutes

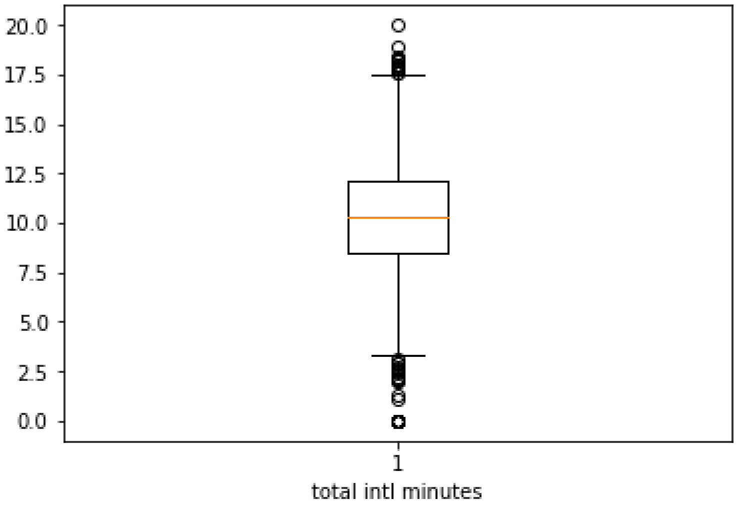

175
150
125
100
75
50
25
0
1
total eve calls

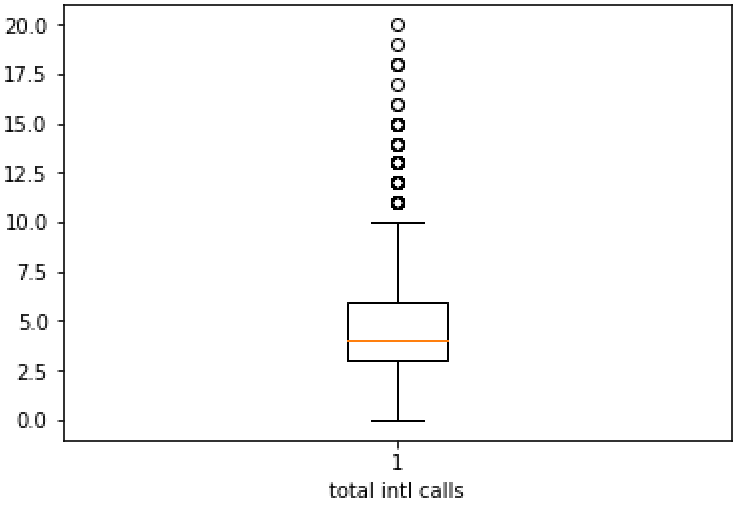

30
25
20
15
10
5
0
1
total eve charge

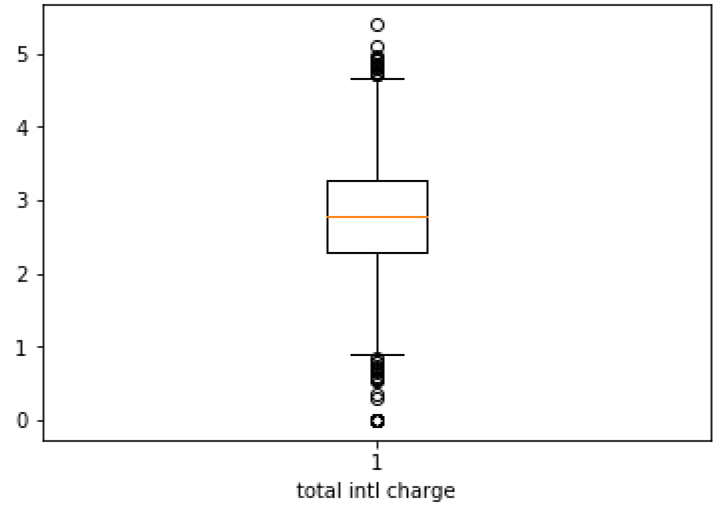

400
350
300
250
200
150
100
50
1
total night minutes

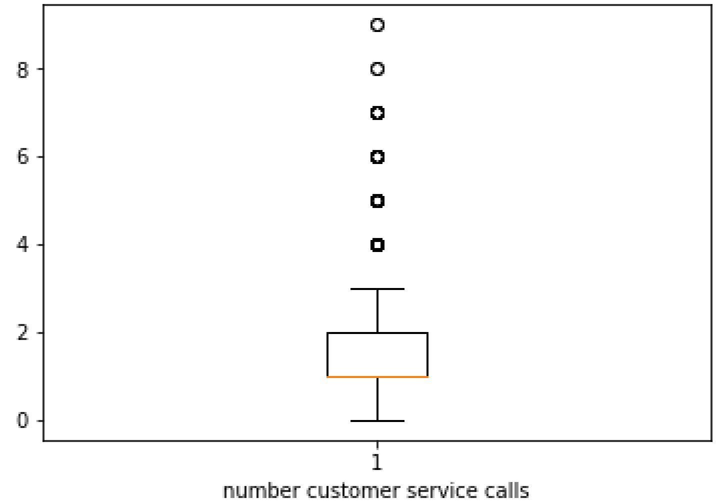

180
160
140
120
100
80
60
40
1
total night calls


17.5
15.0
12.5
10.0
7.5
5.0
2.5
1
total night charge


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


8
6
4
2
0
1
number customer service calls

Looking at the above plots, we can efficiently visualise that some of the predictors have outlier points present in them. We drop such points so that our dataset now reflects a generalized dataset, which would prove efficient in training our model.

2.1.3 Feature Selection

Our dataset has 20 predictors, and it is possible that not all of the predictors are necessary for our classification model. Training a model with 20 models might make the model a little bit complex, and could make the model overfit. Therefore we shall make some analysis, and choose only those features which are required.

The parameters to choose the importance of a feature is to calculate the correlation among all the numerical predictors. Suppose A and B have high correlation among them, then it means that they are producing the same information, therefore we can discard one of them.

One of the convenient ways to visualize correlation is to plot a heatmap, which makes it easy to see which predictors are depended on each other, and therefore we can remove the redundant predictors not required for our model. Find below the heatmap:

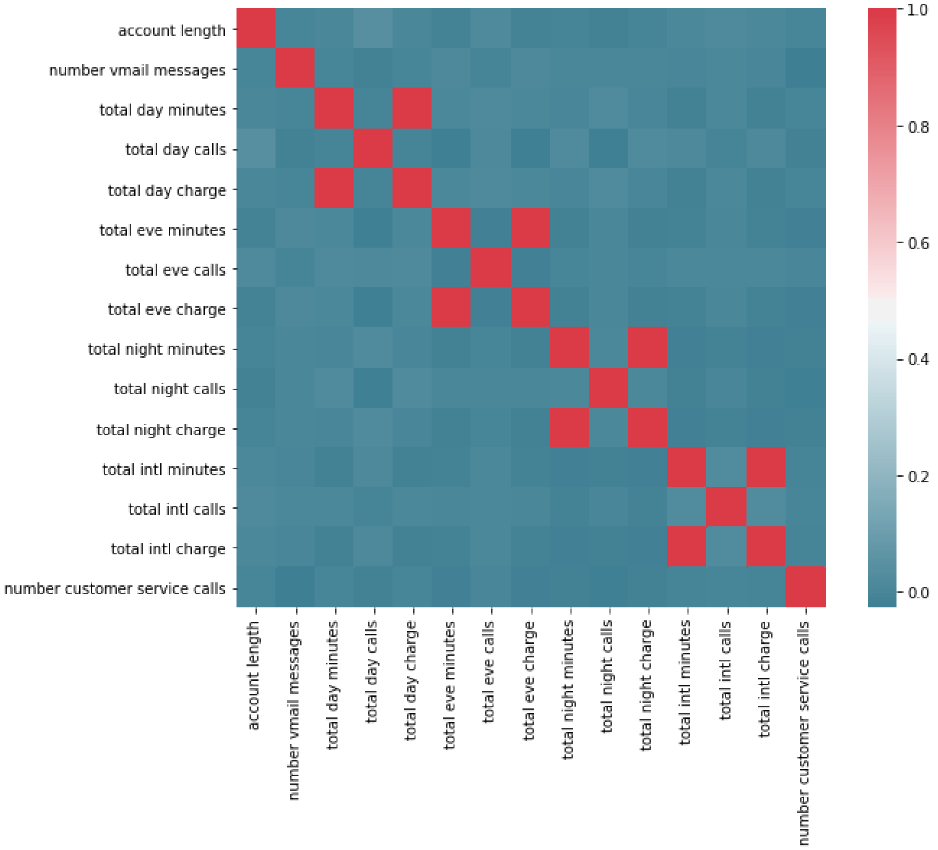


Table 2.3: Heatmap to visualize correlation among numerical predictors

From the above heatmap, we can clearly visualize that predictors like (total day charge, total eve charge, total night charge, total intl charge) have high correlation with other predictors, therefore we remove them from our selection of predictors as they contribute redundant information.

Just like heatmap works for the numerical predictors, we use a **chi-square test** for categorical problems to find out the importance of the categorical predictors with respect to the target variable.

Predictors	**Chi square test value**
state	0.005772462240810621
area code	0.8391405208128838
international plan	3.453377234466883e-52

voice mail plan	1.1790993481239186e-09

Table 2.4: P value between the categorical and the target variable

We are looking at finding out the statistically significant variables.

We have our null hypothesis as follows:
H0: *There is no relationship between the categorical variable and the decision to churn by the employee.*
The alternate hypothesis,
H1: *There is a relationship between the categorical variable and the decision to churn by the employee.*

So if we get a p-value less than 0.05 (chosen significance level), we can reject the null hypothesis, and accept the alternate hypothesis. Otherwise we accept the null hypothesis, which would mean that there is no significance between the two variables.

Based on the above observations we drop the following observations: area code.

From the feature selection step, we end up with a concise list of features which might play an important role in the classification model. Let us move to model development.

2.2 Modelling

2.2.1 Model Selection

There are broadly 2 types of models: Regression and classification.

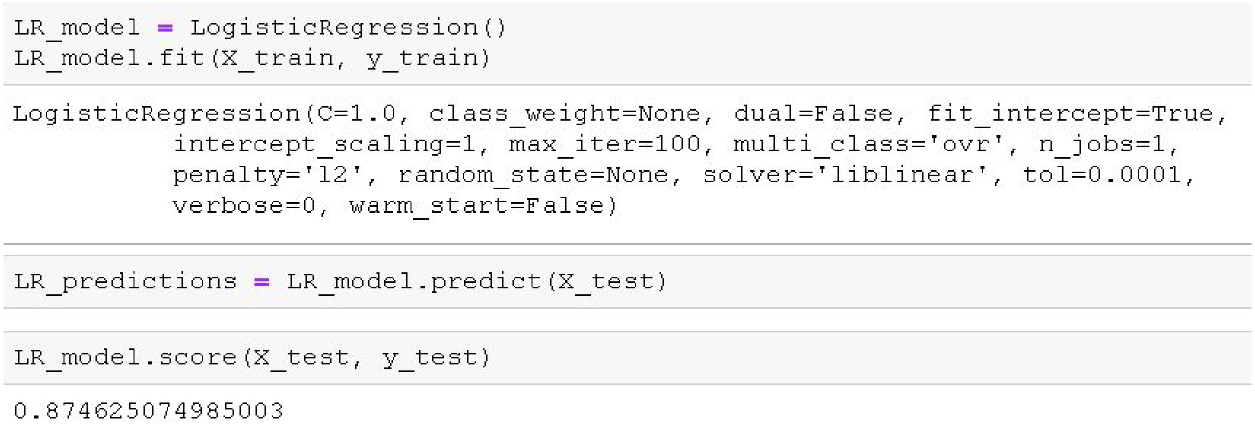
1. If the target variable is nominal (aka categorical), then we perform a classification.
2. If the target variable can be any number, then it is a regression problem.

In our case, the target variable is “Churn” is a boolean, which could take the value: True or False and denotes a category, therefore we shall be developing a classification model.

The datasets are provided as train_data and test_data. We shall utilize the train_data to train out model, and test_data to test its accuracy and validation.
Before starting with our model development, we would convert our boolean category variables to binary (0 denoting False, and 1 denoting True).

We always start with making a simple model first, and them moving on something more complex. Therefore let us start with the first model, a Logistic Regression.

2.2.2 Logistic Regressor



```
LR_model = LogisticRegression()
LR_model.fit(X_train, y_train)
```

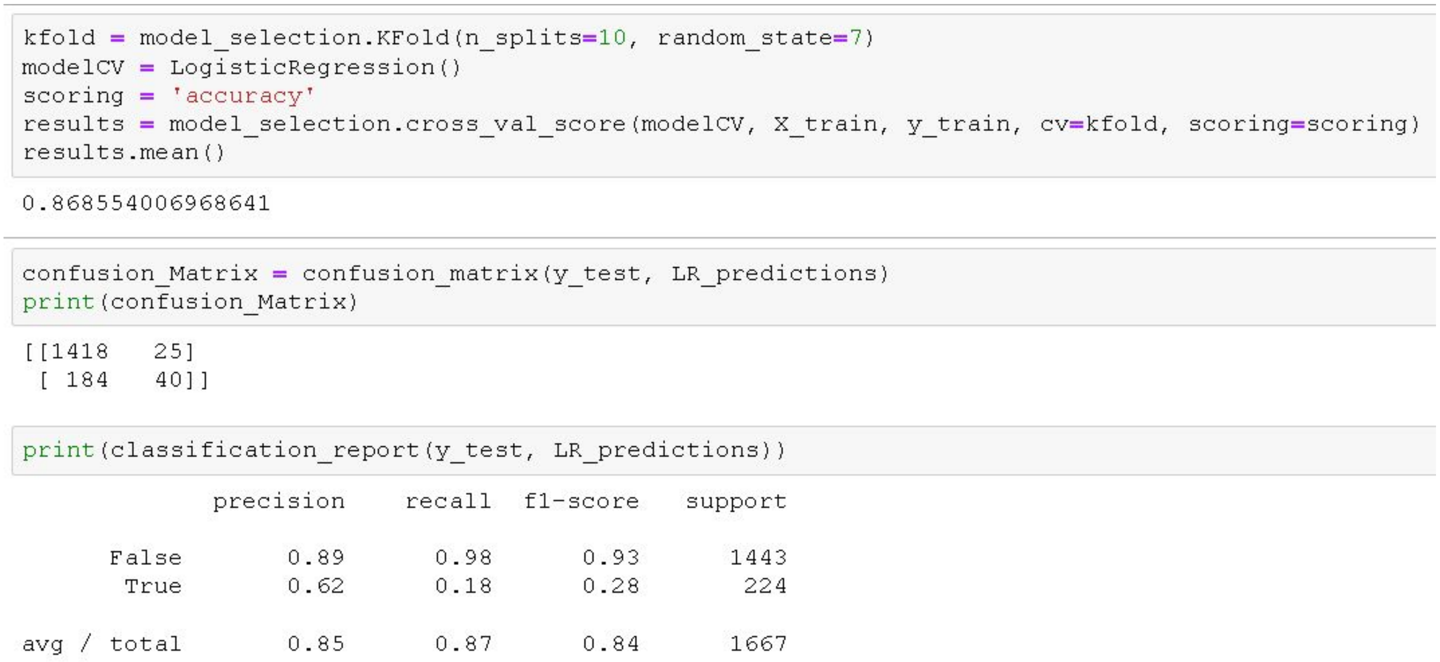
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
          penalty='l2', random_state=None, solver='liblinear', tol=0.0001,
          verbose=0, warm_start=False)
```

```
LR_predictions = LR_model.predict(X_test)
```

```
LR_model.score(X_test, y_test)
```

```
0.874625074985003
```

One of the efficient ways to check if the model built is performing well is to check the accuracy.
For the above Logistic Regression the accuracy is 87.46 %.



```
kfold = model_selection.KFold(n_splits=10, random_state=7)
modelCV = LogisticRegression()
scoring = 'accuracy'
results = model_selection.cross_val_score(modelCV, X_train, y_train, cv=kfold, scoring=scoring)
results.mean()
```

```
0.868554006968641
```

```
confusion_Matrix = confusion_matrix(y_test, LR_predictions)
print(confusion_Matrix)
```

```
[[1418   25]
 [ 184   40]]
```

```
print(classification_report(y_test, LR_predictions))
```

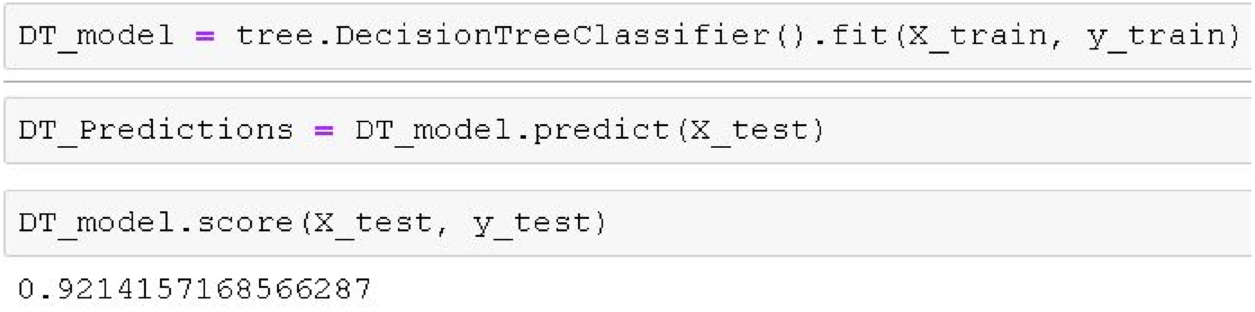
```
             precision    recall  f1-score   support

      False       0.89      0.98      0.93      1443
       True       0.62      0.18      0.28       224

avg / total       0.85      0.87      0.84      1667
```

Another important parameter to look for is the validation accuracy. In our analysis, we utilize k-fold validation. The above model has a validation accuracy of 86.85 %.
The validation accuracy is almost same as accuracy, which tells us that the model is trained properly.

An additional information is to plot the Confusion matrix, and finding attributes like precision, recall and f1-score.

Now we can always try for a complex model to check if we can get a better model accuracy and validation accuracy.. Let us take the case of a Decision Tree Classifier.

2.2.3 Decision Tree Classifier



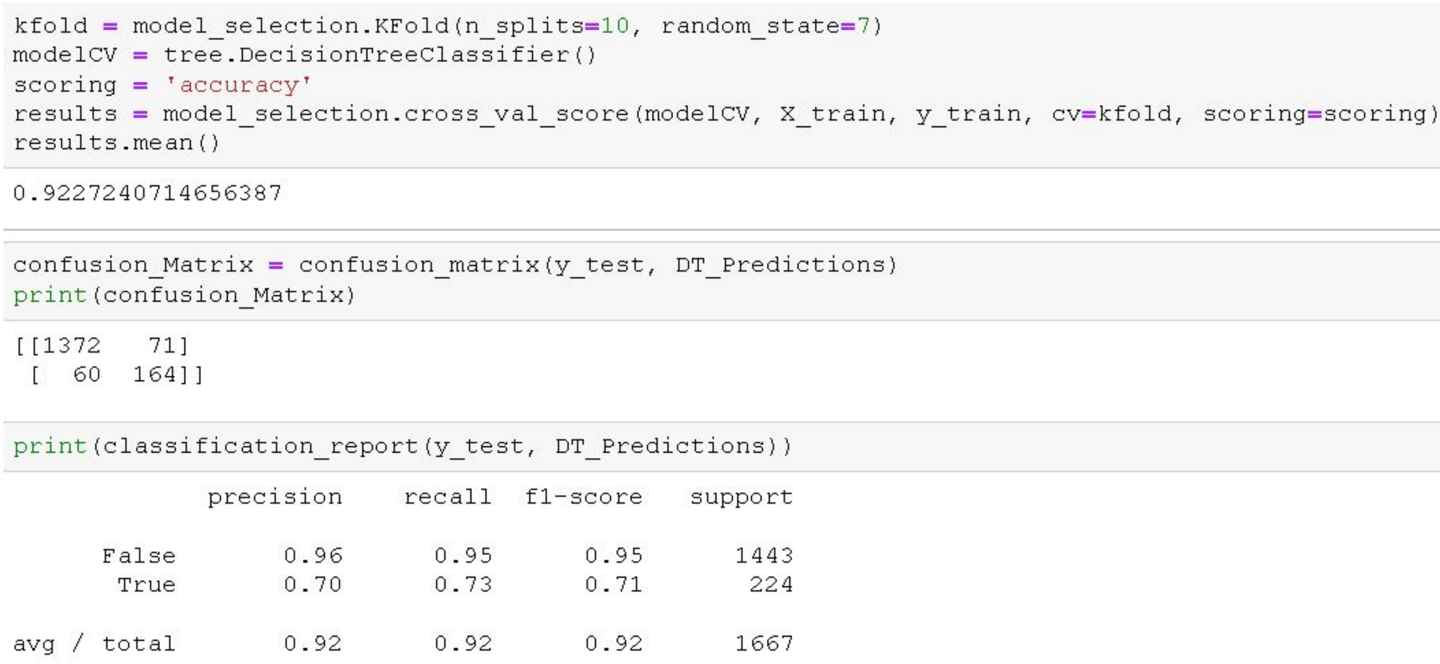
```
DT_model = tree.DecisionTreeClassifier().fit(X_train, y_train)
```

```
DT_Predictions = DT_model.predict(X_test)
```

```
DT_model.score(X_test, y_test)
```

```
0.9214157168566287
```

The model accuracy in case of the Decision Tree is 92.14 % which is more than Logistic Regression model.



```
kfold = model_selection.KFold(n_splits=10, random_state=7)
modelCV = tree.DecisionTreeClassifier()
scoring = 'accuracy'
results = model_selection.cross_val_score(modelCV, X_train, y_train, cv=kfold, scoring=scoring)
results.mean()
```

```
0.9227240714656387
```

```
confusion_Matrix = confusion_matrix(y_test, DT_Predictions)
print(confusion_Matrix)
```

```
[[1372   71]
 [  60  164]]
```

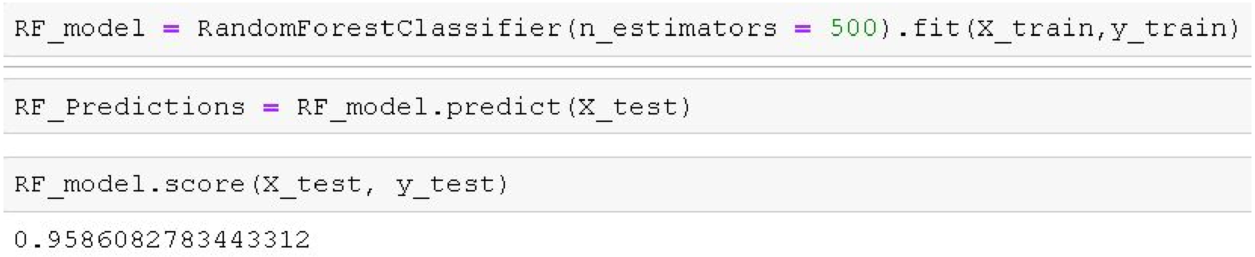
```
print(classification_report(y_test, DT_Predictions))
```

```
             precision    recall  f1-score   support

      False       0.96      0.95      0.95      1443
       True       0.70      0.73      0.71       224

avg / total       0.92      0.92      0.92      1667
```

The validation accuracy is 92.27 %, which is a good news. We got a better accuracy and a better validation accuracy, stating us that a Decision Tree is better at classifying for our dataset.

Let us go a step further. Let us try to implement a Random Forest Classifier. A Random forest is basically a collection of decision forests, and it averages out the results, therefore in theory it should produce better results. Let us check it out.

2.2.4 Random Forest Classifier



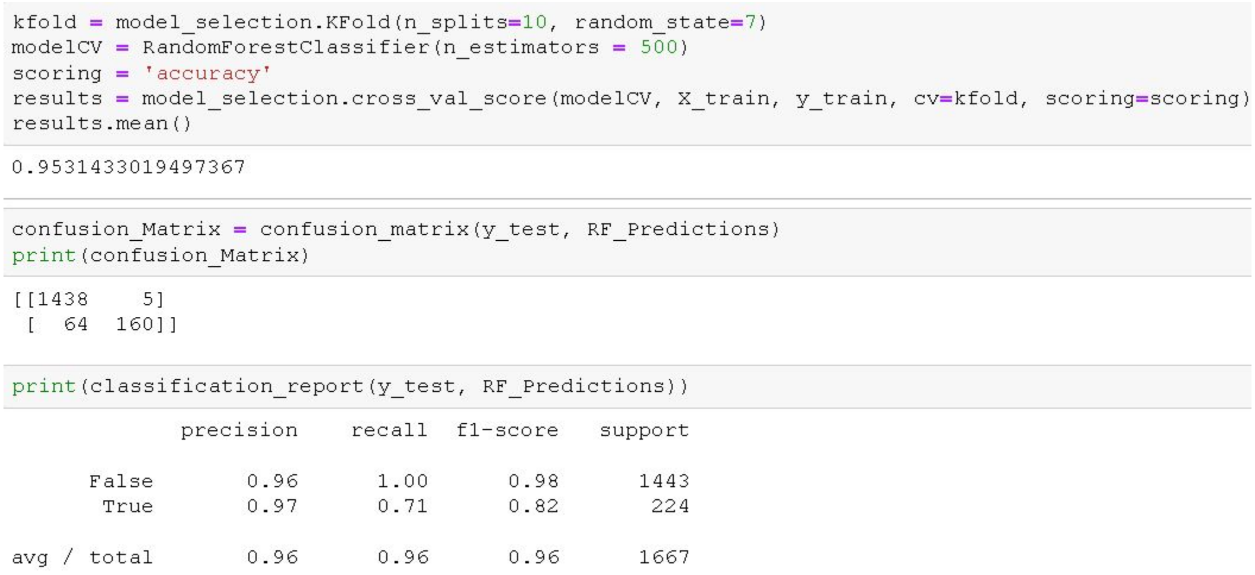
```
RF_model = RandomForestClassifier(n_estimators = 500).fit(X_train,y_train)
```

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

```
RF_model.score(X_test, y_test)
```

```
0.9586082783443312
```

The model accuracy in case of the Random Forest Classifier is 95.86 % which is more than the Decision Tree model.



```
kfold = model_selection.KFold(n_splits=10, random_state=7)
modelCV = RandomForestClassifier(n_estimators = 500)
scoring = 'accuracy'
results = model_selection.cross_val_score(modelCV, X_train, y_train, cv=kfold, scoring=scoring)
results.mean()
```

```
0.9531433019497367
```

```
confusion_Matrix = confusion_matrix(y_test, RF_Predictions)
print(confusion_Matrix)
```

```
[[1438    5]
 [  64  160]]
```

```
print(classification_report(y_test, RF_Predictions))
```

```
             precision    recall  f1-score   support

      False       0.96      1.00      0.98      1443
       True       0.97      0.71      0.82       224

avg / total       0.96      0.96      0.96      1667
```

The validation accuracy is 95.31 %, which is an amazing news. We got a better accuracy and a better validation accuracy, giving us a better model than that of a Decision Tree at classifying our dataset.

Chapter 3

Analysis and Conclusion

3.1 Analysis

We have got the Random Forest as being the most accurate, and also validated in predicting the decision of an employee to churn.

Using the **feature_importances_** parameter, we find out that the features with the most weights in our model is (total day minutes, total evening minutes).
My analysis on the above result is as follows:

1. If an employee is looking at leaving the company, he/she would definitely be looking to switch jobs, therefore he/she would be in talks with the HR of another company, or be giving telephonic interviews during free time.
2. Professional conversations usually happen during morning and evening hours, that is the reason that (total night minutes) are not significant, as people usually do not want to be disturbed at night.

3.2 Summary

We have analysed our dataset: Churn.

1. We started off with the pre processing steps: Missing value analysis, Outlier analysis, Feature selection.
2. Our dataset represents a classification problem.
3. We prepared models using Logistic regression, Decision Tree Classifier and Random Forest Classifier.
4. We find out the model accuracy, the validation accuracy and show some extra information like precision, recall and f1-score

3.3 Conclusion

There was a difference in the accuracy and validation accuracy for all the 3 models, with Random Forests performing the best. Therefore we suggest to use the Random Forest Classifier model..

3.4 Limitations

It is important for us to state the limitations of our work, and also suggest some improvements.

1. In real world scenario, we would usually have unformatted data, and the process of pre processing takes up a huge chunk of the entire cycle.
2. For a better predictions, it is important that the dataset collected comes from all the branches of the office, not just one. If it is collected from a single source, it will lead to biased predictions.
3. There are a much more complex, and possibly accurate ways to handle the above scenario, and usually a lot of tuning of hyperparameters are required to arrive at an efficient model.

THANK YOU.