**PROJECT:**

**Churn Reduction**

**-Abhishek.R**

**INDEX**

**1)Problem Statement**

**2)Data**

**3)Pre Processing**

**4)Model selection**

**5)Feature Selection**

**6)Hyper tuning**

**7)Model implementation**

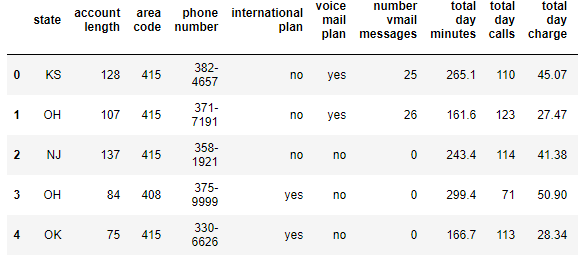
**8)Conclusion**

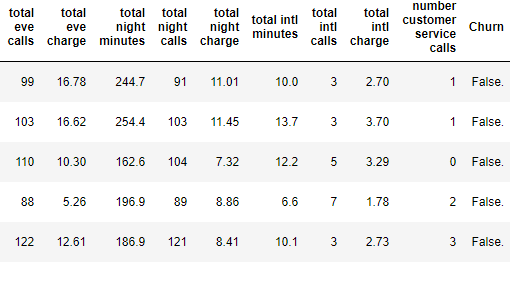
**1)Problem Statement**

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. Lets first look at the data:

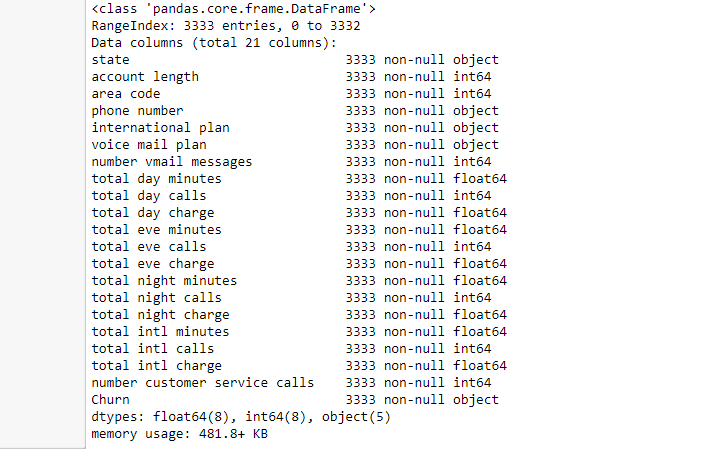
**2)Data**

Before anything else we shall first see first few rows of data that we are dealing with and try to get a basic understanding:



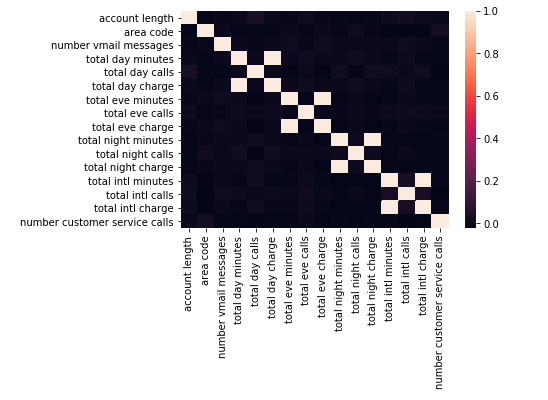
****

We can see that Churn column is our target variable and all the remaining columns are our features. Now lets see if all the columns have the right data type and if there are any missing values:

**3)Pre Processing**

Observing the results above, there does not seem to be any null values. We can confirm the same from an assert statement.

As part of Exploratory data analysis we shall first take a look at the correlation between numerical variables. A heatmap showing the same is as below:

****

We can see that the below pairs are highly correlated with each other:

1)'total day calls' and 'total day charge'

2)'total eve calls' and 'total eve charge'

3)'total night calls' and 'total night charge'

4)'total intl calls' and 'total intl charge'

After taking a look at numerical variables, the columns of type “object” were converted to categorical and then to numerical by getting dummies.

Target and feature variables were then separated for both train data. The same pre processing techniques were used on test data as well.

Some models do not perform well if we feed correlated columns as we are actually feeding duplicate information. But on the models that were selected which we will see in the next chapter, this did not have a major impact.

**4)Model selection**

After pre processing, we will then check which model may be a better selection for this classification prediction.

We will consider two models, Descision Tree and Random Forest

A decision tree is a graph that uses a branching method to illustrate every possible outcome of a decision

Random forest (or random forests) is a trademark term for an ensemble classifier that consists of many decision trees and outputs the class that is the mode of the classes output by individual trees. Random forests are collections of trees, all slightly different. It randomize the algorithm, not the training data.

In this case we will start with Decision Tree classification model and note the results and later see if Random Forest will match the scores of Decision Tree classification.

Speaking about scores, as our aim is to reduce churn, along with accuracy, recall will be our key metric.

Recall=True Positives/(True Positives + False Negatives)

We will also consider ROU\_AUC.

While the Recall score is an important metric for measuring the accuracy of a classification algorithm, it puts too much weight on the number of False Negatives. On the other hand, Precision is concentrated on the number of False Positives.

The combination of those two results in the ROC curve allows us to measure both recall and precision. The area under the ROC curve is calculated as the AUC score.

The result of Decision Tree is as below:

Accuracy\_Score = 91.90161967606478

Recall = 70.53571428571429

ROC\_AUC = 82.87700475200474

For the same data, results of Random Forest is as below:

Accuracy\_Score = 94.24115176964607

Recall = 58.92857142857143

ROC\_AUC = 79.32568557568558

We can see that Decision Tree is performing well in terms of Recall and Random Forest is performing well in terms of Accuracy\_Score.

**5)Feature Selection**

Among other things, Decision Trees and Random Forest

are very popular because of their interpretability. Many

models can provide accurate predictions, but Decision

Trees and Random Forest can also quantify the effect of

the different features on the target. Here, it can tell you

which features have the strongest and weakest impacts on the decision to leave the company. In sklearn, we can get

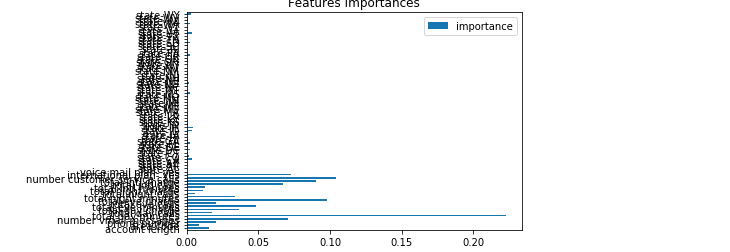
this information by using the feature\_importances\_

attribute.

Important features were extracted based on results of the model and a new train data was created which only

consisted of important features. A bar plot showing

importance is as below:



**6)Hyperparameter**

**tuning**

A hyperparameter is a parameter inside a function. For example, max\_depth or min\_samples\_leaf are hyperparameters of the DecisionTreeClassifier() function. Hyperparameter tuning is the process of testing different values of hyperparameters to find the optimal ones: the one that gives the best predictions according to our objectives. In sklearn, we can use GridSearch to test different combinations of hyperparameters. Even better, we can use GridSearchCV() to test different combinations and run cross-validation on them in one function!

In this case we are going to prepare the different values we want to test for max\_depth and min\_samples\_leaf for Decision Tree and n\_estimators, max\_depth and min\_samples\_leaf for Random Forest. We will then put these in a dictionary, because that’s what is required for GridSearchCV() and try to find the best parameters:

* the dictionary keys will be the hyperparameters names
* the dictionary values will be the attributes (the hyperparameter values) you want to test

**7)Model Implementation**

After feature selection and hypertuning parameters, output of models are as below:

The result of Decision Tree is as below:

Accuracy\_Score = 91.42171565686863

Recall = 79.46428571428571

ROC\_AUC = 86.37108949608951

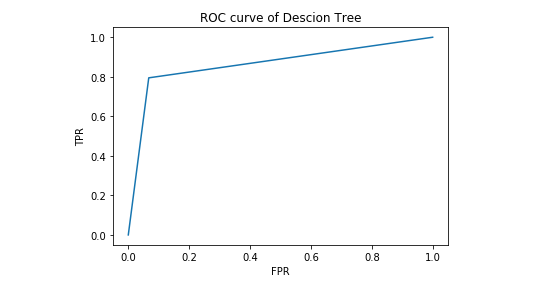
For the same data, results of Random Forest is as below:

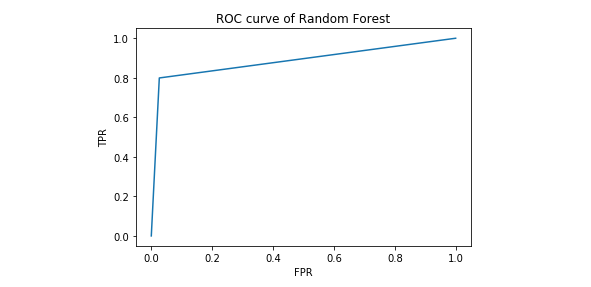
Accuracy\_Score = 95.02099580083984

Recall = 79.91071428571429

ROC\_AUC = 88.63865582615583

We can clearly see that random forest is performing better both in terms of accuracy\_score and recall. Now lets see roc curve for both the models:





We can see that graph for Random Forest has more area under the curve. Hence we can go ahead and freeze

Random Forest model.

**8)Conclusion**

In Python, the model has an accuracy of 95% which means that our model’s prediction is right 95% of the times

The model’s recall is close to 80% which means that our model is correctly predicting customers who churn roughly 4 out of 5 times.

In R, the Random forest model achieved an accuracy of 95% and recall score of 70%