Telecom Customer Churn Prediction Using Machine Learning Models

University of Western Ontario, London, Ontario
Introduction to Data Science

Akanksha Nayer
anayer@uwo.ca
251001953

Jessica Patel
jpate272@uwo.ca
251038682

Priyanka Naikade
pnaikade@uwo.ca
251002802

ABSTRACT

Telecom business is a versatile domain, with a billion-dollar business. For it to flourish it is necessary to consider all the factors leading to its failure or success, or potential downfall or growth. Data science in the business domain that includes the rate at which the business can proliferate, using various models for various problems. In this paper we will discuss about how various machine learning techniques can be used to predict future churn behavior of customers and select the best model by comparing the performances of the all models. The concepts of Data Preprocessing, Sampling dataset and Classification are applied for the same.

Introduction

The industry is at the epicenter of growth with the advancements being made in mobile and internet services. With the increasing number of competitors in the market, it has become a challenge to meet the growing customer expectations and provide delightful customer services. Thus, the customers are switching telecom companies for various reasons and as a result the business gets adversely affected. This has been continuing for years, but for managing the business and bringing up the customer number and profit, it is necessary that the telecom companies must either retain the existing customers or draw new customers. But the latter is difficult since acquisition of new customers is time consuming and expensive for the companies. Henceforth, this leaves us with the major option of retaining existing customers and in order to do so, the company needs to identify the customers who are likely to churn and target them by devising and imposing different marketing strategies upon them. The predictive machine learning models can facilitate the telecom companies in achieving the same.

The Data

Source of the dataset is IBM Sample Dataset where dataset was extracted in csv format. The Tabular data consists of 7049 rows and 23 columns, where the columns represent the customer attributes/features and the rows corresponds to the individual customers. Out of the 23 attributes, 22 will be the predictor variables and one will be the Target Variable(Churn) to forecast. The dataset constitutes data of various types like numerical, alphanumeric and categorical.

Input Data

Independent variables being considered - (Customer Id, Phone services, Gender, Age, Senior Citizen, Partner, Online Security, Online backup, Device Protection, Multiple Lines, Tech Support, Streaming TV, Streaming Movies, Paperless Billing, Payment Method, Monthly Charges, Total Charges, Internet Service, Contract, Tenure, Dependents)

Output Data

Target/Response being considered - Churn (the dependent target variable which is dichotomous and mutually exclusive)
Practically predicting a model allows the company to have scientific basis for predicting the likelihood for churn and therefore helping them optimize their business development efforts.

Initial Impression

The data seemed suggestive, but not usable straightaway, especially given the large number of records that had categorical attributes in large proportion. The dataset needs to be pre-processed to identify outliers, missing values and analyzed further to determine the features relevant to the business problem statement, i.e., predicting the Churn Status.

Data Pre-Processing

Exploratory Data Analysis

The exploratory data analysis is used for performing critical investigation of data to identify hidden patterns, spot anomalies, test hypothesis with the help of summary statistics and visualization methods. We have used various predefined functions from the pandas library to perform the exploratory analysis. The shape() function tells us the number of rows and columns presents in the dataset. The info() provides details about the type of the data columns used, which in our case are – Object, Float and Int data types.

The very first step of any data analysis would be to check if the dataset is clean or has any missing values/outliers present in it. We have used isnull() to check if there are any NULL values present in any of the columns.

Describe() is used to provide the summary statistics of the numerical data attribute. It returns the count, minimum, maximum, mean, median and quartiles of the data. On visualizing the numerical attributes using a boxplot we observed that there is a huge difference between the Maximum value and 75% percentile value of 'MonthlyCharges' and 'TotalCharges' column indicating the presence of Outliers.

There were several features of type – 'Objects' indicating that they were categorical. We checked for its distinct values using unique() function and replaced their corresponding categorical values to numerical since machine learning algorithms cannot handle categorical values.

Data Transformation

We have replaced the NULL values with '0' and categorical values to numerical based on our analysis done in the previous step. There were 11 NULL values present in the 'TotalCharges' Column, which we replaced with the value '0'.On identifying the underlying hidden pattern of those 11 observations we found that the values for its corresponding column 'Tenure' were '0'. Hence, we replaced the NULL values of 'TotalCharges' with '0' instead of filling it with the Mean value. All the features were of type int and float post transformation of categorical values.

Data Visualization

As per our initial data investigation we found that there were outliers present in 'MonthlyCharges' and 'TotalCharges' Column. We have used the Inter-Quartile Range (IQR) method to detect and remove the outliers, wherein the points which are farther from 25% and 75% quartile were removed. Since the values were unrealistic like extremely high and negative values that lay far from the rest of the distribution.

Histogram plots

We have implemented histogram plots as a data visualization part in order to understand the shape of data, for example, normal, skewed etc., and understanding the value of variable i.e., plotting of each variable with respect to churn.

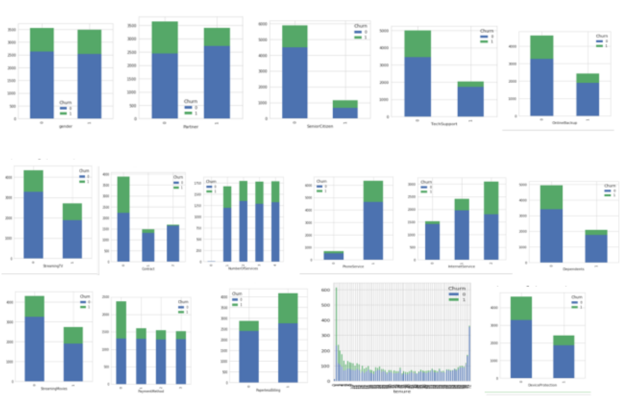


Figure 1 Histogram Plots of features with respect to Target

Feature Engineering

Feature Engineering is the process of creating new features using the existing ones or modify the existing features by infusing the domain knowledge in such a way that it increases the accuracy of the machine learning algorithms. In our dataset we grouped the continuous values of 'Age' column into 5 different groups and labelled each using categorical values from 0-4. This will help the industry to target any particular age group and accordingly devise marketing strategies because the company would be interested in focusing on a particular age group rather than individuals.

Feature Selection using correlation analysis

We have generated a correlation matrix for our dataset using corr() function which uses *Pearson's* method which is based on the method of covariance. The Matrix as shown in figure 2 shows how strong the features are associated with each other and their direction of relationship, i.e., positive, negative or no association.

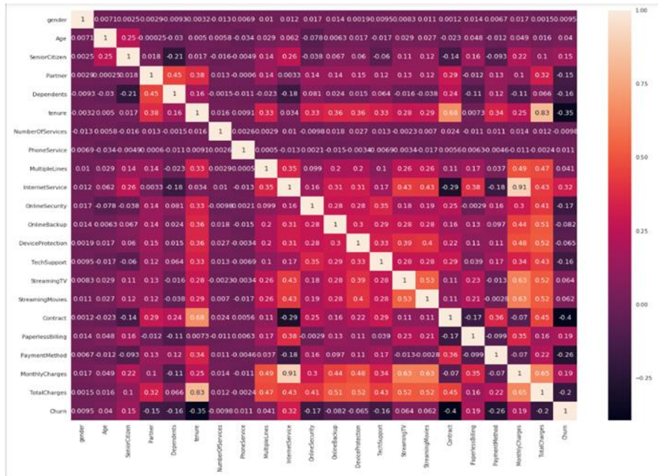
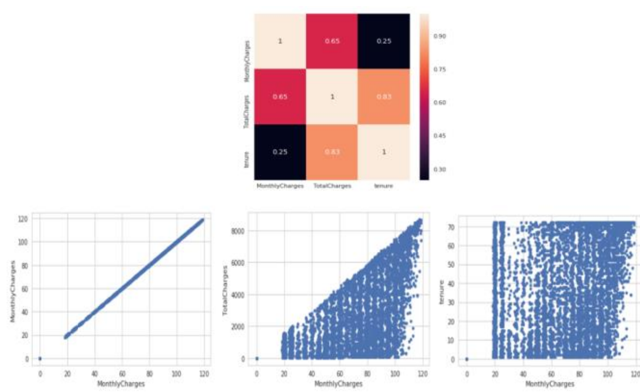


Figure 2 Correlation Matrix

As given in the figure 2.a) we observed that 'MonthlyCharges' and 'tenure' were not strongly associated with each other with the correlation coefficient value being 0.25, whereas 'TotalCharges' was highly correlated with 'MonthlyCharges' and 'tenure' with coefficient values being 0.65 and 0.83. Based on these observations we dropped the 'TotalCharges' column in order to avoid multi-collinearity and also the redundancy which is also quite obvious from the fact that 'TotalCharges' can be determined when 'tenure' and 'MonthlyCharges' is known.



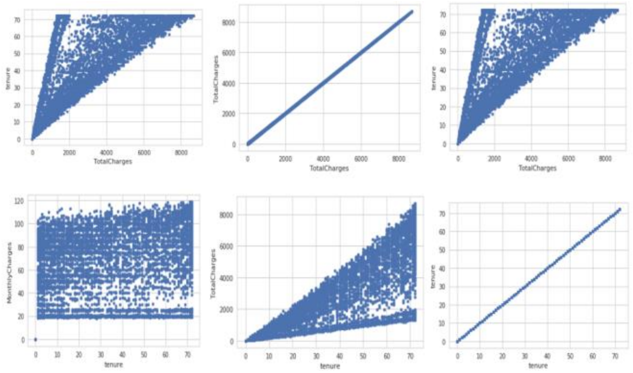


Figure 2.a) Correlation matrix and Scatter Plots for 3 features - MonthlyCharges, TotalCharges and Tenure

Features like 'CustomerId' and 'NumberofServices' had no association with the target variable (Churn) and hence we dropped these columns as well prior to model building.

Training Set and Test set

We have split the data using the train_test_split() function from the model_selection library. The data is divided into training set and test set in the ratio of 80:20. We trained our models using the training dataset and validated using the test set.

Classifiers

After the preprocessing our data we implemented the following machine learning algorithms on our training data for future predictions

Logistic Regression

Logistic regression is a supervised classification algorithm where the target variable (or output) can take only discrete values for given set of features (or inputs). In our case the target variable (Churn) is categorical and can have only 2 possible types: "0" or "1". Thus, we use binomial logistic regression for our model.

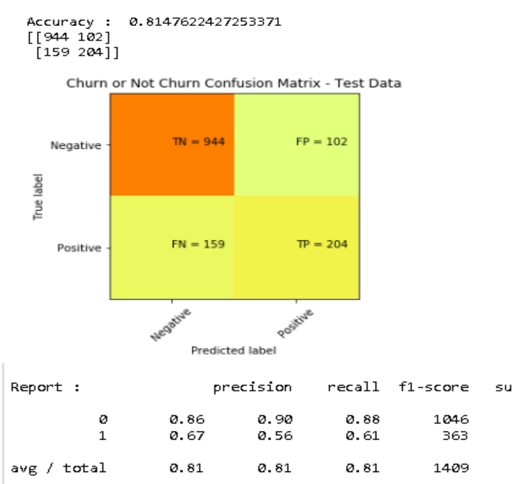


Figure 3.a) Accuracy for Logistic Regression

On testing the Logistic Regression classifier using the test set, the model resulted in 81% accuracy. Based on our observation from figure 3.a) the model has predicted more number of negative labels compared to positives resulting in huge difference between the Precision/Recall values for 0s and 1s(Not Churn & Churn). This indicates that the dataset is highly imbalanced where large proportion of dataset has negative labels than positives. There is a risk where the model trained on this data will always predict results in favor of majority class label. In order to avoid this and balance the data we upsampled the entire dataset, wherein the observations from minority class were duplicated such that negative and positive label ratio was 50:50. The number of negative and positive samples were 5174 and 1869 respectively. Using resample() function we duplicated the minority class to level up to 5174 rows, same as negative(majority) class.

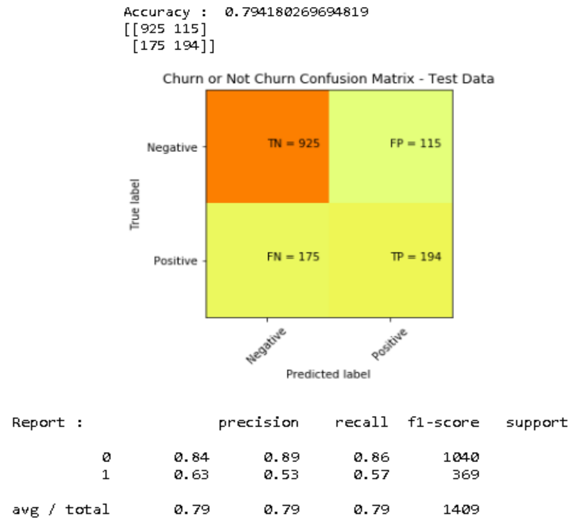


Figure 3.b) Accuracy for Logistic Regression after sampling

Support Vector Machines (SVM)

Support Vector Machine is supervised classification algorithm which works by identifying an optimal hyper-plane (defined by the tuples near the class boundaries, known as Support Vectors) that separates the two classes of labels. Coming to our dataset we need to determine which SVM to use: Simple SVC (Support Vector Classifier) or Kernel SVM for the classification. Simple SVC finds a decision boundary for linearly separable data and kernel SVM maps the non-linear inputs into higher dimensional feature space using kernel functions and then identifies a decision boundary. We have trained the model using both Simple SVC and Radial Basis Function (RBF) Kernel methods and identified which suits best for our dataset.

Linear Kernel: $k(x, y) = x^T y + c$

Gaussian Radial Basis Function:

$$k(x, y) = \exp\left(-\frac{\|x - y\|^2}{2\sigma^2}\right)$$

For Simple SVC, we tuned the model using the cost function, C, which is a penalty to allow certain misclassifications by maximizing the margin width. For our dataset the model gave better accuracy approximately 81.97% when the value of C=1. For RBF SVM, we trained the model using two hyper parameters - C Penalty and Gamma Parameters which resulted in accuracy approximately 71%, wherein the model predicted all the observations as negatives with no positives for various combinations of C and Gamma values as shown in figure 4.b). The RBF model predicted all the observations as negatives even after upsampling the dataset. Hence we selected the simple SVC as the SVM model for our dataset.

SVM Linear		
Tuning Parameter	**Accuracy**	**AUC-ROC score**
C=0.5	79.91%	0.71
C=1	81.97%	0.74
C=5	81.68%	0.73
C=10	80.62%	0.72
C=100	79.92%	0.71

Table 1: Results of Linear SVM for different cost functions

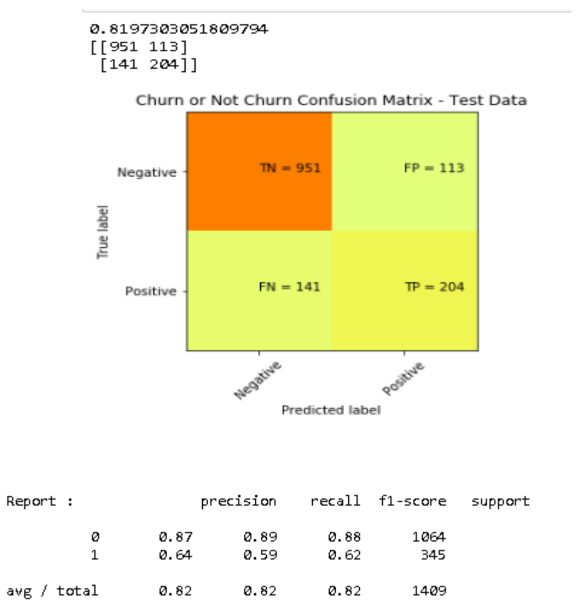


Figure 4.a) Results for Linear SVM when C=1

SVM RBF	
Tuning Parameters	**Resulted Labels**
C=1, Gamma =0.1	All Negatives
C=1, Gamma =0.01	All Negatives
C=1, Gamma =0.001	All Negatives
C=1, Gamma =0.0001	All Negatives
C=5, Gamma =0.1	All Negatives
C=5, Gamma =0.01	All Negatives
C=5, Gamma =0.001	All Negatives
C=5, Gamma =0.0001	All Negatives
C=10, Gamma =0.1	All Negatives
C=10, Gamma =0.01	All Negatives
C=10, Gamma =0.001	All Negatives
C=10, Gamma =0.0001	All Negatives

Figure 4.b) Results for RBF SVM

Decision Tree:

It is a supervised learning algorithm that makes prediction by making simple decision rules/attribute value test that in return split the source set into subsets. Initially for our model building we do not pass any parameters in the Decision Tree Classifier such that the default depth and minimum sample per leaf are set to unlimited which lead to formation of a fully grown and an unpruned tree as shown in figure 5.a)

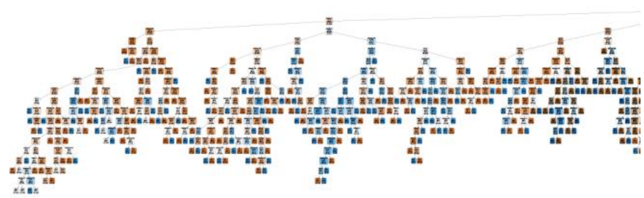


Figure 5.a) The Unpruned Tree

The unpruned tree will include all nodes trying to fit each training example perfectly. This model will give us a higher accuracy of 1 on training set but the accuracy decreases on test set to 0.73. This means that we are overfitting the model.

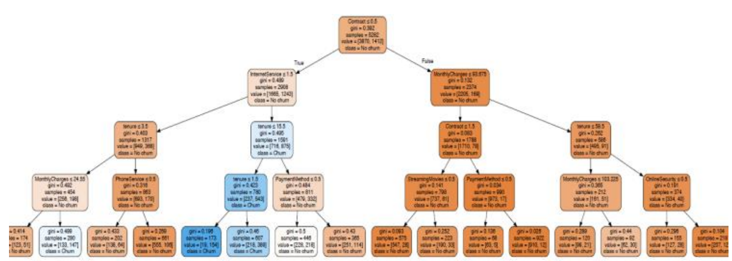


Figure 5.b) Pruned tree with depth 4

Trimming the tree

We took the max_depth parameter of the classifier as 4 and then checked the accuracy for this unpruned tree on both training and test set that comes out to be 0.79 for both. Though the classifier is not highly accurate for the training set but it does perform well on the test data with accuracy of 0.79.

```
Accuracy of unpruned decision tree classifier on training set: 1.00
Accuracy of unpruned decision tree classifier on test set: 0.73
Accuracy of decision tree classifier on training set: 0.79
Accuracy of decision tree classifier on test set: 0.79
```

```
Report :              precision    recall  f1-score   support

         0.0       0.84      0.90      0.87      1045
         1.0       0.65      0.50      0.56       364

avg / total       0.79      0.80      0.79      1409
```

Random Forest

Random forest is an ensembled supervised learning algorithm which is used for both classification and regression. Random Forest classifier creates a set of decision trees from randomly selected subset of training data and gets the output/predictions from each decision tree and then selects the best prediction using the majority votes for a given class from every tree, in case of classification. More the numbers of trees used, more robust is the forest.

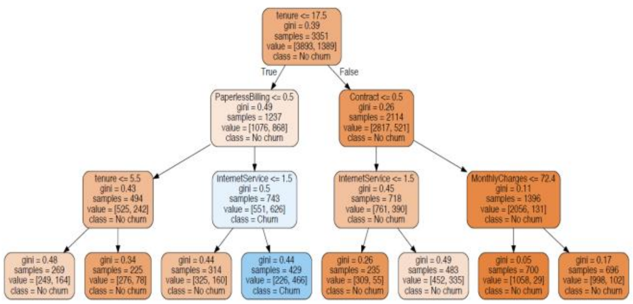


Figure 6.a) Single Decision Tree from a Random Forest with n_estimators =20

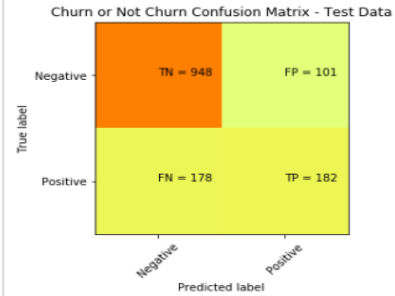
Random Forest	Accuracy
n_estimators=10	78.14%
n_estimators=20	79.20%
n_estimators=30	78.14%
n_estimators=50	78.06%
n_estimators=100	78.14%

For Trees = 10	Accuracy	For Trees = 20	Accuracy	For Trees=100	Accuracy
Max_Depth = 3	79.84%	Max_Depth =3	77%	Max_Depth =3	76.08%
Max_Depth = 6	80.19%	Max_Depth =6	78.06%	Max_Depth=6	79.91%
Max_Depth = 10	77.92%	Max_Depth =10	79.70%	Max_Depth=10	79.77%
Max_Depth = 12	79.34%	Max_Depth = 12	78.77%	Max_Depth = 12	77%

Figure 6.b) Results of Random Forest Classifier using various combination of parameters - Number of Trees & Maximum Depth of tree

We trained the random forest classifier using hyper-tuning parameters - N_estimators which is the number of trees in a forest and Max_Depth which is the depth of a tree, as given in figure 6.b). The model showed higher accuracy of about 80.19% when the number of trees were 10 and the maximum depth of the trees was 6.

```
Accuracy :  0.8019872249822569
[[948 101]
 [178 182]]
```




```
Report :              precision    recall  f1-score   support

           0       0.83      0.89      0.86      1002
           1       0.66      0.55      0.60       407

avg / total       0.78      0.79      0.78      1409
```

Figure 6.c) Accuracy is high when Number of Trees used = 10 and Maximum Depth = 6

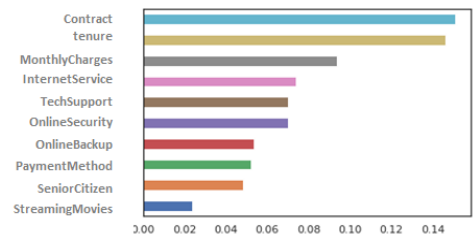


Figure 6.d) Feature Importance Score using Mean Decrease in Accuracy

Random Forest also offers a good feature selection indicator. It uses Gini Importance and Mean Decrease in Impurity (MDI) to calculate the importance of each feature. It shows by what percent the accuracy will be decreased by removing that feature. Higher decrease in accuracy indicates the feature is most significance. For our dataset the model built based on the important features as shown in figure 6.d) resulted in accuracy of approximately 80% same as that of previous.

K- Fold Cross Validation on Models -

We have used K cross validation method for validating and selecting the best model for our dataset. Using cross validation the classifier parameters were optimized and regularized to avoid overfitting in case of Decision tree and random forest classifier.

Models	5-Fold Cross Validation	10-Fold Cross Validation
Logistic Regression	66.14%	74.65%
Decision Tree	54.64%	69.98%
Random Forest	50.05%	70.06%
Support Vector Machine	65.76%	74.62%

Table 2 - Results of 5-fold and 10-fold Cross Validations

As depicted in table 2, we noticed that the 10-fold cross validation generated higher accuracy for all the models compared to 5-fold. Thus, we selected the value of K=10 for the cross validation. Secondly, both logistic regression and SVM classifier had almost same accuracy which was high compared to the other two models, indicating that they are the best models for our dataset.

Performance Evaluation for all models

Post K-fold Cross Validation, we made predictions on our best models using the test data and computed the following:

1. **Confusion Matrix:** We used this for our model to calculate the performance metrics (Precision, Recall, F1 Score and Support) of the classifier.

2. **AUROC (Area Under the Receiving Operating Characteristics Curve):** We calculated the Area Under the Curve(AUC) which determines how much a model is capable of distinguishing the classes. Closer the value is to 1 better is the classifier for our data.

Metrics/Models	Logistic Regression	Decision Tree	Random Forest	Support Vector Machine
Accuracy	81.47%	79%	80.19%	81.97%
AUC-ROC	0.73	0.73	0.72	0.74
Precision	81%	79%	78%	82%
Recall	81%	80%	79%	82%
F1 Score	81%	79%	78%	82%
Support	1409	1409	1409	1409

Table 3 - Comparison of Models based on the performance evaluation on the test data

Conclusion

Practically predicting a model allows the company to have scientific basis for predicting the likelihood for churn and therefore helping them optimize their business development efforts. In this paper we have discussed various prediction models with comparison of their quality measures in terms of accuracy using evaluation techniques. We observed that Logistic Regression is the best model in predicting the churn behavior of a customer. Secondly, the upsampling of data distribution done to balance the class labels did not result in significant increase in the accuracy of the models like Logistic Regression and Support Vector Machines.

Future Scope

The future scope of this paper will be to combine the structured data along with the unstructured data which can be gathered by the feedbacks received from the Customers about the services and the opinions of customers collected from various social networking sites. Combining these data would increase the customer churn prediction precision as the unstructured data holds valuable information about how happy or unhappy the customer is about the services being provided by the company. We can use various Machine learning techniques like Deep Learning methods to analyze the unstructured data to do sentimental analysis. Secondly we can combine multiple algorithms together in order to make precise churn predictions. This would ease and facilitate the telecom industry to identify the customers who are likely to leave the company in a better and precise way.

References

1. https://www.kaggle.com/zagarsuren/telecomcchurn-dataset-ibm-watson-analytics/version/1
2. https://seaborn.pydata.org/
3. https://towardsdatascience.com/random-forest-in-python-24d0893d51c0
4. https://towardsdatascience.com/exploratory-data-analysis-8fc1cb20fd15
5. https://scikit-learn.org/0.16/modules/generated/sklearn.tree.export_graphviz.html
6. https://scikit-learn.org/stable/auto_examples/svm/plot_iris.html
7. Customer Churn Prediction Using Sentiment Analysis and Text Classification of VOC - Yiou Wang, Koji Satake, Takeshi Onishi, Hiroshi Masuichi
8. https://jakevdp.github.io/PythonDataScienceHandbook/05.08-random-forests.html
9. https://medium.com/greyatom/lets-learn-about-auc-roc-curve-4a94b4d88152
10. Customer Churn Analysis in Telecom Industry - Kiran Dahiya, Surbhi Bhati
11. .https://www.kaggle.com/bandiatindra/telecom-churn-prediction

APPENDIX A

Detailed Snapshots of results achieved from project:

DATA PRE-PROCESSING

EXPLORATORY DATA ANALYSIS

```
273]: df.head()
273]:
```

	customerID	gender	Age	SeniorCitizen	Partner	Dependents	tenure	NumberOfServices	PhoneService	MultipleLines	...	DeviceProtection	T
0	5575-GNVDE	Male	24	No	No	No	34	4	No	No	...	Yes	
1	3668-QPYBK	Male	28	No	No	No	2	1	No	No	...	No	
2	7795-CFOCW	Male	32	No	No	No	45	2	No	No phone service	...	Yes	
3	6713-OKOMC	Female	29	No	No	No	10	4	No	No phone service	...	No	
4	6388-TABGU	Male	34	No	No	Yes	62	2	No	No	...	No	

5 rows × 23 columns

```
In [274]: df.shape
Out[274]: (7049, 23)

In [275]: df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7049 entries, 0 to 7048
Data columns (total 23 columns):
customerID          7049 non-null object
gender              7049 non-null object
Age                 7049 non-null int64
SeniorCitizen       7049 non-null object
Partner             7049 non-null object
Dependents          7049 non-null object
tenure              7049 non-null int64
NumberOfServices    7049 non-null int64
PhoneService        7049 non-null object
MultipleLines       7049 non-null object
InternetService     7049 non-null object
OnlineSecurity      7049 non-null object
OnlineBackup        7049 non-null object
DeviceProtection    7049 non-null object
TechSupport         7049 non-null object
StreamingTV         7049 non-null object
StreamingMovies     7049 non-null object
Contract            7049 non-null object
PaperlessBilling    7049 non-null object
PaymentMethod       7049 non-null object
MonthlyCharges      7049 non-null float64
TotalCharges        7038 non-null float64
Churn               7049 non-null object
dtypes: float64(2), int64(3), object(18)
memory usage: 1.2+ MB
```

```
: df.isnull().sum()

: customerID          0
  gender              0
  Age                 0
  SeniorCitizen       0
  Partner             0
  Dependents          0
  tenure              0
  NumberOfServices    0
  PhoneService        0
  MultipleLines       0
  InternetService     0
  OnlineSecurity      0
  OnlineBackup        0
  DeviceProtection    0
  TechSupport         0
  StreamingTV         0
  StreamingMovies     0
  Contract            0
  PaperlessBilling    0
  PaymentMethod       0
  MonthlyCharges      0
  TotalCharges       11
  Churn               0
  dtype: int64
```

```
]: df['gender'].unique()
]: array(['Male', 'Female', 0], dtype=object)
```

```
]: df['gender'].unique()
]: array(['Male', 'Female', 0], dtype=object)
```

```
]: df['SeniorCitizen'].unique()
]: array(['No', 'Yes', 0], dtype=object)
```

```
]: df['Partner'].unique()
]: array(['No', 'Yes', 0], dtype=object)
```

```
]: df['Dependents'].unique()
]: array(['No', 'Yes', 0], dtype=object)
```

```
]: df['NumberOfServices'].unique()
]: array([4, 1, 2, 3, 0])
```

```
]: df['PhoneService'].unique()
]: array(['No', 0, 'Yes'], dtype=object)
```

```
]: df['MultipleLines'].unique()
]: array(['No', 'No phone service', 'Yes', 0], dtype=object)
```

```
]: df['InternetService'].unique()
]: array(['DSL', 0, 'Fiber optic', 'No'], dtype=object)
```

```
df['OnlineSecurity'].unique()
array(['Yes', 0, 'No internet service', 'No'], dtype=object)
```

```
df['OnlineBackup'].unique()
array(['No', 'Yes', 0, 'No internet service'], dtype=object)
```

```
df['DeviceProtection'].unique()
array(['Yes', 'No', 0, 'No internet service'], dtype=object)
```

```
df['TechSupport'].unique()
array(['No', 'Yes', 0, 'No internet service'], dtype=object)
```

```
df['StreamingTV'].unique()
array(['No', 'Yes', 0, 'No internet service'], dtype=object)
```

```
df['StreamingMovies'].unique()
array(['No', 'Yes', 0, 'No internet service'], dtype=object)
```

```
df['Contract'].unique()
array(['One year', 'Month-to-month', 'Two year', 0], dtype=object)
```

```
df['PaperlessBilling'].unique()
array(['No', 'Yes', 0], dtype=object)
```

```
df['PaymentMethod'].unique()
array(['Mailed check', 'Bank transfer (automatic)',
       'Credit card (automatic)', 'Electronic check', 0], dtype=object)
```

```
[379]: df[df['TotalCharges'].isna()==True]=0
```

```
[380]: df.isnull().sum()
[380]: customerID          0
       gender              0
       Age                 0
       SeniorCitizen       0
       Partner             0
       Dependents          0
       tenure              0
       NumberOfServices    0
       PhoneService        0
       MultipleLines       0
       InternetService     0
       OnlineSecurity      0
       OnlineBackup        0
       DeviceProtection    0
       TechSupport         0
       StreamingTV         0
       StreamingMovies     0
       Contract            0
       PaperlessBilling    0
       PaymentMethod       0
       MonthlyCharges      0
       TotalCharges        0
       Churn               0
       dtype: int64
```

```
df['gender'].replace(['Male','Female'],[0,1],inplace=True)
df['SeniorCitizen'].replace(['Yes','No'],[1,0],inplace=True)
df['Partner'].replace(['Yes', 'No'],[1,0],inplace=True)
df['Dependents'].replace(['Yes', 'No'],[1,0],inplace=True)
df['PhoneService'].replace(['Yes','No'],[1,0],inplace=True)
df['MultipleLines'].replace(['No phone service', 'No', 'Yes'],[0, 0, 1],inplace=True)
df['InternetService'].replace(['No', 'DSL', 'Fiber optic'],[0, 1, 2],inplace=True)
df['OnlineSecurity'].replace(['No', 'Yes', 'No internet service'],[0, 1, 0],inplace=True)
df['OnlineBackup'].replace(['Yes', 'No', 'No internet service'],[1, 0, 0],inplace=True)
df['DeviceProtection'].replace(['No', 'Yes', 'No internet service'],[0, 1, 0],inplace=True)
df['TechSupport'].replace(['No', 'Yes', 'No internet service'],[0, 1, 0],inplace=True)
df['StreamingTV'].replace(['No', 'Yes', 'No internet service'],[0, 1, 0],inplace=True)
df['StreamingMovies'].replace(['No', 'Yes', 'No internet service'],[0, 1, 0],inplace=True)
df['Contract'].replace(['Month-to-month', 'One year', 'Two year'],[0,1,2],inplace=True)
df['PaperlessBilling'].replace(['Yes', 'No'],[1, 0],inplace=True)
df['PaymentMethod'].replace(['Electronic check', 'Mailed check', 'Bank transfer (automatic)'
df['Churn'].replace(['Yes', 'No'],[1, 0],inplace=True)
```

```
df.info(3)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7049 entries, 0 to 7048
Data columns (total 22 columns):
gender              7049 non-null int64
Age                 7049 non-null int64
SeniorCitizen       7049 non-null int64
Partner             7049 non-null int64
Dependents          7049 non-null int64
tenure              7049 non-null int64
NumberOfServices    7049 non-null int64
PhoneService        7049 non-null int64
MultipleLines       7049 non-null int64
InternetService     7049 non-null int64
OnlineSecurity      7049 non-null int64
OnlineBackup        7049 non-null int64
DeviceProtection    7049 non-null int64
TechSupport         7049 non-null int64
StreamingTV         7049 non-null int64
StreamingMovies     7049 non-null int64
Contract            7049 non-null int64
PaperlessBilling    7049 non-null int64
PaymentMethod       7049 non-null int64
MonthlyCharges      7049 non-null float64
TotalCharges        7049 non-null float64
Churn               7049 non-null int64
dtypes: float64(2), int64(20)
memory usage: 1.2 MB
```

```
df.head(3)
```

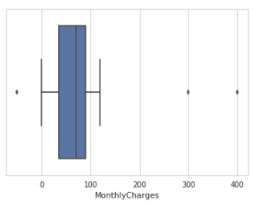
	customerID	gender	Age	SeniorCitizen	Partner	Dependents	tenure	NumberOfServices	PhoneService	MultipleLines	...	DeviceProtection	Tec
0	5575-GNVDE	0	24	0	0	0	34	4	0	0	...	1	
1	3668-QPYBK	0	28	0	0	0	2	1	0	0	...	0	
2	7795-CFOCW	0	32	0	0	0	45	2	0	0	...	1	

3 rows × 23 columns

DATA VISUALIZATION:

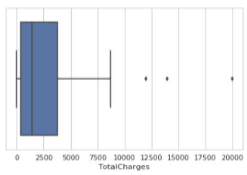
```
sns.boxplot(x=df['MonthlyCharges'])
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f0e3a510ef0>




```
sns.boxplot(x=df['TotalCharges'])
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f0e3877f0b8>




```
Q1 = df['MonthlyCharges'].quantile(0.25)
Q3 = df['MonthlyCharges'].quantile(0.75)
IQR = Q3 - Q1
print(IQR)
```

54.39999999999999

```
print((df['MonthlyCharges'] < (Q1 - 1.5 * IQR)) | df['MonthlyCharges'] > (Q3 + 1.5 * IQR))
```

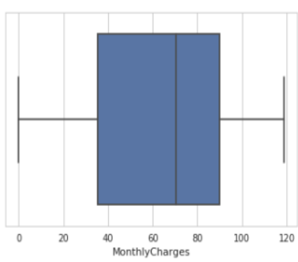
```
df = df.drop(df[(df.MonthlyCharges < (Q1 - 1.5 * IQR)) | (df.MonthlyCharges > (Q3 + 1.5 * IQR))].index)
```

```
Q1 = df['TotalCharges'].quantile(0.25)
Q3 = df['TotalCharges'].quantile(0.75)
IQR = Q3 - Q1
print(IQR)
df = df.drop(df[(df.TotalCharges < (Q1 - 1.5 * IQR)) | (df.TotalCharges > (Q3 + 1.5 * IQR))].index)
```

Post removal of outliers present in both the columns:

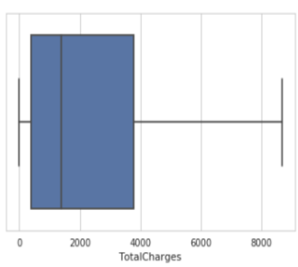
```
sns.boxplot(x=df['MonthlyCharges'])
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f0e392a1ef0>

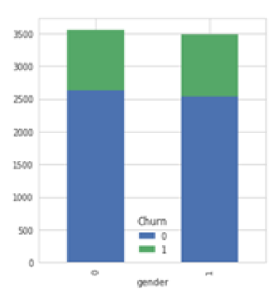


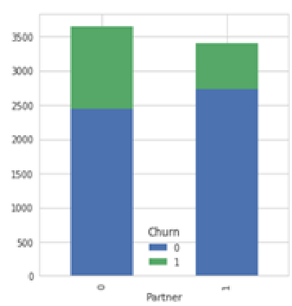

```
sns.boxplot(x=df['TotalCharges'])
```

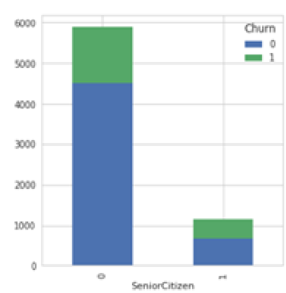
<matplotlib.axes._subplots.AxesSubplot at 0x7f0e38b4a710>

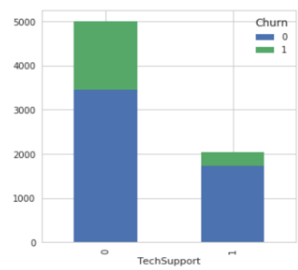


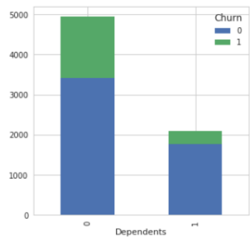
Histogram plots

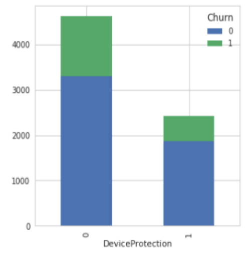


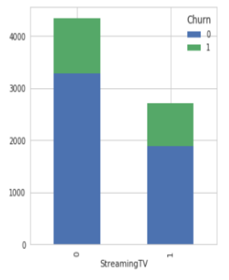


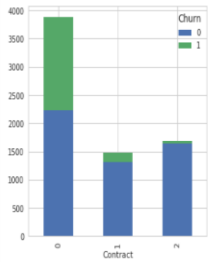


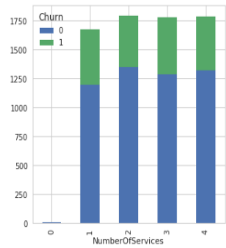


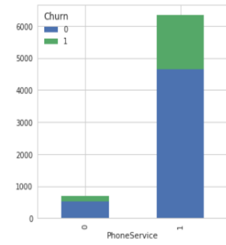


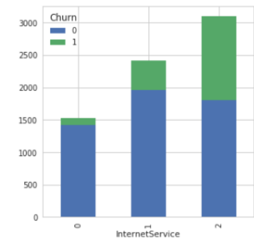


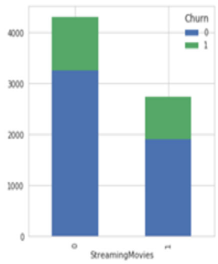


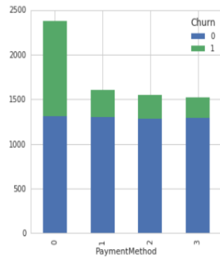


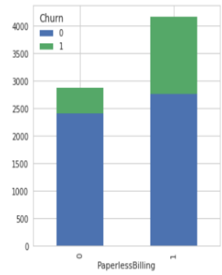


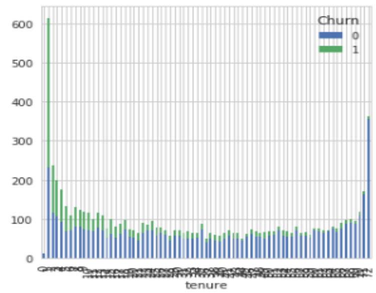












Feature Engineering:

```
df['AgeBand'] = pd.cut(df['Age'], 6)
df[['AgeBand', 'Churn']].groupby(['AgeBand'], as_index=False).mean().sort_values(by='Churn', ascending=True)
```

	AgeBand	Churn
0	(-0.096, 16.0]	0.000000
2	(32.0, 48.0]	0.251565
1	(16.0, 32.0]	0.261867
3	(48.0, 64.0]	0.288622
4	(64.0, 80.0]	0.324275
5	(80.0, 96.0]	0.375000

```
df.loc[df['Age'] <= 16, 'Age'] = 0
df.loc[(df['Age'] > 16) & (df['Age'] <= 32), 'Age'] = 1
df.loc[(df['Age'] > 32) & (df['Age'] <= 48), 'Age'] = 2
df.loc[(df['Age'] > 48) & (df['Age'] <= 64), 'Age'] = 3
df.loc[(df['Age'] > 64) & (df['Age'] <= 80), 'Age'] = 4
df.loc[(df['Age'] > 80), 'Age'] = 5
```

```
df.head(5)
```

	customerID	gender	Age	SeniorCitizen	Partner	Dependents	tenure	NumberOfServices	PhoneServi
0	5575-GNVDE	0	1	0	0	0	34	4	
1	3668-QPYBK	0	1	0	0	0	2	1	
2	7795-CFOCW	0	1	0	0	0	45	2	
3	6713-OKOMC	1	1	0	0	0	10	4	
4	6388-TABGU	0	2	0	0	1	62	2	

5 rows × 24 columns

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7049 entries, 0 to 7048
Data columns (total 24 columns):
customerID          7049 non-null object
gender              7049 non-null int64
Age                 7049 non-null int64
SeniorCitizen       7049 non-null int64
Partner             7049 non-null int64
Dependents          7049 non-null int64
tenure              7049 non-null int64
NumberOfServices    7049 non-null int64
PhoneService        7049 non-null int64
MultipleLines       7049 non-null int64
InternetService     7049 non-null int64
OnlineSecurity      7049 non-null int64
OnlineBackup        7049 non-null int64
DeviceProtection    7049 non-null int64
TechSupport         7049 non-null int64
StreamingTV         7049 non-null int64
StreamingMovies     7049 non-null int64
Contract            7049 non-null int64
PaperlessBilling    7049 non-null int64
PaymentMethod       7049 non-null int64
MonthlyCharges      7049 non-null float64
TotalCharges        7049 non-null float64
Churn               7049 non-null int64
AgeBand             7049 non-null category
dtypes: category(1), float64(2), int64(20), object(1)
memory usage: 1.2+ MB
```

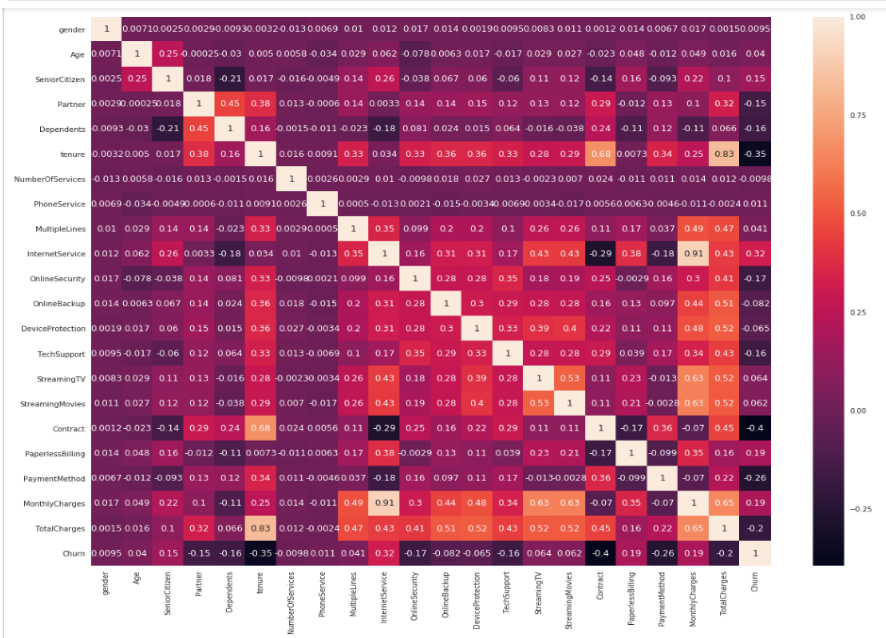
FEATURE SELECTION USING CORRELATION ANALYSIS:

```
df.pop('customerID')
df.pop('AgeBand')
df.head()
```

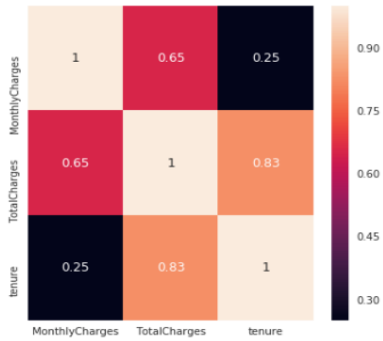
	gender	Age	SeniorCitizen	Partner	Dependents	tenure	NumberOfServices	PhoneService	MultipleLines	InternetService	...	DeviceProtection	TechSu
0	0	1	0	0	0	34	4	0	0	1	...	1	
1	0	1	0	0	0	2	1	0	0	1	...	0	
2	0	1	0	0	0	45	2	0	0	1	...	1	
3	1	1	0	0	0	10	4	0	0	1	...	0	
4	0	2	0	0	1	62	2	0	0	1	...	0	

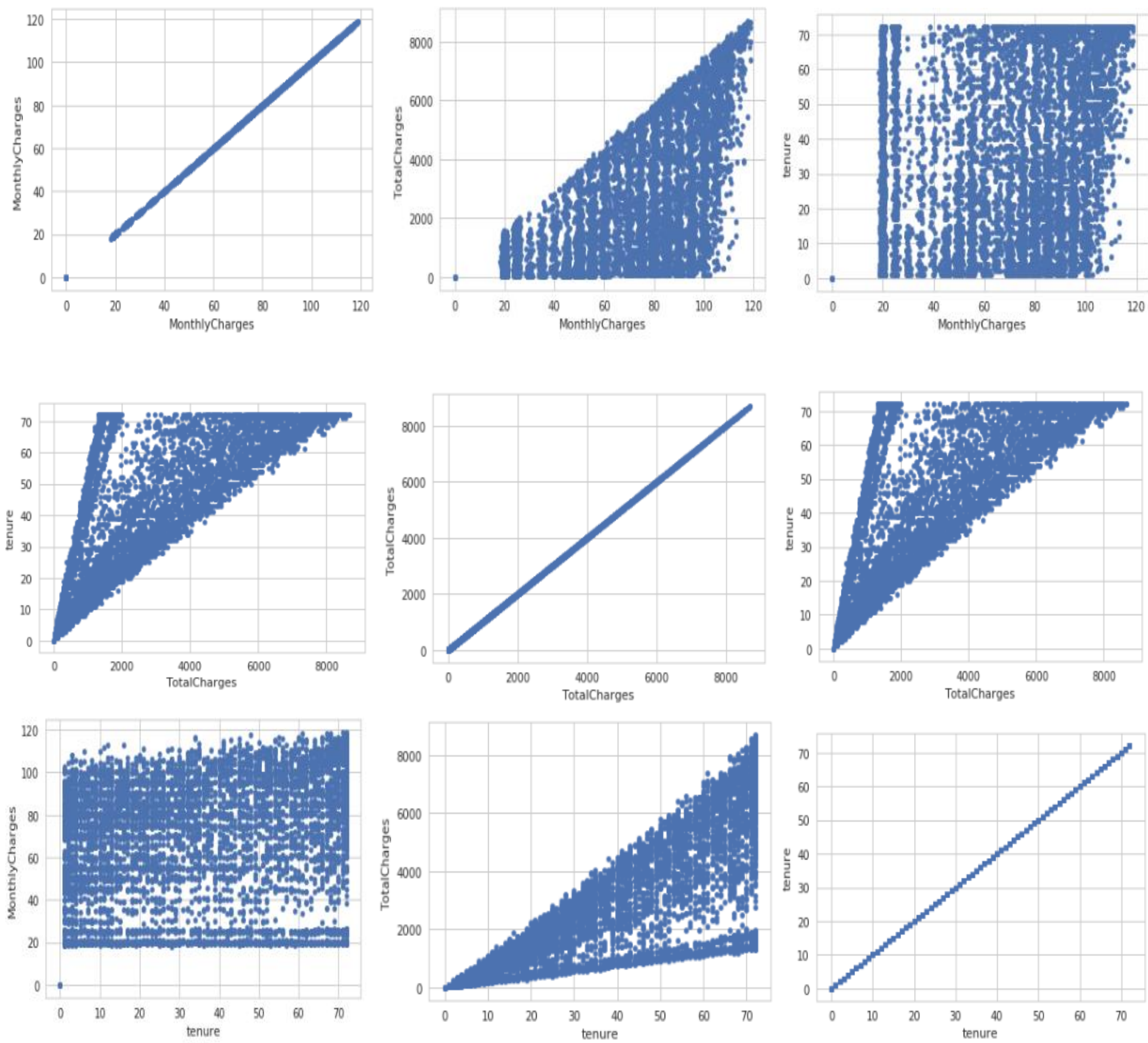
5 rows × 22 columns

```
corr = df.corr()
sns.heatmap(corr, xticklabels=corr.columns.values, yticklabels=corr.columns.values, annot = True, annot_kws={'size':12})
heat_map=plt.gcf()
heat_map.set_size_inches(20,15)
plt.xticks(fontsize=10)
plt.yticks(fontsize=10)
plt.show()
```



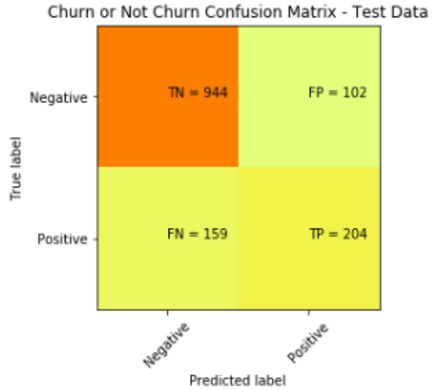
Below 3 attributes from the correlation matrix and try to visualize their relationship using scatterplot.





Logistic Regression:

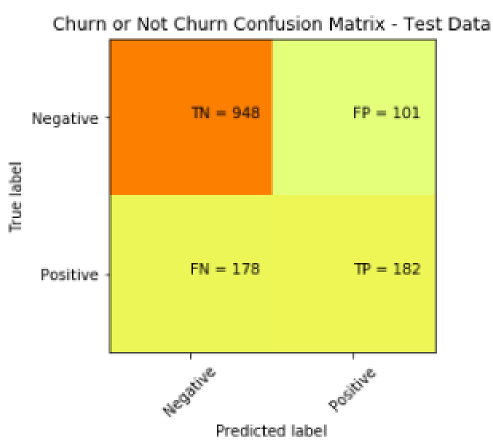
```
Accuracy :  0.8147622427253371
[[944 102]
 [159 204]]
```

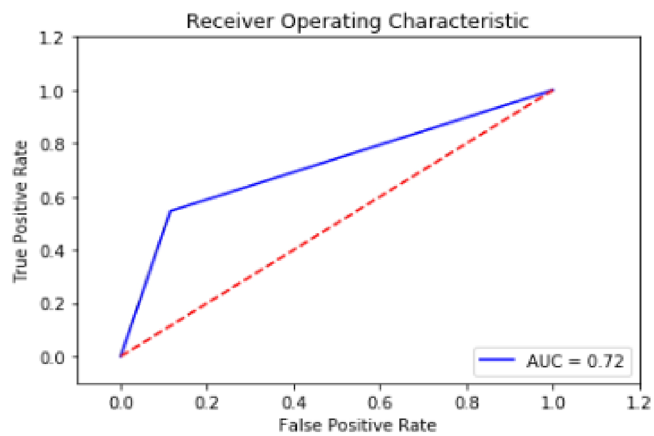



```
Report :               precision    recall  f1-score   support

          0       0.86      0.90      0.88      1046
          1       0.67      0.56      0.61       363

avg / total       0.81      0.81      0.81      1409
```

Random Forest:

```
Accuracy :  0.8019872249822569
[[948 101]
 [178 182]]
```

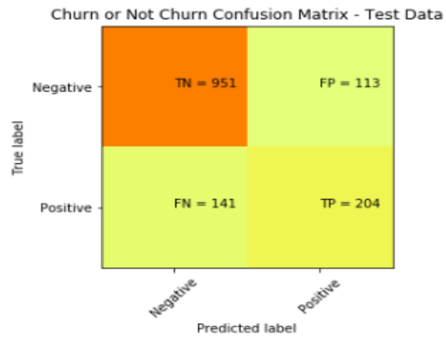


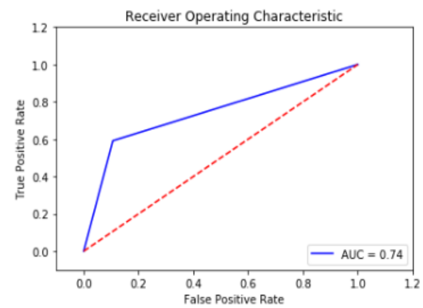



```
Report :               precision    recall  f1-score   support

          0       0.83      0.89      0.86      1002
          1       0.66      0.55      0.60       407

avg / total       0.78      0.79      0.78      1409
```

SVM - Linear C-1

```
0.8197303051809794
[[951 113]
 [141 204]]
```

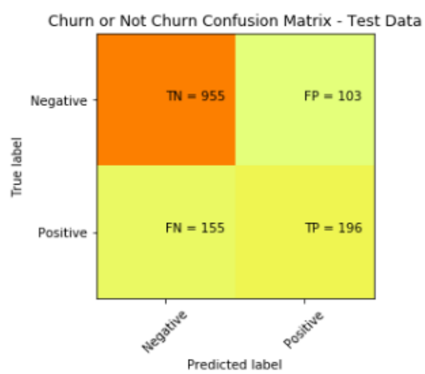


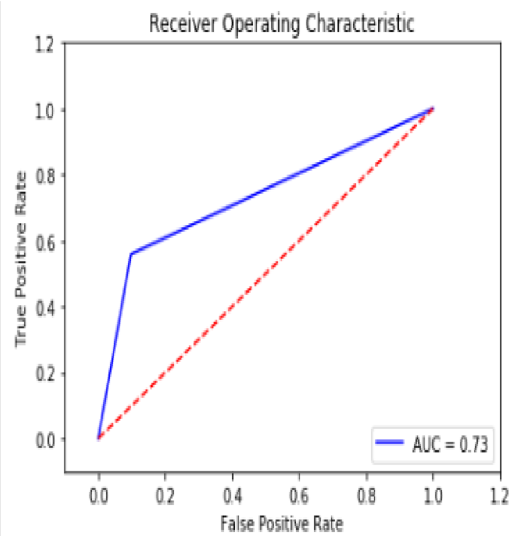



```
Report :              precision    recall  f1-score   support

          0       0.87      0.89      0.88      1064
          1       0.64      0.59      0.62       345

avg / total       0.82      0.82      0.82      1409
```

C=5
Before sampling

```
0.8168914123491838
[[955 103]
 [155 196]]
```

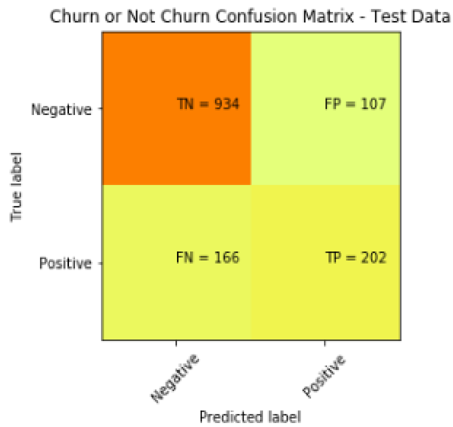


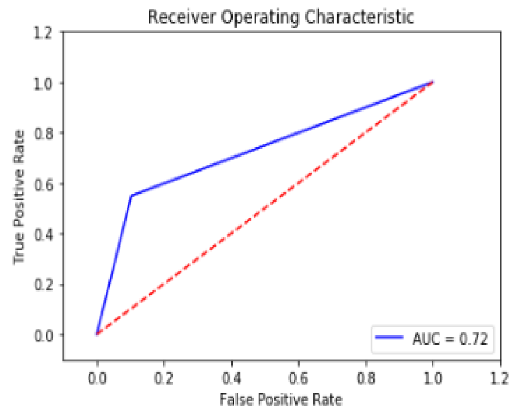



```
Report :              precision    recall  f1-score   support

          0       0.86      0.90      0.88      1058
          1       0.66      0.56      0.60       351

avg / total       0.81      0.82      0.81      1409
```

c=10

```
0.8062455642299503
[[934 107]
 [166 202]]
```

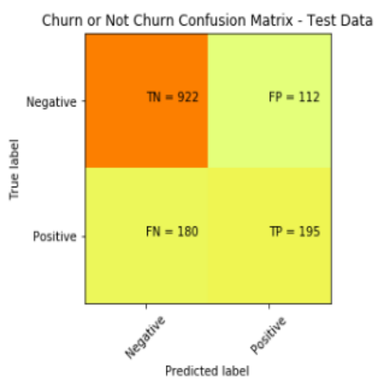


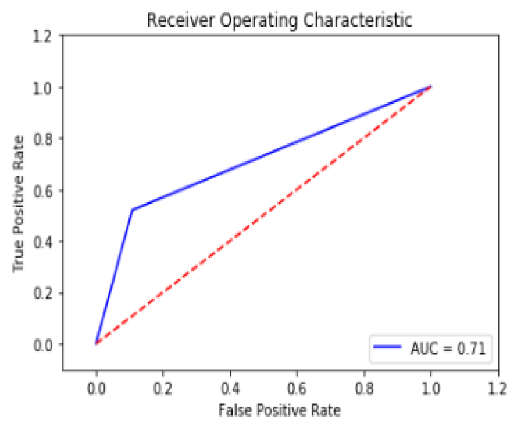



```
Report :              precision    recall  f1-score   support

          0       0.85      0.90      0.87      1041
          1       0.65      0.55      0.60       368

avg / total       0.80      0.81      0.80      1409
```

c=100

```
0.7927608232789212
[[922 112]
 [180 195]]
```






```
Report :              precision    recall  f1-score   support

          0       0.84      0.89      0.86      1034
          1       0.64      0.52      0.57       375

avg / total       0.78      0.79      0.79      1409
```

SVM Linear		
Tuning Parameter	**Accuracy**	**AUC-ROC score**
C=1	81.97%	74%
C=5	81.68%	73%
C=10	80.62%	72%
C=100	79.92%	71%

SVM RBF	
Tuning Parameters	**Resulted Labels**
C=1, Gamma =0.1	All Negatives
C=1, Gamma =0.01	All Negatives
C=1, Gamma =0.001	All Negatives
C=1, Gamma =0.0001	All Negatives
C=5, Gamma =0.1	All Negatives

C=5, Gamma =0.01	All Negatives
C=5, Gamma =0.001	All Negatives
C=5, Gamma =0.0001	All Negatives
C=10, Gamma =0.1	All Negatives
C=10, Gamma =0.01	All Negatives
C=10, Gamma =0.001	All Negatives
C=10, Gamma =0.0001	All Negatives

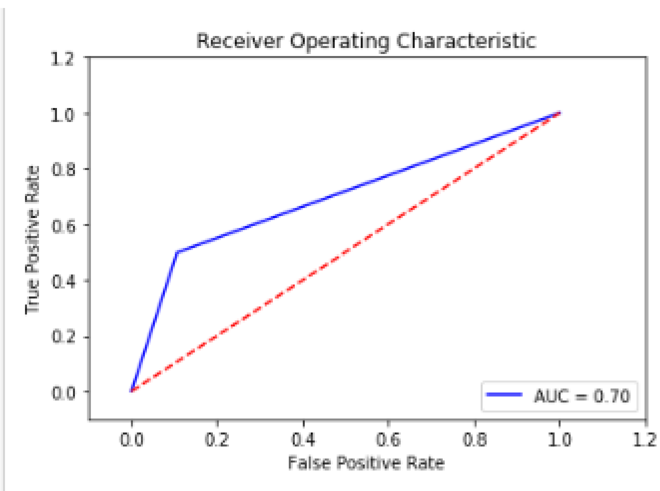
Cross Validation k=5

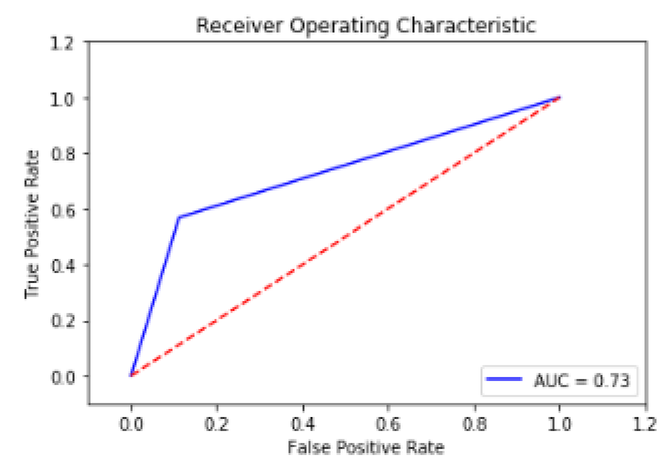
Logistic Regression

```
5-fold Cross Validation :
[0.73456352 0.54222853 0.85237757 0.74378992 0.43425729]
0.6614433661561401
```

Decision Tree

```
5-fold Cross Validation :
[0.73527324 0.27040454 0.81476224 0.63875089 0.27292111]
0.5464224048569886
```





Random Forest

```
5-fold Cross Validation :
[0.73598297 0.26046842 0.7707594  0.38679915 0.34896944]
0.5005958749293178
```

SVM

```
5-fold Cross Validation :
[0.73456352 0.52803407 0.85379702 0.74520937 0.42643923]
0.6576086413718693
```

K=10

Logistic Regression **Decision Tree**

```
10-fold Cross Validation :
[0.73900709 0.77588652 0.60851064 0.72056738 0.84801136 0.92045455
 0.81960227 0.84232955 0.60653409 0.58463727]
0.7465540718235287
```

```
10-fold Cross Validation :
[0.73475177 0.75602837 0.5929078  0.73617021 0.890625   0.88778409
 0.73153409 0.67045455 0.50710227 0.49075391]
0.6998112067834912
```

Random Forest **SVM**

```
10-fold Cross Validation :
[0.74893617 0.71914894 0.57021277 0.70496454 0.79545455 0.859375
 0.68465909 0.74005682 0.59659091 0.58748222]
0.7006880994045048
```

```
10-fold Cross Validation :
[0.74326241 0.7787234  0.60567376 0.73475177 0.859375   0.92755682
 0.8125     0.81534091 0.60795455 0.57752489]
0.746266351355937
```