

Telecom Customer Churn Prediction

TEAM MEMBERS

VINIL RAYALA, A20220246

ALISETTY KALYAN KUMAR, A20199542

GANDHI SAHIL, A20207207

HARSHIT AGARWAL, A20219377

KYLE MCCALLUM, A20128454

Table of Contents

1. Executive Summary	1
2. Scope of the project	2
3. Project Schedule	
GANTT Chart	4
4. Data Preparation	
Data Access	5
Data Consolidation	5
Data Cleaning	5
Data Transformation	6
Data Reduction	8
Data Dictionary	11
5. Analysis	
Model Selection	12
Data Splitting and sub sampling	13
Models Built	18
Model Assessment	24
6. Appendix	28

Executive Summary

We all know that the cost of retaining a customer is far less when compared to cost of acquiring a new customer. Churn occurs when the customer decides to switch to another company or service provider, also sometimes churn occurs when the customer shifts to a new place or unfortunate death. Here we are focusing on the first type of churn as it helps in determining the bond between the customers and the company. In short churn rate is the customer's turnover to the company in peer to peer network and, we can say that it is one of the best ways to measure success rate of the company.

In this analysis we will be working on the Telco Customer churn rate where the data is contains various factor such as customer's demographics with their usage of services. These factors will not only help us analyze the churn of their customers but also, we can segregate the customers and provide recommendation to the company as to how can they reduce the churn of their customers and increase their revenue.

Statement of Scope

Objectives:

The objective of this study is to understand the patterns and relationships involved in customer churn. By understanding the reasons customers leave for a competitor, customer churn can be reduced, thereby retaining business value. The sample dataset has 7043 rows or customers that will be generalized to the broader population of current Telco customers. Our main objective can be broken into the following sub-objectives:

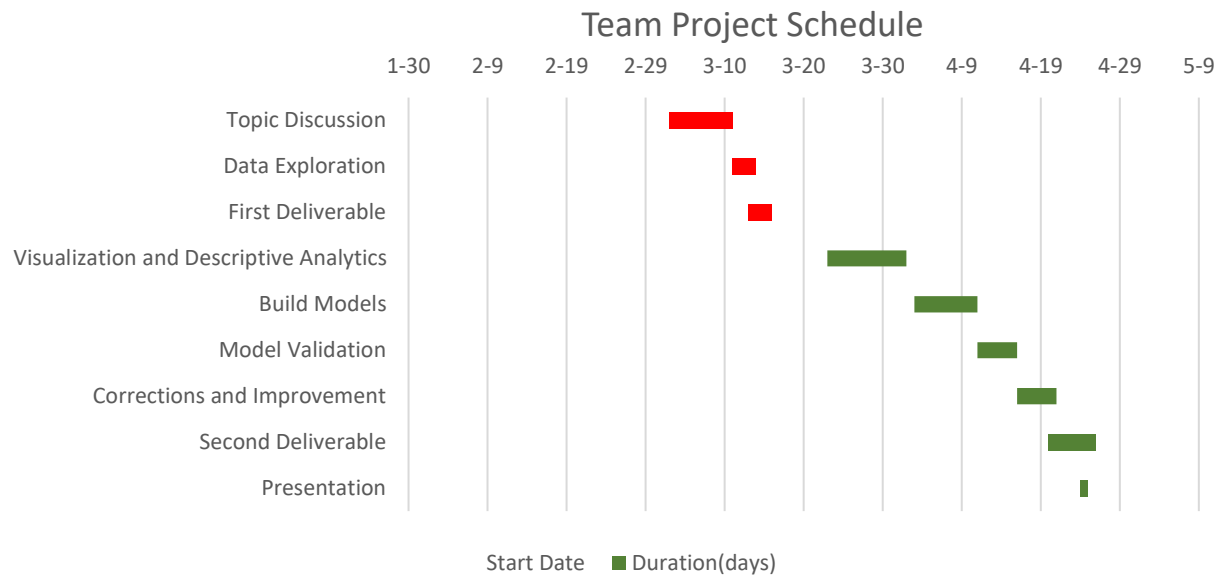
- Segment customers based upon:
 - Demographics and behaviour
 - Revenue information
 - Contract state
- Characterize customer profiles based upon segments
- Quantify customer churn rate
- Quantify customer churn risk
 - Patterns associated with risk
- Accurately predict customer churn
 - Plan for customer retention based upon churn analysis

After the segmentation of customers, the data can be analyzed to find patterns within natural groupings. Customer retention is vitally important from loyalty and business value perspectives. Customer retention is also easier than finding new clientele to add value to the business. Customer churn is directly related to customer satisfaction, service usage, and customer-related variables.

Variables:

- Target:
 - Churn
- Predictors:
 - Demographics or behaviour:
 - Gender
 - SeniorCitizen
 - Partner
 - Dependents
 - Revenue Information:
 - Tenure
 - MonthlyCharges
 - TotalCharges
 - PaymentMethod
 - PaperlessBilling
 - Contract State:
 - PhoneService
 - MultipleLines
 - InternetService
 - OnlineSecurity
 - Contract
 - OnlineBackup
 - DeviceProtection
 - StreamingMovies
 - StreamingTV

Project Schedule



The project is estimated to take about 7 weeks barring unforeseen circumstances in which case expedient decisions will be made. The team will meet physically about 5 times and would mostly communicate through the already created WhatsApp group chat, Zoom, and Skype video calls. All the team members have worked and collaborated on all the aspects on the project and prepared the deliverable. The team has adjusted to changes to remain fluid and made some re assignments for the project.

Data Preparation

Data Access

We have chosen a dataset from Kaggle platform which has data about telecom company customers. We have selected this source because customer data is private and generally won't be available on a public platform for scraping. This dataset is provided by IBM. The dataset is in the following link

<https://www.kaggle.com/blatchar/telco-customer-churn>

Data Consolidation

As the dataset contains all the variables required for our project in a single file. There is no need for data consolidation at this moment as we feel the variables in the data are enough for our analysis and as they are customer data, we can't combine with external sources.

Data Cleaning

As the dataset contains few missing values in some variables, we have replaced them accordingly based on the values of the other variables.

For example: We have replaced the missing values in the variable called 'TotalCharges' with 0 for customers who have tenure as 0 because they haven't paid yet for any month. We chose this imputation instead of mean imputation as this makes more sense in this particular scenario.

We looked for outliers in the data using box plots but there are no potential outliers.

If the customer has no internet service, then the records have “No_internet_Service” for variables like online security, device protection, etc., Here, “No_Internet_Service” itself is not a different level for those variables and we replaced it with “No” so that there won’t be irrelevant levels and those become binary variables. Similarly, “No_Phone_Service” level is dealt with relevant cases.

We have converted some of the variables into categorical accordingly as some variables are read as object type when we imported the dataset. “Senior Citizen Status' ' is read as an integer by default, we converted it to a binary variable.

We dummy coded Yes or No binary variables to 1 and 0 respectively as this helps for our modeling phase.

Data Transformation

This is a continuous process as the project progresses and we can foresee normalization and text transformation during model fitting. As of now, there are 3 numeric columns in our data: Tenure, Monthly Charges, Total charges

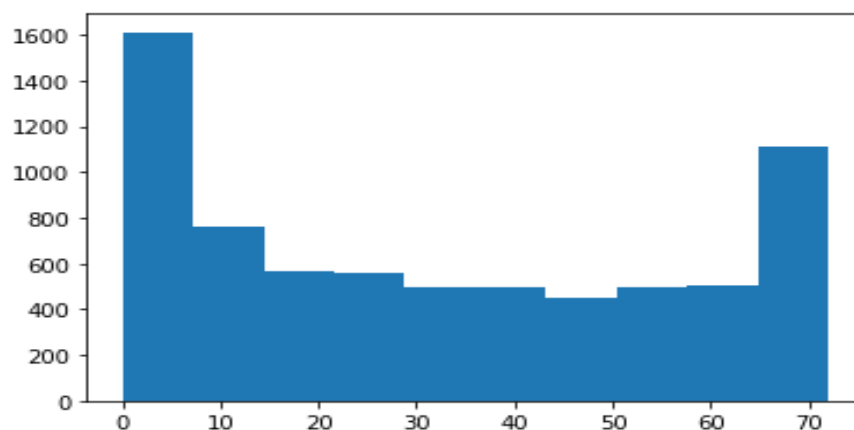


Fig 1.1 Histogram of Tenure

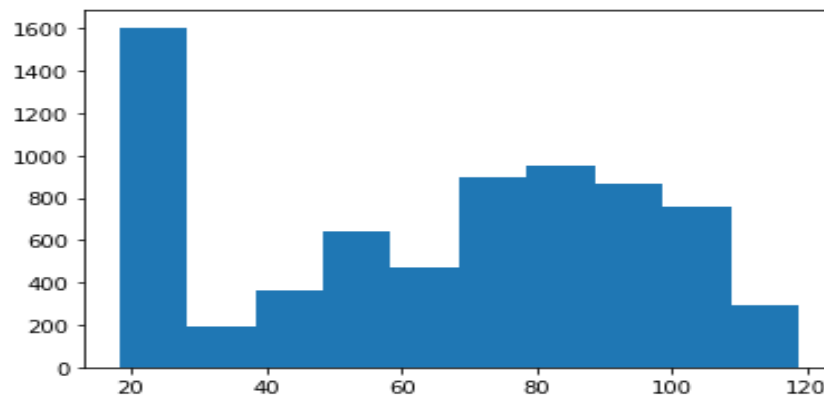


Fig 1.2 Histogram of Monthly Charges

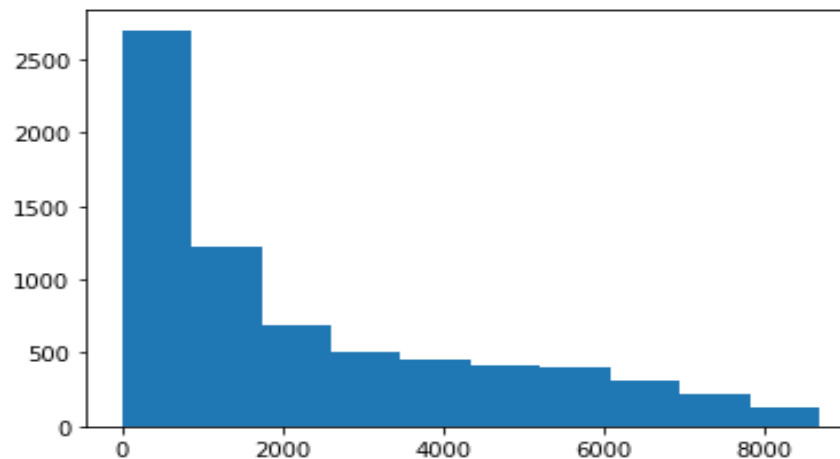


Fig 1.3 Histogram of Total Charges

Each of these variables doesn't look like they have a normal distribution. If we see Tenure and Charges, their units are different. So, we performed Standardization on these variables, so they come to similar terms. We performed Standard Scalar function using Sklearn Preprocessing because this is more relevant and useful before performing PCA. These transformations are not shaping transformations. We haven't performed shape-changing transformation like log or power at this point as interpretation of variables becomes difficult and we don't require high

transformation for algorithms like decision trees. We will again revisit these after the modelling phase.

Data Reduction

As the PCA works better for standardized data we have converted interval variables into a standardized format. Since PCA helps a lot in visualizing the data for machine learning applications, we have applied this technique to the final dataset obtained after performing data cleaning and data transformation.

Visualizing up to 3 dimensions is good while visualizations are best for 2 dimensions. The features are standardized to resemble standard normal distribution with mean = 0 and standard deviation = 1 using StandardScaler function in Python.

Principal Component Analysis results provide the components which explain the maximum variance. Thus, if the variables are not scaled, then the features which might vary less/more compared to other features would influence the overall components resulting in biased results.

We performed PCA for different numbers of components to see the variance explained by the components. The results have shown that the combination of Principal component 1 and Principal component 2 has explained 54% of the variance in the target, with 36.9 and 16.8 % respectively. The PC 3 explains 6.5% of the total variance while the components 4 to 10 explain approximately 2-3% total variance individually.

So, after running PCA, we have reduced the dataset from 28 features to 10 Principal components which explain 82% of the total variance in the target variable that is, CHURN in this case. It should

be noted that after performing the dimensionality reduction on the dataset, there is no such meaning associated with each of the principal components.

Plot showing the Visualization with Principal components 1 and 2 is as below. More principal components are not added while creating the plot since the visualizations are tough to interpret when the principal components are greater than 3.

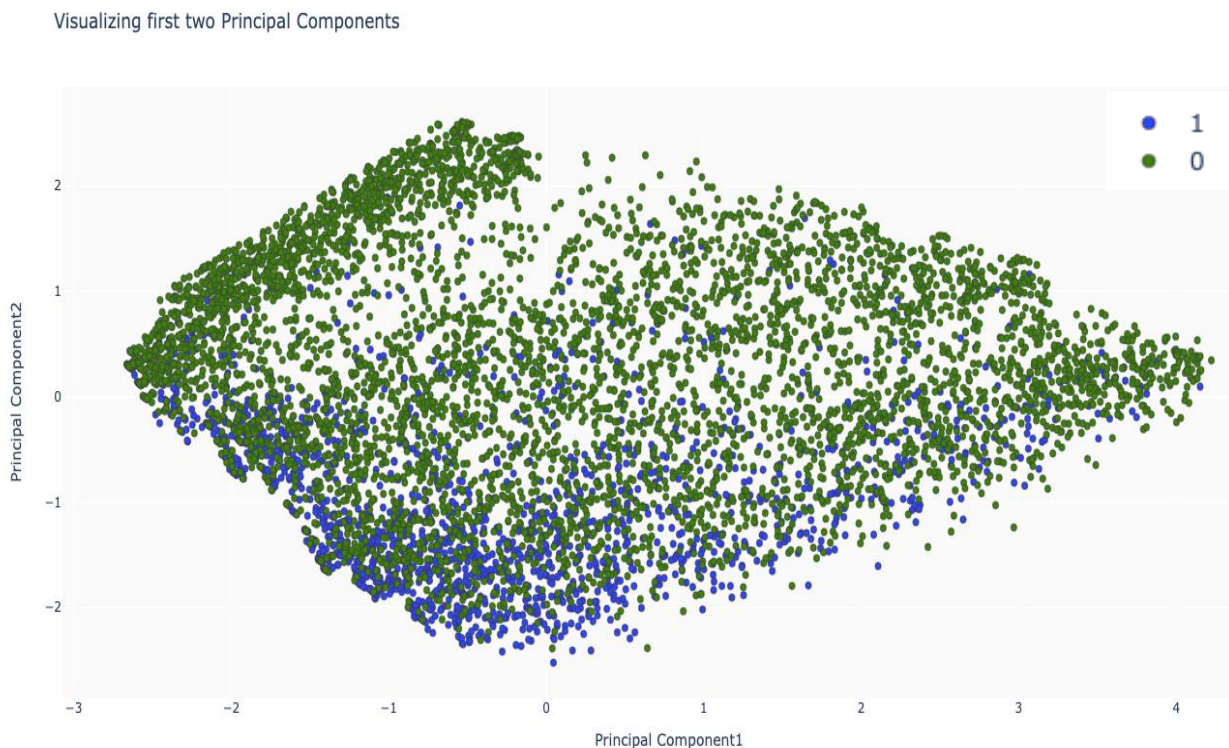


Fig 1.4 First two principal components

The visualization shows that the classes 0 and 1 of Target (Churn and not churn) are not well separated from each other indicating that these 2 principal components alone cannot explain most of the variance in the target variable as can be seen from the PCA results, which is the reason why we have chosen 10 principal components.

The below graph explains the amount of variance explained by the top 15 components

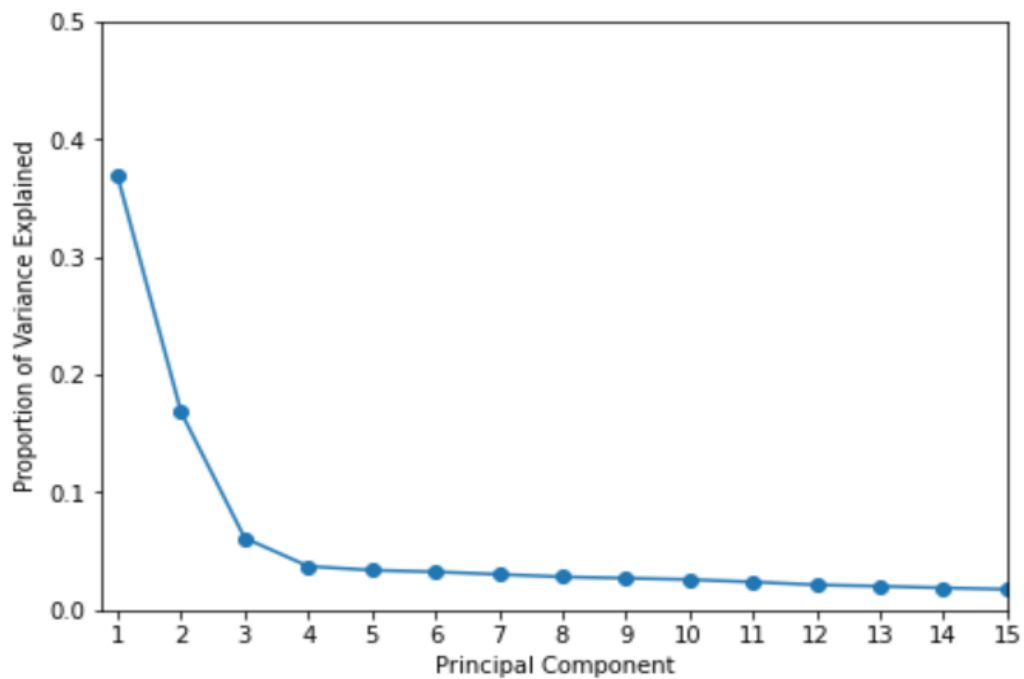


Fig 1.5 Variance explained by top 15 components

After the factor analysis, we have found that variables such as MonthlyCharges_T, Contract__Month-to-month, IS__DSL, Contract__One year, PM__Bank transfer (automatic) are most important.

Data Dictionary

Attribute	Type	Data Description
customerID	ID	Uniquely identifies each Telco customer.
gender	String	Indicates whether the person is Male or Female.
SeniorCitizen	Binary	States that if the person is a senior citizen then 1 else 0.
Partner	String	Values in this column are “Yes” or “No” stating whether the customer has a partner or not.
Dependents	String	Values in this column are “Yes” or “No” stating whether the customer has a dependent or not.
tenure	Integer	Numerical variable indicating the number of months the customer has been with the company.
PhoneService	String	Values in this column are “Yes” or “No” stating whether the customer has a phone service or not.
MultipleLines	String	Values in this column are “Yes”, “No” or “No phone service” stating whether the customer has multiple lines or not and if the multiple lines people have phone service or not.
InternetService	String	Indicates the internet service provider of the customer such as whether DSL, Fiber optic.
OnlineSecurity	String	Values in this column are “Yes”, “No” or “No internet service” stating whether the customer has online security or not and if the internet service is there for the customer or not.
OnlineBackup	String	Values in this column are “Yes”, “No” or “No internet service” stating whether the customer has online backup or not and if the internet service is there for the customer or not.
DeviceProtection	String	States whether the customer has a device protection or not.
TechSupport	String	Indicates in terms of “Yes” or “No” if the customer has a tech support or not.
StreamingTV	String	Indicates in terms of “Yes” or “No” if the customer streams TV not.
StreamingMovies	String	Indicates in terms of “Yes” or “No” if the customer streams movies not.
Contract	String	Indicates what type of contract the customer has with the company such as monthly or yearly.
PaperlessBilling	String	States whether the customer uses paperless billing or not.
PaymentMethod	String	Indicates what type of method id used by the customer to do their payment such as check, credit card, bank transfer.
MonthlyCharges	Float	States the amount of money charged to the customer on a monthly basis for the services used.
TotalCharges	Float	States the total bill amount charged to the customer based on their contract.
Churn	String	Variable indicating if the customer has churned or not.

Analysis

Modeling Technique Selection

Here, we have a target variable as categorical and a combination of categorical and continuous as independent variables. As the target variable is categorical, we have used different supervised classification algorithm to classify if a customer is about to Churn or Not. The different classification algorithms include decision trees, logistic regression and neural networks. The model helps the telecom companies to predict if customer is about to Churn and helps them to act accordingly. We have used logistic regression to explain how the odds of target variables changes with the change in the independent variables such as gender, Senior Citizen, Having Partner... As most of the independent variables are binary, a linear relationship is not possible. Hence, the assumptions of linear regression are not required for logistic regression model. There are two main assumptions in case of logistic regression which are observations to be independent of each other and little or no multicollinearity among the independent variables. The former assumption is satisfied in our case and the latter assumption also holds true to most part and the results are mentioned in the analysis part. We have used Decision Trees as the attractiveness is largely since it helps in representing the rules both English and SQL. This makes interpretation straightforward and simple. Also, they can be used for data exploration and knowing the variables importance with a powerful first step in model building. There are no assumptions for decision trees for our data. As the neural network can work on unstructured problems, we have used it as one of the algorithms to classify the Churn of customer using different non-linear variables. As we have already standardized the variables, we have satisfied neural net assumptions. In addition to that we looked at the binary response with logistic regression and decision trees.

Data Splitting and Sub sampling technique

SMOTE – Synthetic minority oversampling technique

We tend to see the proportion of data within the target variable. We see that 26% were Churn and the remaining 74% were Not Churn. Due to the imbalance of the data distribution we used the SMOTE method to do the oversampling of the Churn category in the data. It works by selecting examples that are close in the feature space, drawing a line between the examples in the feature space and drawing a new sample at a point along that line. By default, SMOTE creates instances of minor class in order to make same number of instances in both major and minor class in the training dataset. By creating equal instances in the data, we could not receive a good f1-score. Hence, we tried creating unequal instances. Out of different proportions created, 40% Churn and 60% Not Churn distribution of the data resulted in a best score of all the evaluation metrics.

Feature Reduction and Feature Selection techniques

Correlation is the measure of relationship among the variables. Higher the correlation stronger the relation between the variables and therefore it results in redundant usage of variables. Therefore, we have removed the highly correlated values from the dataset before any analysis. In order to do this, we must specify the threshold value of the correlation. We have specified 0.7 as the cutoff value initially as 1.0 indicates the variables are highly correlated. Taking 0.7 as the cut off value, we found that the variables ‘monthly charges’ and ‘tenure’ are having high relation and therefore we removed the ‘monthly charges’ variable from analysis. Further when the modeling was performed, we couldn’t achieve best result. So, we have changed the cut-off value to 0.8 and

observed that the variables 'total charges' and 'tenure' were strongly correlated resulting in removal of the variable 'total charges' from our analysis.

Recursive Feature Elimination Technique - RFE

It is a feature selection method that removes the weakest feature or variable from the model. RFE recursively removes the features one by one and build the model and calculates its accuracy. It has the capability to mix and match the attributes of the features and add to the expectation on the target variable. We initially had 26 independent variables and 1 target variable. The recursive feature selection method has selected 10 important features. The below is the screenshot of the variables selected.

```
X_train_lr.columns  
  
Index(['PhoneService', 'OnlineSecurity', 'Contract__One year',  
      'Contract__Two year', 'PM__Bank transfer (automatic)',  
      'PM__Credit card (automatic)', 'PM__Electronic check',  
      'PM__Mailed check', 'tenure_T', 'MonthlyCharges_T'],  
      dtype='object')
```

Fig 2.1 Variables selected from feature elimination method

A common way to get data for classifiers is to split the available data into two sets, a training set and a test set. The classification model is built on the training set and is applied to the test set. The test set has never been seen by the model so the resulting performance will be a good guide to what will be seen when the model is applied to unseen data.

Whenever there is large data available and we need to tune the hyper-parameters of the models, then 3 way split of data into training, validation, testing is useful where the models are trained on training data, compared and tuned on validation data and finally scored the test data for generalization. In our case, as the data is not very large and the problem of class imbalance is there, we chose to do 70-30 split where 70% of the data is used for training of models and 30% of the data is used for testing. We chose this instead of 60-40 split to avoid underfitting of our models due to class imbalance issue. Very often, the proportion chosen is 70% for the training set and 30% for the test. The idea is that more training data is a good thing because it makes the classification model better whilst more test data makes the error estimate more accurate. Hence, we have chosen to move with the industry standard of splitting the data

After splitting the data, we have analyzed the distribution of different variables.

Training Data Distribution:

	gender	SeniorCitizen	Partner	Dependents	PhoneService	MultipleLines	OnlineSecurity	OnlineBackup	DeviceProtection	TechSupport
count	6023	6023	6023	6023	6023	6023	6023	6023	6023	6023
unique	2	2	2	2	2	2	2	2	2	2
top	1	0	0	0	1	0	0	0	0	0
freq	3029	5038	3294	4426	5450	3447	4474	4081	4086	4490

	StreamingMovies	PaperlessBilling	IS_DSL	IS_Fiber optic	IS_No	Contract_Month-to-month	Contract_One year	Contract_Two year	PM_Bank transfer (automatic)
count	6023	6023	6023	6023	6023	6023	6023	6023	6023
unique	2	2	2	2	2	2	2	2	2
top	0	1	0	0	0	1	0	0	0
freq	3635	3852	4054	3090	4902	3725	4893	4858	4826

	PM_Electronic check	PM_Mailed check
count	6023	6023
unique	2	2
top	0	0
freq	3687	4716

Fig 2.2 Distribution of categorical data in training dataset

	tenure_T	MonthlyCharges_T
count	6023.000000	6023.000000
mean	-0.120609	0.057756
std	0.997725	0.970643
min	-1.318165	-1.527580
25%	-1.073843	-0.666243
50%	-0.340876	0.290972
75%	0.799294	0.847401
max	1.613701	1.791029

Fig 2.3 Distribution of continuous data in training dataset

Test Data Distribution:

	gender	SeniorCitizen	Partner	Dependents	PhoneService	MultipleLines	OnlineSecurity	OnlineBackup	DeviceProtection	TechSupport
count	2113	2113	2113	2113	2113	2113	2113	2113	2113	2113
unique	2	2	2	2	2	2	2	2	2	2
top	0	0	0	0	1	0	0	0	0	0
freq	1089	1758	1114	1487	1908	1221	1527	1390	1383	1480

	StreamingMovies	PaperlessBilling	IS_DSL	IS_Fiber optic	IS_No	Contract_Month-to-month	Contract_One year	Contract_Two year	PM_Bank transfer (automatic)
count	2113	2113	2113	2113	2113	2113	2113	2113	2113
unique	2	2	2	2	2	2	2	2	2
top	0	1	0	0	0	1	0	0	0
freq	1297	1214	1398	1189	1639	1154	1692	1575	1643

	PM_Electronic check	PM_Mailed check
count	2113	2113
unique	2	2
top	0	0
freq	1415	1630

Fig 2.4 Distribution of categorical data in testing dataset

	tenure_T	MonthlyCharges_T
count	2113.000000	2113.000000
mean	-0.002857	-0.009409
std	1.005818	1.013008
min	-1.318165	-1.545860
25%	-0.951682	-1.040674
50%	-0.137274	0.184071
75%	0.921455	0.832172
max	1.613701	1.794352

Fig 2.5 Distribution of continuous data in testing dataset

The above figures depict the distribution of the different categorical and continuous variables in the test and train datasets. We could observe that there is no significant difference between variables across training and testing data. The distribution of data is reasonably good across training and testing.

Target Variable Distribution:

```
y_train.describe()
count      6023
unique       2
top      Not Churn
freq       3614
dtype: object

y_test.describe()
Churn
count      2113
unique       2
top      Not Churn
freq       1560
```

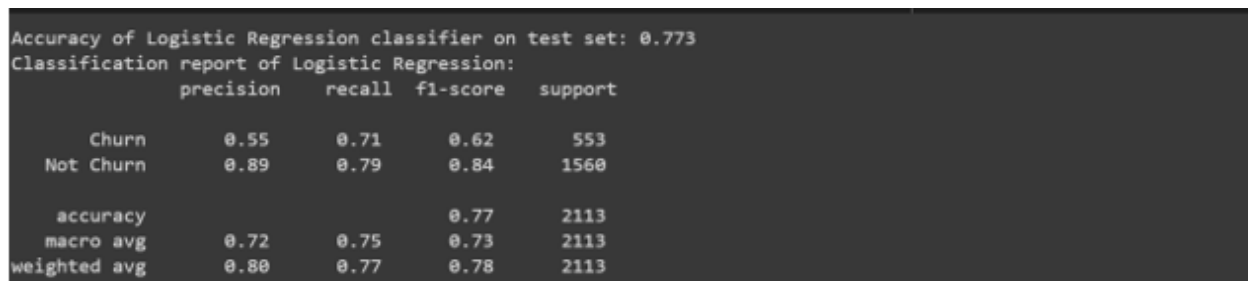
Fig 2.6 Distribution of target in training and testing dataset

Across target variable, there is difference in ratio of churn to non-churn cases across training and testing data. This is due to the SMOTE, we have performed to improve the cases of churn in training data. As we need to score on original data, the pattern in test data seems to be similar to initial full data set before split. This difference in our case across training and testing data is valid and is useful in not underfitting the data and to check the model performance accurately on the original data.

Built Models

Logistic Regression

Regression can be used to fit both linear and logistic regression model. In this the logistic regression attempts to predict the probability that a binary or ordinal target will acquire the outcome. In this model we can also determine the input parameters that help in the modeling such as the selection criteria (forward, stepwise, backward). Also, we can set the significance level for both entry and exit. The below screenshot presents you the results of the logistic regression model with Churn as dependent variable and 10 independent variables selected from recursive feature selection method.



```
Accuracy of Logistic Regression classifier on test set: 0.773
Classification report of Logistic Regression:
      precision    recall  f1-score   support

   Churn         0.55      0.71      0.62        553
  Not Churn         0.89      0.79      0.84       1560

 accuracy         0.77         0.77       2113
 macro avg         0.72         0.75      0.73       2113
weighted avg         0.80         0.77      0.78       2113
```

Fig 2.7 Logistic Regression evaluation metrics

The accuracy of the model is 0.773. Here we tend not to use the accuracy to evaluate the model performance since the data given is imbalanced and we get a biased result. Therefore, we calculate the precision and recall from the confusion matrix to obtain the f1-score. There is a trade of between precision and recall with one going up the other slides down. As the cost of both, predicting a Churn customer as Not Churn or Not Churn customer as Churn is high, we have chosen f1-score (that is the harmonic mean of precision and recall) as the evaluation metric for the model performance. Here the above model gives us a f1-score of 0.73.

F1-score of Churn is 0.62

F1-score of Not Churn is 0.84

Total model avg F1-score is 0.73

The below figure explains the confusion matrix diagrammatically.

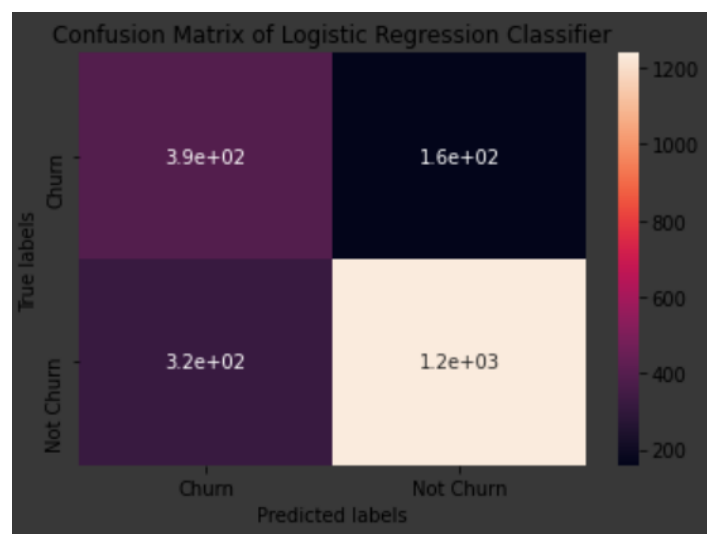


Fig 2.8 Confusion Matrix on test data

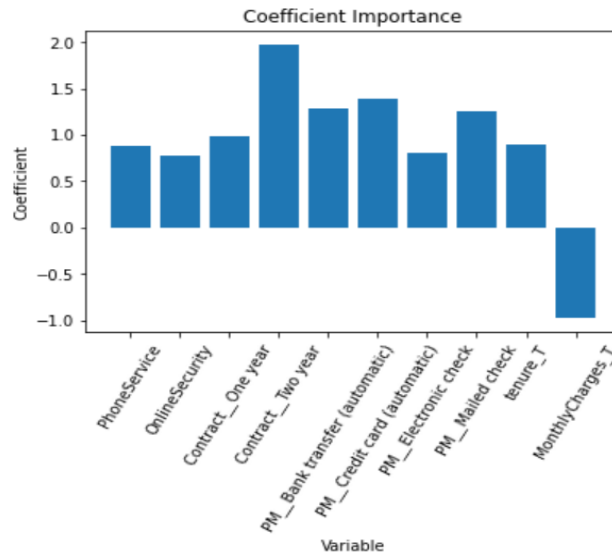


Fig 2.9 Logistic Regression Coefficients

So, this explains that that if a customer has phone service then he/she is more likely to Churn, if he has online security then the customer chances of Churn would increase. Similarly, with the presence of that one-year contract, two-year contract, automatic bank transfer or automatic credit payment method, Electronic or mailed check payment method increases the probability of customer to Churn. But only for the variable ‘monthly charges’, the customer would Not Churn with the increase in monthly charges.

Decision Tree

Decision Tree is a supervised learning algorithm used mainly for solving classification problems. It represents the segmentation of the data that is created by applying a series of simple rules. Using this the hierarchy of the tree is built resulting in nodes and leaf. For each leaf a decision is made and applied to all observation in the leaf. Every node in the tree also gives the probability of the classification. It makes it easier for the analyst to see the rules and make business decision on the prediction value.

Classification report of Decision Tree classifier:				
	precision	recall	f1-score	support
Churn	0.53	0.69	0.60	553
Not Churn	0.88	0.78	0.83	1560
accuracy			0.76	2113
macro avg	0.70	0.74	0.71	2113
weighted avg	0.79	0.76	0.77	2113

Fig 2.10 Decision Tree evaluation metrics

The above screenshot presents you with the model parameter details. The accuracy of the model is 0.745 with a f1-score of 0.71. Although this model gives us the best true positives among all the models but the overall f1-score of lower. This model performs worse than the logistic regression model.

F1-score of Churn is 0.60

F1-score of Not Churn is 0.81

Total model F1-score is 0.71

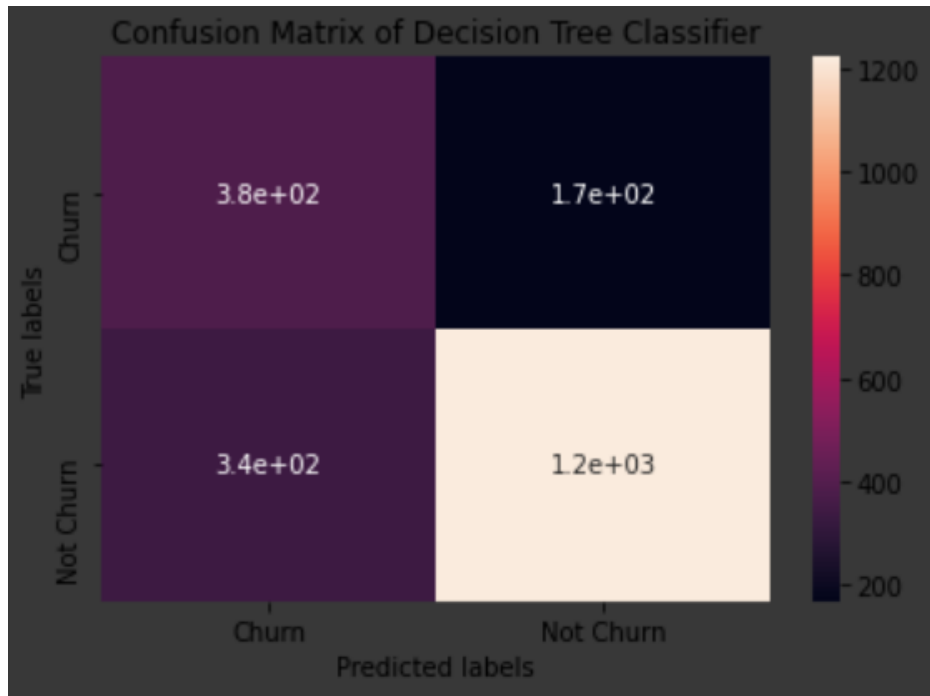


Fig 2.11 Confusion Matrix on test data

The below diagram visualizes the variables importance.

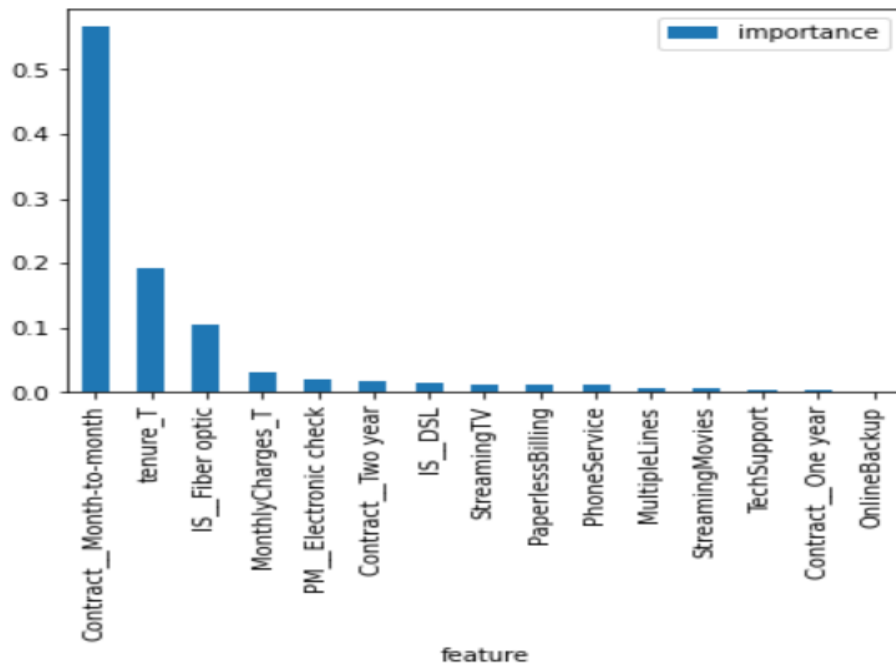


Fig 2.12 Decision Tree Variable Importance

Out of 23 variables in the training dataset the decision tree has selected 15 variables as important. The below figure gives a glimpse of the decision tree if-then rules. We can say that if a customer is having month-to-month contract, then he/she is more likely to Not Churn. In turn, if he/she has fiber optic then that customer is likely to not churn.

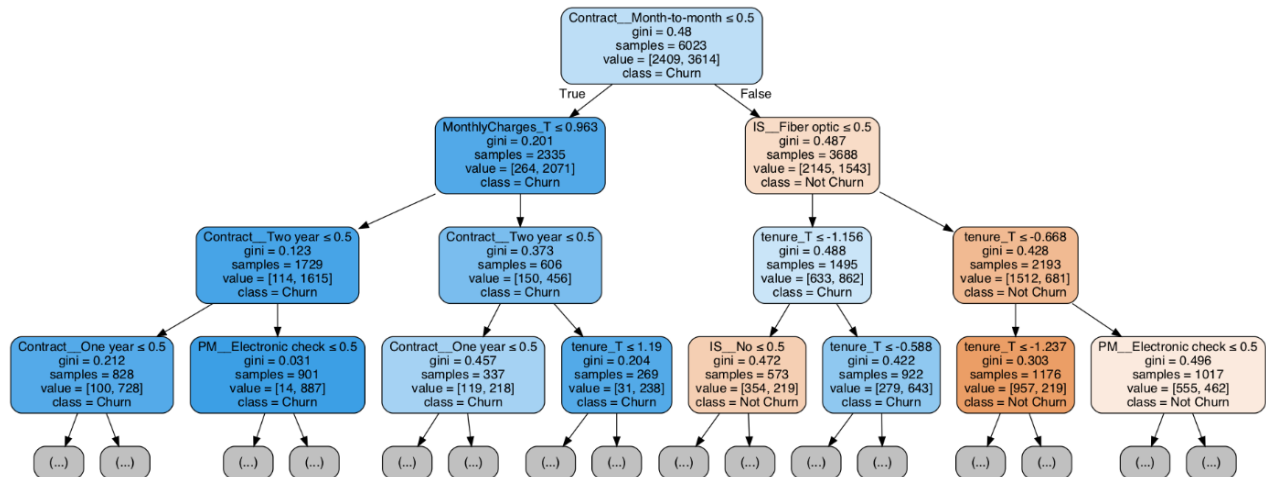


Fig 2.13 Decision Tree Rules

Neural Network

Neural networks are modeled after the human neuron system designed to recognize patterns. We can also say that neural networks can be trained to produced supervised outputs given a set of inputs. It helps in classifying the data and develop a predictive model using hidden neurons and activation function. It contains a hidden layer that takes input weights and activation function to produce the output.

Classification report of Neural Network:				
	precision	recall	f1-score	support
Churn	0.57	0.68	0.62	553
Not Churn	0.88	0.82	0.85	1560
accuracy			0.78	2113
macro avg	0.72	0.75	0.73	2113
weighted avg	0.80	0.78	0.79	2113

Fig 2.14 Neural Net evaluation metrics

From the above output of Neural Net, we can say that the accuracy of the model is 0.79 with an f1-score of 0.74. Since this is a black box, we cannot explain the whole process as it uses weights and 'RELU' activation function for predicting the results. This neural model consists of two hidden layers with 100 neurons each.

F1-score of Churn is 0.62

F1-score of Not Churn is 0.85

Total model F1-score is 0.79

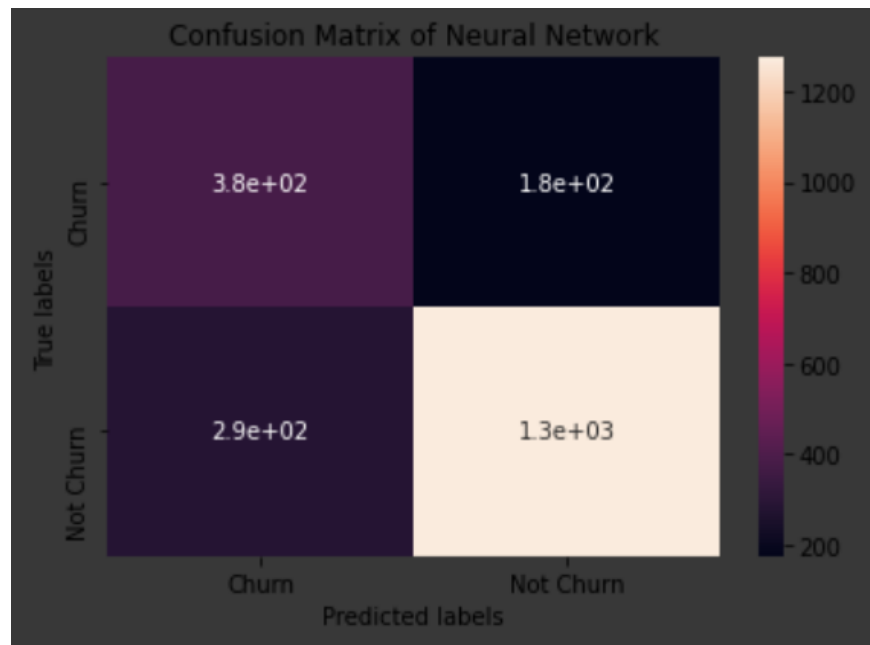


Fig 2.15 Confusion Matrix on test data

Model Assessment

Here we have chosen the f1-score for assessment as the prediction of both Churn and Not Churn is important for any of the telecom company. We haven't used the model accuracy to evaluate the model performance since the data given is imbalanced and we would be ended up with biased result. Therefore, we calculate the precision and recall from the confusion matrix to obtain the F1-score.

Accuracy = Correct Predictions / Total Predictions

Accuracy = (TP + TN) / (TP + FN + FP + TN)

When the class appropriation is somewhat skewed, accuracy can even now be a valuable measurement. When the skewness in the class appropriations are serious, accuracy can turn into a

problematic proportion of model execution. Accomplishing 90 percent characterization accuracy, or even 99 percent arrangement accuracy, might be insignificant on an imbalanced grouping issue.

$$\text{Precision} = \text{TP} / (\text{TP} + \text{FP})$$

$$\text{Recall} = \text{TP} / (\text{TP} + \text{FN})$$

*TP = True Positive

*TN = True Negative

*FP = False Positive

*FN = False Negative

There is a trade off between precision and recall with one going up the other slides down. As the cost of both, predicting a Churn customer as Not Churn or Not Churn customer as Churn is high, we have chosen F1-score (harmonic mean of precision and recall) as the evaluation metric of the model performance.

The **formula** for the **F1 score** is: $F1 = 2 * (\text{precision} * \text{recall}) / (\text{precision} + \text{recall})$

Strength of the models:

Logistic Regression: It is computationally inexpensive, and the results are easy to interpret.

Decision Tree: Missing values in the dataset likewise doesn't influence the way toward building decision tree to any impressive degree. Its helps to understand the rules easily and implement them as they are in the if-then kind of rules

Neural Network: They address one of the most testing issues in AI: working out the correct features for an issue

Weakness of the models:

Logistic Regression: It cannot perform feature selection therefore we had to use a different feature selection technique. Here we have used the RFE feature selection technique. If some of the assumptions are violated, then the results could be very biased.

Decision Tree: It has a property of low bias and high variance so sometimes there is a possibility that it might perform well on training dataset. But in the testing data set it might not perform the same due to the overfitting of the data.

Neural Network: It is a black box method due to which we cannot assess the process and the method it performs within the model. Also, they are computationally expensive to train.

ROC Curve

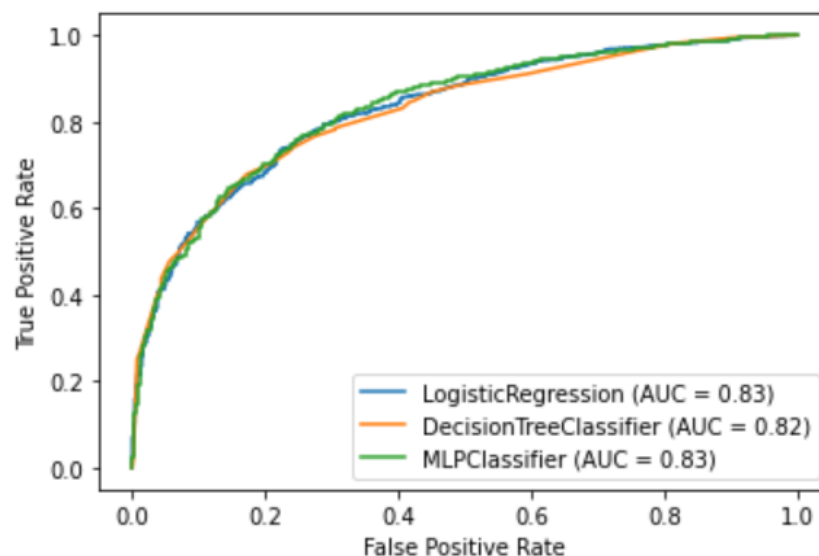


Fig 2.16 Comparison of three models ROC curves

ROC stand for receiver operating characteristic curve which uses a measure called AUC (area under the curve) which determines the accuracy of quantitative tests. In general AUC greater than 0.8 is considered good. In ROC curve we plot the true positive verses true negative of the models. Here

we have compared logistic regression , decision tree and MLP(Multilayer perceptron) classifier. We see that the logistic and MLP classifier performs the best among the three models with a value of 0.83.

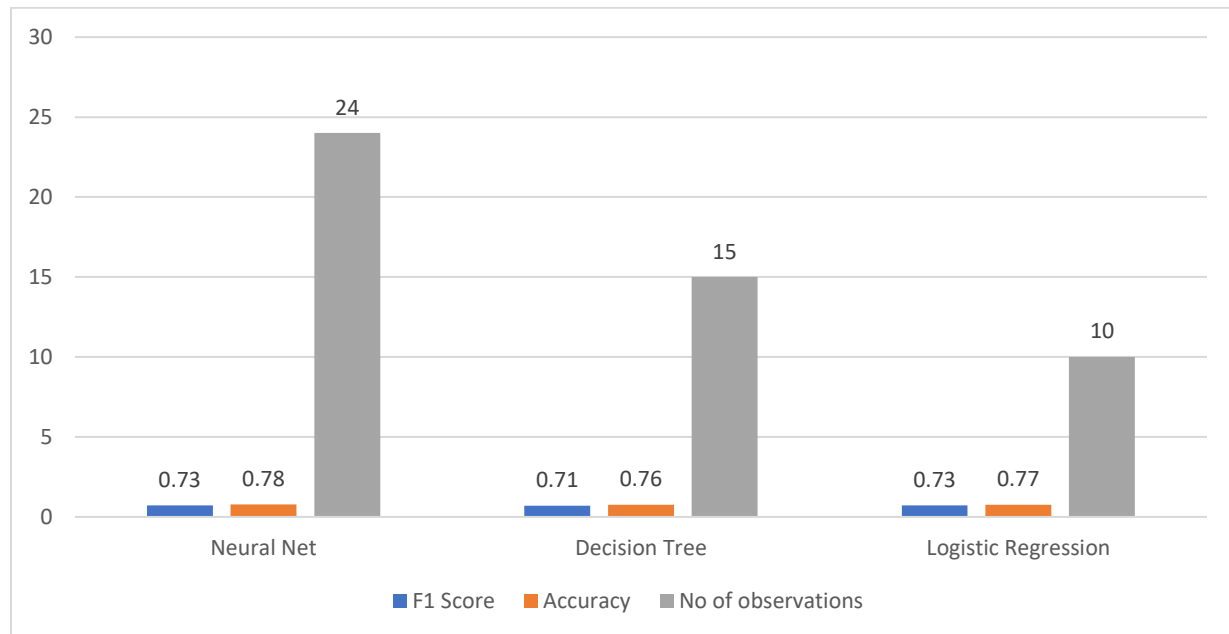


Fig 2.17 Comparison of three models

The above figure shows the comparison between different models built to predict if a customer is going to Churn or Not Churn. Out of all the models, we have chosen logistic regression as the best model because it has the highest F1-score which means it predicts both the categories of customers (i.e., Churn and Not Churn) correctly compared to other models. And, the logistic regression model has used only 10 independent variables to predict the Churn of a customer with an accuracy of 77%. Thus, these models can be employed by telecom companies in order to predict if a customer is about to Churn or Not

Appendix

```
#Importing different libraries
```

```
import pandas as pd
```

```
import numpy as np
```

```
import os
```

```
import matplotlib.pyplot as plt
```

```
from sklearn import preprocessing
```

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

```
from sklearn.decomposition import PCA as pca
```

```
from factor_analyzer import FactorAnalyzer
```

```
from sklearn.feature_selection import VarianceThreshold, RFE
```

```
from sklearn.linear_model import LogisticRegression
```

```
from sklearn import metrics
```

```
from sklearn.metrics import classification_report, accuracy_score, confusion_matrix
```

```
from sklearn.utils import shuffle
```

```
from sklearn.metrics import roc_auc_score, roc_curve, plot_roc_curve
```

```
from sklearn.feature_extraction.image import grid_to_graph
```

```
from sklearn import tree
```

```
import seaborn as sns
```

```
from sklearn.neural_network import MLPClassifier
```

```
from imblearn.over_sampling import SMOTENC
```

```
#For displaying the tree
```

```

from sklearn.tree import export_graphviz

from sklearn.externals.six import StringIO

from IPython.display import Image, display

import pydotplus

import plotly.graph_objs as go

import plotly.io as pio

import plotly


#Importing the data

data= pd.read_csv('/content/drive/My Drive/Company Project/churn_data.csv')


#Converting to respective datatypes

data['gender'] = data['gender'].astype('category', copy = False)

data['Partner'] = data['Partner'].astype('category', copy = False)

data['Dependents'] = data['Dependents'].astype('category', copy = False)

data['PhoneService'] = data['PhoneService'].astype('category', copy = False)

data['MultipleLines'] = data['MultipleLines'].astype('category', copy = False)

data['InternetService'] = data['InternetService'].astype('category', copy = False)

data['OnlineSecurity'] = data['OnlineSecurity'].astype('category', copy = False)

data['OnlineBackup'] = data['OnlineBackup'].astype('category', copy = False)

data['DeviceProtection'] = data['DeviceProtection'].astype('category', copy = False)

data['TechSupport'] = data['TechSupport'].astype('category', copy = False)

data['StreamingTV'] = data['StreamingTV'].astype('category', copy = False)

```



```
data['StreamingMovies'] = data['StreamingMovies'].astype('category', copy = False)
```

```
data['Contract'] = data['Contract'].astype('category', copy = False)
```

```
data['PaperlessBilling'] = data['PaperlessBilling'].astype('category', copy = False)
```

```
data['PaymentMethod'] = data['PaymentMethod'].astype('category', copy = False)
```

```
data['Churn'] = data['Churn'].astype('category', copy = False)
```

```
data['TotalCharges'] = data['TotalCharges'].apply(pd.to_numeric, errors='coerce')
```

```
data['InternetService'].unique()
```

```
#Histogram of Tenure
```

```
plt.hist(data['tenure'])
```

```
#Boxplot of Tenure
```

```
plt.boxplot(data['tenure'])
```

```
#Replacing missing values in the variables total charges to 0
```

```
data['TotalCharges'].fillna(0, inplace=True)
```

```
plt.hist(data['TotalCharges'])
```

```
#Small modifications in the data
```

```
data[data.MultipleLines == 'No phone service']
```

```
data.loc[data['MultipleLines'] == 'No phone service', 'MultipleLines'] = 'No'
```

```
data.loc[data['OnlineSecurity'] == 'No internet service', 'OnlineSecurity'] = 'No'
```

```

data.loc[data['OnlineBackup'] == 'No internet service', 'OnlineBackup'] = 'No'
data.loc[data['DeviceProtection'] == 'No internet service', 'DeviceProtection'] = 'No'
data.loc[data['TechSupport'] == 'No internet service', 'TechSupport'] = 'No'
data.loc[data['StreamingTV'] == 'No internet service', 'StreamingTV'] = 'No'
data.loc[data['StreamingMovies'] == 'No internet service', 'StreamingMovies'] = 'No'
data.sort_values(by='tenure')

```

#Binary coding

```

data.gender.replace(['Male','Female'],[0,1], inplace=True)

binary_cols = ['Partner','Dependents','PhoneService', 'MultipleLines', 'OnlineSecurity',
               'OnlineBackup', 'DeviceProtection', 'TechSupport', 'StreamingTV',
               'StreamingMovies', 'PaperlessBilling']

for i in binary_cols:
    data[i] = data[i].replace({'No': 0, 'Yes':1})

```

```

data.Churn.replace(['No','Yes'], ['Not Churn','Churn'],inplace=True)

data['Churn'].value_counts(normalize=True)

```

#Dummy variable creation

```

data_dummy1 = pd.get_dummies(data['InternetService'], prefix='IS_')

data_dummy1.head()

data = data.join(data_dummy1)

```

```

data_dummy2 = pd.get_dummies(data['Contract'], prefix='Contract_')
data_dummy2.head()
data = data.join(data_dummy2)

data_dummy3 = pd.get_dummies(data['PaymentMethod'], prefix='PM_')
data_dummy3.head()
data = data.join(data_dummy3)
data.dtypes

#Variable standization

data1 = data.loc[:,['tenure','MonthlyCharges','TotalCharges']]
scaler = preprocessing.StandardScaler()
transformed = scaler.fit_transform(data1)
transformed = pd.DataFrame(transformed)
data = data.join(transformed)

data = data.rename(columns={0: 'tenure_T', 1: 'MonthlyCharges_T', 2: 'TotalCharges_T'})

data_final = data.copy()
data_final = data_final.drop(['InternetService', 'Contract', 'PaymentMethod', 'tenure',
'MonthlyCharges', 'TotalCharges', 'customerID'], axis=1)
data_final.columns

#Moving the target variable to the end

```

```

cols = list(data_final.columns.values) #List of all of the columns

cols.pop(cols.index('Churn')) #Remove price from list

data_final = data_final[cols+['Churn']]


#Obtain eigenvalues

data_final1 = data_final.drop(['Churn'], axis = 1)

pca_result = pca(n_components=5).fit(data_final1)

pca_result.explained_variance_


#Components from the PCA

pca_result.components_.T * np.sqrt(pca_result.explained_variance_)


#Variance explained by PCA

plt.figure(figsize=(7,5))

plt.plot([1,2,3,4,5], pca_result.explained_variance_ratio_, '-o')

plt.ylabel('Proportion of Variance Explained')

plt.xlabel('Principal Component')

plt.xlim(0.75,4.25)

plt.ylim(0,0.5)

plt.xticks([1,2,3,4,5])


#Factor Analysis

fa = FactorAnalyzer()

```

```

x1 = data_final1.astype(object)

fa.analyze(x1, 3, rotation='varimax')

fa.loadings

x1['tenure_T'] = x1['tenure_T'].astype(float)

x1['MonthlyCharges_T'] = x1['MonthlyCharges_T'].astype(float)

x1['TotalCharges_T'] = x1['TotalCharges_T'].astype(float)

x1.isnull().sum()

x1.dtypes

x1.iloc[:,0:23] = x1.iloc[:,0:23].astype(int)

np.isfinite(x1).sum()


fa.analyze(x1, 5, rotation='varimax')

fa.loadings


# Removing Correlated features

corr_features = set()

# Separating X and y

X = data_final.drop('Churn', axis=1)

y = data_final[['Churn']]

corr_matrix = X.corr()


for i in range(len(corr_matrix.columns)):

    for j in range(i):

```

```

if abs(corr_matrix.iloc[i, j]) > 0.8:
    colname = corr_matrix.columns[i]
    corr_features.add(colname)
    if colname in data_final.columns:
        del data_final[colname]
    if colname in X.columns:
        del X[colname]

y = y.astype('category', copy = False)
X.iloc[:,0:23] = X.iloc[:,0:23].astype('category', copy = False)

#SMOTE sampling
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)

target = ['Churn','Not Churn']
freq = [2409,3614]
sampling = dict(zip(target,freq))
print(sampling)

```

```

smt = SMOTENC(random_state=42,categorical_features=[0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,
18,19,20,21,22,23], sampling_strategy=sampling)

X_train1, y_train1 = smt.fit_resample(X_train, y_train)

X_train = pd.DataFrame(X_train1, columns=X_train.columns)

y_train = pd.Series(y_train1)


#Converting objects to categorical variables

X_train['gender'] = X_train['gender'].astype('category', copy = False)

X_train.iloc[:,0:23] = X_train.iloc[:,0:23].astype('category')

X_test.iloc[:,0:23] = X_test.iloc[:,0:23].astype('category', copy = False)


values, counts = np.unique(y_train, return_counts=True)

counts


# Performing Recursive Feature elimination

logreg = LogisticRegression()

rfe = RFE(logreg, 10, step=1)

rfe = rfe.fit(X_train, y_train.values.ravel())

print(rfe.support_)

print(rfe.ranking_)

features_selected = rfe.get_support(1) #the most important features

X_train_lr = X_train[X_train.columns[features_selected]] # final features`

```

```
X_test_lr = X_test[X_test.columns[features_selected]]
```

```
X_train_lr.columns
```

```
# Logistic Regression Classifier model
```

```
logreg = LogisticRegression()
```

```
logreg.fit(X_train_lr, y_train)
```

```
y_pred = logreg.predict(X_test_lr)
```

```
#Confusion matrix
```

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

```
print(confusion_matrix_lr)
```

```
#Evaluation Metrics
```

```
print('Accuracy of Logistic Regression classifier on test set: {:.3f}'.format(accuracy_score(y_test,  
y_pred)))
```

```
print('Classification report of Logistic Regression:\n', classification_report(y_test, y_pred))
```

```
clf = LogisticRegression().fit(X_train_lr, y_train)
```

```
print(clf.coef_, clf.intercept_)
```

```
# The below values are from clf.coef_
```

```
values1 = [ 0.87794826, 0.77451129, 0.97895797, 1.9781441 , 1.28622139,
```



```
1.38411178, 0.80588321, 1.25078858, 0.89129903, -0.97918123]
```

```
#Variable Importance
```

```
index = np.arange(len(X_train_lr.columns))
```

```
plt.bar(index, values1)
```

```
plt.xlabel('Variable', fontsize=10)
```

```
plt.ylabel('Coefficient', fontsize=10)
```

```
plt.xticks(index, X_train_lr.columns, fontsize=10, rotation=60)
```

```
plt.title('Coefficient Importance')
```

```
plt.show()
```

```
#Heat map of confusion matrix
```

```
ax= plt.subplot()
```

```
sns.heatmap(confusion_matrix_lr, annot=True, ax = ax)
```

```
ax.set_xlabel('Predicted labels')
```

```
ax.set_ylabel('True labels')
```

```
ax.set_title('Confusion Matrix of Logistic Regression Classifier')
```

```
ax.xaxis.set_ticklabels(['Churn', 'Not Churn'])
```

```
ax.yaxis.set_ticklabels(['Churn', 'Not Churn'])
```

```
#Classnames and column names
```

```
col_names = list(data_final.columns.values)
```

```
classnames = list(data_final.Churn.unique())
```

```

#Decision Tree Model

tre2 = tree.DecisionTreeClassifier().fit(X_train, y_train)


# Tree Visualization

dot_data = StringIO()

tree.export_graphviz(tre2, out_file=dot_data,

                      feature_names=col_names[0:25],

                      class_names=classnames,

                      max_depth = 3,

                      filled=True,

                      rounded=True,

                      special_characters=True)

graph = pydotplus.graph_from_dot_data(dot_data.getvalue())

display(Image(graph.create_png()))


#Evaluation Metric of Decision Tree

clf = tree.DecisionTreeClassifier(max_depth = 5, random_state = 0)

tre2 = clf.fit(X_train, y_train)

predicted = tre2.predict(X_test)

cm_decisiont = metrics.confusion_matrix(y_test, predicted)

print('Confusion Matrix of Decision Tree classifier on test set:\n')

print(cm_decisiont)

```

```

accuracy_decision = clf.score(X_test, y_test)

print('Accuracy of Decision Tree classifier on test set: {:.3f}', (round(accuracy_decision,3)))

print('\n Classification report of Decision Tree classifier:\n', metrics.classification_report(y_test,
predicted))

# Feature importance for Decision Tree model

importances_decision_tree =
pd.DataFrame({'feature':X_train.columns,'importance':np.round(clf.feature_importances_,3)})
importances_decision_tree =
importances_decision_tree.sort_values('importance',ascending=False)

print(importances_decision_tree)

# List to store the average RMSE for each value of max_depth:

max_depth_range = list(range(1, 10))

accuracy = []

for depth in max_depth_range:

    clf = tree.DecisionTreeClassifier(max_depth = depth,

                                     random_state = 0)

    clf.fit(X_train, y_train)

    score = clf.score(X_test, y_test)

    accuracy.append(score)

```

```
#Variable Importance Bar Chart
```

```
importances_decision_tree[:15].plot(x="feature", y="importance", kind="bar")
```

```
#Heat map of confusion matrix from Decision Tree
```

```
ax= plt.subplot()
```

```
sns.heatmap(cm_decisiont, annot=True, ax = ax)
```

```
ax.set_xlabel('Predicted labels')
```

```
ax.set_ylabel('True labels')
```

```
ax.set_title('Confusion Matrix of Decision Tree Classifier')
```

```
ax.xaxis.set_ticklabels(['Churn', 'Not Churn'])
```

```
ax.yaxis.set_ticklabels(['Churn', 'Not Churn'])
```

```
# Neural Network
```

```
nnclass2 = MLPClassifier(activation='relu', solver='sgd', hidden_layer_sizes=(100,100))
```

```
nnclass2.fit(X_train, y_train)
```

```
nnclass2_pred = nnclass2.predict(X_test)
```

```
#Evaluation Metrics
```

```
cm_neural = metrics.confusion_matrix(y_test, nnclass2_pred)
```

```
print('Confusion Matrix of Neural Net classifier on test set:\n')
```

```
print(cm_neural)
```

```
accuracy_decision = nnclass2.score(X_test, y_test)
```

```

print('\n Accuracy of Neural Net on test set: {:.3f}', (round(accuracy_decision,3)))

print('\n Classification report of Neural Network:\n', metrics.classification_report(y_test,
nnclass2_pred))

# Confusion Matrix for Neural Network model

ax= plt.subplot()

sns.heatmap(cm_neural, annot=True, ax = ax)

ax.set_xlabel('Predicted labels')

ax.set_ylabel('True labels')

ax.set_title('Confusion Matrix of Neural Network')

ax.xaxis.set_ticklabels(['Churn', 'Not Churn'])

ax.yaxis.set_ticklabels(['Churn', 'Not Churn'])


#Comparision of Roc Curves

disp = plot_roc_curve(logreg, X_test_lr, y_test)

plot_roc_curve(clf, X_test, y_test, ax=disp.ax_);

plot_roc_curve(nnclass2, X_test, y_test, ax=disp.ax_);


#Distribution of Target variable

y_train.describe()

y_test.describe()

```

```
#Distribution of Continuous variables
```

```
X_train.describe()
```

```
X_test.describe()
```

```
#Distribution of Categorical variables
```

```
X_train.iloc[:,0:23].describe()
```

```
X_test.iloc[:,0:23].describe()
```