Data Science

Project Name: Churn Reduction

Report

Submitted By: Akash Dash

29th August 2018

Contents

1 Introduction

1.1 Problem Statement . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

1.2 Data . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

2 Methodology

2.1 Pre Processing . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

2.1.1 Missing Value Analysis

2.1.1 Outlier Analysis . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

2.1.2 Feature Selection. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

2.1.3 Feature Scaling

2.2 Modelling. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

2.2.1 Preparing Data for Models. . . . . . . . . . . . . . . . . . . . . . .

2.2.2 Model Selection . . . . . . . . . . . . . . . . . . . . . . . . . . . .

2.2.3 Decision Tree Classifier . . . . . . . . . . . . . . . . . . . . . . .

2.2.4 Random Forest . . . . . . . . . . . . . . . . . . . . . . . . . . . .

2.2.5 Logistic regression . . . . . . . . . . . . . . . . . . . . . . . . . .

2.2.6 KNN . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

2.2.7 Naïve Bayes . . . . . . . . . . . . . . . . . . . . . . . . . . . .

Conclusion

3.1 Model Evaluation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

3.1.1 Accuracy

3.1.2 False Negative Rate (FNR) . . . . . . . . . . . . . . . . . . . . . . . . .

3.1.3 Recall . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

3.1.4 Specificity

3.2 Model Selection . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

A Extra Figures

B Pyton Code

1 Introduction

1.1 Problem Statement

Churn (loss of customers to competition) is a problem for companies because it is more expensive to acquire a new customer than to keep your existing one from leaving. The objective of this Case is to predict customer behaviour.

Based on the problem statement we categorize the problem as a “Classification Problem”.

1.2 Data

The objective is to build classification models which classifies the customer depending on various service usage factors. Given below is the sample of the data set that we will use to classify the customers.

Table 1: Training Data (Columns 1-6)

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

Table 2: Training Data (Columns 7-11)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| number.vmail.messages | total.day.minutes | total.day.calls | total.day.charge | total.eve.minutes |
| 25 | 265.1 | 110 | 45.07 | 197.4 |
| 26 | 161.6 | 123 | 27.47 | 195.5 |
| 0 | 243.4 | 114 | 41.38 | 121.2 |
| 0 | 299.4 | 71 | 50.9 | 61.9 |
| 0 | 166.7 | 113 | 28.34 | 148.3 |

Table 3: Training Data (Columns 12-16)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| total.eve.calls | total.eve.charge | total.night.minutes | total.night.calls | total.night.charge |
| 99 | 16.78 | 244.7 | 91 | 11.01 |
| 103 | 16.62 | 254.4 | 103 | 11.45 |
| 110 | 10.3 | 162.6 | 104 | 7.32 |
| 88 | 5.26 | 196.9 | 89 | 8.86 |
| 122 | 12.61 | 186.9 | 121 | 8.41 |

Table 4: Training Data (Columns 17-21)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| total.intl.minutes | total.intl.calls | total.intl.charge | number.customer.service.calls | Churn |
| 10 | 3 | 2.7 | 1 | False. |
| 13.7 | 3 | 3.7 | 1 | False. |
| 12.2 | 5 | 3.29 | 0 | False. |
| 6.6 | 7 | 1.78 | 2 | False. |
| 10.1 | 3 | 2.73 | 3 | False. |

The following features are provided to classify the customers and predict the future cases

**Table 5: Features Available**

Sl.No. Features

* 1. state
  2. account length
  3. area code
  4. phone number
  5. international plan
  6. voicemail plan
  7. number of voicemail messages
  8. total day minutes used
  9. day calls made
  10. total day charge
  11. total evening minutes
  12. total evening calls
  13. total evening charge
  14. total night minutes
  15. total night calls
  16. total night charge
  17. total international minutes used
  18. total international calls made
  19. total international charge
  20. number of customer service calls made("calls.to.cus.service)

2 Methodology

2.1 Pre Processing

Data often comes in messy/noisy format, exploration and pre-processing of data is one of the most important step before analysis. It involves processes like cleaning, organizing and structuring of raw data. Predictive modeling requires that we explore the data before we start modeling. Exploring data involves cleaning as well as visualizing the data through graphs and plots. This is often called as Exploratory Data Analysis. To begin with, we first separate the continuous and categorical variables. Then we look at the distribution of continuous variables. For predictive analysis the data should be normally distributed. This can be visualized by glancing at the distribution plots of the variables.

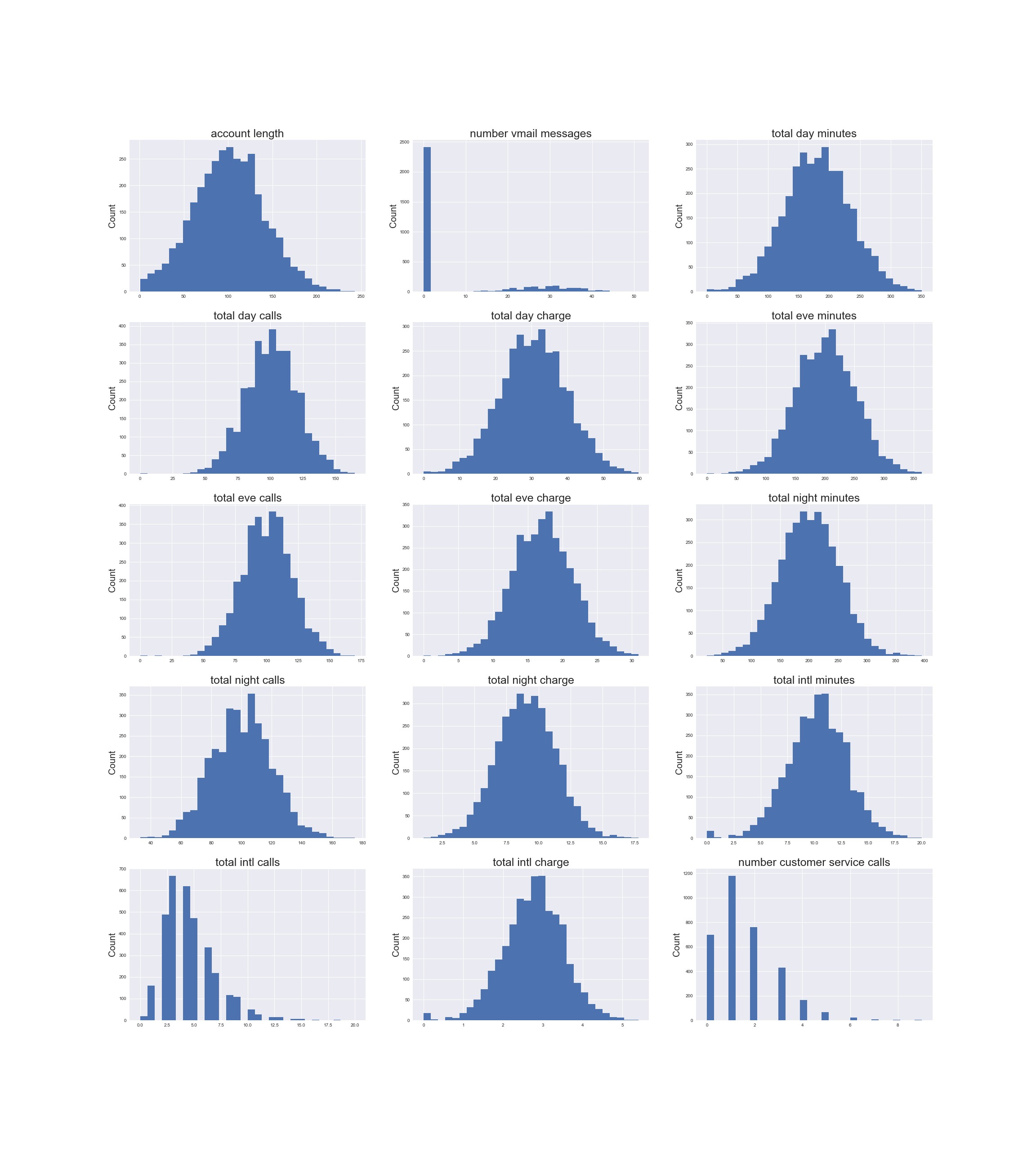


Figure 1: Distribution plots

In Fig.1 we have plotted distributions of the continuous variables in the data and we can observe that most of them are uniformly distributed excluding variables like “number of voicemail messages”, “total international calls” and “number of service calls”.

Plotting the counts of the target variable i.e. “Churn” shown in Fig.2, a target class imbalance can be observed. To deal with target class imbalance, we go for SMOTE (Synthetic Minority Oversampling Technique)

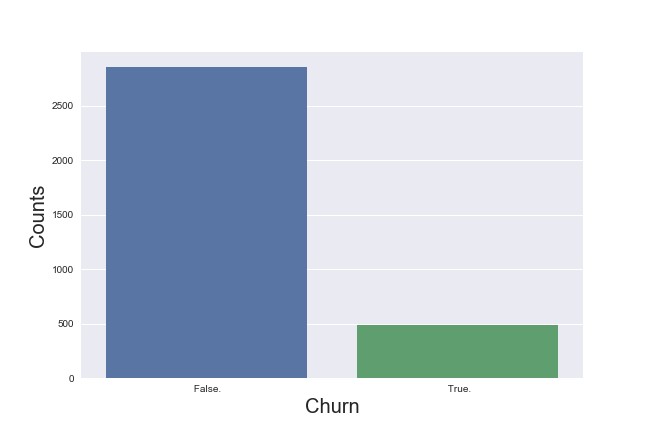


Figure 2: Count plot of Target Class

2.1.1 Missing Value Analysis

There is always a possibility of missing values in the data, which could be because of various reasons like Human error, no response from the respondent or optional question. To perform the MVA, we should understand the reason due to which the data might be missing. Steps that be taken to deal with missing values can be

1. Imputing Value
2. Dropping observation
3. If the variable has > 30% missing values; dropping the variable

Since, the data has no missing values, the steps end with checking the Missing values for each variable.

2.1.2 Outlier Analysis

As we have seen the variables “number of voicemail messages”, “total international calls” and “number of service calls” are skewed indicating it could be due to outliers, but plotting at the number of “voicemail messages” without the zero value; which has clearly the most number of count, the rest of the data is uniformly distributed as shown in Fig.3 also the zero value is not specific to any of the target class as seen in Fig.4, so we will leave the zero value as it is, otherwise on dropping these observations we will lose too much information and the zero value of this variable has a significance. The variables “total international calls” and “number of service calls” also have similar distributions for both classes as shown in appendix in fig.8 and 9 and the range of these variables are not big enough to remove the outliers so we will also leave these variables as it is.

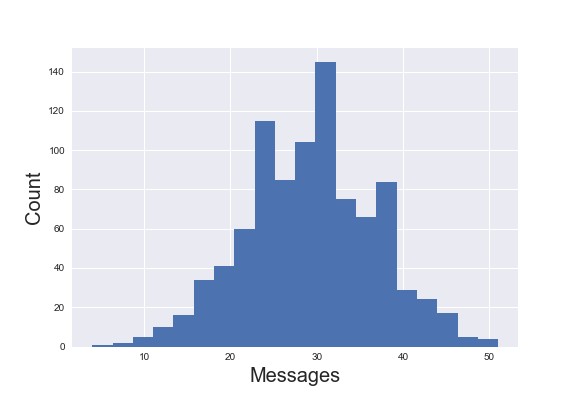


Figure 3: Number of voice mail messages distribution without zero

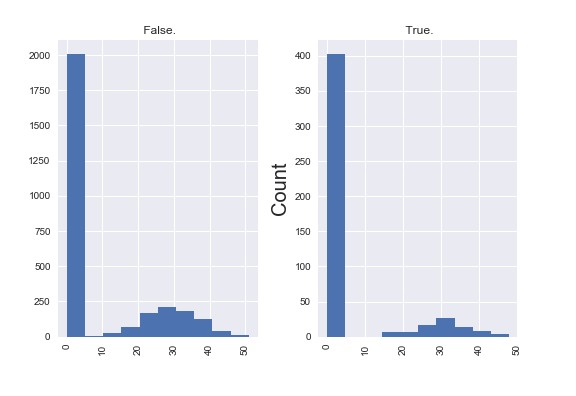


Figure 4: Number of voice mail messages distribution by Target Class

2.1.2 Feature Selection

Prior performing any sort of modeling assessing the importance of each predictor variable is most essential for our analysis. There may be a possibility that many variables do not have much information or are not significant to deal with the problem of class prediction. There are various methods of assessing the significance, but we have used Chi Square test to assess the significance of the categorical variables. We can observe from the p-values of each variable that except the “area code” variable all are significant.

We also check multicollinearity or whether the independent variables are correlated. We can do so by plotting heatmap of the correlation matrix of the continuous variables as shown in Fig.5. We can clearly see that all the charge and subsequent minutes variable pairs are strongly correlated which is obvious, so we should drop one of these from the data.

All the other variables should have different significance as we can say from knowledge that the variables like total day calls, total eve calls, total night calls should have different importance in deciding whether the customer will keep using the service or not.

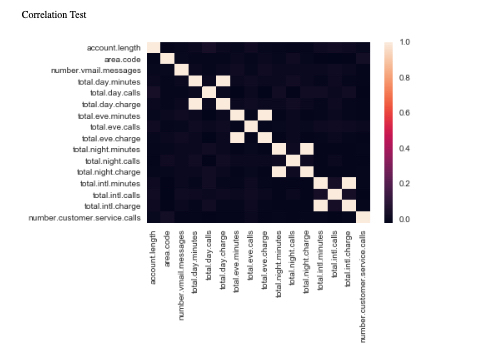


Figure 5: Correlation plot

2.1.3 Feature Scaling

Feature scaling is also called as variable scaling; it is one of the most important steps when we deal with continuous variables of different scales. As we can see, the variables like “total day minutes”, “total day calls”, “total voicemail messages”, “international plan” all are of different scales. For analyses, the variable should be scaled unitarily. For this purpose we used Normalization and Standardization of variables.

Normalization is the process of reducing unwanted variation within or between variables. It is a process to bring all the variables in proportion with one another.

Whereas, standardization is the process of converting each data point of a variable to a unit of standard deviation. We perform standardization on the variables which are uniformly distributed, for the rest continuous variables we perform normalization. From the distribution plots shown in fig 1, we can see variables "number of voicemail messages" and "number of customer service calls" are not normally distributed. Whereas, variables like "account length", "total day minutes", "total day calls", "total evening minutes", "total evening calls", "total night minutes", "total night calls" and "total international minutes" are normally distributed. Thus, normalization and standardization are performed on respective variables. It is also important to note that the variables are considered after dropping the variables which are either not significant or have multicollinearity with other variables; in other words after feature selection the feature scaling is performed on the selected variables.

2.2 Modelling

2.2.1 Preparing Data for Models

Now we have to prepare the data for the model to understand. Dummy variables could have been created for the categorical variables but as all other variables apart from “state” variable all the categorical variables are binary. The target variable can be substituted with values with 0 and 1 for No or Yes or False or True respectively as to avoid increasing the dimensionality of the data. We can also observe from Fig.6 that the state variable is also uniformly distributed so we can either create dummy variables for each state or we could assign a code value to each state. We have assigned code values since the model works better on the later.

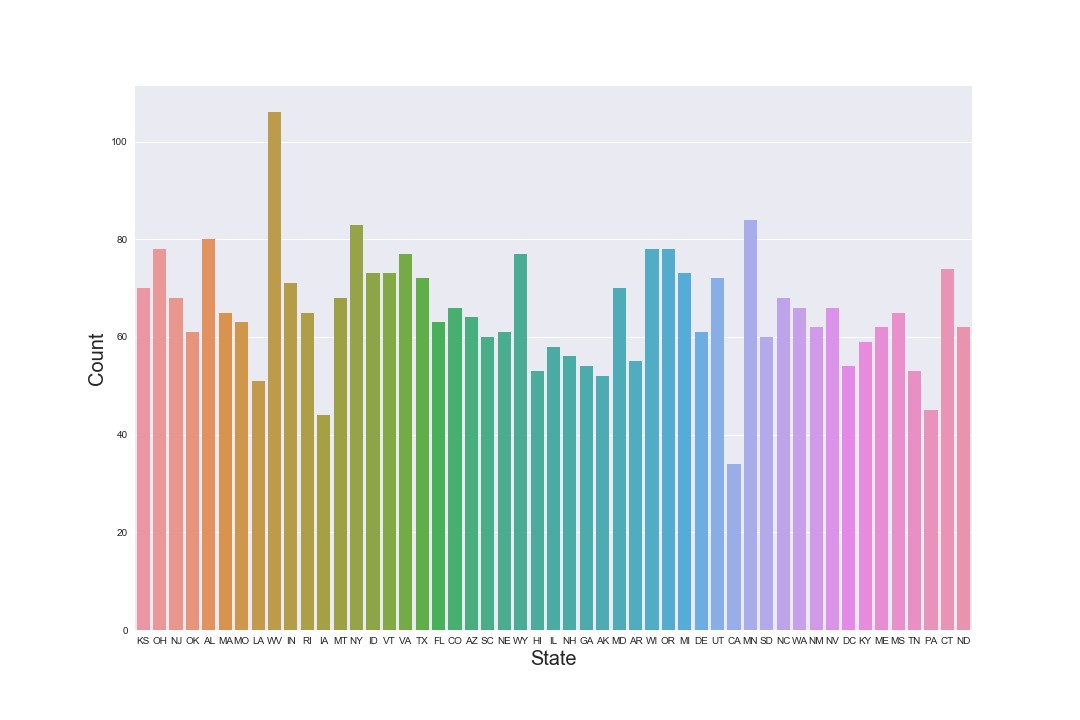


Figure 6: States distribution plot

Also we can also see in Fig.2 that there is a target class imbalance, we deal with it by oversampling the data using Informed over sampling by Synthetic minority over-sampling Technique (SMOTE) to balance the classes. Here we created a train test split of data in the proportion of 80:20 and saved the independent variable in X\_train and X\_test and independent variables under Y\_train and Y\_test. Since the presence of target class imbalance, after application of SMOTE we saved the independent variables in X\_train\_over and X\_test\_over variables and dependent variable in Y\_train\_over and Y\_test\_over variables.

2.2.2 Model Selection

The dependent variable “Churn” is binary, the predictive analytics we use is Classification. So we build various classifier machine learning algorithms like Decision Tree, Random Forest, Logistic regression, KNN, and Naïve Bayes, to predict our target variable and then select whichever works the best.

2.2.3 Decision Tree Classifier

One of the basic supervised machine learning algorithm, decision tree helps us in predicting/classifying the target class variable “Churn”. We created a DT model named clf and ran it on the X\_test\_over to predict and compare it with Y\_test\_over.

Accuracy : 88.87%

FNR: 12.52%

Recall: 87.4

Specificity: 90.28%

2.2.4 Random Forest

Random forest is a supervised machine learning algorithm that is an ensemble consisting of many trees. We used RF\_model for classification by considering 100, 300 and 500 trees to find the best model to classify the target variable.

|  |  |  |  |
| --- | --- | --- | --- |
| Trees | 100 | 300 | 500 |
| Accuracy | 91.92 | 91.39% | 92.03% |
| FNR | 12.64 | 13.89% | 12.41% |
| Recall | 87.35 | 86.18% | 87.58% |
| Specificity | 96.48 | 96.60% | 96.48% |

We can see the model performs best with 500 trees.

2.2.5 Logistic regression

Our main importance in the prediction is to keep the False negative rate to a minimum so we can select a custom threshold probability instead of default 0.5. Here our outcome is in probability that we are converting into 0 or 1. For probability < 0.5, we classify the data point as 0 whereas for a probability > 0.5 it is classified as 1.

|  |  |  |  |
| --- | --- | --- | --- |
| **Dep. Variable:** | Churn | **No. Observations:** | 4014 |
| **Model:** | Logit | **Df Residuals:** | 4000 |
| **Method:** | MLE | **Df Model:** | 13 |
| **Date:** | Thu, 29 Nov 2018 | **Pseudo R-squ.:** | 0.1476 |
| **Time:** | 04:19:37 | **Log-Likelihood:** | -1405.6 |
| **converged:** | True | **LL-Null:** | -1649.0 |
|  |  | **LLR p-value:** | 9.014e-96 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **coef** | **std err** | **z** | **P>|z|** | **[0.025** | **0.975]** |
| **account.length** | 0.0395 | 0.048 | 0.819 | 0.413 | -0.055 | 0.134 |
| **number.vmail.messages** | 0.4756 | 0.632 | 0.753 | 0.452 | -0.763 | 1.714 |
| **total.day.minutes** | 0.5736 | 0.051 | 11.318 | 0.000 | 0.474 | 0.673 |
| **total.day.calls** | 0.0462 | 0.049 | 0.952 | 0.341 | -0.049 | 0.141 |
| **total.eve.minutes** | 0.2469 | 0.049 | 5.012 | 0.000 | 0.150 | 0.343 |
| **total.eve.calls** | -0.0449 | 0.049 | -0.916 | 0.360 | -0.141 | 0.051 |
| **total.night.minutes** | 0.1983 | 0.048 | 4.099 | 0.000 | 0.104 | 0.293 |
| **total.night.calls** | -0.0314 | 0.049 | -0.643 | 0.520 | -0.127 | 0.064 |
| **total.intl.minutes** | 0.1610 | 0.049 | 3.271 | 0.001 | 0.065 | 0.258 |
| **total.intl.calls** | -0.1111 | 0.025 | -4.529 | 0.000 | -0.159 | -0.063 |
| **number.customer.service.calls** | -0.1204 | 0.153 | -0.789 | 0.430 | -0.419 | 0.179 |
| **international.plan\_0.0** | -1.5862 | 2.22e+06 | -7.13e-07 | 1.000 | -4.36e+06 | 4.36e+06 |
| **international.plan\_1.0** | 0.3289 | 2.22e+06 | 1.48e-07 | 1.000 | -4.36e+06 | 4.36e+06 |
| **voice.mail.plan\_0.0** | 0.0810 | 2.22e+06 | 3.64e-08 | 1.000 | -4.36e+06 | 4.36e+06 |
| **voice.mail.plan\_1.0** | -1.3383 | 2.22e+06 | -6.02e-07 | 1.000 | -4.36e+06 | 4.36e+06 |

From the output of logistic regression summary we infer this model takes only few features like categorical variables as significant from p-value.

Accuracy: 86.61%

FNR: 87.87

Recall: 12.12%

Specificity: 98.12

The model underperforms due to high FNR.

2.2.6 K-Nearest Neighbor

KNN stores all available cases and classifies new cases based on similarity measures. This machine learning algorithm is used when data is labelled, noise-free and small. It is one of the lazy learning algorithms as it takes into account each data points and tries to classify based on similarity of features.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| K | 3 | 5 | 7 | 9 | 11 |
| Accuracy | 59.70% | 60.42% | 59.19% | 59.89% | 60.42% |
| FNR | 52.22% | 46.60% | 44.96% | 40.74% | 38.64% |
| Recall | 47.77% | 53.33% | 55.03% | 59.25% | 61.35% |
| Specificity | 71.66% | 67.44% | 63.34 | 60.53% | 59.48% |

Table 8: Values of K and the corresponding metrics

2.2.7 Naïve Bayes

Naïve Bayes is another supervised machine learning algorithm, which works on Bayes theorem of probability to classify class of target variable. Here as the training data increases the performance of the model to predict the probability of target variable increases. We use Gaussian naïve Bayes model NB\_model which is run on X\_train\_over and Y\_train\_over oversampled data to predict on X\_test\_over data.

Accuracy: 69.20%

FNR:18.96

Recall:81.03%

Specificity57.37%

3 Conclusion

3.1 Model Evaluation

Now that we have a few models for predicting the target variable, we can decide which one to choose. There are several criteria that exist for evaluating and comparing models. We can compare the models using any of the following criteria:

1. Predictive performance

2. Interpretability

3. Computational Efficiency

In this case, Interpretability and Computation Efficiency, do no hold much significance. Therefore, we use Predictive performance as the criterion to compare and evaluate models.

Predictive performance can be measured by comparing Predictions of the models with real values of the target variables, and calculating average error measure.

3.1.1 False Negative Rate (FNR)

False negative rate is the percentage of misclassified positives. It can be calculated by creating confusion matrix.

3.1.2 Specificity

Specificity is the proportion of actual negative cases which are correctly predicted by the model.

3.1.3 Recall

Recall is the proportion of actual positive cases which are correctly identified by the model.

3.2 Model Selection

Empowered by many models to classify the target variable, we thus select the one which will not only minimise the “False negative rate” but also give accuracy of model. But according to the problem statement more than accuracy it is important to predict and correctly classify the negative Churns, in order to retain them by providing additional incentives. So we will select the model which gives us the least False negative rate.

Comparing the metrics of all the models we see that Random Forest model seems to be working best on our test data. So we will select Random Forest as our predictive model. Although the FNR of Decision Tree is less but it also is less accurate, if from the business aspect the high false positive rate is not an issue we can also select Decision Tree as our model. Here we have chosen Random Forest as to be our classifier.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model | Accuracy | FNR | Recall | Specificity |
| Decision Tree | 88.87% | 12.52% | 87.40% | 90.28% |
| Random Forest | 92.03% | 12.41% | 87.58% | 96.48% |
| Logistic Regression | 86.61% | 87.87% | 12.12% | 98.12% |
| KNN | 60.42% | 38.64% | 61.35% | 59.48% |
| Naïve Bayes | 69.20% | 18.03% | 81.03% | 57.37% |

Table 10: Final results

A Extra Figures

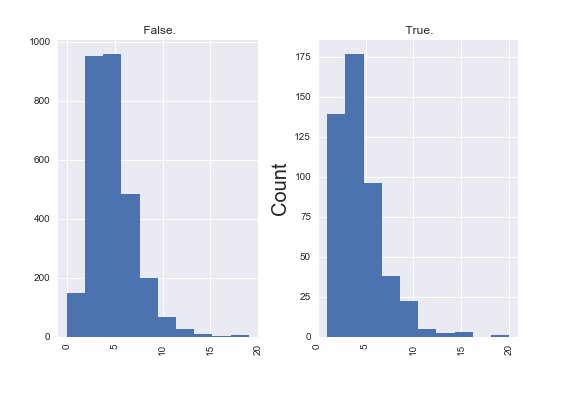


Figure 8: International Calls by Class

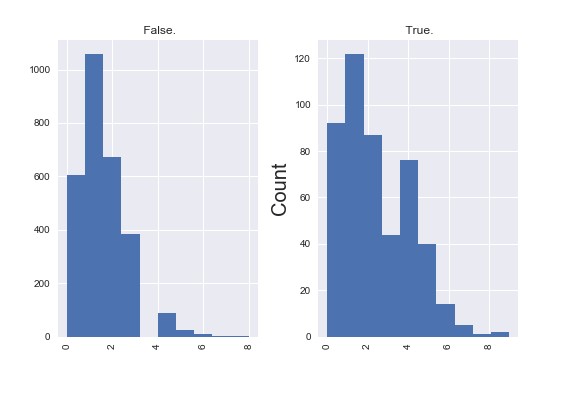


Figure 9: Customer Service Calls by Class

Python Code:

*# Import Libraries*

**import** **pandas** **as** **pd**

**import** **numpy** **as** **np**

**import** **matplotlib** **as** **mlt**

**import** **seaborn** **as** **sns**

**import** **scipy**

**import** **matplotlib.pyplot** **as** **plt**

**import** **os**

**from** **string** **import** ascii\_letters

**from** **pylab** **import** rcParams

In [ ]:

*# Load Libraries*

**from** **fancyimpute** **import** KNN

**from** **scipy.stats** **import** chi2\_contingency

**from** **random** **import** randrange, uniform

**from** **sklearn.model\_selection** **import** train\_test\_split

In [ ]:

*# Set working directory*

os.chdir("/Users/ad/Desktop/Project 1")

*# Check working directory*

os.getcwd()

In [ ]:

*# Load Data to python*

train = pd.read\_csv("Train\_data.csv")

test = pd.read\_csv("Test\_data.csv")

In [ ]:

data = train.append(test)

# Exploratory Data Analysis

In [ ]:

data.head(5)

In [ ]:

data.shape

In [ ]:

data.dtypes

In [ ]:

data.columns

In [ ]:

data.describe()

*# Assigning codes to each state*

keys = data['state'].unique().tolist()

values = list(range(len(keys)))

state\_codes = dict(zip(keys,values))

data['state'] = data['state'].map(state\_codes)

In [ ]:

*# Separate Continuous and Categorical Variables*

*# Excluding phone.number variable and dependent variable*

cnames= ["account.length","number.vmail.messages", "total.day.minutes","total.day.calls", "total.day.charge",

"total.eve.minutes","total.eve.calls","total.eve.charge","total.night.minutes",

"total.night.calls","total.night.charge","total.intl.minutes","total.intl.calls", "total.intl.charge",

"number.customer.service.calls"]

cat\_names= ["state", "area.code", "international.plan","voice.mail.plan"]

In [ ]:

*# Assigning levels to the categories*

lis = []

**for** i **in** range(0, data.shape[1]):

*#print(i)*

**if**(data.iloc[:,i].dtypes == 'object'):

data.iloc[:,i] = pd.Categorical(data.iloc[:,i])

*#print(data.iloc[i])*

data.iloc[:,i] = data.iloc[:,i].cat.codes

data.iloc[:,i] = data.iloc[:,i].astype('object')

lis.append(data.columns[i])

In [ ]:

*## Checking correlations values of continous variables*

corr = data.corr()

corr.style.background\_gradient()

In [ ]:

*## Checking correlations of continous variables*

*# Correlation Plot*

df\_corr = data.loc[:,cnames]

corr = df\_corr.corr()

sns.heatmap(corr,

xticklabels=corr.columns.values,

yticklabels=corr.columns.values)

In [ ]:

*# Checking dependency of dependent variable on categorical variables*

*# Loop for chi square values*

*# Variable area.code not significant*

**for** i **in** cat\_names:

print(i)

chi2, p, dof, ex = chi2\_contingency(pd.crosstab(data['Churn'], data[i]))

print(p)

In [ ]:

*#Checking counts of Target variable*

plt.figure(figsize = (9,6)),

sns.set(font\_scale = 1),

sns.countplot(x = 'Churn', data = data)

plt.xlabel ( 'Churn' , fontsize = 20)

plt.ylabel ( 'Counts' , fontsize = 20)

In [ ]:

*# Check Number of Voicemail Messages by Class*

plt.figure(figsize = (10,15))

data.hist('number.vmail.messages', by = 'Churn')

plt.ylabel('Count' , fontsize = 20)

In [ ]:

*# Plot total.intl.calls by class*

plt.figure(figsize = (10,15))

data.hist('total.intl.calls', by = 'Churn')

plt.ylabel('Count' , fontsize = 20)

In [ ]:

*# Plot number.customer.service.calls by class*

plt.figure(figsize = (10,15))

data.hist('number.customer.service.calls', by = 'Churn')

plt.ylabel('Count' , fontsize = 20)

In [ ]:

*# Plot of States*

plt.figure(figsize = (15,10))

sns.countplot('state', data= data)

plt.xlabel('State',fontsize = 20)

plt.ylabel('Count',fontsize = 20)

# MISSING VALUE ANALYSIS

In [ ]:

*# Checking Missing Values*

missing\_val= pd.DataFrame(data.isnull().sum())

missing\_val

*#No missing values*

# Outlier Analysis

In [ ]:

*# Boxplot to visualize outliers*

%**matplotlib** inline

plt.boxplot(data['total.intl.minutes'])

In [ ]:

*# Detect and delete outliers from data*

**for** i **in** cnames:

print(i)

q75, q25 = np.percentile(data.loc[:,i], [75, 25])

iqr = q75 - q25

min = q25 - (iqr \* 1.5)

max = q75 + (iqr \* 1.5)

print(min)

print(max)

data.loc[data.loc[:,i] < min,i]= np.nan

data.loc[data.loc[:,i] > max,i]= np.nan

In [ ]:

*# Calculate the missing values*

missing\_val= pd.DataFrame(data.isnull().sum())

print(data.isnull().any())

In [ ]:

missing\_val

In [ ]:

*# Impute missing values with KNN*

data = pd.DataFrame(KNN(k = 3).fit\_transform(data), columns = data.columns)

# Feature Selection

In [ ]:

data\_copy = data.copy()

*#data = data\_copy*

In [ ]:

*# Drop Variables*

data = data.drop(["phone.number","area.code","total.day.charge","total.eve.charge","total.night.charge","total.intl.charge"], axis= 1)

In [ ]:

data.shape

# Feature Scaling

In [ ]:

data.head()

In [ ]:

*# Normality Check*

%**matplotlib** inline

plt.hist(data['number.customer.service.calls'], bins='auto')

In [ ]:

not\_norm = ["number.vmail.messages","number.customer.service.calls"]

In [ ]:

*# Normalisation*

**for** i **in** not\_norm:

print(i)

data[i] = (data[i] - data[i].min())/ (data[i].max()- data[i].min())

In [ ]:

data.shape

In [ ]:

var\_norm = ["account.length","total.day.minutes", "total.day.calls", "total.eve.minutes", "total.eve.calls","total.night.minutes","total.night.calls","total.intl.minutes"]

In [ ]:

*# Standardization*

**for** i **in** var\_norm:

print(i)

data[i]= (data[i]- data[i].mean())/data[i].std()

In [ ]:

data.shape

# MODELLING

In [ ]:

*# Divide the data into train and test*

**from** **sklearn.model\_selection** **import** train\_test\_split

X = data.values[:, 0:14] *# Independent Variable*

Y = data.values[:,14] *# Dependent Variable*

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X, Y, test\_size = 0.2)

In [ ]:

*# Oversampling of minor class target variable*

**from** **imblearn.over\_sampling** **import** SMOTE

smote = SMOTE()

print("Before oversampling, count of class '1': **{}**".format(sum(Y\_train == 1)))

print("Before oversampling, count of class '0': **{}**".format(sum(Y\_train == 0)))

In [ ]:

X\_train\_over, Y\_train\_over = smote.fit\_sample(X\_train, Y\_train.ravel())

In [ ]:

print("After oversampling, count of class '1': **{}**".format(sum(Y\_train\_over == 1)))

print("After oversampling, count of class '0': **{}**".format(sum(Y\_train\_over == 0)))

In [ ]:

print("Before oversampling, count of class '1': **{}**".format(sum(Y\_test == 1)))

print("Before oversampling, count of class '0': **{}**".format(sum(Y\_test == 0)))

In [ ]:

X\_test\_over, Y\_test\_over = smote.fit\_sample(X\_test, Y\_test.ravel())

In [ ]:

print("After oversampling, count of class '1': **{}**".format(sum(Y\_test\_over == 1)))

print("After oversampling, count of class '0': **{}**".format(sum(Y\_test\_over == 0)))

# Decision Tree

In [ ]:

*#Import Libraries*

**from** **sklearn** **import** tree

**from** **sklearn.metrics** **import** accuracy\_score

In [ ]:

clf = tree.DecisionTreeClassifier(criterion= 'entropy').fit(X\_train\_over, Y\_train\_over)

In [ ]:

*# Predict new test cases*

y\_pred = clf.predict(X\_test\_over)

In [ ]:

*# Create dot file to visualize tree ( http://webgraphviz.com/)*

*#dotfile= open("pt.dot", 'w')*

*#df = tree.export\_graphviz(clf, out\_file= dotfile, feature\_names= data.columns)*

In [ ]:

*# Checking the accuracy*

accuracy\_score(Y\_test\_over, y\_pred)\*100

In [ ]:

*# Build the Confusion Matrix*

**from** **sklearn.metrics** **import** confusion\_matrix

CM = confusion\_matrix(Y\_test\_over, y\_pred)

In [ ]:

CM

In [ ]:

*# Build Confusion Matrix*

CM = pd.crosstab(Y\_test\_over, y\_pred)

*# Store TP,TN,FP,FN values*

TN = CM.iloc[0,0]

FN = CM.iloc[1,0]

TP = CM.iloc[1,1]

FP = CM.iloc[0,1]

*# Accuracy*

*#(TN+TP)/(TN+TP+FN+FP)*

*# FNR*

(FN\*100)/(FN+TP)

*# Recall*

(TP\*100)/(TP+FN)

*# Specificity*

(TN\*100)/(TN+FP)

# Random Forest

In [ ]:

*# Import Libraries*

**from** **sklearn.ensemble** **import** RandomForestClassifier

RF\_model= RandomForestClassifier(n\_estimators = 500).fit(X\_train\_over, Y\_train\_over)

In [ ]:

RF\_predictions = RF\_model.predict(X\_test\_over)

In [ ]:

*# Build Confusion Matrix*

CM = pd.crosstab(Y\_test\_over, RF\_predictions)

*# Store TP,TN,FP,FN values*

TN = CM.iloc[0,0]

FN = CM.iloc[1,0]

TP = CM.iloc[1,1]

FP = CM.iloc[0,1]

*# Check Accuracy of the model*

*#((TN+TP)\*100)/(TN+FN+TP+FP)*

*# Check FNR*

*#(FN\*100)/(FN+TP)*

*# Recall*

*#(TP\*100)/(TP+FN)*

*# Specificity*

*#(TN\*100)/(TN+FP)*

# Logistic Regression

In [ ]:

*# Replace Target Variable with 0 and 1*

data['Churn']= data['Churn'].replace('No', 0)

data['Churn']= data['Churn'].replace('Yes', 1)

In [ ]:

data\_logit= pd.DataFrame(data['Churn'])

In [ ]:

data\_logit.shape

In [ ]:

cnames= ["account.length", "number.vmail.messages", "total.day.minutes","total.day.calls",

"total.eve.minutes","total.eve.calls","total.night.minutes",

"total.night.calls","total.intl.minutes","total.intl.calls",

"number.customer.service.calls"]

In [ ]:

*# Add continuous variable*

data\_logit = data\_logit.join(data[cnames])

In [ ]:

*# Create dummies for categorical variables*

cat\_names= ["international.plan","voice.mail.plan"]

**for** i **in** cat\_names:

temp = pd.get\_dummies(data[i], prefix = i)

data\_logit = data\_logit.join(temp)

data\_logit.shape

sample\_index = np.random.rand(len(data\_logit)) < 0.8

train = data\_logit[sample\_index]

test = data\_logit[~sample\_index]

*# Select columns indexes for independent variables*

train\_cols = train.columns[1:17]

In [ ]:

train\_cols

In [ ]:

*# Build logistic regression model*

**import** **statsmodels.api** **as** **sm**

logit = sm.Logit(train['Churn'], train[train\_cols]).fit()

In [ ]:

logit.summary()

In [ ]:

*# Predict Test Data*

test['Actual\_prob'] = logit.predict(test[train\_cols])

In [ ]:

test.head()

In [ ]:

test['Actualval'] = 1

test.loc[test.Actual\_prob < 0.5, 'Actualval'] = 0

In [ ]:

*# Build Confusion Matrix*

CM = pd.crosstab(test['Churn'], test['Actualval'])

*# Store TP,TN,FP,FN values*

TN = CM.iloc[0,0]

FN = CM.iloc[1,0]

TP = CM.iloc[1,1]

FP = CM.iloc[0,1]

*# Check Accuracy of the model*

*#((TN+TP)\*100)/(TN+FN+TP+FP)*

*# Check FNR*

*#(FN\*100)/(FN+TP)*

*# Recall*

*#(TP\*100)/(TP+FN)*

*# Specificity*

*#(TN\*100)/(TN+FP)*

# KNN

In [ ]:

**from** **sklearn.neighbors** **import** KNeighborsClassifier

KNN\_model = KNeighborsClassifier(n\_neighbors = 11).fit(X\_train\_over, Y\_train\_over)

In [ ]:

*# Predict the test cases*

KNN\_pred = KNN\_model.predict(X\_test\_over)

In [ ]:

*# Build Confusion Matrix*

CM = pd.crosstab(Y\_test\_over, KNN\_pred)

*# Store TP,TN,FP,FN values*

TN = CM.iloc[0,0]

FN = CM.iloc[1,0]

TP = CM.iloc[1,1]

FP = CM.iloc[0,1]

*# Check Accuracy of the model*

((TN+TP)\*100)/(TN+FN+TP+FP)

*# Check FNR*

*#(FN\*100)/(FN+TP)*

*# Recall*

*#(TP\*100)/(TP+FN)*

*# Specificity*

*#(TN\*100)/(TN+FP)*

Naive Bayes

In [ ]:

**from** **sklearn.naive\_bayes** **import** GaussianNB

*# Build Naive Bayes model*

NB\_model = GaussianNB().fit(X\_train\_over, Y\_train\_over)

In [ ]:

*# Predict the test cases*

NB\_pred = NB\_model.predict(X\_test\_over)

In [ ]:

*# Build Confusion Matrix*

CM = pd.crosstab(Y\_test\_over, NB\_pred)

*# Store TP,TN,FP,FN values*

TN = CM.iloc[0,0]

FN = CM.iloc[1,0]

TP = CM.iloc[1,1]

FP = CM.iloc[0,1]

*# Check Accuracy of the model*

*#((TN+TP)\*100)/(TN+FN+TP+FP)*

*# Check FNR*

*#(FN\*100)/(FN+TP)*

*# Recall*

*#(TP\*100)/(TP+FN)*

*# Specificity*

*#(TN\*100)/(TN+FP)*