**Churn Reduction**

Minumula Rajashekar

10 November 2018

Contents

**1.**[Introduction 3](#_Toc529726233)

[**1.1** **Problem Statement** 3](#_Toc529726234)

[**1.2 Data** 3](#_Toc529726235)

**2.**[**Methodology** 4](#_Toc529726237)

[**2.1** **Pre Processing** 4](#_Toc529726238)

[**2.1.1 Data Cleaning** 5](#_Toc529726239)

[**2.1.2 Feature Selection** 8](#_Toc529726245)

[**2.2** **Modeling** 11](#_Toc529726247)

[**2.2.1** **Model Selection** 11](#_Toc529726248)

[**2.2.2 Decision Tree** 11](#_Toc529726249)

[**2.2.3 Logistic Regression** 14](#_Toc529726251)

[**2.2.4 K Nearest Neighbors** 16](#_Toc529726252)

[**2.2.5 Naïve Bayes** 16](#_Toc529726256)

[**3 Conclusion** **17**](#_Toc111189)

[**3.1 Model Evaluation 17**](#_Toc529726263)

[**3.1.1** **Accuracy and False Negative Rate** 17](#_Toc529726264)

[**3.2** **Model Selection** 19](#_Toc529726266)

[**4.1 Complete R Code** 19](#_Toc529726267)

[**4.2 Complete Python Code** 26](#_Toc529726348)

# Introduction

* 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 behavior from the public dataset that has customer usage pattern and if the customer has moved or not. We would like to develop an algorithm to predict the churn score based on usage pattern

* 1. **Data**

Our task is to build classification models which will predict the churn score based on the customer behavior of the public data. Given below is a sample of the data set that we are using to predict churn score.

churn reduction (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 |

churn reduction (Columns: 7-15)

| **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** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 25 | 265.1 | 110 | 45.07 | 197.4 | 99 | 16.78 | 244.7 | 91 |
| 26 | 161.6 | 123 | 27.47 | 195.5 | 103 | 16.62 | 254.4 | 103 |
| 0 | 243.4 | 114 | 41.38 | 121.2 | 110 | 10.30 | 162.6 | 104 |
| 0 | 299.4 | 71 | 50.90 | 61.9 | 88 | 5.26 | 196.9 | 89 |

churn reduction (Columns: 16-21)

| **total.night.charge** | **total.intl.minutes** | **total.intl.calls** | **total.intl.charge** | **number.customer.service.calls** | **Churn** |
| --- | --- | --- | --- | --- | --- |
| 11.01 | 10.0 | 3 | 2.70 | 1 | False |
| 11.45 | 13.7 | 3 | 3.70 | 1 | False |
| 7.32 | 12.2 | 5 | 3.29 | 0 | False |
| 8.86 | 6.6 | 7 | 1.78 | 2 | False |

As you can see in the table below we have the following 20 variables, using which we have to correctly predict the churn(loss of customers to competition) score:

| **S.No.** | **Predictor** |
| --- | --- |
| 1 | state |
| 2 | account.length |
| 3 | area.code |
| 4 | phone.number |
| 5 | international.plan |
| 6 | voice.mail.plan |
| 7 | number.vmail.messages |
| 8 | total.day.minutes |
| 9 | total.day.calls |
| 10 | total.day.charge |
| 11 | total.eve.minutes |
| 12 | total.eve.calls |
| 13 | total.eve.charge |
| 14 | total.night.minutes |
| 15 | total.night.calls |
| 16 | total.night.charge |
| 17 | total.intl.minutes |
| 18 | total.intl.calls |
| 19 | total.intl.charge |
| 20 | number.customer.service.calls |

Predictor Variables

**Chapter 2**

**Methodology**

**2.1 Pre Processing**

Any predictive modeling requires that we look at the data before we start modeling. However, in data mining terms *looking at data* refers to so much more than just looking. Looking at data refers to exploring the data, cleaning the data as well as visualizing the data through graphs and plots. This is often called as **Exploratory Data Analysis**. To start this process we will first try and look at all the probability distributions of the variables. Most analysis like regression, require the data to be normally distributed. We can visualize that in a glance by looking at the probability distributions or probability density functions of the variable.

As we are provided with train and test data so we will first test the missing values in both test and train data tests in 2.1.1 Data Cleaning section.

**2.1.1 Data Cleaning**

In this section we will check missing values in both training data and test data. Below are the missing values we found in both train and test data.

**R Code**

**#training data**

churn\_train=read.csv("Train\_data.csv",sep=",")

missing\_val\_train= data.frame(apply(churn\_train,2,function(x){sum(is.na(x))}))

missing\_val\_train

**#test data**

churn\_test=read.csv("Test\_data.csv",sep=",")

missing\_val\_test= data.frame(apply(churn\_test,2,function(x){sum(is.na(x))}))

missing\_val\_test

state 0

account.length 0

area.code 0

phone.number 0

international.plan 0

voice.mail.plan 0

number.vmail.messages 0

total.day.minutes 0

total.day.calls 0

total.day.charge 0

total.eve.minutes 0

total.eve.calls 0

total.eve.charge 0

total.night.minutes 0

total.night.calls 0

total.night.charge 0

total.intl.minutes 0

total.intl.calls 0

total.intl.charge 0

number.customer.service.calls 0

Churn 0

**Fig 2.1.1 Missing values in Train Data**

state 0

account.length 0

area.code 0

phone.number 0

international.plan 0

voice.mail.plan 0

number.vmail.messages 0

total.day.minutes 0

total.day.calls 0

total.day.charge 0

total.eve.minutes 0

total.eve.calls 0

total.eve.charge 0

total.night.minutes 0

total.night.calls 0

total.night.charge 0

total.intl.minutes 0

total.intl.calls 0

total.intl.charge 0

number.customer.service.calls 0

Churn 0

**BOX PLOT R Code**

**# churn score boxplots for train**

library(ggplot2)

num\_index=sapply(churn\_train,is.numeric)

num\_data=churn\_train[,num\_index]

cnames=colnames(num\_data)

for (i in 1:length(cnames))

{

assign(paste0("gn",i), ggplot(aes\_string(y = (cnames[i]), x = 'Churn'), data = churn\_train)+

stat\_boxplot(geom = "errorbar", width = 0.5) +

geom\_boxplot(outlier.colour="red", fill = "grey" ,outlier.shape=18,

outlier.size=1, notch=FALSE) +

theme(legend.position="bottom")+

labs(y=cnames[i],x="Churn")+

ggtitle(paste("Box plot of Churn vs",cnames[i])))

}

gridExtra::grid.arrange(gn1,gn5,gn2,gn3,gn4,gn6,ncol=6)

gridExtra::grid.arrange(gn7,gn8,gn9,gn10,gn11,ncol=5)

gridExtra::grid.arrange(gn12,gn13,gn14,gn15,gn16,ncol=5)

**# churn score boxplots for test**

library(ggplot2)

num\_index=sapply(churn\_test,is.numeric)

num\_data=churn\_test[,num\_index]

cnames=colnames(num\_data)

for (i in 1:length(cnames))

{

assign(paste0("gn",i), ggplot(aes\_string(y = (cnames[i]), x = 'Churn'), data = churn\_test)+

stat\_boxplot(geom = "errorbar", width = 0.5) +

geom\_boxplot(outlier.colour="red", fill = "grey" ,outlier.shape=18,

outlier.size=1, notch=FALSE) +

theme(legend.position="bottom")+

labs(y=cnames[i],x="Churn")+

ggtitle(paste("Box plot of Churn vs",cnames[i])))

}

gridExtra::grid.arrange(gn1,gn5,gn2,gn3,gn4,gn6,ncol=6)

gridExtra::grid.arrange(gn7,gn8,gn9,gn10,gn11,ncol=5)

gridExtra::grid.arrange(gn12,gn13,gn14,gn15,gn16,ncol=5)

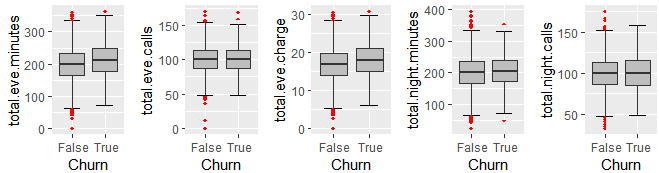
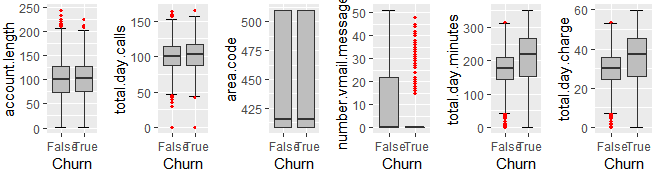
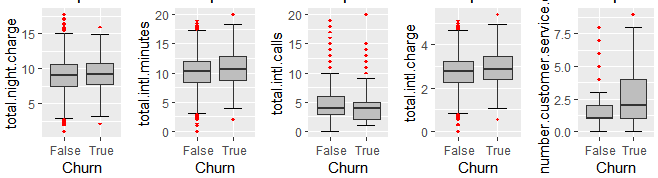
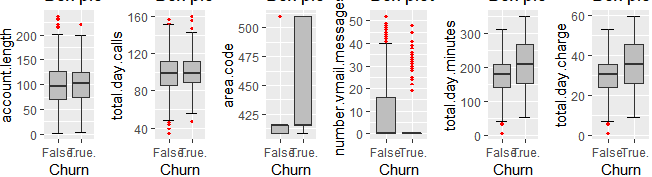
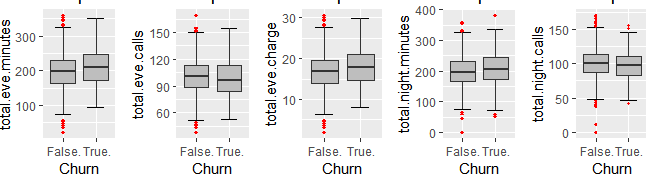
**Fig 2.1.2 Missing values in Test Data Churn score box plots**  

Fig 2.1 Churn score vs Numeric Predictor box plots in test data(R-code shown above)

**Churn score box plots**





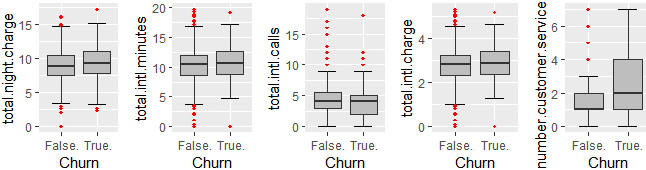


Fig 2.2 Churn score vs Numeric Predictor box plots in test data(R Code shown above)

From the above Fig 2.1 and Fig 2.2 it is clear that not much outliers are present in train and test data of above predicators, so we are not proceeding with outlier analysis.

**2.1.2 Feature Selection**

Before performing any type of modeling we need to assess the importance of each predictor variable in our analysis. There is a possibility that many variables in our analysis are not important at all to the problem of class prediction. There are several methods of doing that. we have both train and test data so performing feature selection only on train data enough to select required predicators on both train and test data. To select required predictors from numeric data we will use **Correlation Analysis** for Numeric variables and **Chi square Test** for categorical variable.

**Correlation Analysis** : Correlation analysis is a method of statistical evaluation used to study the strength of a relationship between two, numerically measured, continuous variables (e.g. height and weight). This particular type of analysis is useful when a researcher wants to establish if there are possible connections between variables. It is often misunderstood that correlation analysis determines cause and effect; however, this is not the case because other variables that are not present in the research may have impacted on the results.

If correlation is found between two variables it means that when there is a systematic change in one variable, there is also a systematic change in the other; the variables alter together over a certain period of time. If there is correlation found, depending upon the numerical values measured, this can be either positive or negative.

**Positive correlation** exists if one variable **increases** simultaneously with the other, i.e. the high numerical values of one variable relate to the high numerical values of the other.

**Negative correlation** exists if one variable **decreases** when the other increases, i.e. the high numerical values of one variable relate to the low numerical values of the other.

**R-Code**

library(corrgram)

numeric\_index= sapply(churn\_train,is.numeric)

numeric\_data=churn\_train[,numeric\_index]

corrgram(churn\_data[,numeric\_index],order=F,upper.panel=panel.pie,text.panel=panel.txt,

main="correlation plot")

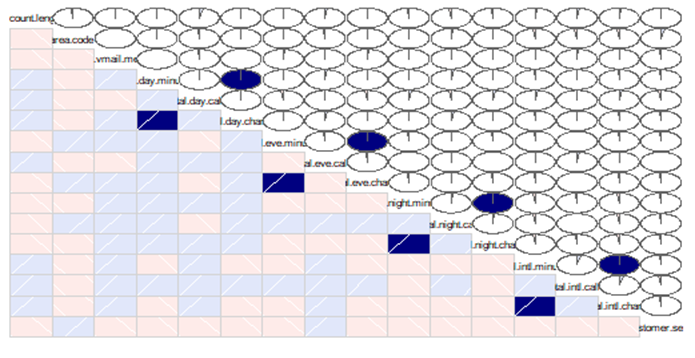
****

Fig 2.3 Correlation Diagram for Numeric predicator.

From the above correlation Diagram we will show 2 positively correlated variables with dark blue and 2 negatively correlated variables with dark red. From the above diagram it’s clear that below variables are highly correlated.

Total day minutes are highly positively correlated with total day charge.

Total eve minutes are highly positively correlated with total eve charge.

Total night minutes are highly positively correlated with total night charge.

Total intl minutes are highly positively correlated with total intl charge.

So we will remove total day charge ,total eve charge, total night charge and total intl charge from our analysis.

## **Chi-Squared Test:** In order to establish that 2 categorical variables are dependent, the chi-squared statistic should be above a certain cutoff. This cutoff increases as the number of classes within the variable increases.

we assume this test as a null hypothesis and an alternate hypothesis.  
The main thing is, we reject the null hypothesis if the p-value that comes out in the result is less than a predetermined significance level, which is 0.05 usually, then we reject the null hypothesis.  
H0: The two variables are independent  
H1: The two variables relate to each other.  
In case of a null hypothesis chi-squared test is to test the two variables that are independent.

We will use above test to check dependency between independent categorical variable and target variable to know the importance of the categorical variable.

**R-Code**

**#chi-square test**

factor\_index=sapply(churn\_train,is.factor)

factor\_data=churn\_train[,factor\_index]

colnames(factor\_data)

for(i in 1:4)

{

print(names(factor\_data[i]))

print(chisq.test(table(factor\_data$Churn,factor\_data[,i])))

}

[1] "state"

Pearson's Chi-squared test

data: table(factor\_data$Churn, factor\_data[, i])

X-squared = 83.044, df = 50, p-value = 0.002296

[1] "phone.number"

Pearson's Chi-squared test

data: table(factor\_data$Churn, factor\_data[, i])

X-squared = 3333, df = 3332, p-value = 0.4919

[1] "international.plan"

Pearson's Chi-squared test with Yates' continuity correction

data: table(factor\_data$Churn, factor\_data[, i])

X-squared = 222.57, df = 1, p-value < 2.2e-16

[1] "voice.mail.plan"

Pearson's Chi-squared test with Yates' continuity correction

data: table(factor\_data$Churn, factor\_data[, i])

X-squared = 34.132, df = 1, p-value = 5.151e-09

**Fig 2.4 chi -square test results**

From the above results it’s clear that phone number has p-value 0.49 greater than 0.05 so we will accept reject null hypothesis saying that phone.number and churn score is independent, so we will remove phone.number from our further analysis.

**2.2 Modeling**

**2.2.1 Model Selection**

In our early stages of analysis during pre-processing we have come to understand phone.number , total.day .charges, total.eve.charges, total.night.charges and total.intl.charges from our modeling.

The dependent variable can fall in either of the four categories:

1. Nominal
2. Ordinal
3. Interval
4. Ratio

Based on the dependent of variable of our dataset we use a model accordingly. There are two types of supervised learning models. They are classification and regression.we choose a type from them depending on the predicting variable. In this case our depending variable is categorical variable we use classifcation models. If the variable is categorical we go classification models. we try different classification models and based on error metrics we will choose best model.

**2.2.2 Decision Tree**

First we will try to use Decision Tree to predict churn score. After 35 trials we are able to predict better churn score. Below is the Decision Tree algorithm in R and it’s summary output .

**#Decision Tree for classification**

churn\_train=read.csv("Train\_data\_c.csv",sep=",")

churn\_test=read.csv("Test\_data\_c.csv",sep=",")

churn\_train\_final=subset(churn\_train,select=-c(phone.number,total.day.charge,total.eve.charge,total.night.charge,total.intl.charge))

churn\_test\_final=subset(churn\_test,select=-c(phone.number,total.day.charge,total.eve.charge,total.night.charge,total.intl.charge))

**##Decision tree for classification**

**#Develop Model on training data**

C50\_model = C5.0(Churn ~., churn\_train\_final, trials = 35, rules = TRUE)

**#Summary of DT model**

summary(C50\_model)

**#OUTPUT**

----- Trial 34: -----

Rules:

Rule 34/1: (25.5, lift 1.6)

total.intl.minutes <= 1.3

-> class False. [0.964]

Rule 34/2: (81.2/6.9, lift 1.5)

international.plan = no

total.day.minutes <= 244.9

total.eve.minutes > 243.3

total.intl.calls > 6

-> class False. [0.905]

Rule 34/3: (3266.6/1312.5, lift 1.0)

total.intl.minutes > 1.3

-> class False. [0.598]

Rule 34/4: (173.7, lift 2.5)

international.plan = yes

total.intl.calls <= 2

-> class True. [0.994]

Rule 34/5: (161.5/1.1, lift 2.5)

voice.mail.plan = no

total.day.minutes > 244.9

total.eve.minutes > 187.9

total.night.minutes > 212.5

-> class True. [0.987]

Rule 34/6: (32, lift 2.5)

voice.mail.plan = no

total.day.minutes > 244.9

total.night.minutes > 287.4

-> class True. [0.971]

Rule 34/7: (28.9, lift 2.5)

total.day.calls > 138

number.customer.service.calls > 3

-> class True. [0.968]

Rule 34/8: (36.4/1.5, lift 2.4)

total.night.calls <= 67

number.customer.service.calls > 3

-> class True. [0.934]

Rule 34/9: (304.4/23.5, lift 2.3)

state in {AL, CA, CT, FL, GA, IA, IN, KS, KY, MA, MD, ME, MI, MN, MO,

MT, NC, NE, NH, NJ, NV, NY, OH, OK, OR, PA, SC, SD, TN, WA,

WI, WV, WY}

voice.mail.plan = no

total.day.minutes > 244.9

total.eve.minutes > 187.9

-> class True. [0.920]

Rule 34/10: (156.4/25.5, lift 2.1)

voice.mail.plan = no

total.day.minutes > 277.5

total.intl.minutes > 1.3

number.customer.service.calls <= 3

-> class True. [0.833]

Rule 34/11: (274.9/53.3, lift 2.0)

state in {AR, CA, CT, IN, MA, MI, MN, MS, MT, NH, NJ, NV, NY, OR, PA,

SC, SD, TN, TX, UT, WI, WV, WY}

total.night.calls <= 132

total.intl.minutes > 1.3

number.customer.service.calls > 3

-> class True. [0.804]

Rule 34/12: (332.9/76.3, lift 2.0)

state in {AR, CA, CO, CT, DE, IN, KS, MD, ME, MI, MN, MS, MT, ND, NE,

NJ, NM, NY, OR, PA, RI, SC, SD, TX, UT, VA, WI, WV}

number.vmail.messages <= 28

total.day.minutes > 159.3

total.eve.minutes > 243.3

total.night.minutes > 154.6

total.intl.minutes > 1.3

total.intl.calls <= 6

number.customer.service.calls <= 3

-> class True. [0.769]

Rule 34/13: (232.9/85.5, lift 1.6)

state in {CT, GA, KS, LA, ME, MI, MS, NC, NE, NJ, NY, OH, SC, UT}

account.length > 84

area.code <= 415

total.day.minutes > 162.8

total.day.minutes <= 244.9

total.eve.minutes <= 243.3

total.night.minutes > 163.7

total.night.calls > 77

total.intl.minutes > 1.3

-> class True. [0.632]

Default class: False.

Evaluation on training data (3333 cases):

Trial Rules

----- ----------------

No Errors

0 19 146( 4.4%)

1 14 345(10.4%)

2 32 312( 9.4%)

3 36 470(14.1%)

4 32 331( 9.9%)

5 12 300( 9.0%)

6 17 371(11.1%)

7 24 344(10.3%)

8 35 420(12.6%)

9 35 277( 8.3%)

10 26 312( 9.4%)

11 19 366(11.0%)

12 27 357(10.7%)

13 25 276( 8.3%)

14 9 224( 6.7%)

15 21 572(17.2%)

16 37 335(10.1%)

17 23 265( 8.0%)

18 23 294( 8.8%)

19 17 403(12.1%)

20 16 390(11.7%)

21 32 249( 7.5%)

22 27 444(13.3%)

23 12 268( 8.0%)

24 34 369(11.1%)

25 32 309( 9.3%)

26 26 242( 7.3%)

27 20 397(11.9%)

28 11 403(12.1%)

29 22 367(11.0%)

30 16 228( 6.8%)

31 15 302( 9.1%)

32 31 458(13.7%)

33 14 227( 6.8%)

34 13 362(10.9%)

boost 45( 1.4%) <<

(a) (b) <-classified as

---- ----

2850 (a): class False.

45 438 (b): class True.

Attribute usage:

100.00% international.plan

100.00% total.day.minutes

100.00% total.eve.minutes

100.00% total.intl.minutes

99.97% number.customer.service.calls

99.67% state

99.34% total.night.minutes

99.22% total.day.calls

98.98% total.intl.calls

97.15% number.vmail.messages

94.99% total.eve.calls

94.60% total.night.calls

92.62% account.length

77.20% voice.mail.plan

73.81% area.code

Time: 2.5 secs

**2.2.3 Logistic Regression**

Now We will use Logistic regression to predict test churn score. Below is logistic regression R code and summary output.

**#Logistic Regression**

**#logit Model**

logit\_model = glm(Churn ~ ., data = churn\_train\_final, family = "binomial")

**#summary of the model**

summary(logit\_model)

Call:

glm(formula = Churn ~ ., family = "binomial", data = churn\_train\_final)

Deviance Residuals:

Min 1Q Median 3Q Max

-1.9140 -0.4980 -0.3121 -0.1660 3.0485

Coefficients:

Estimate Std. Error z value Pr(>|z|)

(Intercept) -9.4898994 1.1513237 -8.243 < 2e-16 \*\*\*

stateAL 0.3356586 0.7641644 0.439 0.660481

stateAR 0.9086542 0.7535213 1.206 0.227865

stateAZ 0.1207686 0.8447303 0.143 0.886316

stateCA 1.8280080 0.7821857 2.337 0.019437 \*

stateCO 0.6761464 0.7625883 0.887 0.375269

stateCT 1.0217643 0.7263012 1.407 0.159485

stateDC 0.6955514 0.8092320 0.860 0.390053

stateDE 0.7623205 0.7495842 1.017 0.309158

stateFL 0.5957627 0.7615817 0.782 0.434056

stateGA 0.6782471 0.7781702 0.872 0.383431

stateHI -0.2124380 0.8956004 -0.237 0.812500

stateIA 0.2362685 0.9032521 0.262 0.793649

stateID 0.8746814 0.7478901 1.170 0.242189

stateIL -0.2060254 0.8332019 -0.247 0.804700

stateIN 0.4437105 0.7538195 0.589 0.556119

stateKS 1.0719442 0.7303969 1.468 0.142208

stateKY 0.8055450 0.7659453 1.052 0.292937

stateLA 0.5654870 0.8359384 0.676 0.498742

stateMA 1.1757655 0.7436069 1.581 0.113840

stateMD 1.1444916 0.7170866 1.596 0.110482

stateME 1.3537005 0.7283085 1.859 0.063071 .

stateMI 1.3878147 0.7141185 1.943 0.051968 .

stateMN 1.1707875 0.7158404 1.636 0.101935

stateMO 0.5992235 0.7747027 0.773 0.439233

stateMS 1.3602477 0.7279699 1.869 0.061686 .

stateMT 1.8703096 0.7173417 2.607 0.009127 \*\*

stateNC 0.6074598 0.7545805 0.805 0.420802

stateND 0.1561239 0.7971095 0.196 0.844718

stateNE 0.3251548 0.8053140 0.404 0.686388

stateNH 1.1918811 0.7684756 1.551 0.120909

stateNJ 1.5950322 0.7097725 2.247 0.024624 \*

stateNM 0.4755184 0.7875852 0.604 0.545998

stateNV 1.2538913 0.7254150 1.729 0.083896 .

stateNY 1.1672256 0.7203608 1.620 0.105160

stateOH 0.6869122 0.7472375 0.919 0.357955

stateOK 0.8831745 0.7542027 1.171 0.241597

stateOR 0.7802912 0.7363023 1.060 0.289262

statePA 1.1597331 0.7799445 1.487 0.137030

stateRI -0.1026449 0.8198389 -0.125 0.900364

stateSC 1.7795155 0.7373544 2.413 0.015805 \*

stateSD 0.8352573 0.7619242 1.096 0.272971

stateTN 0.2826616 0.8213454 0.344 0.730738

stateTX 1.6525874 0.7083195 2.333 0.019642 \*

stateUT 1.0500132 0.7441516 1.411 0.158239

stateVA -0.4347832 0.8228538 -0.528 0.597232

stateVT 0.1013589 0.7784119 0.130 0.896398

stateWA 1.4239736 0.7246427 1.965 0.049406 \*

stateWI 0.2802135 0.7809287 0.359 0.719729

stateWV 0.5856769 0.7334453 0.799 0.424564

stateWY 0.3029853 0.7548802 0.401 0.688149

account.length 0.0009765 0.0014344 0.681 0.496014

area.code -0.0006102 0.0013505 -0.452 0.651402

international.plan yes 2.1880896 0.1532837 14.275 < 2e-16 \*\*\*

voice.mail.plan yes -2.1069708 0.5931036 -3.552 0.000382 \*\*\*

number.vmail.messages 0.0374685 0.0186121 2.013 0.044102 \*

total.day.minutes 0.0131085 0.0011091 11.820 < 2e-16 \*\*\*

total.day.calls 0.0040448 0.0028579 1.415 0.156986

total.eve.minutes 0.0077682 0.0011840 6.561 5.35e-11 \*\*\*

total.eve.calls 0.0009857 0.0028880 0.341 0.732887

total.night.minutes 0.0039274 0.0011513 3.411 0.000646 \*\*\*

total.night.calls 0.0001875 0.0029250 0.064 0.948901

total.intl.minutes 0.0834837 0.0210899 3.958 7.54e-05 \*\*\*

total.intl.calls -0.0899720 0.0256846 -3.503 0.000460 \*\*\*

number.customer.service.calls 0.5367253 0.0409336 13.112 < 2e-16 \*\*\*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 2758.3 on 3332 degrees of freedom

Residual deviance: 2072.0 on 3268 degrees of freedom

AIC: 2202

Number of Fisher Scoring iterations: 6

From above \* indicates significance of that predicator. If it has more stars then it’s more important. And above it

S indicating p-value ,z-score and std deviation of each predicator.

**2.2.4 K Nearest Neighbors**

Below is the KNN classification model and it’s R-code

**#applying Knn classification model**

library(class)

KNN\_Predications=knn(churn\_train\_final\_knn[,1:15],churn\_test\_final\_knn[,1:15],churn\_train\_final\_knn$Churn,k=1)

**2.2.5 Naïve Bayes**

Below is the Naïve bayes classification model and it’s R-code

**#naive Bayes**

library(e1071)

**#Develop model**

NB\_model = naiveBayes(Churn ~ ., data = churn\_train\_final\_b)

**Chapter 3**

# Conclusion

**3.1 Model Evaluation**

Now that we have a few models for predicting the target variable, we need to 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 our case of Churn score i.e., True or False , the latter two, *Interpretability* and *Computation Efficiency*, do not hold much significance. Therefore we will use *Predictive performance* as the criteria 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 error metrics.

* + 1. **Accuracy and False Negative Rate**

We will calculate the Accuracy and Negative Rate for all the models and select model which have high accuracy and low false negative rate.

**Decision Tree:**

##Decision tree for classification

C50\_model = C5.0(Churn ~., churn\_train\_final, trials = 35, rules = TRUE)

#Lets predict for test cases

C50\_Predictions = predict(C50\_model, churn\_test\_final[,-16], type = "class")

##Evaluate the performance of classification model

ConfMatrix\_C50 = table(factor(C50\_Predictions),factor(churn\_test\_final$Churn))

**#accuracy =95.8**

**#False Negative rate =27.67**

**Logistic Regression:**

#Logistic Regression

logit\_model = glm(Churn ~ ., data = churn\_train\_final, family = "binomial")

#predict using logistic regression

logit\_Predictions = predict(logit\_model, newdata = churn\_test\_final, type = "response")

#convert prob

logit\_Predictions = ifelse(logit\_Predictions > 0.5, 1, 0)

##Evaluate the performance of classification model

ConfMatrix\_lt = table(churn\_test\_final$Churn, logit\_Predictions)

**#accuracy=87.04**

**#False negative rate=75.44**

**KNN classification**

KNN\_Predications=knn(churn\_train\_final\_knn[,1:15],churn\_test\_final\_knn[,1:15],churn\_train\_final\_knn$Churn,k=1)

#confusion matrix to calculate error metrics

Conf\_matrix = table(observed = churn\_test\_final\_knn[,16], predicted = KNN\_Predications)

**#Accuray=82.3**

**#False negative rate=63.39**

**Naïve Baise**

NB\_model = naiveBayes(Churn ~ ., data = churn\_train\_final\_b)

#predict on test cases #raw

NB\_Predictions = predict(NB\_model,churn\_test\_final\_b[,0:15], type = 'class')

#Look at confusion matrix

Conf\_matrix = table(predicted = NB\_Predictions,observed = churn\_test\_final\_b[,16])

**#Accuracy 88.12**

**#False Negative Rate=71.43**

* 1. **Model Selection**

From the above all models Decision tree gives high accuracy and low false negative rate on test data set. So we can select Decision Tree model for our dataset.

**4.1 Complete R Code**

rm(list=ls())

setwd("C:/Users/Rajashekar/Videos/Project/Churn")

getwd()

x = c("ggplot2", "corrgram", "DMwR", "caret", "randomForest", "unbalanced", "C50", "dummies", "e1071", "Information","MASS", "rpart", "gbm", "ROSE", 'sampling', 'DataCombine', 'inTrees')

install.packages("corrgram")

lapply(x, require, character.only = TRUE)

library(gridExtra)

#reading data

churn\_data=read.csv("Train\_data.csv",sep = ",")

head(churn\_data[, 1:6]) %>% kable(caption = "churn reduction (Columns: 1-6)",booktabs = TRUE, longtable = TRUE)

head(churn\_data[, 7:15]) %>% kable(caption = "churn reduction (Columns: 7-15)", booktabs = TRUE, longtable = TRUE)

head(churn\_data[, 16:21]) %>% kable(caption = "churn reduction (Columns: 16-21)", booktabs = TRUE, longtable = TRUE)

var <- colnames(churn\_data)[-ncol(churn\_data)]

num <- 1:length(var)

df = data.frame(S.No. = num, Predictor = var)

kable(df, caption = "Predictor Variables", booktabs = TRUE,longtable = TRUE)

#Missing value analysis

#training data

churn\_train=read.csv("Train\_data.csv",sep=",")

missing\_val\_train= data.frame(apply(churn\_train,2,function(x){sum(is.na(x))}))

#test data

churn\_test=read.csv("Test\_data.csv",sep=",")

missing\_val\_test= data.frame(apply(churn\_test,2,function(x){sum(is.na(x))}))

missing\_val\_test

#boxplot

# churn score boxplots for train

library(ggplot2)

num\_index=sapply(churn\_train,is.numeric)

num\_data=churn\_train[,num\_index]

cnames=colnames(num\_data)

for (i in 1:length(cnames))

{

assign(paste0("gn",i), ggplot(aes\_string(y = (cnames[i]), x = 'Churn'), data = churn\_train)+

stat\_boxplot(geom = "errorbar", width = 0.5) +

geom\_boxplot(outlier.colour="red", fill = "grey" ,outlier.shape=18,

outlier.size=1, notch=FALSE) +

theme(legend.position="bottom")+

labs(y=cnames[i],x="Churn")+

ggtitle(paste("Box plot of Churn vs",cnames[i])))

}

gridExtra::grid.arrange(gn1,gn5,gn2,gn3,gn4,gn6,ncol=6)

gridExtra::grid.arrange(gn7,gn8,gn9,gn10,gn11,ncol=5)

gridExtra::grid.arrange(gn12,gn13,gn14,gn15,gn16,ncol=5)

# churn score boxplots for test

library(ggplot2)

num\_index=sapply(churn\_test,is.numeric)

num\_data=churn\_test[,num\_index]

cnames=colnames(num\_data)

for (i in 1:length(cnames))

{

assign(paste0("gn",i), ggplot(aes\_string(y = (cnames[i]), x = 'Churn'), data = churn\_test)+

stat\_boxplot(geom = "errorbar", width = 0.5) +

geom\_boxplot(outlier.colour="red", fill = "grey" ,outlier.shape=18,

outlier.size=1, notch=FALSE) +

theme(legend.position="bottom")+

labs(y=cnames[i],x="Churn")+

ggtitle(paste("Box plot of Churn vs",cnames[i])))

}

gridExtra::grid.arrange(gn1,gn5,gn2,gn3,gn4,gn6,ncol=6)

gridExtra::grid.arrange(gn7,gn8,gn9,gn10,gn11,ncol=5)

gridExtra::grid.arrange(gn12,gn13,gn14,gn15,gn16,ncol=5)

#correlation analysis

library(corrgram)

numeric\_index= sapply(churn\_train,is.numeric)

numeric\_data=churn\_train[,numeric\_index]

corrgram(churn\_data[,numeric\_index],order=F,upper.panel=panel.pie,text.panel=panel.txt,

main="correlation plot")

#chi-square test

factor\_index=sapply(churn\_train,is.factor)

factor\_data=churn\_train[,factor\_index]

colnames(factor\_data)

for(i in 1:4)

{

print(names(factor\_data[i]))

print(chisq.test(table(factor\_data$Churn,factor\_data[,i])))

}

#dimensionality reduction

churn\_data\_train=subset(churn\_data,select=-c(phone.number,total.day.charge,total.eve.charge

,total.night.charge,total.intl.charge))

colnames(churn\_data)

#Decision Tree for classification

#Reading both train and test for DT classification

churn\_train=read.csv("Train\_data\_c.csv",sep=",")

churn\_test=read.csv("Test\_data\_c.csv",sep=",")

churn\_train\_final=subset(churn\_train,select=-c(phone.number,total.day.charge,total.eve.charge,total.night.charge,total.intl.charge))

churn\_test\_final=subset(churn\_test,select=-c(phone.number,total.day.charge,total.eve.charge,total.night.charge,total.intl.charge))

##Decision tree for classification

#Develop Model on training data

C50\_model = C5.0(Churn ~., churn\_train\_final, trials = 35, rules = TRUE)

#Summary of DT model

summary(C50\_model)

#write rules into disk

write(capture.output(summary(C50\_model)), "c50Rules.txt")

#Lets predict for test cases

C50\_Predictions = predict(C50\_model, churn\_test\_final[,-16], type = "class")

##Evaluate the performance of classification model

ConfMatrix\_C50 = table(factor(C50\_Predictions),factor(churn\_test\_final$Churn))

confusionMatrix(ConfMatrix\_C50)

#accuracy =95.8

#False Negative rate =27.67

# removed random forest as ‘state’ attribute has 51 levels and RF can't handle morethan 32 levels

#Logistic Regression

#logit Model

logit\_model = glm(Churn ~ ., data = churn\_train\_final, family = "binomial")

#summary of the model

summary(logit\_model)

#predict using logistic regression

logit\_Predictions = predict(logit\_model, newdata = churn\_test\_final, type = "response")

#convert prob

logit\_Predictions = ifelse(logit\_Predictions > 0.5, 1, 0)

##Evaluate the performance of classification model

ConfMatrix\_lt = table(churn\_test\_final$Churn, logit\_Predictions)

#accuracy=87.04

#False negative rate=75.44

#KNN Classification

#Data reading for KNN

churn\_train\_knn=read.csv("Train\_data.csv",sep=",")

churn\_test\_knn=read.csv("Test\_data.csv",sep=",")

churn\_train\_final\_knn=subset(churn\_train\_knn,select=-c(phone.number,total.day.charge,total.eve.charge,total.night.charge,total.intl.charge))

churn\_test\_final\_knn=subset(churn\_test\_knn,select=-c(phone.number,total.day.charge,total.eve.charge,total.night.charge,total.intl.charge))

#assigning levels to factor predicators

for(i in 1:ncol(churn\_train\_final\_knn)){

if(class(churn\_train\_final\_knn[,i]) == 'factor'){

churn\_train\_final\_knn[,i]=factor(churn\_train\_final\_knn[,i], labels=(1:length(levels(factor(churn\_train\_final\_knn[,i])))))

}

}

for(i in 1:ncol(churn\_test\_final\_knn)){

if(class(churn\_test\_final\_knn[,i]) == 'factor'){

churn\_test\_final\_knn[,i]=factor(churn\_test\_final\_knn[,i], labels=(1:length(levels(factor(churn\_test\_final\_knn[,i])))))

}

}

#converting factor predicators to numeric to apply KNN classification

cnames=sapply(churn\_train\_final\_knn,is.factor)

factor\_data=churn\_train\_final\_knn[,cnames]

factor\_index=colnames(factor\_data)

factor\_index[1]

for(i in 1:length(factor\_index)-1){

churn\_train\_final\_knn[,factor\_index[i]]=as.numeric(churn\_train\_final\_knn[,factor\_index[i]])

}

for(i in 1:length(factor\_index)-1){

churn\_test\_final\_knn[,factor\_index[i]]=as.numeric(churn\_test\_final\_knn[,factor\_index[i]])

}

#applying Knn classification model

library(class)

KNN\_Predications=knn(churn\_train\_final\_knn[,1:15],churn\_test\_final\_knn[,1:15],churn\_train\_final\_knn$Churn,k=1)

knn\_output=cbind(churn\_test\_final\_knn$Churn,KNN\_Predications)

#confusion matrix to calculate error metrics

Conf\_matrix = table(observed = churn\_test\_final\_knn[,16], predicted = KNN\_Predications)

#Accuray=82.3

#False negative rate=63.39

#data read for bayes

churn\_train\_b=read.csv("Train\_data.csv",sep=",")

churn\_test\_b=read.csv("Test\_data.csv",sep=",")

churn\_train\_final\_b=subset(churn\_train\_b,select=-c(phone.number,total.day.charge,total.eve.charge,total.night.charge,total.intl.charge))

churn\_test\_final\_b=subset(churn\_test\_b,select=-c(phone.number,total.day.charge,total.eve.charge,total.night.charge,total.intl.charge))

#naive Bayes

library(e1071)

#Develop model

NB\_model = naiveBayes(Churn ~ ., data = churn\_train\_final\_b)

#predict on test cases #raw

NB\_Predictions = predict(NB\_model,churn\_test\_final\_b[,0:15], type = 'class')

#Look at confusion matrix

Conf\_matrix = table(predicted = NB\_Predictions,observed = churn\_test\_final\_b[,16])

confusionMatrix(Conf\_matrix)

#Accuracy 88.12

#False Negative Rate=71.43

**4.2 Complete Python Code**

#packages for python project

import os

import pandas as pd

import seaborn as sns

from random import randrange, uniform

import numpy as np

import matplotlib.pyplot as plt

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

from sklearn.tree import DecisionTreeRegressor

from sklearn import metrics

from scipy.stats import chi2\_contingency

os.chdir("C:/Users/Rajashekar/Videos/project/Churn")

os.getcwd()

#reading train data set

churn\_train=pd.read\_csv("Train\_data.csv",sep=",")

#numeric columns for correlation analysis

numeric\_cnames=['account length', 'area code','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']

#Correlation analysis

churn\_corr=churn\_train.loc[:,numeric\_cnames]

#below is correlation matrix

f, ax = plt.subplots(figsize=(7, 5))

#Generate correlation matrix

corr = churn\_corr.corr()

#Plot using seaborn library

sns.heatmap(corr, mask=np.zeros\_like(corr, dtype=np.bool), cmap=sns.diverging\_palette(220, 10, as\_cmap=True),

square=True, ax=ax)

# categorical variables

cat\_cnames=['state','phone number','international plan', 'voice mail plan','Churn']

#chi-square analysis for categorical variables

for i in cat\_cnames:

#print(i)

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

#print(p,chi2,dof)

#removal cnames

removal\_cnames= ['phone number','total day charge','total eve charge','total night charge',

'total intl charge']

#reading data for Decision Tree analysis

churn\_train\_dt=pd.read\_csv("Train\_data.csv",sep=",")

churn\_test\_dt=pd.read\_csv("Test\_data.csv",sep=",")

churn\_train\_final\_dt=churn\_train\_dt.drop(removal\_cnames,axis=1)

churn\_test\_final\_dt=churn\_test\_dt.drop(removal\_cnames,axis=1)

#Import Libraries for decision tree

from sklearn import tree

from sklearn.metrics import accuracy\_score

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

#replace target categories with Yes or No

churn\_train\_final\_dt['Churn'] = churn\_train\_final\_dt['Churn'].replace(' False', 'No')

churn\_train\_final\_dt['Churn'] = churn\_train\_final\_dt['Churn'].replace(' True', 'Yes')

churn\_test\_final\_dt['Churn'] = churn\_test\_final\_dt['Churn'].replace( ' True', 'Yes')

churn\_test\_final\_dt['Churn'] = churn\_test\_final\_dt['Churn'].replace( ' False', 'No')

#objects types are not supported by decision tree so converting object types (Training Dataset)decision tree required types

from sklearn import preprocessing

le = preprocessing.LabelEncoder()

for column\_name in churn\_train\_final\_dt.columns:

if churn\_train\_final\_dt[column\_name].dtype == object:

churn\_train\_final\_dt[column\_name] = le.fit\_transform(churn\_train\_final\_dt[column\_name])

else:

pass

#objects types are not supported by decision tree so converting object types (Test Dataset)decision tree required types

from sklearn import preprocessing

le = preprocessing.LabelEncoder()

for column\_name in churn\_test\_final\_dt.columns:

if churn\_test\_final\_dt[column\_name].dtype == object:

churn\_test\_final\_dt[column\_name] = le.fit\_transform(churn\_test\_final\_dt[column\_name])

else:

pass

#Applying DT regression

C50\_model = tree.DecisionTreeClassifier(criterion='entropy').fit(churn\_train\_final\_dt.iloc[:,0:15],churn\_train\_final\_dt.iloc[:,15])

#predict new test cases

C50\_Predictions = C50\_model.predict(churn\_test\_final\_dt.iloc[:,0:15])

#Create dot file to visualise tree #http://webgraphviz.com/

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

df = tree.export\_graphviz(C50\_model, out\_file=dotfile, feature\_names =churn\_train\_final\_dt.iloc[:,0:15].columns)

#build confusion matrix

CM = pd.crosstab(churn\_test\_final\_dt.iloc[:,15], C50\_Predictions)

#df=pd.DataFrame(churn\_test\_final\_dt.iloc[:,15], C50\_Predictions)

#let us save TP, TN, FP, FN

TN = CM.iloc[0,0]

FN = CM.iloc[1,0]

TP = CM.iloc[1,1]

FP = CM.iloc[0,1]

#check accuracy of model

#accuracy\_score(y\_test, y\_pred)\*100

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

#False Negative rate

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

#Results

Accuracy: 91.72165566886623

FNR: 31.25

#Random Forest

from sklearn.ensemble import RandomForestClassifier

RF\_model = RandomForestClassifier(n\_estimators = 275).fit(churn\_train\_final\_dt.iloc[:,0:15],churn\_train\_final\_dt.iloc[:,15])

RF\_Predictions = RF\_model.predict(churn\_test\_final\_dt.iloc[:,0:15])

#build confusion matrix

CM = pd.crosstab(churn\_test\_final\_dt.iloc[:,15], RF\_Predictions)

#let us save TP, TN, FP, FN

TN = CM.iloc[0,0]

FN = CM.iloc[1,0]

TP = CM.iloc[1,1]

FP = CM.iloc[0,1]

#check accuracy of model

#accuracy\_score(y\_test, y\_pred)\*100

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

#False Negative rate

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

#Results

Accuracy:95.62087582483504

FNR: 29.910714285714285

#Create logistic data. Save target variable first

churn\_train\_dt=pd.read\_csv("Train\_data.csv",sep=",")

churn\_test\_dt=pd.read\_csv("Test\_data.csv",sep=",")

churn\_train\_logit = pd.DataFrame(churn\_train\_dt['Churn'])

churn\_test\_logit = pd.DataFrame(churn\_test\_dt['Churn'])

cnames=['account length', 'area code', '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']

#Add continous variables

churn\_train\_logit = churn\_train\_logit.join(churn\_train\_dt[cnames])

churn\_test\_logit = churn\_test\_logit.join(churn\_test\_dt[cnames])

##Create dummies for categorical variables

cat\_names = ['state','international plan','voice mail plan',]

for i in cat\_names:

temp = pd.get\_dummies(churn\_train\_dt[i], prefix = i)

churn\_train\_logit = churn\_train\_logit.join(temp)

for i in cat\_names:

temp = pd.get\_dummies(churn\_test\_dt[i], prefix = i)

churn\_test\_logit = churn\_test\_logit.join(temp)

#objects types are not supported by decision tree so converting object types (Training Dataset)decision tree required types

from sklearn import preprocessing

le = preprocessing.LabelEncoder()

for column\_name in churn\_train\_logit.columns:

if churn\_train\_logit[column\_name].dtype == object:

churn\_train\_logit[column\_name] = le.fit\_transform(churn\_train\_logit[column\_name])

else:

pass

#objects types are not supported by decision tree so converting object types (Test Dataset)decision tree required types

from sklearn import preprocessing

le = preprocessing.LabelEncoder()

for column\_name in churn\_test\_logit.columns:

if churn\_test\_logit[column\_name].dtype == object:

churn\_test\_logit[column\_name] = le.fit\_transform(churn\_test\_logit[column\_name])

else:

pass

#select column indexes for independent variables

train\_cols = churn\_train\_logit.columns[1:68]

test\_cols = churn\_test\_logit.columns[1:68]

#Built Logistic Regression

import statsmodels.api as sm

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

logit.summary()

#Predict test data

churn\_test\_logit['Actual\_prob'] = logit.predict(churn\_test\_logit[test\_cols])

churn\_test\_logit['Actual\_Val'] = 1

churn\_test\_logit.loc[churn\_test\_logit.Actual\_prob < 0.5, 'Actual\_Val'] = 0

#Build confusion matrix

CM = pd.crosstab(churn\_test\_logit['Churn'], churn\_test\_logit['Actual\_Val'])

#let us save TP, TN, FP, FN

TN = CM.iloc[0,0]

FN = CM.iloc[1,0]

TP = CM.iloc[1,1]

FP = CM.iloc[0,1]

accuracy of model=86.98260347930415

False Negative Score=75.44642857142857

#reading data for KNN analysis

churn\_train\_dt=pd.read\_csv("Train\_data.csv",sep=",")

churn\_test\_dt=pd.read\_csv("Test\_data.csv",sep=",")

churn\_train\_final\_dt=churn\_train\_dt.drop(removal\_cnames,axis=1)

churn\_test\_final\_dt=churn\_test\_dt.drop(removal\_cnames,axis=1)

#replace target categories with Yes or No

churn\_train\_final\_dt['Churn'] = churn\_train\_final\_dt['Churn'].replace(' False', 'No')

churn\_train\_final\_dt['Churn'] = churn\_train\_final\_dt['Churn'].replace(' True', 'Yes')

churn\_test\_final\_dt['Churn'] = churn\_test\_final\_dt['Churn'].replace( ' True', 'Yes')

churn\_test\_final\_dt['Churn'] = churn\_test\_final\_dt['Churn'].replace( ' False', 'No')

#objects types are not supported by decision tree so converting object types (Training Dataset)decision tree required types

from sklearn import preprocessing

le = preprocessing.LabelEncoder()

for column\_name in churn\_train\_final\_dt.columns:

if churn\_train\_final\_dt[column\_name].dtype == object:

churn\_train\_final\_dt[column\_name] = le.fit\_transform(churn\_train\_final\_dt[column\_name])

else:

pass

#objects types are not supported by decisiontree so converting object types (Test Dataset)decision tree required types

from sklearn import preprocessing

le = preprocessing.LabelEncoder()

for column\_name in churn\_test\_final\_dt.columns:

if churn\_test\_final\_dt[column\_name].dtype == object:

churn\_test\_final\_dt[column\_name] = le.fit\_transform(churn\_test\_final\_dt[column\_name])

else:

pass

#KNN implementation

from sklearn.neighbors import KNeighborsClassifier

KNN\_model = KNeighborsClassifier(n\_neighbors=1).fit(churn\_train\_final\_dt.iloc[:,0:15],churn\_train\_final\_dt.iloc[:,15])

#predict test cases

KNN\_Predictions = KNN\_model.predict(churn\_test\_final\_dt.iloc[:,0:15])

#build confusion matrix

CM = pd.crosstab(churn\_test\_final\_dt.iloc[:,15], KNN\_Predictions)

#let us save TP, TN, FP, FN

TN = CM.iloc[0,0]

FN = CM.iloc[1,0]

TP = CM.iloc[1,1]

FP = CM.iloc[0,1]

Accuracy: 82.30353929214156

FNR: 63.392857142857146

#Naive Bayes Classification

#Naive Bayes

from sklearn.naive\_bayes import GaussianNB

#Naive Bayes implementation

NB\_model = GaussianNB().fit(churn\_train\_final\_dt.iloc[:,0:15],churn\_train\_final\_dt.iloc[:,15])

#predict test cases

NB\_Predictions = NB\_model.predict(churn\_test\_final\_dt.iloc[:,0:15],)

#Build confusion matrix

CM = pd.crosstab(churn\_test\_final\_dt.iloc[:,15], NB\_Predictions)

#let us save TP, TN, FP, FN

TN = CM.iloc[0,0]

FN = CM.iloc[1,0]

TP = CM.iloc[1,1]

FP = CM.iloc[0,1]

Accuracy: 85.84283143371326

FNR: 60.267857142857146