**USE CASE STUDY REPORT**

**Group No**.: Group 22

**Student Names**: Harikrishnan Shajil and Saumya Vora

Executive Summary:

The goal of the study was to devise a supervised machine learning algorithm based on prescriptive analytics to predict whether a customer will churn or not.

Telecommunication service providers usually track their customers based on their demographics (area), services signed up such as internet, TV streaming etc. and last but not the least their account charges/billing. These are used to predict behavior of different customers and could there help in deriving insights about them. In terms of data collection, we had extended our search in the Kaggle repositories.

These steps have been taken in the Case Study respectively:

1. Data Preprocessing to eliminate redundant data
2. Data Reduction & Transformation to generate information rich structured data.
3. Exploratory Data Analysis to derive patterns and insights.
4. Data Mining using Machine Learning Algorithms
5. Model Evaluation to compare and pick the best possible classifying model.
6. Interpretation and Course of Action

Data mining techniques used were K- nearest neighbor, Naïve Bayes, CART, Logistic Regression, Artificial Neural Network, Linear Discriminant Analysis, Support Vector Machine. Lift chart, Decile wise Lift chart, Confusion matrix and ROC curve was used to obtain performance measure for each model. Random Forest was observed to the best classifying model for the given dataset.

The Random Forest Algorithm provides us with the best accuracy which is 78.35%.

After classifying customers accurately, the future course of action for customers who are most likely to churn will be to provide them with Priority mails, offers on mobile plans, Discount on calling rates etc.

# I. Background and Introduction

Telecom provider’s main concern is “Whether the customer will churn (discontinue phone service)?”Based on the massive dataset, it is extremely difficult to predict individual customer behavior and provide a solution for so by manual statistical computations.

We have taken the initiative to provide a solution to this problem by possibly predicting customer behavior and thereby classifying their churn outcomes accurately using

Prescriptive Analytics.

After classifying customers accurately, the future course of action for customers who are most likely to churn will be to provide them with Priority mails, Offers on mobile plans, Discount on calling rates etc.

# II. Data Exploration and Visualization

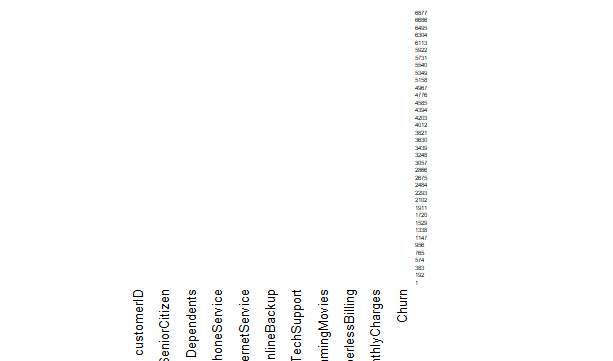
**Distribution plot**

preprocessing part, we are going to remove the missing values rows.

|  |  |
| --- | --- |
| 1.Total Churn Count    Total Number of Customers changing their telecom company. Nearly 2000 people are churning which is approximately 1/3 of the total customers which is a serious threat for company. | 2.Gender, Senior Citizen, Partner, Dependents    Churning based on the Gender, Senior Citizen, Partner and Dependents. Both the gender shows equal churning. Senior Citizens ,Having Partner and dependents shows less churning. |
| 3.Phoneservice, Multiple lines, Online Security, Internet Service    Customers with No internet service and No Online Security are less likely to churn. | 4.Online Backup, Device Protection, Tech Support, Streaming TV    Customers who have Online Backup, Device Protection,Tech Support are less likely to churn. |
| Customers who have opted for one and two year contract ,No paperless billing and are not churning much. | Based on Monthly Charges paid by customers if the range is between 25 to 85 the customers are not churning. |

**Missing Values**

Heatmap clearly shows that really a smaller number of missing data. In the data



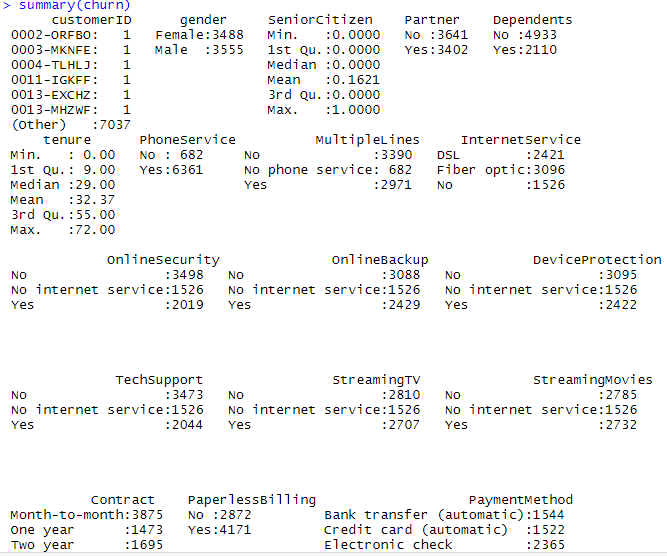
**Rescaling**

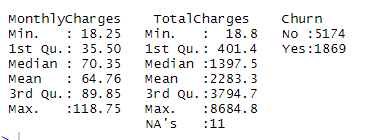
It is only done for Monthly Charges and Total Charges in Artificial Neural Networks.

# III. Data Preparation and Preprocessing

**Data Summary**

The Original Dataset Summary



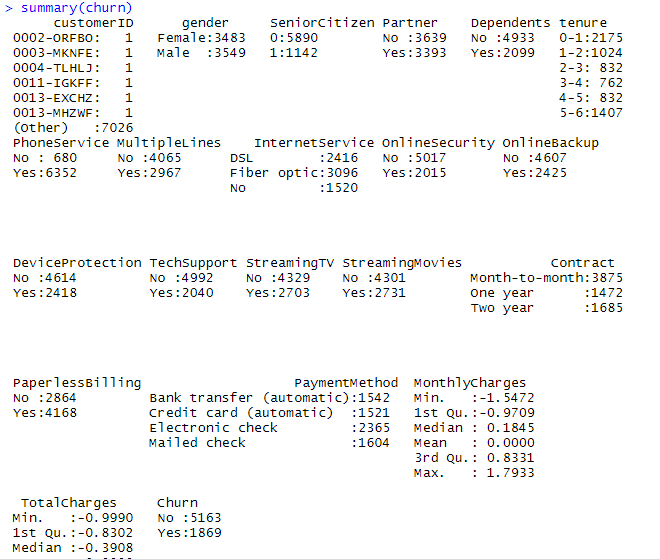


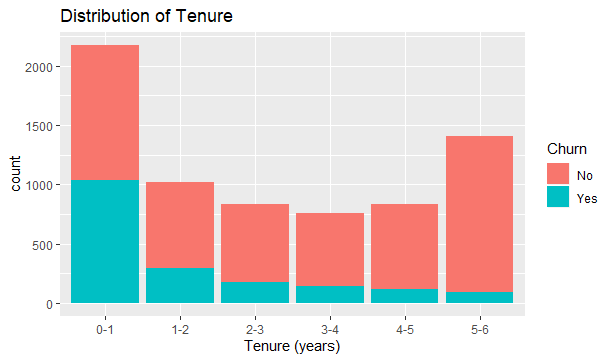
**Dimension Reduction**

**Variable Converting**

1. Removing 11 missing values rows.
2. Applied the function to convert all the rows containing values“ No phone service” to “No”. Other function to convert all the rows containing values “No internet service” to “No”. These functions are applied to have uniformity in the all categorical values of the columns “PhoneService” and “MultipleLines”.
3. Converting Months to Years for the “Tenure” Variable.
4. Binned Tenure variable from 0-1,1-2,2-3,3-4,4-5,5-6.
5. Standardizing columns Monthly Charges and Total Charge
6. Creating Dummy variables for “Tenure”, ”Monthly Charges” and “Total Charges”.

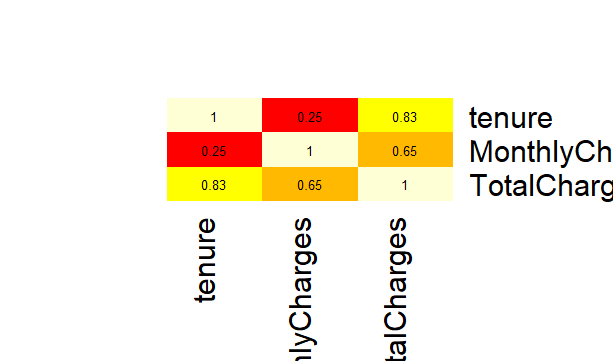
**Visualization after Preprocessing**





**Variable Selection**

All variables except “Customer ID” are selected.

**Correlation Analysis**

Tenure and Monthly Charges are correlated by 0.25.

Tenure and Total Charges are correlated by 0.83

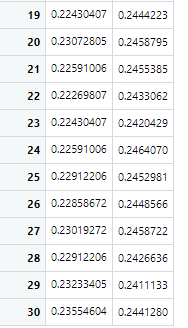
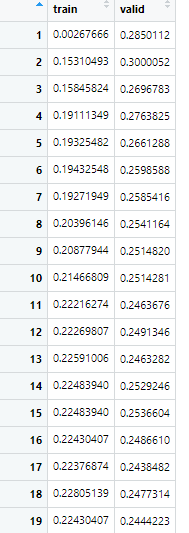
Monthly Charges and Total Charges are correlated by 0.65

Since Tenure is highly correlated with total Charges ,Tenure is binned and converted to categorical variable.

# IV. Data Mining Techniques and Implementation

MODEL 1: K-Nearest Neighbors (K-NN)

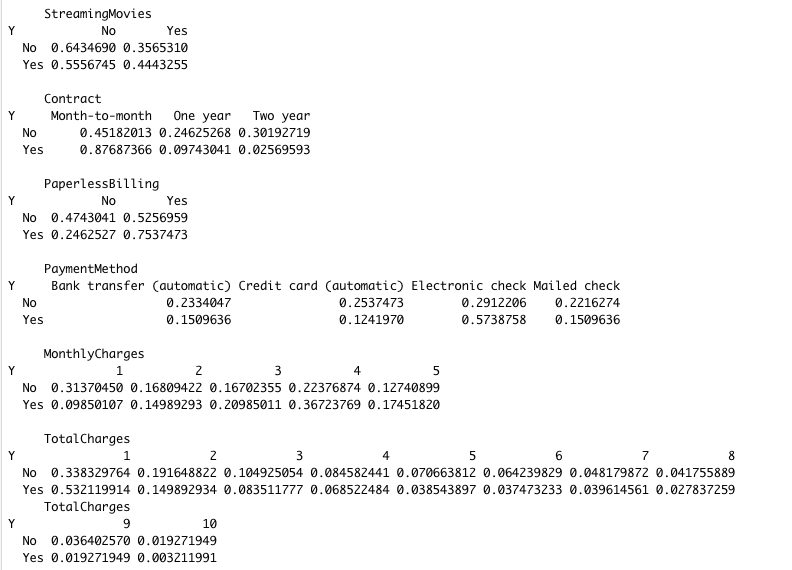
Knn model was applied to the dataset using both oversampled and normally sampled data. We notice that error decreases in the validation set till K=23 and starts increasing again. Thus, the best K is 23. Error at K=23 is 0.2420429.



While comparing oversampled data with normally sampled data, accuracy is 74.33 % for normally sampled and 70.92% for oversampled data. However, sensitivity is 85.53% for oversampled and 69.20% for normally sampled. Thus giving balanced accuracy of 74.94% for over sampled and 73.17% for normally sampled.

MODEL 2: Naïve Bayes

|  |  |
| --- | --- |
|  |  |

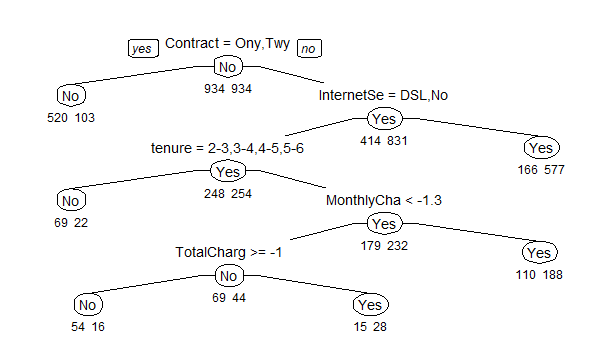


While comparing oversampled data with normally sampled data, accuracy is 74.23 % for normally sampled and 74.79% for oversampled data. However, sensitivity is 78.50% for oversampled and 73.26% for normally sampled. Thus giving balanced accuracy of 75.98% for over sampled and 74.01% for normally sampled.

MODEL 3: CART-Random Forest-Boosted Trees

Classification Tree

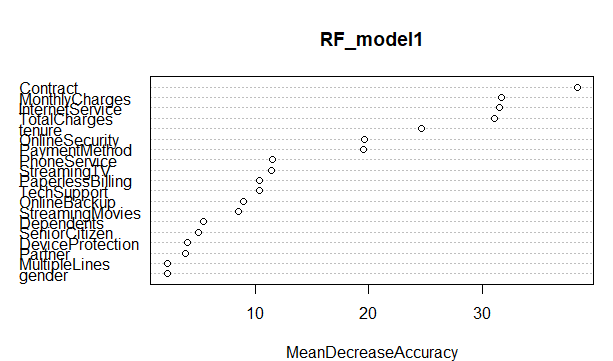
CART algorithm gives the following rules:



Pruning of the classification tree does not improve the performance.

While comparing oversampled data with normally sampled data, accuracy is 76.32 % for normally sampled and 75.53% for oversampled data. However, sensitivity is 75.51% for oversampled and 60.42% for normally sampled. Thus giving balanced accuracy of 75.53% for over sampled and 72.73% for normally sampled.

Random Forest:

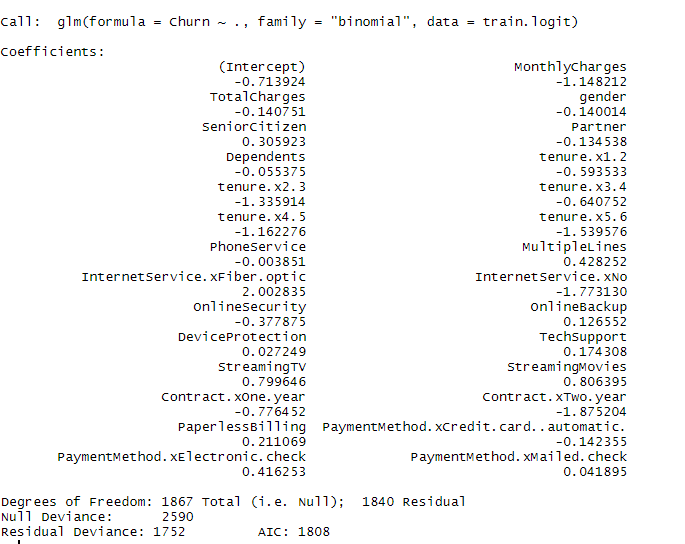


While comparing oversampled data with normally sampled data, accuracy is 78.04 % for normally sampled and 78.35% for oversampled data. However, sensitivity is 76.26% for oversampled and 61.07% for normally sampled. Thus giving balanced accuracy of 77.68% for over sampled and 74.21% for normally sampled.

Boosted Trees:

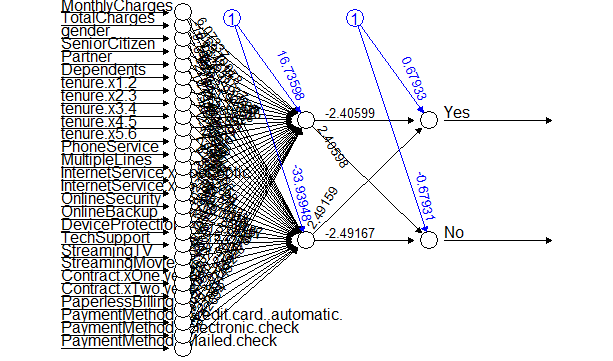
While comparing oversampled data with normally sampled data, accuracy is 76.74 % for normally sampled and 75.62% for oversampled data. However, sensitivity is 76.47% for oversampled and 62.12% for normally sampled. Thus giving balanced accuracy of 75.89% for over sampled and 73.44% for normally sampled.

MODEL 4: Logistic Regression



While comparing oversampled data with normally sampled data, accuracy is 76.28 % for normally sampled and 73.71% for oversampled data. However, sensitivity is 78.82% for oversampled and 63.04% for normally sampled. Thus giving balanced accuracy of 75.34% for over sampled and 73.29% for normally sampled.

MODEL 5: Artificial Neural Network



Since monthly charges and total charges have right skewed distribution we transform them using square root and cube root respectively.

|  |  |
| --- | --- |
|  |  |
|  |  |

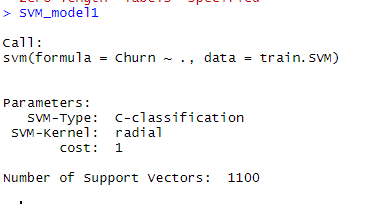
While comparing oversampled data with normally sampled data, accuracy is 76.28 % for normally sampled and 75.02% for oversampled data. However, sensitivity is 77.11% for oversampled and 62.25% for normally sampled. Thus giving balanced accuracy of 75.69% for over sampled and 73.11% for normally sampled.

MODEL 6: Linear Discriminant Analysis

|  |  |
| --- | --- |
|  |  |

While comparing oversampled data with normally sampled data, accuracy is 75.63 % for normally sampled and 73.88% for oversampled data. However, sensitivity is 80.11% for oversampled and 63.30% for normally sampled. Thus giving balanced accuracy of 75.87% for over sampled and 72.84% for normally sampled.

MODEL 7: Support Vector Machine



While comparing oversampled data with normally sampled data, accuracy is 76.00 % for normally sampled and 75.36% for oversampled data. However, sensitivity is 78.50% for oversampled and 64.09% for normally sampled. Thus giving balanced accuracy of 76.36% for over sampled and 73.31% for normally sampled.

# V. Performance Evaluation

MODEL 1: K-Nearest Neighbors (K-NN)

|  |  |
| --- | --- |
|  |  |

MODEL 2: Naïve Bayes

|  |  |
| --- | --- |
|  |  |
|  |  |

MODEL 3: CART-Random Forest-Boosted Trees

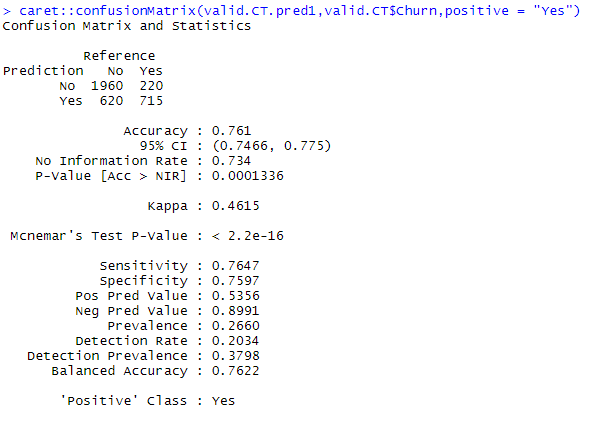
Classification Tree

|  |  |
| --- | --- |
|  |  |
|  |  |

Random Forest

|  |  |
| --- | --- |
|  |  |
|  |  |

Boosted Trees



MODEL 4: Logistic Regression

|  |  |
| --- | --- |
|  |  |
|  |  |

MODEL 5: Artificial Neural Network

|  |  |
| --- | --- |
|  |  |
|  |  |

MODEL 6: Linear Discriminant Analysis

|  |  |
| --- | --- |
|  |  |
|  |  |

MODEL 7: Support Vector Machine

|  |  |
| --- | --- |
|  |  |

# VI. Discussion and Recommendation

From the performance evaluation of models discussed above, we observe that Random Forest gives the highest predictive accuracy followed by Boosted Trees. Thus, for the prediction of telecom churn rate, we can successfully implement Random Forest model and predict churn rate with a confidence of 80%. The models have been trained on oversampled data exposing them to higher positive cases thus providing us with a higher sensitivity and balanced accuracy.

Finally, cases predicted as “Churn” are to be analyzed further and provided with a course of action such as offers on mobile plans, discounts and priority emails.

# VII. Summary

The case study mainly aims at solving one of the most common business problems faced in many industries, specially telecommunication. The objective of this study is to use prescriptive analytics to model a successful algorithm and recommend the best course of action. This case study involves all methods in data mining from exploration and preprocessing to choosing the best possible model. Finally, we conclude that Random Forest is the best method to model the dataset along with oversampling and the cases which are predicted as “Churn” are to be considered for further analysis and thus provided with the best course of action.

# Appendix: R Code for use case study

.

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#Preprocessing & Visualization

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#Importing Data

churn <- read.csv("new churn.csv")

str(churn)

summary(churn)

#-------------------------

#data exploration & visualization

library(ggplot2)

library(dplyr)

library(gplots)

#To detect missing values

heatmap(1\*is.na(churn),Rowv = NA,Colv = NA)

#To detect missing values in corresponding rows/columns

lapply(churn,function(x) which(is.na(x)))

#Deleting observations with missing values

churn <- churn[complete.cases(churn),]

#To detect correlation among numerical variables

corr <- cor(churn[,c("tenure","MonthlyCharges","TotalCharges")])

gplots::heatmap.2(corr, Rowv = FALSE, Colv = FALSE, dendrogram = "none",cellnote = round(corr,2),notecol = "black", key = FALSE, trace = 'none', margins = c(10,10))

#plots

ggplot2::ggplot(churn)+geom\_bar(mapping = aes(x = Churn,fill = Churn))+ggtitle("Churn Count")

ggplot2::ggplot(churn)+geom\_boxplot(mapping = aes(y=MonthlyCharges))+ggtitle("Boxplot of Monthly Charges")

ggplot2::ggplot(churn)+geom\_boxplot(mapping = aes(y=TotalCharges))+ggtitle("Boxplot of Total Charges")

ggplot2::ggplot(churn)+geom\_boxplot(mapping = aes(y=tenure))+ggtitle("Boxplot of Tenure")

ggplot2::ggplot(churn)+geom\_bar(mapping = aes(tenure,fill = tenure))+xlab("Tenure (Month)")+ggtitle("Distribution of Tenure")

g <- ggplot2::ggplot(churn)+geom\_bar(mapping = aes(x=gender,fill = Churn))

s <- ggplot2::ggplot(churn)+geom\_bar(mapping = aes(x=SeniorCitizen,fill = Churn))

p <- ggplot2::ggplot(churn)+geom\_bar(mapping = aes(x=Partner,fill = Churn))

d <- ggplot2::ggplot(churn)+geom\_bar(mapping = aes(x=Dependents,fill = Churn))

gridExtra::grid.arrange(g,s,p,d)

ps <- ggplot2::ggplot(churn)+geom\_bar(mapping = aes(x=PhoneService,fill = Churn))

ml <- ggplot2::ggplot(churn)+geom\_bar(mapping = aes(x=MultipleLines,fill = Churn))

is <- ggplot2::ggplot(churn)+geom\_bar(mapping = aes(x=InternetService,fill = Churn))

os <- ggplot2::ggplot(churn)+geom\_bar(mapping = aes(x=OnlineSecurity,fill = Churn))

gridExtra::grid.arrange(ps,ml,is,os)

ob <- ggplot2::ggplot(churn)+geom\_bar(mapping = aes(x=OnlineBackup,fill = Churn))

dp <- ggplot2::ggplot(churn)+geom\_bar(mapping = aes(x=DeviceProtection,fill = Churn))

ts <- ggplot2::ggplot(churn)+geom\_bar(mapping = aes(x=TechSupport,fill = Churn))

st <- ggplot2::ggplot(churn)+geom\_bar(mapping = aes(x=StreamingTV,fill = Churn))

gridExtra::grid.arrange(ob,dp,ts,st)

sm <- ggplot2::ggplot(churn)+geom\_bar(mapping = aes(x=StreamingMovies,fill = Churn))

c <- ggplot2::ggplot(churn)+geom\_bar(mapping = aes(x=Contract,fill = Churn))

pb <- ggplot2::ggplot(churn)+geom\_bar(mapping = aes(x=PaperlessBilling,fill = Churn))

pm <- ggplot2::ggplot(churn)+geom\_bar(mapping = aes(x=PaymentMethod,fill = Churn))

gridExtra::grid.arrange(sm,c,pb,pm)

ggplot2::ggplot(churn)+geom\_boxplot(mapping = aes(y = MonthlyCharges,fill = Churn))

ggplot2::ggplot(churn)+geom\_boxplot(mapping = aes(y = TotalCharges,fill = Churn))

ggplot2::ggplot(churn)+geom\_histogram(mapping = aes(x=MonthlyCharges))

ggplot2::ggplot(churn)+geom\_histogram(mapping = aes(x=TotalCharges))

#-------------------------

#Preprocessing

#Function to change 'No Phone/Internet Service to No'

sub1 <- function(x){

gsub("No phone service","No",x)

}

sub2 <- function(x){

gsub("No internet service","No",x)

}

#Applying function sub to data frame

churn <- data.frame(lapply(churn, sub1))

churn <- data.frame(lapply(churn, sub2))

#Converting factor to numeric

churn$tenure <- as.numeric(as.character(churn$tenure))

churn$MonthlyCharges <- as.numeric(as.character(churn$MonthlyCharges))

churn$TotalCharges <- as.numeric(as.character(churn$TotalCharges))

#Function to convert months to years

conv <- function(x){

x/12

}

churn$tenure <- sapply(churn$tenure,conv)

#Binning tenure

churn$tenure[churn$tenure >= 0 & churn$tenure <=1] = '0-1'

churn$tenure[churn$tenure > 1 & churn$tenure <=2] = '1-2'

churn$tenure[churn$tenure > 2 & churn$tenure <=3] = '2-3'

churn$tenure[churn$tenure > 3 & churn$tenure <=4] = '3-4'

churn$tenure[churn$tenure > 4 & churn$tenure <=5] = '4-5'

churn$tenure[churn$tenure > 5 & churn$tenure <=6] = '5-6'

churn$tenure <- as.factor(churn$tenure)

#Standardizing columns Monthly Charges and Total Charges

churn[,c('MonthlyCharges','TotalCharges')] = scale(churn[,c('MonthlyCharges','TotalCharges')])

#Visualization after preprocessing

ggplot2::ggplot(churn)+geom\_bar(mapping = aes(tenure,fill = Churn))+xlab("Tenure (years)")+ggtitle("Distribution of Tenure")

#-------------------------

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#KNN

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#Importing Data

churn <- read.csv("new churn.csv")

str(churn)

summary(churn)

#-------------------------

#Preprocessing

#Deleting observations with missing values

churn <- churn[complete.cases(churn),]

#Function to change 'No Phone/Internet Service to No'

sub1 <- function(x){

gsub("No phone service","No",x)

}

sub2 <- function(x){

gsub("No internet service","No",x)

}

#Applying function sub to data frame

churn <- data.frame(lapply(churn, sub1))

churn <- data.frame(lapply(churn, sub2))

#Converting factor to numeric

churn$tenure <- as.numeric(as.character(churn$tenure))

churn$MonthlyCharges <- as.numeric(as.character(churn$MonthlyCharges))

churn$TotalCharges <- as.numeric(as.character(churn$TotalCharges))

#Function to convert months to years

conv <- function(x){

x/12

}

churn$tenure <- sapply(churn$tenure,conv)

#Binning tenure

churn$tenure[churn$tenure >= 0 & churn$tenure <=1] = '0-1'

churn$tenure[churn$tenure > 1 & churn$tenure <=2] = '1-2'

churn$tenure[churn$tenure > 2 & churn$tenure <=3] = '2-3'

churn$tenure[churn$tenure > 3 & churn$tenure <=4] = '3-4'

churn$tenure[churn$tenure > 4 & churn$tenure <=5] = '4-5'

churn$tenure[churn$tenure > 5 & churn$tenure <=6] = '5-6'

churn$tenure <- as.factor(churn$tenure)

#Standardizing columns Monthly Charges and Total Charges

churn[,c('MonthlyCharges','TotalCharges')] = scale(churn[,c('MonthlyCharges','TotalCharges')])

#-------------------------

#Partioning Data

#Original ratio

set.seed(123)

or <- sum(churn$Churn == "Yes")/sum(churn$Churn == "No")

churn.yes.index <- churn$Churn == "Yes"

churn.no.index <- churn$Churn == "No"

churn.yes.df <- churn[churn.yes.index,]

churn.no.df <- churn[churn.no.index,]

#Training/Validation

#Yes

train.yes.index <- sample(c(1:dim(churn.yes.df)[1]),dim(churn.yes.df)[1]/2)

train.yes.df <- churn.yes.df[train.yes.index,]

valid.yes.df <- churn.yes.df[-train.yes.index,]

#No

train.no.index <- sample(c(1:dim(churn.no.df)[1]),dim(churn.yes.df)[1]/2)

train.no.df <- churn.no.df[train.no.index,]

valid.no.df <- churn.no.df[-train.no.index,]

valid.no.index <- sample(c(1:dim(valid.no.df)[1]),(dim(train.yes.df)[1]/or))

valid.no.df <- churn.no.df[valid.no.index,]

#Combining Train/Valid

train.df <- rbind(train.yes.df,train.no.df)

valid.df <- rbind(valid.yes.df,valid.no.df)

#-------------------------

#KNN

#Dummy Variable for KNN

#m dummies

#Function to create dummy variable

dum\_knn <- function(x){

model.matrix(~x-1,data = churn)

}

#Categorical Columns & Numerical Columns

cat <- churn[,-c(1,19,20,21)]

num <- churn[,c(1,19,20,21)]

#Creating Dummy Variables

dummy <- data.frame(sapply(cat, dum\_knn))

#Combining variables to final dataset

churn.knn <- cbind(num,dummy)

str(churn.knn)

#-------------------------

#Oversampling

train.knn <- churn.knn[rownames(train.df),]

valid.knn <- churn.knn[rownames(valid.df),]

#Sampling

churn.sam <- rbind(train.knn,valid.knn)

train.index <- sample(c(1:dim(churn.sam)[1]),0.60\*dim(churn.sam)[1])

train.sam <- churn.sam[train.index,]

valid.sam <- churn.sam[-train.index,]

#-------------------------

#KNN

library(class)

i <- 1

error <- data.frame(matrix(ncol = 2,nrow = 0))

error\_name <- c("train","valid")

colnames(error) <- error\_name

while (i<=30) {

print(i)

knn\_model1 <- class::knn(train = train.knn[,-c(1,4)],test = valid.knn[,-c(1,4)],cl = train.knn[,4],k = i)

knn\_model2 <- class::knn(train = train.knn[,-c(1,4)],test = train.knn[,-c(1,4)],cl = train.knn[,4],k = i)

cm1 <- caret::confusionMatrix(knn\_model1,valid.knn$Churn,positive = "Yes")

cm2 <- caret::confusionMatrix(knn\_model2,train.knn$Churn,positive = "Yes")

error[i,1] <- 1 - cm2$byClass[11]

error[i,2] <- 1 - cm1$byClass[11]

i = i + 1

}

#Oversampled

#K=23

knn\_model1 <- class::knn(train = train.knn[,-c(1,4)],test = valid.knn[,-c(1,4)],cl = train.knn[,4],k = 23)

#Sampled

#K=23

knn\_model2 <- class::knn(train = train.sam[,-c(1,4)],test = valid.sam[,-c(1,4)],cl = train.sam[,4],k = 23)

#-------------------------

#Model Performance

library(verification)

library(gmodels)

library(caret)

#Oversampled

#Confusion Matrix

gmodels::CrossTable(knn\_model1,valid.knn[,4],prop.r = FALSE,prop.c = FALSE,prop.t = FALSE,prop.chisq = FALSE)

caret::confusionMatrix(knn\_model1,valid.knn$Churn,positive = "Yes")

#ROC Curve

verification::roc.plot(ifelse(valid.knn$Churn == "Yes",1,0),ifelse(knn\_model1 == "Yes",1,0))

#Sampled

#Confusion Matrix

gmodels::CrossTable(knn\_model2,valid.sam[,4],prop.r = FALSE,prop.c = FALSE,prop.t = FALSE,prop.chisq = FALSE)

caret::confusionMatrix(knn\_model2,valid.sam$Churn,positive = "Yes")

#ROC Curve

verification::roc.plot(ifelse(valid.sam$Churn == "Yes",1,0),ifelse(knn\_model2 == "Yes",1,0))

#-------------------------

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#Naive Bayes

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#Importing Data

churn <- read.csv("new churn.csv")

str(churn)

summary(churn)

#-------------------------

#Preprocessing

#Deleting observations with missing values

churn <- churn[complete.cases(churn),]

#Function to change 'No Phone/Internet Service to No'

sub1 <- function(x){

gsub("No phone service","No",x)

}

sub2 <- function(x){

gsub("No internet service","No",x)

}

#Applying function sub to data frame

churn <- data.frame(lapply(churn, sub1))

churn <- data.frame(lapply(churn, sub2))

#Converting factor to numeric

churn$tenure <- as.numeric(as.character(churn$tenure))

churn$MonthlyCharges <- as.numeric(as.character(churn$MonthlyCharges))

churn$TotalCharges <- as.numeric(as.character(churn$TotalCharges))

#Function to convert months to years

conv <- function(x){

x/12

}

churn$tenure <- sapply(churn$tenure,conv)

#Binning tenure

churn$tenure[churn$tenure >= 0 & churn$tenure <=1] = '0-1'

churn$tenure[churn$tenure > 1 & churn$tenure <=2] = '1-2'

churn$tenure[churn$tenure > 2 & churn$tenure <=3] = '2-3'

churn$tenure[churn$tenure > 3 & churn$tenure <=4] = '3-4'

churn$tenure[churn$tenure > 4 & churn$tenure <=5] = '4-5'

churn$tenure[churn$tenure > 5 & churn$tenure <=6] = '5-6'

churn$tenure <- as.factor(churn$tenure)

#Binning of total charges and Monthly Charges

library(OneR)

churn$MonthlyCharges <- OneR::bin(churn$MonthlyCharges, nbins = 5, labels = c(1,2,3,4,5))

churn$TotalCharges <- OneR::bin(churn$TotalCharges, nbins = 10, labels = c(1,2,3,4,5,6,7,8,9,10))

summary(churn$MonthlyCharges)

summary(churn$TotalCharges)

#-------------------------

#Partioning Data

#Original ratio

set.seed(123)

or <- sum(churn$Churn == "Yes")/sum(churn$Churn == "No")

churn.yes.index <- churn$Churn == "Yes"

churn.no.index <- churn$Churn == "No"

churn.yes.df <- churn[churn.yes.index,]

churn.no.df <- churn[churn.no.index,]

#Training/Validation

#Yes

train.yes.index <- sample(c(1:dim(churn.yes.df)[1]),dim(churn.yes.df)[1]/2)

train.yes.df <- churn.yes.df[train.yes.index,]

valid.yes.df <- churn.yes.df[-train.yes.index,]

#No

train.no.index <- sample(c(1:dim(churn.no.df)[1]),dim(churn.yes.df)[1]/2)

train.no.df <- churn.no.df[train.no.index,]

valid.no.df <- churn.no.df[-train.no.index,]

valid.no.index <- sample(c(1:dim(valid.no.df)[1]),(dim(train.yes.df)[1]/or))

valid.no.df <- churn.no.df[valid.no.index,]

#Combining Train/Valid

train.df <- rbind(train.yes.df,train.no.df)

valid.df <- rbind(valid.yes.df,valid.no.df)

#Oversampling

train.NB <- train.df

valid.NB <- valid.df

#Sampling

churn.sam <- rbind(train.NB,valid.NB)

train.index <- sample(c(1:dim(churn.sam)[1]),0.60\*dim(churn.sam)[1])

train.sam <- churn.sam[train.index,]

valid.sam <- churn.sam[-train.index,]

#-------------------------

#Naive Bayes

library(e1071)

#Oversampled

train.NB <- train.NB[,-1]

valid.NB <- valid.NB[,-1]

NB\_model1 <- e1071::naiveBayes(Churn~.,data = train.NB,type="class")

valid.NB.pred1 <- predict(NB\_model1,newdata = valid.NB)

pred.prob1 <- predict(NB\_model1,newdata = valid.NB,type = "raw")

#Sampled

train.sam <- train.sam[,-1]

valid.sam <- valid.sam[,-1]

NB\_model2 <- e1071::naiveBayes(Churn~.,data = train.sam,type="class")

valid.NB.pred2 <- predict(NB\_model2,newdata = valid.sam)

pred.prob2 <- predict(NB\_model2,newdata = valid.sam,type = "raw")

#-------------------------

#Model Performance

library(gmodels)

library(caret)

library(gains)

library(verification)

#Oversampled

#Confusion Matrix

gmodels::CrossTable(valid.NB.pred1,valid.NB$Churn,prop.r = FALSE,prop.c = FALSE,prop.t = FALSE,prop.chisq = FALSE)

caret::confusionMatrix(valid.NB.pred1,valid.NB$Churn,positive = "Yes")

#Lift Chart

gain <- gains(ifelse(valid.NB$Churn=="Yes",1,0), pred.prob1[,2], groups=100)

plot(c(0,gain$cume.pct.of.total\*sum(valid.NB$Churn=="Yes"))~c(0,gain$cume.obs),

xlab="# cases", ylab="Cumulative", main="", type="l")

lines(c(0,sum(valid.NB$Churn=="Yes"))~c(0, dim(valid.NB)[1]), lty=2)

#Decile-wise Lift Chart

heights <- gain$mean.resp/mean(ifelse(valid.NB$Churn=="Yes",1,0))

midpoints <- barplot(heights, names.arg = gain$depth, ylim = c(0,9),

xlab = "Percentile", ylab = "Mean Response", main = "Decile-wise lift chart")

#ROC Curve

verification::roc.plot(ifelse(valid.NB$Churn=="Yes",1,0),ifelse(valid.NB.pred1 == "Yes",1,0))

#Sampled

#Confusion Matrix

gmodels::CrossTable(valid.NB.pred2,valid.sam$Churn,prop.r = FALSE,prop.c = FALSE,prop.t = FALSE,prop.chisq = FALSE)

caret::confusionMatrix(valid.NB.pred2,valid.sam$Churn,positive = "Yes")

#Lift Chart

gain <- gains(ifelse(valid.sam$Churn=="Yes",1,0), pred.prob2[,2], groups=100)

plot(c(0,gain$cume.pct.of.total\*sum(valid.sam$Churn=="Yes"))~c(0,gain$cume.obs),

xlab="# cases", ylab="Cumulative", main="", type="l")

lines(c(0,sum(valid.sam$Churn=="Yes"))~c(0, dim(valid.sam)[1]), lty=2)

#Decile-wise Lift Chart

heights <- gain$mean.resp/mean(ifelse(valid.sam$Churn=="Yes",1,0))

midpoints <- barplot(heights, names.arg = gain$depth, ylim = c(0,9),

xlab = "Percentile", ylab = "Mean Response", main = "Decile-wise lift chart")

#ROC Curve

verification::roc.plot(ifelse(valid.sam$Churn=="Yes",1,0),ifelse(valid.NB.pred2 == "Yes",1,0))

#-------------------------

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#CART/Random Forest/Boosted Trees

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#Importing Data

churn <- read.csv("new churn.csv")

str(churn)

summary(churn)

#-------------------------

#Preprocessing

#Deleting observations with missing values

churn <- churn[complete.cases(churn),]

#Function to change 'No Phone/Internet Service to No'

sub1 <- function(x){

gsub("No phone service","No",x)

}

sub2 <- function(x){

gsub("No internet service","No",x)

}

#Applying function sub to data frame

churn <- data.frame(lapply(churn, sub1))

churn <- data.frame(lapply(churn, sub2))

#Converting factor to numeric

churn$tenure <- as.numeric(as.character(churn$tenure))

churn$MonthlyCharges <- as.numeric(as.character(churn$MonthlyCharges))

churn$TotalCharges <- as.numeric(as.character(churn$TotalCharges))

#Function to convert months to years

conv <- function(x){

x/12

}

churn$tenure <- sapply(churn$tenure,conv)

#Binning tenure

churn$tenure[churn$tenure >= 0 & churn$tenure <=1] = '0-1'

churn$tenure[churn$tenure > 1 & churn$tenure <=2] = '1-2'

churn$tenure[churn$tenure > 2 & churn$tenure <=3] = '2-3'

churn$tenure[churn$tenure > 3 & churn$tenure <=4] = '3-4'

churn$tenure[churn$tenure > 4 & churn$tenure <=5] = '4-5'

churn$tenure[churn$tenure > 5 & churn$tenure <=6] = '5-6'

churn$tenure <- as.factor(churn$tenure)

#Standardizing columns Monthly Charges and Total Charges

churn[,c('MonthlyCharges','TotalCharges')] = scale(churn[,c('MonthlyCharges','TotalCharges')])

churn$Churn <- as.factor(churn$Churn)

#-------------------------

#Partioning Data

#Original ratio

set.seed(123)

or <- sum(churn$Churn == "Yes")/sum(churn$Churn == "No")

churn.yes.index <- churn$Churn == "Yes"

churn.no.index <- churn$Churn == "No"

churn.yes.df <- churn[churn.yes.index,]

churn.no.df <- churn[churn.no.index,]

#Training/Validation

#Yes

train.yes.index <- sample(c(1:dim(churn.yes.df)[1]),dim(churn.yes.df)[1]/2)

train.yes.df <- churn.yes.df[train.yes.index,]

valid.yes.df <- churn.yes.df[-train.yes.index,]

#No

train.no.index <- sample(c(1:dim(churn.no.df)[1]),dim(churn.yes.df)[1]/2)

train.no.df <- churn.no.df[train.no.index,]

valid.no.df <- churn.no.df[-train.no.index,]

valid.no.index <- sample(c(1:dim(valid.no.df)[1]),(dim(train.yes.df)[1]/or))

valid.no.df <- churn.no.df[valid.no.index,]

#Combining Train/Valid

train.df <- rbind(train.yes.df,train.no.df)

valid.df <- rbind(valid.yes.df,valid.no.df)

#Oversampling

train.CT <- train.df

valid.CT <- valid.df

#Sampling

churn.sam <- rbind(train.CT,valid.CT)

train.index <- sample(c(1:dim(churn.sam)[1]),0.60\*dim(churn.sam)[1])

train.sam <- churn.sam[train.index,]

valid.sam <- churn.sam[-train.index,]

#-------------------------

#Classification Tree

library(rpart)

library(rpart.plot)

#Oversampled

train.CT <- train.CT[,-1]

valid.CT <- valid.CT[,-1]

CT\_model1 <- rpart::rpart(Churn~.,data = train.CT,method = "class")

rpart.plot::prp(CT\_model1,type = 1, extra = 1, split.font = 1, varlen = -10,under = TRUE)

valid.CT.pred1 <- as.factor(predict(CT\_model1,valid.CT,type = "class"))

pred.prob1 <- predict(CT\_model1,valid.CT,type = "prob")

#Sampled

train.sam <- train.sam[,-1]

valid.sam <- valid.sam[,-1]

CT\_model2 <- rpart::rpart(Churn~.,data = train.sam,method = "class")

rpart.plot::prp(CT\_model2,type = 1, extra = 1, split.font = 1, varlen = -10,under = TRUE)

valid.CT.pred2 <- as.factor(predict(CT\_model2,valid.sam,type = "class"))

pred.prob2 <- predict(CT\_model2,valid.sam,type = "prob")

#-------------------------

#Model Performance

library(gmodels)

library(caret)

library(gains)

library(verification)

#Oversampled

#Confusion Matrix

gmodels::CrossTable(valid.CT.pred1,valid.CT$Churn,prop.r = FALSE,prop.c = FALSE,prop.t = FALSE,prop.chisq = FALSE)

caret::confusionMatrix(valid.CT.pred1,valid.CT$Churn,positive = "Yes")

#Lift Chart

gain <- gains(ifelse(valid.CT$Churn=="Yes",1,0), pred.prob1[,2], groups=100)

plot(c(0,gain$cume.pct.of.total\*sum(valid.CT$Churn=="Yes"))~c(0,gain$cume.obs),

xlab="# cases", ylab="Cumulative", main="", type="l")

lines(c(0,sum(valid.CT$Churn=="Yes"))~c(0, dim(valid.CT)[1]), lty=2)

#Decile-wise Lift Chart

heights <- gain$mean.resp/mean(ifelse(valid.CT$Churn=="Yes",1,0))

midpoints <- barplot(heights, names.arg = gain$depth, ylim = c(0,9),

xlab = "Percentile", ylab = "Mean Response", main = "Decile-wise lift chart")

#ROC Curve

verification::roc.plot(ifelse(valid.CT$Churn=="Yes",1,0),ifelse(valid.CT.pred1 == "Yes",1,0))

#Sampled

#Confusion Matrix

gmodels::CrossTable(valid.CT.pred2,valid.sam$Churn,prop.r = FALSE,prop.c = FALSE,prop.t = FALSE,prop.chisq = FALSE)

caret::confusionMatrix(valid.CT.pred2,valid.sam$Churn,positive = "Yes")

#Lift Chart

gain <- gains(ifelse(valid.sam$Churn=="Yes",1,0), pred.prob2[,2], groups=100)

plot(c(0,gain$cume.pct.of.total\*sum(valid.sam$Churn=="Yes"))~c(0,gain$cume.obs),

xlab="# cases", ylab="Cumulative", main="", type="l")

lines(c(0,sum(valid.sam$Churn=="Yes"))~c(0, dim(valid.sam)[1]), lty=2)

#Decile-wise Lift Chart

heights <- gain$mean.resp/mean(ifelse(valid.sam$Churn=="Yes",1,0))

midpoints <- barplot(heights, names.arg = gain$depth, ylim = c(0,9),

xlab = "Percentile", ylab = "Mean Response", main = "Decile-wise lift chart")

#ROC Curve

verification::roc.plot(ifelse(valid.sam$Churn=="Yes",1,0),ifelse(valid.CT.pred2 == "Yes",1,0))

#-------------------------

#Pruning Classification Tree

#Oversampled

CT\_pruned1 <- rpart::prune(CT\_model1, cp = CT\_model1$cptable[which.min(CT\_model1$cptable[,"xerror"]),"CP"])

rpart.plot::prp(CT\_pruned1,type = 1, extra = 1, split.font = 1, varlen = -10,under = TRUE)

valid.CT.pred1 <- as.factor(predict(CT\_pruned1,valid.CT,type = "class"))

pred.prob1 <- predict(CT\_pruned1,valid.CT,type = "prob")

#Sampled

CT\_pruned2 <- rpart::prune(CT\_model2, cp = CT\_model2$cptable[which.min(CT\_model2$cptable[,"xerror"]),"CP"])

rpart.plot::prp(CT\_pruned2,type = 1, extra = 1, split.font = 1, varlen = -10,under = TRUE)

valid.CT.pred2 <- as.factor(predict(CT\_pruned2,valid.sam,type = "class"))

pred.prob2 <- predict(CT\_pruned2,valid.sam,type = "prob")

#-------------------------

#Model Performance

library(gmodels)

library(caret)

library(gains)

library(verification)

#Oversampled

#Confusion Matrix

gmodels::CrossTable(valid.CT.pred1,valid.CT$Churn,prop.r = FALSE,prop.c = FALSE,prop.t = FALSE,prop.chisq = FALSE)

caret::confusionMatrix(valid.CT.pred1,valid.CT$Churn,positive = "Yes")

#Lift Chart

gain <- gains(ifelse(valid.CT$Churn=="Yes",1,0), pred.prob1[,2], groups=100)

plot(c(0,gain$cume.pct.of.total\*sum(valid.CT$Churn=="Yes"))~c(0,gain$cume.obs),

xlab="# cases", ylab="Cumulative", main="", type="l")

lines(c(0,sum(valid.CT$Churn=="Yes"))~c(0, dim(valid.CT)[1]), lty=2)

#Decile-wise Lift Chart

heights <- gain$mean.resp/mean(ifelse(valid.CT$Churn=="Yes",1,0))

midpoints <- barplot(heights, names.arg = gain$depth, ylim = c(0,9),

xlab = "Percentile", ylab = "Mean Response", main = "Decile-wise lift chart")

#ROC Curve

verification::roc.plot(ifelse(valid.CT$Churn=="Yes",1,0),ifelse(valid.CT.pred1 == "Yes",1,0))

#Sampled

#Confusion Matrix

gmodels::CrossTable(valid.CT.pred2,valid.sam$Churn,prop.r = FALSE,prop.c = FALSE,prop.t = FALSE,prop.chisq = FALSE)

caret::confusionMatrix(valid.CT.pred2,valid.sam$Churn,positive = "Yes")

#Lift Chart

gain <- gains(ifelse(valid.sam$Churn=="Yes",1,0), pred.prob2[,2], groups=100)

plot(c(0,gain$cume.pct.of.total\*sum(valid.sam$Churn=="Yes"))~c(0,gain$cume.obs),

xlab="# cases", ylab="Cumulative", main="", type="l")

lines(c(0,sum(valid.sam$Churn=="Yes"))~c(0, dim(valid.sam)[1]), lty=2)

#Decile-wise Lift Chart

heights <- gain$mean.resp/mean(ifelse(valid.sam$Churn=="Yes",1,0))

midpoints <- barplot(heights, names.arg = gain$depth, ylim = c(0,9),

xlab = "Percentile", ylab = "Mean Response", main = "Decile-wise lift chart")

#ROC Curve

verification::roc.plot(ifelse(valid.sam$Churn=="Yes",1,0),ifelse(valid.CT.pred2 == "Yes",1,0))

#-------------------------

#Random Forest

library(randomForest)

#Oversampled

RF\_model1 <- randomForest::randomForest(Churn ~ ., data = train.CT, ntree = 500, mtry = 4, nodesize = 5, importance = TRUE)

#Variable Importance Plot

varImpPlot(RF\_model1, type = 1)

valid.CT.pred1 <- as.factor(predict(RF\_model1,valid.CT,type = "class"))

pred.prob1 <- predict(RF\_model1,valid.CT,type = "prob")

#Sampled

RF\_model2 <- randomForest::randomForest(Churn ~ ., data = train.sam, ntree = 500, mtry = 4, nodesize = 5, importance = TRUE)

#Variable Importance Plot

varImpPlot(RF\_model2, type = 1)

valid.CT.pred2 <- as.factor(predict(RF\_model2,valid.sam,type = "class"))

pred.prob2 <- predict(RF\_model2,valid.sam,type = "prob")

#-------------------------

#Model Performance

library(gmodels)

library(caret)

library(gains)

library(verification)

#Oversampled

#Confusion Matrix

gmodels::CrossTable(valid.CT.pred1,valid.CT$Churn,prop.r = FALSE,prop.c = FALSE,prop.t = FALSE,prop.chisq = FALSE)

caret::confusionMatrix(valid.CT.pred1,valid.CT$Churn,positive = "Yes")

#Lift Chart

gain <- gains(ifelse(valid.CT$Churn=="Yes",1,0), pred.prob1[,2], groups=100)

plot(c(0,gain$cume.pct.of.total\*sum(valid.CT$Churn=="Yes"))~c(0,gain$cume.obs),

xlab="# cases", ylab="Cumulative", main="", type="l")

lines(c(0,sum(valid.CT$Churn=="Yes"))~c(0, dim(valid.CT)[1]), lty=2)

#Decile-wise Lift Chart

heights <- gain$mean.resp/mean(ifelse(valid.CT$Churn=="Yes",1,0))

midpoints <- barplot(heights, names.arg = gain$depth, ylim = c(0,9),

xlab = "Percentile", ylab = "Mean Response", main = "Decile-wise lift chart")

#ROC Curve

verification::roc.plot(ifelse(valid.CT$Churn=="Yes",1,0),ifelse(valid.CT.pred1 == "Yes",1,0))

#Sampled

#Confusion Matrix

gmodels::CrossTable(valid.CT.pred2,valid.sam$Churn,prop.r = FALSE,prop.c = FALSE,prop.t = FALSE,prop.chisq = FALSE)

caret::confusionMatrix(valid.CT.pred2,valid.sam$Churn,positive = "Yes")

#Lift Chart

gain <- gains(ifelse(valid.sam$Churn=="Yes",1,0), pred.prob2[,2], groups=100)

plot(c(0,gain$cume.pct.of.total\*sum(valid.sam$Churn=="Yes"))~c(0,gain$cume.obs),

xlab="# cases", ylab="Cumulative", main="", type="l")

lines(c(0,sum(valid.sam$Churn=="Yes"))~c(0, dim(valid.sam)[1]), lty=2)

#Decile-wise Lift Chart

heights <- gain$mean.resp/mean(ifelse(valid.sam$Churn=="Yes",1,0))

midpoints <- barplot(heights, names.arg = gain$depth, ylim = c(0,9),

xlab = "Percentile", ylab = "Mean Response", main = "Decile-wise lift chart")

#ROC Curve

verification::roc.plot(ifelse(valid.sam$Churn=="Yes",1,0),ifelse(valid.CT.pred2 == "Yes",1,0))

#-------------------------

#Boosted Trees

library(adabag)

#Oversampled

boost\_model1 <- adabag::boosting(Churn~.,data = train.CT)

valid.CT.pred1 <- as.factor(predict(boost\_model1,valid.CT)$class)

pred.prob1 <- predict(boost\_model1,valid.CT)$prob

#Sampled

boost\_model2 <- adabag::boosting(Churn~.,data = train.sam)

valid.CT.pred2 <- as.factor(predict(boost\_model2,valid.sam)$class)

pred.prob2 <- predict(boost\_model2,valid.sam)$prob

#-------------------------

#Model Performance

library(gmodels)

library(caret)

library(gains)

library(verification)

#Oversampled

#Confusion Matrix

gmodels::CrossTable(valid.CT.pred1,valid.CT$Churn,prop.r = FALSE,prop.c = FALSE,prop.t = FALSE,prop.chisq = FALSE)

caret::confusionMatrix(valid.CT.pred1,valid.CT$Churn,positive = "Yes")

#Lift Chart

gain <- gains(ifelse(valid.CT$Churn=="Yes",1,0), pred.prob1[,2], groups=100)

plot(c(0,gain$cume.pct.of.total\*sum(valid.CT$Churn=="Yes"))~c(0,gain$cume.obs),

xlab="# cases", ylab="Cumulative", main="", type="l")

lines(c(0,sum(valid.CT$Churn=="Yes"))~c(0, dim(valid.CT)[1]), lty=2)

#Decile-wise Lift Chart

heights <- gain$mean.resp/mean(ifelse(valid.CT$Churn=="Yes",1,0))

midpoints <- barplot(heights, names.arg = gain$depth, ylim = c(0,9),

xlab = "Percentile", ylab = "Mean Response", main = "Decile-wise lift chart")

#ROC Curve

verification::roc.plot(ifelse(valid.CT$Churn=="Yes",1,0),ifelse(valid.CT.pred1 == "Yes",1,0))

#Sampled

#Confusion Matrix

gmodels::CrossTable(valid.CT.pred2,valid.sam$Churn,prop.r = FALSE,prop.c = FALSE,prop.t = FALSE,prop.chisq = FALSE)

caret::confusionMatrix(valid.CT.pred2,valid.sam$Churn,positive = "Yes")

#Lift Chart

gain <- gains(ifelse(valid.sam$Churn=="Yes",1,0), pred.prob2[,2], groups=100)

plot(c(0,gain$cume.pct.of.total\*sum(valid.sam$Churn=="Yes"))~c(0,gain$cume.obs),

xlab="# cases", ylab="Cumulative", main="", type="l")

lines(c(0,sum(valid.sam$Churn=="Yes"))~c(0, dim(valid.sam)[1]), lty=2)

#Decile-wise Lift Chart

heights <- gain$mean.resp/mean(ifelse(valid.sam$Churn=="Yes",1,0))

midpoints <- barplot(heights, names.arg = gain$depth, ylim = c(0,9),

xlab = "Percentile", ylab = "Mean Response", main = "Decile-wise lift chart")

#ROC Curve

verification::roc.plot(ifelse(valid.sam$Churn=="Yes",1,0),ifelse(valid.CT.pred2 == "Yes",1,0))

#-------------------------

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#Logistic Regression

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#Importing Data

churn <- read.csv("new churn.csv")

str(churn)

summary(churn)

#-------------------------

#Preprocessing

#Deleting observations with missing values

churn <- churn[complete.cases(churn),]

#Function to change 'No Phone/Internet Service to No'

sub1 <- function(x){

gsub("No phone service","No",x)

}

sub2 <- function(x){

gsub("No internet service","No",x)

}

#Applying function sub to data frame

churn <- data.frame(lapply(churn, sub1))

churn <- data.frame(lapply(churn, sub2))

#Converting factor to numeric

churn$tenure <- as.numeric(as.character(churn$tenure))

churn$MonthlyCharges <- as.numeric(as.character(churn$MonthlyCharges))

churn$TotalCharges <- as.numeric(as.character(churn$TotalCharges))

#Function to convert months to years

conv <- function(x){

x/12

}

churn$tenure <- sapply(churn$tenure,conv)

#Binning tenure

churn$tenure[churn$tenure >= 0 & churn$tenure <=1] = '0-1'

churn$tenure[churn$tenure > 1 & churn$tenure <=2] = '1-2'

churn$tenure[churn$tenure > 2 & churn$tenure <=3] = '2-3'

churn$tenure[churn$tenure > 3 & churn$tenure <=4] = '3-4'

churn$tenure[churn$tenure > 4 & churn$tenure <=5] = '4-5'

churn$tenure[churn$tenure > 5 & churn$tenure <=6] = '5-6'

churn$tenure <- as.factor(churn$tenure)

#Standardizing columns Monthly Charges and Total Charges

churn[,c('MonthlyCharges','TotalCharges')] = scale(churn[,c('MonthlyCharges','TotalCharges')])

#-------------------------

#Partioning Data

#Original ratio

set.seed(123)

or <- sum(churn$Churn == "Yes")/sum(churn$Churn == "No")

churn.yes.index <- churn$Churn == "Yes"

churn.no.index <- churn$Churn == "No"

churn.yes.df <- churn[churn.yes.index,]

churn.no.df <- churn[churn.no.index,]

#Training/Validation

#Yes

train.yes.index <- sample(c(1:dim(churn.yes.df)[1]),dim(churn.yes.df)[1]/2)

train.yes.df <- churn.yes.df[train.yes.index,]

valid.yes.df <- churn.yes.df[-train.yes.index,]

#No

train.no.index <- sample(c(1:dim(churn.no.df)[1]),dim(churn.yes.df)[1]/2)

train.no.df <- churn.no.df[train.no.index,]

valid.no.df <- churn.no.df[-train.no.index,]

valid.no.index <- sample(c(1:dim(valid.no.df)[1]),(dim(train.yes.df)[1]/or))

valid.no.df <- churn.no.df[valid.no.index,]

#Combining Train/Valid

train.df <- rbind(train.yes.df,train.no.df)

valid.df <- rbind(valid.yes.df,valid.no.df)

#-------------------------

#Dummy Variable for other algorithms

#m-1 dummies

#Categorical Columns & Numerical Columns

cat <- churn[,-c(1,19,20,21)]

num <- churn[,c(1,19,20,21)]

#Function to create dummy variable

dum <- function(x){

model.matrix(~x-1,data = churn)[,-1]

}

#Creating Dummy Variables

dummy <- data.frame(sapply(cat, dum))

#Combining variables to final dataset

churn.logit <- cbind(num,dummy)

str(churn.logit)

#-------------------------

#Oversampling

train.logit <- churn.logit[rownames(train.df),]

valid.logit <- churn.logit[rownames(valid.df),]

#Sampling

churn.sam <- rbind(train.logit,valid.logit)

train.index <- sample(c(1:dim(churn.sam)[1]),0.60\*dim(churn.sam)[1])

train.sam <- churn.sam[train.index,]

valid.sam <- churn.sam[-train.index,]

#-------------------------

#Logistic Regression

#Oversampled

train.logit <- train.logit[,-1]

valid.logit <- valid.logit[,-1]

logit\_model1 <- glm(Churn~.,data = train.logit,family = "binomial")

valid.logit.pred1 <- as.factor(ifelse(predict(logit\_model1,valid.logit,type = "response") > 0.50,"Yes","No"))

pred.prob1 <- predict(logit\_model1,valid.logit,type = "response")

#Sampled

train.sam <- train.sam[,-1]

valid.sam <- valid.sam[,-1]

logit\_model2 <- glm(Churn~.,data = train.sam,family = "binomial")

valid.logit.pred2 <- as.factor(ifelse(predict(logit\_model2,valid.sam,type = "response") > 0.50,"Yes","No"))

pred.prob2 <- predict(logit\_model1,valid.sam,type = "response")

#-------------------------

#Model Performance

library(gmodels)

library(caret)

library(gains)

library(verification)

#Oversampled

#Confusion Matrix

gmodels::CrossTable(valid.logit.pred1,valid.logit$Churn,prop.r = FALSE,prop.c = FALSE,prop.t = FALSE,prop.chisq = FALSE)

caret::confusionMatrix(valid.logit.pred1,valid.logit$Churn,positive = "Yes")

#Lift Chart

gain <- gains(ifelse(valid.logit$Churn=="Yes",1,0), pred.prob1, groups=100)

plot(c(0,gain$cume.pct.of.total\*sum(valid.logit$Churn=="Yes"))~c(0,gain$cume.obs),

xlab="# cases", ylab="Cumulative", main="", type="l")

lines(c(0,sum(valid.logit$Churn=="Yes"))~c(0, dim(valid.logit)[1]), lty=2)

#Decile-wise Lift Chart

heights <- gain$mean.resp/mean(ifelse(valid.logit$Churn=="Yes",1,0))

midpoints <- barplot(heights, names.arg = gain$depth, ylim = c(0,9),

xlab = "Percentile", ylab = "Mean Response", main = "Decile-wise lift chart")

#ROC Curve

verification::roc.plot(ifelse(valid.logit$Churn=="Yes",1,0),ifelse(valid.logit.pred1 == "Yes",1,0))

#Sampled

#Confusion Matrix

gmodels::CrossTable(valid.logit.pred2,valid.sam$Churn,prop.r = FALSE,prop.c = FALSE,prop.t = FALSE,prop.chisq = FALSE)

caret::confusionMatrix(valid.logit.pred2,valid.sam$Churn,positive = "Yes")

#Lift Chart

gain <- gains(ifelse(valid.sam$Churn=="Yes",1,0), pred.prob2, groups=100)

plot(c(0,gain$cume.pct.of.total\*sum(valid.sam$Churn=="Yes"))~c(0,gain$cume.obs),

xlab="# cases", ylab="Cumulative", main="", type="l")

lines(c(0,sum(valid.sam$Churn=="Yes"))~c(0, dim(valid.sam)[1]), lty=2)

#Decile-wise Lift Chart

heights <- gain$mean.resp/mean(ifelse(valid.sam$Churn=="Yes",1,0))

midpoints <- barplot(heights, names.arg = gain$depth, ylim = c(0,9),

xlab = "Percentile", ylab = "Mean Response", main = "Decile-wise lift chart")

#ROC Curve

verification::roc.plot(ifelse(valid.sam$Churn=="Yes",1,0),ifelse(valid.logit.pred2 == "Yes",1,0))

#-------------------------

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#Neural Nets

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#Importing Data

churn <- read.csv("new churn.csv")

str(churn)

summary(churn)

#-------------------------

#Preprocessing

library(scales)

#Deleting observations with missing values

churn <- churn[complete.cases(churn),]

#Function to change 'No Phone/Internet Service to No'

sub1 <- function(x){

gsub("No phone service","No",x)

}

sub2 <- function(x){

gsub("No internet service","No",x)

}

#Applying function sub to data frame

churn <- data.frame(lapply(churn, sub1))

churn <- data.frame(lapply(churn, sub2))

#Converting factor to numeric

churn$tenure <- as.numeric(as.character(churn$tenure))

churn$MonthlyCharges <- as.numeric(as.character(churn$MonthlyCharges))

churn$TotalCharges <- as.numeric(as.character(churn$TotalCharges))

#Function to convert months to years

conv <- function(x){

x/12

}

churn$tenure <- sapply(churn$tenure,conv)

#Binning tenure

churn$tenure[churn$tenure >= 0 & churn$tenure <=1] = '0-1'

churn$tenure[churn$tenure > 1 & churn$tenure <=2] = '1-2'

churn$tenure[churn$tenure > 2 & churn$tenure <=3] = '2-3'

churn$tenure[churn$tenure > 3 & churn$tenure <=4] = '3-4'

churn$tenure[churn$tenure > 4 & churn$tenure <=5] = '4-5'

churn$tenure[churn$tenure > 5 & churn$tenure <=6] = '5-6'

churn$tenure <- as.factor(churn$tenure)

#Normalizing/Scaling columns Monthly Charges and Total Charges

#Neural Nets Normalizaing/Scaling

churn$MonthlyCharges = scales::rescale(churn$MonthlyCharges)

churn$TotalCharges = scales::rescale(churn$TotalCharges)

#Transforming MonthlyCharges by squareroot & TotalCharges by Cuberoot

#Function for cuberoot

cbrt <- function(x){

sign(x) \* abs(x)^(1/3)

}

ggplot(churn)+geom\_histogram(mapping = aes(MonthlyCharges),bins = 50)

ggplot(churn)+geom\_histogram(mapping = aes(TotalCharges),bins = 50)

churn$MonthlyCharges = sqrt(churn$MonthlyCharges)

churn$TotalCharges = cbrt(churn$TotalCharges)

ggplot(churn)+geom\_histogram(mapping = aes(MonthlyCharges),bins = 50)

ggplot(churn)+geom\_histogram(mapping = aes(TotalCharges),bins = 50)

#-------------------------

#Partioning Data

#Original ratio

set.seed(123)

or <- sum(churn$Churn == "Yes")/sum(churn$Churn == "No")

churn.yes.index <- churn$Churn == "Yes"

churn.no.index <- churn$Churn == "No"

churn.yes.df <- churn[churn.yes.index,]

churn.no.df <- churn[churn.no.index,]

#Training/Validation

#Yes

train.yes.index <- sample(c(1:dim(churn.yes.df)[1]),dim(churn.yes.df)[1]/2)

train.yes.df <- churn.yes.df[train.yes.index,]

valid.yes.df <- churn.yes.df[-train.yes.index,]

#No

train.no.index <- sample(c(1:dim(churn.no.df)[1]),dim(churn.yes.df)[1]/2)

train.no.df <- churn.no.df[train.no.index,]

valid.no.df <- churn.no.df[-train.no.index,]

valid.no.index <- sample(c(1:dim(valid.no.df)[1]),(dim(train.yes.df)[1]/or))

valid.no.df <- churn.no.df[valid.no.index,]

#Combining Train/Valid

train.df <- rbind(train.yes.df,train.no.df)

valid.df <- rbind(valid.yes.df,valid.no.df)

#-------------------------

#Dummy Variable for other algorithms

#m-1 dummies

#Categorical Columns & Numerical Columns

cat <- churn[,-c(1,19,20,21)]

num <- churn[,c(1,19,20,21)]

#Function to create dummy variable

dum <- function(x){

model.matrix(~x-1,data = churn)[,-1]

}

#Creating Dummy Variables

dummy <- data.frame(sapply(cat, dum))

#Combining variables to final dataset

churn.NN <- cbind(num,dummy)

str(churn.NN)

#-------------------------

#Oversampling

train.NN <- churn.NN[rownames(train.df),]

valid.NN <- churn.NN[rownames(valid.df),]

#Sampling

churn.sam <- rbind(train.NN,valid.NN)

train.index <- sample(c(1:dim(churn.sam)[1]),0.60\*dim(churn.sam)[1])

train.sam <- churn.sam[train.index,]

valid.sam <- churn.sam[-train.index,]

#-------------------------

#Neural Nets

library(neuralnet)

#Oversampling

train.NN <- train.NN[,-1]

valid.NN <- valid.NN[,-1]

NN\_model1 <- neuralnet::neuralnet(Churn~.,data = train.NN,linear.output = FALSE,hidden = 2)

plot(NN\_model1,rep = "best")

valid.NN.pred1 <- as.factor(ifelse(apply(neuralnet::compute(NN\_model1,valid.NN)$net.result,1,which.max)-1 == 1,"Yes","No"))

pred.prob1 <- predict(NN\_model1,valid.NN,type = "response")

#Sampling

train.sam <- train.sam[,-1]

valid.sam <- valid.sam[,-1]

NN\_model2 <- neuralnet::neuralnet(Churn~.,data = train.sam,linear.output = FALSE,hidden = 2)

plot(NN\_model2,rep = "best")

valid.NN.pred2 <- as.factor(ifelse(apply(neuralnet::compute(NN\_model2,valid.sam)$net.result,1,which.max)-1 == 1,"Yes","No"))

pred.prob2 <- predict(NN\_model2,valid.sam,type = "response")

#-------------------------

#Model Performance

library(gmodels)

library(caret)

library(gains)

library(verification)

#Oversampling

#Confusion Matrix

gmodels::CrossTable(valid.NN.pred1,valid.NN$Churn,prop.r = FALSE,prop.c = FALSE,prop.t = FALSE,prop.chisq = FALSE)

caret::confusionMatrix(valid.NN.pred1,valid.NN$Churn,positive = "Yes")

#Lift Chart

gain <- gains(ifelse(valid.NN$Churn=="Yes",1,0), pred.prob1[,2], groups=100)

plot(c(0,gain$cume.pct.of.total\*sum(valid.NN$Churn=="Yes"))~c(0,gain$cume.obs),

xlab="# cases", ylab="Cumulative", main="", type="l")

lines(c(0,sum(valid.NN$Churn=="Yes"))~c(0, dim(valid.NN)[1]), lty=2)

#Decile-wise Lift Chart

heights <- gain$mean.resp/mean(ifelse(valid.NN$Churn=="Yes",1,0))

midpoints <- barplot(heights, names.arg = gain$depth, ylim = c(0,9),

xlab = "Percentile", ylab = "Mean Response", main = "Decile-wise lift chart")

#ROC Curve

verification::roc.plot(ifelse(valid.NN$Churn=="Yes",1,0),ifelse(valid.NN.pred1=="Yes",1,0))

#Sampling

#Confusion Matrix

gmodels::CrossTable(valid.NN.pred2,valid.sam$Churn,prop.r = FALSE,prop.c = FALSE,prop.t = FALSE,prop.chisq = FALSE)

caret::confusionMatrix(valid.NN.pred2,valid.sam$Churn,positive = "Yes")

#Lift Chart

gain <- gains(ifelse(valid.sam$Churn=="Yes",1,0), pred.prob2[,2], groups=100)

plot(c(0,gain$cume.pct.of.total\*sum(valid.sam$Churn=="Yes"))~c(0,gain$cume.obs),

xlab="# cases", ylab="Cumulative", main="", type="l")

lines(c(0,sum(valid.sam$Churn=="Yes"))~c(0, dim(valid.sam)[1]), lty=2)

#Decile-wise Lift Chart

heights <- gain$mean.resp/mean(ifelse(valid.sam$Churn=="Yes",1,0))

midpoints <- barplot(heights, names.arg = gain$depth, ylim = c(0,9),

xlab = "Percentile", ylab = "Mean Response", main = "Decile-wise lift chart")

#ROC Curve

verification::roc.plot(ifelse(valid.sam$Churn=="Yes",1,0),ifelse(valid.NN.pred2=="Yes",1,0))

#-------------------------

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#Linear Discriminant Analysis

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#Importing Data

churn <- read.csv("new churn.csv")

str(churn)

summary(churn)

#-------------------------

#Preprocessing

#Deleting observations with missing values

churn <- churn[complete.cases(churn),]

#Function to change 'No Phone/Internet Service to No'

sub1 <- function(x){

gsub("No phone service","No",x)

}

sub2 <- function(x){

gsub("No internet service","No",x)

}

#Applying function sub to data frame

churn <- data.frame(lapply(churn, sub1))

churn <- data.frame(lapply(churn, sub2))

#Converting factor to numeric

churn$tenure <- as.numeric(as.character(churn$tenure))

churn$MonthlyCharges <- as.numeric(as.character(churn$MonthlyCharges))

churn$TotalCharges <- as.numeric(as.character(churn$TotalCharges))

#Function to convert months to years

conv <- function(x){

x/12

}

churn$tenure <- sapply(churn$tenure,conv)

#Standardizing columns Monthly Charges and Total Charges

churn[,c('tenure','MonthlyCharges','TotalCharges')] = scale(churn[,c('tenure','MonthlyCharges','TotalCharges')])

#Categorical Columns & Numerical Columns

cat <- churn[,-c(1,6,19,20,21)]

num <- churn[,c(1,6,19,20,21)]

#Converting Categorical to Numerical

cat <- data.frame(lapply(cat,as.numeric))

#Combining variables to final dataset

churn.LDA <- cbind(num,cat)

str(churn.LDA)

#-------------------------

#Partioning Data

#Original ratio

set.seed(123)

or <- sum(churn$Churn == "Yes")/sum(churn$Churn == "No")

churn.yes.index <- churn$Churn == "Yes"

churn.no.index <- churn$Churn == "No"

churn.yes.df <- churn[churn.yes.index,]

churn.no.df <- churn[churn.no.index,]

#Training/Validation

#Yes

train.yes.index <- sample(c(1:dim(churn.yes.df)[1]),dim(churn.yes.df)[1]/2)

train.yes.df <- churn.yes.df[train.yes.index,]

valid.yes.df <- churn.yes.df[-train.yes.index,]

#No

train.no.index <- sample(c(1:dim(churn.no.df)[1]),dim(churn.yes.df)[1]/2)

train.no.df <- churn.no.df[train.no.index,]

valid.no.df <- churn.no.df[-train.no.index,]

valid.no.index <- sample(c(1:dim(valid.no.df)[1]),(dim(train.yes.df)[1]/or))

valid.no.df <- churn.no.df[valid.no.index,]

#Combining Train/Valid

train.df <- rbind(train.yes.df,train.no.df)

valid.df <- rbind(valid.yes.df,valid.no.df)

#-------------------------

#Oversampling

train.LDA <- churn.LDA[rownames(train.df),]

valid.LDA <- churn.LDA[rownames(valid.df),]

#Sampling

churn.sam <- rbind(train.LDA,valid.LDA)

train.index <- sample(c(1:dim(churn.sam)[1]),0.60\*dim(churn.sam)[1])

train.sam <- churn.sam[train.index,]

valid.sam <- churn.sam[-train.index,]

#-------------------------

#Linear Discriminant Analysis

library(MASS)

#Oversampling

train.LDA <- train.LDA[,-1]

valid.LDA <- valid.LDA[,-1]

LDA\_model1 <- MASS::lda(Churn~.,data = train.LDA)

valid.LDA.pred1 <- as.factor((predict(LDA\_model1,valid.LDA)$class))

pred.prob1 <- predict(LDA\_model1,valid.LDA)$posterior

#Sampling

train.sam <- train.sam[,-1]

valid.sam <- valid.sam[,-1]

LDA\_model2 <- MASS::lda(Churn~.,data = train.sam)

valid.LDA.pred2 <- as.factor((predict(LDA\_model2,valid.sam)$class))

pred.prob2 <- predict(LDA\_model2,valid.sam)$posterior

#-------------------------

#Model Performance

library(gmodels)

library(caret)

library(gains)

library(verification)

#Oversampling

#Confusion Matrix

gmodels::CrossTable(valid.LDA.pred1,valid.LDA$Churn,prop.r = FALSE,prop.c = FALSE,prop.t = FALSE,prop.chisq = FALSE)

caret::confusionMatrix(valid.LDA.pred1,valid.LDA$Churn,positive = "Yes")

#Lift Chart

gain <- gains(ifelse(valid.LDA$Churn=="Yes",1,0), pred.prob1[,2], groups=100)

plot(c(0,gain$cume.pct.of.total\*sum(valid.LDA$Churn=="Yes"))~c(0,gain$cume.obs),

xlab="# cases", ylab="Cumulative", main="", type="l")

lines(c(0,sum(valid.LDA$Churn=="Yes"))~c(0, dim(valid.LDA)[1]), lty=2)

#Decile-wise Lift Chart

heights <- gain$mean.resp/mean(ifelse(valid.LDA$Churn=="Yes",1,0))

midpoints <- barplot(heights, names.arg = gain$depth, ylim = c(0,9),

xlab = "Percentile", ylab = "Mean Response", main = "Decile-wise lift chart")

#ROC Curve

verification::roc.plot(ifelse(valid.LDA$Churn=="Yes",1,0),ifelse(valid.LDA.pred1=="Yes",1,0))

#Sampling

#Confusion Matrix

gmodels::CrossTable(valid.LDA.pred2,valid.sam$Churn,prop.r = FALSE,prop.c = FALSE,prop.t = FALSE,prop.chisq = FALSE)

caret::confusionMatrix(valid.LDA.pred2,valid.sam$Churn,positive = "Yes")

#Lift Chart

gain <- gains(ifelse(valid.sam$Churn=="Yes",1,0), pred.prob2[,2], groups=100)

plot(c(0,gain$cume.pct.of.total\*sum(valid.sam$Churn=="Yes"))~c(0,gain$cume.obs),

xlab="# cases", ylab="Cumulative", main="", type="l")

lines(c(0,sum(valid.sam$Churn=="Yes"))~c(0, dim(valid.sam)[1]), lty=2)

#Decile-wise Lift Chart

heights <- gain$mean.resp/mean(ifelse(valid.sam$Churn=="Yes",1,0))

midpoints <- barplot(heights, names.arg = gain$depth, ylim = c(0,9),

xlab = "Percentile", ylab = "Mean Response", main = "Decile-wise lift chart")

#ROC Curve

verification::roc.plot(ifelse(valid.sam$Churn=="Yes",1,0),ifelse(valid.LDA.pred2=="Yes",1,0))

#-------------------------

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#Support Vector Machine

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#Importing Data

churn <- read.csv("new churn.csv")

str(churn)

summary(churn)

#-------------------------

#Preprocessing

#Deleting observations with missing values

churn <- churn[complete.cases(churn),]

#Function to change 'No Phone/Internet Service to No'

sub1 <- function(x){

gsub("No phone service","No",x)

}

sub2 <- function(x){

gsub("No internet service","No",x)

}

#Applying function sub to data frame

churn <- data.frame(lapply(churn, sub1))

churn <- data.frame(lapply(churn, sub2))

#Converting factor to numeric

churn$tenure <- as.numeric(as.character(churn$tenure))

churn$MonthlyCharges <- as.numeric(as.character(churn$MonthlyCharges))

churn$TotalCharges <- as.numeric(as.character(churn$TotalCharges))

#Function to convert months to years

conv <- function(x){

x/12

}

churn$tenure <- sapply(churn$tenure,conv)

#Binning tenure

churn$tenure[churn$tenure >= 0 & churn$tenure <=1] = '0-1'

churn$tenure[churn$tenure > 1 & churn$tenure <=2] = '1-2'

churn$tenure[churn$tenure > 2 & churn$tenure <=3] = '2-3'

churn$tenure[churn$tenure > 3 & churn$tenure <=4] = '3-4'

churn$tenure[churn$tenure > 4 & churn$tenure <=5] = '4-5'

churn$tenure[churn$tenure > 5 & churn$tenure <=6] = '5-6'

churn$tenure <- as.factor(churn$tenure)

#Standardizing columns Monthly Charges and Total Charges

churn[,c('MonthlyCharges','TotalCharges')] = scale(churn[,c('MonthlyCharges','TotalCharges')])

#-------------------------

#Partioning Data

#Original ratio

set.seed(123)

or <- sum(churn$Churn == "Yes")/sum(churn$Churn == "No")

churn.yes.index <- churn$Churn == "Yes"

churn.no.index <- churn$Churn == "No"

churn.yes.df <- churn[churn.yes.index,]

churn.no.df <- churn[churn.no.index,]

#Training/Validation

#Yes

train.yes.index <- sample(c(1:dim(churn.yes.df)[1]),dim(churn.yes.df)[1]/2)

train.yes.df <- churn.yes.df[train.yes.index,]

valid.yes.df <- churn.yes.df[-train.yes.index,]

#No

train.no.index <- sample(c(1:dim(churn.no.df)[1]),dim(churn.yes.df)[1]/2)

train.no.df <- churn.no.df[train.no.index,]

valid.no.df <- churn.no.df[-train.no.index,]

valid.no.index <- sample(c(1:dim(valid.no.df)[1]),(dim(train.yes.df)[1]/or))

valid.no.df <- churn.no.df[valid.no.index,]

#Combining Train/Valid

train.df <- rbind(train.yes.df,train.no.df)

valid.df <- rbind(valid.yes.df,valid.no.df)

#-------------------------

#Dummy Variable for other algorithms

#m-1 dummies

#Categorical Columns & Numerical Columns

cat <- churn[,-c(1,19,20,21)]

num <- churn[,c(1,19,20,21)]

#Function to create dummy variable

dum <- function(x){

model.matrix(~x-1,data = churn)[,-1]

}

#Creating Dummy Variables

dummy <- data.frame(sapply(cat, dum))

#Combining variables to final dataset

churn.SVM <- cbind(num,dummy)

str(churn.SVM)

#-------------------------

#Oversampling

train.SVM <- churn.SVM[rownames(train.df),]

valid.SVM <- churn.SVM[rownames(valid.df),]

#Sampling

churn.sam <- rbind(train.SVM,valid.SVM)

train.index <- sample(c(1:dim(churn.sam)[1]),0.60\*dim(churn.sam)[1])

train.sam <- churn.sam[train.index,]

valid.sam <- churn.sam[-train.index,]

#-------------------------

#Support Vector Machine

library(e1071)

#Oversampling

train.SVM <- train.SVM[,-1]

valid.SVM <- valid.SVM[,-1]

SVM\_model1 <- e1071::svm(Churn~.,data = train.SVM)

valid.SVM.pred1 <- as.factor(predict(SVM\_model1,valid.SVM))

#Sampling

train.sam <- train.sam[,-1]

valid.sam <- valid.sam[,-1]

SVM\_model2 <- e1071::svm(Churn~.,data = train.sam)

valid.SVM.pred2 <- as.factor(predict(SVM\_model2,valid.sam))

#-------------------------

#Model Performance

library(gmodels)

library(caret)

library(verification)

#Oversampling

#Confusion Matrix

gmodels::CrossTable(valid.SVM.pred1,valid.SVM$Churn,prop.r = FALSE,prop.c = FALSE,prop.t = FALSE,prop.chisq = FALSE)

caret::confusionMatrix(valid.SVM.pred1,valid.SVM$Churn,positive = "Yes")

#ROC Curve

verification::roc.plot(ifelse(valid.SVM$Churn=="Yes",1,0),ifelse(valid.SVM.pred1=="Yes",1,0))

#Sampling

#Confusion Matrix

gmodels::CrossTable(valid.SVM.pred2,valid.sam$Churn,prop.r = FALSE,prop.c = FALSE,prop.t = FALSE,prop.chisq = FALSE)

caret::confusionMatrix(valid.SVM.pred2,valid.sam$Churn,positive = "Yes")

#ROC Curve

verification::roc.plot(ifelse(valid.sam$Churn=="Yes",1,0),ifelse(valid.SVM.pred2=="Yes",1,0))

#-------------------------