Data cleaning process

Load the data set and replace all empty strings with NA

```
data = read.csv("TrainingData.csv", header = T, na.strings=c(""))
```

Remove columns which have all values as NA

```
trdata <- data
trdata = trdata[, !sapply(trdata, function(x)all(is.na(x))), drop=F]
```

Remove columns which have only one unique value other than NA

```
trdata = trdata[, !sapply(trdata, function(x)all( length(unique(x)) == 2 )),
drop=F]
```

This code removes TargetVariable also, so add that column later

There are four columns which have only one unique value,NA and "-", so remove them

```
cols.dont.want <- c("Variable142OPEN", "Variable142HIGH", "Variable142LOW",
"Variable142LAST")
trdata <- trdata[, ! names(trdata) %in% cols.dont.want, drop = F]
```

Handling missing values

Function to get the mode of given vector

```
getmode <- function(v) {
     uniqv <- unique(v)
     uniqv[which.max(tabulate(match(v, uniqv)))]
 }
```

For columns which have only two unique values other than NA, replace all NA values with the mode of that column, here there are four such columns

```
for (i in 1:519) { if(length(unique(trdata[,i])) == 3) {
     resultmode <- getmode(trdata[,i])
     trdata[,i][is.na(trdata[,i])] <- resultmode
 } }
```

Replace NA values of remaining columns by mean value of the column

```
for (i in 1:519) { for (j in 1:5922) {if (is.na(trdata[j,i])) trdata[j,i] =
mean(trdata[,i], na.rm = T)} }
```

Create Returns Variables

Remove Timestamp column, Calculate Returns Variable for each independent variable by subtracting each value by it's value after one hour, Add TargetVariable column and remove last 12 observations as they are not converted into Returns Variables

```
cols.dont.want <- "Timestamp"
trdata <- trdata[, ! names(trdata) %in% cols.dont.want, drop = F]
for(i in 1:518) { for(j in 1:5910) { trdata[j,i] = (trdata[j,i]-
trdata[j+12,i]) } }
trdata$TargetVariable = data$TargetVariable
trdata = trdata[1:5910,]
train = trdata[1:4000,]
test = trdata[4001:5910,]
```

Logistic Regression models:

model1: with all possible independent variables

```
model1 = glm (TargetVariable ~ .,data=train , family = binomial)
```

Interpreting results of our model

```
predicttrain = predict(model1, type="response")
table(train$TargetVariable, predicttrain>0.5)
```

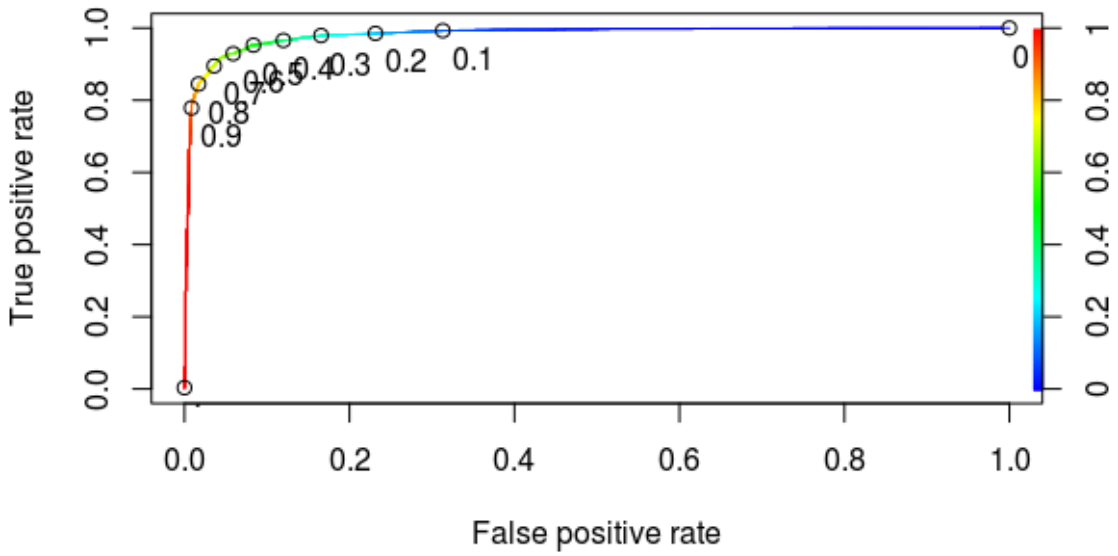
gave accuracy of 0.93 on train data

```
predicttest = predict(model1 , type = "response" , newdata = test)
table(test$TargetVariable , predicttest>0.5)
```

gave accuracy of 0.81 on test data

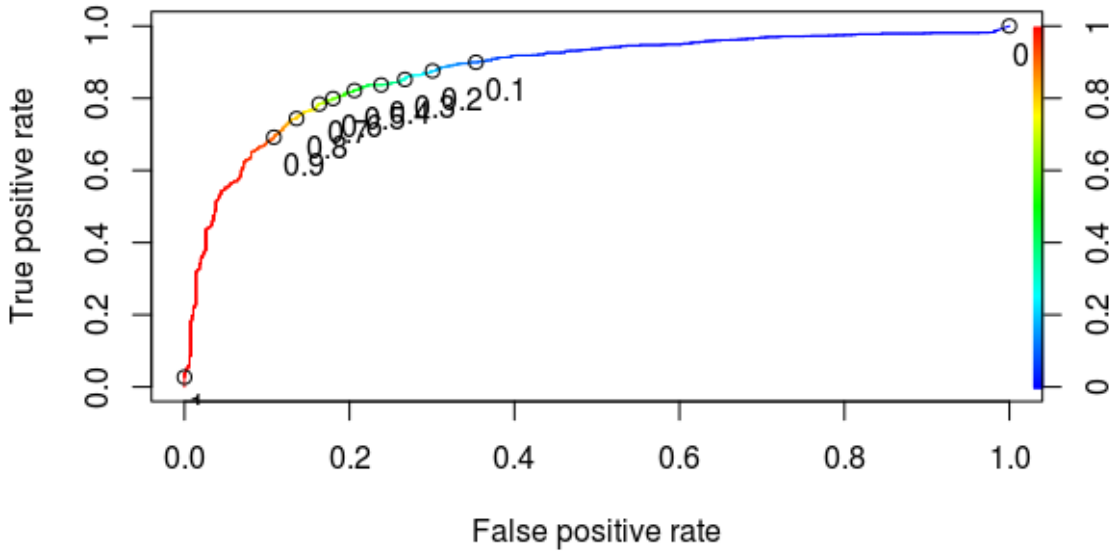
ROC plots to access quality of the model:

```
ROCRpred = prediction(predicttrain, train$TargetVariable)
ROCRperf = performance(ROCRpred, "tpr", "fpr")
auc.tmp <- performance(ROCRpred,"auc")
auc <- as.numeric(auc.tmp@y.values)
plot(ROCRperf, colorize=TRUE, print.cutoffs.at=seq(0,1,0.1), text.adj=c(-
0.2,1.7))
```



AUC for ROC plot gave 0.98 for predicttrain

similarly ROC plot for prediction on test data gave:



AUC for ROC plot gave 0.87 for predicttest

model2: improving model1 by stepwise variable selection

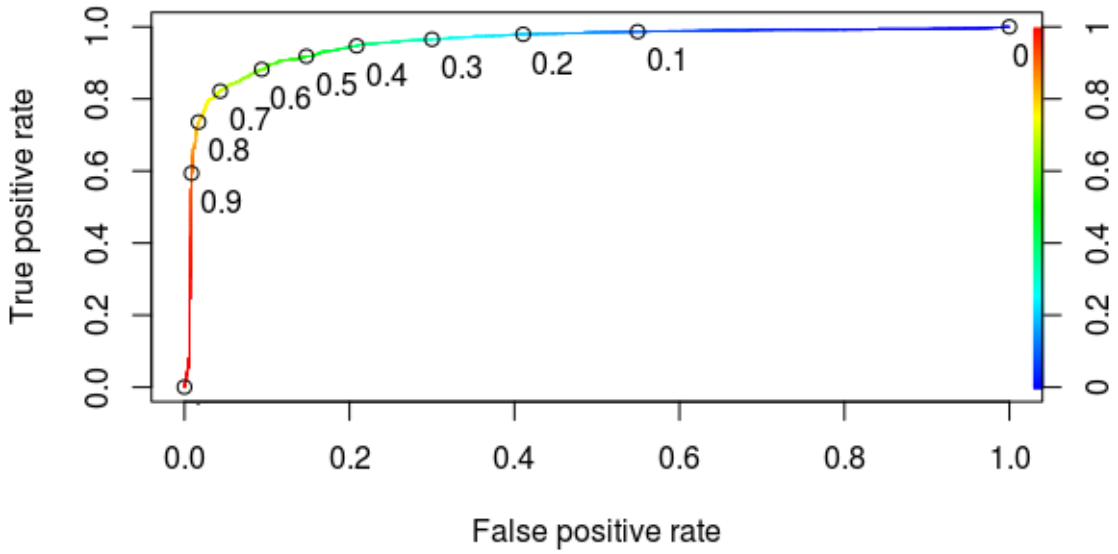
```
nomodel <- glm(TargetVariable ~ 1, data=train, family=binomial)
forwards = step(nomodel,
scope=list(lower=formula(nomodel),upper=formula(model1)),
direction="forward")
```

By running this function we can select the following three variables by forward stepwise variable selection algorithm: "Variable74LAST_PRICE", "Variable55LAST_PRICE" and "Variable169LAST"

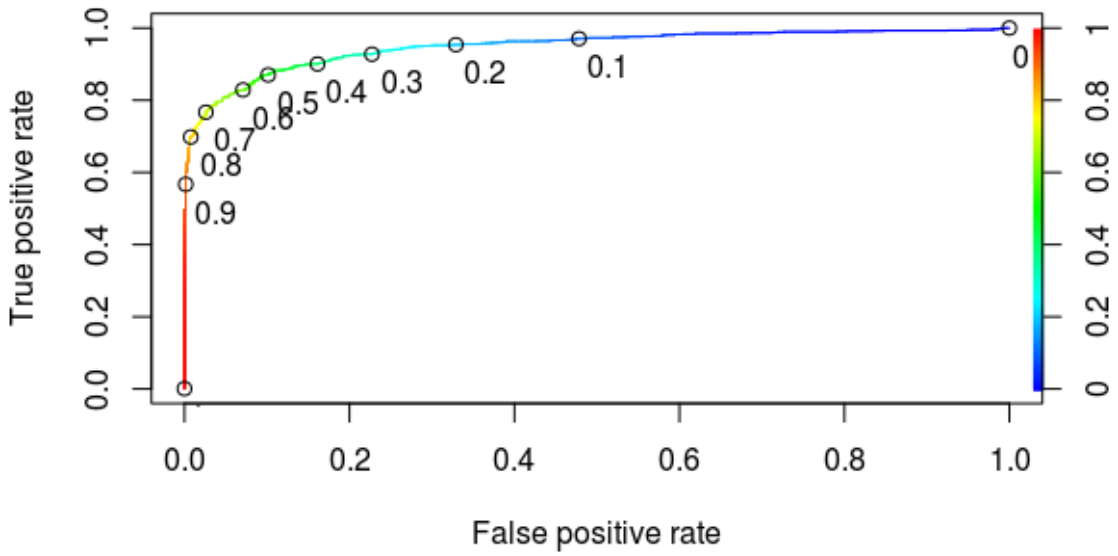
```
model2 = glm(TargetVariable ~ Variable74LAST_PRICE + Variable55LAST_PRICE +
Variable169LAST, data=train, family = binomial)
```

Interpreting results similar to model1 as above gave following for model2: accuracy on test data =0.88, train data = 0.89, auc for ROC plot for predicttrain gave 0.95 and predicttest gave 0.94

model2 ROC plot for predicttrain:



model2 ROC plot for predicttest:

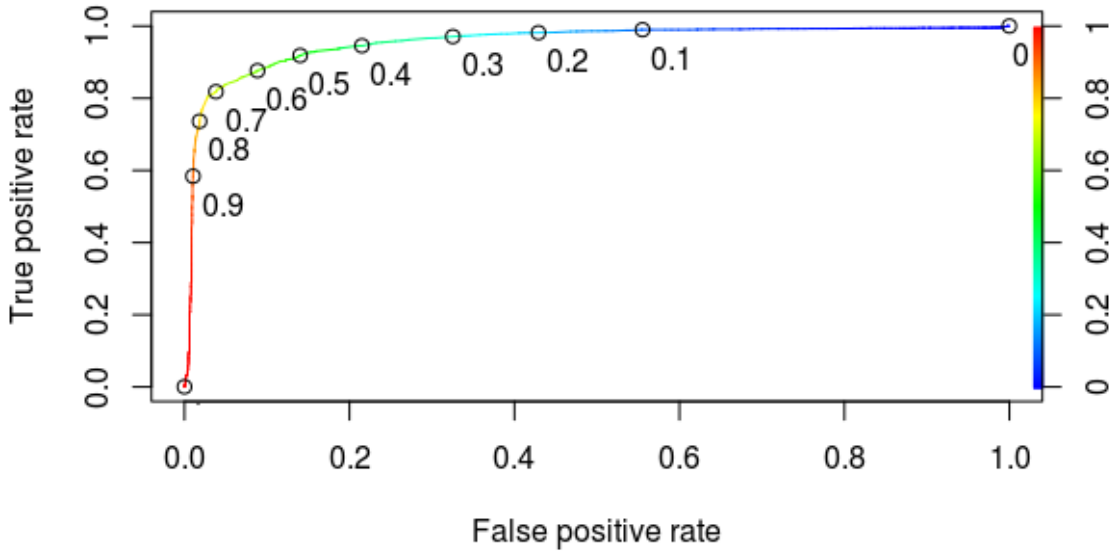


model3: logistic regression on most correlated variable only

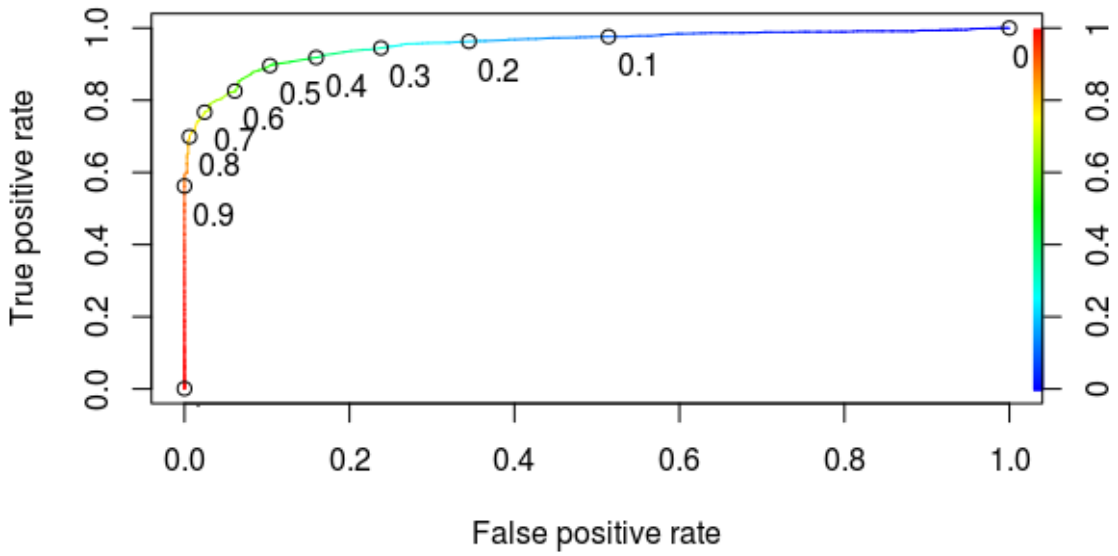
```
model3 = glm(TargetVariable ~ Variable74LAST_PRICE, data=train, family =
binomial)
```

This model gave: accuracy on test data =0.893, train data = 0.894, auc for ROC plot for predicttrain gave 0.955 and predicttest gave 0.954

model3 ROC plot for predicttrain:



model3 ROC plot for predicttest:



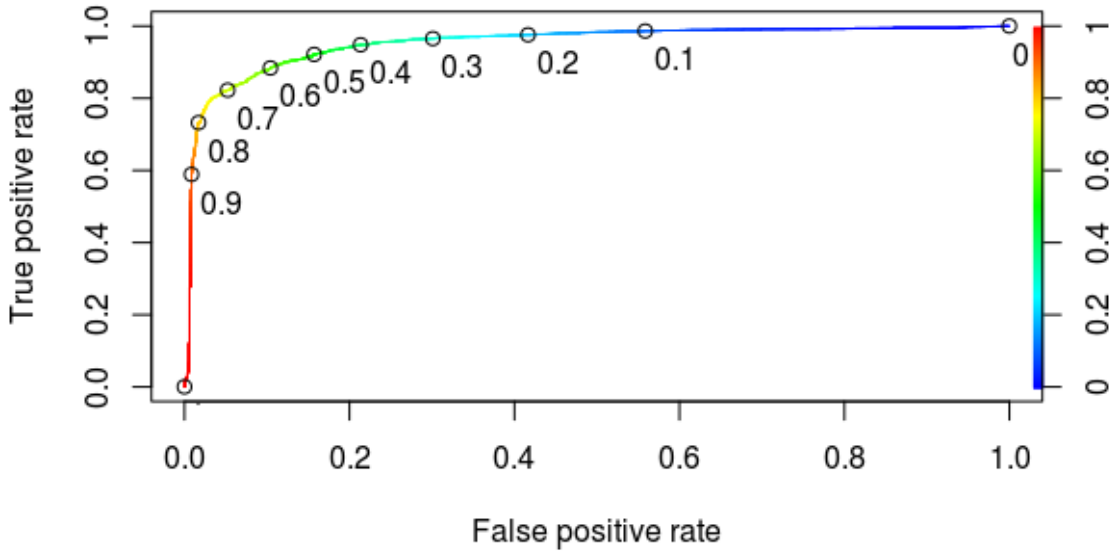
model4: Time-of-Day normalization of Returns

```
cols.dont.want <- "TargetVariable"
subset1 = trdata[,!names(trdata) %in% cols.dont.want]
#there are 79 observations for each day in the given data set
```

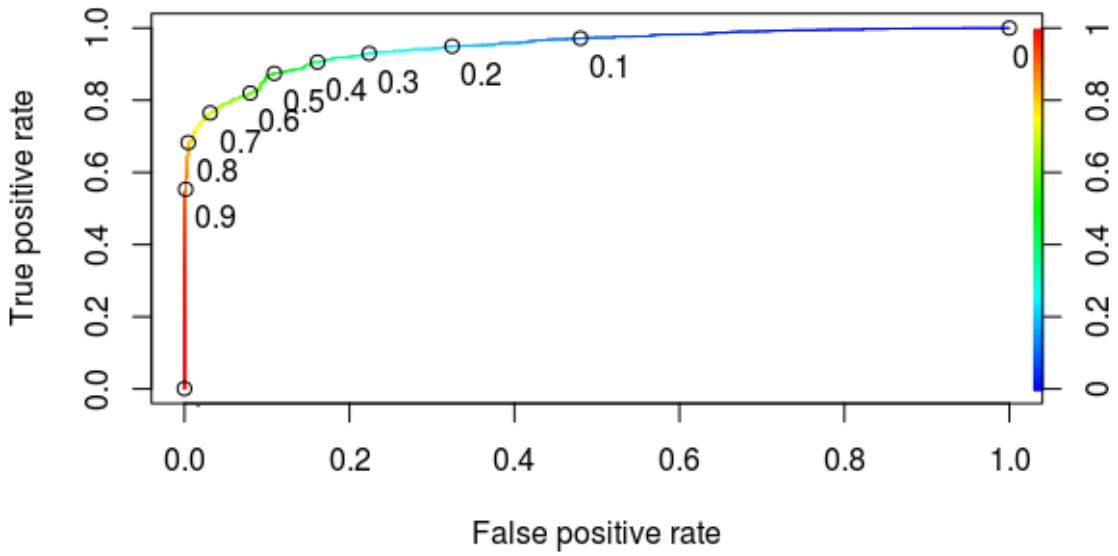
```
for(i in 1:518){ for (j in 1:79){
  sdvector = c(subset1[j,i])
  k = j+80
  while(k<=5910){
   sdvector <- c(sdvector, subset1[k,i])
   k = k+80
  }
 sdvalue = sd(sdvector, na.rm = FALSE)
 if (sdvalue != 0){
  l = j
  while(l<=5910){
   subset1[l,i] = subset1[l,i]/sdvalue
   l = l+80
  }
 } else {
  print(i)
  print(j)
 }
  } }
subset1$TargetVariable = trdata$TargetVariable
train1 = subset1[1:4029,]
test1 = subset1[4030:5910,]
model4 = glm(TargetVariable ~ Variable74LAST_PRICE + Variable55LAST_PRICE +
Variable169LAST, data=train1, family = binomial)
```

This model gave accuracy on test data =0.879, train data = 0.889, auc for ROC plot for predicttrain gave 0.953 and predicttest gave 0.948

model4 ROC plot for predicttrain:



model4 ROC plot for predicttest:



model5: L2 regularization with logistic regression using "LiblineaR" package

```
myvars <- c("Variable74LAST_PRICE", "Variable55LAST_PRICE",
"Variable169LAST")
x = trdata[myvars]
y = factor(trdata[,519])
numbertrain=sample(1:dim(trdata)[1],size=4000,replace = FALSE)
xTrain = x[numbertrain,]
xTest=x[-numbertrain,]
yTrain=y[numbertrain]
yTest=y[-numbertrain]
s=scale(xTrain,center=TRUE,scale=TRUE)
model5 =
LiblineaR(data=s,target=yTrain,type=0,cost=1,bias=TRUE,cross=5,verbose=FALSE)
s2=scale(xTest,attr(s,"scaled:center"),attr(s,"scaled:scale"))
pr=TRUE  #should be true for type=0 or type=7 in LiblineaR function
p=predict(model5,s2,proba=pr,decisionValues=TRUE)
res=table(p$predictions,yTest)
BCR=mean(c(res[1,1]/sum(res[,1]),res[2,2]/sum(res[,2])))
```

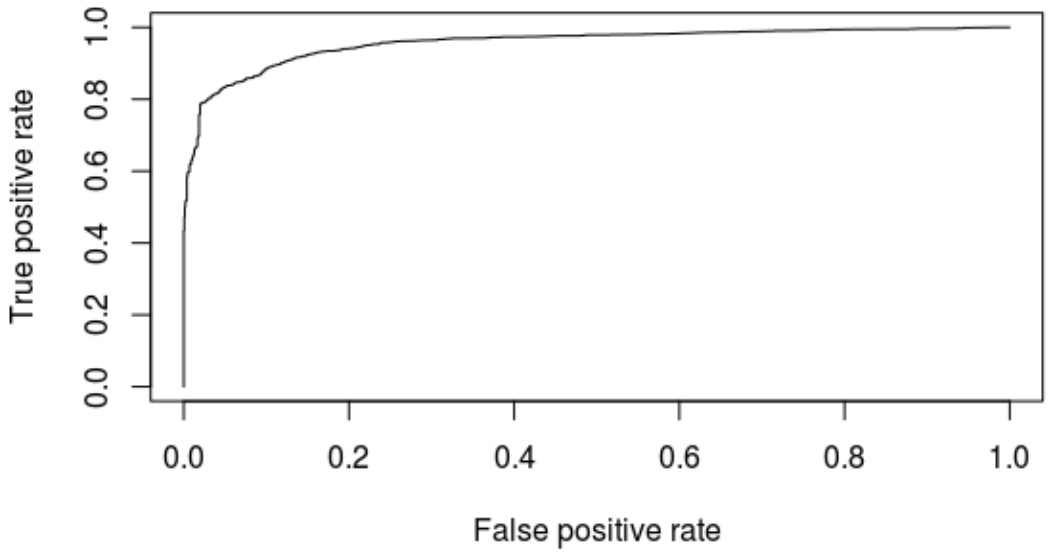
This model gave accuracy of 0.897, BCR value is 0.886, on 5-fold cross validation gave an average accuracy of 0.8865

SVM models using "e1071" package:

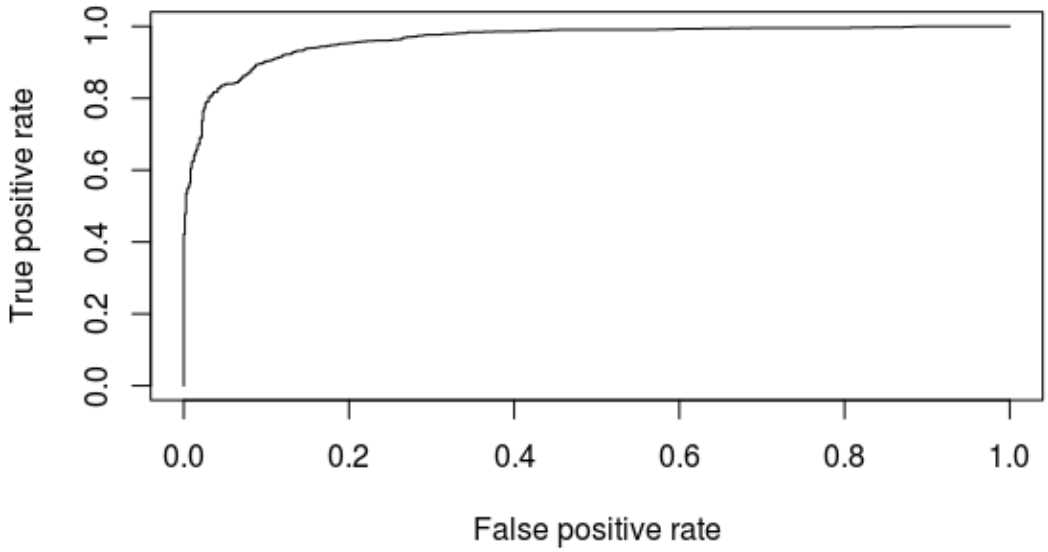
model6: by using radial kernel

```
model6 <- svm(xTrain, yTrain, type='C-classification', kernel='radial',
cost=1)
svmpred <- predict(model6, xTest,decision.values = TRUE)
print(confusionMatrix(svmpred,yTest))
# "caret" package is needed for confusionMatrix function
# this model gave accuracy of 0.904 on test data and 0.894 on train data
svm.roc <- prediction(attributes(svmpred)$decision.values, yTest)
svm.auc <- performance(svm.roc, 'tpr', 'fpr')
aucsvm <- performance(svm.roc, 'auc')
plot(svm.auc)
auc <- as.numeric(aucsvm@y.values)  # gave auc of 0.964 for test data and
0.957 for train data
```

model6 ROC plot for predicttrain:



model6 ROC plot for predicttest:

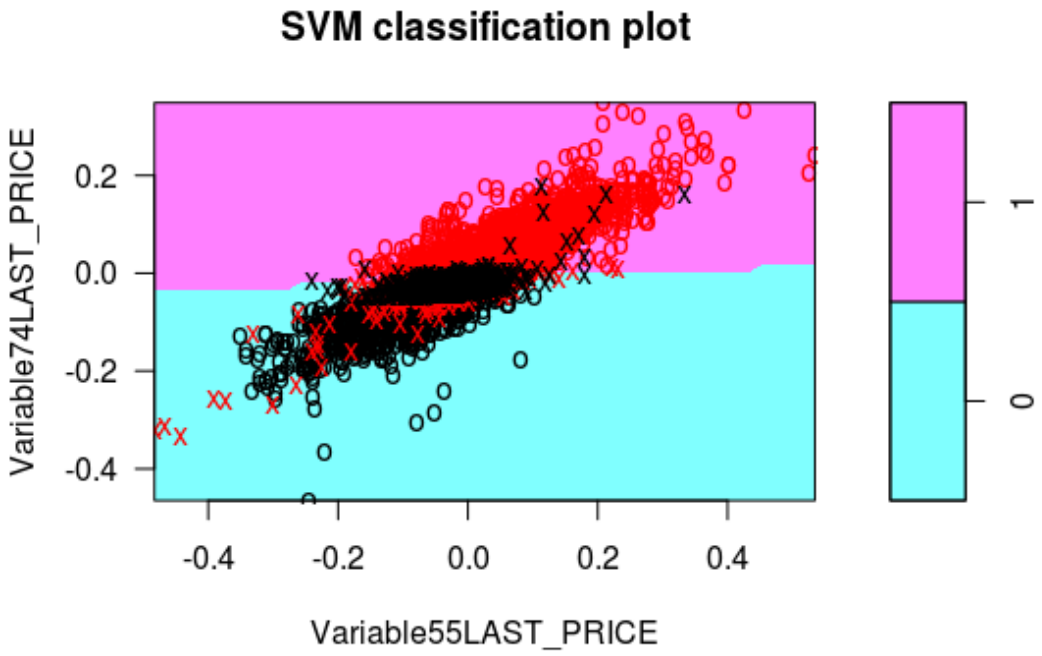


sample models: Taking only top two correlated variables so as to get SVM classification plot of svm object

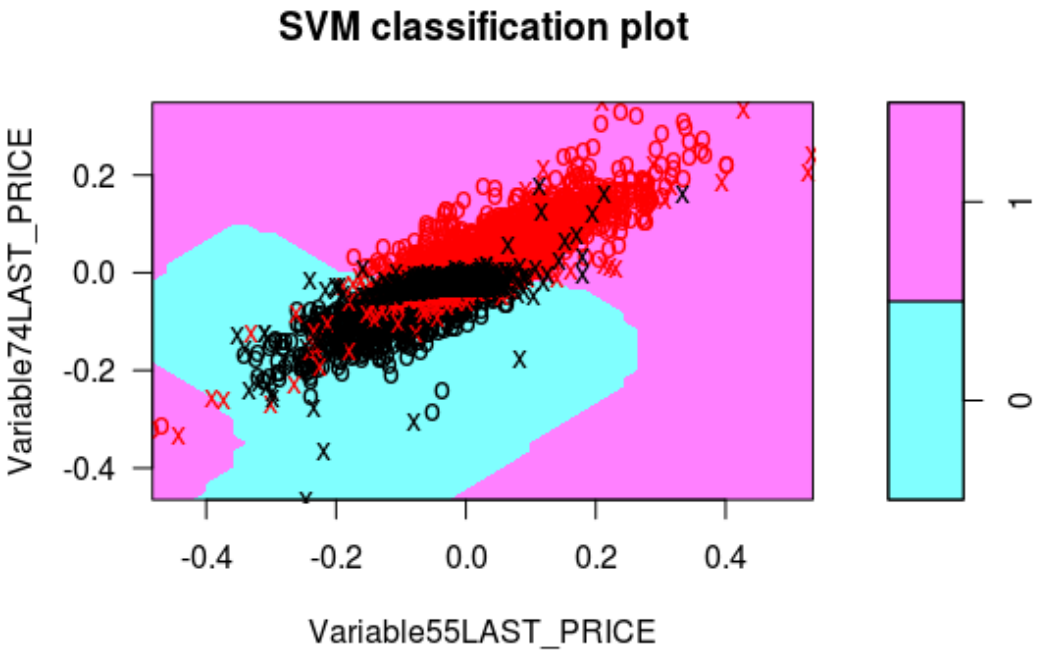
```
myvars <- c("Variable74LAST_PRICE", "Variable55LAST_PRICE")
S = trdata[myvars]
```

```
sTrain=S[numbertrain,]
sTest=S[-numbertrain,]
sTrain$TargetVariable = yTrain
sTest$TargetVariable = yTest
fit = svm(TargetVariable ~ ., data=sTrain, type='C-classification',
kernel='linear')
plot(fit, sTrain)
fit1 = svm(TargetVariable ~ ., data=sTrain, type='C-classification',
kernel='radial')
plot(fit1, sTrain)
```

For Linear Kernel:

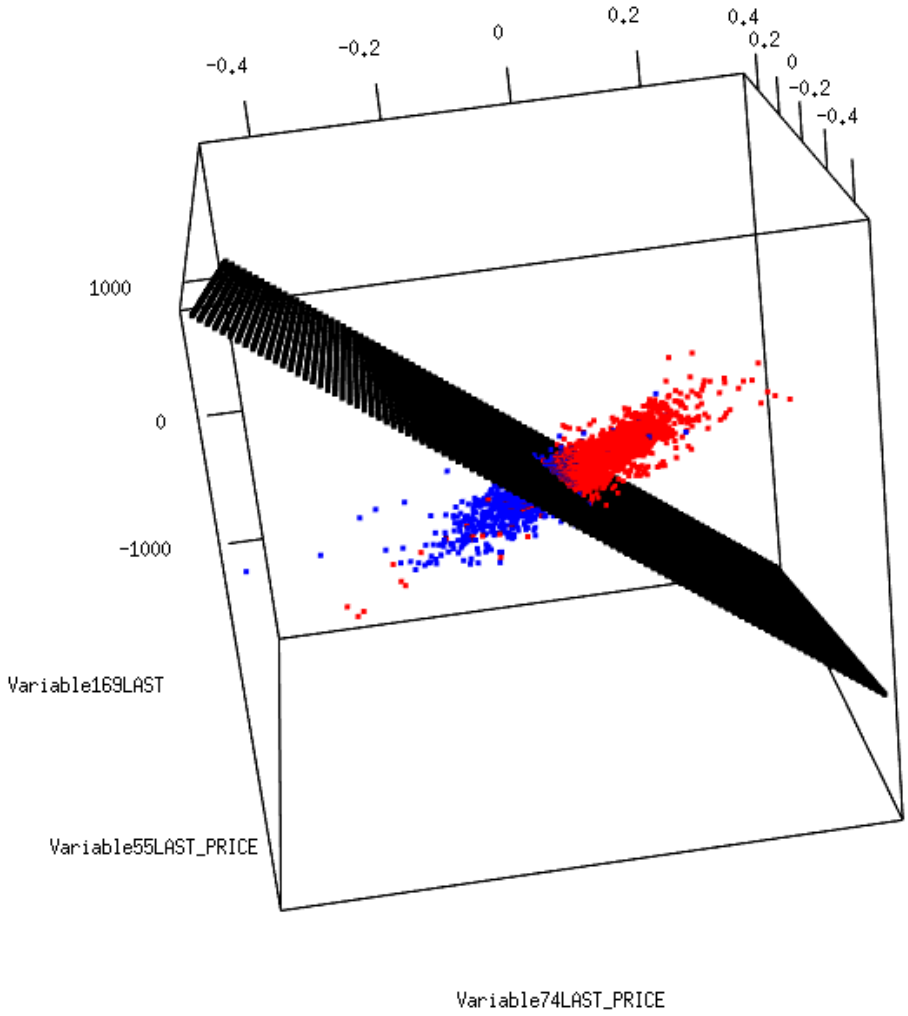


For radial kernel:



model7: 3D plot of SVM classification using "rgl" package

```
myvars <- c("Variable74LAST_PRICE", "Variable55LAST_PRICE",
"Variable169LAST")
m = trdata[myvars]
mTrain=m[numbertrain,]
mTest=m[-numbertrain,]
mTrain$TargetVariable = yTrain
mTest$TargetVariable = yTest
svm_model <- svm(TargetVariable~., data=mTrain, type='C-classification',
kernel='linear',scale=FALSE)
w <- t(svm_model$coefs) %*% svm_model$SV
detalization <- 100
grid <-
expand.grid(seq(from=min(mTrain$Variable74LAST_PRICE),to=max(mTrain$Variable7
4LAST_PRICE),length.out=detalization),seq(from=min(mTrain$Variable55LAST_PRIC
E),to=max(mTrain$Variable55LAST_PRICE),length.out=detalization))
z <- (svm_model$rho- w[1,1]*grid[,1] - w[1,2]*grid[,2]) / w[1,3]
plot3d(grid[,1],grid[,2],z, xlab="Variable74LAST_PRICE",
ylab="Variable55LAST_PRICE", zlab="Variable169LAST")
points3d(mTrain$Variable74LAST_PRICE[which(mTrain$TargetVariable==1)],mTrain$
Variable55LAST_PRICE[which(mTrain$TargetVariable==1)],mTrain$Variable169LAST[
which(mTrain$TargetVariable==1)], col='red')
points3d(mTrain$Variable74LAST_PRICE[which(mTrain$TargetVariable==0)],mTrain$
Variable55LAST_PRICE[which(mTrain$TargetVariable==0)],mTrain$Variable169LAST[
which(mTrain$TargetVariable==0)], col='blue')
rgl.postscript("rglplot.pdf","pdf")
```



Conclusion: SVM model gave better accuracy for this data set when compared to Logistic Regression.