**PROJECT REPORT**

**ITA6015 – Accounting & Financial Management**

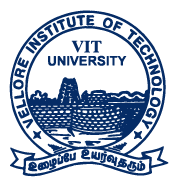
**Anomaly Detection Using Auto Encoder**

*Submitted By*

DIVYA B (18MCA1049)

*Under the guidance of*

DR. SHARON SOPHIA



School of Computing Science and Engineering

**June 2019**

**Abstract**

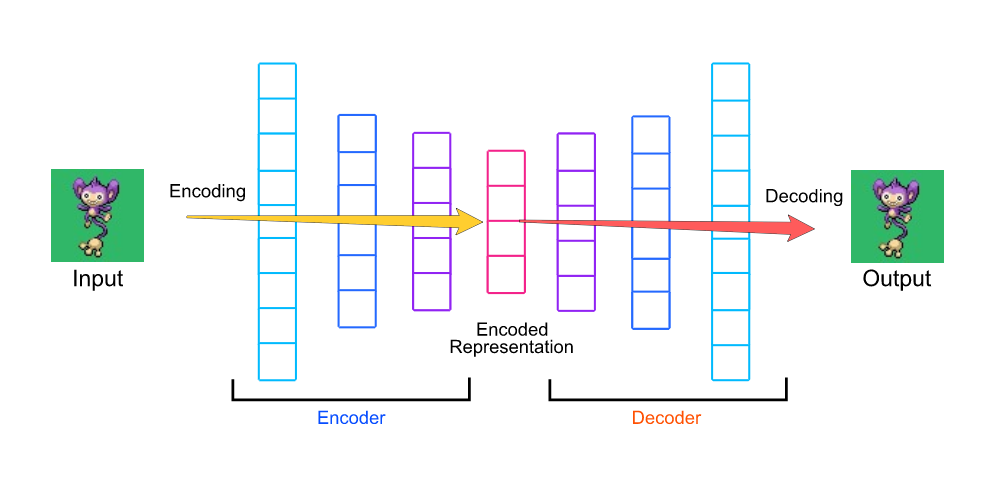
In [data mining](https://en.wikipedia.org/wiki/Data_mining), **anomaly detection** (also **outlier detection**) is the identification of rare items, events or observations which raise suspicions by differing significantly from the majority of the data.[[1]](https://en.wikipedia.org/wiki/Anomaly_detection#cite_note-:0-1) Typically the anomalous items will translate to some kind of problem such as [bank fraud](https://en.wikipedia.org/wiki/Bank_fraud), a structural defect, medical problems or errors in a text. Anomalies are also referred to as [outliers](https://en.wikipedia.org/wiki/Outlier), novelties, noise, deviations and exceptions**.** In this project Auto Encoders (AEs) model with Random Forest model in deep learning are. Auto encoder model first performs for unsupervised feature learning in a layer wise manner then followed with supervised parameter fine-tuning. By finding the reconstruction error with the help of auto encoder, random forest classifier classifies the file is anomaly or not.

**Introduction**

Anomaly detection is a technique used to identify unusual patterns that do not conform to expected behavior, called outliers. It has many applications in business, from intrusion detection (identifying strange patterns in network traffic that could signal a hack) to system health monitoring (spotting a malignant tumor in an MRI scan), and from fraud detection in credit card transactions to fault detection in operating environments.

**AutoEncoders:**

Neural networks can slowly approximate any function that maps inputs to outputs through an iterative optimization process also called as training. If we set the output to be the same as the input, we call this neural network as an auto encoder because it encodes a more dense data representation of input data. It is a lower dimensional compression of input that preserves its features. This representation is widely used in image colorization, dialogue generation and anomaly detection



**Code**

prosPath = system.file("extdata", "prostate.csv", package = "h2o")

prostate\_df <- read.csv(prosPath)

prostate\_df <- prostate\_df[,-1]

summary(prostate\_df)

#splitting into training and testing data

set.seed(1234)

random\_splits <- runif(nrow(prostate\_df))

train\_df <- prostate\_df[random\_splits < .5,]

dim(train\_df)

validate\_df <- prostate\_df[random\_splits >=.5,]

dim(validate\_df)

#applying random forest model

install.packages('randomForest')

library(randomForest)

outcome\_name <- 'CAPSULE'

feature\_names <- setdiff(names(prostate\_df), outcome\_name)

print(feature\_names)

set.seed(1234)

rf\_model <- randomForest(x=train\_df[,feature\_names],

y=as.factor(train\_df[,outcome\_name]),

importance=TRUE, ntree=20, mtry = 3)

validate\_predictions <- predict(rf\_model, newdata=validate\_df[,feature\_names], type="prob")

#calculating auc score for the model

install.packages('pROC')

library(pROC)

auc\_rf = roc(response=as.numeric(as.factor(validate\_df[,outcome\_name]))-1,

predictor=validate\_predictions[,2])

plot(auc\_rf, print.thres = "best", main=paste('AUC:',round(auc\_rf$auc[[1]],3)))

abline(h=1,col='blue')

abline(h=0,col='green')

#using autoencoder for diving into hard and easy sets based on reconstruction error

library(h2o)

localH2O = h2o.init()

prostate.hex<-as.h2o(train\_df, destination\_frame="train.hex")

prostate.dl = h2o.deeplearning(x = feature\_names, training\_frame = prostate.hex,

autoencoder = TRUE,

reproducible = T,

seed = 1234,

hidden = c(6,5,6), epochs = 50)

prostate.anon = h2o.anomaly(prostate.dl, prostate.hex, per\_feature=FALSE)

head(prostate.anon)

err <- as.data.frame(prostate.anon)

plot(sort(err$Reconstruction.MSE), main='Reconstruction Error')

# applying random forest over data split according to reconstruction error < 0.1

train\_df\_auto <- train\_df[err$Reconstruction.MSE < 0.1,]

set.seed(1234)

rf\_model <- randomForest(x=train\_df\_auto[,feature\_names],

y=as.factor(train\_df\_auto[,outcome\_name]),

importance=TRUE, ntree=20, mtry = 3)

validate\_predictions\_known <- predict(rf\_model, newdata=validate\_df[,feature\_names], type="prob")

auc\_rf = roc(response=as.numeric(as.factor(validate\_df[,outcome\_name]))-1,

predictor=validate\_predictions\_known[,2])

plot(auc\_rf, print.thres = "best", main=paste('AUC:',round(auc\_rf$auc[[1]],3)))

abline(h=1,col='blue')

abline(h=0,col='green')

#applying random forest over data split according to reconstruction error > 0.1

train\_df\_auto <- train\_df[err$Reconstruction.MSE >= 0.1,]

set.seed(1234)

rf\_model <- randomForest(x=train\_df\_auto[,feature\_names],

y=as.factor(train\_df\_auto[,outcome\_name]),

importance=TRUE, ntree=20, mtry = 3)

validate\_predictions\_unknown <- predict(rf\_model, newdata=validate\_df[,feature\_names], type="prob")

auc\_rf = roc(response=as.numeric(as.factor(validate\_df[,outcome\_name]))-1,

predictor=validate\_predictions\_unknown[,2])

plot(auc\_rf, print.thres = "best", main=paste('AUC:',round(auc\_rf$auc[[1]],3)))

abline(h=1,col='blue')

abline(h=0,col='green')

#taking equal sets of data from both the splits

valid\_all <- (validate\_predictions\_known[,2] + validate\_predictions\_unknown[,2]) / 2

auc\_rf = roc(response=as.numeric(as.factor(validate\_df[,outcome\_name]))-1,

predictor=valid\_all)

plot(auc\_rf, print.thres = "best", main=paste('AUC:',round(auc\_rf$auc[[1]],3)))

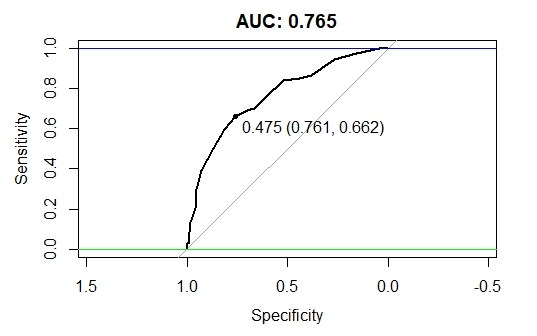
abline(h=1,col='blue')

abline(h=0,col='green')

**Result**

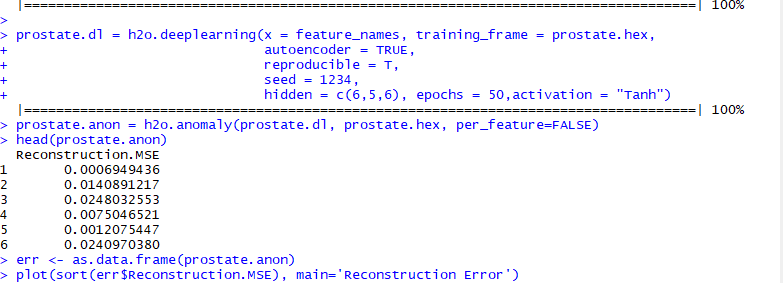
**Random Forest**

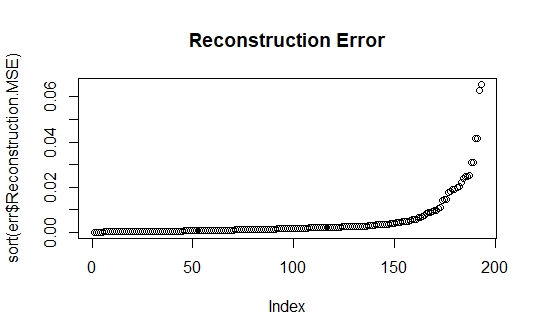
* Random Forest is a supervised learning algorithm.
* Random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction.



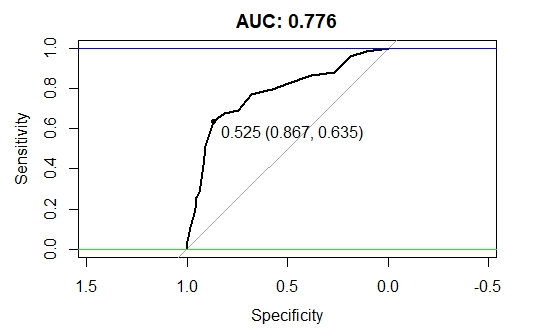
**Autoencoder**

**Reconstruction Error**

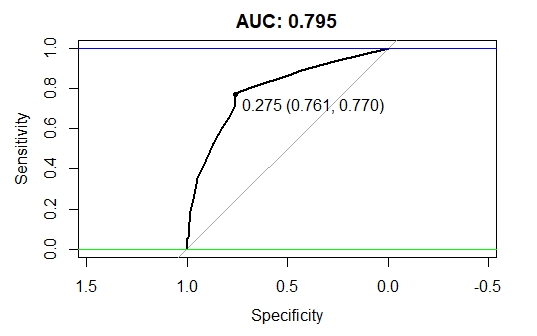




**Reconstruction Error >0.01**

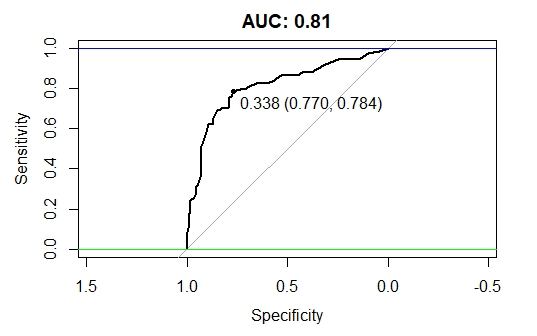


**Reconstruction Error <= 0.01**



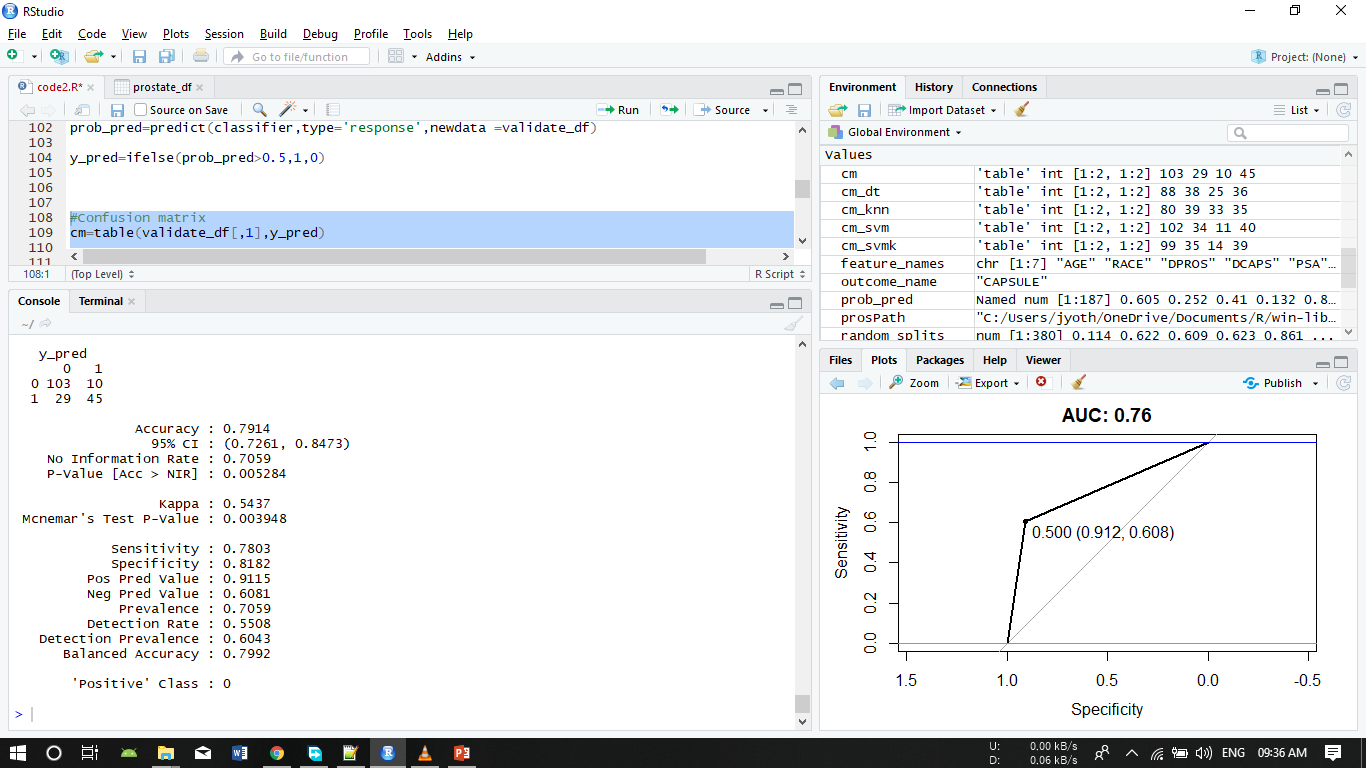
**Joining both result**

Adding both prediction vectors together and dividing the total by two



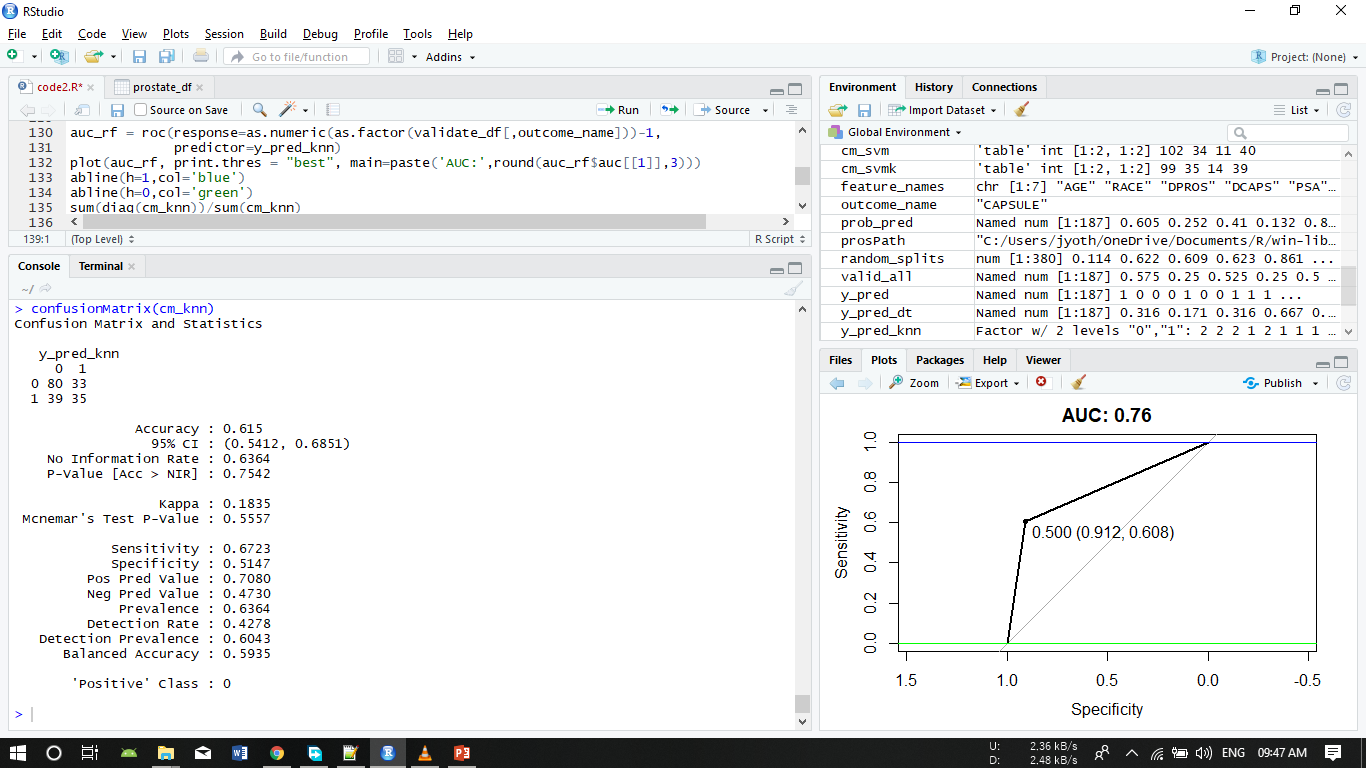
**Logistic Regression Model**

* [Logistic regression](http://www.statisticssolutions.com/academic-solutions/membership-resources/member-profile/data-analysis-plan-templates/data-analysis-plan-logistic-regression/) is the appropriate regression analysis to conduct when the dependent variable is dichotomous (binary).
* Like all regression analyses, the logistic regression is a predictive analysis.
* Logistic regression is used to describe data and to explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables.



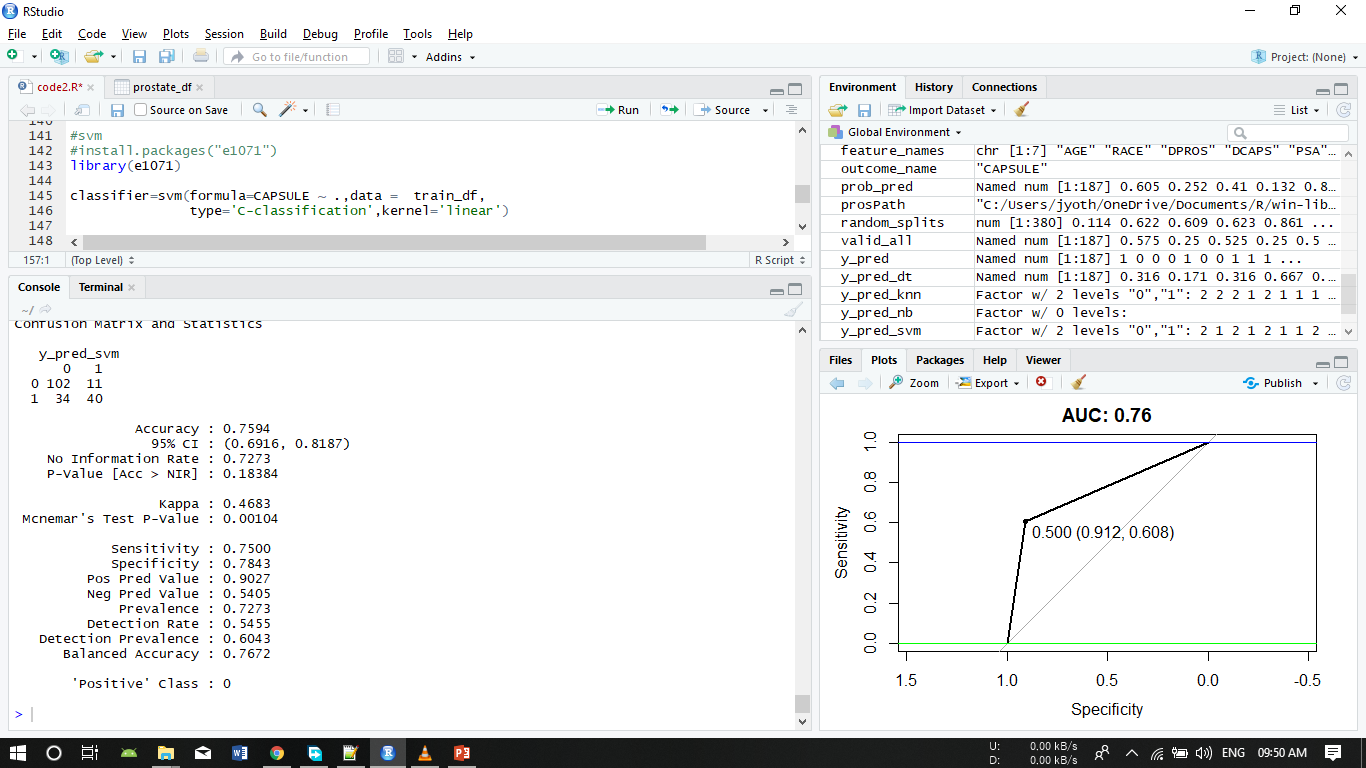
**KNN**

**KNN** is a **non-parametric, lazy**learning algorithm. Its purpose is to use a database in which the data points are separated into several classes to predict the classification of a new sample point.



**SVM**

* SVMs are based on the idea of finding a hyperplane that best divides a dataset into two classes
* Support vectors are the data points nearest to the hyperplane, the points of a data set that, if removed, would alter the position of the dividing hyperplane. Because of this, they can be considered the critical elements of a data set.



**Kernel SVM**

* SVM algorithms use a set of mathematical functions that are defined as the kernel. The function of kernel is to take data as input and transform it into the required form. Different SVM algorithms use different types of kernel functions. These functions can be different types. For example***linear, nonlinear, polynomial, radial basis function (RBF), and sigmoid.***
* 