Failure prediction in bosch production lines

**M.LOKIN SAI (13Z232)**

12Z820 - Project Work - I

Dissertation submitted in partial fulfillment of the requirements for the degree of

**BACHELOR OF ENGINEERING**

BRANCH: **COMPUTER SCIENCE AND ENGINEERING**

**Of Anna University**



**April 2017**

**department of computer science and engineering**

**PSG COLLEGE OF TECHNOLOGY**

(Autonomous Institution)

**COIMBATORE - 641 004**

**PSG COLLEGE OF TECHNOLOGY**

(Autonomous institution)

**COIMBATORE - 641 004**

**FAILURE PREDICTION IN BOSCH PRODUCTION LINES**

12Z820 - Project Work - I

Bonafide record of work done by

**M.LOKIN SAI (13Z232)**

Dissertation submitted in partial fulfillment of the requirements for the degree of

**BACHELOR OF ENGINEERING**

BRANCH**: COMPUTER SCIENCE AND ENGINEERING**

**Of Anna University**

**April 2017**

**.................................................... .........................................................**

**Dr.L.S.JAYASHREE Dr.R.VENKATESAN**

**FACULTY GUIDE HEAD OF THE DEPARTMENT**

CONTENTS

**CHAPTER Page No.**

[ACKNOWLEDGMENT 4](#_Toc481132037)

[ABSTRACT 5](#_Toc481132038)

[LIST OF FIGURES 6](#_Toc481132039)

[1. INTRODUCTION 7](#_Toc481132040)

[1.1. PROBLEM DEFINITION 7](#_Toc481132041)

[1.2. OBJECTIVES 7](#_Toc481132042)

[1.3. SCOPE AND PURPOSE OF THE PROJECT 7](#_Toc481132043)

[1.4. PROJECT DOMAIN KNOWLEDGE 8](#_Toc481132044)

[2. SYSTEM ANALYSIS 9](#_Toc481132045)

[2.1. SYSTEM ENVIRONMENT 9](#_Toc481132046)

[3. LITERATURE SURVEY 10](#_Toc481132047)

[4. DESCRIPTION 11](#_Toc481132048)

[4.1. PREDICTIVE MODELLING 11](#_Toc481132049)

[4.2 STEPS IN PREDICTIVE MODELLING 11](#_Toc481132050)

[4.2.1 DEFINING THE OBJECTIVE 11](#_Toc481132051)

[4.2.2 GATHERING THE DATA 11](#_Toc481132052)

[4.2.3 PREPARING THE DATA MODELLING 12](#_Toc481132053)

[4.2.4 FEATURE SELECTION AND TRANSFORMATION 12](#_Toc481132054)

[4.2.5 PROCESSING AND EVALUATING THE MODEL 12](#_Toc481132055)

[4.2.6 VALIDATING THE MODEL 13](#_Toc481132056)

[4.3 DEFINING BUSINESS PROBLEM 13](#_Toc481132057)

[4.4 COLLECTING REAL TIME BOSCH DATASETS 13](#_Toc481132058)

[4.5 DATA PREPARATION AND TRANSFORMATION 14](#_Toc481132059)

[4.6 SELECTING RELEVANT ATTRIBUTES 14](#_Toc481132060)

[4.8 VALIDATING THE MODEL 15](#_Toc481132061)

[5. IMPLEMENTATION 16](#_Toc481132062)

[5.1. DATA IMPUTATION 16](#_Toc481132063)

[5.2. BALANCING THE IMBALANCE CLASS 17](#_Toc481132064)

[5.4. CENTER AND SCALING 18](#_Toc481132065)

[5.5. BORUTA FEATURE SELECTION ALGORITHM 19](#_Toc481132066)

[5.6. PRINCIPAL COMPONENT ANALYSIS 20](#_Toc481132067)

[5.7. LOGISTIC REGRESSION 21](#_Toc481132068)

[5.8. SUPPORT VECTOR MACHINE 22](#_Toc481132069)

[5.9. DECISION TREE MODELLING 23](#_Toc481132070)

[5.10. MODELLING AND DIVIDE TRAIN, TEST 24](#_Toc481132071)

[5.11. R CODE FOR SAMPLE PREDICTIONS FOR ALL MODELS 29](#_Toc481132072)

[5.12. FINAL R CODE OF PREDICTIVE MODELLING 32](#_Toc481132073)

[6. CONCLUSION 38](#_Toc481132074)

[7. REFERENCES 39](#_Toc481132075)

# ACKNOWLEDGMENT

A very sincere thanks to **Dr.R.Rudramoorthy,** Principal, PSG College Of Technology for giving the opportunity to express gratitude and profound and for providing a wonderful opportunity to develop and complete a project.

A sincere thanks to **Dr.R.Venkatesan,** Professor and Head, Department of Computer Science and Engineering, for his support towards the completion of our project work.

A sincere thanks to **Dr.L.S.Jayshree,** Professor, Department of Computer Science and Engineering, for acting as our guide and for her constant support and help throughout the course and for his ideas and encouragement in completing our project work.

And also a sincere thanks to **Dr.G.R.Karpagam,** Professor and Program coordinator, Department of Computer Science and Engineering, for her support.

We also would like to thank our tutor **Dr.V.Santhi,** Professor, Department of Computer Science and Engineering, in helping us out with the nuances of the project report to be submitted.

Finally, a sincere thanks to all the faculty of the Department of Computer Science and Engineering, for their constant help in resolving any problems incurred in the duration of our project work.

# ABSTRACT

The idea of the project is to predict the failures in Bosch Production Lines which consists of 4 lines and 51 stations. We need to predict the failures which ideally should be more accurate in predicting the failures. The project moves forward with detail steps of data cleaning, variable transformation, feature selection, data modelling and finally modelling validation. The feature selection process is done by Boruta feature selection which segregates all relevant attributes using Z statistic score as threshold. We have also used PCA (Principal Component Analysis) which converts higher dimensional data to lower dimensions which consists of components. To cover most of the variance from the dataset we took the top 100 components and used for modelling. The project also uses three types of modelling as follows: Logistic regression, Support vector machine and Decision Tree. The three models with their predicted probabilities are ensemble using stack method thereby creating a supervised model from these base models to improve the accuracy.

# LIST OF FIGURES

[Fig 1.3: Bosch Assembly Lines 8](#_Toc481132012)

[Fig. 5.1: Imputation with Zero 16](#_Toc481132013)

[Fig 5.4: Center and scaled 18](#_Toc481132014)

[Fig 5.5: Boruta output 19](#_Toc481132015)

[Fig 5.6: Principal Components 20](#_Toc481132016)

[Fig 5.7: Logistic Regression graph 21](#_Toc481132017)

[Fig 5.8: Support Vector Machine 22](#_Toc481132018)

[Fig 5.9: Decision Tree 23](#_Toc481132019)

[Fig 5.10.1: Confusion Matrix of logistic regression 25](#_Toc481132020)

[Fig 5.10.2: Confusion Matrix of SVM 26](#_Toc481132021)

[Fig 5.10.3: Confusion Matrix of Decision Tree 27](#_Toc481132022)

[Fig 5.10.4: Confusion Matrix of Weighted Average Ensemble 28](#_Toc481132023)

[Fig 5.10.5: Model metrics for all models 28](#_Toc481132024)

CHAPTER 1

# INTRODUCTION

## 1.1. PROBLEM DEFINITION

To develop a machine learning model and predict the failures in Bosch hardware production lines from sensor information which reduces the cost of failure for quality product

## 1.2. OBJECTIVES

* To do feature selection and variable transformations used for selection of relevant variables these are the set of predictors which are essential to predict the binary output.
* To perform principal component analysis, this technique is used to decrease the dimensionality of features into components of predictors showing most variation in the first component and components are orthogonal between each other.
* To understand different model tuning parameters and their respective metrics, some models are well versed in predicting good and rich dataset and some would not perform well on such datasets but even though model is powerful.
* To do Ensemble learning, which uses multiple learning algorithms to obtain better predictive performance which learns the diversity of all models.

## 1.3. SCOPE AND PURPOSE OF THE PROJECT

The scope of machine learning comes into picture when there are challenges been faced in wide variety of disciplines which involves multi-disciplinary research in collaboration and applications. It has the capacity in making certain decisions or any calculations through processing of algorithms and concepts involved in it. Machine learning as a term refers to the gathering, analysis and processing of Big Data for applications of machines to complex problems. Bosch failure prediction comes under this scope wherein there was need to predict the rare occurrence of failures to avoid unnecessary cost for repairs and augment the quality of their finished products.

## 1.4. PROJECT DOMAIN KNOWLEDGE

Bosch production lines are used to manufacture hardware components with the help of robots and sensors. There two types of production lines classified into synchronous and asynchronous production lines. Basic issues in these production lines are minimal capacity i.e number of machines required to achieve certain throughput. There are other issues such as degree of paralleling, location of buffers in between stations and allocation of tasks.

Synchronous production lines move to next station simultaneously, the number of jobs will always be constant and there no buffers at all. The timeline is further classified into paced and unpaced. In paced, the time allowed to work is limited which may lead to incomplete processing. On the other hand, for unpaced there is no maximum limit with the time. Asynchronous production lines operate job as soon as one becomes available. They consist of starved blocks and number of jobs will get fluctuated.

In our project the Bosch production lines come under synchronous unpaced production lines in which the task times are non-determistic and also show shifted distribution. The product moves to next station as soon as it finishes its work on the previous station. Line stations moves to next stations when all workers complete their respective tasks. In these lines there were overtime labor problems this overtime occurs when the line does not meet the daily production quota



Fig 1.3: Bosch Assembly Lines

CHAPTER 2

# SYSTEM ANALYSIS

## 2.1. SYSTEM ENVIRONMENT

**2.1.1 TRAINING ENVIRONMENT**

* **Hardware requirements:**
  + Processor : Intel Core i5
  + RAM : 6 GB
* **Software requirements:**
  + Operating System : Windows 10
  + Programming language : R

CHAPTER 3

# LITERATURE SURVEY

**3.1 Synchronous Unpaced Flow Lines with Worker differences and Overtime Cost:**[1]

The paper illustrates the design of the synchronous unpaced production lines from the perspective workers who operate at different skill level and also discusses the cons for workers who work overtime if in case the required quota is not meet in the daily production lines. The paper also discusses the green lever policy and shifted distribution of task time which constitutes deterministic time and additional time.

**3.2 Building Predictive Models in R Using the caret Package:** [2]

Caret short for classification and regression training, have numerous tools for developing good predictive modelling. We focused on simplifying training as well as tuning model across wide variety of model techniques also includes, pre-processing data, calculating variable importance and model visualizations.

**3.3 Feature Selection with Boruta Package:** [3]

The article describes a R package Boruta, which is novel feature selection algorithm to find all relevant variables. It is a wrapper around a Random Forest Classification algorithm. The novel algorithm removes features iteratively by setting a threshold which is the Z statistic scores. It removes less relevant than the random probe.

**3.4 ROSE: A package for Binary Imbalanced Learning:** [4]

The articles describe about the package to deal with binary imbalance in the presence of binary imbalance class. The function creates artificial samples of data using smoothed bootstrap approach and allows both phases of estimation and accuracy evaluation of the binary classifier.

CHAPTER 4

# DESCRIPTION

## 4.1. PREDICTIVE MODELLING

Modelling is the processing of forecasting probabilities to predict the response variable using a set of predictors. Each model is made up of required number of predictors, relevant predictors are collected from the required data and a statistical model is formulated. The prediction model is made up of simple linear equation or may be complex neural network. It has many applications associated which are failure prediction, fraud detection and also used for classification and regression. Other applications include planning, disaster recovery.

* 1. **STEPS IN PREDICTIVE MODELLING**
     1. **DEFINING THE OBJECTIVE**

The first step in building the predictive model is to define the objective of the business problem and understand the scope of the problem. Several modelling techniques are discussed such as logistic regression, support vector machine, KNN and decision tree. On the other hand, the domain knowledge regarding the problem are also gathered and relate to the dataset

* + 1. **GATHERING THE DATA**

Collecting data can be ingested from various sources with different formats. It includes different ways in collecting data and generate valid data samples for model development. Data is collected from different data repositories and put into one data store called data warehouse. Accurate, actionable, accessible data is the lifeblood of any successful model.

* + 1. **PREPARING THE DATA MODELLING**

Any modeller spends major amount of time in preparing data such as data cleaning, eliminating data inconsistency, removing noise in data and eliminating outliers or detecting outliers through outlier detection algorithm. Treating missing values through imputation includes through imputation by mean, median or could use any other imputation algorithm such as Amelia.

* + 1. **FEATURE SELECTION AND TRANSFORMATION**

Determining the best fit is the underlying objective of any model which is essentially good for any model. The relation between the independent variables with dependent variable determines the power and longevity of the model. This steps includes binning and transforming independent variables. Feature selection process is the way of collecting the relevant variables, there are various algorithms as well as techniques to collect these variables as such Boruta, Random forest importance, Information gain and OneR. Feature engineering is the process of using domain knowledge to create features that make machine learning algorithms work and is both difficult and expensive.

* + 1. **PROCESSING AND EVALUATING THE MODEL**

After preparation of data using feature engineering and feature selection we go for creating model using the required relevant attributes from the set of independent variables. Creating model and evaluating the best fit model from the models used and understand the nature of data.

* + 1. **VALIDATING THE MODEL**

Validating model by definition, that the model should perform well with hold-out samples and should also perform well with other validation methods like cross validation, repeated cross validation. A true test model is how well does it perform from different time periods. Bootstrapping is the simple technique popularly used for validation purposes. Key variable analysis calculates important factors which are much involved in the model.

* 1. **DEFINING BUSINESS PROBLEM**

Objective of the project has already been proposed by the Bosch as well as the data is the real time collection from different sensors in the production lines. Different modelling techniques have been proposed for failure prediction such as logistic regression which is used for predicting the class variables from predicted probabilities. This technique is basically used when the class variable is categorical. Support Vector Machine abbreviated as SVM which uses the concept of hyperplane which acts linearly separable plane to divide two different classes. Decision tree are formed through information gain which divides the root and its respective children. The decision tree known as rpart in R language will be used normally for classification and regression. Logistic Regression modelling for classification will also be used. After, proper running of the models we ensemble all the models using weighted average method thereby knowing which model is performing well and showing more accuracy. Eventually, predicting on the test dataset.

* 1. **COLLECTING REAL TIME BOSCH DATASETS**

Bosch has given real time data which is the information from different sensors in the production lines. The train numeric dataset which has 970 variables with 59,000 observations. The dataset variables are named as L as line number in the production lines, S as station number and F as feature been manufactured on the product. For example, L0\_S0\_F0. In addition to these attributes, the dataset also contain binary response variable 1 for failure and 0 for non-failure.ID is the unique product id in the production lines.

## 4.5 DATA PREPARATION AND TRANSFORMATION

This step involves imputation, data cleaning, eliminating data inconsistencies. In Bosch dataset the missing values in the numeric dataset are treated with replacing zero. These missing values which are present in the numeric dataset show dependencies between the products within the production lines. The missing values are not missing at random. The whole dataset is again treated with center and scaling of each attribute. Center scaling for all attributes in the dataset helps us to eliminate the skewness of the dataset. Before executing scaling we have removed zero variance as well as near zero variance predictors. These predictors have unique values and have less frequency when compared to other predictors. High correlated values which affect the modelling also are removed from the data set with a cutoff of 0.75 using the function called findCorrelation

**4.6 SELECTING RELEVANT ATTRIBUTES**

Feature selection algorithm Boruta which selects relevant attributes according to the z score. The z score is calculated from the shadow attributes which are repeated iterations of columns from the dataset. Due to these repetitive iterations from the dataset the Z score statistic which acts as threshold will divide as important attributes which are above the threshold and below the threshold will be considered as unimportant attributes. From the 970 attributes Boruta algorithm was executed with 2 iterations. The reduced set from 970 to 325 attributes using Boruta. Principal Component Analysis abbreviated as PCA used to transform from higher dimension to lower dimension which essentially consists of components. We selected the top 100 components which cover over 85 percent of variance and go for modelling with logistic regression, support vector machine and decision tree.

**4.7 MODEL PROCESSING AND EVALUATION**

In this stage, we have used logistic regression model which is a generalized linear model works based on odds ratio. This regression technique shows significant attributes according to their significant levels. The top 100 components which are divided into train and test set wherein the components are in terms of rows. These 100 components with 25000 observations are trained using logistic regression and on other hand the remaining observations come under test dataset. The accuracy in logistic regression was 65 percent. The same dataset used for decision tree with accuracy of 88 percent and finally the svm accuracy was 92 percent. Lastly, after getting the predicted probabilities the next step is to make ensemble of all these three models using weighted average method. The weighted average method ensemble has given significant accuracy about 94 percent.

**4.8** **VALIDATING THE MODEL**

Validating the model by definition should perform hold out samples by resampling methods to check how significant these models classify the dataset. We have used repeated cross validation with repeats equal to 5. The method is performed on all the three models and the result was accuracy more or less equal to one which is not performed with cross validation.

CHAPTER 5

# IMPLEMENTATION

## 5.1. DATA IMPUTATION

Imputation is the process of filling NA values with different methods such as mean, median etc. In this Bosch project, the NA values are being replaced with “0”. This is because the stations which manufacture product would not necessarily start from first station itself. It solely depends upon the product requirements accordingly features are been manufactured. Therefore, the missing values are missing not at random.

newdata2[is.na(newdata2)]<-0

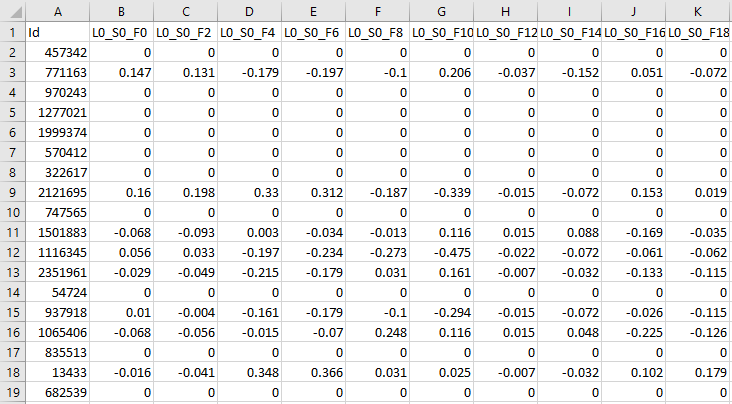


Fig. 5.1: Imputation with Zero

We haven’t considered any outliers because classifying outliers depends upon the application domain of Bosch production lines.

## 5.2. BALANCING THE IMBALANCE CLASS

The class variable in the Bosch dataset is and binary variable class which depicts failure with 1 and non-failures with 0. The number of failures is 0.6 percent when compared to non-failure class of 99.4 percent. In order to balance the class using oversampling and under sampling technique we have used R package called ROSE which does both types of sampling using smoothed bootstrap approach.

library(ROSE)

data.bal2<-ovun.sample(Response~.,data=Data2,method="both",p=0.5)$data

After use this library ROSE the data set is has resulted in balanced class with equal number of Zeros and Ones.

**5.3 REMOVING ZERO VARAINCE PREDICTORS AND HIGH CORRELATED ATTRIBUTES**

The zero variance predictors are those independent variables which are have more unique as well as the variance is less when compared to other attributes. Highly correlated attributes are being removed with a cutoff of 0.75 using the function findcorrelation.

y = nearZeroVar(train\_numeric, saveMetrics = TRUE)

write.csv(y,file="nearzero\_970.csv")

descrcorr2<-cor(newdata2)

write.csv(descrcorr2,file="descrcorr2.csv")

highCorr <- findCorrelation(descrcorr2, 0.75)

length(highCorr)

newdata2<-newdata2[,-highCorr]

## 5.4. CENTER AND SCALING

Due to the presence of skewness in each attributes of the dataset. The attributes are removed with skewness using method called center and scaling which is a variable transformation method. It takes the min and max from each attribute from the dataset and been scaled.

maxs <- apply(newdata2, 2, max)

mins <- apply(newdata2, 2, min)

scaled.newdata2 <- as.data.frame(scale(newdata2, center = mins, scale = maxs - mins))

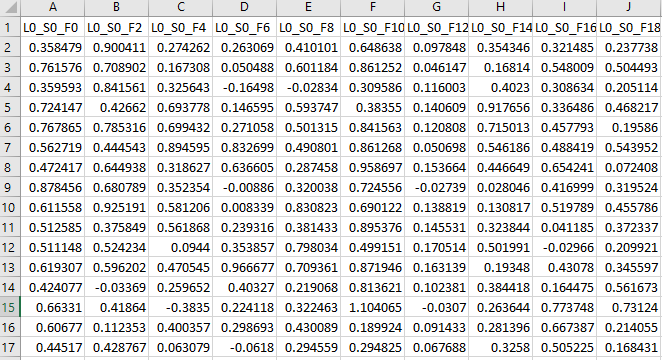


Fig 5.4: Center and scaled

## 5.5. BORUTA FEATURE SELECTION ALGORITHM

Boruta is a wrapper around Random Forest classification algorithm iteratively removes less relevant attributes using shadow attributes and uses Z score instead of accuracy and standard deviation Z score takes into account of fluctuations of mean accuracy among the trees. The final Z score after repeated iterations acts as a threshold to segregate into important and unimportant attributes.

Boruta.n<-Boruta(Response~.,data=data.rose2,doTrace=2,ntree=100)

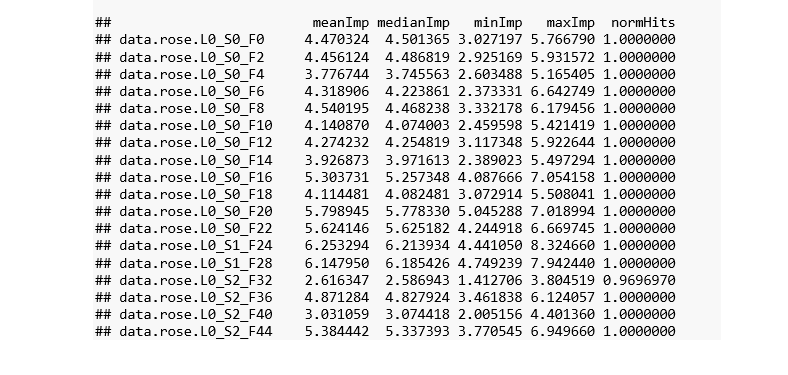


Fig 5.5: Boruta output

1. It adds randomness to the given data set by creating shuffled copies of all features
2. Then, it trains a random forest classifier to evaluate the importance of each feature where higher means more important.
3. At every iteration, it checks whether a real feature has a higher importance than the best of its shadow features, whether the feature has a higher Z score than the maximum Z score of its shadow features and constantly removes features which are highly unimportant.
4. Finally, the algorithm stops either when all features gets confirmed or rejected or it reaches a specified limit of random forest runs

## 5.6. PRINCIPAL COMPONENT ANALYSIS

* Principal component analysis is a method of extracting important variables (in form of components) from a large set of variables available in a data set. It extracts low dimensional set of features from a high dimensional data set with a motive to capture as much information as possible.

Data.bal.pc<-prcomp(Data.bal2)

summary(Data.bal.pc)

Data3<- predict(Data.bal.pc,newdata=Data.bal2)

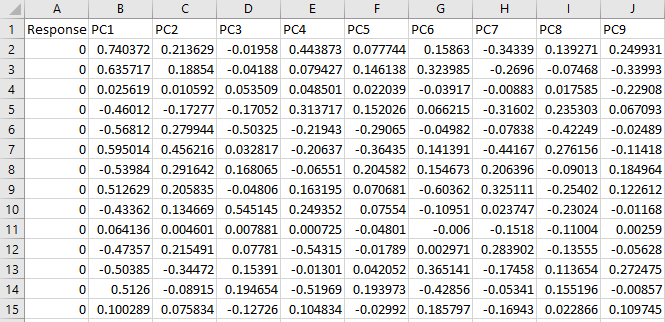


Fig 5.6: Principal Components

## 5.7. LOGISTIC REGRESSION

Logistic regression comes under generalized linear models which are used to predict the categorical response variable. This technique comes under glm() function which used to fit generalized linear models and comes under binomial family with link function. The odds ratio is the link function used in logistic regression. The odds ratio gives the ration between the number of favorable outcomes to number of non-favorable outcomes. Depending upon the odds ratio and significant attributes it predicts the binary outcome from a set of predictor variables. Logistic regression is used to describe data and which tells the relationship between the predictor variable and the class variable. Before going to this kind of model, it has few assumptions to be satisfied:

1. The response variable should be dichotomous
2. There should be no outliers in data
3. There should be no high intercorrelations among the predictor variables.

model\_lr3<-train(Response~.,data=train.final3,family="binomial",method="glm")

Logistic regression equation:

log(П/1-П) =α0+Ʃpj=1 αjXj

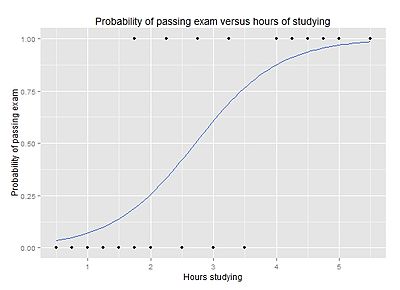


Fig 5.7: Logistic Regression graph

## 5.8. SUPPORT VECTOR MACHINE

Support Vector Machine is a machine learning algorithm which used for classification i.e predicting the class variable using hyperplane. The hyperplane is used to differentiate the two classes. In SVM,it is easy to form linear hyperplane but there is another feature called kernel trick. This kind of feature takes low dimensional feature and transform into higher dimensional feature space. Kernel tricks are widely used for extremely complex problems as well as when the data is nonlinear separable problem.

Cost and gamma are tuning parameters in SVM.The cost tuning parameter tells how complex the hyper plane should. By default, the tuning parameter will be equal to 1 and ranges from 0 to 1. The kernel trick used for this project is “svmRadial” and cost value is given as 0.05.

svm.model<-svm(Response~.,data=train.final3,cost=0.05,method="svmRadial")



Fig 5.8: Support Vector Machine

Equation of hyperplane can be written as:

w.x-b=0

w is normal vector to the hyperplane and x is the nearest point from either of the group

## 5.9. DECISION TREE MODELLING

Decision Tree is the learning of trees from tree like structure, where each internal node denotes the test on an attribute, each branch represents the test outcome and each leaf node holds a class label. The topmost node is the tree node. The construction of decision tree doesn’t require any domain knowledge or parameter setting. These trees can handle high dimensional data. They are the representation of acquired knowledge in tree form is easily understood by humans. The learning and classification steps of decision tree induction are simple and fast. During the tree construction, the attribute selection measures are used to select the attributes that best partitions the tuples into distinct classes.

Tree pruning attempts to identify and remove such branches, with goal of improving classification accuracy on unseen data.

model\_rpart<-rpart(Response~.,data=train.final3,method="class",cp=0.003)

Credit

Student

Fig 5.9: Decision Tree

## 5.10. MODELLING AND DIVIDE TRAIN, TEST

A sample of 50000 observations are sampled from raw principal component dataset, and divided the 50000 observation into 50 percent train and 50 percent test dataset.

Data2 <- scaled.newdata2[sample(nrow(scaled.newdata2),50000),]

Data2$Id<-NULL

train <- sample(nrow(Data3), 0.5\*nrow(Data3))

train.final2<- Data3[train,]

test.final2<- Data3[-train,]

After dividing the train and test dataset the train data set is trained using logistic regression and predicted on test dataset. 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. Rather than choosing parameters that minimize the sum of squared errors (like in ordinary regression), estimation in logistic regression chooses parameters that maximize the likelihood of observing the sample values.

model\_lr3<-train(Response~.,data=train.final3,family="binomial",method="glm") ##logistic regression modelling

summary(model\_lr3)

pred\_lr3<-predict(model\_lr3,newdata=test.final3,type="prob") ##prediction on the test dataset

summary(pred\_lr3) ##summarising the probabilities

pred\_new\_lr3<-ifelse(pred\_lr3[,2]>0.498492,"pass","fail") ##converting into categorical from the predicted probabilities

pred\_new\_lr3<-as.factor(pred\_new\_lr3)

confusionMatrix(test.final3$Response,pred\_new\_lr3) ##The Confusion Matrix

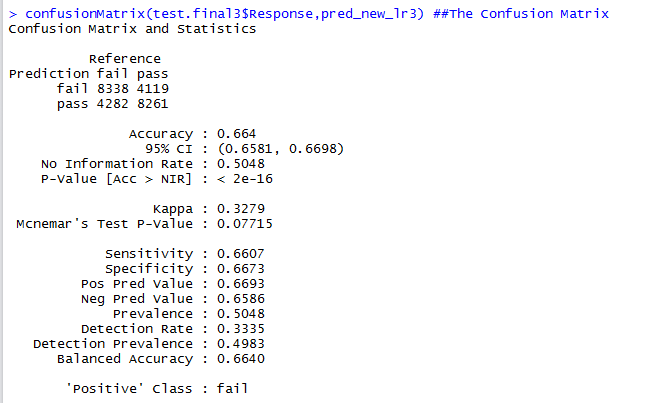


Fig 5.10.1: Confusion Matrix of logistic regression

##SVM modelling

svm.model<-svm(Response~.,data=train.final3,probability=TRUE) ##SVM modelling

pred\_svm3<-predict(svm.model,newdata=test.final3,type="response",probability=TRUE) ##Prediction on test dataset

prob3<-attr(pred\_svm3, "probabilities")

summary(prob3[,1]) ##summarizing the probabilities

prob3\_svm<-ifelse(prob3[,1]>0.9992000,"pass","fail") ##converting into categorical from the predicted probabilities

confusionMatrix(test.final3$Response,prob3\_svm) ##The Confusion Matrix

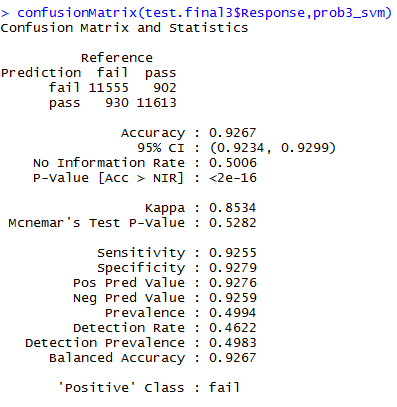


Fig 5.10.2: Confusion Matrix of SVM

Third Model, a decision tree can be used to visually and explicitly represent decisions and decision making. A decision tree is a simple representation for classifying examples. A tree can be "learned" by splitting the source [set](https://en.wikipedia.org/wiki/Set_(mathematics)) into subsets based on an attribute value test. This process is repeated on each derived subset in a recursive manner called recursive partitioning.

model\_rpart<-rpart(Response~.,data=train.final3,method="class") ##decision tree modelling

pred\_rpart<-predict(model\_rpart,newdata=test.final3,type="prob") ##prediction on the test dataset

summary(pred\_rpart[,2]) ##summarising the probabilities

plot(pred\_rpart[,2]) ##probability plot

pred\_rpart\_new<-ifelse(pred\_rpart[,2]>0.4979,"pass","fail") ##converting into categorical from the predicted probabilities

pred\_rpart\_new<-as.factor(pred\_rpart\_new)

confusionMatrix(pred\_rpart\_new,test.final3$Response) ##The Confusion Matrix

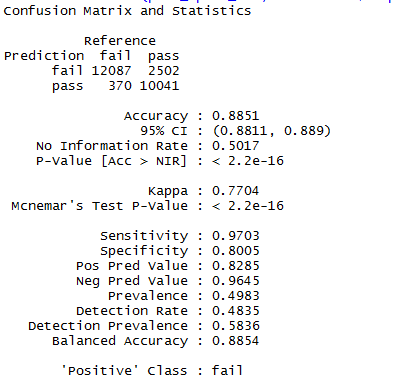


Fig 5.10.3: Confusion Matrix of Decision Tree

Ensemble the three models predicted probabilities using weighted average method to improve and understand the combined result of all three methods and got the confusion matrix at a required threshold

##ensembling using weighted average

ensemble\_prob<-data.frame(pred\_rpart[,2],pred\_lr3[,2],prob3[,1]) ##combining predicted probabilities from all three models

class(ensemble\_prob)

ensemble\_avg<-(ensemble\_prob$pred\_rpart...2.\*0.4+ensemble\_prob$pred\_lr3...2.\*0.2+ensemble\_prob$prob3...1.\*0.4) ##taking weighted average

summary(ensemble\_avg) ##summarising the probabilities

ensemble\_avg\_class<-ifelse(ensemble\_avg>0.54630 ,"pass","fail") ##converting into categorical from the predicted probabilities

confusionMatrix(ensemble\_avg\_class,test.final3$Response) ##The Confusion Matrix

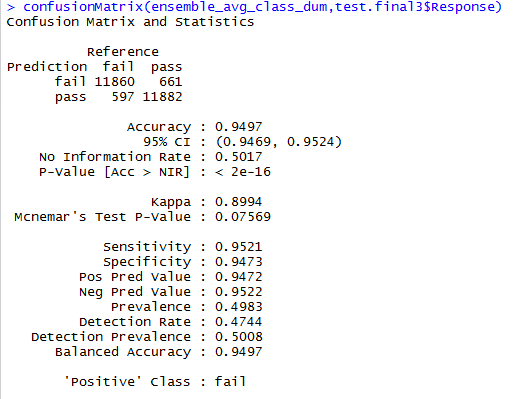


Fig 5.10.4: Confusion Matrix of Weighted Average Ensemble

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Model** | **No of samples** | **Accuracy** | **Sensitivity** | **Specificity** | **Positive**  **Predicted**  **Value** | **Negative Predicted Value** |
| SVM | 25000 | 92.67% | 92.55% | 92.79% | 92.76% | 92.59% |
| Decision Tree | 25000 | 88.51% | 97.03% | 80.05% | 82.85% | 96.45% |
| Logistic Regression | 25000 | 66.4% | 66.07% | 66.73% | 66.93% | 65.86% |
| Ensemble | 25000 | 94.97% | 95.21% | 94.73% | 94.72% | 95.22% |

Fig 5.10.5: Model metrics for all models

## 5.11. R CODE FOR SAMPLE PREDICTIONS FOR ALL MODELS

sample1<-test.final3[sample(nrow(test.final3),5000),]

## Decision Tree prediction

pred\_rpart1<-predict(model\_rpart,newdata=sample1,type="prob")

summary(pred\_rpart1[,2])

pred\_rpart\_new1<-ifelse(pred\_rpart1[,2]> 0.2097,"pass","fail") ##converting into categorical from the predicted probabilities

pred\_rpart\_new1<-as.factor(pred\_rpart\_new1)

confusionMatrix(pred\_rpart\_new1,sample1$Response) ##The Confusion Matrix library(caret)

##Logistic Regression Prediction

pred\_lr1<-predict(model\_lr3,newdata=sample1,type="prob") ##prediction on the test dataset

summary(pred\_lr1) ##summarising the probabilities

pred\_new\_lr1<-ifelse(pred\_lr1[,2]>0.495567,"pass","fail") ##converting into categorical from the predicted probabilities

pred\_new\_lr1<-as.factor(pred\_new\_lr1)

confusionMatrix(sample1$Response,pred\_new\_lr1) ##The Confusion Matrix library(caret)

##SVM Prediction

pred\_s1<-predict(svm.model,newdata=sample1,type="response",probability=TRUE)

prob1<-attr(pred\_s1, "probabilities")

summary(prob1[,1]) ##summarising the probabilities

prob1\_s1<-ifelse(prob1[,1]>0.446100,"pass","fail") ##converting into categorical from the predicted probabilities

confusionMatrix(sample1$Response,prob1\_s1)

sample2<-test.final3[sample(nrow(test.final3),5000),]

##Logistic Regression Prediction

pred\_lr2<-predict(model\_lr3,newdata=sample2,type="prob") ##prediction on the test dataset

summary(pred\_lr2) ##summarising the probabilities

pred\_new\_lr2<-ifelse(pred\_lr2[,2]>0.496452,"pass","fail") ##converting into categorical from the predicted probabilities

pred\_new\_lr2<-as.factor(pred\_new\_lr2)

confusionMatrix(sample2$Response,pred\_new\_lr2) ##The Confusion Matrix library(caret)

##SVM Prediction

pred\_s2<-predict(svm.model,newdata=sample2,type="response",probability=TRUE)

prob2<-attr(pred\_s2, "probabilities")

summary(prob2[,1]) ##summarising the probabilities

prob2\_s2<-ifelse(prob2[,1]>0.413100,"pass","fail") ##converting into categorical from the predicted probabilities

confusionMatrix(sample2$Response,prob2\_s2)

##Decision tree prediction

pred\_rpart2<-predict(model\_rpart,newdata=sample2,type="prob")

summary(pred\_rpart2[,2])

pred\_rpart\_new2<-ifelse(pred\_rpart2[,2]> 0.2097,"pass","fail") ##converting into categorical from the predicted probabilities

pred\_rpart\_new2<-as.factor(pred\_rpart\_new2)

confusionMatrix(pred\_rpart\_new2,sample2$Response)

sample3<-test.final3[sample(nrow(test.final3),5000),]

#Logistic Regression Prediction

pred\_lr3\_1<-predict(model\_lr3,newdata=sample3,type="prob") ##prediction on the test dataset

summary(pred\_lr3\_1) ##summarising the probabilities

pred\_new\_lr3\_1<-ifelse(pred\_lr3\_1[,2]>0.486548,"pass","fail") ##converting into categorical from the predicted probabilities

pred\_new\_lr3\_1<-as.factor(pred\_new\_lr3\_1)

confusionMatrix(sample3$Response,pred\_new\_lr3\_1) ##The Confusion Matrix library(caret)

##SVM prediction

pred\_s3<-predict(svm.model,newdata=sample3,type="response",probability=TRUE)

prob3\_s3<-attr(pred\_s3, "probabilities")

summary(prob3\_s3[,1]) ##summarising the probabilities

prob3\_s3\_new<-ifelse(prob3\_s3[,1]>0.4461000,"pass","fail") ##converting into categorical from the predicted probabilities

confusionMatrix(sample3$Response,prob3\_s3\_new)

length(sample3$Response)

##Decision Tree prediction

pred\_rpart3<-predict(model\_rpart,newdata=sample3,type="prob")

summary(pred\_rpart3[,2])

pred\_rpart\_new3<-ifelse(pred\_rpart3[,2]> 0.2097,"pass","fail") ##converting into categorical from the predicted probabilities

pred\_rpart\_new3<-as.factor(pred\_rpart\_new3)

confusionMatrix(pred\_rpart\_new3,sample3$Response) ##The Confusion Matrix library(caret)

sample4<-test.final3[sample(nrow(test.final3),10000),]

##Logistic Regression Prediction

pred\_lr4<-predict(model\_lr3,newdata=sample4,type="prob") ##prediction on the test dataset

summary(pred\_lr4) ##summarising the probabilities

pred\_new\_lr4<-ifelse(pred\_lr4[,2]>0.496452,"pass","fail") ##converting into categorical from the predicted probabilities

pred\_new\_lr4<-as.factor(pred\_new\_lr4)

confusionMatrix(sample4$Response,pred\_new\_lr4) ##The Confusion Matrix library(caret)

##Decision Tree Prediction

pred\_rpart4<-predict(model\_rpart,newdata=sample4,type="prob")

summary(pred\_rpart4[,2])

pred\_rpart\_new4<-ifelse(pred\_rpart4[,2]> 0.2097,"pass","fail") ##converting into categorical from the predicted probabilities

pred\_rpart\_new4<-as.factor(pred\_rpart\_new4)

confusionMatrix(pred\_rpart\_new4,sample4$Response) ##The Confusion Matrix library(caret)

##SVM Prediction

pred\_s4<-predict(svm.model,newdata=sample4,type="response")

prob4<-attr(pred\_s4, "probabilities")

summary(prob4[,1]) ##summarising the probabilities

prob4\_s4<-ifelse(prob4[,1]>0.4155000,"pass","fail") ##converting into categorical from the predicted probabilities

confusionMatrix(sample4$Response,prob4\_s4)

## 5.12. FINAL R CODE OF PREDICTIVE MODELLING

setwd("D://Lokin//Data Analytics") ##setting directory

library(data.table) ##Loading required libraries

library(caret)

library(ROSE)

library(e1071)

library(rpart)

train\_numeric<-fread("train\_numeric.csv") ##importing the train dataset csv file

y = nearZeroVar(train\_numeric, saveMetrics = TRUE) ##Finding near zero variance vectors

write.csv(y,file="nearzero\_970.csv") ##writing list of vectors into seperate file

##Removing subset of near-zero variance vectors from the original dataset

newdata2<-subset(train\_numeric,select=-c(L0\_S15\_F409 ,

L1\_S24\_F751 ,

L1\_S24\_F761 ,

L1\_S24\_F1486,

L1\_S25\_F2187,

L1\_S25\_F2190,

L1\_S25\_F2761,

L1\_S25\_F2764,

L3\_S40\_F3990,

L3\_S47\_F4173,

L0\_S2\_F52 ,

L0\_S3\_F88 ,

L0\_S8\_F146 ,

L0\_S8\_F149 ,

L0\_S10\_F269 ,

L0\_S14\_F378 ,

L0\_S18\_F435 ,

L0\_S20\_F463 ,

L0\_S20\_F466 ,

L1\_S24\_F1371,

L1\_S24\_F1840,

L1\_S25\_F1865,

L1\_S25\_F1873,

L1\_S25\_F2478,

L1\_S25\_F2481,

L1\_S25\_F2857,

L2\_S26\_F3055,

L2\_S26\_F3077,

L2\_S26\_F3125,

L2\_S27\_F3148,

L2\_S28\_F3241,

L2\_S28\_F3263,

L2\_S28\_F3311,

L3\_S29\_F3485,

L3\_S29\_F3488,

L3\_S29\_F3491,

L3\_S30\_F3594,

L3\_S30\_F3599,

L3\_S30\_F3614,

L3\_S30\_F3619,

L3\_S30\_F3654,

L3\_S30\_F3659,

L3\_S30\_F3694,

L3\_S30\_F3699,

L3\_S30\_F3714,

L3\_S30\_F3719,

L3\_S30\_F3724,

L3\_S30\_F3729,

L3\_S30\_F3734,

L3\_S30\_F3739,

L3\_S30\_F3779,

L3\_S30\_F3789,

L3\_S30\_F3814,

L3\_S30\_F3824,

L3\_S31\_F3838,

L3\_S33\_F3867,

L3\_S33\_F3869,

L3\_S33\_F3871,

L3\_S33\_F3873,

L3\_S34\_F3876,

L3\_S34\_F3878,

L3\_S34\_F3880,

L3\_S34\_F3882,

L3\_S35\_F3884,

L3\_S35\_F3898,

L3\_S35\_F3903,

L3\_S35\_F3908,

L3\_S35\_F3913,

L3\_S36\_F3926,

L3\_S36\_F3930,

L3\_S36\_F3934,

L3\_S36\_F3938,

L3\_S37\_F3944,

L3\_S37\_F3946,

L3\_S37\_F3948,

L3\_S37\_F3950,

L3\_S39\_F3964,

L3\_S39\_F3968,

L3\_S39\_F3972,

L3\_S39\_F3976,

L3\_S40\_F3988,

L3\_S40\_F3992,

L3\_S40\_F3994,

L3\_S41\_F3996,

L3\_S41\_F3998,

L3\_S41\_F4002,

L3\_S43\_F4060,

L3\_S43\_F4065,

L3\_S43\_F4070,

L3\_S43\_F4075,

L3\_S43\_F4080,

L3\_S43\_F4090,

L3\_S44\_F4100,

L3\_S44\_F4103,

L3\_S44\_F4106,

L3\_S44\_F4109,

L3\_S45\_F4126,

L3\_S45\_F4128,

L3\_S45\_F4130,

L3\_S45\_F4132,

L3\_S47\_F4178,

L3\_S47\_F4183,

L3\_S47\_F4188,

L3\_S48\_F4200,

L3\_S48\_F4202,

L3\_S48\_F4204,

L3\_S49\_F4206,

L3\_S49\_F4216,

L3\_S49\_F4221,

L3\_S49\_F4226,

L3\_S49\_F4231,

L3\_S50\_F4245,

L3\_S50\_F4247,

L3\_S50\_F4249,

L3\_S50\_F4251,

L3\_S51\_F4256,

L3\_S51\_F4258,

L3\_S51\_F4260,

L3\_S51\_F4262))

newdata2<-as.data.frame(newdata2)

newdata2[is.na(newdata2)]<-0 ## imputing NA values with zeros

descrcorr2<-cor(newdata2)

write.csv(descrcorr2,file="descrcorr2.csv")

highCorr <- findCorrelation(descrcorr2, 0.75) ##finding highly correlated attributes with cutoff 0.75

length(highCorr)

newdata2<-newdata2[,-highCorr] ##Removing highly correlated attributes and reducing from 970 to 601 attributes

maxs <- apply(newdata2, 2, max)

mins <- apply(newdata2, 2, min) ##applying center and scaling method to eliminate skewness

scaled.newdata2 <- as.data.frame(scale(newdata2, center = mins, scale = maxs - mins))

Data2 <- scaled.newdata2[sample(nrow(scaled.newdata2),50000),] ##Sampling 50000 observations from original train dataset

Data2$Id<-NULL ##Making ID column null

Data.bal2<-ovun.sample(Response~.,data=Data2,method="both",p=0.5)$data ##Over and under sampling from

## ROSE library to balance the class

Response<-Data.bal2[[1]] ##seperating the response variable to do PCA

Data.bal2[[1]]<-NULL

Data.bal.pc<-prcomp(Data.bal2) ##using prcomp function to perfom PCA

summary(Data.bal.pc) ##Shows cummulative variance and respective loadings of each component

Data3<- predict(Data.bal.pc,newdata=Data.bal2) ##Transforming Components in terms of 50000 observations

Data3<-data.frame(Response,Data3) ## Attaching the response varaible to the component dataset

dim(Data3) ## to see the dimensionality

train <- sample(nrow(Data3), 0.5\*nrow(Data3)) ##dividing the dataset into 50 percent train & test

train.final2<- Data3[train,]

test.final2<- Data3[-train,]

train.final3 <- train.final2[,1:101] ##selecting the first 100 components which cover over 85 percent varaince

test.final3 <- test.final2[,1:101] ##selecting the first 100 components which cover over 85 percent varaince

train.final3$Response<-ifelse(train.final3$Response==1,"fail","pass") ##changing the binary into names

test.final3$Response<-ifelse(test.final3$Response==1,"fail","pass")

train.final3$Response<-as.factor(train.final3$Response) ##Making variable names into factor levels

test.final3$Response<-as.factor(test.final3$Response)

##SVM modelling

svm.model<-svm(Response~.,data=train.final3,probability=TRUE,cost=0.05) ##library(e1071)

pred\_svm3<-predict(svm.model,newdata=test.final3,type="response",probability=TRUE) ##Prediction on test dataset

prob3<-attr(pred\_svm3, "probabilities")

summary(prob3[,1]) ##summarising the probabilities

prob3\_svm<-ifelse(prob3[,1]>0.9992000,"pass","fail") ##converting into categorical from the predicted probabilities

confusionMatrix(test.final3$Response,prob3\_svm) ##The Confusion Matrix library(caret)

##decision tree modelling

model\_rpart<-rpart(Response~.,data=train.final3,method="class",cp=0.03) ##library(rpart)

pred\_rpart<-predict(model\_rpart,newdata=test.final3,type="prob") ##prediction on the test dataset

summary(pred\_rpart[,2]) ##summarising the probabilities

plot(pred\_rpart[,2]) ##probability plot

pred\_rpart\_new<-ifelse(pred\_rpart[,2]>0.4979,"pass","fail") ##converting into categorical from the predicted probabilities

pred\_rpart\_new<-as.factor(pred\_rpart\_new)

confusionMatrix(pred\_rpart\_new,test.final3$Response) ##The Confusion Matrix library(caret)

##Logistic regression modelling

model\_lr3<-train(Response~.,data=train.final3,family="binomial",method="glm") ##logistic regression modelling

summary(model\_lr3)

pred\_lr3<-predict(model\_lr3,newdata=test.final3,type="prob") ##prediction on the test dataset

summary(pred\_lr3) ##summarising the probabilities

pred\_new\_lr3<-ifelse(pred\_lr3[,2]>0.498492,"pass","fail") ##converting into categorical from the predicted probabilities

pred\_new\_lr3<-as.factor(pred\_new\_lr3)

confusionMatrix(test.final3$Response,pred\_new\_lr3) ##The Confusion Matrix library(caret)

##ensembling using weighted average

ensemble\_prob<-data.frame(pred\_rpart[,2],pred\_lr3[,2],prob3[,1]) ##combining predicted probabilities from all three models

class(ensemble\_prob)

ensemble\_avg\_dum<-(ensemble\_prob$pred\_rpart...2.\*0.5+ensemble\_prob$pred\_lr3...2.\*0.1+ensemble\_prob$prob3...1.\*0.4) ##taking weighted average

summary(ensemble\_avg\_dum) ##summarising the probabilities

plot(ensemble\_avg\_dum)

ensemble\_avg\_class\_dum<-ifelse(ensemble\_avg\_dum>0.39300,"pass","fail") ##converting into categorical from the predicted probabilities

confusionMatrix(ensemble\_avg\_class\_dum,test.final3$Response)

CHAPTER 6

# CONCLUSION

From the ensemble learning using weighted average method which used to multiply weights with predictions from each models. We get to know the diversity of models with each has its own pros. The confusion matrix from ensemble learning will show performance of an ensemble. The performance of its processing can be seen through the number of True positives, True Negatives, False negatives and False positives as well as several metrics can be drawn out.

* Condition Positives – 12457 number of real positive cases
* Condition Negatives- 12543 number of real negative cases
* True Positives (TP)- 11860 are actually yes and predicted yes.
* True Negatives (TN)- 11882 are actually no and predicted no.
* False Positives (FP)– 597 are actually no but predicted yes.
* False Negatives (FN)- 661 are actually yes but predicted no.
* Accuracy gives percent of how often the classifier is correct, here ensemble is predicting 94.47 percent of failures were predicted correctly.

ACC= (TP+TN)/TOTAL

* Sensitivity or True Positive Rate was 95.21 percent which defines how often does the failure predicted as failure.

Sensitivity=TP/predicted as failure

* Specificity or True Negative Rate was 94.73 percent which defines how often does it predict as non-failure.

Specificity=TN/predicted as non-failure

* Precision is a good measure which defines how often it is correct when predicted failure. In the Bosch case the precision or positive predicted value was 94.72 percent which is a good sign that the model is working absolutely with correct failure prediction.

PPV=TP/(TP+FP)

CHAPTER 7

# REFERENCES

1. Kenneth H. Doerr, Theodore D. Klastorin, and Michael J. Magazine, “Synchronous Unpaced Flow Lines with Worker Differences and Overtime Cost”, March 2000, pp 421-435.
2. Max Kuhn, “Building Predictive Modelling in R using the caret Package”, American Statistical Association, April 2008.
3. Miron B. Kursa, Witold R. Rudnick, “Feature Selection with the Boruta Package”, American Statistical Association, December 2009.
4. Nicola Lunardon, Giovanna Menardi, and Nicola Torelli,” ROSE: A package for Binary Imbalanced Learning”, The R Journal Vol. 6/1, June 2014