Marketing Campaign for Term Deposit

Niranjan Adhikari

Machine Learning course project

1. Recap of data set and Variables used

Number of Observations : 4521 Number of attributes : 17

Input variables:

Bank client data

1.age (numeric)

2.job: type of job (categorical: 'admin.', 'blue collar', 'entrepreneur', 'housemaid', 'management', 'retired', 'self-employed', 'services', 'student', 'technician', 'unemployed')

3.marital : marital status (categorical: 'divorced','married','single')

4.education: (categorical:'primary','secondary','tertiary')

5.default: has credit in default? (categorical: 'no','yes'

6.balance (numeric)

7.housing: has housing loan? (categorical: 'no','yes')

8.loan: has personal loan? (categorical: 'no','yes')

Related with the last contact of the current campaign

9.contact: contact communication type (categorical: 'cellular','telephone')

10.day: contact day of the months (numeric)

11.month: last contact month of year (categorical: 'jan', 'feb', 'mar', ., 'nov', 'dec')

12.duration: last contact duration, in seconds (numeric).

Other attributes

13.campaign: number of contacts performed during this campaign and for this client (numeric, includes last contact)

14.pdays: number of days that passed by after the client was last contacted from a previous campaign

15.previous: number of contacts performed before this campaign and for this client (numeric)

16.poutcome: outcome of the previous marketing campaign (categorical: 'failure','other','success')

Additional attributes for Unknown responses

17.Job_unk: unknown job categories

18.edu_unk: unknown education level of clients

19.cont_unk: unknown means of contact

20.pout_unk: unknow outcome of the previous marketing campaign contact

Output variable (desired target):

21.y : has the client subscribed a term deposit? (binary: 'yes','no')

2. Objective

The final goal of this project is to fit possible set of models to predict whether or not the marketing campaign is successful in acquisition of customers into the bank's term deposit. We analyzed the performancces of three different machine learning algorithms by training and testing data sets and selected the best according to the degree of accuracy. This would suggest if the marketing campaign team of the Portugeuse bank should continue investing into tele-marketing their term deposit scheme.

The objective of this phase of project will include

1. Methodlogy for building algorithms
2. Details of the algorithms and fine-tuning over the data set
3. Performance comparison and choosing the best model
4. Limitations of the algorithms
5. Summary and Conclusion

3. Methodology

3.1. Building Machine Learning Algorithms

As the data set is cleaned and preprocessed in the first phase, three machine learning framework shortlisted for this problem are discussed below. Reading and preprocessing the data for each of the three algorithms are performed separately. There are further modification and manipulation of data tailored to need of specific algorithms. Prediction check, misclassification errors and cross table performed for each model.

3.1.1 Standard Logistic Regression Using Lasso Regularisation

One of the suitable approach for this classification problem is the logistic regression algorithm as outcome variables in the data set contains binary responses. Selecting the significant variables for the model is primary aspect of regression approach. Exploring the possibility of improving the model is another important aspect of model building process. So, Lasso (least absolute shrinkage and selection operator) is applied to perform the variable selection and regularization to improve the prediction accuracy and interpretability of the model. It is mandatory to normalize the data set because of different range of data values. After appropriately normalisation the variables using the min/max normalization method, the data set is split into training group and testing group in the ratio of 80 to 20 percent. A standard logistic regression is built on training set and test data set is used to get the confusion matrix. Detailed informations are derived from the cross tabulation.

3.1.2 *K*-Nearest Neighbor using Bias Variance Trade-off

The k-nearest neighbours algorithms(k-NN) is another sensible machine learning approach for this classification problem because K-NN algorithm is the simplest non-parametric method that we can effectively implemented for classification problem. The random sample of the whole data set has been choosen to perform the algorithm. In this case the data is split into three groups, training, validation and testing sets. Training and validation data for different values of k is used to run the nearest neighbours algorithm. The value of k for final K-NN classifier has been picked out from the bias variance trade off plot between training and validation sets. The algorithm is then applied to the test set to get the confusion matrix and misclassification error rate.

3.1.3 Naïve Bayes Classifier

Naïve Bayes classifier is a probability-based classifier for a classification machine learning problem. It is based on Bayes theorem with an assumption of independence between varibles. The response variable to be predicted here is classed as "yes" and "No" and fundamental nature of predictors appear to be relatively independent of each other. Moreover, because our training set is relatively small, possibility of having noisy and unknown data is there, so this approach stands to be suitable. Another advantage of Naïve Bayes Classifier is that, probability for a prediction can be easily calculated. Also a diagonostic analysis of the model is performed before any conclusion on the model is made.

3.2. Performance Analysis of Algorithms

Primary means of analysing the algorithms are using the misclassification error rate of prediction on the test data set. The model with least misclassification error represents a model with better accuracy. Definitely, the model with small misclassification error is chosen as the final model from the three algorithms. Overall adeqauacy of the algorithms were also performed.

4. Model building and predictions

4.1. Logistic Regression Model

4.1.1 Loading the data and Preprocessing

It needs to be taken care to ensure that the response/dependent variable are dichotomous (or nominal) in order to apply logistic regression algorithm. As already discussed the response variable in this problem is to predict either "yes" or "no" for a term deposit. The character variables are converted into numeric and also to include the unknown responses of the attributes in the model, four new variables are created. The following chunks of code load the required library to perfrom the logistic regression.

```
rm(list=ls(all=TRUE))
library(ggplot2)     # We'll need to use ggplot to create some graphs.
library(stringr)     # This is used for string manipulations.
library(glmnet)      # This is where ridge and LASSO reside
library(doParallel) # Install parallel processing for R.  This allows multiple processor codes to be us
library(class)
set.seed(45)         # Since we're going to split our data we need to ensure the split is repeatable.
```

The following code runs the data into R and do the necessary preprocessing. Short description of each code is written along with the code

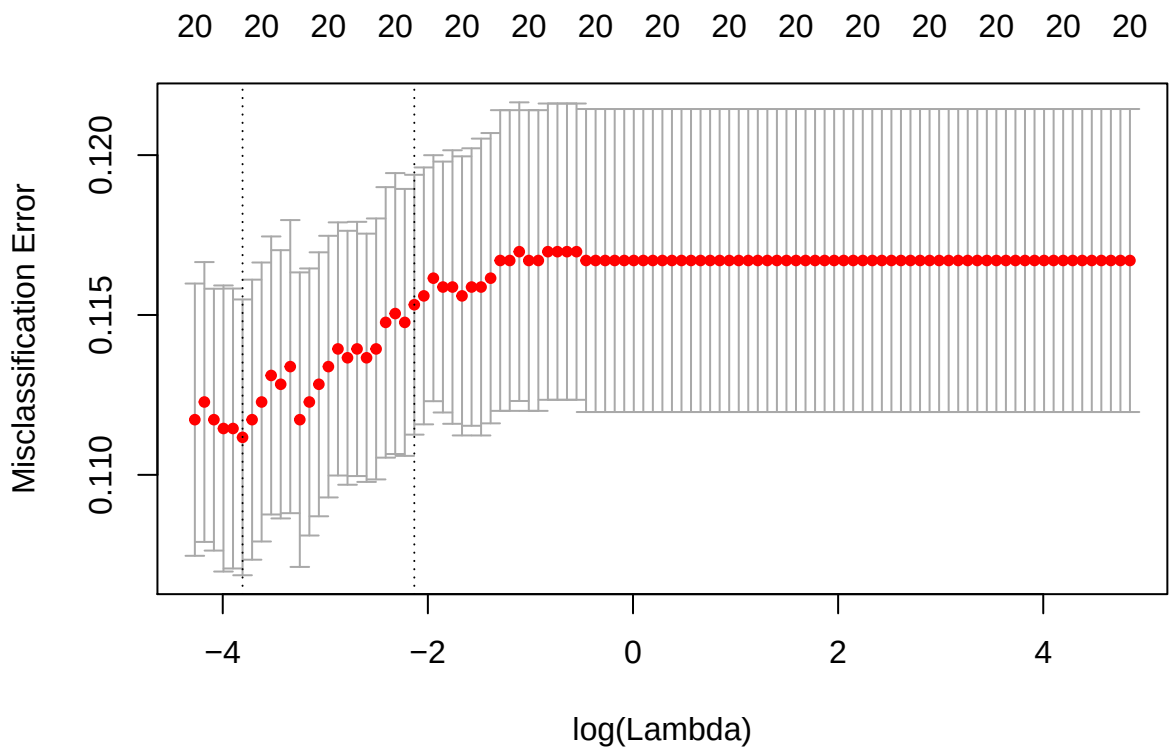
```
#Import data to R
bank<-read.csv("bank1.csv", stringsAsFactors = FALSE, header = T)
#View(bank)

# This code of chunks create extra column for variables with unknown values
bank$job_unk <- ifelse(bank$job == "unknown", 1, 0)
bank$edu_unk <- ifelse(bank$education == "unknown", 1, 0)
bank$cont_unk <- ifelse(bank$contact == "unknown", 1, 0)
bank$pout_unk <- ifelse(bank$poutcome == "unknown", 1, 0)

# This code of chunk make the character data into numeic format
bank$job <- as.numeric(as.factor(bank$job))
bank$marital <- as.numeric(as.factor(bank$marital))
bank$education <- as.numeric(as.factor(bank$education))
bank$default<- ifelse(bank$default == "yes", 1, 0)
bank$housing <- ifelse(bank$housing== "yes", 1, 0)
bank$loan<- ifelse(bank$loan== "yes", 1, 0)
bank$month <- as.numeric(as.factor(bank$month))
bank$contact <- as.numeric(as.factor(bank$contact))
bank$poutcome <- as.numeric(as.factor(bank$poutcome))
bank$y <- ifelse(bank$y== "yes", 1, 0)


# create normalization function
normalize <- function(x) {
  return ((x - min(x)) / (max(x) - min(x)))
}
# normalize the data to get rid of outliers if present in the data set
bank <- as.data.frame(lapply(bank, normalize))

# Creating design matrix and target vector
mydata.X <- model.matrix(y ~ -1+., data= bank)
mydata.X <- as.data.frame(mydata.X)
mydata.Y <- bank$y

#Now we split the data into training and test.
cuts <- c(training = .8, test = .2)
g <- sample(cut(seq(nrow(mydata.X)), nrow(mydata.X)*cumsum(c(0,cuts)), labels = names(cuts)))
final.X <- split(mydata.X, g)
final.Y <- split(mydata.Y, g)
```

First of all, we build the ridge regression. This helps to interpret the data and suggest for further necessity of regularization.

```
bank.ridge <- cv.glmnet(x= as.matrix(final.X$training), y = as.matrix(final.Y$training), nfolds=10,
                        type.measure="class", family='binomial', alpha = 0, nlambda=100)
print(bank.ridge$lambda.min)
```

```
## [1] 0.0222165
```

```
plot(bank.ridge)
```



When lambda = 0, the penalty term has no effect, and ridge regression will produce the least squares estimates. Since, our minimumm value of the lambda is tends to zero. Thus, from the above plot it is conformed that we do not need to do the regularization.

The following code manipulates the coeffiecient of the ridge regression

```
# Create a dataframe with the coefficient values
ridge.coefs <- as.data.frame(as.vector(coef(bank.ridge, s = bank.ridge$lambda.min)),
                             row.names = rownames(coef(bank.ridge)))
names(ridge.coefs) <- 'coefficient'
```

Now we perform regression using LASSO setting alpha to be 1.

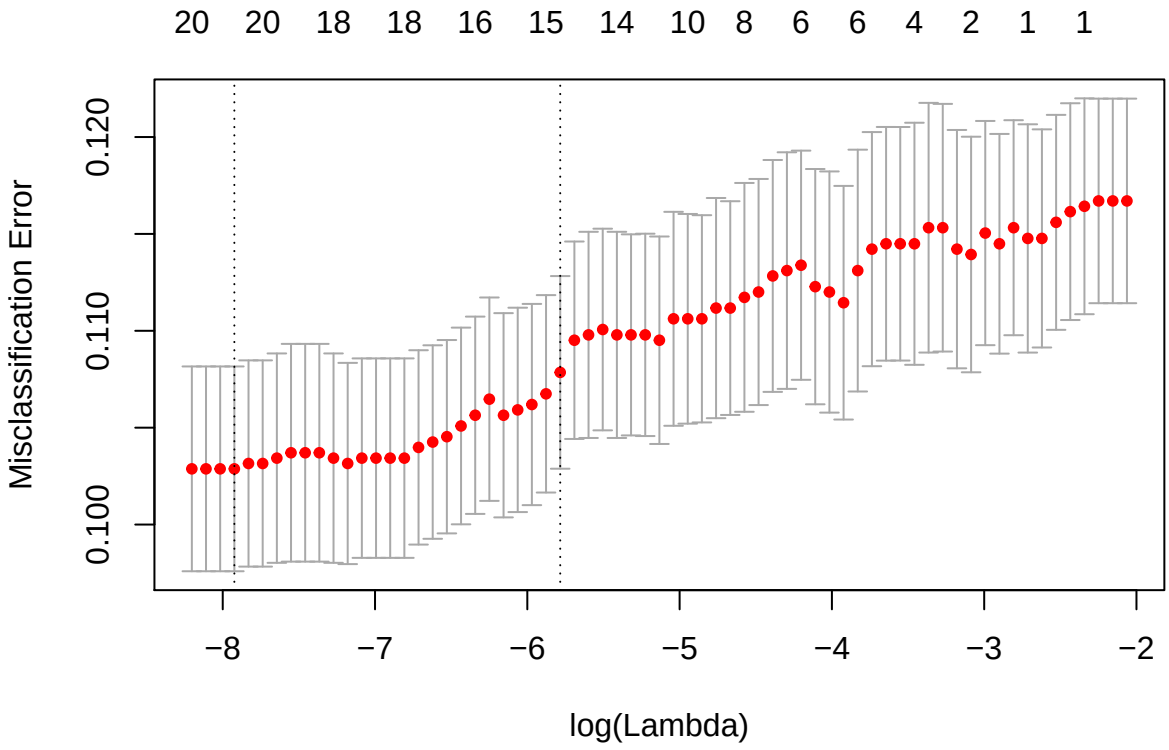
```
bank.lasso <- cv.glmnet(x = as.matrix(final.X$training), y = as.matrix(final.Y$training), nfolds=10,
                        type.measure="class", parallel=TRUE, family='binomial', alpha = 1, nlambda=100)
```

```
## Warning: executing %dopar% sequentially: no parallel backend registered
```

```
print(bank.lasso$lambda.min)
```

```
## [1] 0.0003620736
```

```
plot(bank.lasso)
```



Clearly the lasso leads to qualitatively similar behavior to ridge regression. Since best vallue of value of lambda is still close to zero. thus from the above plot of misclassification error versus lambda, it is conformed that we do not need to do the regularization.

The following code manipulates the coeffiecient of the lasso regression

```
# Create a dataframe with the coefficient values
lasso.coefs <- as.data.frame(as.vector(coef(bank.lasso, s = bank.lasso$lambda.min)),
                             row.names = rownames(coef(bank.lasso)))
print(lasso.coefs)
```

```
##             as.vector(coef(bank.lasso, s = bank.lasso$lambda.min))
## (Intercept)                                          -3.08988161
## age                                                   0.06413239
## job                                                   0.06696785
## marital                                              -0.23404764
## education                                             0.68617942
## default                                               0.60243061
## balance                                               0.06970906
## housing                                              -0.64725541
## loan                                                 -0.87518791
## contact                                              -0.38216155
## day                                                   0.14976069
## month                                                 0.27393274
## duration                                             12.14782553
## campaign                                             -4.48726794
## pdays                                                 0.04721665
## previous                                              0.45412373
```

```
## poutcome                                                 2.89513370
## job_unk                                                  0.11127912
## edu_unk                                                 -0.87211573
## cont_unk                                                -0.87235056
## pout_unk                                                -3.02499118
```

```
names(lasso.coefs) <- 'coefficient'
```

Get the features with the non zero lasso coefficient.

```
features <- rownames(lasso.coefs)[lasso.coefs != 0]
print(features)
```

```
##  [1] "(Intercept)" "age"         "job"         "marital"     "education"
##  [6] "default"     "balance"     "housing"     "loan"        "contact"
## [11] "day"         "month"       "duration"    "campaign"    "pdays"
## [16] "previous"    "poutcome"    "job_unk"     "edu_unk"     "cont_unk"
## [21] "pout_unk"
```

From the above output it is observed that all the above predictors would be important to perform the logistic regression. The following code manipulates the data into the new matrix with the nonzero features and re split the data into training and test sets.

```
# Creates a new matrix with only the non-zero features
lasso_bank <- bank[, intersect(colnames(bank), features)]
# Re-do the split into training and test
bank <- as.matrix(lasso_bank)
bank <- as.data.frame(bank)
bank$Y <- mydata.Y
bank_1 <- split(bank, g)
```

Now standard logistic regression is run using non zero features identified by a LASSO.

```
model_std <- glm(Y ~ ., family = binomial(link = "logit"),  data = bank_1$training)
summary(model_std)
```

```
##
## Call:
## glm(formula = Y ~ ., family = binomial(link = "logit"), data = bank_1$training)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -4.2816  -0.4221  -0.2892  -0.1717   3.1019
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -3.20496    0.47973  -6.681 2.38e-11 ***
## age          0.08953    0.42775   0.209  0.83422
## job          0.07655    0.22045   0.347  0.72841
## marital     -0.25272    0.22111  -1.143  0.25307
## education    0.73348    0.29452   2.490  0.01276 *
## default      0.64322    0.46171   1.393  0.16358
## balance      0.15287    1.49130   0.103  0.91835
## housing     -0.65386    0.13574  -4.817 1.46e-06 ***
## loan        -0.89749    0.21791  -4.119 3.81e-05 ***
## contact     -0.38870    0.52138  -0.746  0.45596
## day          0.17584    0.23272   0.756  0.44989
## month        0.29496    0.22189   1.329  0.18375
```

```
## duration    12.25276    0.65712  18.646  < 2e-16 ***
## campaign    -4.68997    1.54804  -3.030  0.00245 **
## pdays        0.15434    0.85661   0.180  0.85701
## previous     0.45669    1.10047   0.415  0.67815
## poutcome     3.05204    0.42743   7.140 9.30e-13 ***
## job_unk      0.14966    0.58138   0.257  0.79685
## edu_unk     -0.94375    0.36859  -2.560  0.01045 *
## cont_unk    -0.88946    0.54048  -1.646  0.09983 .
## pout_unk    -3.11801    0.39038  -7.987 1.38e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2605.7  on 3615  degrees of freedom
## Residual deviance: 1884.9  on 3595  degrees of freedom
## AIC: 1926.9
##
## Number of Fisher Scoring iterations: 6
```

```
names(model_std$coefficients)
```

```
##  [1] "(Intercept)" "age"         "job"         "marital"     "education"
##  [6] "default"     "balance"     "housing"     "loan"        "contact"
## [11] "day"         "month"       "duration"    "campaign"    "pdays"
## [16] "previous"    "poutcome"    "job_unk"     "edu_unk"     "cont_unk"
## [21] "pout_unk"
```

4.1.2 Prediction and misclassification of the model

```
predictions <- predict.glm(model_std, newdata=bank_1$test, type= "response")
predictions[predictions > 0.5] <- 1
predictions[predictions <= 0.5] <- 0
```

```
1 - length(predictions[predictions == bank_1$test$Y]) / length(predictions)
```

```
## [1] 0.09392265
```

Confusion matrix from the test data

```
table(predictions, bank_1$test$Y)
```

```
##
## predictions   0   1
##           0 790  69
##           1  16  30
```

The confusion matrix with a more infomative outputs offered by CrossTable() in gmodels package helps analyse the prediction accuracy of the model. The output table includes proportion in each cell that tells the percentage of table's row, column or overall total counts on the class of the response variable.

From the cross table below it is observed that using seed as 45, 92% of people not subscribing the term deposit in the data set is predicted correctly while 65% of people subscribing term deposit is predicted correctly.

Thus, from the confusion matrix or Cross table we can safely say that the model perfoms well to predict the customer subscribe term deposit with misclassification error of 9%.

```
library(gmodels)
```

```
## Warning: package 'gmodels' was built under R version 3.4.4
```

```
CrossTable(predictions, bank_1$test$Y, prop.chisq = FALSE)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  905
##
##
##              | bank_1$test$Y
##  predictions |         0 |         1 | Row Total |
## -------------|-----------|-----------|-----------|
##            0 |       790 |        69 |       859 |
##              |     0.920 |     0.080 |     0.949 |
##              |     0.980 |     0.697 |           |
##              |     0.873 |     0.076 |           |
## -------------|-----------|-----------|-----------|
##            1 |        16 |        30 |        46 |
##              |     0.348 |     0.652 |     0.051 |
##              |     0.020 |     0.303 |           |
##              |     0.018 |     0.033 |           |
## -------------|-----------|-----------|-----------|
## Column Total |       806 |        99 |       905 |
##              |     0.891 |     0.109 |           |
## -------------|-----------|-----------|-----------|
##
##
```

Note: The following code is run for *Performance Comparison of the Algorithms* which will be discussed later

Residual Analysis

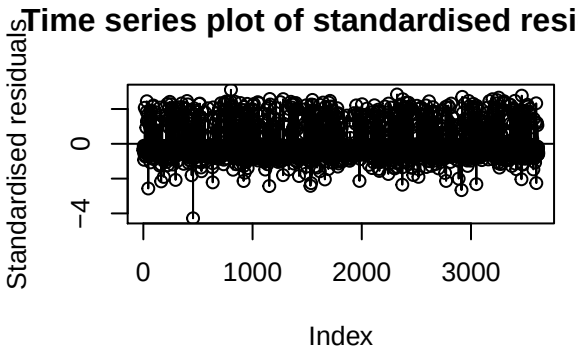
The follwing code builds the function to perform residual analysis which can be used to do residual checks for all the three models.

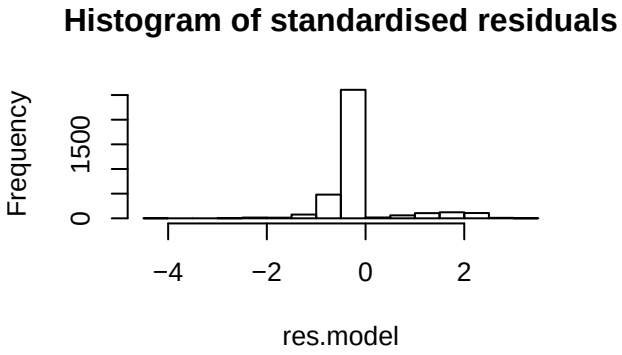
```
residual.analysis <- function(model, std = TRUE){
  library(TSA)
  library(FitAR)
  if (std == TRUE){
    res.model = rstandard(model)
```

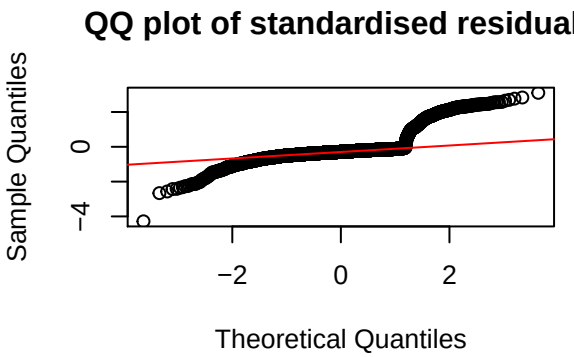
```
  }else{
    res.model = residuals(model)
  }
  par(mfrow=c(2,2))
  plot(res.model,type='o',ylab='Standardised residuals', main="Time series plot of standardised residual
  abline(h=0)
  hist(res.model,main="Histogram of standardised residuals")
  qqnorm(res.model,main="QQ plot of standardised residuals")
  qqline(res.model, col = 2)
  acf(res.model,main="ACF of standardised residuals")
  print(shapiro.test(res.model))
}
```

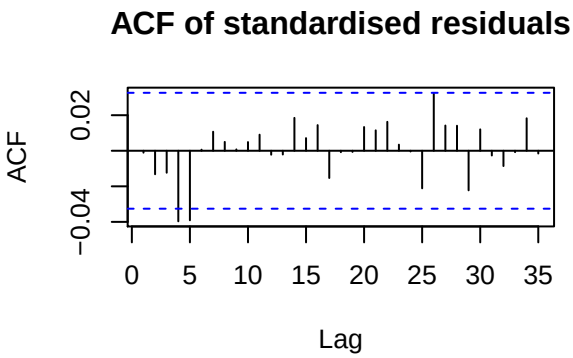
```
residual.analysis(model_std)
```

```
## Warning: package 'TSA' was built under R version 3.4.4
## Loading required package: leaps
## Warning: package 'leaps' was built under R version 3.4.4
## Loading required package: locfit
## Warning: package 'locfit' was built under R version 3.4.4
## locfit 1.5-9.1    2013-03-22
## Loading required package: mgcv
## Loading required package: nlme
## This is mgcv 1.8-17. For overview type 'help("mgcv-package")'.
## Loading required package: tseries
## Warning: package 'tseries' was built under R version 3.4.3
##
## Attaching package: 'TSA'
## The following objects are masked from 'package:stats':
##
##     acf, arima
## The following object is masked from 'package:utils':
##
##     tar
## Warning: package 'FitAR' was built under R version 3.4.4
## Loading required package: lattice
## Loading required package: ltsa
## Warning: package 'ltsa' was built under R version 3.4.1
## Loading required package: bestglm
## Warning: package 'bestglm' was built under R version 3.4.4
```










```
##
##  Shapiro-Wilk normality test
##
## data:  res.model
## W = 0.69562, p-value < 2.2e-16
```

Durbin-Watsson test for autocorrelation of residuals

```
library(car)
```

```
## Warning: package 'car' was built under R version 3.4.3
```

```
##
## Attaching package: 'car'
```

```
## The following object is masked from 'package:FitAR':
##
##     Boot
```

```
durbinWatsonTest(model_std)
```

```
##  lag Autocorrelation D-W Statistic p-value
##    1      0.04239909       1.91499   0.014
##  Alternative hypothesis: rho != 0
```

```
vif(model_std)
```

```
##       age       job   marital education   default   balance   housing
```

```
##  1.409843  1.113330  1.237114  1.400767  1.017904  1.033876  1.231414
##      loan   contact       day     month  duration  campaign     pdays
##  1.041983  8.534122  1.041642  1.144579  1.096505  1.072828  3.451843
##  previous  poutcome   job_unk   edu_unk  cont_unk  pout_unk
##  2.014030  6.280537  1.097560  1.385447  8.621303  8.512286
```

4.2 *K*-Nearest Neighbor Model

4.2.1 Loading the data and Preprocessing

The following chunks of code load the required library to perfrom the K-NN algorithm

```
rm(list=ls(all=TRUE))

library(FNN)         # This is where the fast KNN algorithm sits
library(reshape2)    # Used for reshaping data for use with ggplot.
library(ggplot2)     # We will need to use ggplot to create some graphs.

set.seed(45)         # To ensure the split is repeatable.
```

The following code runs the data into R and do the necessary preprocessing.

```
bank<-read.csv("bank1.csv", stringsAsFactors = FALSE, header = T)
#View(bank)

# This code of chunks creates extra column for variables with unknown values
bank$job_unk <- ifelse(bank$job == "unknown", 1, 0)
bank$edu_unk <- ifelse(bank$education == "unknown", 1, 0)
bank$cont_unk <- ifelse(bank$contact == "unknown", 1, 0)
bank$pout_unk <- ifelse(bank$poutcome == "unknown", 1, 0)

# This code of chunk make the character data into numeric format
bank$job <- as.numeric(as.factor(bank$job))
bank$marital <- as.numeric(as.factor(bank$marital))
bank$education <- as.numeric(as.factor(bank$education))
bank$default<- ifelse(bank$default == "yes", 1, 0)
bank$housing <- ifelse(bank$housing== "yes", 1, 0)
bank$loan<- ifelse(bank$loan== "yes", 1, 0)
bank$month <- as.numeric(as.factor(bank$month))
bank$contact <- as.numeric(as.factor(bank$contact))
bank$poutcome <- as.numeric(as.factor(bank$poutcome))
bank$y <- ifelse(bank$y== "yes", 1, 0)

# Creates normalization function
normalize <- function(x) {
  return ((x - min(x)) / (max(x) - min(x)))
}
# normalize the data
bank <- as.data.frame(lapply(bank, normalize))
# We create our design matrix and target vector
mydata <- bank
mydata.X <- model.matrix(y ~ -1+., data= bank)
mydata.X <- as.data.frame(mydata.X)
mydata.Y <- bank$y
```

```
# Splitting data into training, test and validation sets.
cuts <- c(training = .7, test = .2, validation = .1)
g <- sample(cut(seq(nrow(mydata.X)), nrow(mydata.X)*cumsum(c(0,cuts)), labels = names(cuts)))
final.X <- split(mydata.X, g)
final.Y <- split(mydata.Y, g)
raw <- split(mydata, g)
```

In order to demonstrate bias variance tradeoff the data set needs to be smaller and stratified into samples. K-NN can be affected by having a large quantity of observations in a single class. The R implementation of K-NN uses Euclidean distance.

The following chunk of codes select the random sample of size 500 from the population and split the data in to training, validation and test sets.

```
size <- 4000
sp <- split(mydata, list(mydata$y))
samples <- lapply(sp, function(x) x[sample(1:nrow(x), 500, replace = FALSE), ])
mydata_sample <- do.call(rbind, samples)
row.names(mydata_sample) <- NULL

# this function creats the design matrix and target variables
mydata_sample.X <- model.matrix(y ~ -1+., data= mydata_sample)
mydata_sample.X <- as.data.frame(mydata_sample.X)
mydata_sample.Y <- mydata_sample$y

# We will split the data into training, test and validation sets.
cuts <- c(training = .6, test = .2, validation = .2)

g <- sample(cut(seq(nrow(mydata_sample.X)), nrow(mydata_sample.X)*cumsum(c(0,cuts)), labels = names(cuts
final_sample.X <- split(mydata_sample.X, g)
final_sample.Y <- split(mydata_sample.Y, g)
raw <- split(mydata_sample, g)
```

4.2.2 K-NN Prediction and misclassification Error

On the validation set, initially a random value of k=3 is chosen to make the prediction of the algorithm. Also, to see the accuracy of the model the misclassification error is compouted. The computations are demonstrated by the following codes.

```
nn <- 3 # we choose the random value for validation set
knn.pred <- knn(final_sample.X$training, final_sample.X$validation, final_sample.Y$training,  k = nn, p
```

```
error <- 1 - length(final_sample.Y$validation[final_sample.Y$validation==knn.pred]) / length(final_sampl
error
```

```
## [1] 0.345
```

The code in the for loop will create a data frame in order to plot the bias-variance tradeoff.

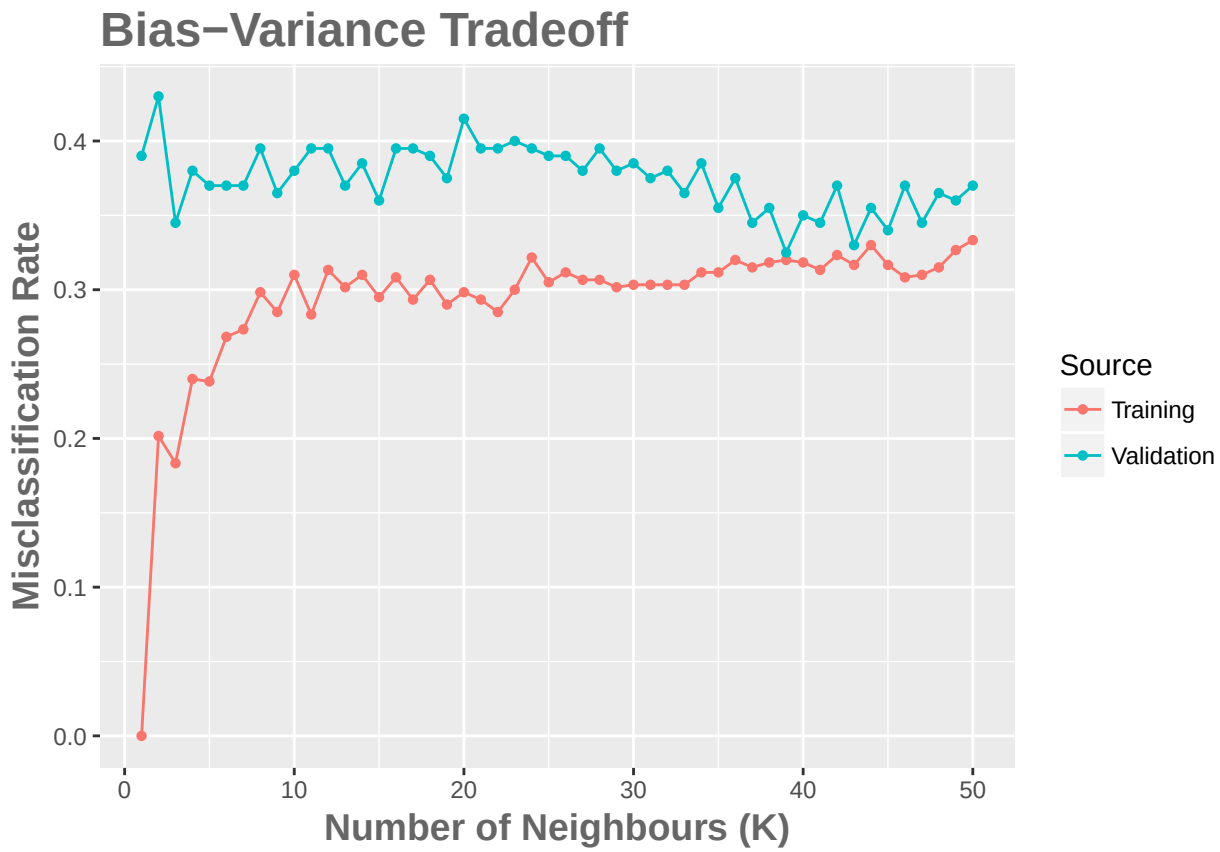
```
maxiter <- 50
bv <- data.frame(k=integer(), Training=double(), Validation=double())
for (nn in 1:maxiter){
  knn.pred1 <- knn(final_sample.X$training, final_sample.X$training, final_sample.Y$training, k=nn)

  knn.pred2 <- knn(final_sample.X$training, final_sample.X$validation, final_sample.Y$training, k=nn)
```

```
  cat("iteration: ", include=FALSE, nn, "\n")
  terr <- 1 - length(final_sample.Y$training[final_sample.Y$training==knn.pred1]) / length(final_sample
  verr <- 1 - length(final_sample.Y$validation[final_sample.Y$validation==knn.pred2]) / length(final_sa
  rec <- data.frame(k=nn, Training=terr, Validation=verr)
  bv <- rbind(bv, rec)
}
```

4.2.3 Bias- Variance Trade-off

Following the execution of the above code the bias variance can be ploted as follows.It is clear that as the number of neighbours(k) increases, the misclassification error increases and stabilizes between 30 to 40 percent. The validation set also shows that the error rate is steady within similar range as the training set.
Also, from the plot it appears the "best" value of K is between 4 or 8. We choose 4 as it is a simpler model and this will be used it to score our test set

```
bv_melt <- melt(bv, id.vars = "k", variable.name = "Source", value.name = "Error")
title <- "Bias-Variance Tradeoff"
ggplot(bv_melt, aes(x=k, y=Error, color=Source)) +
  geom_point(shape=16) + geom_line() +
  xlab("Number of Neighbours (K)") +
  ylab("Misclassification Rate") +
  ggtitle(title) +
  theme(plot.title = element_text(color="#666666", face="bold", size=18, hjust=0)) +
  theme(axis.title = element_text(color="#666666", face="bold", size=14))
```



```
nn <- 4
knn.pred3 <- knn(final_sample.X$training, final_sample.X$test, final_sample.Y$training, k=nn)
```

4.2.4 Misclassification error in the test set

```
error1 <- 1 - length(final_sample.Y$test[final_sample.Y$test==knn.pred3]) / length(final_sample.Y$test)
error1
```

```
## [1] 0.365
```

Confusion matrix from test set

```
table(knn.pred3, final_sample.Y$test)
```

```
##
## knn.pred3  0  1
##         0 82 50
##         1 23 45
```

Cross Table will return the confusion matrix with more infomation where we can get the proportion of each cell value making easy coparision. From the following cross table it is observed that the model predicted 62% of non-subscription of term deposit correctly and 66% of the subscription of term deposit correctly.

```
library(gmodels)
CrossTable(knn.pred3, final_sample.Y$test, prop.chisq = FALSE)
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |           N / Row Total |
## |           N / Col Total |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  200
##
##
##              | final_sample.Y$test
##    knn.pred3 |         0 |         1 | Row Total |
## -------------|-----------|-----------|-----------|
##            0 |        82 |        50 |       132 |
##              |     0.621 |     0.379 |     0.660 |
##              |     0.781 |     0.526 |           |
##              |     0.410 |     0.250 |           |
## -------------|-----------|-----------|-----------|
##            1 |        23 |        45 |        68 |
##              |     0.338 |     0.662 |     0.340 |
##              |     0.219 |     0.474 |           |
##              |     0.115 |     0.225 |           |
## -------------|-----------|-----------|-----------|
## Column Total |       105 |        95 |       200 |
##              |     0.525 |     0.475 |           |
```

```
## -------------|-----------|-----------|-----------|
##
##
```

4.3. Naive Bayes Model

4.3.1 Loading the data and Preprocessing

The following chunks of code load the required library to perfrom the logistic regression

```
rm(list=ls())
library(ggplot2) # we need ggplot2 to run caret pakage
library(caret)   # We'll need to use caret to create confusion matrix.
                 # This allows for cross validation using Naive Bayes using the Klar library.
library(e1071)   # e1071 library and employ the naiveBayes function to build the classifier

set.seed(45)     # Since we're going to split our data we need to ensure the split is repeatable.
```

The following code runs the data into R and do the necessary preprocessing.

```
#Import data to R
bank<-read.csv("bank1.csv", stringsAsFactors = FALSE, header = T)
#View(bank)

# This code of chunks creates extra column for variables with unknown values
bank$job_unk <- ifelse(bank$job == "unknown", 1, 0)
bank$edu_unk <- ifelse(bank$education == "unknown", 1, 0)
bank$cont_unk <- ifelse(bank$contact == "unknown", 1, 0)
bank$pout_unk <- ifelse(bank$poutcome == "unknown", 1, 0)

# This code of chunk make the character data into data frame format
bank$job <- as.numeric(as.factor(bank$job))
bank$marital <- as.numeric(as.factor(bank$marital))
bank$education <- as.numeric(as.factor(bank$education))
bank$default<- ifelse(bank$default == "yes", 1, 0)
bank$housing <- ifelse(bank$housing== "yes", 1, 0)
bank$loan<- ifelse(bank$loan== "yes", 1, 0)
bank$month <- as.numeric(as.factor(bank$month))
bank$contact <- as.numeric(as.factor(bank$contact))
bank$poutcome <- as.numeric(as.factor(bank$poutcome))

# We need the target varible in the factor format.
bank$y <- as.factor(bank$y)

ind = sample(2, nrow(bank), replace = TRUE, prob=c(0.7, 0.3))
trainset = bank[ind == 1,]
testset = bank[ind == 2,]

# This code returns the dimention of our training and test sets. first column represents the numebr of
dim(trainset)
```

```
## [1] 3140   21
```

```
dim(testset)
```

```
## [1] 1381   21
```

The following code returns the percentage of customer subscribing term deposit from whole data set

```
pctPos <- nrow(testset[testset$y == "yes",]) / nrow(testset)
pctPos
```

```
## [1] 0.1115134
```

Only eleven percentage of people in our test set suscribe the term deposite. Let's see how much will predict by our model.

Employ the naive Bayes function to build the classifier

```
model <- naiveBayes(trainset[, !names(trainset) %in% c("y")],
                    trainset$y, na.action = na.pass)
# Type classifier to examine the function call, a-priori probability, and conditional
#probability
model
```

```
##
## Naive Bayes Classifier for Discrete Predictors
##
## Call:
## naiveBayes.default(x = trainset[, !names(trainset) %in% c("y")],
##     y = trainset$y, na.action = na.pass)
##
## A-priori probabilities:
## trainset$y
##       no      yes
## 0.883121 0.116879
##
## Conditional probabilities:
##           age
## trainset$y     [,1]     [,2]
##        no  40.94266 10.17058
##        yes 42.31608 12.89802
##
##           job
## trainset$y     [,1]     [,2]
##        no  5.409304 3.272852
##        yes 5.509537 3.128276
##
##           marital
## trainset$y     [,1]      [,2]
##        no  2.148936 0.5889462
##        yes 2.177112 0.6642464
##
##           education
## trainset$y     [,1]      [,2]
##        no  2.223585 0.7524317
##        yes 2.269755 0.7248408
##
##           default
## trainset$y       [,1]      [,2]
##        no  0.01767039 0.1317741
##        yes 0.01907357 0.1369704
##
##           balance
```

```
## trainset$y     [,1]     [,2]
##        no  1349.691 2813.552
##        yes 1610.853 2601.014
##
##           housing
## trainset$y      [,1]      [,2]
##        no  0.5791561 0.4937836
##        yes 0.4277929 0.4954341
##
##           loan
## trainset$y       [,1]      [,2]
##        no  0.15831230 0.3650994
##        yes 0.09264305 0.2903274
##
##           contact
## trainset$y     [,1]      [,2]
##        no  1.694194 0.9165538
##        yes 1.326975 0.6870443
##
##           day
## trainset$y     [,1]     [,2]
##        no  15.86441 8.181408
##        yes 15.47411 8.151487
##
##           month
## trainset$y     [,1]     [,2]
##        no  6.610530 2.949636
##        yes 6.427793 3.430750
##
##           duration
## trainset$y     [,1]     [,2]
##        no  227.5662 218.0825
##        yes 537.3542 377.1710
##
##           campaign
## trainset$y     [,1]     [,2]
##        no  2.843851 3.170446
##        yes 2.141689 1.972926
##
##           pdays
## trainset$y     [,1]      [,2]
##        no  37.09809  96.73482
##        yes 69.76839 122.61713
##
##           previous
## trainset$y      [,1]     [,2]
##        no  0.4835918 1.626186
##        yes 1.1389646 2.166507
##
##           poutcome
## trainset$y     [,1]      [,2]
##        no  3.571944 0.9990792
##        yes 3.305177 1.0633232
##
```

```
##           job_unk
## trainset$y        [,1]       [,2]
##        no  0.009376127 0.09639277
##        yes 0.008174387 0.09016495
##
##           edu_unk
## trainset$y       [,1]      [,2]
##        no  0.04543815 0.2083007
##        yes 0.03269755 0.1780866
##
##           cont_unk
## trainset$y      [,1]      [,2]
##        no  0.3137396 0.4640956
##        yes 0.1253406 0.3315567
##
##           pout_unk
## trainset$y      [,1]      [,2]
##        no  0.8362784 0.3700895
##        yes 0.6348774 0.4821218
```

The priori and conditional probabilities has been observed from the above outputs.

Rename the data (predictors) to performe the prediction

```
x<- testset[, !names(testset) %in% c("y")]
y <- testset$y
```

4.3.2 Prediction and misclassification Error

```
bayes.table <- table(predict(model, x), y)
bayes.table
```

```
##      y
##         no  yes
##   no  1055   76
##   yes  172   78
```

```
1-sum(bayes.table[row(bayes.table)==col(bayes.table)])/sum(bayes.table)
```

```
## [1] 0.17958
```

We got about 17% misclassification error while doing the prediction of customer suscribe to term deposit.

Confusion matrix

```
confusionMatrix(bayes.table)
```

```
## Confusion Matrix and Statistics
##
##      y
##         no  yes
##   no  1055   76
##   yes  172   78
##
##                Accuracy : 0.8204
##                  95% CI : (0.7991, 0.8403)
##     No Information Rate : 0.8885
```

```
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.2879
##  Mcnemar's Test P-Value : 1.614e-09
##
##             Sensitivity : 0.8598
##             Specificity : 0.5065
##          Pos Pred Value : 0.9328
##          Neg Pred Value : 0.3120
##              Prevalence : 0.8885
##          Detection Rate : 0.7639
##    Detection Prevalence : 0.8190
##       Balanced Accuracy : 0.6832
##
##        'Positive' Class : no
##
```

The confusion matrix above shows the 82% of acuracy and 95% confidence interval of the perdicted acuracy. The cross table from the confusion matrix shows that model predicted more closely for the customer who did not suscribe term deposit but for thoese custome who had suscribed term deposit has been predicted badly.

5. Performance Comparison of the Algorithms

As already seen the three models we have built have their own accuracy of predicting whether a client will say "yes" or "no" to a term deposit of the bank. As expected there is some variation in the misclassification error rate among the three classification algorithm.

Algorithms Error Rate

1. Regression Model 9.40%
2. K-NN classifier 36.5%
3. Naive Bayes Classifier 17.95%

Based on the misclassification error rate, the most reliable model for the data set appears to be the logistic regression model with just 9.4%.

5.1 Adequacy of Logistic regression Model

Note: refer to the outputs from the end of the regresson algorithms

Residual Analysis

The residual analysis includes check for normality, autocorrelation and time series plot to inspect if there is any trend present in the residuals. As per the previous output in the regression algorithm section, there is no autocorrelation parts in this model confirming a white noise process. However, the residuals are not purely normally distributed which is seen from Shapiro-Wilk test and histogram of the standarized residuals. Time series plot of residuals indicates residuals have almost equal change in variances and non- existence of trends.

Variable Inflation Factor (VIF) Test for Multicollinarity of Variables'

Presence of collinearity among the variables negatively affects logistic regression model. The measure of VIF for a variables greater than 5 is usually considered to create collinearity. From the output it is seen almost all the variables have VIF less than 5 which proves that logistic regression is not influenced by collinearity. There are few atributes with vif greater than 5, but they are not significant to the model.

Durbin-Watsson test of model

Durbin-Watson is another test to find the effect of autocorrelation in a data set. The appropriate hypothesis for this test is H0: Errors (residuals) are uncorrelated H1: Errors (residuals) are correlated

Since the p-value is less than 0.05 we have enough evidence to reject H0. This implies that the residuals are correlated.

6. Advantages and Limitations of Algorithms

The advantage of logistic regression is that it is easy to interpret, it directs model logistic probability, and provides a confidence interval for the result. However, the main drawback of the logistic algorithm is that it suffers from multicollinearity and, therefore, the explanatory variables must be linearly independent. But, it is certain that the variables of the model do not exhibit multicollinearity as shown by VIF test.

Some limitations of logistic regression approach in context to the above model are

i. Model have some class of unkown predictors which are significant in the model. These variables actually does not carry any useful information fundamentally but their significance might affect the predictability of the model.

ii. Residuals of the model are are not normally distributed when doing the residual analysis are not reliable though the model demonstrate a good accuracy.

iii. Residuals are correlated in the Durbin-Watson test.The test for autocorrelation using the Durbin-Watson test proved that the residuals are correlated. This shows that the residuals are have an autocorrelation effect which might accect the models accuracy.

7. Conclusion

From the study conducted, the results are impressinve and convincing in terms of using a machine learning algorithm to decide on the marketing campaign of the bank. Almost all of the attributes contribute significantly to the building of a predictive model. Among the three classification approach used to model the data, the logistic regression model yielded the best accuracy with just 9.4% misclassification error rate. This model is simple and easy to implement.

The bank marketing manager can identify potential client by using the model if the client's information like education, housing loan, Personal loan, duration of call, number of contacts performed during this campaign, previous outcomes, etc is available. This will help in minimizing the cost to the bank by avoiding to call customers who are unlikely to subscribe the term deposit. They can run a more successful tele-marketing campaign using this model.

Acknowlegement

We would like to express our sincere thanks of gratitude to our project mentor and guide Dr. Vural Aksakalli and Mr. Nigel Clay for giving us the opportunity to do this wonderful project. It is a great honour to get continuous support and guidance throughout the project from both of them. It was quite a testing time in the initially phase as we were challenged understanding the data set and to pick out the best machine learning algorithms. The meetings we had with Mr. Nigel helped us visualize the problem more practically and gained numerous other ideas on handling a machine learning assignment. This made us more equipped and kept us encouraged for the task. Our, acknowledgement would be incomplete without the mention of the weekly laboratory sessions delivered by Mr. Nigel which contributed hugely into our learning and compiling of the entire project.

References

UCI Machine Learning Repository, *Bank Marketing Data Set* viewed online on

https://archive.ics.uci.edu/ml/datasets/Bank+Marketing

James, G, Witten, D, Hastie, T, and Tibshirani, R 2013, "*An Introduction to Statistical Learning With Application in R*"

Yu-Wei and Chiu, 2015, "*Macchine Larning With R Cookbook*" , Published by Packt Publishing Ltd. Livery Place, 35 Livery Street, Birmingham B3 2PB, UK.

Lantz, B, 2015, "*Macchine Larning With R* ", second edition , Published by Packt Publishing Ltd. Livery Place, 35 Livery Street, Birmingham B3 2PB, UK.