#======================================================================================

# First Part

#======================================================================================

# Import and prepare the EWCS dataset.

ewcs=read.table("EWCS\_2016.csv",sep=",",header=TRUE)

ewcs[,][ewcs[, ,] == -999] <- NA

kk=complete.cases(ewcs)

ewcs=ewcs[kk,]

# library

require(graphics)

library(ggplot2)

library(factoextra)

library(corrplot)

library(magrittr)

library(ggfortify)

ewcs\_summary <- prcomp(ewcs,center = TRUE,scale = TRUE)

print(ewcs\_summary)

# It demonstrates degree of contribution of each variable

screeplot(ewcs\_summary, main = "Screeplot of ewcs", col = "steelblue", type = "line", pch = 1, npcs = length(ewcs\_summary$sdev))

summary(ewcs\_summary)

# PC6's standard deviation tells us it's 0.75 dispersed in relation to the mean

# of the data, proportion of variance tells us the influence contributed and

# cumulative proportion tells us that 83% of the data is explained by PC6.

biplot(ewcs\_summary)

# The closer the distance and direction, the higher the correlation of the

# variables.Thus we can assume that Q87a ~ Q87e are correlated and Q90a,Q90b,

# Q90c and Q90f are correlated.

# Hence change the column names and group them into two different classification,

# "Life" and "Work". L for life and W for work

ewcs <- setNames(ewcs, c("Gender","Age","L\_Cheerful","L\_Calm","L\_Active","L\_WakeUpFreshed","L\_Interesting","W\_Energetic","W\_Enthusiastic","W\_TimeFlies","W\_Expert"))

# Extract variables to look at the various aspects of the data

LifeWork\_satisfaction <- ewcs[, 3:11]

life\_satisfaction <- ewcs[, 3:7]

work\_satisfaction <- ewcs[, 8:11]

# Among "Life", "WakeUpFreshed" has the biggest variance

apply(life\_satisfaction, 2, var)

# Among "Work", "Enthusiastic" has the biggest variance

apply(work\_satisfaction, 2, var)

# Life and work satisfaction variables

LifeWork\_satisfaction\_summary <- prcomp(LifeWork\_satisfaction, center = TRUE,scale = TRUE)

print(LifeWork\_satisfaction\_summary)

screeplot(LifeWork\_satisfaction\_summary, main = "Screeplot of life and work satisfaction summary", col = "steelblue", type = "lines", pch = 1, npcs = length(LifeWork\_satisfaction\_summary$sdev))

summary(LifeWork\_satisfaction\_summary)

# For life-work satisfaction which includes both “life” and “work” variables,

# 5 principal component (PC) seems to be the elbow point, a point at which the

# slope changes. With the summary and plot, we can identify that PC5 has

# cumulative proportion of 85% which then means with 5 PC, we can explain 85% of

# the variation. Hence, 5 PC could be a simpler substitute for all 9 factors

# as it could explain most of the variability without losing copious amount of

# its initial variability.

life\_satisfaction\_summary <- prcomp(life\_satisfaction, center = TRUE,scale = TRUE)

print(life\_satisfaction\_summary)

screeplot(life\_satisfaction\_summary, main = "Screeplot of life satisfaction summary", col = "steelblue", type = "lines", pch = 1, npcs = length(life\_satisfaction\_summary$sdev))

summary(life\_satisfaction\_summary) # PC1, 71%

work\_satisfaction\_summary <- prcomp(work\_satisfaction,center = TRUE, scale = TRUE)

print(work\_satisfaction\_summary)

screeplot(work\_satisfaction\_summary, main = "Screeplot of work satisfaction summary", col = "steelblue", type = "lines", pch = 1, npcs = length(work\_satisfaction\_summary$sdev))

summary(work\_satisfaction\_summary)# PC2, 74%

# zoom out to see the clear plot

ewcs\_cor <- cor(ewcs)

corrplot(ewcs\_cor, main="Correlation matrix of ewcs",method="number")

# From the correlation matrix, we can identify that gender and age have very low correlation to

# each other and also to life and work satisfaction variables. Furthermore, we can once again identify that

# life satisfaction variables(cheerful,calm,active,wakeupfreshed,interesting) and

# work satisfaction variables(energetic,enthusiastic,timeflies,expert) have correlation within

# each of the variables.

# Hence, to see the clear picture, we will sum up the score of each life and work satisfaction.

ewcs\_sum <- ewcs

ewcs\_sum$Life <- ewcs\_sum$L\_Cheerful + ewcs\_sum$L\_Calm + ewcs\_sum$L\_Active + ewcs\_sum$L\_WakeUpFreshed + ewcs\_sum$L\_Interesting

ewcs\_sum$Work <- ewcs\_sum$W\_Energetic + ewcs\_sum$W\_Enthusiastic + ewcs\_sum$W\_TimeFlies + ewcs\_sum$W\_Expert

# Drop unnecessary columns

ewcs\_sum <- ewcs\_sum[,-(3:11)]

ewcs\_sum\_cor <- cor(ewcs\_sum)

corrplot(ewcs\_sum\_cor, main = "Correlation matrix of adjusted ewcs",method="number")

# There is correlation between life and work.

# Deep dive into PCA

# Turn Gender into a factor

ewcs$Gender <- factor(ifelse(ewcs$Gender > 1, "Female", "Male"))

ewcs1<- prcomp(ewcs[,2:11],center = TRUE,scale = TRUE)

# Degree of contribution of each variable

print(ewcs1)

screeplot(ewcs1, main = "Screeplot of ewcs1", col = "steelblue", type = "line", pch = 1, npcs = length(ewcs1$sdev))

summary(ewcs1)

# With 4 PC, we can explain 76% of the variance.

autoplot(ewcs1, data = ewcs, colour = 'Gender',

label = FALSE,

loadings = TRUE, loadings.colour = 'steelblue',

loadings.label = TRUE,

loadings.label.size = 4,

loadings.label.colour = "black",

loadings.label.repel=T) +

theme(legend.text = element\_text(size = 8),

legend.title = element\_text(size = 10),

axis.title = element\_text(size = 8))

# Adjusted ewcs

ewcs2<- prcomp(ewcs\_sum[,2:4],center = TRUE,scale = TRUE)

print(ewcs2)

screeplot(ewcs2, main = "Screeplot of ewcs2", col = "steelblue", type = "line", pch = 1, npcs = length(ewcs2$sdev))

summary(ewcs2)

# The first and second PC explains 50% and 33% of the total variance. First PC

# has large negative associations of Life and Work while second PC has positive

# associations of age. We can once again identify with a clear picture that “Life”

# and “Work” have positive correlation to each other but not to gender and age.

autoplot(ewcs2, data = ewcs, colour = 'Gender',

label = FALSE,

loadings = TRUE, loadings.colour = 'steelblue',

loadings.label = TRUE,

loadings.label.size = 4,

loadings.label.colour = "black",

loadings.label.repel=T) +

theme(legend.text = element\_text(size = 8),

legend.title = element\_text(size = 10),

axis.title = element\_text(size = 8))

#======================================================================================

# Second Part

#======================================================================================

library(dplyr)

library(caret)

library(caTools)

library(randomForest)

library(glmnet)

library(car)

library(ggpubr)

library(e1071)

# Import and prepare the student performance dataset.

school1=read.table("student-mat.csv",sep=";",header=TRUE) # Mathematics

school2=read.table("student-por.csv",sep=";",header=TRUE) # Portuguese

schools=merge(school1, school2, by=c("school","sex","age","address","famsize","Pstatus","Medu","Fedu","Mjob","Fjob","reason","nursery","internet"))

# We don't see any missing values in the data

any(is.na(schools))

summary(schools)

# EDA==============================================================================

schools\_sex <- schools %>%

group\_by(sex) %>%

summarise(Average\_G3\_mat = mean(G3.x),Average\_G3\_por = mean(G3.y))

ggplot(data = schools\_sex, mapping = aes(x = sex, y = Average\_G3\_mat)) +

geom\_bar(stat='identity',fill="steelblue") +

geom\_text(mapping = aes(label = round(Average\_G3\_mat,0), fontface = 'bold', vjust = -0.2), size = 2) +

labs(title = "Average mat G3 ",x = "Gender", y ="Average G3")+

theme(plot.title = element\_text(hjust = 0.5))

ggplot(data = schools\_sex, mapping = aes(x = sex, y = Average\_G3\_por)) +

geom\_bar(stat='identity',fill="steelblue") +

geom\_text(mapping = aes(label = round(Average\_G3\_por,0), fontface = 'bold', vjust = -0.2), size = 2) +

labs(title = "Average por G3",x = "Gender", y ="Average G3")+

theme(plot.title = element\_text(hjust = 0.5))

# Female students tend to do better in Portuguese language then male students.

# Male students tend to do better in Mathematics then female students.

# Multiple linear regression======================================================

# We will first Assume that the data meets the assumptions for linear regression.

# Then later on, we will adjust the model to meet its assumptions.

# To build model for G3, drop G1 and G2.

drop <- c("G1.x","G2.x","G1.y","G2.y")

schools\_G3 = schools[,!(names(schools) %in% drop)]

# Switch back to two distinct subjects with complete period grades.

mat <- schools\_G3[,1:31]

por <- schools\_G3[, -c(14:31)]

# See the correlation between them

mat\_numeric <- select\_if(mat, is.numeric)

mat\_numeric\_cor <-cor(mat\_numeric)

corrplot(mat\_numeric\_cor,main = "Correlation matrix of mat", method="number")

por\_numeric <- select\_if(por, is.numeric)

mat\_numeric\_por <-cor(por\_numeric)

corrplot(mat\_numeric\_por, main = "Correlation matrix of por",method="number")

#we can observe that G3 is slightly correlated to medu, fedu, and studytime

hist(mat$G3.x)

hist(por$G3.y)

# Seems left skewed

set.seed(1)

# Create a list of 70% of the rows in the mat and por.

mat\_training\_sample <- createDataPartition(mat$G3.x,p = 0.7, list = FALSE)

por\_training\_sample <- createDataPartition(por$G3.y,p = 0.7, list = FALSE)

# Use the 70% of data to train the model

mat\_dataset<- mat[mat\_training\_sample,]

por\_dataset<- por[por\_training\_sample,]

# Use the 30% of the data to validate the model.

mat\_validation <- mat[-mat\_training\_sample,]

por\_validation <- por[-por\_training\_sample,]

# Split input and output for training and validation

## Training

mat\_dataset\_output <- mat\_dataset[,31]

por\_dataset\_output <- por\_dataset[,31]

mat\_dataset\_input <- mat\_dataset[,1:30]

por\_dataset\_input <- por\_dataset[,1:30]

## Validation

mat\_validation\_output <- mat\_validation[,31]

por\_validation\_output <- por\_validation[,31]

mat\_validation\_input <- mat\_validation[,1:30]

por\_validation\_input <- por\_validation[,1:30]

# Train

set.seed(1)

mat\_lm <- lm(G3.x~.,mat\_dataset)

por\_lm <- lm(G3.y~.,por\_dataset)

sqrt(mean(mat\_lm$residuals^2)) # 3.730919

sqrt(mean(por\_lm$residuals^2)) # 2.110835

summary(mat\_lm)$r.squared # 0.3454868

summary(por\_lm)$r.squared # 0.4543227

summary(mat\_lm)

summary(por\_lm)

# Test

mat\_lm\_validate <- lm(G3.x~.,mat\_validation)

por\_lm\_validate <- lm(G3.y~.,por\_validation)

sqrt(mean(mat\_lm\_validate$residuals^2)) # 3.341797

sqrt(mean(por\_lm\_validate$residuals^2)) # 2.00442

summary(mat\_lm\_validate)$r.squared # 0.5227731

summary(por\_lm\_validate)$r.squared # 0.5901086

# <lm>

## <Train>

## mat por

## RMSE 3.730919 2.110835

## R2 0.3454868 0.4543227

## <Test>

## mat por

## RMSE 3.341797 2.00442

## R2 0.5227731 0.5901086

# < normality test >

# We will test the assumptions of a multiple linear regression

par(mfrow=c(2,2))

plot(mat\_lm)

plot(por\_lm)

# No distinctive pattern for residuals vs. fitted.

# Looking at normal Q-Q, distribution of residuals seem heavy on the tails.

# Homoscedasticity for Scale-location.

# Looking at residuals vs. leverage, we

# cant see Cook’s distance lines, hence there are no influential case.

# Furthermore, besides Q-Q plot, we can check the normality with density plot

# and normality test.

# Density plot

ggdensity(mat$G3.x)+

labs(title = "Density plot of G3.x",x = "G3", y ="Density")

ggdensity(por$G3.y)+

labs(title = "Density plot of G3.y",x = "G3", y ="Density")

skewness(mat$G3.x) #-0.70 = moderately skewed

skewness(por$G3.y) #-0.99 = moderately skewed

# density plots seem moderately left skewed

# Shapiro-Wilk Normality Test

shapiro.test(mat\_lm$residuals)

shapiro.test(por\_lm$residuals)

# Both of the p-values are < 0.05, thus the residuals are not normally distributed.

# Identify Multicollinearity

vif(mat\_lm)

vif(por\_lm)

# They are all below 5, there are no multicollinearity detected

# Identify outliers

mat\_standard\_residuals <-rstandard(mat\_lm)

por\_standard\_residuals <-rstandard(por\_lm)

mat\_lm\_sr <- cbind(mat\_dataset, mat\_standard\_residuals)

por\_lm\_sr <- cbind(por\_dataset, por\_standard\_residuals)

#sort standardized residuals by descending and ascending order

head(mat\_lm\_sr[order(-mat\_standard\_residuals),]$mat\_standard\_residuals)

head(mat\_lm\_sr[order(mat\_standard\_residuals),]$mat\_standard\_residuals)

head(por\_lm\_sr[order(-por\_standard\_residuals),]$por\_standard\_residuals)

head(por\_lm\_sr[order(por\_standard\_residuals),]$por\_standard\_residuals)

# mat's standardized residual of #199 and por's standardized residual of #371, 382, 331 exceeds -3.

# This is an outlier. We should investigate them further

# to verify that they’re not a result of a data entry error or some other odd occurrence.

# Furthermore, we could try using log(y) or root(y) to adjust values of y to

# make it normally distributed.

# Log transformation

## log(y) (log-linear model)

mat\_lm\_log <- lm(log(G3.x+1)~.,mat\_dataset)

por\_lm\_log <- lm(log(G3.y+1)~.,por\_dataset)

par(mfrow=c(2,2))

plot(mat\_lm\_log)

plot(por\_lm\_log)

shapiro.test(mat\_lm\_log$residuals)

shapiro.test(por\_lm\_log$residuals)

# both not normally distributed as P-value < 0.05

## sqrt(y)

mat\_lm\_sq <- lm(sqrt(G3.x)~.,mat\_dataset)

por\_lm\_sq <- lm(sqrt(G3.y)~.,por\_dataset)

par(mfrow=c(2,2))

plot(mat\_lm\_sq)

plot(por\_lm\_sq)

shapiro.test(mat\_lm\_sq$residuals)

shapiro.test(por\_lm\_sq$residuals)

# both not normally distributed as P-value < 0.05

# As we make y smaller, it seems to get worse, hence try increasing y.

## y^(1.3 ~ 1.7) gives normally distributed result

mat\_lm\_adjusted <- lm((G3.x)^(3/2)~.,mat\_dataset)

por\_lm\_adjusted <- lm((G3.y)^(3/2)~.,por\_dataset)

par(mfrow=c(2,2))

plot(mat\_lm\_adjusted, main= "mat adjusted")

plot(por\_lm\_adjusted, main= "por adjusted")

shapiro.test(mat\_lm\_adjusted$residuals)

shapiro.test(por\_lm\_adjusted$residuals)

# both normally distributed as p-value > 0.05

# y^ to 1.3 ~ 1.7 seems normally distributed. Hence we will use 3/2 or 1.5, the number in between.

summary(mat\_lm\_adjusted)

summary(por\_lm\_adjusted)

# Regularization with lasso regression

# k-fold cross-validation to find optimal lambda value

mat\_dataset\_input\_matrix <- model.matrix(G3.x~.-1, mat\_dataset)

por\_dataset\_input\_matrix <- model.matrix(G3.y~.-1, por\_dataset)

mat\_validation\_input\_matrix <- model.matrix(G3.x~.-1, mat\_validation)

por\_validation\_input\_matrix <- model.matrix(G3.y~.-1, por\_validation)

set.seed(1)

mat\_lasso <- glmnet(mat\_dataset\_input\_matrix, mat\_dataset\_output^(3/2), alpha = 1)

por\_lasso <- glmnet(por\_dataset\_input\_matrix, por\_dataset\_output^(3/2), alpha = 1)

cv\_mat <- cv.glmnet(mat\_dataset\_input\_matrix, mat\_dataset\_output^(3/2), alpha = 1)

cv\_por <- cv.glmnet(por\_dataset\_input\_matrix, por\_dataset\_output^(3/2), alpha = 1)

# Lambda value which minimizes MSE testing

best\_cv\_mat\_lambda <- cv\_mat$lambda.min

best\_cv\_mat\_lambda # 1.125309

best\_cv\_por\_lambda <- cv\_por$lambda.min

best\_cv\_por\_lambda # 0.5314042

# Plotting MSE test

par(mfrow=c(1,1))

plot(cv\_mat)

plot(cv\_por)

# Coefficients of the best model

best\_mat <- glmnet(mat\_dataset\_input\_matrix, mat\_dataset\_output^(3/2), alpha = 1, lambda = best\_cv\_mat\_lambda)

coef(best\_mat)

best\_por <- glmnet(por\_dataset\_input\_matrix, por\_dataset\_output^(3/2), alpha = 1, lambda = best\_cv\_por\_lambda)

coef(best\_por)

# Predictors without coefficient tells us that it is not influential to the model.

# Use lasso regression model to predict response value

best\_mat\_lasso <- predict(best\_mat, s = best\_cv\_mat\_lambda, type="coefficients")

best\_por\_lasso <- predict(best\_por, s = best\_cv\_por\_lambda, type="coefficients")

# As we have adjusted our model by y^3/2 to make it normally distributed, we will adjust it back

# by y^2/3

# Training lasso regression model

mat\_dataset\_pred <- predict(mat\_lasso, s = best\_cv\_mat\_lambda, newx = mat\_dataset\_input\_matrix)

por\_dataset\_pred <- predict(por\_lasso, s = best\_cv\_por\_lambda, newx = por\_dataset\_input\_matrix)

postResample(mat\_dataset\_pred^(2/3),mat\_dataset\_output)

# RMSE Rsquared MAE

# 4.0240604 0.2673442 3.0042883

postResample(por\_dataset\_pred^(2/3),por\_dataset\_output)

# RMSE Rsquared MAE

# 2.2005416 0.4270536 1.6611030

# Testing lasso regression model

mat\_validation\_pred <- predict(mat\_lasso, s = best\_cv\_mat\_lambda, newx = mat\_validation\_input\_matrix)

por\_validation\_pred <- predict(por\_lasso, s = best\_cv\_por\_lambda, newx = por\_validation\_input\_matrix)

postResample(mat\_validation\_pred^(2/3),mat\_validation\_output)

# RMSE Rsquared MAE

# 4.7071885 0.1065235 3.4617938

postResample(por\_validation\_pred^(2/3),por\_validation\_output)

# RMSE Rsquared MAE

# 2.8858860 0.1598111 1.9985826

# <lasso>

## <Train>

## mat por

## RMSE 4.0240604 2.2005416

## R2 0.2673442 0.4270536

## <Test>

## mat por

## RMSE 4.7071885 2.8858860

## R2 0.1065235 0.1598111

# Random Forest==============================================================

# Furthermore, we will try using tree based method for improvements.

control <- trainControl(method="repeatedcv", number=10, repeats = 3)

metric <- "RMSE"

# Train the model

set.seed(1)

mat\_rf\_train <- train(G3.x~., data=mat\_dataset, method="rf", importance=TRUE

,tuneGrid= expand.grid(.mtry=sqrt(ncol(mat\_dataset\_input))),metric=metric, trControl=control)

set.seed(1)

por\_rf\_train <- train(G3.y~., data=por\_dataset, method="rf", importance=TRUE

,tuneGrid= expand.grid(.mtry=sqrt(ncol(por\_dataset\_input))),metric=metric, trControl=control)

mat\_rf\_train

# RMSE Rsquared MAE

# 3.936823 0.2973943 2.996881

por\_rf\_train

# RMSE Rsquared MAE

# 2.345072 0.3521341 1.793668

# Test the model

set.seed(1)

mat\_rf\_validate <- train(G3.x~., data=mat\_validation, method="rf", importance=TRUE

,tuneGrid= expand.grid(.mtry=sqrt(ncol(mat\_validation\_input))),metric=metric, trControl=control)

set.seed(1)

por\_rf\_validate <- train(G3.y~., data=por\_validation, method="rf", importance=TRUE

,tuneGrid= expand.grid(.mtry=sqrt(ncol(por\_validation\_input))),metric=metric, trControl=control)

mat\_rf\_validate

# RMSE Rsquared MAE

# 4.417505 0.2494508 3.419602

por\_rf\_validate

# RMSE Rsquared MAE

# 2.76248 0.2834576 2.02071

# ==========<RF>==========

## <Train>

## mat por

## RMSE 3.936823 2.345072

## R2 0.2973943 0.3521341

## <Test>

## mat por

## RMSE 4.417505 2.76248

## R2 0.2494508 0.2834576

# or other way to do this is by

set.seed(1)

rf\_test\_mat <- randomForest(G3.x~ ., data = mat\_dataset, mtry = sqrt(ncol(mat\_dataset\_input)),

importance = TRUE)

rf\_test\_por <- randomForest(G3.y~ ., data = por\_dataset, mtry = sqrt(ncol(por\_dataset\_input)),

importance = TRUE)

rf\_test\_mat

rf\_test\_por

varImp(rf\_test\_mat)

varImp(rf\_test\_por)

varImpPlot(rf\_test\_mat)

varImpPlot(rf\_test\_por)

# Shows importance for each variable to the model.

# train RF

## mat

y\_hat\_mat\_train <- predict(rf\_test\_mat, mat\_dataset)

y\_hat\_mat\_train\_scored <- as\_tibble(cbind(mat\_dataset, y\_hat\_mat\_train))

y\_hat\_mat\_train\_rmse <- yardstick::rmse(y\_hat\_mat\_train\_scored, truth=mat\_dataset\_output, estimate=y\_hat\_mat\_train)

y\_hat\_mat\_train\_rmse # 1.9

## por

y\_hat\_por\_train <- predict(rf\_test\_por, por\_dataset)

y\_hat\_por\_train\_scored <- as\_tibble(cbind(por\_dataset, y\_hat\_por\_train))

y\_hat\_por\_train\_rmse <- yardstick::rmse(y\_hat\_por\_train\_scored, truth=por\_dataset\_output, estimate=y\_hat\_por\_train)

y\_hat\_por\_train\_rmse # 1.19

# test RF

## mat

y\_hat\_mat\_test <- predict(rf\_test\_mat, mat\_validation)

y\_hat\_mat\_test\_scored <- as\_tibble(cbind(mat\_validation, y\_hat\_mat\_test))

y\_hat\_mat\_test\_rmse <- yardstick::rmse(y\_hat\_mat\_test\_scored, truth=mat\_validation\_output, estimate=y\_hat\_mat\_test)

y\_hat\_mat\_test\_rmse # 4.19

## por

y\_hat\_por\_test <- predict(rf\_test\_por, por\_validation)

y\_hat\_por\_test\_scored <- as\_tibble(cbind(por\_validation, y\_hat\_por\_test))

y\_hat\_por\_test\_rmse <- yardstick::rmse(y\_hat\_por\_test\_scored, truth=por\_validation\_output, estimate=y\_hat\_por\_test)

y\_hat\_por\_test\_rmse # 2.76

# conclusion

# ===<Linear Regression>====

## <Train>

## mat por

## RMSE 3.730919 2.110835

## R2 0.3454868 0.4543227

## <Test>

## mat por

## RMSE 3.341797 2.00442

## R2 0.5227731 0.5901086

# ========<lasso>==========

## <Train>

## mat por

## RMSE 4.0240604 2.2005416

## R2 0.2673442 0.4270536

## <Test>

## mat por

## RMSE 4.7071885 2.8858860

## R2 0.1065235 0.1598111

# =====<Random Forest>=====

## <Train>

## mat por

## RMSE 3.936823 2.345072

## R2 0.2973943 0.3521341

## <Test>

## mat por

## RMSE 4.417505 2.76248

## R2 0.2494508 0.2834576

# As shown in the table, por datasets tend to fit better into the model.

#======================================================================================

# Third Part

#======================================================================================

# Import and prepare the bank marketing dataset.

bank = read.table("bank.csv",sep=";",header=TRUE, stringsAsFactors = T)

# Exclude column "duration" since our intention is to have a realistic predictive model.

bank1 <- bank[,-12]

str(bank1)

# set "yes" as the positive class

bank1$y <- relevel(bank1$y, ref = "yes")

levels(bank1$y)

set.seed(1)

# Create a list of 70% of the rows in the bank1 dataset.

bank1\_training\_sample <- createDataPartition(bank1$y,p = 0.7, list = FALSE)

# Use the 70% of data for training and testing the models.

bank1\_dataset<- bank1[bank1\_training\_sample,]

# Use the 30% of the data to validate the model.

bank1\_validation <- bank1[-bank1\_training\_sample,]

# split input and output

output <- bank1\_dataset[16]

input <- bank1\_dataset[1:15]

## Training

bank1\_dataset\_output <- bank1\_dataset[16]

bank1\_dataset\_input <- bank1\_dataset[1:15]

## Validation

bank1\_validation\_output <- bank1\_validation[16]

bank1\_validation\_input <- bank1\_validation[1:15]

# Run algorithms using 10-fold cross validation

# Logistic Regression=============================================================

control <- trainControl(method="repeatedcv", number=10, repeats = 3)

metric <- "Accuracy"

set.seed(1)

bank1\_fit\_glm <- train(y~., data=bank1\_dataset, method="glm", metric=metric, trControl=control, family="binomial")

summary(bank1\_fit\_glm)

confusionMatrix(bank1\_fit\_glm)

bank1\_pred <- predict(bank1\_fit\_glm, bank1\_validation[,-16])

head(data.frame(original = bank1\_validation\_output, pred = bank1\_pred))

table(bank1\_dataset\_output)

# From the table, we can observe that "yes" is much less compared to "no".

# This may causes class imbalance which results to bias.

# Since we are interested on the clients status of subscription after the phone call,

# we need to revise the model because not subscribed status is dominated over subscribed status.

prop.table(table(bank1\_dataset\_output))

# Proportion rate of class shows 12% for subscription to term deposit and 88% for non subscription

# to term deposit after the call.

par(mfrow=c(1, 1))

barplot(prop.table(table(bank1\_dataset\_output)),

ylim = c(0, 0.9),

main = "Class Distribution")

# To make the class balanced, we will be using 3 techniques to balance the class

library(ROSE)

set.seed(1)

# Under sampling

undersampling <- ovun.sample(y~., data=bank1\_dataset, method = "under",N = 720 ,seed = 1)$data

table(undersampling$y)

# Oversampling

oversampling <- ovun.sample(y~., data=bank1\_dataset, method = "over",N = 5600,seed = 1)$data

table(oversampling$y)

# Bothsampling (Over & Under)

bothsampling <- ovun.sample(y~., data=bank1\_dataset, method = "both",N = 3165, p=.5 ,seed = 1)$data

table(bothsampling$y)

# With 3 techniques, compute the model using each data and evaluate its accuracy.

# Build decision tree models

library(rpart)

set.seed(1)

tree.over <- rpart(y~., data=oversampling)

tree.under <- rpart(y~., data=undersampling)

tree.both <- rpart(y~., data=bothsampling)

pred.tree.over <- predict(tree.over, newdata = bank1\_dataset)

pred.tree.under <- predict(tree.under, newdata = bank1\_dataset)

pred.tree.both <- predict(tree.both, newdata = bank1\_dataset)

#AUC Undersampling

roc.curve(bank1\_dataset$y, pred.tree.under[,2])

# Area under the curve (AUC): 0.755

#AUC Oversampling

roc.curve(bank1\_dataset$y, pred.tree.over[,2])

# Area under the curve (AUC): 0.754

#AUC Bothsampling

roc.curve(bank1\_dataset$y, pred.tree.both[,2])

# Area under the curve (AUC): 0.772

# ROC of Bothsampling method shows the highest AUC score

# Use Bothsampling method as it gives highest AUC score.

set.seed(1)

bank1\_fit\_glm\_bothsampling <- train(y~., data=bothsampling, method="glm", metric=metric, trControl=control, family="binomial")

bank1\_fit\_glm\_bothsampling\_pred <- predict(bank1\_fit\_glm\_bothsampling, bank1\_validation)

# glm variable importance

varImp(bank1\_fit\_glm\_bothsampling,scale=FALSE)

confusionMatrix(bank1\_fit\_glm\_bothsampling\_pred, bank1\_validation$y,positive="yes")

# Random Forest

control1 <- trainControl(method="cv", number=10)

set.seed(1)

bank1\_fit\_rf <- train(y~., data=bank1\_dataset, method="rf", metric=metric, trControl=control1)

bank1\_fit\_rf

# It shows that mtry of 21 has the highest accuracy for bank1\_fit\_rf.

summary(bank1\_fit\_rf)

confusionMatrix(bank1\_fit\_rf)

bank1\_pred\_rf <- predict(bank1\_fit\_rf, bank1\_validation[,-16])

bank1\_pred\_rf

head(data.frame(original = bank1\_validation\_output, pred = bank1\_pred\_rf))

bank1\_pred\_rf %>% confusionMatrix(bank1\_validation$y)

# Using Bothsampling method,

set.seed(1)

bank1\_fit\_rf\_bothsampling <- train(y~., data=bothsampling, method="rf", metric=metric, trControl=control1)

bank1\_fit\_rf\_bothsampling

#Random Forest variable importance

varImp(bank1\_fit\_rf\_bothsampling,scale=FALSE)

summary(bank1\_fit\_rf\_bothsampling)

confusionMatrix(bank1\_fit\_rf\_bothsampling)

bank1\_pred\_rf\_bothsampling <- predict(bank1\_fit\_rf\_bothsampling, bank1\_validation[,-16])

bank1\_pred\_rf\_bothsampling

head(data.frame(original = bank1\_validation\_output, pred = bank1\_pred\_rf\_bothsampling))

bank1\_pred\_rf\_bothsampling %>% confusionMatrix(bank1\_validation$y)

# To conclude, as we look at the accuracy rate of both Logistic and Random

# Forest model, Random Forest model seemed to show better accuracy as it

# resulted higher accuracy rate. However, since our interest was on correctly

# identifying the clients who would subscribe to a term deposit, logistic

# regression model would be preferred as it showed 63% of sensitivity rate

# compared to Random Forest model which gave 35%. Furthermore, we should keep in

# mind that accuracy is not the only evaluation to consider when evaluating the

# model. One should also consider the flexibility and robustness of the model

# when facing new datasets.