# Cost Prediction using stacking of random Forest,Gradient Boosting , XGBoost and Support Vector Machine in R.

if (!require("dplyr")) install.packages("dplyr")

if (!require("ggplot2")) install.packages("ggplot2")

if (!require("readxl")) install.packages("readxl")

if (!require("mice")) install.packages("mice")

if (!require("VIM")) install.packages("VIM")

if (!require("missForest")) install.packages("missForest")

if (!require("car")) install.packages("car")

if (!require("magrittr")) install.packages("magrittr")

if (!require("skimr")) install.packages("skimr")

if (!require("mctest")) install.packages("mctest")

library(dplyr)

library(ggplot2)

library(readxl)

library(mice)

library(VIM)

library(missForest)

library(caret)

library(caretEnsemble)

library(Amelia)

library(GGally)

library(car)

library(magrittr)

library(skimr)

library(mctest)

#set parallel backend

library(parallel)

library(parallelMap)

parallelStartSocket(cpus = detectCores())

mydata<- read\_excel("Sample Data.xlsx",sheet = "A")

mydata$`Feature A`<-recode(mydata$`Feature A`,"'-' =sample b obs ;NA= sample c obs ")

mice\_plot <- aggr(mydata, col=c('navyblue','yellow'),

                  numbers=TRUE, sortVars=TRUE,

                  labels=names(mydata), cex.axis=.45,

                  gap=3, ylab=c("Missing data","Pattern"),bars=F)

colnames\_num<-c("a\_num"," b\_num "," c\_num "," d\_num "," e\_num "," f\_num ")

mydata[,colnames\_num] %<>% lapply(function(x) as.numeric(as.character(x)))

colnames\_fact=c("Feature B","Unit Country")

mydata[,colnames\_fact] %<>% lapply(function(x) as.factor(x))

mydata\_num<-dplyr::select\_if(mydata, is.numeric)

mydata\_fact<-dplyr::select\_if(mydata, is.factor)

ggpairs(mydata\_num)

#Box Cox Transformation as the data is highly skeweed.

fun\_boxcox<- function(var1)

{

  BoxCox(var1, lambda=0)

}

set.seed(5432)

boxcox\_output=as.data.frame(sapply(mydata\_num,fun\_boxcox))

fun\_invboxcox<- function(var2)

{

  InvBoxCox(var2, lambda=0)

}

set.seed(5432)

Invboxcox\_output=as.data.frame(sapply(boxcox\_output,fun\_invboxcox))

mydata\_cat<- mydata[c("Feature B","Unit Country")]

ggpairs(boxcox\_output)

mydata<- cbind(mydata\_cat,boxcox\_output)

seed <- 543654

set.seed(seed)

intrain<- createDataPartition(mydata$price ,p=0.8,list = FALSE)

trainData <- mydata[intrain,]

testData <- mydata[-intrain,]

########### Regression #######################

trControl = trainControl("repeatedcv", number = 10,repeats=3)

set.seed(seed)

modelfit <- train(price ~., data = trainData, method = "lm",tuneLength = 5)

summary(modelfit)

pred=predict(modelfit,newdata=testData)

#Predicted Values

Invboxcox\_output=as.data.frame(sapply(pred,fun\_invboxcox))

row.names(Invboxcox\_output) <- NULL

names(Invboxcox\_output)<-"Predicted Value"

predicted\_value=Invboxcox\_output$`Predicted Value`

#Test Data Price

Invboxcox\_price=as.data.frame(sapply(testData$price,fun\_invboxcox))

row.names(Invboxcox\_output) <- NULL

names(Invboxcox\_price)<-"Test Price"

test\_price=Invboxcox\_price$`Test Price`

plot(predicted\_value,type="b")

lines(test\_price,type = "l",col="blue")

ggplot(testData, aes(1:length(predicted\_value))) +

  geom\_line(aes(y = predicted\_value, colour = "predicted\_value")) +

  geom\_line(aes(y = test\_price, colour = "test\_price"))+labs(title="Predicted vs Actual Price", x="No of Obs.", y="Price")

# Stepwise Regression

library(MASS)

set.seed(seed)

mymodel\_stepwise= lm(price~.,data=trainData)

step.model <- stepAIC(mymodel\_stepwise, direction = "both",

                      trace = FALSE)

summary(step.model)

# Testing for other regressions.

# Using caret package

# Setup a grid range of lambda values:

lambda <- 10^seq(-3, 3, length = 100)

alpha <- seq(0,1,length=10)

# Note that for alpha = 0 ( for ridge), 1 ( for lasso) and we will use above alpha function to

# determine optimum alpha level for elastic regression.

# Elastic net regression:

# Build the model

set.seed(seed)

elastic <- train(

  price ~., data = trainData, method = "glmnet",

  trControl = trainControl("repeatedcv", number = 10,repeats=3), #tuneLength=10- It is better option

  tuneGrid = expand.grid(alpha = alpha, lambda = lambda)  # can also be used

)

# Model coefficients

coef(elastic$finalModel, elastic$bestTune$lambda)

# Make predictions

predictions\_elasticnet <- elastic %>% predict(testData)

# Model prediction performance

data.frame(

  RMSE = RMSE(predictions\_elasticnet, testData$price),

  Rsquare = R2(predictions\_elasticnet, testData$price)

)

trcontrol = trainControl(

  method = "cv",

  number = 5,

  allowParallel = TRUE,

  verboseIter = FALSE,

  returnData = FALSE

)

xgb\_Grid <- expand.grid(nrounds = c(100,200),

                       max\_depth = c(10, 15, 20, 25),

                       colsample\_bytree = seq(0.5, 0.9, length.out = 5),

                       eta = 0.1,

                       gamma=0,

                       min\_child\_weight = 1,

                       subsample = 1

)

set.seed(seed)

xgbtree\_model <- train(price~., data=trainData, method="xgbTree", trControl=trcontrol,tuneGrid=xgb\_Grid)

# Make predictions

predictions\_xgbtree <- xgbtree\_model  %>% predict(testData)

# Model prediction performance

data.frame(

  RMSE = RMSE(predictions\_xgbtree, testData$price),

  Rsquare = R2(predictions\_xgbtree, testData$price)

)

gbm\_grid <- expand.grid(n.trees=c(10,20,50,100,500,1000),

                    shrinkage=c(0.01,0.05,0.1,0.5),

                    n.minobsinnode = c(3,5,10),

                    interaction.depth=c(1,5,10))

set.seed(seed)

gbm\_model <- train(price~., data=trainData, method="gbm", trControl=trcontrol,tuneGrid=gbm\_grid)

# Make predictions

predictions\_gbm<- gbm\_model %>% predict(testData)

# Model prediction performance

data.frame(

  RMSE = RMSE(predictions\_gbm, testData$price),

  Rsquare = R2(predictions\_gbm, testData$price)

)

set.seed(seed)

rf\_grid <- expand.grid(.mtry=c(1:ncol(trainData)))

rf\_model <- train(price~., data=trainData, method="rf", tuneGrid=rf\_grid, trControl=trcontrol)

# Make predictions

predictions\_rf<- rf\_model %>% predict(testData)

# Model prediction performance

data.frame(

  RMSE = RMSE(predictions\_rf, testData$price),

  Rsquare = R2(predictions\_rf, testData$price)

)

set.seed(seed)

svm\_Radial <- train(price ~., data = trainData, method = "svmRadial",

                           trControl=trcontrol,

                           tuneLength = 10)

# Make predictions

predictions\_svm<- svm\_Radial %>% predict(testData)

# Model prediction performance

data.frame(

  RMSE = RMSE(predictions\_svm, testData$price),

  Rsquare = R2(predictions\_svm, testData$price)

)

results <- resamples(list(`Random Forest` =rf\_model, SVM = svm\_Radial,GBM=gbm\_model,XGBTree=xgbtree\_model))

(modelCor(results))

ggcorrplot::ggcorrplot(modelCor(results),method = "circle",lab = T)

# Stacking the models- Method 1

predDF <- data.frame(predictions\_xgbtree, predictions\_gbm, predictions\_rf, price = testData$price, stringsAsFactors = F)

modelStack <- train(price ~ ., data = predDF, method = "gbm")

combPred\_stack\_manual <- predict(modelStack, predDF)

# Model prediction performance

data.frame(

  RMSE = RMSE(combPred\_stack\_manual, testData$price),

  Rsquare = R2(combPred\_stack\_manual, testData$price)

)

# Stacking the models- Method 2

0.2184544

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

algorithmList <- c('rf', 'gbm', 'xgbTree', 'svmRadial')

set.seed(seed)

models <- caretList(price~., data=trainData, trControl=control, methodList=algorithmList)

results <- resamples(models)

summary(results)

dotplot(results)

# correlation between results

modelCor(results)

splom(results)

# stack using lm

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

set.seed(seed)

stack.glm <- caretStack(models, method="gbm", metric="RMSE", trControl=stackControl)

combPred\_stack\_auto <- predict(stack.glm, testData)

print(combPred)

data.frame(

  RMSE = RMSE(combPred\_stack\_auto, testData$price),

  Rsquare = R2(combPred\_stack\_auto, testData$price))

# Predict the output on new dataset.

test\_p<-read\_excel("test\_p.xlsx")

#

test\_p<- test\_p%>%dplyr::rename(part\_wt=`Part Weight (in kg)`)

mydata<-test\_p

colnames\_num<-c("part\_wt","DIM\_C\_METRIC","DIM\_D\_METRIC","DIM\_E\_METRIC","DIM\_F\_METRIC","DIM\_G\_METRIC")

colnames\_fact=c("Feature B","Unit Country")

mydata[,colnames\_num] %<>% lapply(function(x) as.numeric(as.character(x)))

mydata[,colnames\_fact] %<>% lapply(function(x) as.factor(x))

mydata\_num<-dplyr::select\_if(mydata, is.numeric)

mydata\_fact<-dplyr::select\_if(mydata, is.factor)

set.seed(5432)

boxcox\_output=t(as.data.frame(sapply(mydata\_num,fun\_boxcox)))

rownames(boxcox\_output)<-NULL

testdata<- cbind(mydata\_fact,boxcox\_output)

predictions\_gbm<- gbm\_model %>% predict(testData)

predictions\_xgbtree <- xgbtree\_model %>% predict(testData)

predictions\_rf<- rf\_model %>% predict(testData)

predictions\_gbm\_1<- gbm\_model %>% predict(testdata)

predictions\_xgbtree\_1 <- xgbtree\_model %>% predict(testdata)

predictions\_rf\_1<- rf\_model %>% predict(testdata)

stack\_input<-cbind("predictions\_gbm"=predictions\_gbm\_1,"predictions\_xgbtree"=predictions\_xgbtree\_1,"predictions\_rf"=predictions\_rf\_1)

predDF <- data.frame(predictions\_xgbtree, predictions\_gbm, predictions\_rf, price = testData$price, stringsAsFactors = F)

modelStack <- train(price ~ ., data = predDF, method = "gbm")

combPred\_stack\_manual <- exp(predict(modelStack, stack\_input))

#

# model\_gbm<- train(trainData[,predictors\_top],trainData[,outcomeName],method='gbm',trControl=fitControl,tuneLength=3)

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

algorithmList <- c('rf', 'gbm', 'xgbTree')

set.seed(seed)

models <- caretList(price~., data=trainData, trControl=control, methodList=algorithmList)

# stack using gbm

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

set.seed(seed)

stack.gbm <- caretStack(models, method="gbm", metric="RMSE", trControl=stackControl)

combPred\_stack\_auto <- exp(predict(stack.gbm, testdata))