**Packaging Made Faster: BMW Automated Logistics Value Chain**

Kenny Barza

Karen Bassil

Emily El Hajj

Jason Eid

Presented to: Dr. Imad Bou-Hamad

**Appendix F – R Code**

data <- read.csv("1\_packaging\_planning\_data - Copy.csv")

library(magrittr)

library(tidyverse)

library(stats)

library(dplyr)

library(MASS)

library(car)

library(PresenceAbsence)

library(rpart)

library(rpart.plot)

data <- filter(data, product\_x\_dim !=0 & product\_y\_dim!=0 & product\_zdim!=0 & product\_weight!=0 & products\_per\_packaging!=0 & packaging\_x\_dim!=0 & packaging\_y\_dim!=0 & packaging\_z\_dim!=0 & packaging\_weight!=0 & packaging\_load\_capacity!=0 & packagings\_per\_load\_unit!=0 & load\_unit\_x\_dim!=0 & load\_unit\_y\_dim!=0 & load\_unit\_z\_dim!=0 & load\_unit\_weight!=0)

attach(data)

#product\_per\_packaging

par(mfrow=c(1,2))

hist(products\_per\_packaging)

boxplot(products\_per\_packaging)

summary(products\_per\_packaging)

#product\_x\_dim

par(mfrow=c(1,2))

hist(product\_x\_dim)

boxplot(product\_x\_dim)

summary(product\_x\_dim)

#product\_zdim

par(mfrow=c(1,2))

hist(product\_zdim)

boxplot(product\_zdim)

summary(product\_zdim)

#product\_y\_dim

par(mfrow=c(1,2))

hist(product\_y\_dim)

boxplot(product\_y\_dim)

summary(product\_y\_dim)

#product\_weight

par(mfrow=c(1,2))

hist(product\_weight)

boxplot(product\_weight)

summary(product\_weight)

#packaging\_x\_dim

par(mfrow=c(1,2))

hist(packaging\_x\_dim)

boxplot(packaging\_x\_dim)

summary(packaging\_x\_dim)

#packaging\_y\_dim

par(mfrow=c(1,2))

hist(packaging\_y\_dim)

boxplot(packaging\_y\_dim)

summary(packaging\_y\_dim)

#packaging\_z\_dim

par(mfrow=c(1,2))

hist(packaging\_z\_dim)

boxplot(packaging\_z\_dim)

summary(packaging\_z\_dim)

#product\_quality\_index

par(mfrow=c(1,2))

hist(product\_quality\_index, xlim = c(0,800),breaks=100)

boxplot(product\_quality\_index)

summary(product\_quality\_index)

# relation between products feature(s) and packaging\_x\_dim

plot(product\_weight~packaging\_x\_dim,col='blue',main='Scatterplot for product weight and package x dimensions')

cor.test(product\_weight,packaging\_x\_dim)

plot(product\_x\_dim~packaging\_x\_dim,col='blue',main='Scatterplot for product x dimension and package x dimensions')

cor.test(product\_x\_dim,packaging\_x\_dim)

plot(product\_y\_dim~packaging\_x\_dim,col='blue',main='Scatterplot for product y dimension and package x dimensions')

cor.test(product\_y\_dim,packaging\_x\_dim)

plot(product\_zdim~packaging\_x\_dim,col='blue',main='Scatterplot for product z dimension and package x dimensions')

cor.test(product\_zdim,packaging\_x\_dim)

plot(product\_quality\_index~packaging\_x\_dim,col='blue',main='Scatterplot for product quality index and package x dimensions')

cor.test(product\_quality\_index,packaging\_x\_dim)

# relation between products feature(s) and packaging\_y\_dim

plot(product\_weight~packaging\_y\_dim,col='blue',main='Scatterplot for product weight and package y dimensions')

cor.test(product\_weight,packaging\_y\_dim)

plot(product\_x\_dim~packaging\_x\_dim,col='blue',main='Scatterplot for product x dimension and package y dimensions')

cor.test(product\_x\_dim,packaging\_y\_dim)

plot(product\_y\_dim~packaging\_y\_dim,col='blue',main='Scatterplot for product y dimension and package y dimensions')

cor.test(product\_y\_dim,packaging\_y\_dim)

plot(product\_zdim~packaging\_y\_dim,col='blue',main='Scatterplot for product z dimension and package y dimensions')

cor.test(product\_zdim,packaging\_y\_dim)

plot(product\_quality\_index~packaging\_y\_dim,col='blue',main='Scatterplot for product quality index and package y dimensions')

cor.test(product\_quality\_index,packaging\_y\_dim)

# relation between products feature(s) and packaging\_z\_dim

plot(product\_weight~packaging\_z\_dim,col='blue',main='Scatterplot for product weight and package z dimensions')

cor.test(product\_weight,packaging\_z\_dim)

plot(product\_x\_dim~packaging\_z\_dim,col='blue',main='Scatterplot for product x dimension and package z dimensions')

cor.test(product\_x\_dim,packaging\_z\_dim)

plot(product\_y\_dim~packaging\_z\_dim,col='blue',main='Scatterplot for product y dimension and package z dimensions')

cor.test(product\_y\_dim,packaging\_z\_dim)

plot(product\_zdim~packaging\_z\_dim,col='blue',main='Scatterplot for product z dimension and package z dimensions')

cor.test(product\_zdim,packaging\_z\_dim)

plot(product\_quality\_index~packaging\_z\_dim,col='blue',main='Scatterplot for product quality index and package z dimensions')

cor.test(product\_quality\_index,packaging\_z\_dim)

# relation between products feature(s) and packaging\_weight

plot(product\_weight~packaging\_weight,col='blue',main='Scatterplot for product weight and package weight')

cor.test(product\_weight,packaging\_weight)

plot(product\_x\_dim~packaging\_weight,col='blue',main='Scatterplot for product x dimension and package weight')

cor.test(product\_x\_dim,packaging\_weight)

plot(product\_y\_dim~packaging\_weight,col='blue',main='Scatterplot for product y dimension and package weight')

cor.test(product\_y\_dim,packaging\_weight)

plot(product\_zdim~packaging\_weight,col='blue',main='Scatterplot for product z dimension and package weight')

cor.test(product\_zdim,packaging\_weight)

plot(product\_quality\_index~packaging\_weight,col='blue',main='Scatterplot for product quality index and package weight')

cor.test(product\_quality\_index,packaging\_weight)

# relation between products feature(s) and products\_per\_package

plot(product\_weight~products\_per\_packaging,col='blue',main='Scatterplot for product weight and products per packaging')

cor.test(product\_weight,products\_per\_packaging)

plot(product\_x\_dim~products\_per\_packaging,col='blue',main='Scatterplot for product x dimension and products per packaging')

cor.test(product\_x\_dim,products\_per\_packaging)

plot(product\_y\_dim~products\_per\_packaging,col='blue',main='Scatterplot for product y dimension and products per packaging')

cor.test(product\_y\_dim,products\_per\_packaging)

plot(product\_zdim~products\_per\_packaging,col='blue',main='Scatterplot for product z dimension and products per packaging')

cor.test(product\_zdim,products\_per\_packaging)

plot(product\_quality\_index~products\_per\_packaging,col='blue',main='Scatterplot for product quality index and products per packaging')

cor.test(product\_quality\_index,products\_per\_packaging)

# relation between products feature(s) and packaging\_load\_capacity

plot(product\_weight~packaging\_load\_capacity,col='blue',main='Scatterplot for product weight and packaging load capacity')

cor.test(product\_weight,packaging\_load\_capacity)

plot(product\_x\_dim~packaging\_load\_capacity,col='blue',main='Scatterplot for product x dimension and packaging load capacity')

cor.test(product\_x\_dim,packaging\_load\_capacity)

plot(product\_y\_dim~packaging\_load\_capacity,col='blue',main='Scatterplot for product y dimension and packaging load capacity')

cor.test(product\_y\_dim,packaging\_load\_capacity)

plot(product\_zdim~packaging\_load\_capacity,col='blue',main='Scatterplot for product z dimension and packaging load capacity')

cor.test(product\_zdim,packaging\_load\_capacity)

plot(product\_quality\_index~packaging\_load\_capacity,col='blue',main='Scatterplot for product quality index and packaging load capacity')

cor.test(product\_quality\_index,packaging\_load\_capacity)

#relation between categorical product features (product\_is\_esp & product\_is\_dangerous\_good) and packaging\_x\_dim

aov(packaging\_x\_dim~product\_is\_esp)

boxplot(packaging\_x\_dim~product\_is\_esp)

aov(packaging\_x\_dim~product\_is\_dangerous\_good)

boxplot(packaging\_x\_dim~product\_is\_dangerous\_good)

#relation between categorical product features (product\_is\_esp & product\_is\_dangerous\_good) and packaging\_y\_dim

aov(packaging\_y\_dim~product\_is\_esp)

boxplot(packaging\_y\_dim~product\_is\_esp)

aov(packaging\_y\_dim~product\_is\_dangerous\_good)

boxplot(packaging\_y\_dim~product\_is\_dangerous\_good)

#relation between categorical product features (product\_is\_esp & product\_is\_dangerous\_good) and packaging\_z\_dim

aov(packaging\_z\_dim~product\_is\_esp)

boxplot(packaging\_z\_dim~product\_is\_esp)

aov(packaging\_z\_dim~product\_is\_dangerous\_good)

boxplot(packaging\_z\_dim~product\_is\_dangerous\_good)

#relation between categorical product features (product\_is\_esp & product\_is\_dangerous\_good) and packaging\_weight

aov(packaging\_weight~ product\_is\_esp)

boxplot(packaging\_weight ~ product\_is\_esp)

aov(packaging\_weight ~ product\_is\_dangerous\_good)

boxplot(packaging\_weight ~ product\_is\_dangerous\_good)

#relation between categorical product features (product\_is\_esp & product\_is\_dangerous\_good) and products\_per\_packaging

aov(products\_per\_packaging ~ product\_is\_esp)

boxplot(products\_per\_packaging ~ product\_is\_esp)

aov(products\_per\_packaging ~ product\_is\_dangerous\_good)

boxplot(products\_per\_packaging ~ product\_is\_dangerous\_good)

#relation between categorical product features (product\_is\_esp & product\_is\_dangerous\_good) and packaging\_load\_capacity

aov(packaging\_load\_capacity ~ product\_is\_esp)

boxplot(packaging\_load\_capacity ~ product\_is\_esp)

aov(packaging\_load\_capacity ~ product\_is\_dangerous\_good)

boxplot(packaging\_load\_capacity ~ product\_is\_dangerous\_good)

par(mfrow=c(1,5))

boxplot(df$product\_zdim~df$packaging\_is\_special,col="cadetblue4", xlab = ' Packaging is special', main= "Boxplot") #1

boxplot(df$product\_y\_dim~df$packaging\_is\_special,col="cadetblue4", xlab = ' Packaging is special', main= "Boxplot")#1

boxplot(df$product\_x\_dim~df$packaging\_is\_special,col="cadetblue4", xlab = ' Packaging is special', main="Boxplots")#1

boxplot(df$product\_weight~df$packaging\_is\_special,col="cadetblue4", xlab = ' Packaging is special', main= 'Boxplot')

boxplot(df$product\_quality\_index~df$packaging\_is\_special,col="cadetblue4", xlab= ' Packaging is special', main='Boxplot')

###### relation between product\_per\_packaging and packaging special #1

par(mfrow=c(1,2))

tbl=table(df$packaging\_is\_special,df$product\_is\_esp)

prop\_tbl=prop.table(tbl,2)

barplot(prop\_tbl, col=c("cadetblue4","lightgoldenrod1"), beside=T, legend.text = T, main = "Clustered Barchart", ylab ='Packaging is special', xlab = 'product is esp' )

###### relation between product\_per\_packaging and packaging special

tbl=table(df$packaging\_is\_special,df$product\_is\_dangerous\_good)

prop\_tbl=prop.table(tbl,2)

barplot(prop\_tbl,col=c("cadetblue4","lightgoldenrod1"),beside=T, legend.text = T, main = "Clustered BarChart", ylab = 'Packaging is special', xlab = "Product is dangerous")

###### relation between product\_per\_packaging and packaging special

par(mfrow=c(1,5))

boxplot(df$product\_zdim~df$packaging\_is\_oneway,col="cadetblue4", xlab=' Packaging is oneway', main = "Boxplot")

boxplot(df$product\_y\_dim~df$packaging\_is\_oneway,col="cadetblue4", xlab = ' Packaging is oneway', main = "Boxplot")

boxplot(df$product\_x\_dim~df$packaging\_is\_oneway,col="cadetblue4", xlab = ' Packaging is oneway', main = "Boxplot")

boxplot(df$product\_weight~df$packaging\_is\_oneway,col="cadetblue4", xlab = ' Packaging is oneway', main = "Boxplot")

boxplot(df$product\_quality\_index~df$packaging\_is\_oneway,col="cadetblue4", xlab = ' Packaging is oneway', main = "Boxplot")

###### relation between product\_per\_packaging and packaging special

par(mfrow=c(1,2))

tbl=table(df$packaging\_is\_oneway,df$product\_is\_esp)

prop\_tbl=prop.table(tbl,2)

barplot(prop\_tbl,col=c("cadetblue4","lightgoldenrod1"),beside=T, legend.text = T, main = "Clustered BarChart", ylab = 'Packaging is oneway', xlab = "Product is esp" )

###### relation between product\_per\_packaging and packaging special

tbl=table(df$packaging\_is\_oneway,df$product\_is\_dangerous\_good)

prop\_tbl=prop.table(tbl,2)

barplot(prop\_tbl,col=c("cadetblue4","lightgoldenrod1"),beside=T, legend.text = T, main = "Clustered BarChart", ylab = 'Packaging is oneway', xlab = "Product is dangerous")

#Qualitative + Quantitative (Boxplot and ANOVA)

#Load Unit is One Way

par(mfrow=c(1,6))

boxplot(packaging\_x\_dim~load\_unit\_is\_oneway, main = 'Boxplot', xlab = 'Load Unit is One Way', col='cadetblue4')

summary(aov(packaging\_x\_dim~load\_unit\_is\_oneway))

boxplot(packaging\_y\_dim~load\_unit\_is\_oneway, main = 'Boxplot', xlab = 'Load Unit is One Way', col='cadetblue4')

summary(aov(packaging\_y\_dim~load\_unit\_is\_oneway))

boxplot(packaging\_z\_dim~load\_unit\_is\_oneway, main = 'Boxplot', xlab = 'Load Unit is One Way', col='cadetblue4')

summary(aov(packaging\_z\_dim~load\_unit\_is\_oneway))

boxplot(packaging\_weight~load\_unit\_is\_oneway, main = 'Boxplot', xlab = 'Load Unit is One Way', col='cadetblue4')

summary(aov(packaging\_weight~load\_unit\_is\_oneway))

boxplot(packaging\_load\_capacity~load\_unit\_is\_oneway, main = 'Boxplot', xlab = 'Load Unit is One Way', col='cadetblue4')

summary(aov(packaging\_load\_capacity~load\_unit\_is\_oneway))

boxplot(products\_per\_packaging~load\_unit\_is\_oneway, main = 'Boxplot', xlab = 'Load Unit is One Way', col='cadetblue4')

summary(aov(products\_per\_packaging~load\_unit\_is\_oneway))

#Load Unit is Special

par(mfrow=c(1,6))

boxplot(packaging\_x\_dim~load\_unit\_is\_special, main = 'Boxplot', xlab = 'Load Unit is Special', col='cadetblue4')

summary(aov(packaging\_x\_dim~load\_unit\_is\_special))

boxplot(packaging\_y\_dim~load\_unit\_is\_special, main = 'Boxplot', xlab = 'Load Unit is Special', col='cadetblue4')

summary(aov(packaging\_y\_dim~load\_unit\_is\_special))

boxplot(packaging\_z\_dim~load\_unit\_is\_special, main = 'Boxplot', xlab = 'Load Unit is Special', col='cadetblue4')

summary(aov(packaging\_z\_dim~load\_unit\_is\_special))

boxplot(packaging\_weight~load\_unit\_is\_special, main = 'Boxplot', xlab = 'Load Unit is Special', col='cadetblue4')

summary(aov(packaging\_weight~load\_unit\_is\_special))

boxplot(packaging\_load\_capacity~load\_unit\_is\_special, main = 'Boxplot', xlab = 'Load Unit is Special', col='cadetblue4')

summary(aov(packaging\_load\_capacity~load\_unit\_is\_special))

boxplot(products\_per\_packaging~load\_unit\_is\_special, main = 'Boxplot', xlab = 'Load Unit is Special', col='cadetblue4')

summary(aov(products\_per\_packaging~load\_unit\_is\_special))

#Qualitative + Qualitative (Chi-square + Clustered Barcharts)

#Load Unit is One Way

par(mfrow=c(2,2))

tbl11 = table(load\_unit\_is\_oneway, packaging\_is\_oneway)

chisq.test(tbl11)

prop\_tbl11 = prop.table(tbl11, 2)

barplot(prop\_tbl11, col=c('cadetblue4','lightgoldenrod1'), main='Clustered Barchart', xlab='packaging is oneway', ylab = 'load unit is oneway', beside=T, legend.text = T)

tbl11 #include in the report along the barchart to clearly see the numbers

tbl12 = table(load\_unit\_is\_oneway, packaging\_is\_special)

chisq.test(tbl12)

prop\_tbl12 = prop.table(tbl12, 2)

barplot(prop\_tbl12, col=c('cadetblue4','lightgoldenrod1'), main='Clustered Barchart', xlab='packaging is special', ylab='load unit is oneway', beside=T, legend.text = T)

tbl12 #include in the report along the barchart to clearly see the numbers

#Load Unit is Special

tbl21 = table(load\_unit\_is\_special, packaging\_is\_oneway)

chisq.test(tbl21)

prop\_tbl21 = prop.table(tbl21, 2)

barplot(prop\_tbl21, col=c('cadetblue4','lightgoldenrod1'), main='Clustered Barchart', xlab='packaging is oneway', ylab = 'load unit is Special', beside=T, legend.text = T)

tbl21 #include in the report along the barchart to clearly see the numbers

tbl22 = table(load\_unit\_is\_special, packaging\_is\_special)

chisq.test(tbl22)

prop\_tbl22 = prop.table(tbl22, 2)

barplot(prop\_tbl22, col=c('cadetblue4','lightgoldenrod1'), main='Clustered Barchart', xlab='packaging is special', ylab='load unit is Special', beside=T, legend.text = T)

tbl22 #include in the report along the barchart to clearly see the numbers

par(mfrow=c(1,1))

#Create training and validation (testing) data

set.seed(100)

split=sample(1:2, nrow(data), replace = TRUE, prob=c(0.7, 0.3))

Train=data[split==1,]

valid=data[split==2,]

##Packaging X dim##

#Linear model 1 - all predictors

Linear\_model1=lm(packaging\_x\_dim ~ product\_x\_dim + product\_is\_esp +product\_weight +

product\_quality\_index + product\_y\_dim +product\_zdim + product\_is\_dangerous\_good, data=train)

summary(Linear\_model1)

pred=predict(Linear\_model1,val)

perform\_linearModel1=acc\_error(val$packaging\_x\_dim,pred)

perform\_linearModel1

mean(packaging\_x\_dim)

par(mfrow=c(2,2))

plot(Linear\_model1)

#Linear model 2 - all numerical predictors

Linear\_model2=lm(packaging\_x\_dim ~ product\_x\_dim +product\_weight +

product\_quality\_index + product\_y\_dim +product\_zdim , data=train)

summary(Linear\_model2)

pred=predict(Linear\_model2,val)

perform\_linearModel2=acc\_error(val$packaging\_x\_dim,pred)

perform\_linearModel2

mean(packaging\_x\_dim)

#Linear\_model 3 - log( all numerical predictors)

Linear\_model3= lm(packaging\_x\_dim ~ log(product\_x\_dim) +log(product\_y\_dim)

+ log(product\_weight) +log(product\_quality\_index) + log(product\_zdim), data=train)

summary(Linear\_model3)

pred=predict(Linear\_model3,val)

perform\_linearModel3=acc\_error(val$packaging\_x\_dim,pred)

perform\_linearModel3

mean(packaging\_x\_dim)

#Tree model1 - all predictors - not pruned

tree\_model1 <- rpart(packaging\_x\_dim ~ product\_x\_dim +product\_y\_dim + product\_zdim

+product\_weight + product\_quality\_index+ product\_is\_esp + product\_is\_dangerous\_good,

data=train, control = rpart.control(cp=0.0001))

printcp(tree\_model1)

tree\_model1$cptable

bestcp=tree\_model1$cptable[which.min(tree\_model1$cptable[,"xerror"]),"CP"]

splits=tree\_model1$cptable[which.min(tree\_model1$cptable[,"xerror"]),"nsplit"]

splits

tree.pruned=prune(tree\_model1, cp= bestcp)

rpart.plot(tree.pruned,type=4,extra="Auto")

pred=predict(tree.pruned,val)

perform\_treeModel1=acc\_error(val$packaging\_x\_dim,pred)

perform\_treeModel1

#Tree model2 - all predictors - pruned

tree\_model2\_pruned <- rpart(products\_per\_packaging ~ product\_x\_dim +product\_y\_dim + product\_zdim

+product\_weight + product\_quality\_index+ product\_is\_esp + product\_is\_dangerous\_good,

data=train, control = rpart.control(cp=0.0001))

printcp(tree\_model2\_pruned)

tree.pruned2=prune(tree\_model2\_pruned, cp= 0.0026228036 )

rpart.plot(tree.pruned2,type=4,extra="Auto")

pred=predict(tree\_model2\_pruned,val)

perform\_treeModel2\_pruned=acc\_error(val$products\_per\_packaging,pred)

perform\_treeModel2\_pruned

#Random forest1 - all predictors - ntree=150

Randomtree1 <- randomForest(packaging\_x\_dim ~ product\_x\_dim+product\_y\_dim +product\_zdim +product\_weight+

product\_is\_esp + product\_is\_dangerous\_good + product\_quality\_index,

data=train, mtry=3, ntree= 150, importance=TRUE, na.action=na.omit)

pred = predict(Randomtree1, val)

perform\_randomTree1 = acc\_error(val$packaging\_x\_dim, pred)

perform\_randomTree1

##Packaging\_y\_dim##

#Linear model 1 - all predictors

Linear\_model1=lm(packaging\_y\_dim ~ product\_is\_esp + product\_is\_dangerous\_good + product\_weight

+ product\_x\_dim + product\_y\_dim + product\_zdim + product\_quality\_index, data=train)

summary(Linear\_model1)

pred=predict(Linear\_model1,val)

perform\_linearModel1=acc\_error(val$packaging\_y\_dim,pred)

perform\_linearModel1

mean(packaging\_y\_dim)

#Linear model 2 - eliminate insignificant (product\_is\_dangerous\_good)

Linear\_model2=lm(packaging\_y\_dim ~ product\_is\_esp + product\_weight

+ product\_x\_dim + product\_y\_dim + product\_zdim + product\_quality\_index, data=train)

summary(Linear\_model2)

pred=predict(Linear\_model2,val)

perform\_linearModel2=acc\_error(val$packaging\_y\_dim,pred)

perform\_linearModel2

mean(packaging\_y\_dim)

#Linear model 3 - all numerical predictors

Linear\_model3=lm(packaging\_y\_dim ~ product\_weight

+ product\_x\_dim + product\_y\_dim + product\_zdim + product\_quality\_index, data=train)

summary(Linear\_model3)

pred=predict(Linear\_model3,val)

perform\_linearModel3=acc\_error(val$packaging\_y\_dim,pred)

perform\_linearModel3

#Linear model 4 - log(all numerical predictors)

Linear\_model4=lm(packaging\_y\_dim ~log(product\_weight) + log(product\_x\_dim) + log(product\_y\_dim)

+log(product\_zdim) + log(product\_quality\_index) , data=train)

summary(Linear\_model4)

pred=predict(Linear\_model4,val)

perform\_linearModel4=acc\_error(val$packaging\_y\_dim,pred)

perform\_linearModel4

#Tree model1 - all predictors - not pruned

tree\_model1 <- rpart(packaging\_y\_dim ~ product\_x\_dim +product\_y\_dim + product\_zdim

+product\_weight + product\_quality\_index+ product\_is\_esp + product\_is\_dangerous\_good,

data=train, control = rpart.control(cp=0.0001))

printcp(tree\_model1)

tree\_model1$cptable

bestcp=tree\_model1$cptable[which.min(tree\_model1$cptable[,"xerror"]),"CP"]

splits=tree\_model1$cptable[which.min(tree\_model1$cptable[,"xerror"]),"nsplit"]

splits

tree.pruned=prune(tree\_model1, cp= bestcp)

rpart.plot(tree.pruned,type=4,extra="Auto")

pred=predict(tree.pruned,val)

perform\_treeModel1=acc\_error(val$packaging\_y\_dim,pred)

perform\_treeModel1

#Tree model2 - all predictors - pruned

tree\_model2\_pruned <- rpart(packaging\_y\_dim ~ product\_x\_dim +product\_y\_dim + product\_zdim

+product\_weight + product\_quality\_index+ product\_is\_esp + product\_is\_dangerous\_good,

data=train, control = rpart.control(cp=0.0001))

printcp(tree\_model2\_pruned)

tree.pruned2=prune(tree\_model2\_pruned, cp= 0.0040273359 )

rpart.plot(tree.pruned2,type=4,extra="Auto")

pred=predict(tree\_model2\_pruned,val)

perform\_treeModel2\_pruned=acc\_error(val$packaging\_y\_dim,pred)

perform\_treeModel2\_pruned

#Random forest1 - all predictors - ntree=150

Randomtree1 <- randomForest(packaging\_y\_dim ~ product\_x\_dim+product\_y\_dim +product\_zdim +product\_weight+

product\_is\_esp + product\_is\_dangerous\_good + product\_quality\_index,

data=train, mtry=3, ntree= 150, importance=TRUE, na.action=na.omit)

pred = predict(Randomtree1, val)

perform\_randomTree1 = acc\_error(val$packaging\_y\_dim, pred)

perform\_randomTree1

##Packaging\_z\_dim##

#Linear model 1 - all predictors

Linear\_model1=lm(packaging\_z\_dim ~ product\_is\_esp + product\_is\_dangerous\_good + product\_weight

+ product\_x\_dim + product\_y\_dim + product\_zdim + product\_quality\_index, data=train)

summary(Linear\_model1)

pred=predict(Linear\_model1,val)

perform\_linearModel1=acc\_error(val$packaging\_z\_dim,pred)

perform\_linearModel1

mean(packaging\_y\_dim)

#Linear model 2 - all numerical predictors

Linear\_model2=lm(packaging\_z\_dim ~ product\_weight

+ product\_x\_dim + product\_y\_dim + product\_zdim + product\_quality\_index, data=train)

summary(Linear\_model2)

pred=predict(Linear\_model2,val)

perform\_linearModel2=acc\_error(val$packaging\_y\_dim,pred)

perform\_linearModel2

#Linear model 3 - log(all numerical predictors)

Linear\_model3=lm(packaging\_z\_dim ~log(product\_weight) + log(product\_x\_dim) + log(product\_y\_dim)

+log(product\_zdim) + log(product\_quality\_index) , data=train)

summary(Linear\_model3)

pred=predict(Linear\_model3,val)

perform\_linearModel3=acc\_error(val$packaging\_z\_dim,pred)

perform\_linearModel3

#Tree model1 - all predictors - not pruned

tree\_model1 <- rpart(packaging\_z\_dim ~ product\_x\_dim +product\_y\_dim + product\_zdim

+product\_weight + product\_quality\_index+ product\_is\_esp + product\_is\_dangerous\_good,

data=train, control = rpart.control(cp=0.0001))

printcp(tree\_model1)

tree\_model1$cptable

bestcp=tree\_model1$cptable[which.min(tree\_model1$cptable[,"xerror"]),"CP"]

splits=tree\_model1$cptable[which.min(tree\_model1$cptable[,"xerror"]),"nsplit"]

splits

tree.pruned=prune(tree\_model1, cp= bestcp)

rpart.plot(tree.pruned,type=4,extra="Auto")

pred=predict(tree.pruned,val)

perform\_treeModel1=acc\_error(val$packaging\_z\_dim,pred)

perform\_treeModel1

#Tree model2 - all predictors - pruned

tree\_model2\_pruned <- rpart(packaging\_z\_dim ~ product\_x\_dim +product\_y\_dim + product\_zdim

+product\_weight + product\_quality\_index+ product\_is\_esp + product\_is\_dangerous\_good,

data=train, control = rpart.control(cp=0.0001))

printcp(tree\_model2\_pruned)

tree.pruned2=prune(tree\_model2\_pruned, cp= 0.0024629475 )

rpart.plot(tree.pruned2,type=4,extra="Auto")

pred=predict(tree\_model2\_pruned,val)

perform\_treeModel2\_pruned=acc\_error(val$packaging\_z\_dim,pred)

perform\_treeModel2\_pruned

#Random forest1 - all predictors - ntree=150

Randomtree1 <- randomForest(packaging\_z\_dim ~ product\_x\_dim+product\_y\_dim +product\_zdim +product\_weight+

product\_is\_esp + product\_is\_dangerous\_good + product\_quality\_index,

data=train, mtry=3, ntree= 150, importance=TRUE, na.action=na.omit)

pred = predict(Randomtree1, val)

perform\_randomTree1 = acc\_error(val$packaging\_z\_dim, pred)

perform\_randomTree1

## Packaging\_weight##

#Linear model 1 - all predictors

Linear\_model1=lm(packaging\_weight ~ product\_x\_dim + product\_is\_esp +product\_weight +

product\_quality\_index + product\_y\_dim +product\_zdim + product\_is\_dangerous\_good, data=train)

summary(Linear\_model1)

pred=predict(Linear\_model1,val)

perform\_linearModel1=acc\_error(val$packaging\_weight,pred)

perform\_linearModel1

mean(packaging\_weight)

#Linear model 2 - all numerical predictors

Linear\_model2=lm(packaging\_weight~ product\_x\_dim +product\_weight +

product\_quality\_index + product\_y\_dim +product\_zdim , data=train)

summary(Linear\_model2)

pred=predict(Linear\_model2,val)

perform\_linearModel2=acc\_error(val$packaging\_weight,pred)

perform\_linearModel2

mean(packaging\_weight)

#Linear\_model 3 - log( all numerical predictors)

Linear\_model3= lm(packaging\_weight ~ log(product\_x\_dim) +log(product\_y\_dim)

+ log(product\_weight) +log(product\_quality\_index) + log(product\_zdim), data=train)

summary(Linear\_model3)

pred=predict(Linear\_model3,val)

perform\_linearModel3=acc\_error(val$packaging\_weight,pred)

perform\_linearModel3

mean(packaging\_x\_dim)

#Tree model1 - all predictors - not pruned

tree\_model1 <- rpart(packaging\_weight ~ product\_x\_dim +product\_y\_dim + product\_zdim

+product\_weight + product\_quality\_index+ product\_is\_esp + product\_is\_dangerous\_good,

data=train, control = rpart.control(cp=0.0001))

printcp(tree\_model1)

tree\_model1$cptable

bestcp=tree\_model1$cptable[which.min(tree\_model1$cptable[,"xerror"]),"CP"]

splits=tree\_model1$cptable[which.min(tree\_model1$cptable[,"xerror"]),"nsplit"]

splits

tree.pruned=prune(tree\_model1, cp= bestcp)

rpart.plot(tree.pruned,type=4,extra="Auto")

pred=predict(tree.pruned,val)

perform\_treeModel1=acc\_error(val$packaging\_weight,pred)

perform\_treeModel1

#Tree model2 - all predictors - pruned

tree\_model2\_pruned <- rpart(packaging\_weight ~ product\_x\_dim +product\_y\_dim + product\_zdim

+product\_weight + product\_quality\_index+ product\_is\_esp + product\_is\_dangerous\_good,

data=train, control = rpart.control(cp=0.0001))

printcp(tree\_model2\_pruned)

tree.pruned2=prune(tree\_model2\_pruned, cp= 0.0072496533 )

rpart.plot(tree.pruned2,type=4,extra="Auto")

pred=predict(tree\_model2\_pruned,val)

perform\_treeModel2\_pruned=acc\_error(val$packaging\_weight,pred)

perform\_treeModel2\_pruned

#Random forest1 - all predictors - ntree=150

Randomtree1 <- randomForest(packaging\_weight ~ product\_x\_dim+product\_y\_dim +product\_zdim +product\_weight+

product\_is\_esp + product\_is\_dangerous\_good + product\_quality\_index,

data=train, mtry=3, ntree= 150, importance=TRUE, na.action=na.omit)

pred = predict(Randomtree1, val)

perform\_randomTree1 = acc\_error(val$packaging\_weight, pred)

perform\_randomTree1

##products\_per\_packaging##

#Linear model 1 - all predictors

Linear\_model1=lm(products\_per\_packaging ~ product\_x\_dim + product\_is\_esp +product\_weight +

product\_quality\_index + product\_y\_dim +product\_zdim + product\_is\_dangerous\_good, data=train)

summary(Linear\_model1)

pred=predict(Linear\_model1,val)

perform\_linearModel1=acc\_error(val$products\_per\_packaging,pred)

perform\_linearModel1

mean(products\_per\_packaging)

par(mfrow=c(2,2))

plot(Linear\_model1)

#Linear model 2 - remove insignificant

Linear\_model2=lm(products\_per\_packaging ~ product\_is\_esp

+ product\_quality\_index+ product\_zdim, data=train)

summary(Linear\_model2)

pred=predict(Linear\_model2,val)

perform\_linearModel2=acc\_error(val$products\_per\_packaging,pred)

perform\_linearModel2

#Linear model 3 - log(all numerical predictors)

Linear\_model3=lm(products\_per\_packaging ~log(product\_weight) + log(product\_x\_dim) + log(product\_y\_dim)

+ log(product\_quality\_index)+log(product\_zdim), data=train)

summary(Linear\_model3)

pred=predict(Linear\_model3,val)

perform\_linearModel3=acc\_error(val$products\_per\_packaging,pred)

perform\_linearModel3

#Tree model1 - all predictors - not pruned

tree\_model1 <- rpart(products\_per\_packaging ~ product\_x\_dim +product\_y\_dim + product\_zdim

+product\_weight + product\_quality\_index+ product\_is\_esp + product\_is\_dangerous\_good,

data=train, control = rpart.control(cp=0.0001))

printcp(tree\_model1)

tree\_model1$cptable

bestcp=tree\_model1$cptable[which.min(tree\_model1$cptable[,"xerror"]),"CP"]

splits=tree\_model1$cptable[which.min(tree\_model1$cptable[,"xerror"]),"nsplit"]

splits

tree.pruned=prune(tree\_model1, cp= bestcp)

rpart.plot(tree.pruned,type=4,extra="Auto")

pred=predict(tree.pruned,val)

perform\_treeModel1=acc\_error(val$products\_per\_packaging,pred)

perform\_treeModel1

#Tree model2 - all predictors - pruned

tree\_model2\_pruned <- rpart(products\_per\_packaging ~ product\_x\_dim +product\_y\_dim + product\_zdim

+product\_weight + product\_quality\_index+ product\_is\_esp + product\_is\_dangerous\_good,

data=train, control = rpart.control(cp=0.0001))

printcp(tree\_model2\_pruned)

tree.pruned2=prune(tree\_model2\_pruned, cp= 0.0008959847 )

rpart.plot(tree.pruned2,type=4,extra="Auto")

pred=predict(tree\_model2\_pruned,val)

perform\_treeModel2\_pruned=acc\_error(val$products\_per\_packaging,pred)

perform\_treeModel2\_pruned

#Random forest1 - all predictors - ntree=150

Randomtree1 <- randomForest(products\_per\_packaging ~ product\_x\_dim+product\_y\_dim +product\_zdim +product\_weight+

product\_is\_esp + product\_is\_dangerous\_good + product\_quality\_index,

data=train, mtry=3, ntree= 150, importance=TRUE, na.action=na.omit)

pred = predict(Randomtree1, val)

perform\_randomTree1 = acc\_error(val$products\_per\_packaging, pred)

perform\_randomTree1

##packaging\_load\_capacity##

#Linear model 1 - all predictors

Linear\_model1=lm(packaging\_load\_capacity ~ product\_is\_esp + product\_is\_dangerous\_good + product\_weight

+ product\_x\_dim + product\_y\_dim + product\_zdim + product\_quality\_index, data=train)

summary(Linear\_model1)

pred=predict(Linear\_model1,val)

perform\_linearModel1=acc\_error(val$packaging\_load\_capacity,pred)

perform\_linearModel1

mean(packaging\_load\_capacity)

#Linear model 2 - eliminate insignificant (product\_is\_dangerous\_good)

Linear\_model2=lm(packaging\_load\_capacity ~ product\_is\_esp + product\_weight

+ product\_x\_dim + product\_y\_dim + product\_zdim + product\_quality\_index, data=train)

summary(Linear\_model2)

pred=predict(Linear\_model2,val)

perform\_linearModel2=acc\_error(val$packaging\_load\_capacity,pred)

perform\_linearModel2

mean(packaging\_load\_capacity)

#Linear model 3 - all numerical predictors

Linear\_model3=lm(packaging\_load\_capacity ~ product\_weight

+ product\_x\_dim + product\_y\_dim + product\_zdim + product\_quality\_index, data=train)

summary(Linear\_model3)

pred=predict(Linear\_model3,val)

perform\_linearModel3=acc\_error(val$packaging\_load\_capacity,pred)

perform\_linearModel3

#Linear model 4 - log(all numerical predictors)

Linear\_model4=lm(packaging\_load\_capacity ~log(product\_weight) + log(product\_x\_dim) + log(product\_y\_dim)

+log(product\_zdim) + log(product\_quality\_index) , data=train)

summary(Linear\_model4)

pred=predict(Linear\_model4,val)

perform\_linearModel4=acc\_error(val$packaging\_load\_capacity,pred)

perform\_linearModel4

#Tree model1 - all predictors - not pruned

tree\_model1 <- rpart(packaging\_load\_capacity ~ product\_x\_dim +product\_y\_dim + product\_zdim

+product\_weight + product\_quality\_index+ product\_is\_esp + product\_is\_dangerous\_good,

data=train, control = rpart.control(cp=0.0001))

printcp(tree\_model1)

tree\_model1$cptable

bestcp=tree\_model1$cptable[which.min(tree\_model1$cptable[,"xerror"]),"CP"]

splits=tree\_model1$cptable[which.min(tree\_model1$cptable[,"xerror"]),"nsplit"]

splits

tree.pruned=prune(tree\_model1, cp= bestcp)

rpart.plot(tree.pruned,type=4,extra="Auto")

pred=predict(tree.pruned,val)

perform\_treeModel1=acc\_error(val$packaging\_load\_capacity,pred)

perform\_treeModel1

#Tree model2 - all predictors - pruned

tree\_model2\_pruned <- rpart(packaging\_load\_capacity ~ product\_x\_dim +product\_y\_dim + product\_zdim

+product\_weight + product\_quality\_index+ product\_is\_esp + product\_is\_dangerous\_good,

data=train, control = rpart.control(cp=0.0001))

printcp(tree\_model2\_pruned)

tree.pruned2=prune(tree\_model2\_pruned, cp= 0.0065238912 )

rpart.plot(tree.pruned2,type=4,extra="Auto")

pred=predict(tree\_model2\_pruned,val)

perform\_treeModel2\_pruned=acc\_error(val$packaging\_load\_capacity,pred)

perform\_treeModel2\_pruned

#Random forest1 - all predictors - ntree=150

Randomtree1 <- randomForest(packaging\_load\_capacity ~ product\_x\_dim+product\_y\_dim +product\_zdim +product\_weight+

product\_is\_esp + product\_is\_dangerous\_good + product\_quality\_index,

data=train, mtry=3, ntree= 150, importance=TRUE, na.action=na.omit)

pred = predict(Randomtree1, val)

perform\_randomTree1 = acc\_error(val$packaging\_load\_capacity, pred)

perform\_randomTree1

####### LOGISTIC REGRESSION ################

####### WE used all variables of product features #########

#BUILDING THE MODEL#

lmod1=glm(packaging\_is\_oneway~+product\_is\_dangerous\_good+product\_is\_esp+product\_quality\_index+product\_weight+product\_x\_dim+product\_y\_dim+product\_zdim, data=train)

summary(lmod)

#coefficient#

cof=lmod1$coefficients

OR\_vec=exp(lmod1$coefficients)

CI\_OR=exp(confint.default(lmod1))

#Predicting and evaluating the mode

pred\_1=predict(lmod1,newdata=val,type="response")

act\_pred=data.frame(ID=1:nrow(val),val$packaging\_is\_oneway,pred\_1)

conf\_mat1=cmx(act\_pred)

total\_acc1=pcc(conf\_mat1)

sens1=sensitivity(conf\_mat1)

spec1=specificity(conf\_mat1)

auc(act\_pred)

par(mfrow=c(1,1))

auc.roc.plot(act\_pred)

conf\_mat1

total\_acc1

sens1

spec1

#### Only 1 variables showed important relation, I tried to build the model but gave poor results so I did not bother putting it#

lmod2=glm(packaging\_is\_oneway~product\_is\_esp, data=train)

summary(lmod2)

#Coefficient

cof=lmod2$coefficients

OR\_vec=exp(lmod2$coefficients)

CI\_OR=exp(confint.default(lmod2))

#Predictions and evaulate the performance

pred\_2=predict(lmod2,newdata=val,type="response")

act\_pred2=data.frame(ID=1:nrow(val),val$packaging\_is\_oneway,pred\_2)

conf\_mat2=cmx(act\_pred2)

total\_acc2=pcc(conf\_mat2)

sens2=sensitivity(conf\_mat2)

spec2=specificity(conf\_mat2)

auc(act\_pred2)

par(mfrow=c(1,1))

auc.roc.plot(act\_pred2)

conf\_mat2

total\_acc2

sens2

spec2

#Comparing both models#

act\_pred\_mult=data.frame(ID=1:nrow(val),val$packaging\_is\_oneway,pred\_1,pred\_2)

auc.roc.plot(act\_pred\_mult,col=c(2,3),line.type=c(1,2),threshold = 1001, main="ROC Curves",legend.text=c("Model 1", "Model 2"))

######## 4 SAMPLING TECHNIQUES ###########

#Taking the columns we need#

df\_REGR<-df %>% select(packaging\_is\_oneway,product\_is\_dangerous\_good,product\_is\_esp,product\_quality\_index,product\_weight,product\_x\_dim,product\_y\_dim,product\_zdim)

set.seed(100)

index=sample(1:2, nrow(df\_REGR), replace=TRUE, prob=c(0.7,0.3))

train=df\_REGR[index==1,]

val=df\_REGR[index==2,]

table(train$packaging\_is\_oneway)

#Oversampling#

df\_oversampling <- ovun.sample(packaging\_is\_oneway ~ ., data =train , method = "over",N = 8952+8952)$data

table(df\_oversampling$packaging\_is\_oneway)

#Undersampling#

data\_balanced\_under <- ovun.sample(packaging\_is\_oneway ~ ., data = train, method = "under", N = 264+264, seed = 100)$data

table(data\_balanced\_under$packaging\_is\_oneway)

#Both#

data\_balanced\_both <- ovun.sample(packaging\_is\_oneway ~ ., data = train, method = "both", p=0.5,N=8952+264, seed = 100)$data

table(data\_balanced\_both$packaging\_is\_oneway)

#Rose#

data.rose <- ROSE(packaging\_is\_oneway ~ ., data = train, seed = 100)$data

table(data.rose$packaging\_is\_oneway)

#Building logistic regression using oversampling data#

lmod=glm(packaging\_is\_oneway~., data=df\_oversampling)

summary(lmod)

cof=lmod$coefficients

OR\_vec=exp(lmod$coefficients)

CI\_OR=exp(confint.default(lmod))

#Evaluating the mode#

pred\_3=predict(lmod,newdata=val,type="response")

act\_pred3=data.frame(ID=1:nrow(val),val$packaging\_is\_oneway,pred\_3)

act\_pred3

conf\_mat1=cmx(act\_pred3)

total\_acc1=pcc(conf\_mat1)

sens1=sensitivity(conf\_mat1)

spec1=specificity(conf\_mat1)

auc(act\_pred3)

par(mfrow=c(1,1))

auc.roc.plot(act\_pred3)

conf\_mat1

total\_acc1

sens1

spec1

#Building using the under data

lmod=glm(packaging\_is\_oneway~., data=data\_balanced\_under)

summary(lmod)

cof=lmod$coefficients

OR\_vec=exp(lmod$coefficients)

CI\_OR=exp(confint.default(lmod))

#Evaluating model#

pred\_4=predict(lmod,newdata=val,type="response")

act\_pred4=data.frame(ID=1:nrow(val),val$packaging\_is\_oneway,pred\_4)

act\_pred4

conf\_mat1=cmx(act\_pred4)

total\_acc1=pcc(conf\_mat1)

sens1=sensitivity(conf\_mat1)

spec1=specificity(conf\_mat1)

auc(act\_pred4)

par(mfrow=c(1,1))

auc.roc.plot(act\_pred4)

conf\_mat1

total\_acc1

sens1

spec1

#Building using both method#

lmod=glm(packaging\_is\_oneway~., data=data\_balanced\_both)

summary(lmod)

cof=lmod$coefficients

OR\_vec=exp(lmod$coefficients)

CI\_OR=exp(confint.default(lmod))

#Evaluating#

pred\_5=predict(lmod,newdata=val,type="response")

act\_pred5=data.frame(ID=1:nrow(val),val$packaging\_is\_oneway,pred\_5)

act\_pred5

conf\_mat1=cmx(act\_pred5)

total\_acc1=pcc(conf\_mat1)

sens1=sensitivity(conf\_mat1)

spec1=specificity(conf\_mat1)

auc(act\_pred5)

par(mfrow=c(1,1))

auc.roc.plot(act\_pred5)

conf\_mat1

total\_acc1

sens1

spec1

#Comparing the roc curves

act\_pred\_mult=data.frame(ID=1:nrow(val),val$packaging\_is\_oneway,pred\_3,pred\_4,pred\_5)

act\_pred\_mult

auc.roc.plot(act\_pred\_mult,col=c(1,2,3),line.type=c(1,2,3),threshold = 1001, main="Packaging is One Way (ROC CURVES)",legend.text=c("Oversampling", "Undersampling","Both"))

act\_pred\_mult=data.frame(ID=1:nrow(val),val$packaging\_is\_oneway,pred\_1,pred\_5)

act\_pred\_mult

auc.roc.plot(act\_pred\_mult,col=c(1,2),line.type=c(1,2),threshold = 1001, main="Packaging is One Way (ROC CURVES)",legend.text=c("Model 1", "Model2"))

act\_pred\_mult=data.frame(ID=1:nrow(val),val$packaging\_is\_oneway,pred\_1,pred\_3,pred\_4,pred\_5)

act\_pred\_mult

auc.roc.plot(act\_pred\_mult,col=c(1,2,3,4),line.type=c(1,2,3,4),threshold = 1001, main="Packaging is One Way (ROC CURVES)",legend.text=c("Stand Logistic Regression","Oversampling", "Undersampling","Both"))

######### 4 SAMPLING TECHNIQUES ############

##### Removing insignificant variable ##########

library(ROSE)

library(dplyr)

df\_REGR<-df %>% select(packaging\_is\_oneway,product\_is\_esp,product\_x\_dim,product\_y\_dim,product\_zdim)

set.seed(100)

index=sample(1:2, nrow(df\_REGR), replace=TRUE, prob=c(0.7,0.3))

train=df\_REGR[index==1,]

val=df\_REGR[index==2,]

table(train$packaging\_is\_oneway)

df\_oversampling <- ovun.sample(packaging\_is\_oneway ~ ., data =train , method = "over",N = (8952+8952), seed=100)$data

table(df\_oversampling$packaging\_is\_oneway)

data\_balanced\_under <- ovun.sample(packaging\_is\_oneway ~ ., data = train, method = "under", N = 264+264, seed = 100)$data

table(data\_balanced\_under$packaging\_is\_oneway)

data\_balanced\_both <- ovun.sample(packaging\_is\_oneway ~ ., data = train, method = "both", p=0.5,N=11051+376, seed = 100)$data

table(data\_balanced\_both$packaging\_is\_oneway)

data.rose <- ROSE(packaging\_is\_oneway ~ ., data = train, seed = 100)$data

table(data.rose$packaging\_is\_oneway)

lmod7=glm(packaging\_is\_oneway~., data=df\_oversampling)

summary(lmod7)

cof=lmod7$coefficients

OR\_vec=exp(lmod7$coefficients)

CI\_OR=exp(confint.default(lmod7))

pred\_7=predict(lmod7,newdata=val,type="response")

act\_pred7=data.frame(ID=1:nrow(val),val$packaging\_is\_oneway,pred\_7)

act\_pred7

conf\_mat1=cmx(act\_pred7)

total\_acc1=pcc(conf\_mat1)

sens1=sensitivity(conf\_mat1)

spec1=specificity(conf\_mat1)

auc(act\_pred7)

par(mfrow=c(1,1))

auc.roc.plot(act\_pred7)

conf\_mat1

total\_acc1

sens1

spec1

lmod8=glm(packaging\_is\_oneway~., data=data\_balanced\_under)

summary(lmod8)

cof=lmod8$coefficients

OR\_vec=exp(lmod8$coefficients)

CI\_OR=exp(confint.default(lmod8))

pred\_8=predict(lmod8,newdata=val,type="response")

act\_pred8=data.frame(ID=1:nrow(val),val$packaging\_is\_oneway,pred\_8)

act\_pred8

conf\_mat1=cmx(act\_pred8)

total\_acc1=pcc(conf\_mat1)

sens1=sensitivity(conf\_mat1)

spec1=specificity(conf\_mat1)

auc(act\_pred8)

par(mfrow=c(1,1))

auc.roc.plot(act\_pred8)

conf\_mat1

total\_acc1

sens1

spec1

lmod9=glm(packaging\_is\_oneway~., data=data\_balanced\_both)

summary(lmod9)

cof=lmod9$coefficients

OR\_vec=exp(lmod9$coefficients)

CI\_OR=exp(confint.default(lmod9))

pred\_9=predict(lmod9,newdata=val,type="response")

act\_pred9=data.frame(ID=1:nrow(val),val$packaging\_is\_oneway,pred\_9)

act\_pred9

conf\_mat1=cmx(act\_pred9)

total\_acc1=pcc(conf\_mat1)

sens1=sensitivity(conf\_mat1)

spec1=specificity(conf\_mat1)

auc(act\_pred9)

par(mfrow=c(1,1))

auc.roc.plot(act\_pred9)

conf\_mat1

total\_acc1

sens1

spec1

lmod10=glm(packaging\_is\_oneway~., data=data.rose)

summary(lmod10)

cof=lmod10$coefficients

OR\_vec=exp(lmod10$coefficients)

CI\_OR=exp(confint.default(lmod10))

pred\_10=predict(lmod10,newdata=val,type="response")

act\_pred10=data.frame(ID=1:nrow(val),val$packaging\_is\_oneway,pred\_10)

act\_pred10

conf\_mat1=cmx(act\_pred10)

total\_acc1=pcc(conf\_mat1)

sens1=sensitivity(conf\_mat1)

spec1=specificity(conf\_mat1)

auc(act\_pred10)

par(mfrow=c(1,1))

auc.roc.plot(act\_pred10)

conf\_mat1

total\_acc1

sens1

spec1

act\_pred\_mult=data.frame(ID=1:nrow(val),val$packaging\_is\_oneway,pred\_7,pred\_8,pred\_9,pred\_10)

auc.roc.plot(act\_pred\_mult,col=c(1,2,3,4),line.type=c(1,2),threshold = 1001, main="ROC Curves",legend.text=c("Model 7", "Model 8","Model 9","Model 10"))

##### We will pick model 8 here ############

####### SELECT EACH MODEL ############

#### FOR CONSISTENCY BASIS, WE WILL RENAME THE MODEL #######

act\_pred\_mult=data.frame(ID=1:nrow(val),val$packaging\_is\_oneway,pred\_1,pred\_2,pred\_5,pred\_8)

auc.roc.plot(act\_pred\_mult,col=c(1,2,3,4),line.type=c(1,2,3,4),threshold = 1001, main="ROC Curves",legend.text=c("Model 1", "Model 2","Model 3","Model 4"))

###### DECISION TREE ######

##### DECISION TREE ###

###### oneway #####

#Selecting the columns that will be used

df\_tree<-df %>% select(packaging\_is\_oneway,product\_is\_dangerous\_good,product\_is\_esp,product\_quality\_index,product\_weight,product\_x\_dim,product\_y\_dim,product\_zdim)

table(df\_tree$packaging\_is\_oneway)

head(df\_tree)

#Splitting the data

set.seed(100)

index=sample(1:2, nrow(df\_tree), replace=TRUE, prob=c(0.7,0.3))

train=df\_tree[index==1,]

val=df\_tree[index==2,]

#Buildig the model

tree <- rpart(as.factor(packaging\_is\_oneway) ~ ., data = train, control = rpart.control(cp = 0.0001))

dfimp<-printcp(tree)

# Chose the model with the best xerror

tree.pruned=prune(tree, cp =dfimp[5,1])

rpart.plot(tree.pruned,type=2)

#evauating the model

pred11=predict(tree.pruned, newdata=val, type="prob")

pred11=pred11[,2]

act\_pred11=data.frame(ID=1:nrow(val),val$packaging\_is\_oneway,pred11)

act\_pred11

conf\_mat=cmx(act\_pred11)

total\_acc= pcc(conf\_mat,st.dev = FALSE)

sens=sensitivity(conf\_mat,st.dev = FALSE)# to obtain snesitivty

spec=specificity(conf\_mat,st.dev = FALSE)# to obtain specificity

accuracy\_measures=c(total\_acc,sens,spec)

names(accuracy\_measures)=c("Overall accuracy", "Sensitivity", "Specificity")

conf\_mat

total\_acc

sens

spec

accuracy\_measures

arc=auc(act\_pred11)

arc

auc.roc.plot(act\_pred11,col="red")

##### SAMPLING TECHNIQUES ########

df\_tree<-df %>% select(packaging\_is\_oneway,product\_is\_dangerous\_good,product\_is\_esp,product\_quality\_index,product\_weight,product\_x\_dim,product\_y\_dim,product\_zdim)

head(df\_tree)

df\_tree <- df\_tree %>%

mutate(packaging\_is\_oneway = ifelse(packaging\_is\_oneway == 0,'No','Yes'))

df\_tree

set.seed(100)

index=sample(1:2, nrow(df\_tree), replace=TRUE, prob=c(0.7,0.3))

train=df\_tree[index==1,]

val=df\_tree[index==2,]

table(train$packaging\_is\_oneway)

#Oversample#

df\_tree\_oversampling <- ovun.sample(packaging\_is\_oneway ~ ., data =train , method = "over",N = 8952+8952, seed=100)$data

table(df\_tree\_oversampling$packaging\_is\_oneway)

#Undersample#

data\_balanced\_under <- ovun.sample(packaging\_is\_oneway ~ ., data = train, method = "under", N = 264+264, seed = 100)$data

table(data\_balanced\_under$packaging\_is\_oneway)

#both#

data\_balanced\_both <- ovun.sample(packaging\_is\_oneway ~ ., data = train, method = "both", p=0.5,N=8952+8952, seed = 100)$data

table(data\_balanced\_both$packaging\_is\_oneway)

#Rose#

data.rose <- ROSE(packaging\_is\_oneway ~ ., data = train)$data

table(data.rose$packaging\_is\_oneway)

#Building model using Rose#

tree.rose <- rpart(packaging\_is\_oneway ~ ., data = data.rose,cp=0.0001)

dfimp<-printcp(tree.rose)

#Must bet besr xerror among those with splits less than 30

tree.pruned=prune(tree.rose, cp =dfimp[13,1])

rpart.plot(tree.pruned,type=2)

#Evaluating the mode#

pred13=predict(tree.pruned, newdata=val, type="prob")

pred13=pred13[,2]

act\_pred13=data.frame(ID=1:nrow(val),1\*(val$packaging\_is\_oneway=="Yes"),pred13)

conf\_mat=cmx(act\_pred13)

total\_acc= pcc(conf\_mat,st.dev = FALSE)

sens=sensitivity(conf\_mat,st.dev = FALSE)# to obtain snesitivty

spec=specificity(conf\_mat,st.dev = FALSE)# to obtain specificity

accuracy\_measures=c(total\_acc,sens,spec)

names(accuracy\_measures)=c("Overall accuracy", "Sensitivity", "Specificity")

conf\_mat

total\_acc

sens

spec

accuracy\_measures

arc=auc(act\_pred13) #to obtain the area under Roc curve

arc

auc.roc.plot(act\_pred13,col="red")

#Building model using oversampling#

tree.over <- rpart(packaging\_is\_oneway ~ ., data = df\_tree\_oversampling,cp=0.0001)

dfimp<-printcp(tree.over)

dfimp[17,1]

tree.pruned=prune(tree.over, cp =dfimp[17,1])

rpart.plot(tree.pruned,type=2)

#Evaluating the model

pred14=predict(tree.pruned, newdata=val, type="prob")

pred14=pred14[,2]

act\_pred14=data.frame(ID=1:nrow(val),1\*(val$packaging\_is\_oneway=="Yes"),pred14)#

conf\_mat=cmx(act\_pred14)

total\_acc= pcc(conf\_mat,st.dev = FALSE)

sens=sensitivity(conf\_mat,st.dev = FALSE)# to obtain snesitivty

spec=specificity(conf\_mat,st.dev = FALSE)# to obtain specificity

accuracy\_measures=c(total\_acc,sens,spec)

names(accuracy\_measures)=c("Overall accuracy", "Sensitivity", "Specificity")

conf\_mat

total\_acc

sens

spec

accuracy\_measures

auc.roc.plot(act\_pred14,col="red")

auc

#Building the model with UNder methode#

tree.under <- rpart(packaging\_is\_oneway ~ ., data = data\_balanced\_under,cp=0.00001)

dfimp<-printcp(tree.under)

dfimp[10,1]

tree.pruned=prune(tree.over, cp =dfimp[12,1])

rpart.plot(tree.pruned,type=2)

#Evalating#

pred15=predict(tree.pruned, newdata=val, type="prob")

pred15=pred15[,2]

act\_pred15=data.frame(ID=1:nrow(val),1\*(val$packaging\_is\_oneway=="Yes"),pred15)

act\_pred15

conf\_mat=cmx(act\_pred15)

total\_acc= pcc(conf\_mat,st.dev = FALSE)

sens=sensitivity(conf\_mat,st.dev = FALSE)# to obtain snesitivty

spec=specificity(conf\_mat,st.dev = FALSE)# to obtain specificity

accuracy\_measures=c(total\_acc,sens,spec)

names(accuracy\_measures)=c("Overall accuracy", "Sensitivity", "Specificity")

conf\_mat

total\_acc

sens

spec

accuracy\_measures

arc=auc(act\_pred15) #to obtain the area under Roc curve

arc

auc.roc.plot(act\_pred15,col="red")

#Both method#

tree.both <- rpart(packaging\_is\_oneway ~ ., data = data\_balanced\_both,cp=0.0001)

dfimp<-printcp(tree.both)

dfimp[13,1]

tree.pruned=prune(tree.both, cp =dfimp[17,1])

rpart.plot(tree.pruned,type=4, extra = 2)

#Predict and evaluation

pred16=predict(tree.pruned, newdata=val, type="prob")

pred16=pred16[,2]

act\_pred16=data.frame(ID=1:nrow(val),1\*(val$packaging\_is\_oneway=="Yes"),pred16)

act\_pred16

conf\_mat=cmx(act\_pred16)

total\_acc= pcc(conf\_mat,st.dev = FALSE)

sens=sensitivity(conf\_mat,st.dev = FALSE)# to obtain snesitivty

spec=specificity(conf\_mat,st.dev = FALSE)# to obtain specificity

accuracy\_measures=c(total\_acc,sens,spec)

names(accuracy\_measures)=c("Overall accuracy", "Sensitivity", "Specificity")

conf\_mat

total\_acc

sens

spec

accuracy\_measures

arc=auc(act\_pred16) #to obtain the area under Roc curve

arc

auc.roc.plot(act\_pred16,col="red")

#Comparing

act\_pred\_mult=data.frame(ID=1:nrow(val),1\*(val$packaging\_is\_oneway=="Yes"),pred14, pred15, pred16)

auc.roc.plot(act\_pred\_mult,col=c(1,2,3),line.type=c(1,2,3),threshold = 1001, main="Packaging is One Way (ROC CURVES",legend.text=c("Oversampling", "Undersampling","Both"))

act\_pred\_mult=data.frame(ID=1:nrow(val),1\*(val$packaging\_is\_oneway=="Yes"),pred11,pred16)

auc.roc.plot(act\_pred\_mult,col=c(3,4),line.type=c(3,4),threshold = 1001, main="Packaging is One Way (ROC CURVES)",legend.text=c("Model 3", "Model 4"))

act\_pred\_mult=data.frame(ID=1:nrow(val),1\*(val$packaging\_is\_oneway=="Yes"),pred11,pred14, pred15, pred16)

auc.roc.plot(act\_pred\_mult,col=c(1,2,3,4),line.type=c(1,2,3,4),threshold = 1001, main="Packaging is One Way (ROC CURVES)",legend.text=c("Stand Decision Tree","Oversampling", "Undersampling","Both"))

#### FINAL VERDICT ########

act\_pred\_mult=data.frame(ID=1:nrow(val),1\*(val$packaging\_is\_oneway=="Yes"),pred\_3,pred14)

act\_pred\_mult

auc.roc.plot(act\_pred\_mult,col=c(1,2),line.type=c(1,2),threshold = 1001, main="ROC Curves",legend.text=c("Model 2", "Model 5"))

##### auc witht he best 5 curves #######

act\_pred\_mult=data.frame(ID=1:nrow(val),1\*(val$packaging\_is\_oneway=="Yes"),pred\_1, pred\_5, pred11,pred16)

auc.roc.plot(act\_pred\_mult,col=c(1,2,3,4),line.type=c(1,2,3,4),threshold = 1001, main="Pacaging is One way (ROC CURVES)",legend.text=c("Stand Logistic Regression", "Logistic Tree Both","Stand Decision Tree","Decision Tree Both"))

###### LOGISTIC REGRESSION ########

###### WE used all variables of product features ########

### BUILDING THE MODEL ######

lmod1=glm(packaging\_is\_special~product\_is\_dangerous\_good+product\_is\_esp+product\_quality\_index+product\_weight+product\_x\_dim+product\_y\_dim+product\_zdim, data=train)

summary(lmod1)

#### Coefficient ######

cof=lmod1$coefficients

OR\_vec=exp(lmod1$coefficients)

CI\_OR=exp(confint.default(lmod1))

#### Prediction and accuracy measures ######

pred\_1=predict(lmod1,newdata=val,type="response")

act\_pred=data.frame(ID=1:nrow(val),val$packaging\_is\_special,pred\_1)

conf\_mat1=cmx(act\_pred)

total\_acc1=pcc(conf\_mat1)

sens1=sensitivity(conf\_mat1)

spec1=specificity(conf\_mat1)

auc<-auc(act\_pred)

par(mfrow=c(1,1))

auc.roc.plot(act\_pred)

conf\_mat1

total\_acc1

sens1

spec1

auc

###### Prediciting Models that shown to have a relation usng unsignificant model ####

lmod2=glm(packaging\_is\_special~product\_is\_esp+product\_quality\_index+product\_weight+product\_x\_dim, data=train)

summary(lmod2)

##### Coefficients #####

cof=lmod2$coefficients

OR\_vec=exp(lmod2$coefficients)

CI\_OR=exp(confint.default(lmod2))

####### Prediction and accuaracy measures######

pred\_2=predict(lmod2,newdata=val,type="response")

act\_pred2=data.frame(ID=1:nrow(val),val$packaging\_is\_special,pred\_2)

conf\_mat2=cmx(act\_pred2)

total\_acc2=pcc(conf\_mat2)

sens2=sensitivity(conf\_mat2)

spec2=specificity(conf\_mat2)

auc(act\_pred2)

par(mfrow=c(1,1))

auc.roc.plot(act\_pred2)

conf\_mat2

total\_acc2

sens2

spec2

##### Comparing Roc curves of both models ######

act\_pred\_mult=data.frame(ID=1:nrow(val),val$packaging\_is\_special,pred\_1,pred\_2)

auc.roc.plot(act\_pred\_mult,col=c(2,3),line.type=c(1,2),threshold = 1001, main="Packaging is Special (ROC CURVES)",legend.text=c("Model 1", "Model 2"))

### 4 SAMPLING TECHNIQUES ###########

######## Selecting the columns that we will be using ######

df\_REGR<-df %>% select(packaging\_is\_special,product\_is\_dangerous\_good,product\_is\_esp,product\_quality\_index,product\_weight,product\_x\_dim,product\_y\_dim,product\_zdim)

set.seed(100)

index=sample(1:2, nrow(df\_REGR), replace=TRUE, prob=c(0.7,0.3))

train=df\_REGR[index==1,]

val=df\_REGR[index==2,]

#Checking for imbalances#

table(train$packaging\_is\_special)

# OVERSAMPLNG #

df\_oversampling <- ovun.sample(packaging\_is\_special ~ ., data =train , method = "over",N = 6619+6619)$data

table(df\_oversampling$packaging\_is\_special)

# UNDERSAMPLING#

data\_balanced\_under <- ovun.sample(packaging\_is\_special ~ ., data = train, method = "under", N = 2597+2597, seed = 100)$data

table(data\_balanced\_under$packaging\_is\_special)

#BOTH#

data\_balanced\_both <- ovun.sample(packaging\_is\_special ~ ., data = train, method = "both", p=0.5,N=6619+2597, seed = 100)$data

table(data\_balanced\_both$packaging\_is\_special)

#ROSE NOT USED IN THE ARTICAL#

data.rose <- ROSE(packaging\_is\_special ~ ., data = train, seed = 100)$data

table(data.rose$packaging\_is\_special)

#Buildinng logistic model based on the oversampling data#

lmod3=glm(packaging\_is\_special~., data=df\_oversampling)

summary(lmod3)

cof=lmod3$coefficients

OR\_vec=exp(lmod3$coefficients)

CI\_OR=exp(confint.default(lmod3))

pred\_3=predict(lmod3,newdata=val,type="response")

act\_pred3=data.frame(ID=1:nrow(val),val$packaging\_is\_special,pred\_3)

act\_pred3

conf\_mat1=cmx(act\_pred3)

total\_acc1=pcc(conf\_mat1)

sens1=sensitivity(conf\_mat1)

spec1=specificity(conf\_mat1)

auc(act\_pred3)

par(mfrow=c(1,1))

auc.roc.plot(act\_pred3)

conf\_mat1

total\_acc1

sens1

spec1

#Buildinng logistic model based on the undersampling data#

lmod4=glm(packaging\_is\_special~., data=data\_balanced\_under)

summary(lmod4)

cof=lmod4$coefficients

OR\_vec=exp(lmod4$coefficients)

CI\_OR=exp(confint.default(lmod4))

pred\_4=predict(lmod4,newdata=val,type="response")

act\_pred4=data.frame(ID=1:nrow(val),val$packaging\_is\_special,pred\_4)

act\_pred4

conf\_mat1=cmx(act\_pred4)

total\_acc1=pcc(conf\_mat1)

sens1=sensitivity(conf\_mat1)

spec1=specificity(conf\_mat1)

auc(act\_pred4)

par(mfrow=c(1,1))

auc.roc.plot(act\_pred4)

conf\_mat1

total\_acc1

sens1

spec1

#Buildinng logistic model based on the bothe models#

lmod5=glm(packaging\_is\_special~., data=data\_balanced\_both)

summary(lmod5)

cof=lmod5$coefficients

OR\_vec=exp(lmod5$coefficients)

CI\_OR=exp(confint.default(lmod5))

pred\_5=predict(lmod5,newdata=val,type="response")

act\_pred5=data.frame(ID=1:nrow(val),val$packaging\_is\_special,pred\_5)

act\_pred5

conf\_mat1=cmx(act\_pred5)

total\_acc1=pcc(conf\_mat1)

sens1=sensitivity(conf\_mat1)

spec1=specificity(conf\_mat1)

auc(act\_pred5)

par(mfrow=c(1,1))

auc.roc.plot(act\_pred5)

conf\_mat1

total\_acc1

sens1

spec1

#Buildinng logistic model based on the ROSE models#

lmod6=glm(packaging\_is\_special~., data=data.rose)

summary(lmod6)

cof=lmod6$coefficients

OR\_vec=exp(lmod6$coefficients)

CI\_OR=exp(confint.default(lmod6))

pred\_6=predict(lmod6,newdata=val,type="response")

act\_pred6=data.frame(ID=1:nrow(val),val$packaging\_is\_special,pred\_6)

act\_pred6

conf\_mat1=cmx(act\_pred6)

total\_acc1=pcc(conf\_mat1)

sens1=sensitivity(conf\_mat1)

spec1=specificity(conf\_mat1)

auc(act\_pred6)

par(mfrow=c(1,1))

auc.roc.plot(act\_pred6)

conf\_mat1

total\_acc1

sens1

spec1

#COMPARING THESE MODELS#

act\_pred\_mult=data.frame(ID=1:nrow(val),val$packaging\_is\_special,pred\_3,pred\_4,pred\_5)

auc.roc.plot(act\_pred\_mult,col=c(1,2,3),line.type=c(1,2.3),threshold = 1001, main="Packaging is Special (ROC CURVES)",legend.text=c("Oversampling", "Undersampling","Both"))

act\_pred\_mult=data.frame(ID=1:nrow(val),val$packaging\_is\_special,pred\_1,pred\_2,pred\_4)

auc.roc.plot(act\_pred\_mult,col=c(1,2,3),line.type=c(1,2.3),threshold = 1001, main="Packaging is Special (ROC CURVES)",legend.text=c("Oversampling", "Undersampling","Both"))

####### 4 SAMPLING TECHNIQUES ########

##### Removing insignificant variable #######

#Choosing the coumns that are needed#

df\_REGR<-df %>% select(product\_is\_esp,packaging\_is\_special,product\_quality\_index,product\_weight,product\_x\_dim)

set.seed(100)

index=sample(1:2, nrow(df\_REGR), replace=TRUE, prob=c(0.7,0.3))

train=df\_REGR[index==1,]

val=df\_REGR[index==2,]

table(train$packaging\_is\_special)

#Oversampling#

df\_oversampling <- ovun.sample(packaging\_is\_special ~ ., data =train , method = "over",N = 6619+6619)$data

table(df\_oversampling$packaging\_is\_special)

#Undersampling#

data\_balanced\_under <- ovun.sample(packaging\_is\_special ~ ., data = train, method = "under", N = 2597+2597, seed = 100)$data

table(data\_balanced\_under$packaging\_is\_special)

#Both#

data\_balanced\_both <- ovun.sample(packaging\_is\_special ~ ., data = train, method = "both", p=0.5,N=6619+2597, seed = 100)$data

table(data\_balanced\_both$packaging\_is\_special)

#Rose#

data.rose <- ROSE(packaging\_is\_special ~ ., data = train, seed = 100)$data

table(data.rose$packaging\_is\_special)

#Building Model on the oversample data#

lmod7=glm(packaging\_is\_special~., data=df\_oversampling)

summary(lmod7)

cof=lmod7$coefficients

OR\_vec=exp(lmod7$coefficients)

CI\_OR=exp(confint.default(lmod7))

pred\_7=predict(lmod7,newdata=val,type="response")

act\_pred7=data.frame(ID=1:nrow(val),val$packaging\_is\_special,pred\_7)

act\_pred7

conf\_mat1=cmx(act\_pred7)

total\_acc1=pcc(conf\_mat1)

sens1=sensitivity(conf\_mat1)

spec1=specificity(conf\_mat1)

auc(act\_pred7)

par(mfrow=c(1,1))

auc.roc.plot(act\_pred7)

conf\_mat1

total\_acc1

sens1

spec1

#Building Model on the undersample data#

lmod8=glm(packaging\_is\_special~., data=data\_balanced\_under)

summary(lmod8)

cof=lmod8$coefficients

OR\_vec=exp(lmod8$coefficients)

CI\_OR=exp(confint.default(lmod8))

pred\_8=predict(lmod8,newdata=val,type="response")

act\_pred8=data.frame(ID=1:nrow(val),val$packaging\_is\_special,pred\_8)

act\_pred8

conf\_mat1=cmx(act\_pred8)

total\_acc1=pcc(conf\_mat1)

sens1=sensitivity(conf\_mat1)

spec1=specificity(conf\_mat1)

auc(act\_pred8)

par(mfrow=c(1,1))

auc.roc.plot(act\_pred8)

conf\_mat1

total\_acc1

sens1

spec1

#Building Model on the both data#

lmod9=glm(packaging\_is\_special~., data=data\_balanced\_both)

summary(lmod9)

cof=lmod9$coefficients

OR\_vec=exp(lmod9$coefficients)

CI\_OR=exp(confint.default(lmod9))

pred\_9=predict(lmod9,newdata=val,type="response")

act\_pred9=data.frame(ID=1:nrow(val),val$packaging\_is\_special,pred\_9)

act\_pred9

conf\_mat1=cmx(act\_pred9)

total\_acc1=pcc(conf\_mat1)

sens1=sensitivity(conf\_mat1)

spec1=specificity(conf\_mat1)

auc(act\_pred9)

par(mfrow=c(1,1))

auc.roc.plot(act\_pred9)

conf\_mat1

total\_acc1

sens1

spec1

#Building Model on the ROSE data#

lmod10=glm(packaging\_is\_special~., data=data.rose)

summary(lmod10)

cof=lmod10$coefficients

OR\_vec=exp(lmod10$coefficients)

CI\_OR=exp(confint.default(lmod10))

pred\_10=predict(lmod10,newdata=val,type="response")

act\_pred10=data.frame(ID=1:nrow(val),val$packaging\_is\_special,pred\_10)

act\_pred10

conf\_mat1=cmx(act\_pred10)

total\_acc1=pcc(conf\_mat1)

sens1=sensitivity(conf\_mat1)

spec1=specificity(conf\_mat1)

auc(act\_pred10)

par(mfrow=c(1,1))

auc.roc.plot(act\_pred10)

conf\_mat1

total\_acc1

sens1

spec1

#Comparing the sampling method#

act\_pred\_mult=data.frame(ID=1:nrow(val),val$packaging\_is\_special,pred\_7,pred\_8,pred\_9)

auc.roc.plot(act\_pred\_mult,col=c(1,2,3),line.type=c(1,2,3),threshold = 1001, main="Packaging is Special (ROC CURVES)",legend.text=c("Oversampling", "Undersampling","Both"))

act\_pred\_mult=data.frame(ID=1:nrow(val),val$packaging\_is\_special,pred\_1,pred\_2,pred\_4,pred\_9)

auc.roc.plot(act\_pred\_mult,col=c(1,2,3,4),line.type=c(1,2.3,4),threshold = 1001, main="Packaging is Special (ROC CURVES)",legend.text=c("Model 1", "Model 2","Model 3","Model 4"))

####### DECISION TREE ############

#### DECISION TREE ####

##### ISSPECIAL ############

library(dplyr)

###### SELECTING THE MODEL THE BE USED######

df\_tree<-df %>% select(packaging\_is\_special,product\_is\_dangerous\_good,product\_is\_esp,product\_quality\_index,product\_weight,product\_x\_dim,product\_y\_dim,product\_zdim)

table(df\_tree$packaging\_is\_special)

head(df\_tree)

#building the model #######

tree <- rpart(as.factor(packaging\_is\_special) ~ ., data = train, control = rpart.control(cp = 0.0001))

dfimp<-printcp(tree)

##### We took the best model regardless of the accuaracy #####

bestcp=tree$cptable[which.min(tree$cptable[,"xerror"]),"CP"]

bestcp

##### We pruned the tree accordingly####

tree.pruned=prune(tree, cp =bestcp)

rpart.plot(tree.pruned,type=2)

#Pedicting and getting accuaracy measures#

pred11=predict(tree.pruned, newdata=val, type="prob")

pred11=pred11[,2]

act\_pred11=data.frame(ID=1:nrow(val),val$packaging\_is\_special,pred11)

act\_pred11

conf\_mat=cmx(act\_pred11)

total\_acc= pcc(conf\_mat,st.dev = FALSE)

sens=sensitivity(conf\_mat,st.dev = FALSE)# to obtain snesitivty

spec=specificity(conf\_mat,st.dev = FALSE)# to obtain specificity

accuracy\_measures=c(total\_acc,sens,spec)

names(accuracy\_measures)=c("Overall accuracy", "Sensitivity", "Specificity")

conf\_mat

total\_acc

sens

spec

accuracy\_measures

arc=auc(act\_pred11)

arc

auc.roc.plot(act\_pred11,col="red")

##### less complex norma n split less than 30 ######

table(df\_tree$packaging\_is\_special)

head(df\_tree)

#Building the model#

tree <- rpart(as.factor(packaging\_is\_special) ~ ., data = train, control = rpart.control(cp = 0.0001))

dfimp<-printcp(tree)

dfimp

#Picking the cp with ebst xerror while making sure that the split is less than 30#

tree.pruned=prune(tree, cp =dfimp[10,1])

rpart.plot(tree.pruned,type=4)

#Predicting and calculating accuaracies measure#

pred12=predict(tree.pruned, newdata=val, type="prob")

pred12=pred12[,2]

act\_pred12=data.frame(ID=1:nrow(val),val$packaging\_is\_special,pred12)

act\_pred12

conf\_mat=cmx(act\_pred12)

total\_acc= pcc(conf\_mat,st.dev = FALSE)

sens=sensitivity(conf\_mat,st.dev = FALSE)# to obtain snesitivty

spec=specificity(conf\_mat,st.dev = FALSE)# to obtain specificity

accuracy\_measures=c(total\_acc,sens,spec)

names(accuracy\_measures)=c("Overall accuracy", "Sensitivity", "Specificity")

conf\_mat

total\_acc

sens

spec

accuracy\_measures

arc=auc(act\_pred12)

arc

auc.roc.plot(act\_pred12,col="red")

##### SAMPLING TECHNIQUES ####

df\_tree<-df %>% select(packaging\_is\_special,product\_is\_dangerous\_good,product\_is\_esp,product\_quality\_index,product\_weight,product\_x\_dim,product\_y\_dim,product\_zdim)

### i added this step because I was having problem when building sampling techniques

head(df\_tree)

df\_tree <- df\_tree %>%

mutate(packaging\_is\_special = ifelse(packaging\_is\_special == 0,'No','Yes'))

df\_tree

set.seed(100)

index=sample(1:2, nrow(df\_tree), replace=TRUE, prob=c(0.7,0.3))

train=df\_tree[index==1,]

val=df\_tree[index==2,]

table(train$packaging\_is\_special)

#Oversampling#

df\_tree\_oversampling <- ovun.sample(packaging\_is\_special ~ ., data =train , method = "over",N = 6619+6619)$data

table(df\_tree\_oversampling$packaging\_is\_special)

#Undersampling#

data\_balanced\_under <- ovun.sample(packaging\_is\_special ~ ., data = train, method = "under", N = 2597+2597, seed = 1)$data

table(data\_balanced\_under$packaging\_is\_special)

#Both#

data\_balanced\_both <- ovun.sample(packaging\_is\_special ~ ., data = train, method = "both", p=0.5,N=6619+2597, seed = 100)$data

table(data\_balanced\_both$packaging\_is\_special)

#ROSE#

data.rose <- ROSE(packaging\_is\_special ~ ., data = train, seed = 100)$data

table(data.rose$packaging\_is\_special)

tree.rose <- rpart(packaging\_is\_special ~ ., data = data.rose,cp=0.0001)

dfimp<-printcp(tree.rose)

tree.pruned=prune(tree.rose, cp =dfimp[14,1])

rpart.plot(tree.pruned,type=2,

main="titanic survived\n(binary response)")

pred13=predict(tree.pruned, newdata=val, type="prob")

pred13=pred13[,2]

act\_pred13=data.frame(ID=1:nrow(val),1\*(val$packaging\_is\_special=="Yes"),pred13)

conf\_mat=cmx(act\_pred13)

total\_acc= pcc(conf\_mat,st.dev = FALSE)

sens=sensitivity(conf\_mat,st.dev = FALSE)# to obtain snesitivty

spec=specificity(conf\_mat,st.dev = FALSE)# to obtain specificity

accuracy\_measures=c(total\_acc,sens,spec)

names(accuracy\_measures)=c("Overall accuracy", "Sensitivity", "Specificity")

conf\_mat

total\_acc

sens

spec

accuracy\_measures

arc=auc(act\_pred13) #to obtain the area under Roc curve

arc

auc.roc.plot(act\_pred13,col="red")

#Building model with oversampling data

tree.over <- rpart(packaging\_is\_special ~ ., data = df\_tree\_oversampling,cp=0.0001)

dfimp<-printcp(tree.over)

dfimp[10,1]

#Picking the best xerror while making sure that the model does not exceed 30 splits#

tree.pruned=prune(tree.over, cp =dfimp[13,1])

rpart.plot(tree.pruned,type=2)

#Predidcting and getting accuracy measures#

pred14=predict(tree.pruned, newdata=val, type="prob")

pred14=pred14[,2]

act\_pred14=data.frame(ID=1:nrow(val),1\*(val$packaging\_is\_special=="Yes"),pred14)#

conf\_mat=cmx(act\_pred14)

total\_acc= pcc(conf\_mat,st.dev = FALSE)

sens=sensitivity(conf\_mat,st.dev = FALSE)# to obtain snesitivty

spec=specificity(conf\_mat,st.dev = FALSE)# to obtain specificity

accuracy\_measures=c(total\_acc,sens,spec)

names(accuracy\_measures)=c("Overall accuracy", "Sensitivity", "Specificity")

conf\_mat

total\_acc

sens

spec

accuracy\_measures

auc.roc.plot(act\_pred14,col="red")

auc=auc(act\_pred14)

auc

#Building model with under sample data#

tree.under <- rpart(packaging\_is\_special ~ ., data = data\_balanced\_under,cp=0.0001)

dfimp<-printcp(tree.under)

dfimp[14,1]

tree.pruned=prune(tree.over, cp =dfimp[14,1])

rpart.plot(tree.pruned,type=2)

#Prediciting and getting the accuracy measures

pred15=predict(tree.pruned, newdata=val, type="prob")

pred15=pred15[,2]

act\_pred15=data.frame(ID=1:nrow(val),1\*(val$packaging\_is\_special=="Yes"),pred15)

act\_pred15

conf\_mat=cmx(act\_pred15)

total\_acc= pcc(conf\_mat,st.dev = FALSE)

sens=sensitivity(conf\_mat,st.dev = FALSE)# to obtain snesitivty

spec=specificity(conf\_mat,st.dev = FALSE)# to obtain specificity

accuracy\_measures=c(total\_acc,sens,spec)

names(accuracy\_measures)=c("Overall accuracy", "Sensitivity", "Specificity")

conf\_mat

total\_acc

sens

spec

accuracy\_measures

arc=auc(act\_pred15) #to obtain the area under Roc curve

arc

auc.roc.plot(act\_pred15,col="red")

#Building the model using both technique#

tree.both <- rpart(packaging\_is\_special ~ ., data = data\_balanced\_both,cp=0.0001)

dfimp<-printcp(tree.both)

dfimp[13,1]

tree.pruned=prune(tree.over, cp =dfimp[10,1])

rpart.plot(tree.pruned,type=4,extra=2)

#Predicting and evaluating the model

pred16=predict(tree.pruned, newdata=val, type="prob")

pred16=pred16[,2]

act\_pred16=data.frame(ID=1:nrow(val),1\*(val$packaging\_is\_special=="Yes"),pred16)

act\_pred16

conf\_mat=cmx(act\_pred16)

total\_acc= pcc(conf\_mat,st.dev = FALSE)

sens=sensitivity(conf\_mat,st.dev = FALSE)# to obtain snesitivty

spec=specificity(conf\_mat,st.dev = FALSE)# to obtain specificity

accuracy\_measures=c(total\_acc,sens,spec)

names(accuracy\_measures)=c("Overall accuracy", "Sensitivity", "Specificity")

conf\_mat

total\_acc

sens

spec

accuracy\_measures

arc=auc(act\_pred16) #to obtain the area under Roc curve

arc

auc.roc.plot(act\_pred16,col="red")

#### COMPARING THE SAMPLING TECHNIQUE ROC CURVES###

act\_pred\_mult=data.frame(ID=1:nrow(val),1\*(val$packaging\_is\_special=="Yes"),pred14,pred15,pred16)

auc.roc.plot(act\_pred\_mult,col=c(1,2,3),line.type=c(1,2,3),threshold = 1001, main="Packaging is Special (ROC CURVES)",legend.text=c("Oversampling", "Undersampling","Both"))

act\_pred\_mult=data.frame(ID=1:nrow(val),1\*(val$packaging\_is\_special=="Yes"),pred11,pred12,pred14)

auc.roc.plot(act\_pred\_mult,col=c(5,6,7),line.type=c(5,6,7),threshold = 1001, main="Packaging is Special (ROC CURVES)",legend.text=c("Model 5", "Model 6","Model 7"))

act\_pred\_mult=data.frame(ID=1:nrow(val),1\*(val$packaging\_is\_special=="Yes"),pred14,pred15,pred16)

auc.roc.plot(act\_pred\_mult,col=c(1,2,3),line.type=c(1,2,3),threshold = 1001, main="Packaging is Special (ROC CURVES)",legend.text=c("Oversampling", "Undersampling","Both"))

########### FINAL VERDICT ####################

act\_pred\_mult=data.frame(ID=1:nrow(val),1\*(val$packaging\_is\_special=="Yes"),pred\_3,pred14)

act\_pred\_mult

auc.roc.plot(act\_pred\_mult,col=c(1,2),line.type=c(1,2),threshold = 1001, main="ROC Curves",legend.text=c("Model 3", "Model 7"))

act\_pred\_mult=data.frame(ID=1:nrow(val),1\*(val$packaging\_is\_special=="Yes"),pred\_1,pred\_2,pred\_4,pred\_9,pred11,pred12,pred14)

auc.roc.plot(act\_pred\_mult,col=c(1,2,3,4,5,6,7),line.type=c(1,2,3,4,5,6,7),threshold = 1001,

main="Packaging is sepcial (ROC CURVES)",legend.text=c("Model 1", "Model 2","Model 3","Model 4","Model 5","Model 6","Model 7"))

#Load Unit X Dimension

LinearModel1 = lm(load\_unit\_x\_dim~., data = train)

summary(LinearModel1)

pred\_linearModel1 = predict(LinearModel1, val)

perform\_linearModel1 = acc\_error(val$load\_unit\_x\_dim, pred\_linearModel1)

perform\_linearModel1

# LinearModel2 by removing insignificant variables

# Remove insignificant variables

x\_dim\_loadUnit = subset(loadUnit, select = -c(load\_unit\_y\_dim, load\_unit\_z\_dim, load\_unit\_weight, packagings\_per\_load\_unit,

load\_unit\_volume, product\_kogr\_number, product\_quality\_index, product\_y\_dim,

product\_generic\_family\_name, product\_module\_number, product\_supplier\_number,

product\_zdim, product\_is\_dangerous\_good, product\_name, product\_weight,

product\_is\_esp, product\_number, product\_x\_dim))

attach(x\_dim\_loadUnit)

x\_dim\_loadUnit = subset(x\_dim\_loadUnit, select = -c(packaging\_raw\_material\_name, packaging\_is\_oneway))

sum(is.na(loadUnit))

set.seed(100)

split = sample(1:2, nrow(x\_dim\_loadUnit), replace = TRUE, prob = c(0.7, 0.3))

train = x\_dim\_loadUnit[split ==1, ]

val = x\_dim\_loadUnit[split == 2,]

LinearModel2 = lm(load\_unit\_x\_dim~., data = train)

summary(LinearModel2)

pred\_linearModel2 = predict(LinearModel2, val)

perform\_linearModel2 = acc\_error(val$load\_unit\_x\_dim, pred\_linearModel2)

perform\_linearModel2

# LinearModel3 by removing packaging\_raw\_material\_name

# Remove insignificant variable packaging\_raw\_material\_name

x\_dim\_loadUnit = subset(loadUnit, select = -c(load\_unit\_y\_dim, load\_unit\_z\_dim, load\_unit\_weight, packagings\_per\_load\_unit,

load\_unit\_volume, product\_kogr\_number, product\_quality\_index, product\_y\_dim,

product\_generic\_family\_name, product\_module\_number, product\_supplier\_number,

product\_zdim, product\_is\_dangerous\_good, product\_name, product\_weight,

product\_is\_esp, product\_number, product\_x\_dim))

attach(x\_dim\_loadUnit)

x\_dim\_loadUnit = subset(x\_dim\_loadUnit, select = -c(packaging\_raw\_material\_name))

sum(is.na(loadUnit))

set.seed(100)

split = sample(1:2, nrow(x\_dim\_loadUnit), replace = TRUE, prob = c(0.7, 0.3))

train = x\_dim\_loadUnit[split ==1, ]

val = x\_dim\_loadUnit[split == 2,]

LinearModel3 = lm(load\_unit\_x\_dim~., data = train)

summary(LinearModel3)

pred\_linearModel3 = predict(LinearModel3, val)

perform\_linearModel3 = acc\_error(val$load\_unit\_x\_dim, pred\_linearModel3)

perform\_linearModel3

# LinearModel4 by removing categorical variables

# Remove insignificant variables packaging\_raw\_material\_name

x\_dim\_loadUnit = subset(loadUnit, select = -c(load\_unit\_y\_dim, load\_unit\_z\_dim, load\_unit\_weight, packagings\_per\_load\_unit,

load\_unit\_volume, product\_kogr\_number, product\_quality\_index, product\_y\_dim,

product\_generic\_family\_name, product\_module\_number, product\_supplier\_number,

product\_zdim, product\_is\_dangerous\_good, product\_name, product\_weight,

products\_per\_packaging, product\_is\_esp, product\_number, product\_x\_dim))

attach(x\_dim\_loadUnit)

x\_dim\_loadUnit = subset(x\_dim\_loadUnit, select = -c(packaging\_raw\_material\_name, packaging\_is\_oneway, packaging\_is\_special))

sum(is.na(loadUnit))

set.seed(100)

split = sample(1:2, nrow(x\_dim\_loadUnit), replace = TRUE, prob = c(0.7, 0.3))

train = x\_dim\_loadUnit[split ==1, ]

val = x\_dim\_loadUnit[split == 2,]

LinearModel4 = lm(load\_unit\_x\_dim~., data = train)

summary(LinearModel4)

pred\_linearModel4 = predict(LinearModel4, val)

perform\_linearModel4 = acc\_error(val$load\_unit\_x\_dim, pred\_linearModel4)

perform\_linearModel4

library(rpart)

library(rpart.plot)

# TreeModel1 with all predictors

# Remove other numerical dependent variables and product variables

x\_dim\_loadUnit = subset(loadUnit, select = -c(load\_unit\_y\_dim, load\_unit\_z\_dim, load\_unit\_weight, packagings\_per\_load\_unit,

load\_unit\_volume, product\_kogr\_number, product\_quality\_index, product\_y\_dim,

product\_generic\_family\_name, product\_module\_number, product\_supplier\_number,

product\_zdim, product\_is\_dangerous\_good, product\_name, product\_weight,

product\_is\_esp, product\_number, product\_x\_dim))

attach(x\_dim\_loadUnit)

colnames(x\_dim\_loadUnit)

sum(is.na(loadUnit))

set.seed(100)

split = sample(1:2, nrow(x\_dim\_loadUnit), replace = TRUE, prob = c(0.7, 0.3))

train = x\_dim\_loadUnit[split ==1, ]

val = x\_dim\_loadUnit[split == 2,]

treeModel1 = rpart(load\_unit\_x\_dim~. , data = train, control = rpart.control(cp = 0.0001))

bestcp=treeModel1$cptable[which.min(treeModel1$cptable[,"xerror"]),"CP"]

treeModel1\_pruned=prune(treeModel1, cp = bestcp)

pred\_treeModel1\_pruned = predict(treeModel1\_pruned, val)

perform\_treeModel1\_pruned = acc\_error(val$load\_unit\_x\_dim, pred\_treeModel1\_pruned)

perform\_treeModel1\_pruned

printcp(treeModel1\_pruned)

#prune again for better splits

treeModel2\_pruned=prune(treeModel1\_pruned, cp = 0.00085896)

pred\_treeModel2\_pruned = predict(treeModel2\_pruned, val)

perform\_treeModel2\_pruned = acc\_error(val$load\_unit\_x\_dim, pred\_treeModel2\_pruned)

perform\_treeModel2\_pruned

printcp(treeModel2\_pruned)

#Pick best model

perf\_comp\_linear = rbind(perform\_linearModel1, perform\_linearModel2, perform\_linearModel3, perform\_linearModel4)

numb\_predictors = c(26,7,8, 6)

cbind(perf\_comp\_linear, numb\_predictors)

perf\_comp\_tree = rbind(perform\_treeModel1\_pruned,perform\_treeModel2\_pruned)

numb\_splits = c(32,20)

cp = c(bestcp,0.00029044)

cbind(perf\_comp\_tree, numb\_splits,cp)

mean(load\_unit\_x\_dim)

min(load\_unit\_x\_dim)

max(load\_unit\_x\_dim)

#Load unit Y dimn

par(mfrow=c(1,1))

rpart.plot(treeModel2\_pruned,type=4,extra="auto",

main="Load Unit x Dimension (in numerical)")

printcp(treeModel2\_pruned)

attach(x\_dim\_loadUnit)

par(mfrow=c(3,4))

library(visreg)

visreg(treeModel2\_pruned)

#Load unit Y dimens

# LinearModel 1 with all predictors

attach(x\_dim\_loadUnit)

sum(is.na(loadUnit))

set.seed(100)

split = sample(1:2, nrow(x\_dim\_loadUnit), replace = TRUE, prob = c(0.7, 0.3))

train = x\_dim\_loadUnit[split ==1, ]

val = x\_dim\_loadUnit[split == 2,]

LinearModel1 = lm(load\_unit\_y\_dim~., data = train)

summary(LinearModel1)

pred\_linearModel1 = predict(LinearModel1, val)

perform\_linearModel1 = acc\_error(val$load\_unit\_y\_dim, pred\_linearModel1)

perform\_linearModel1

# LinearModel2 by removing insignificant variables

# Remove insignificant variables

x\_dim\_loadUnit = subset(loadUnit, select = -c(load\_unit\_x\_dim, load\_unit\_z\_dim, load\_unit\_weight, packagings\_per\_load\_unit,

load\_unit\_volume, product\_kogr\_number, product\_quality\_index, product\_y\_dim,

product\_generic\_family\_name, product\_module\_number, product\_supplier\_number,

product\_zdim, product\_is\_dangerous\_good, product\_name, product\_weight,

product\_is\_esp, product\_number, product\_x\_dim))

attach(x\_dim\_loadUnit)

x\_dim\_loadUnit = subset(x\_dim\_loadUnit, select = -c(packaging\_raw\_material\_name, packaging\_is\_oneway))

sum(is.na(loadUnit))

set.seed(100)

split = sample(1:2, nrow(x\_dim\_loadUnit), replace = TRUE, prob = c(0.7, 0.3))

train = x\_dim\_loadUnit[split ==1, ]

val = x\_dim\_loadUnit[split == 2,]

LinearModel2 = lm(load\_unit\_y\_dim~., data = train)

summary(LinearModel2)

pred\_linearModel2 = predict(LinearModel2, val)

perform\_linearModel2 = acc\_error(val$load\_unit\_y\_dim, pred\_linearModel2)

perform\_linearModel2

# LinearModel3 by removing packaging\_raw\_material\_name

# Remove insignificant variable packaging\_raw\_material\_name

x\_dim\_loadUnit = subset(loadUnit, select = -c(load\_unit\_x\_dim, load\_unit\_z\_dim, load\_unit\_weight, packagings\_per\_load\_unit,

load\_unit\_volume, product\_kogr\_number, product\_quality\_index, product\_y\_dim,

product\_generic\_family\_name, product\_module\_number, product\_supplier\_number,

product\_zdim, product\_is\_dangerous\_good, product\_name, product\_weight,

product\_is\_esp, product\_number, product\_x\_dim))

attach(x\_dim\_loadUnit)

x\_dim\_loadUnit = subset(x\_dim\_loadUnit, select = -c(packaging\_raw\_material\_name))

sum(is.na(loadUnit))

set.seed(100)

split = sample(1:2, nrow(x\_dim\_loadUnit), replace = TRUE, prob = c(0.7, 0.3))

train = x\_dim\_loadUnit[split ==1, ]

val = x\_dim\_loadUnit[split == 2,]

LinearModel3 = lm(load\_unit\_y\_dim~., data = train)

summary(LinearModel3)

pred\_linearModel3 = predict(LinearModel3, val)

perform\_linearModel3 = acc\_error(val$load\_unit\_y\_dim, pred\_linearModel3)

perform\_linearModel3

# LinearModel4 by removing categorical variables and packaging\_weight since it is insignificant

# Remove insignificant variables packaging\_raw\_material\_name

x\_dim\_loadUnit = subset(loadUnit, select = -c(load\_unit\_x\_dim, load\_unit\_z\_dim, load\_unit\_weight, packagings\_per\_load\_unit,

load\_unit\_volume, product\_kogr\_number, product\_quality\_index, product\_y\_dim,

product\_generic\_family\_name, product\_module\_number, product\_supplier\_number,

product\_zdim, product\_is\_dangerous\_good, product\_name, product\_weight,

product\_is\_esp, product\_number, product\_x\_dim))

attach(x\_dim\_loadUnit)

x\_dim\_loadUnit = subset(x\_dim\_loadUnit, select = -c(packaging\_weight,packaging\_raw\_material\_name, packaging\_is\_oneway, packaging\_is\_special))

sum(is.na(loadUnit))

set.seed(100)

split = sample(1:2, nrow(x\_dim\_loadUnit), replace = TRUE, prob = c(0.7, 0.3))

train = x\_dim\_loadUnit[split ==1, ]

val = x\_dim\_loadUnit[split == 2,]

LinearModel4 = lm(load\_unit\_y\_dim~., data = train)

summary(LinearModel4)

pred\_linearModel4 = predict(LinearModel4, val)

perform\_linearModel4 = acc\_error(val$load\_unit\_y\_dim, pred\_linearModel4)

perform\_linearModel4

library(rpart)

library(rpart.plot)

# TreeModel1 with all predictors

# Remove other numerical dependent variables and product variables

x\_dim\_loadUnit = subset(loadUnit, select = -c(load\_unit\_x\_dim, load\_unit\_z\_dim, load\_unit\_weight, packagings\_per\_load\_unit,

load\_unit\_volume, product\_kogr\_number, product\_quality\_index, product\_y\_dim,

product\_generic\_family\_name, product\_module\_number, product\_supplier\_number,

product\_zdim, product\_is\_dangerous\_good, product\_name, product\_weight,

product\_is\_esp, product\_number, product\_x\_dim))

attach(x\_dim\_loadUnit)

colnames(x\_dim\_loadUnit)

sum(is.na(loadUnit))

set.seed(100)

split = sample(1:2, nrow(x\_dim\_loadUnit), replace = TRUE, prob = c(0.7, 0.3))

train = x\_dim\_loadUnit[split ==1, ]

val = x\_dim\_loadUnit[split == 2,]

treeModel1 = rpart(load\_unit\_y\_dim~. , data = train, control = rpart.control(cp = 0.0001))

bestcp=treeModel1$cptable[which.min(treeModel1$cptable[,"xerror"]),"CP"]

treeModel1\_pruned=prune(treeModel1, cp = bestcp)

pred\_treeModel1\_pruned = predict(treeModel1\_pruned, val)

perform\_treeModel1\_pruned = acc\_error(val$load\_unit\_y\_dim, pred\_treeModel1\_pruned)

perform\_treeModel1\_pruned

printcp(treeModel1\_pruned)

#prune again for better splits

treeModel2\_pruned=prune(treeModel1\_pruned, cp = 0.00140627)

pred\_treeModel2\_pruned = predict(treeModel2\_pruned, val)

perform\_treeModel2\_pruned = acc\_error(val$load\_unit\_y\_dim, pred\_treeModel2\_pruned)

perform\_treeModel2\_pruned

printcp(treeModel2\_pruned)

#Pick best model

perf\_comp\_linear = rbind(perform\_linearModel1, perform\_linearModel2, perform\_linearModel3, perform\_linearModel4)

numb\_predictors = c(26,7,8, 5)

cbind(perf\_comp\_linear, numb\_predictors)

perf\_comp\_tree = rbind(perform\_treeModel1\_pruned, perform\_treeModel2\_pruned)

numb\_splits = c(34,19)

cp = c(bestcp,0.00029881)

cbind(perf\_comp\_tree, numb\_splits,cp)

mean(load\_unit\_y\_dim)

min(load\_unit\_y\_dim)

max(load\_unit\_y\_dim)

par(mfrow=c(1,1))

rpart.plot(treeModel2\_pruned,type=4,extra="auto",

main="Load Unit y Dimension (in numerical)")

par(mfrow=c(3,4))

library(visreg)

visreg(treeModel2\_pruned)

#Load unit Z dimension

LinearModel1 = lm(load\_unit\_z\_dim~., data = train)

summary(LinearModel1)

pred\_linearModel1 = predict(LinearModel1, val)

perform\_linearModel1 = acc\_error(val$load\_unit\_z\_dim, pred\_linearModel1)

perform\_linearModel1

# LinearModel2 by removing insignificant variables

# Remove insignificant variables

x\_dim\_loadUnit = subset(loadUnit, select = -c(load\_unit\_x\_dim, load\_unit\_y\_dim, load\_unit\_weight, packagings\_per\_load\_unit,

load\_unit\_volume, product\_kogr\_number, product\_quality\_index, product\_y\_dim,

product\_generic\_family\_name, product\_module\_number, product\_supplier\_number,

product\_zdim, product\_is\_dangerous\_good, product\_name, product\_weight,

product\_is\_esp, product\_number, product\_x\_dim))

attach(x\_dim\_loadUnit)

x\_dim\_loadUnit = subset(x\_dim\_loadUnit, select = -c(packaging\_raw\_material\_name, packaging\_is\_oneway, packaging\_y\_dim))

sum(is.na(loadUnit))

set.seed(100)

split = sample(1:2, nrow(x\_dim\_loadUnit), replace = TRUE, prob = c(0.7, 0.3))

train = x\_dim\_loadUnit[split ==1, ]

val = x\_dim\_loadUnit[split == 2,]

LinearModel2 = lm(load\_unit\_z\_dim~., data = train)

summary(LinearModel2)

pred\_linearModel2 = predict(LinearModel2, val)

perform\_linearModel2 = acc\_error(val$load\_unit\_z\_dim, pred\_linearModel2)

perform\_linearModel2

# LinearModel3 by removing packaging\_raw\_material\_name

# Remove insignificant variable packaging\_raw\_material\_name

x\_dim\_loadUnit = subset(loadUnit, select = -c(load\_unit\_x\_dim, load\_unit\_y\_dim, load\_unit\_weight, packagings\_per\_load\_unit,

load\_unit\_volume, product\_kogr\_number, product\_quality\_index, product\_y\_dim,

product\_generic\_family\_name, product\_module\_number, product\_supplier\_number,

product\_zdim, product\_is\_dangerous\_good, product\_name, product\_weight,

product\_is\_esp, product\_number, product\_x\_dim))

attach(x\_dim\_loadUnit)

x\_dim\_loadUnit = subset(x\_dim\_loadUnit, select = -c(packaging\_raw\_material\_name))

sum(is.na(loadUnit))

set.seed(100)

split = sample(1:2, nrow(x\_dim\_loadUnit), replace = TRUE, prob = c(0.7, 0.3))

train = x\_dim\_loadUnit[split ==1, ]

val = x\_dim\_loadUnit[split == 2,]

LinearModel3 = lm(load\_unit\_z\_dim~., data = train)

summary(LinearModel3)

pred\_linearModel3 = predict(LinearModel3, val)

perform\_linearModel3 = acc\_error(val$load\_unit\_z\_dim, pred\_linearModel3)

perform\_linearModel3

# LinearModel4 by removing categorical variables and packaging\_volume and packaging\_weight since it is insignificant

# Remove insignificant variables packaging\_raw\_material\_name

x\_dim\_loadUnit = subset(loadUnit, select = -c(load\_unit\_x\_dim, load\_unit\_y\_dim, load\_unit\_weight, packagings\_per\_load\_unit,

load\_unit\_volume, product\_kogr\_number, product\_quality\_index, product\_y\_dim,

product\_generic\_family\_name, product\_module\_number, product\_supplier\_number,

product\_zdim, product\_is\_dangerous\_good, product\_name, product\_weight,

product\_is\_esp, product\_number, product\_x\_dim))

attach(x\_dim\_loadUnit)

x\_dim\_loadUnit = subset(x\_dim\_loadUnit, select = -c(packaging\_volume,packaging\_weight,packaging\_raw\_material\_name, packaging\_is\_oneway, packaging\_is\_special))

sum(is.na(loadUnit))

set.seed(100)

split = sample(1:2, nrow(x\_dim\_loadUnit), replace = TRUE, prob = c(0.7, 0.3))

train = x\_dim\_loadUnit[split ==1, ]

val = x\_dim\_loadUnit[split == 2,]

LinearModel4 = lm(load\_unit\_z\_dim~., data = train)

summary(LinearModel4)

pred\_linearModel4 = predict(LinearModel4, val)

perform\_linearModel4 = acc\_error(val$load\_unit\_z\_dim, pred\_linearModel4)

perform\_linearModel4

library(rpart)

library(rpart.plot)

# TreeModel1 with all predictors

# Remove other numerical dependent variables and product variables

attach(loadUnit)

x\_dim\_loadUnit = subset(loadUnit, select = -c(load\_unit\_x\_dim, load\_unit\_y\_dim, load\_unit\_weight, packagings\_per\_load\_unit,

load\_unit\_volume, product\_kogr\_number, product\_quality\_index, product\_y\_dim,

product\_generic\_family\_name, product\_module\_number, product\_supplier\_number,

product\_zdim, product\_is\_dangerous\_good, product\_name, product\_weight,

product\_is\_esp, product\_number, product\_x\_dim))

attach(x\_dim\_loadUnit)

colnames(x\_dim\_loadUnit)

sum(is.na(loadUnit))

set.seed(100)

split = sample(1:2, nrow(x\_dim\_loadUnit), replace = TRUE, prob = c(0.7, 0.3))

train = x\_dim\_loadUnit[split ==1, ]

val = x\_dim\_loadUnit[split == 2,]

treeModel1 = rpart(load\_unit\_z\_dim~. , data = train, control = rpart.control(cp = 0.0001))

bestcp=treeModel1$cptable[which.min(treeModel1$cptable[,"xerror"]),"CP"]

treeModel1\_pruned=prune(treeModel1, cp = bestcp)

pred\_treeModel1\_pruned = predict(treeModel1\_pruned, val)

perform\_treeModel1\_pruned = acc\_error(val$load\_unit\_z\_dim, pred\_treeModel1\_pruned)

perform\_treeModel1\_pruned

printcp(treeModel1\_pruned)

#prune again for better splits

treeModel2\_pruned=prune(treeModel1\_pruned, cp = 0.00049109)

pred\_treeModel2\_pruned = predict(treeModel2\_pruned, val)

perform\_treeModel2\_pruned = acc\_error(val$load\_unit\_z\_dim, pred\_treeModel2\_pruned)

perform\_treeModel2\_pruned

printcp(treeModel2\_pruned)

library(randomForest)

randomTree1 = randomForest(load\_unit\_z\_dim~., data = train, mtry = 3, ntree=150, importance = TRUE, na.action = na.omit)

pred\_randomTree1 = predict(randomTree1, val)

perform\_randomTree1 = acc\_error(val$load\_unit\_z\_dim, pred\_randomTree1)

perform\_randomTree1

print(randomTree1)

#Pick best model

perf\_comp\_linear = rbind(perform\_linearModel1, perform\_linearModel2, perform\_linearModel3, perform\_linearModel4)

numb\_predictors = c(26,7,8, 5)

cbind(perf\_comp\_linear, numb\_predictors)

perf\_comp\_tree = rbind(perform\_treeModel1\_pruned, perform\_treeModel2\_pruned)

numb\_splits = c(67,42)

cp = c(bestcp,0.00049109)

cbind(perf\_comp\_tree, numb\_splits,cp)

perf\_comp\_randomTree = rbind(perform\_randomTree1)

numb\_trees = c(150)

cbind(perf\_comp\_randomTree, numb\_trees)

mean(load\_unit\_z\_dim)

min(load\_unit\_z\_dim)

max(load\_unit\_z\_dim)

par(mfrow=c(3,4))

library(visreg)

visreg(randomTree1)

#Load unit Weight

LinearModel1 = lm(load\_unit\_weight~., data = train)

summary(LinearModel1)

pred\_linearModel1 = predict(LinearModel1, val)

perform\_linearModel1 = acc\_error(val$load\_unit\_weight, pred\_linearModel1)

perform\_linearModel1

# LinearModel2 by removing insignificant variables

# Remove insignificant variables

x\_dim\_loadUnit = subset(loadUnit, select = -c(load\_unit\_x\_dim, load\_unit\_y\_dim, load\_unit\_z\_dim, packagings\_per\_load\_unit,

load\_unit\_volume, product\_kogr\_number, product\_quality\_index, product\_y\_dim,

product\_generic\_family\_name, product\_module\_number, product\_supplier\_number,

product\_zdim, product\_is\_dangerous\_good, product\_name, product\_weight,

product\_is\_esp, product\_number, product\_x\_dim))

attach(x\_dim\_loadUnit)

x\_dim\_loadUnit = subset(x\_dim\_loadUnit, select = -c(packaging\_raw\_material\_name, packaging\_is\_oneway, packaging\_volume))

sum(is.na(loadUnit))

set.seed(100)

split = sample(1:2, nrow(x\_dim\_loadUnit), replace = TRUE, prob = c(0.7, 0.3))

train = x\_dim\_loadUnit[split ==1, ]

val = x\_dim\_loadUnit[split == 2,]

LinearModel2 = lm(load\_unit\_weight~., data = train)

summary(LinearModel2)

pred\_linearModel2 = predict(LinearModel2, val)

perform\_linearModel2 = acc\_error(val$load\_unit\_weight, pred\_linearModel2)

perform\_linearModel2

# LinearModel3 by removing packaging\_raw\_material\_name

# Remove insignificant variable packaging\_raw\_material\_name

x\_dim\_loadUnit = subset(loadUnit, select = -c(load\_unit\_x\_dim, load\_unit\_y\_dim, load\_unit\_z\_dim, packagings\_per\_load\_unit,

load\_unit\_volume, product\_kogr\_number, product\_quality\_index, product\_y\_dim,

product\_generic\_family\_name, product\_module\_number, product\_supplier\_number,

product\_zdim, product\_is\_dangerous\_good, product\_name, product\_weight,

product\_is\_esp, product\_number, product\_x\_dim))

attach(x\_dim\_loadUnit)

x\_dim\_loadUnit = subset(x\_dim\_loadUnit, select = -c(packaging\_raw\_material\_name))

sum(is.na(loadUnit))

set.seed(100)

split = sample(1:2, nrow(x\_dim\_loadUnit), replace = TRUE, prob = c(0.7, 0.3))

train = x\_dim\_loadUnit[split ==1, ]

val = x\_dim\_loadUnit[split == 2,]

LinearModel3 = lm(load\_unit\_weight~., data = train)

summary(LinearModel3)

pred\_linearModel3 = predict(LinearModel3, val)

perform\_linearModel3 = acc\_error(val$load\_unit\_weight, pred\_linearModel3)

perform\_linearModel3

# LinearModel4 by removing categorical variables and packaging\_y\_dim since it is insignificant

# Remove insignificant variables packaging\_raw\_material\_name

x\_dim\_loadUnit = subset(loadUnit, select = -c(load\_unit\_x\_dim, load\_unit\_y\_dim, load\_unit\_z\_dim, packagings\_per\_load\_unit,

load\_unit\_volume, product\_kogr\_number, product\_quality\_index, product\_y\_dim,

product\_generic\_family\_name, product\_module\_number, product\_supplier\_number,

product\_zdim, product\_is\_dangerous\_good, product\_name, product\_weight,

product\_is\_esp, product\_number, product\_x\_dim))

attach(x\_dim\_loadUnit)

x\_dim\_loadUnit = subset(x\_dim\_loadUnit, select = -c(packaging\_y\_dim,packaging\_raw\_material\_name, packaging\_is\_oneway, packaging\_is\_special))

sum(is.na(loadUnit))

set.seed(100)

split = sample(1:2, nrow(x\_dim\_loadUnit), replace = TRUE, prob = c(0.7, 0.3))

train = x\_dim\_loadUnit[split ==1, ]

val = x\_dim\_loadUnit[split == 2,]

LinearModel4 = lm(load\_unit\_weight~., data = train)

summary(LinearModel4)

pred\_linearModel4 = predict(LinearModel4, val)

perform\_linearModel4 = acc\_error(val$load\_unit\_weight, pred\_linearModel4)

perform\_linearModel4

library(rpart)

library(rpart.plot)

# TreeModel1 with all predictors

# Remove other numerical dependent variables and product variables

x\_dim\_loadUnit = subset(loadUnit, select = -c(load\_unit\_x\_dim, load\_unit\_y\_dim, load\_unit\_z\_dim, packagings\_per\_load\_unit,

load\_unit\_volume, product\_kogr\_number, product\_quality\_index, product\_y\_dim,

product\_generic\_family\_name, product\_module\_number, product\_supplier\_number,

product\_zdim, product\_is\_dangerous\_good, product\_name, product\_weight,

product\_is\_esp, product\_number, product\_x\_dim))

attach(x\_dim\_loadUnit)

colnames(x\_dim\_loadUnit)

sum(is.na(loadUnit))

set.seed(100)

split = sample(1:2, nrow(x\_dim\_loadUnit), replace = TRUE, prob = c(0.7, 0.3))

train = x\_dim\_loadUnit[split ==1, ]

val = x\_dim\_loadUnit[split == 2,]

treeModel1 = rpart(load\_unit\_weight~. , data = train, control = rpart.control(cp = 0.0001))

bestcp=treeModel1$cptable[which.min(treeModel1$cptable[,"xerror"]),"CP"]

treeModel1\_pruned=prune(treeModel1, cp = bestcp)

pred\_treeModel1\_pruned = predict(treeModel1\_pruned, val)

perform\_treeModel1\_pruned = acc\_error(val$load\_unit\_weight, pred\_treeModel1\_pruned)

perform\_treeModel1\_pruned

printcp(treeModel1\_pruned)

#prune again for better splits

treeModel2\_pruned=prune(treeModel1\_pruned, cp = 0.00023849)

pred\_treeModel2\_pruned = predict(treeModel2\_pruned, val)

perform\_treeModel2\_pruned = acc\_error(val$load\_unit\_weight, pred\_treeModel2\_pruned)

perform\_treeModel2\_pruned

printcp(treeModel2\_pruned)

library(randomForest)

randomTree1 = randomForest(load\_unit\_weight~., data = train, mtry = 3, ntree=155, importance = TRUE, na.action = na.omit)

pred\_randomTree1 = predict(randomTree1, val)

perform\_randomTree1 = acc\_error(val$load\_unit\_weight, pred\_randomTree1)

perform\_randomTree1

print(randomTree1)

#Pick best model

perf\_comp\_linear = rbind(perform\_linearModel1, perform\_linearModel2, perform\_linearModel3, perform\_linearModel4)

numb\_predictors = c(26,6,8,5)

cbind(perf\_comp\_linear, numb\_predictors)

perf\_comp\_tree = rbind(perform\_treeModel1\_pruned, perform\_treeModel2\_pruned)

numb\_splits = c(60,38)

cp = c(bestcp,0.00029476)

cbind(perf\_comp\_tree, numb\_splits,cp)

perf\_comp\_randomTree = rbind(perform\_randomTree1)

numb\_trees = c(155)

cbind(perf\_comp\_randomTree, numb\_trees)

mean(load\_unit\_weight)

min(load\_unit\_weight)

max(load\_unit\_weight)

par(mfrow=c(1,1))

par(mfrow=c(3,4))

library(visreg)

visreg(randomTree3)

#Load unit per load unit

LinearModel1 = lm(packagings\_per\_load\_unit~., data = train)

summary(LinearModel1)

pred\_linearModel1 = predict(LinearModel1, val)

perform\_linearModel1 = acc\_error(val$packagings\_per\_load\_unit, pred\_linearModel1)

perform\_linearModel1

# LinearModel2 by removing insignificant variables

# Remove insignificant variables

attach(loadUnit)

x\_dim\_loadUnit = subset(loadUnit, select = -c(load\_unit\_x\_dim, load\_unit\_y\_dim, load\_unit\_z\_dim, load\_unit\_weight,

load\_unit\_volume, product\_kogr\_number, product\_quality\_index, product\_y\_dim,

product\_generic\_family\_name, product\_module\_number, product\_supplier\_number,

product\_zdim, product\_is\_dangerous\_good, product\_name, product\_weight,

product\_is\_esp, product\_number, product\_x\_dim, packagings\_per\_load\_unit))

attach(x\_dim\_loadUnit)

x\_dim\_loadUnit["packaging\_x\_dim"] = log(packaging\_x\_dim)

x\_dim\_loadUnit["packaging\_y\_dim"] = log(packaging\_y\_dim)

x\_dim\_loadUnit["packaging\_z\_dim"] = log(packaging\_z\_dim)

x\_dim\_loadUnit["packaging\_volume"] = log(packaging\_volume)

x\_dim\_loadUnit["packaging\_load\_capacity"] = log(packaging\_load\_capacity)

x\_dim\_loadUnit["packaging\_weight"] = log(packaging\_weight)

x\_dim\_loadUnit["products\_per\_packaging"] = log(products\_per\_packaging)

x\_dim\_loadUnit["packagings\_per\_load\_unit"] = log(packagings\_per\_load\_unit)

attach(x\_dim\_loadUnit)

x\_dim\_loadUnit = subset(x\_dim\_loadUnit, select = -c(packaging\_raw\_material\_name))

sum(is.na(loadUnit))

set.seed(100)

split = sample(1:2, nrow(x\_dim\_loadUnit), replace = TRUE, prob = c(0.7, 0.3))

train = x\_dim\_loadUnit[split ==1, ]

val = x\_dim\_loadUnit[split == 2,]

LinearModel2 = lm(packagings\_per\_load\_unit~., data = train)

summary(LinearModel2)

pred\_linearModel2 = predict(LinearModel2, val)

perform\_linearModel2 = acc\_error(val$packagings\_per\_load\_unit, pred\_linearModel2)

perform\_linearModel2

# LinearModel3 by removing packaging\_raw\_material\_name

# Remove insignificant variable packaging\_raw\_material\_name

attach(loadUnit)

x\_dim\_loadUnit = subset(loadUnit, select = -c(load\_unit\_x\_dim, load\_unit\_y\_dim, load\_unit\_z\_dim, load\_unit\_weight,

load\_unit\_volume, product\_kogr\_number, product\_quality\_index, product\_y\_dim,

product\_generic\_family\_name, product\_module\_number, product\_supplier\_number,

product\_zdim, product\_is\_dangerous\_good, product\_name, product\_weight,

product\_is\_esp, product\_number, product\_x\_dim, packagings\_per\_load\_unit))

attach(x\_dim\_loadUnit)

x\_dim\_loadUnit["packaging\_x\_dim"] = log(packaging\_x\_dim)

x\_dim\_loadUnit["packaging\_y\_dim"] = log(packaging\_y\_dim)

x\_dim\_loadUnit["packaging\_z\_dim"] = log(packaging\_z\_dim)

x\_dim\_loadUnit["packaging\_volume"] = log(packaging\_volume)

x\_dim\_loadUnit["packaging\_load\_capacity"] = log(packaging\_load\_capacity)

x\_dim\_loadUnit["packaging\_weight"] = log(packaging\_weight)

x\_dim\_loadUnit["products\_per\_packaging"] = log(products\_per\_packaging)

x\_dim\_loadUnit["packagings\_per\_load\_unit"] = log(packagings\_per\_load\_unit)

attach(x\_dim\_loadUnit)

x\_dim\_loadUnit = subset(x\_dim\_loadUnit, select = -c(packaging\_raw\_material\_name))

sum(is.na(loadUnit))

set.seed(100)

split = sample(1:2, nrow(x\_dim\_loadUnit), replace = TRUE, prob = c(0.7, 0.3))

train = x\_dim\_loadUnit[split ==1, ]

val = x\_dim\_loadUnit[split == 2,]

LinearModel3 = lm(packagings\_per\_load\_unit~., data = train)

summary(LinearModel3)

pred\_linearModel3 = predict(LinearModel3, val)

perform\_linearModel3 = acc\_error(val$packagings\_per\_load\_unit, pred\_linearModel3)

perform\_linearModel3

# LinearModel4 by removing categorical variables

# Remove insignificant variables packaging\_raw\_material\_name

attach(loadUnit)

x\_dim\_loadUnit = subset(loadUnit, select = -c(load\_unit\_x\_dim, load\_unit\_y\_dim, load\_unit\_z\_dim, load\_unit\_weight,

load\_unit\_volume, product\_kogr\_number, product\_quality\_index, product\_y\_dim,

product\_generic\_family\_name, product\_module\_number, product\_supplier\_number,

product\_zdim, product\_is\_dangerous\_good, product\_name, product\_weight,

product\_is\_esp, product\_number, product\_x\_dim, packagings\_per\_load\_unit))

attach(x\_dim\_loadUnit)

x\_dim\_loadUnit["packaging\_x\_dim"] = log(packaging\_x\_dim)

x\_dim\_loadUnit["packaging\_y\_dim"] = log(packaging\_y\_dim)

x\_dim\_loadUnit["packaging\_z\_dim"] = log(packaging\_z\_dim)

x\_dim\_loadUnit["packaging\_volume"] = log(packaging\_volume)

x\_dim\_loadUnit["packaging\_load\_capacity"] = log(packaging\_load\_capacity)

x\_dim\_loadUnit["packaging\_weight"] = log(packaging\_weight)

x\_dim\_loadUnit["products\_per\_packaging"] = log(products\_per\_packaging)

x\_dim\_loadUnit["packagings\_per\_load\_unit"] = log(packagings\_per\_load\_unit)

attach(x\_dim\_loadUnit)

x\_dim\_loadUnit = subset(x\_dim\_loadUnit, select = -c(packaging\_raw\_material\_name, packaging\_is\_oneway, packaging\_is\_special))

sum(is.na(loadUnit))

set.seed(100)

split = sample(1:2, nrow(x\_dim\_loadUnit), replace = TRUE, prob = c(0.7, 0.3))

train = x\_dim\_loadUnit[split ==1, ]

val = x\_dim\_loadUnit[split == 2,]

LinearModel4 = lm(packagings\_per\_load\_unit~., data = train)

summary(LinearModel4)

pred\_linearModel4 = predict(LinearModel4, val)

perform\_linearModel4 = acc\_error(val$packagings\_per\_load\_unit, pred\_linearModel4)

perform\_linearModel4

library(rpart)

library(rpart.plot)

# TreeModel1 with all predictors

# Remove other numerical dependent variables and product variables

attach(loadUnit)

x\_dim\_loadUnit = subset(loadUnit, select = -c(load\_unit\_x\_dim, load\_unit\_y\_dim, load\_unit\_z\_dim, load\_unit\_weight,

load\_unit\_volume, product\_kogr\_number, product\_quality\_index, product\_y\_dim,

product\_generic\_family\_name, product\_module\_number, product\_supplier\_number,

product\_zdim, product\_is\_dangerous\_good, product\_name, product\_weight,

product\_is\_esp, product\_number, product\_x\_dim, packagings\_per\_load\_unit))

attach(x\_dim\_loadUnit)

x\_dim\_loadUnit["packaging\_x\_dim"] = log(packaging\_x\_dim)

x\_dim\_loadUnit["packaging\_y\_dim"] = log(packaging\_y\_dim)

x\_dim\_loadUnit["packaging\_z\_dim"] = log(packaging\_z\_dim)

x\_dim\_loadUnit["packaging\_volume"] = log(packaging\_volume)

x\_dim\_loadUnit["packaging\_load\_capacity"] = log(packaging\_load\_capacity)

x\_dim\_loadUnit["packaging\_weight"] = log(packaging\_weight)

x\_dim\_loadUnit["products\_per\_packaging"] = log(products\_per\_packaging)

x\_dim\_loadUnit["packagings\_per\_load\_unit"] = log(packagings\_per\_load\_unit)

attach(x\_dim\_loadUnit)

colnames(x\_dim\_loadUnit)

sum(is.na(loadUnit))

set.seed(100)

split = sample(1:2, nrow(x\_dim\_loadUnit), replace = TRUE, prob = c(0.7, 0.3))

train = x\_dim\_loadUnit[split ==1, ]

val = x\_dim\_loadUnit[split == 2,]

treeModel1 = rpart(packagings\_per\_load\_unit~. , data = train, control = rpart.control(cp = 0.0001))

bestcp=treeModel1$cptable[which.min(treeModel1$cptable[,"xerror"]),"CP"]

treeModel1\_pruned=prune(treeModel1, cp = bestcp)

pred\_treeModel1\_pruned = predict(treeModel1\_pruned, val)

perform\_treeModel1\_pruned = acc\_error(val$packagings\_per\_load\_unit, pred\_treeModel1\_pruned)

perform\_treeModel1\_pruned

printcp(treeModel1\_pruned)

treeModel2\_pruned=prune(treeModel1\_pruned, cp = 0.00022184)

pred\_treeModel2\_pruned = predict(treeModel2\_pruned, val)

perform\_treeModel2\_pruned = acc\_error(val$packagings\_per\_load\_unit, pred\_treeModel2\_pruned)

perform\_treeModel2\_pruned

printcp(treeModel2\_pruned)

library(randomForest)

randomTree1 = randomForest(packagings\_per\_load\_unit~., data = train, mtry = 3, ntree=155, importance = TRUE, na.action = na.omit)

pred\_randomTree1 = predict(randomTree1, val)

perform\_randomTree1 = acc\_error(val$packagings\_per\_load\_unit, pred\_randomTree1)

perform\_randomTree1

print(randomTree1)

#Pick best model

perf\_comp\_linear = rbind(perform\_linearModel1, perform\_linearModel2, perform\_linearModel4)

numb\_predictors = c(26,8,6)

cbind(perf\_comp\_linear, numb\_predictors)

perf\_comp\_tree = rbind(perform\_treeModel1\_pruned,perform\_treeModel2\_pruned)

numb\_splits = c(52,40)

cp = c(bestcp,0.00022184)

cbind(perf\_comp\_tree, numb\_splits,cp)

perf\_comp\_randomTree = rbind(perform\_randomTree1)

numb\_trees = c(155)

cbind(perf\_comp\_randomTree, numb\_trees)

mean(packagings\_per\_load\_unit)

min(packagings\_per\_load\_unit)

max(packagings\_per\_load\_unit)

par(mfrow=c(1,1))

rpart.plot(treeModel2\_pruned,type=4,extra="auto",

main="Packagings per Load Unit (in numerical)")

par(mfrow=c(3,4))

library(visreg)

visreg(treeModel2\_pruned)

#Load Unit is One Way

# Logistic regression models

model11=glm(load\_unit\_is\_oneway~packaging\_is\_oneway+packaging\_is\_special+products\_per\_packaging+packaging\_load\_capacity+packaging\_weight+packaging\_x\_dim+packaging\_y\_dim+packaging\_z\_dim,data=Train, family="binomial")

model12=glm(load\_unit\_is\_oneway~packaging\_load\_capacity+packaging\_weight+products\_per\_packaging+packaging\_x\_dim+packaging\_y\_dim+packaging\_z\_dim,data=Train, family="binomial")

model13=glm(load\_unit\_is\_oneway~packaging\_load\_capacity+packaging\_x\_dim+packaging\_y\_dim+packaging\_z\_dim,data = Train, family="binomial")

#Explanatory Modeling

summary(model11)

summary(model12)

summary(model13)

# Predicting/ Classifying new cases

pred11=predict(model11, newdata=valid, type="response")

pred12=predict(model12, newdata=valid, type="response")

pred13=predict(model13, newdata=valid, type="response")

act\_pred11=data.frame(ID=1:nrow(valid),valid$load\_unit\_is\_oneway,pred11)

act\_pred12=data.frame(ID=1:nrow(valid),valid$load\_unit\_is\_oneway,pred12)

act\_pred13=data.frame(ID=1:nrow(valid),valid$load\_unit\_is\_oneway,pred13)

#11

conf\_mat11=cmx(act\_pred11)

total\_acc11= pcc(conf\_mat11, st.dev = FALSE)

sens11=sensitivity(conf\_mat11, st.dev = FALSE)

spec11=specificity(conf\_mat11, st.dev = FALSE)

accuracy\_measures11=c(total\_acc11,sens11,spec11)

names(accuracy\_measures11)=c("Overall accuracy", "Sensitivity", "Specificity")

#12

conf\_mat12=cmx(act\_pred12)

total\_acc12= pcc(conf\_mat12, st.dev = FALSE)

sens12=sensitivity(conf\_mat12, st.dev = FALSE)

spec12=specificity(conf\_mat12, st.dev = FALSE)

accuracy\_measures12=c(total\_acc12,sens12,spec12)

names(accuracy\_measures12)=c("Overall accuracy", "Sensitivity", "Specificity")

#13

conf\_mat13=cmx(act\_pred13)

total\_acc13= pcc(conf\_mat13, st.dev = FALSE)

sens13=sensitivity(conf\_mat13, st.dev = FALSE)

spec13=specificity(conf\_mat13, st.dev = FALSE)

accuracy\_measures13=c(total\_acc13,sens13,spec13)

names(accuracy\_measures13)=c("Overall accuracy", "Sensitivity", "Specificity")

#Error Summary

perf\_comp1 = rbind(accuracy\_measures11, accuracy\_measures12, accuracy\_measures13)

numb\_predictors1=c(8,6,4)

cbind(perf\_comp1, numb\_predictors1)

par(mfrow=c(1,1))

#Multiple ROC Curves

act\_pred\_mult=data.frame(ID=1:nrow(valid),valid$load\_unit\_is\_oneway,pred11,pred12,pred13)

auc.roc.plot(act\_pred\_mult,col=c(2,7,4),line.type=c(3,2,1),threshold = 1001, main="Load Unit is One Way (ROC Curves)",legend.text=c("Model 1", "Model 2","Model 3"))

#Load Unit is Special

# Logistic regression models

model21=glm(load\_unit\_is\_special~packaging\_is\_oneway+packaging\_is\_special+products\_per\_packaging+packaging\_load\_capacity+packaging\_weight+packaging\_x\_dim+packaging\_y\_dim+packaging\_z\_dim,data=Train, family="binomial")

model22=glm(load\_unit\_is\_special~packaging\_weight+packaging\_x\_dim+packaging\_y\_dim+packaging\_z\_dim,data=Train, family="binomial")

model23=glm(load\_unit\_is\_special~packaging\_weight+packaging\_x\_dim,data=Train, family="binomial")

#Explanatory Modeling

summary(model21)

summary(model22)

summary(model23)

# Predicting/ Classifying new cases

pred21=predict(model21, newdata=valid, type="response")

pred22=predict(model22, newdata=valid, type="response")

pred23=predict(model23, newdata=valid, type="response")

act\_pred21=data.frame(ID=1:nrow(valid),valid$load\_unit\_is\_special,pred21)

act\_pred22=data.frame(ID=1:nrow(valid),valid$load\_unit\_is\_special,pred22)

act\_pred23=data.frame(ID=1:nrow(valid),valid$load\_unit\_is\_special,pred23)

#21

conf\_mat21=cmx(act\_pred21)

total\_acc21= pcc(conf\_mat21, st.dev = FALSE)

sens21=sensitivity(conf\_mat21, st.dev = FALSE)

spec21=specificity(conf\_mat21, st.dev = FALSE)

accuracy\_measures21=c(total\_acc21,sens21,spec21)

names(accuracy\_measures21)=c("Overall accuracy", "Sensitivity", "Specificity")

#22

conf\_mat22=cmx(act\_pred22)

total\_acc22= pcc(conf\_mat22, st.dev = FALSE)

sens22=sensitivity(conf\_mat22, st.dev = FALSE)

spec22=specificity(conf\_mat22, st.dev = FALSE)

accuracy\_measures22=c(total\_acc22,sens22,spec22)

names(accuracy\_measures22)=c("Overall accuracy", "Sensitivity", "Specificity")

#23

conf\_mat23=cmx(act\_pred23)

total\_acc23= pcc(conf\_mat23, st.dev = FALSE)

sens23=sensitivity(conf\_mat23, st.dev = FALSE)

spec23=specificity(conf\_mat23, st.dev = FALSE)

accuracy\_measures23=c(total\_acc23,sens23,spec23)

names(accuracy\_measures23)=c("Overall accuracy", "Sensitivity", "Specificity")

#Error Summary

perf\_comp2 = rbind(accuracy\_measures21, accuracy\_measures22, accuracy\_measures23)

numb\_predictors2=c(8,4,2)

cbind(perf\_comp2, numb\_predictors2)

#Multiple ROC Curves

act\_pred\_mult=data.frame(ID=1:nrow(valid),valid$load\_unit\_is\_special,pred21,pred22,pred23)

auc.roc.plot(act\_pred\_mult,col=c(2,7,4),line.type=c(1,2,3),threshold = 1001, main="Load Unit is Special (ROC Curves)",legend.text=c("Model 1", "Model 2","Model 3"))

#Load unit is One Way

par(mfrow=c(1,1))

set.seed(100)

#Decision Trees

tree1 <- rpart(load\_unit\_is\_oneway ~ packaging\_is\_oneway+packaging\_is\_special+products\_per\_packaging+packaging\_load\_capacity+packaging\_weight+packaging\_x\_dim+packaging\_y\_dim+packaging\_z\_dim, data = Train, control = rpart.control(cp = 0.0001), method="class")

rpart.plot(tree1,main="Load Unit is One Way (binary response)")

#Classification tree

#Error Summary

tree1$cptable

bestcp1=tree1$cptable[which.min(tree1$cptable[,"xerror"]),"CP"]

tree1.pruned=prune(tree1, cp = bestcp1)

rpart.plot(tree1.pruned,main="Load Unit is One Way (binary response)")

pred1=predict(tree1.pruned, newdata=valid,type="prob")

pred1=pred1[,2]

act\_pred1=data.frame(ID=1:nrow(valid),valid$load\_unit\_is\_oneway,pred1)

conf\_mat1=cmx(act\_pred1)

total\_acc1= pcc(conf\_mat1,st.dev = FALSE)

sens1=sensitivity(conf\_mat1,st.dev = FALSE)

spec1=specificity(conf\_mat1,st.dev = FALSE)

accuracy\_measures1=c(total\_acc1,sens1,spec1)

names(accuracy\_measures1)=c("Overall accuracy", "Sensitivity", "Specificity")

accuracy\_measures1

auc(act\_pred1)

#Load unit is Special

set.seed(100)

#Decision Trees

tree2 <- rpart(load\_unit\_is\_special ~ packaging\_is\_oneway+packaging\_is\_special+products\_per\_packaging+packaging\_load\_capacity+packaging\_weight+packaging\_x\_dim+packaging\_y\_dim+packaging\_z\_dim, data = Train, control = rpart.control(cp = 0.0001), method="class")

rpart.plot(tree2,main="Load Unit is Special (binary response)")

#Classification tree

#Error Summary

tree2$cptable

bestcp2=tree2$cptable[which.min(tree2$cptable[,"xerror"]),"CP"]

tree2.pruned=prune(tree2, cp = bestcp2)

rpart.plot(tree2.pruned,main="Load Unit is Special (binary response)")

pred2=predict(tree2.pruned, newdata=valid,type="prob")

pred2=pred2[,2]

act\_pred2=data.frame(ID=1:nrow(valid),valid$load\_unit\_is\_special,pred2)

conf\_mat2=cmx(act\_pred2)

total\_acc2= pcc(conf\_mat2,st.dev = FALSE)

sens2=sensitivity(conf\_mat2,st.dev = FALSE)

spec2=specificity(conf\_mat2,st.dev = FALSE)

accuracy\_measures2=c(total\_acc2,sens2,spec2)

names(accuracy\_measures2)=c("Overall accuracy", "Sensitivity", "Specificity")

accuracy\_measures2

auc(act\_pred2)