Objective : The data scientists at BigMart have collected 2013 sales data for 1559 products across 10 stores in different cities. The main objective is to understand whether specific properties of products and/or stores play a significant role in terms of increasing or decreasing sales volume. To achieve this goal, we will build a predictive model and find out the sales of each product at a particular store. This will help BigMart to boost their sales by learning optimised product organization inside stores.

Dataset: The dataset currently being used in the report is the of sales of Bigmart, and is as an example to explain the concept of the project. It will vary as per the needs of the user.

# Exploring The BigMart Dataset

* Call Libraries and Read Data from File.
* Training Dataset lets you extract features and train to fit a model.
* Testing Dataset lets you predict using the model obtained on the training set.

# for plots

library(ggplot2)

library(gridExtra)

# for correlation plots

library(corrplot)

# for manipulation of data

library(plyr)

library(dplyr)

# to use dummy values for factors

library(caret)

# for computation of missing values

library(mice)

Load the train and test datasets

#Load Datasets

train <- read.csv("Train\_Dataset")

test <- read.csv("Test\_Dataset")

Dimension of train and test dataset

> dim(train)

[1] 8523 12

> dim(test)

[1] 5681 11

Structure of train dataset

> str(train)

'data.frame': 8523 obs. of 12 variables:

$ Item\_Identifier : Factor w/ 1559 levels "DRA12","DRA24",..: 157 9 663 1122 1298 759 697 739 441 991 ...

$ Item\_Weight : num 9.3 5.92 17.5 19.2 8.93 ...

$ Item\_Fat\_Content : Factor w/ 5 levels "LF","low fat",..: 3 5 3 5 3 5 5 3 5 5 ...

$ Item\_Visibility : num 0.016 0.0193 0.0168 0 0 ...

$ Item\_Type : Factor w/ 16 levels "Baking Goods",..: 5 15 11 7 10 1 14 14 6 6 ...

$ Item\_MRP : num 249.8 48.3 141.6 182.1 53.9 ...

$ Outlet\_Identifier : Factor w/ 10 levels "OUT010","OUT013",..: 10 4 10 1 2 4 2 6 8 3 ...

$ Outlet\_Establishment\_Year: int 1999 2009 1999 1998 1987 2009 1987 1985 2002 2007 ...

$ Outlet\_Size : Factor w/ 4 levels "","High","Medium",..: 3 3 3 1 2 3 2 3 1 1 ...

$ Outlet\_Location\_Type : Factor w/ 3 levels "Tier 1","Tier 2",..: 1 3 1 3 3 3 3 3 2 2 ...

$ Outlet\_Type : Factor w/ 4 levels "Grocery Store",..: 2 3 2 1 2 3 2 4 2 2 ...

$ Item\_Outlet\_Sales : num 3735 443 2097 732 995 ...

Structure of test dataset

> str(test)

'data.frame': 5681 obs. of 11 variables:

$ Item\_Identifier : Factor w/ 1543 levels "DRA12","DRA24",..: 1104 1068 1407 810 1185 462 605 267 669 171 ...

$ Item\_Weight : num 20.75 8.3 14.6 7.32 NA ...

$ Item\_Fat\_Content : Factor w/ 5 levels "LF","low fat",..: 3 4 3 3 5 5 5 3 5 3 ...

$ Item\_Visibility : num 0.00756 0.03843 0.09957 0.01539 0.1186 ...

$ Item\_Type : Factor w/ 16 levels "Baking Goods",..: 14 5 12 14 5 7 1 1 14 1 ...

$ Item\_MRP : num 107.9 87.3 241.8 155 234.2 ...

$ Outlet\_Identifier : Factor w/ 10 levels "OUT010","OUT013",..: 10 3 1 3 6 9 4 6 8 3 ...

$ Outlet\_Establishment\_Year: int 1999 2007 1998 2007 1985 1997 2009 1985 2002 2007 ...

$ Outlet\_Size : Factor w/ 4 levels "","High","Medium",..: 3 1 1 1 3 4 3 3 1 1 ...

$ Outlet\_Location\_Type : Factor w/ 3 levels "Tier 1","Tier 2",..: 1 2 3 2 3 1 3 3 2 2 ...

$ Outlet\_Type : Factor w/ 4 levels "Grocery Store",..: 2 2 1 2 4 2 3 4 2 2 ...

# Data Manipulation

* We discover some columns need to be corrected for a good analysis. So, we should manipulate some part of data. Let’s first combine the data sets. This will save our time as we don’t need to write separate codes for train and test data sets. To combine the two data frames, we must make sure that they have equal columns, which is not the case. Test data set has one less column (response variable). Let’s first add the column. We can give this column any value.

An intuitive approach would be to extract the mean value of sales from train data set and use it as placeholder for test variable Item\_Outlet\_Sales. Anyways, let’s make it simple for now. I’ve taken a value 0. Now, we’ll combine the data sets.

test$Item\_Outlet\_Sales <- 0

combi <- rbind(train, test)

* Make changes in order to create a proper datasets having values which are distinct.

Also, some entries for Outlet\_Size are empty, let's temporarily call them "Other"

levels(combi$Outlet\_Size)[1]<- "Other"

combi$Outlet\_Size

Check the columns to find which variables are missing

table(is.na(combi))

colSums(is.na(combi))

* Plot the graphs to check/visualize the data in order to see empty values

ggplot(combi, aes(Item\_Type, Item\_Weight)) + geom\_boxplot() +

theme(axis.text.x = element\_text(angle = 70, vjust = 0.5, color = "black")) +

xlab("Item Type") +

ylab("Item Weight") +

ggtitle("Item Weight vs Item Type")

ggplot(combi, aes(Outlet\_Identifier, Item\_Weight)) +

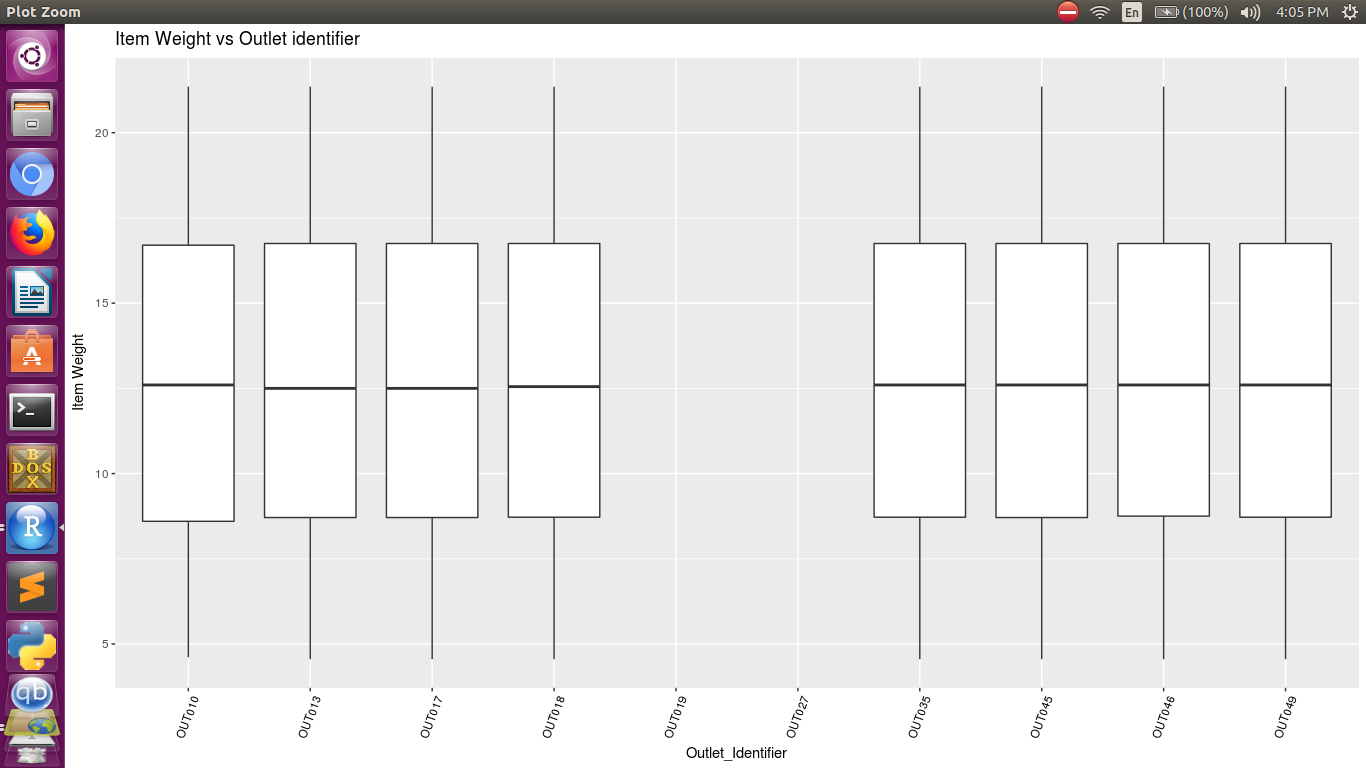
geom\_boxplot() +

theme(axis.text.x = element\_text(angle = 70, vjust = 0.5, color = "black")) +

xlab("Outlet\_Identifier") +

ylab("Item Weight") +

ggtitle("Item Weight vs Outlet identifier")



We find that some data is still missing. Each item\_identifier will identify each unique item. As each unique item will have unique weight, we compute weights by using standard deviation of item identifier

weightsByItem <- as.data.frame(ddply(na.omit(combi), ~Item\_Identifier,

summarise, mean = mean(Item\_Weight), sd = sd(Item\_Weight)))

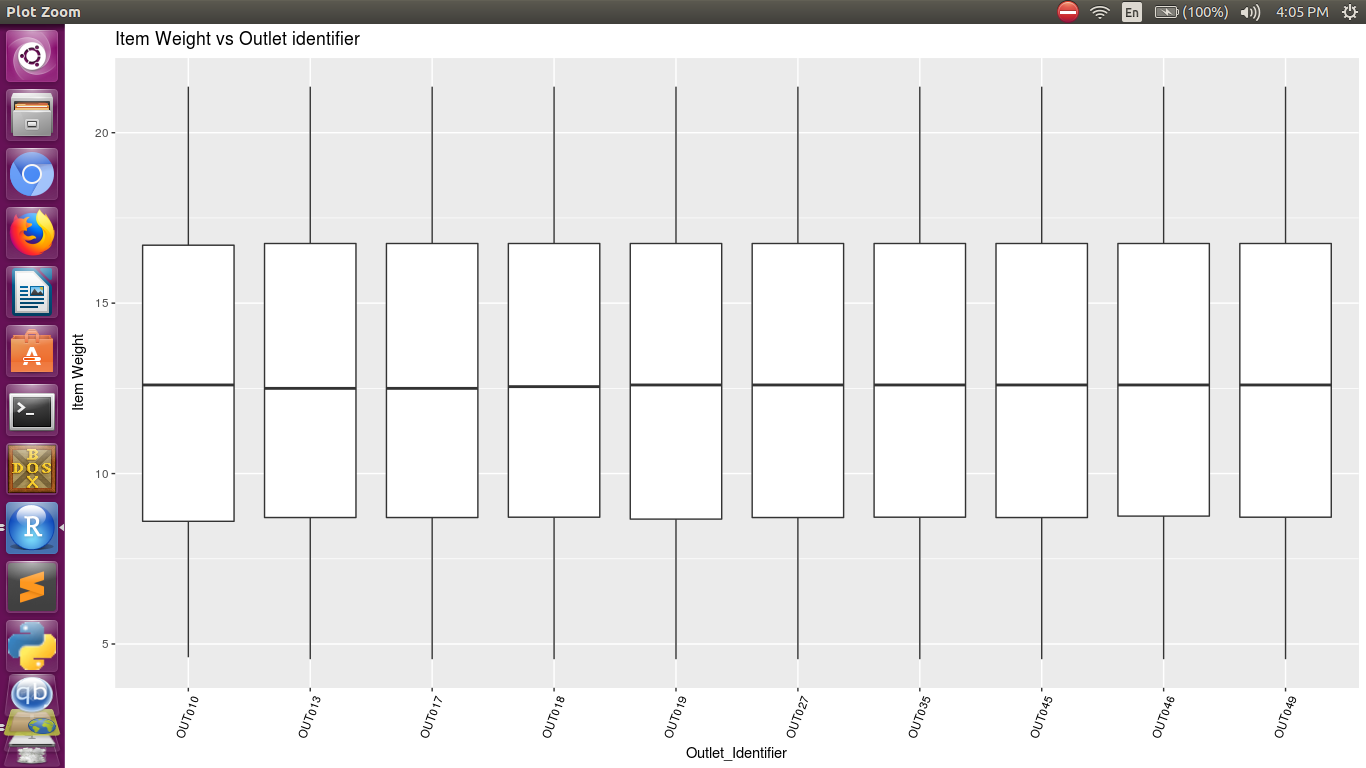
* Use the above computed values to fill the missing weight values

combi$Item\_Weight <- ifelse(is.na(combi$Item\_Weight),

weightsByItem$mean[match(combi$Item\_Identifier, weightsByItem$Item\_Identifier)],

combi$Item\_Weight)

* Replot the graph to confirm whether there are no missing values.



* We compute the number of years since establishment as it is an important parameter for accuracy and modelling.
* Categorize the items by introducing levels for each range

combi$MRP\_Level <- as.factor(

ifelse(combi$Item\_MRP < 69, "Low",

ifelse(combi$Item\_MRP < 136, "Medium",

ifelse(combi$Item\_MRP < 203, "High", "Very\_High"))))

* Plot graph to ensure the expected output.

We find that density is zero for three Item MRP's.

ggplot(combi, aes(x=Item\_MRP)) +

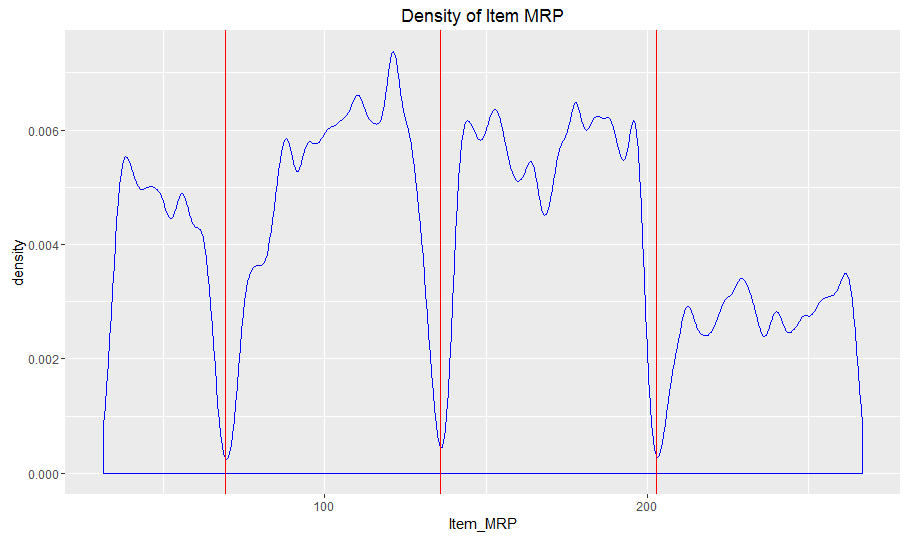
geom\_density(color = "blue", adjust=1/20) +

geom\_vline(xintercept = 69, color="red")+

geom\_vline(xintercept = 136, color="red")+

geom\_vline(xintercept = 203, color="red") +

ggtitle("Density of Item MRP")



Now check the entries in different columns such as Item Visiblity which has several entries as 0.

* Change the entries using MICE where entries are imputed by values.

newCombi <- mice(combi,m=1,maxit=1,meth='pmm',seed=0)

# Data Visualization

* We use the ggplot function to plot graphs of various variables.

Boxplot of Sales vs. Outlet type

ggplot(combi[1:nrow(train),], aes(x = Outlet\_Type, y = Item\_Outlet\_Sales, fill = Outlet\_Size))+

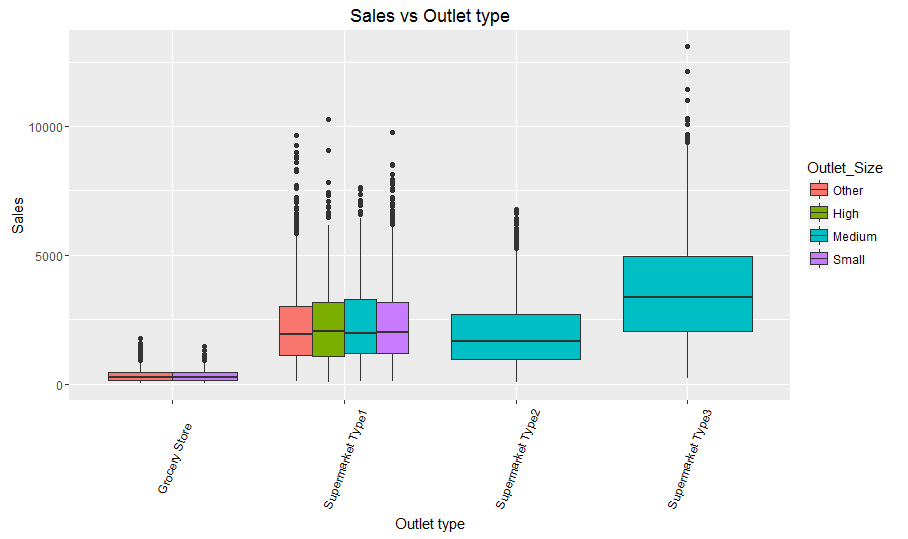
geom\_boxplot() +

theme(axis.text.x = element\_text(angle = 70, vjust = 0.5, color = "black")) +

xlab("Outlet type") +

ylab("Sales") +

ggtitle("Sales vs Outlet type")



Boxplot of Sales vs. Item type

ggplot(combi[1:nrow(train),], aes(x = Item\_Type, y = Item\_Outlet\_Sales, fill = Outlet\_Size)) +

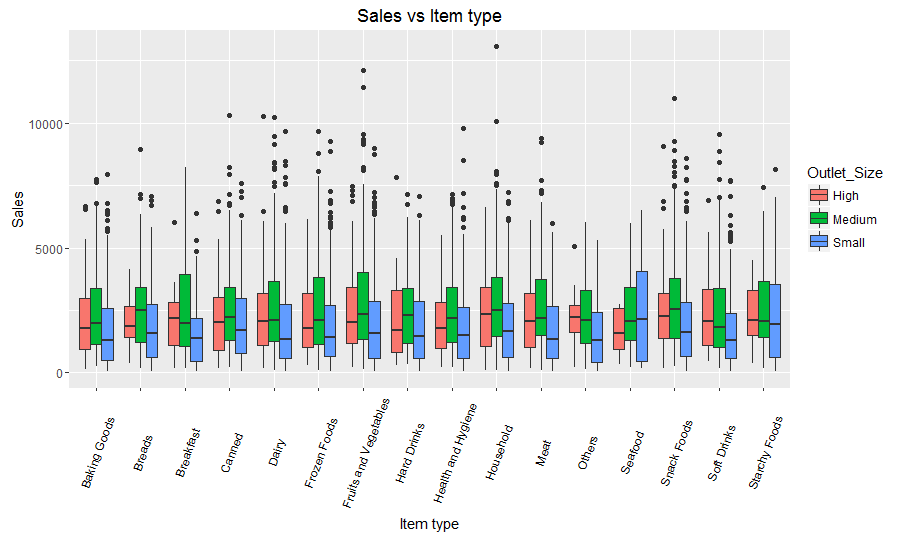
geom\_boxplot() +

theme(axis.text.x = element\_text(angle = 70, vjust = 0.5, color = "black")) +

xlab("Item type") +

ylab("Sales") +

ggtitle("Sales vs Item type")



Boxplot of Sales vs. Item type using fill as Outlet\_Type

ggplot(combi[1:nrow(train),], aes(x = Item\_Type, y = Item\_Outlet\_Sales, fill = Outlet\_Type)) +

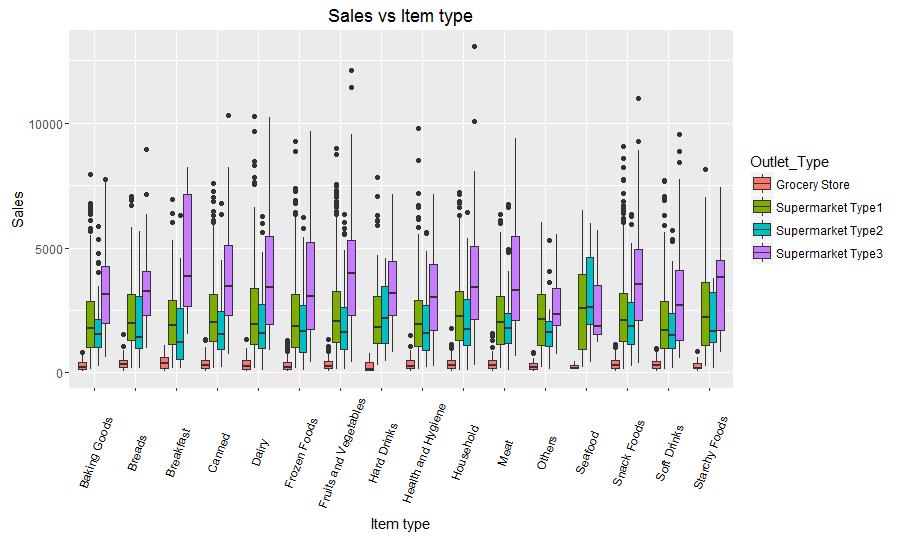
geom\_boxplot() +

theme(axis.text.x = element\_text(angle = 70, vjust = 0.5, color = "black")) +

xlab("Item type") +

ylab("Sales") +

ggtitle("Sales vs Item type")



Boxplot of Visibility vs Item type

ggplot(combi, aes(Item\_Type, Item\_Visibility, fill = Outlet\_Size)) +

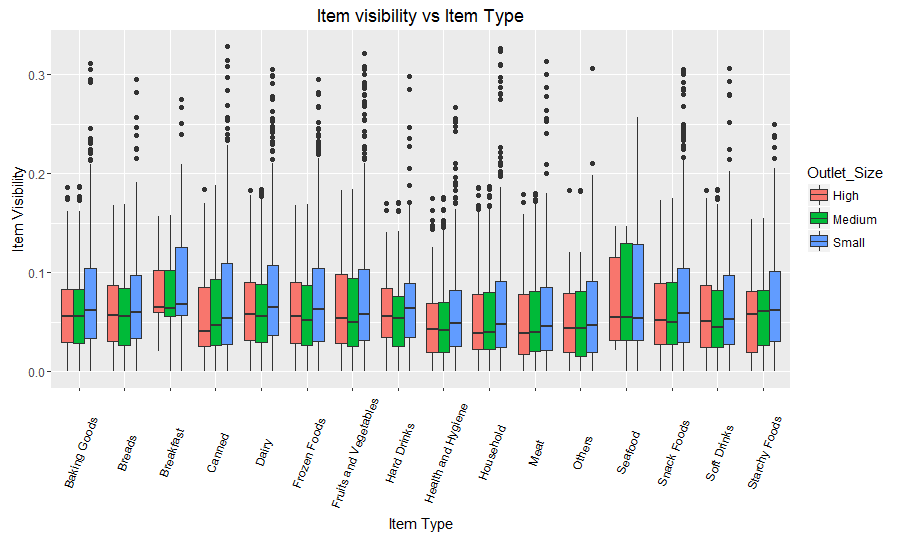
geom\_boxplot() +

theme(axis.text.x = element\_text(angle = 70, vjust = 0.5, color = "black")) +

xlab("Item Type") +

ylab("Item Visibility") +

ggtitle("Item visibility vs Item Type")



Boxplot of Visibility vs. Outlet Identifier

ggplot(combi, aes(Outlet\_Identifier, Item\_Visibility)) +

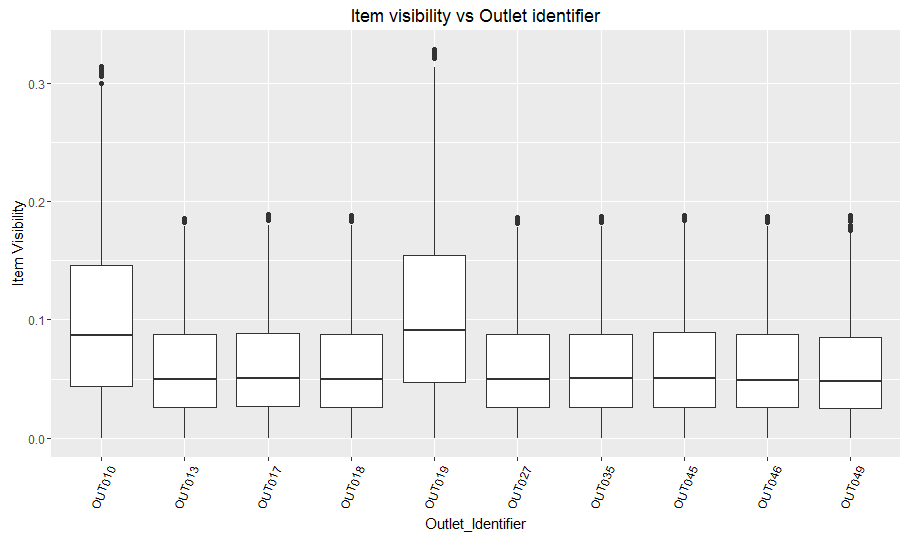
geom\_boxplot() +

theme(axis.text.x = element\_text(angle = 70, vjust = 0.5, color = "black")) +

xlab("Outlet\_Identifier") +

ylab("Item Visibility") +

ggtitle("Item visibility vs Outlet identifier")



# Modeling

* First we try to understand different correlations between parameters of the dataset. We check the correlation between numerical variables.

corMatrix <- cor(combi[1:nrow(train),][sapply(combi[1:nrow(train),], is.numeric)])

corMatrix

corrplot::corrplot(corMatrix, method="number", type="upper")

corrplot::corrplot(corMatrix, method="number", type="upper", order="hclust")

* From the above correlation plot, we understand that certain parameters have different correlation with each other. For instance, Item\_Outlet\_Sales has strong positive correlation with Item\_MRP and weaker negative one with Item\_Visibility.
* Create another sub dataframe inorder to contain columns only of parameters used for correlation.

Here, these parameters are Item\_Outlet\_Sales, Item\_Visibility, and Item\_MRP.

subData <- as.data.frame(cbind(

combi[1:nrow(train),]$Item\_Visibility,

combi[1:nrow(train),]$Item\_MRP,

combi[1:nrow(train),]$Item\_Outlet\_Sales))

names(subData) <- c("Item\_Visibility",

"Item\_MRP",

"Item\_Outlet\_Sales")

sub.groupby <- combi[1:nrow(train),]$Outlet\_Type

str(subData)

subData.pca <- prcomp(subData,

center = TRUE,

scale. = TRUE)

summary(subData.pca)

g <- ggbiplot(subData.pca,

obs.scale = 1,

var.scale = 1,

groups = sub.groupby,

ellipse = TRUE,

circle = TRUE

)

g <- g + scale\_color\_discrete(name = '')

g <- g + theme(legend.direction = 'horizontal',

legend.position = 'top')

print(g)

theta <- seq(0,2\*pi,length.out = 100)

circle <- data.frame(x = cos(theta), y = sin(theta))

p <- ggplot(circle,aes(x,y)) + geom\_path()

loadings <- data.frame(subData.pca$rotation,

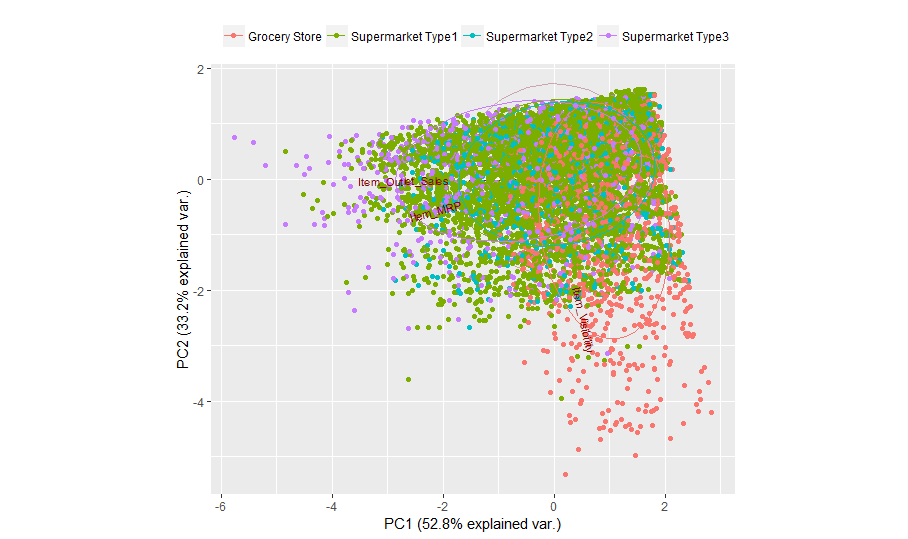
.names = row.names(subData.pca$rotation))

p + geom\_text(data=loadings,

mapping=aes(x = PC1, y = PC2, label = .names, colour = .names)) +

coord\_fixed(ratio=1) +

labs(x = "PC1", y = "PC2")



* Plot graphs inorder to understand distributions of values using the given parameters.

For the given instance, we check the parameters Item Outlet Sales and Item Visibility.

Scatter plot of Item\_Outlet\_Sales vs Item\_Visibility coloured according to the Outlet type

ggplot(combi[1:nrow(train),], aes(Item\_Visibility, Item\_Outlet\_Sales)) +

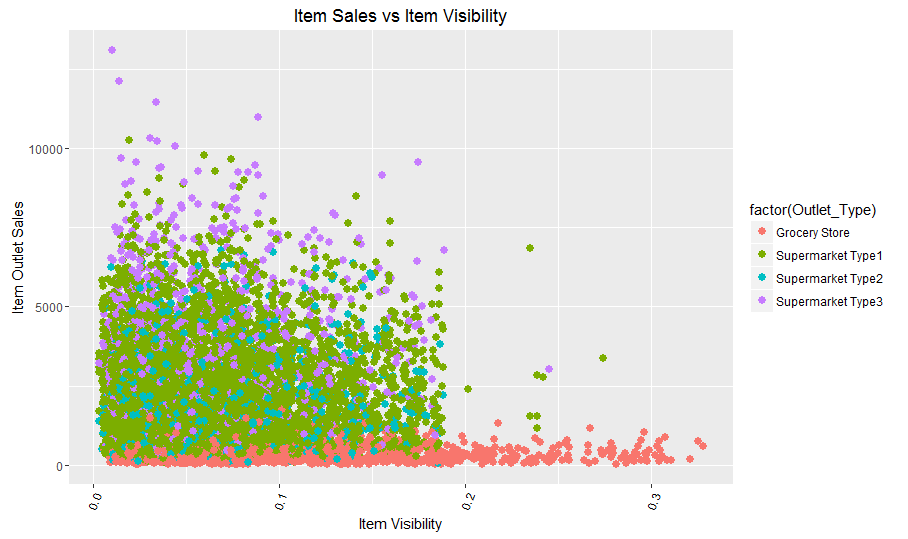
geom\_point(size = 2.5, aes(colour = factor(Outlet\_Type))) +

theme(axis.text.x = element\_text(angle = 70, vjust = 0.5, color = "black")) +

xlab("Item Visibility") +

ylab("Item Outlet Sales") +

ggtitle("Item Sales vs Item Visibility"))



* Make changes in graph for better undersanding.

Dividing sales by MRP does reduce the number of outliers and also emphasizes the differences between the different types of shop, so we'll just do that in the following

combi$Item\_Outlet\_Sales <- combi$Item\_Outlet\_Sales/combi$Item\_MRP

Proportion of Supermarkets vs. Grocery stores in the data

prop.table(table(combi$Outlet\_Type))

* Now, we check the various parameters for each entity.For this instance ,it will be shops.

analyze.shop <- function(shopID = character ) {

cat("RESULTS FOR SHOP ", shopID, "\n", "\n",

file = "ContingencyTables.txt", append = TRUE)

shopData <- as.data.frame(combi[1:nrow(train),][combi[1:nrow(train),]$Outlet\_Identifier %in% shopID,])

shopData$Outlet\_Identifier <- factor(shopData$Outlet\_Identifier)

shopData$Outlet\_Size <- factor(shopData$Outlet\_Size)

shopData$Outlet\_Location\_Type <- factor(shopData$Outlet\_Location\_Type)

shopData$Outlet\_Type <- factor(shopData$Outlet\_Type)

As Size, location type and type have only one level, we can drop them here but what about the establishment year?

cat("Variance of outlet establishment year: ", var(shopData$Year), "\n",

file = "ContingencyTables.txt", append = TRUE)

# since the variance of Outlet\_Establishment\_Year is zero, we

# can also remove that column

shopData <- select(shopData, -c(Outlet\_Identifier,Outlet\_Size,

Outlet\_Location\_Type,

Outlet\_Type,

Year))

Now add the values computed for each shop into the contingency table.

cat("Contingency tables for shop ", shopID, "\n", file = "ContingencyTables.txt", append = TRUE)

capture.output(

table(shopData$Item\_Identifier, shopData$Item\_Fat\_Content),

file = "ContingencyTables.txt", append = TRUE)

capture.output(

table(shopData$Item\_Type, shopData$Item\_Identifier),

file = "ContingencyTables.txt", append = TRUE)

capture.output(

table(shopData$Item\_Type, shopData$Item\_Fat\_Content),

file = "ContingencyTables.txt", append = TRUE)

cat("\n", "\n", file = "ContingencyTables.txt", append = TRUE)

* Analyse the data (here for each shop).

cat("Brief analyses by shop", "\n", file = "ContingencyTables.txt")

for (i in levels(combi$Outlet\_Identifier)) {

analyze.shop(i)

}

There is enough data (data from each shop ) and the parameters (Item\_MRP, Item\_Visibility ) present for prediction.

Restructure the train and test datasets

new\_train <- combi[1:nrow(train),]

Save them, so that we don't have to redo the cleaning over and over again

write.csv(new\_train, file="new\_train.csv", row.names=FALSE, quote = FALSE)

write.csv(new\_test, file="new\_test.csv", row.names=FALSE, quote = FALSE)

* **Feature Engineering:** check variable importance with random feature elimination (RFE) from caret

Scale Sales to be in interval [0,1]

maxSales <- max(new\_train$Item\_Outlet\_Sales)

new\_train$Item\_Outlet\_Sales <- new\_train$Item\_Outlet\_Sales/maxSales

set.seed(0)

One-hot encoding of the factor variables and leave out the intercept column

new\_train <- as.data.frame(model.matrix( ~ . + 0, data = new\_train))

new\_test <- as.data.frame(model.matrix( ~ . + 0, data = new\_test))

str(new\_train)

Define a vector of Item\_Outlet\_Sales and a dataframe of predictors

sales <- new\_train$Item\_Outlet\_Sales

predictors <- subset(new\_train, select=-c(Item\_Outlet\_Sales))

Check relative importance of predictors with caret rfe and do it in parallel

cl <- makeCluster(detectCores()); registerDoParallel(cl)

subsetSizes <- c(1:20, 25, 30, 40, 50, 60, 70, 80, 90, 100, 121)

Number of resamples

N <- 5

seeds <- vector(mode = "list", length = N+1)

for(i in 1:N) seeds[[i]] <- sample.int(1000, length(subsetSizes) + 1)

seeds[[N+1]] <- sample.int(1000, 1)

control <- rfeControl(functions=rfFuncs,

method="cv",

seeds = seeds,

number = N,

repeats = 3,

verbose = TRUE,

allowParallel = TRUE)

Start the clock

ptm <- proc.time()

Run the RFE algorithm

results2 <- rfe(x = predictors,

y = sales,

sizes = subsetSizes,

preProc=c("center", "scale"),

rfeControl=control)

Stop the clock

proc.time() - ptm

Stop the parallel processing and register sequential front-end

stopCluster(cl);

registerDoSEQ();

Summarize the results

print(results2)

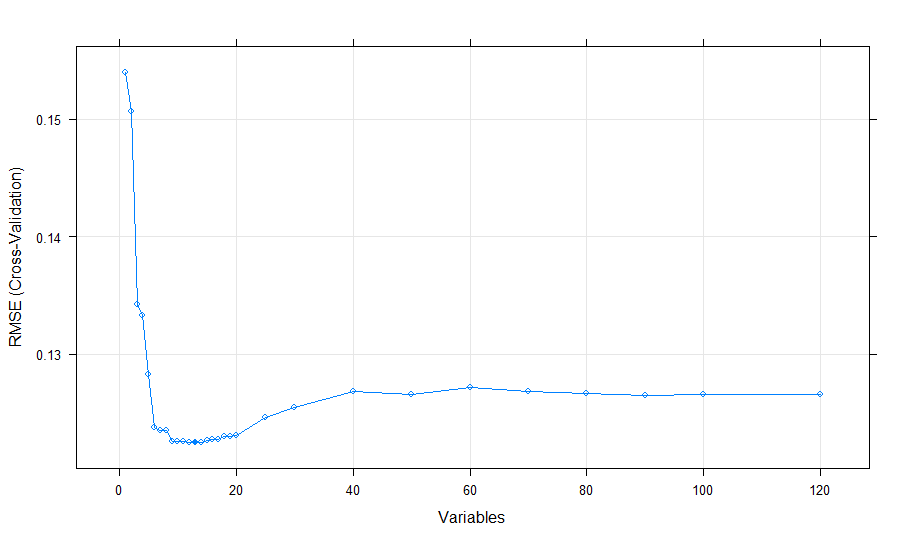
List all features in descending order of importance

listOfPreds <- pickVars(results2$variables, 120)

listOfPreds

Plot the results

plot(results2, type=c("g", "o"))



* Build a data frame containing the predictors ordered by their importance

ordered.preds <- predictors[,listOfPreds[1]]

for (i in 2:length(listOfPreds)) {

ordered.preds <- cbind(ordered.preds, predictors[,listOfPreds[i]])

}

colnames(ordered.preds) <- listOfPreds

ordered.preds <- as.data.frame(ordered.preds)

ordered.test <- new\_test[,listOfPreds[1]]

for (i in 2:length(listOfPreds)) {

ordered.test <- cbind(ordered.test, new\_test[,listOfPreds[i]])

}

colnames(ordered.test) <- listOfPreds

ordered.test <- as.data.frame(ordered.test)

Remove the scaling to [0,1] in sales

sales <- sales\*maxSales

Save those dataframes to disk

write.csv(ordered.preds, file="ordered\_predictors.csv", row.names=FALSE, quote = FALSE)

write.csv(ordered.test, file="ordered\_test.csv", row.names=FALSE, quote = FALSE)

write.csv(sales, file="sales.csv", row.names=FALSE, quote = FALSE)

* Free up some memory

gc(verbose = TRUE)

ls(all = TRUE)

rm(list = ls(all = TRUE))

ls(all = TRUE)

gc(verbose = TRUE)

new\_test <- combi[-(1:nrow(train)),]

# Accuracy model

* When you are building a predictive model, you need a way to evaluate the capability of the model on unseen data. Hence, we use the accuracy model to perform this task. We use the random forest algorithm to perform this task.

Following is the implementation of accuracy model:

library(plyr)

library(randomForest)

library(caret)

setwd(path)

training <- read.csv("new\_train.csv")

testing <- read.csv("new\_test.csv")

t<-training

test$Item\_Outlet\_Sales <- 0

combi <- rbind(train, test)

combi

# define an 80%/20% train/test split of the dataset

set.seed(2357)

in\_train <- createDataPartition( y = training$Outlet\_Type, p = 0.75, list = FALSE)

train <- t[ in\_train, ]

test <- t[ -in\_train, ]

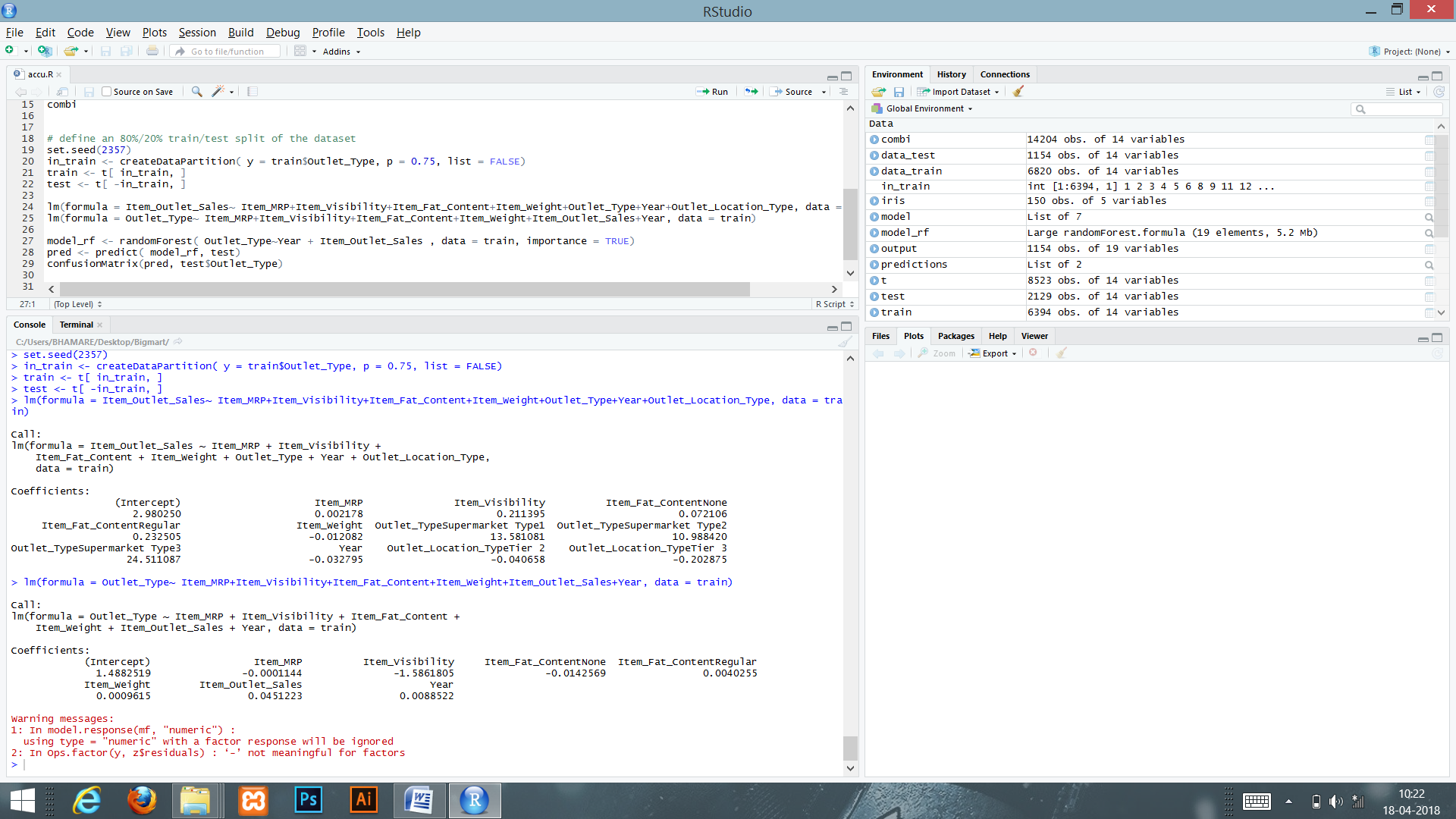
* We use regression model to find out relation of a particular parameter with the parameter. If the value is positive then there is dependency, else there is no dependency between parameters.

Here, two particular parameters used are Item\_Outlet\_Sales and Outlet\_Type.

lm(formula = Item\_Outlet\_Sales~ Item\_MRP+Item\_Visibility+Item\_Fat\_Content+Item\_Weight+Outlet\_Type+Year+Outlet\_Location\_Type, data = train)

lm(formula = Outlet\_Type~ Item\_MRP+Item\_Visibility+Item\_Fat\_Content+Item\_Weight+Item\_Outlet\_Sales+Year, data =

train)



From the above output, we understand that the parameter Outlet\_Type is highly dependent on the parameter Year and Item\_Outlet\_Sales, for naturally people will buy from a established store with good sales.

* The random forest model is used to determine accuracy on the basis of highly dependent parameters present in the dataset.

model\_rf <- randomForest( Outlet\_Type~Year + Item\_Outlet\_Sales , data = train, importance = TRUE)

pred <- predict( model\_rf, test)

confusionMatrix(pred, test$Outlet\_Type)

The output from the above code:

Confusion Matrix and Statistics

Reference

Prediction Grocery Store| Supermarket Type1| Supermarket Type2 |Supermarket Type3

Grocery Store 144 0 0 0

Supermarket Type1 0 1394 0 0

Supermarket Type2 0 0 232 0

Supermarket Type3 126 0 0 233

Overall Statistics

Accuracy : 0.9408

95% CI : (0.9299, 0.9505)

No Information Rate : 0.6548

P-Value [Acc > NIR] : < 2.2e-16

Kappa : 0.8888

Mcnemar's Test P-Value : NA

Class: Grocery Store| Class: Supermarket Type1| Class: Supermarket Type2| Class: Supermarket Type3

Sensitivity 0.53333 1.0000 1.000 1.0000

Specificity 1.00000 1.0000 1.000 0.9335

Pos Pred Value 1.00000 1.0000 1.000 0.6490

Neg Pred Value 0.93652 1.0000 1.000 1.0000

Prevalence 0.12682 0.6548 0.109 0.1094

Detection Rate 0.06764 0.6548 0.109 0.1094

Detection Prevalence 0.06764 0.6548 0.109 0.1686

Balanced Accuracy 0.76667 1.0000 1.000 0.9668