USE CASE STUDY REPORT

Group No.: Group 2

Student Names: Weijie Guan & Rui Han

Executive Summary:

We obtained the dataset of TFI company's restaurants located at 100000 sites around the world from. We preprocessed the data by deleting and converting some variables. The goal of our case study is to build regression models such as random forest and XGBoost, use them to predict new restaurants revenues in order to decide where to open new sites. In the end, we find out that random forest model has a better performance.

I. Background and Introduction

With over 1,200 quick service restaurants across the globe, TFI is the company behind some of the world's most well-known brands: Burger King, Sbarro, Popeyes, Usta Donerci, and Arby's. They employ over 20,000 people in Europe and Asia and make significant daily investments in developing new restaurant sites.

Right now, deciding when and where to open new restaurants is largely a subjective process based on the personal judgement and experience of development teams. This subjective data is difficult to accurately extrapolate across geographies and cultures.

New restaurant sites take large investments of time and capital to get up and running. When the wrong location for a restaurant brand is chosen, the site closes within 18 months and operating losses are incurred.

Finding a mathematical model to increase the effectiveness of investments in new restaurant sites would allow TFI to invest more in other important business areas, like sustainability, innovation, and training for new employees. Using demographic, real estate, and commercial data.

The goal of our study is to build a regression method and use it to predict new restaurants revenues in order to decide where to open new sites. The possible solutions are xgboost and random forest.

II. Data Exploration and Visualization

TFI has provided a dataset with 137 restaurants in the training set, and a test set of 100000 restaurants. The data columns include the open date, location, city type, and three categories of obfuscated data: Demographic data, Real estate data, and Commercial data. The revenue column indicates a (transformed) revenue of the restaurant in a given year and is the target of predictive analysis.

Id: Restaurant id.

Open Date: opening date for a restaurant.

City: City that the restaurant is in. Note that there are unicode in the names.

City Group: Type of the city. Big cities, or Other.

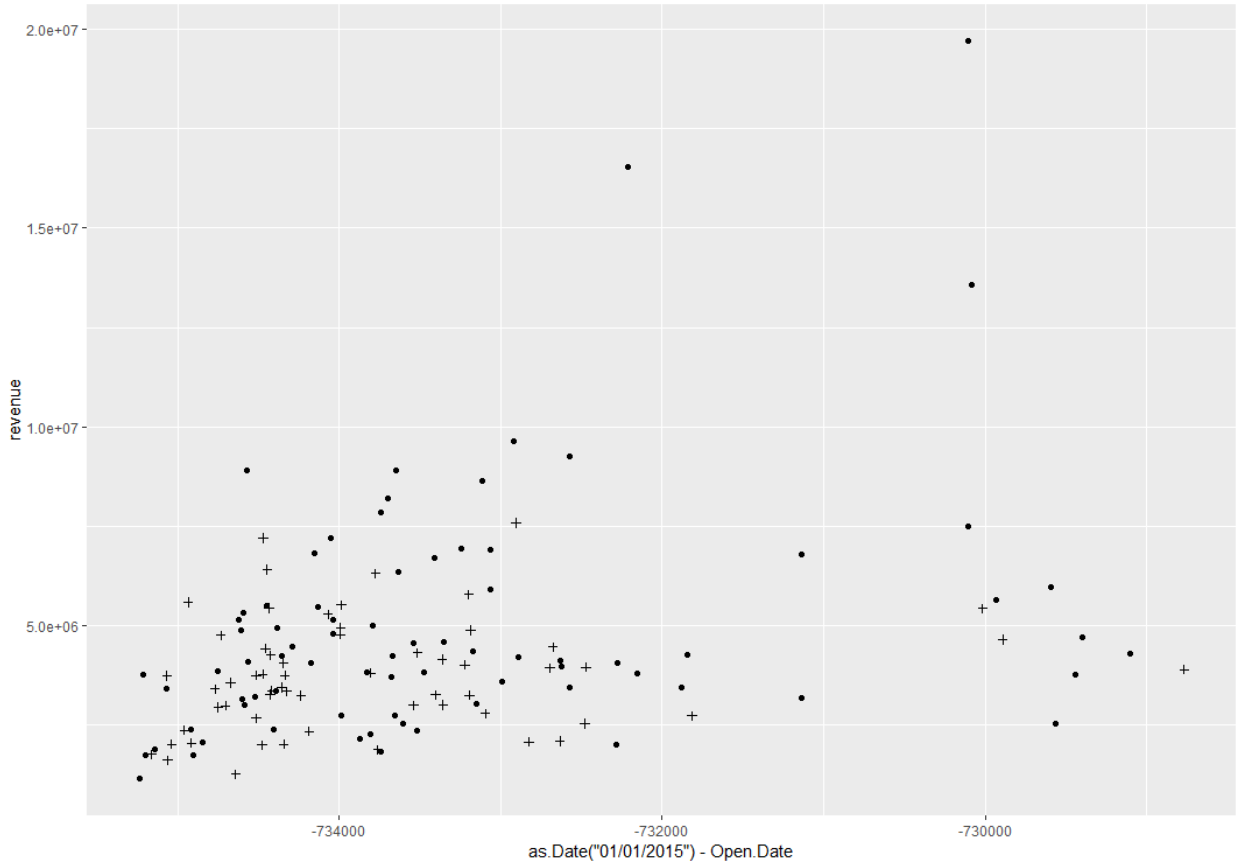
Type: Type of the restaurant. FC: Food Court, IL: Inline, DT: Drive Thru, MB: Mobile

P1, P2 - P37: There are three categories of these obfuscated data. Demographic data are gathered from third party providers with GIS systems. These include population in any given area, age and gender distribution, development scales. Real estate data mainly relate to the m2 of

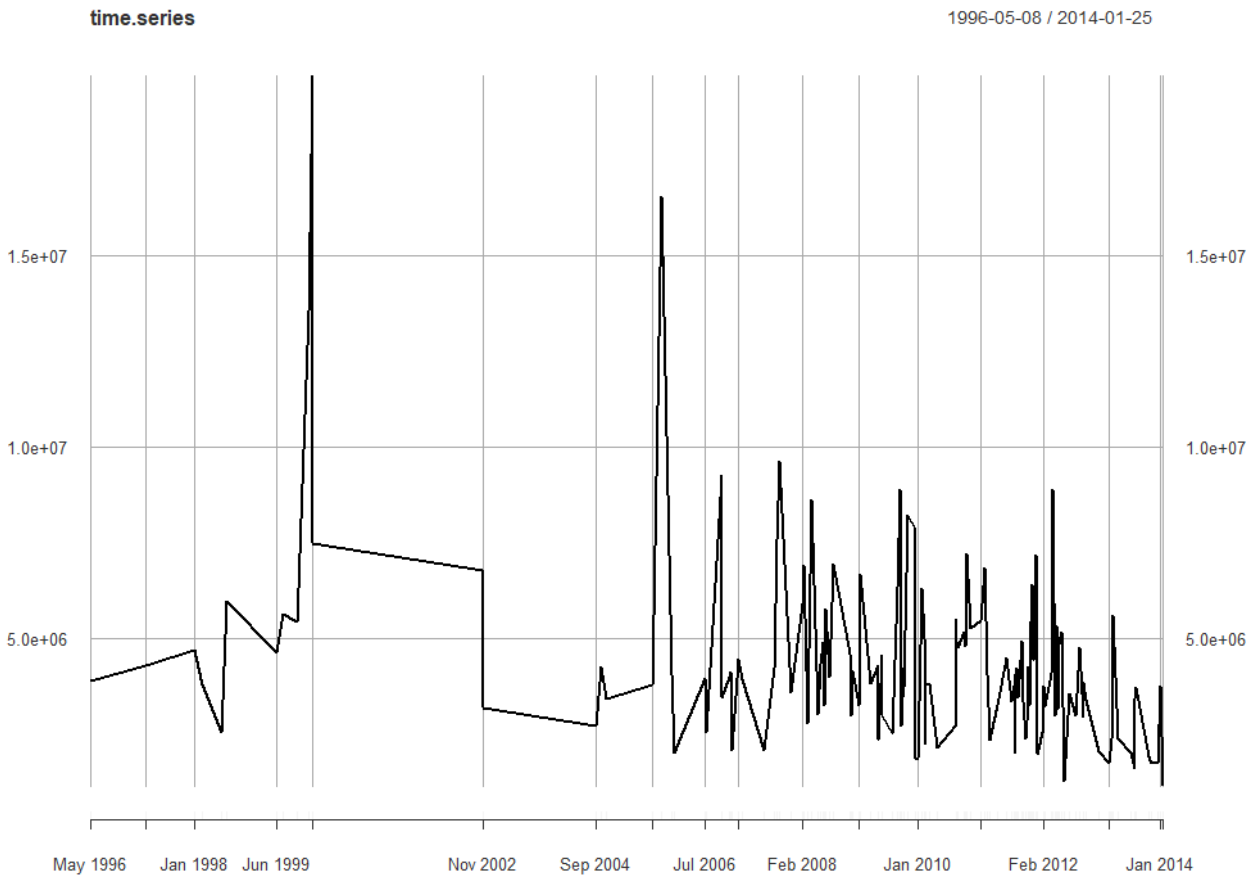
the location, front facade of the location, car park availability. Commercial data mainly include the existence of points of interest including schools, banks, other QSR operators.

Revenue: The revenue column indicates a (transformed) revenue of the restaurant in a given year and is the target of predictive analysis. Please note that the values are transformed so they don't mean real dollar values.

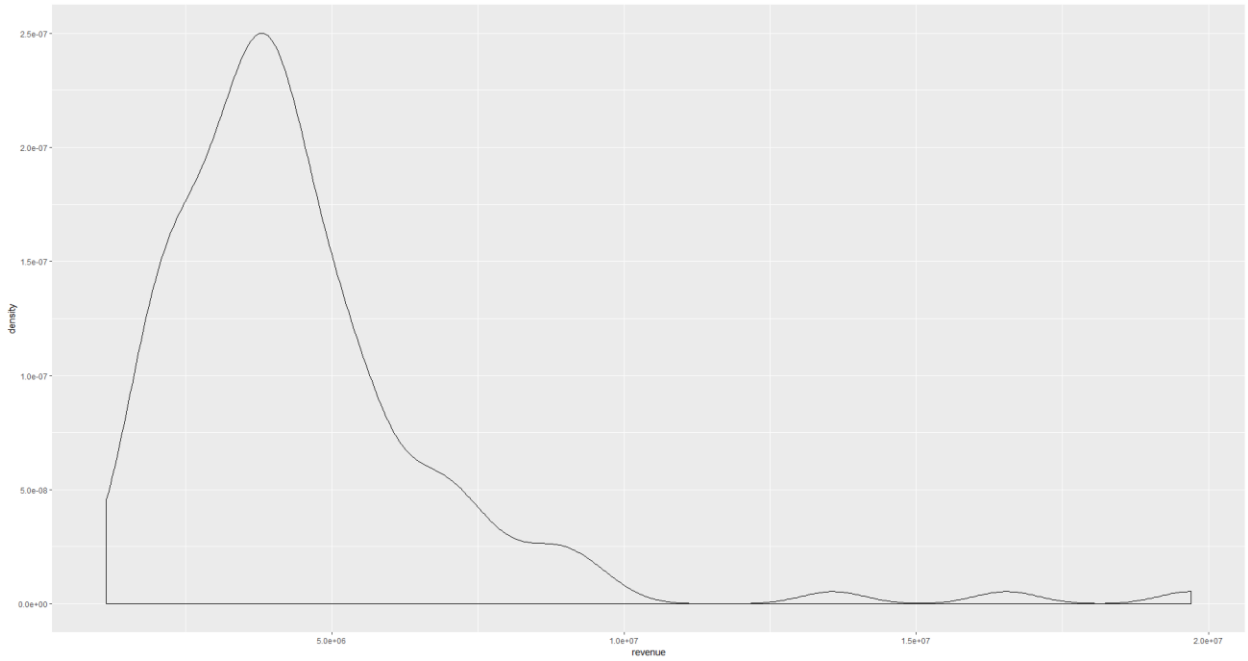
The data quality is good. It is complete, accurate, unique, valid and consistent.



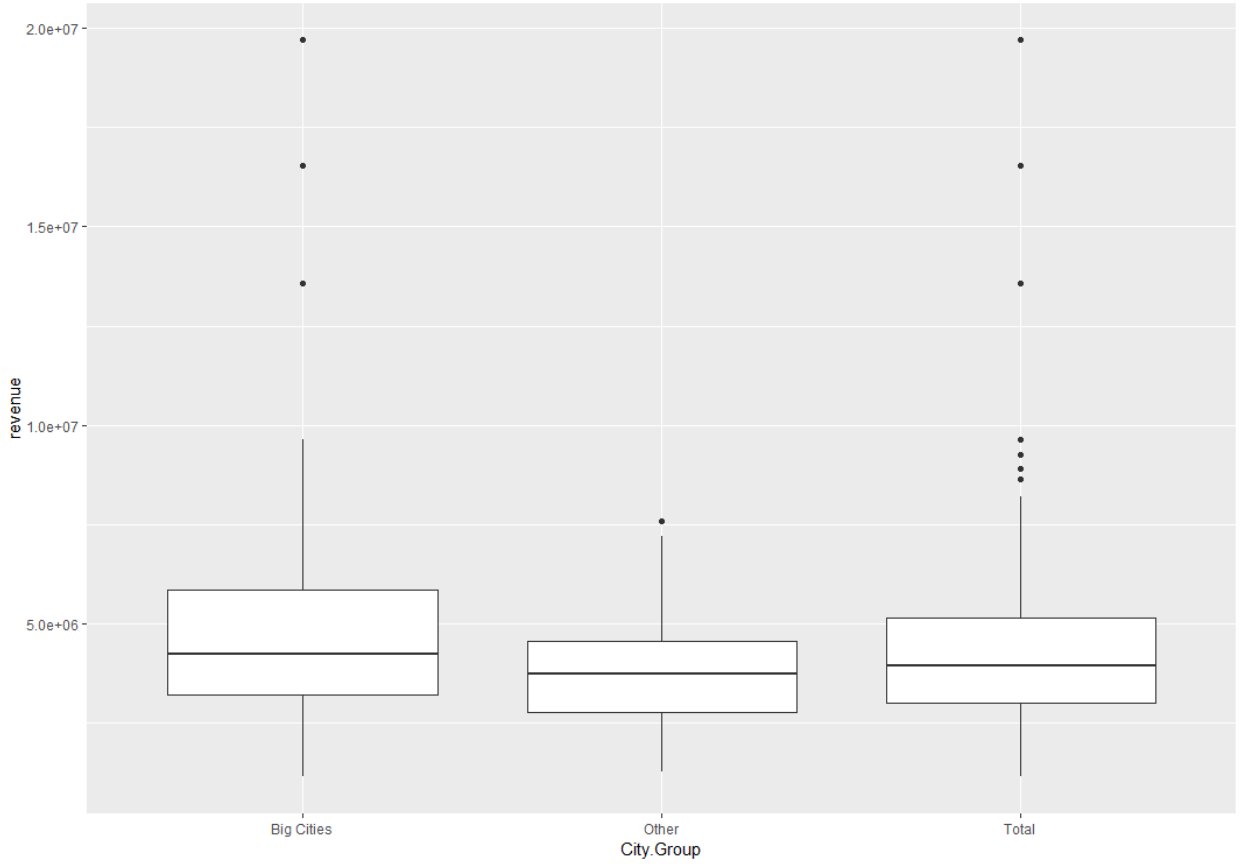
Form this scatter plot we can see that the revenue is more related to the city types. And there is no significant relationship between revenue and opendays.



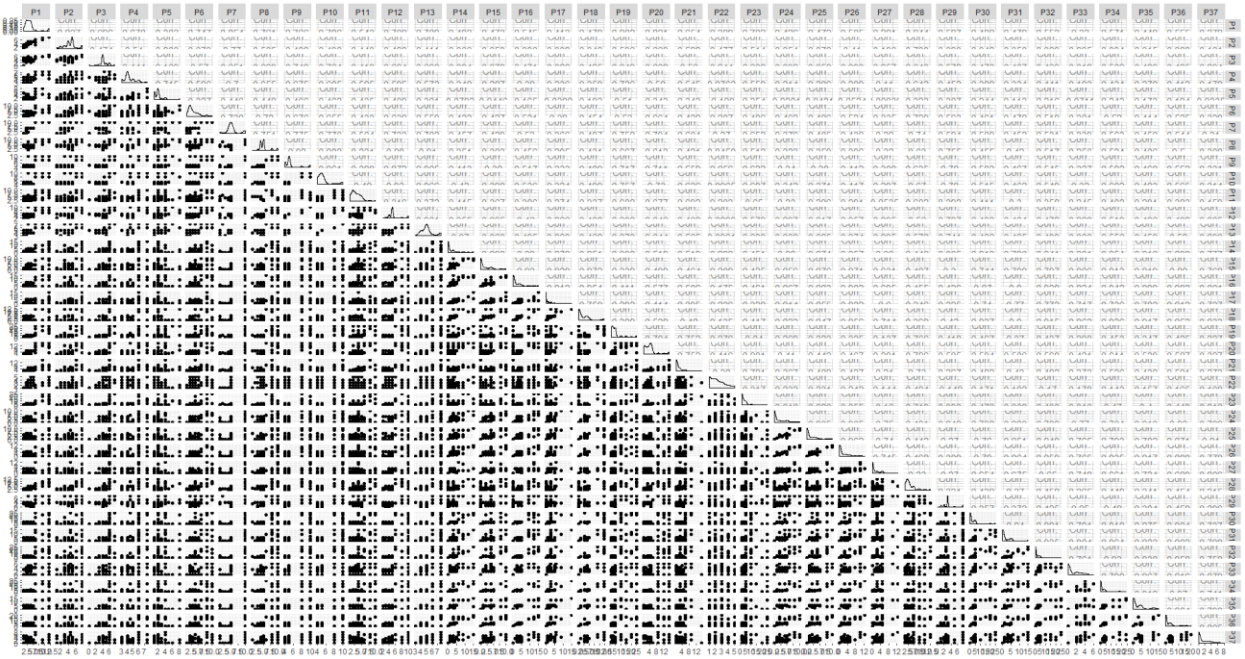
From this plot we can tell that there is no significant time series pattern.



We can see from this density plot that the revenue is left skewed and the revenue of most restaurants is around 4700000.



As we can see in the box plot, the revenue range in big cities is larger than others. There are more outliers in the big cities' boxplot. And the median revenue of restaurants in big cities is slightly higher than others.



Plot the correlation matrix of the feature P1 to P37.

III. Data Preparation and Preprocessing

1. For Random Forest method:

- Delete the "revenue" variable in the original dataset.
- Convert "Open.Date" to dates.
- Create a numeric variable "opendays", which is the days opened since 01/01/2015.
- Make sure that variables P1 – P37 have numeric values.
- Convert variable "Type" to factors with 4 levels "DT", "FC", "IL", "MB".
- Delete some outliers in the dataset and split it into training set and validation set.

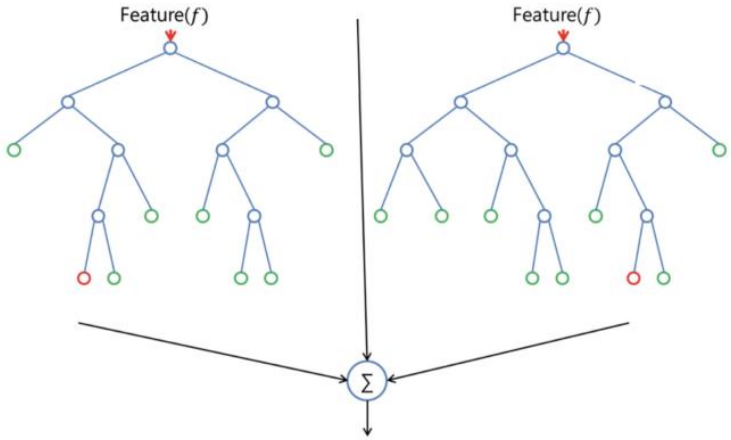
2. For XGBoost method:

- Delete the "revenue" variable in the original dataset.
- Create dummy variables for "City.Group" and "Type", add them into new dataset
- Convert "Open.Date" to dates.
- Create a numeric variable "opendays", which is the days opened since 01/01/2015.
- Combine "opendays", "ifbigcity", P1 to P37 and revenue into dataset.
- Split the dataset into training set and validation set.

IV. Data Mining Techniques and Implementation

1. Random Forest:

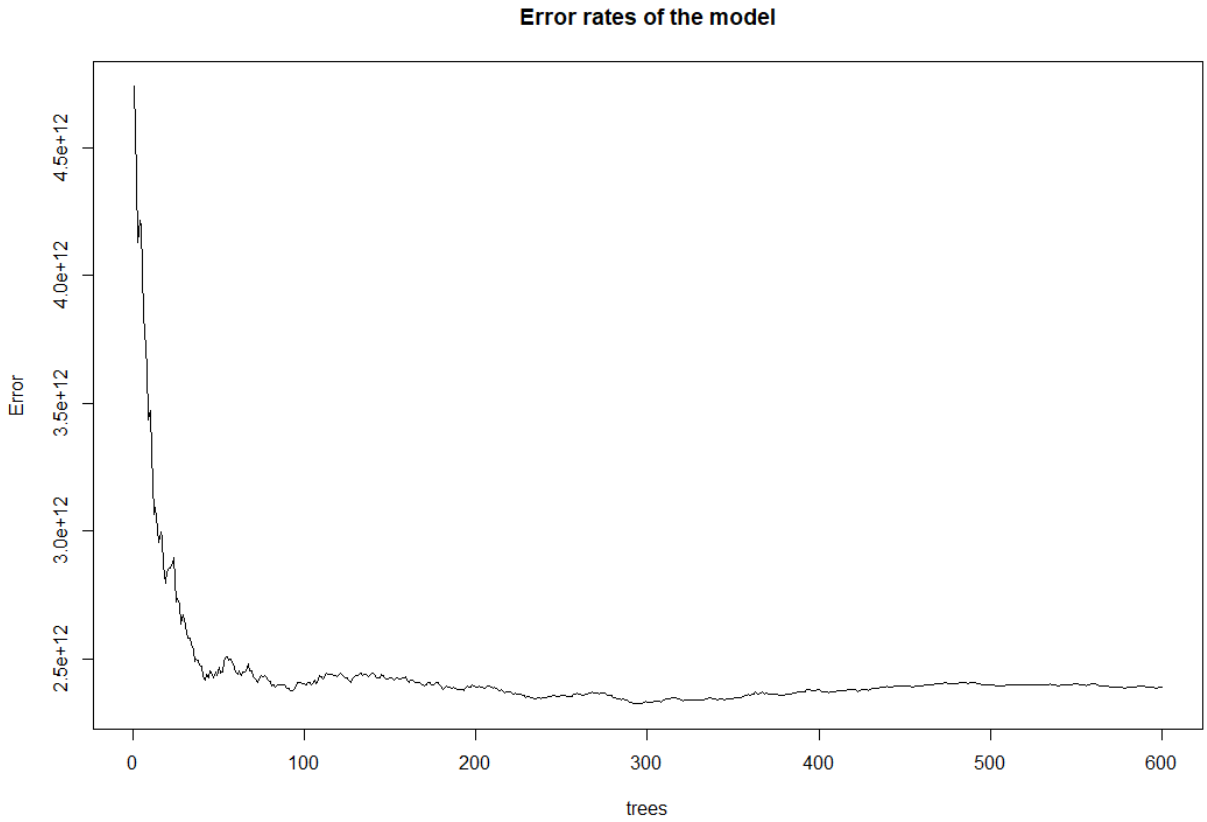
Random Forest is a supervised learning algorithm. Random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction.



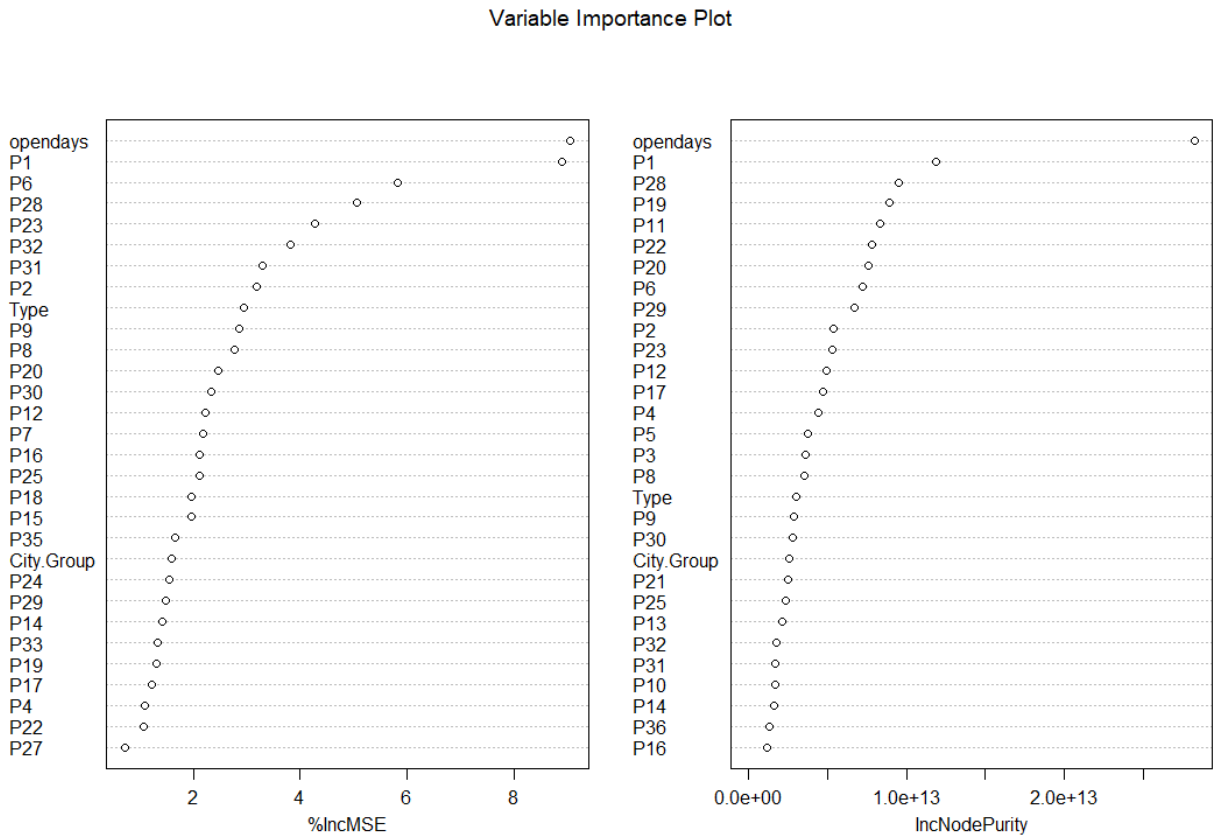
Fit the dataset into random forest model, use "revenue" as the dependent variable, training data and 600 trees.

```
Call:
 randomForest(formula = revenue ~ ., data = train.df, ntree = 600,      importance = TRUE)
               Type of random forest: regression
                     Number of trees: 600
No. of variables tried at each split: 13

          Mean of squared residuals: 2.38931e+12
                    % Var explained: 2.01
```

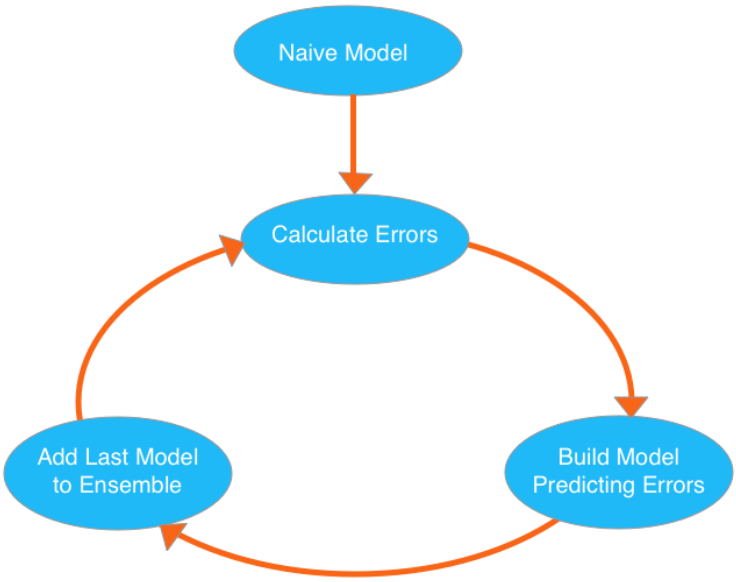


As we can see in the plot, generally the more trees we have, the less errors we get. From 0 to 50 there is a significant error rates drop, but less changes are made as the number of trees increases.



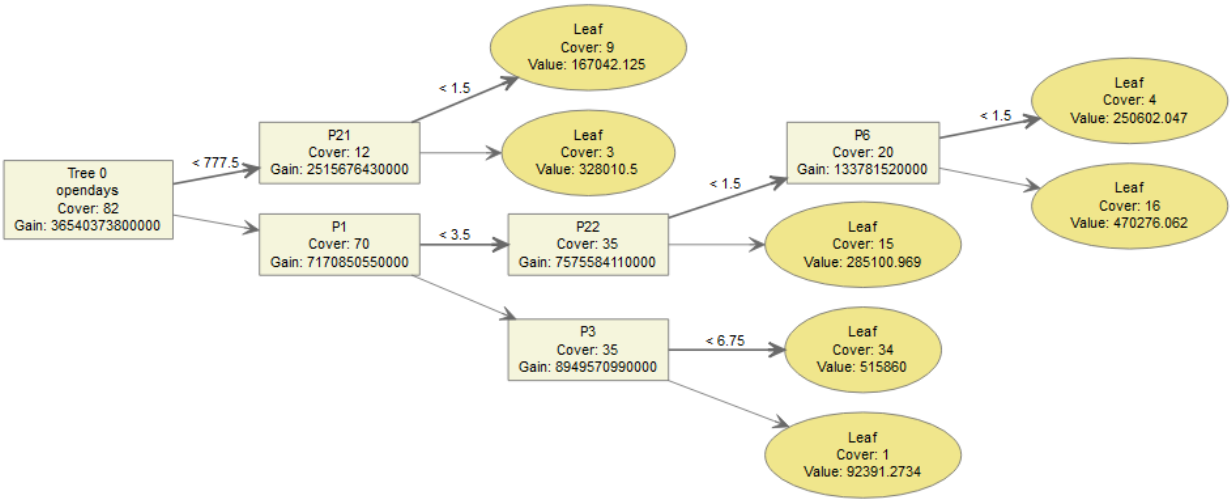
We can see from the plot that the importance of variables is sorted in decreasing order.

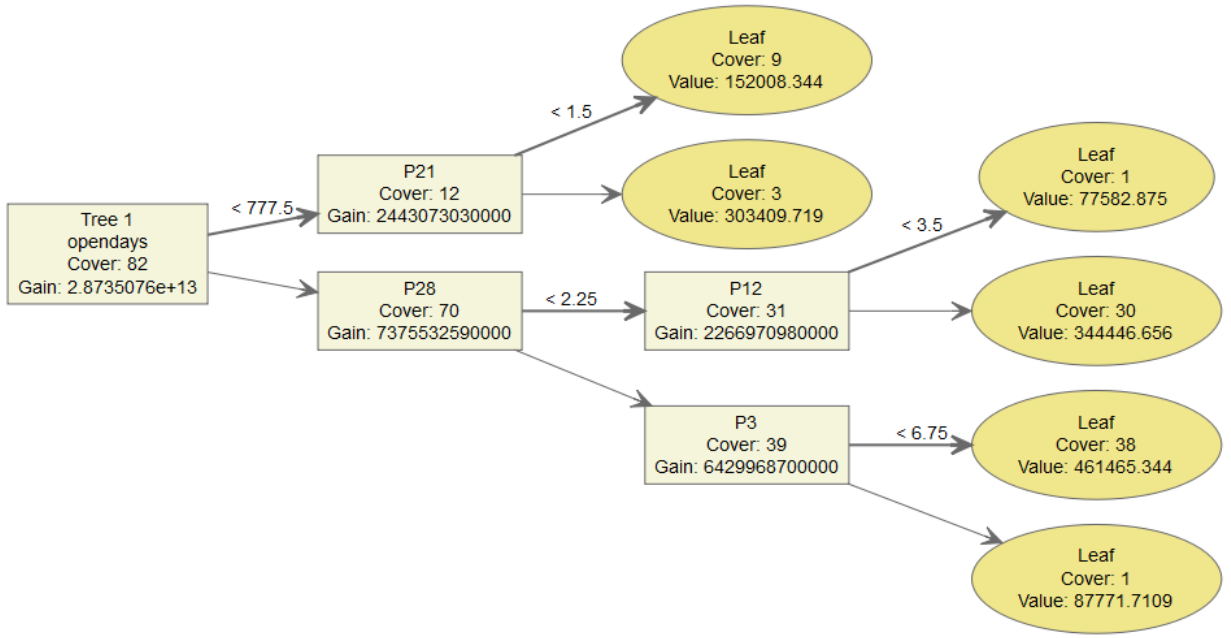
2. XGBoost

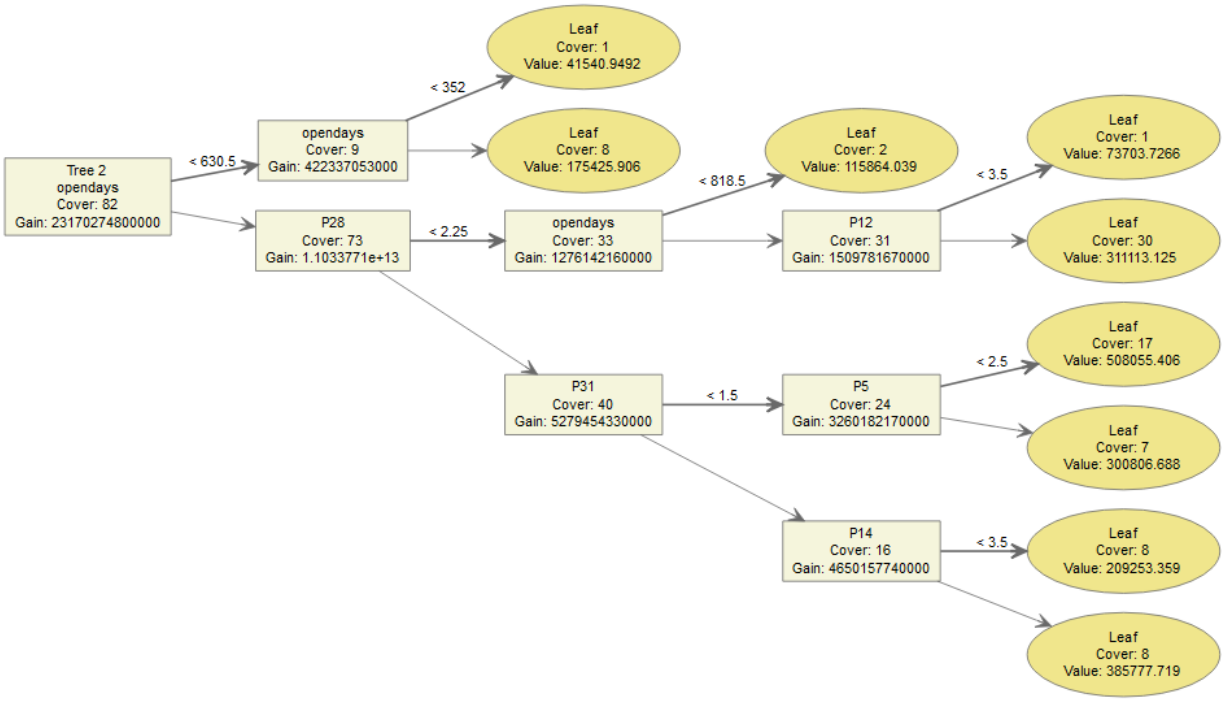


We use xgboost to go through cycles that repeatedly builds new models and combines them into an ensemble model. We start the cycle by calculating the errors for each observation in the dataset. We then build a new model to predict those. We add predictions from this error-predicting model to the "ensemble of models."

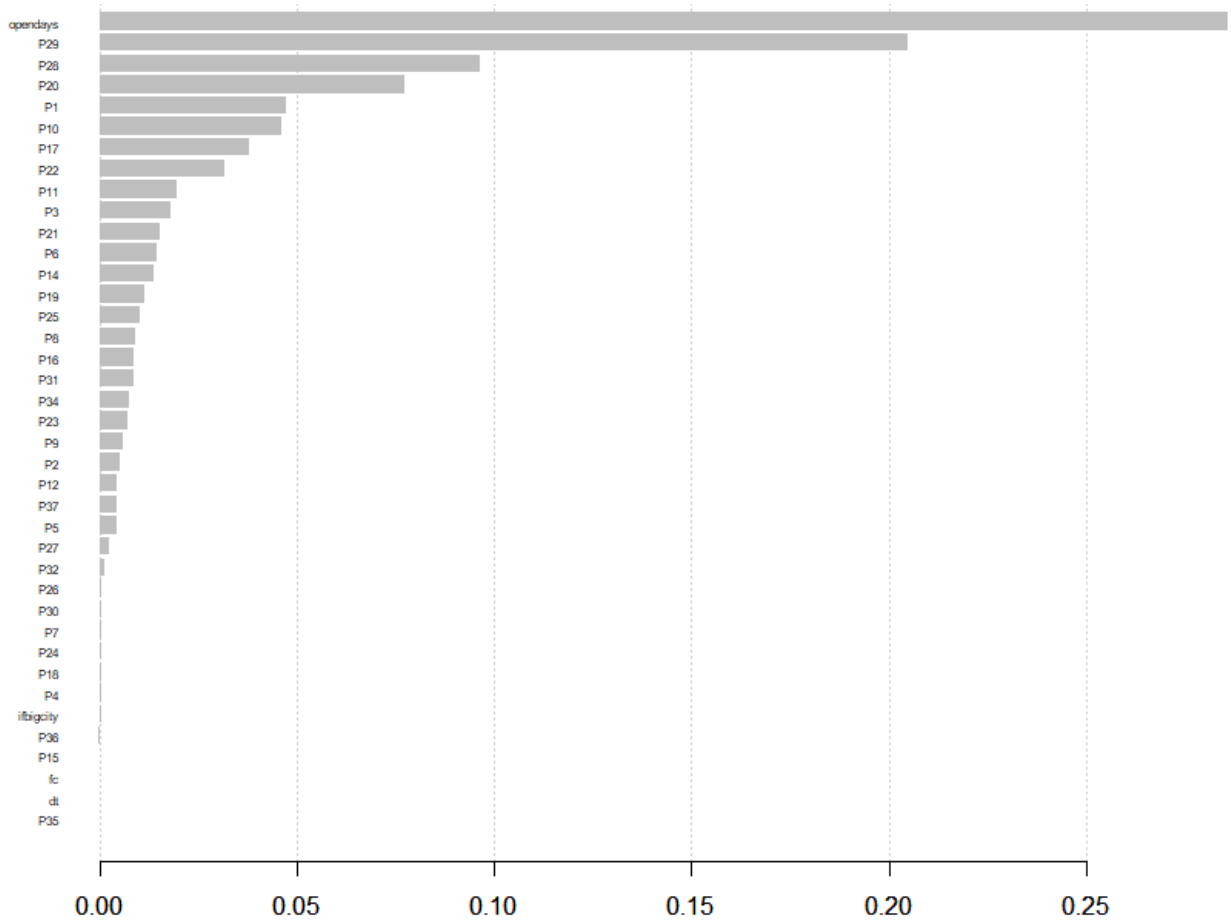
Fit the data into xgboost model using parameters nrounds = 1000 (go through the cycle above 1000 times), max_depth = 4 (maximum depth of a tree is 4), eta = 0.1 (control the learning rate), early_stopping_rounds = 10 (find a way to automatically find the ideal value, early stopping causes the model to stop iterating when the validation score stops improving).







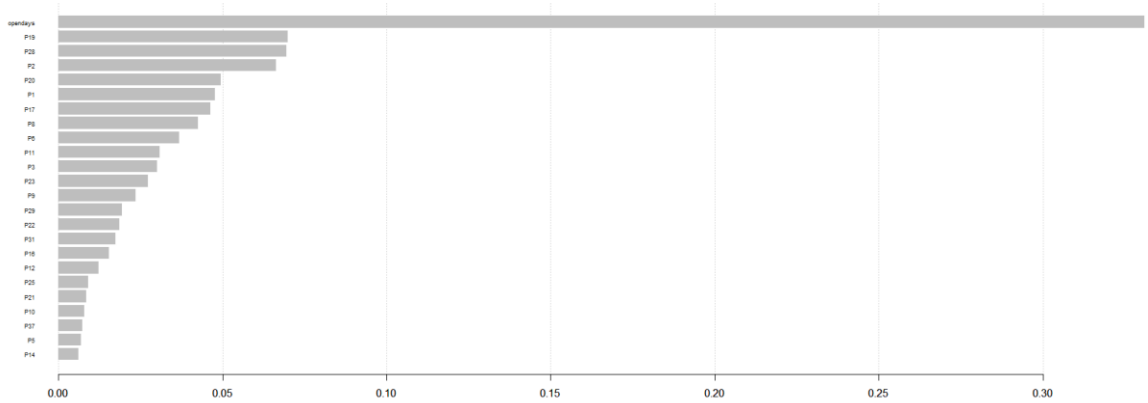
These three trees are examples from our xgboost model.



Plot the variance of importance plot and we can see the importance of features in a decreasing order.

Then we deleted some unimportant features and trained a new xgboost model with rest of the features.

And the new important plot is:



V. Performance Evaluation

For random forest method:

```
> MAE(valid.pred, valid.df[, 41])
[1] 1109142
> RMSE(valid.pred, valid.df[, 41])
[1] 1387844
```

For XGBoost method:

```
> MAE(valid.pred, valid.df[, 42])
[1] 1321249
> RMSE(valid.pred, valid.df[, 42])
[1] 1729822
```

For the tuned XGBoost method:

```
> MAE(new.valid.pred, new.valid.df[, 25])
[1] 1199296
> RMSE(new.valid.pred, new.valid.df[, 25])
[1] 1571894
```

We uploaded our code to Kaggle website, we got public scores 1761462.15075 for random forest method and 1865131.63147 for XGBoost method.

In conclusion, the performance of random forest method is better than XGBoost method on the validation dataset, so it could get a better score in Kaggle.

VI. Discussion and Recommendation

1. Random Forest:

Random forest is a bagging technique for both classification and regression. It has two parameters: number of trees and number of features to be selected at each node.

Advantages:

- It trains each tree independently, using a random sample of the data, so the training model is more reliable than a single decision tree and it is less likely to overfit.
- It is good for parallel or distributed computing, uneven datasets and missing variable.
- It calculates feature importance.

Disadvantages:

- It is much harder and time-consuming to construct
- It requires more computational resources and less intuitive.
- The prediction process is longer than other algorithms.

2. XGBoost:

XGBoost, whose name stands for eXtreme Gradient Boosting, is an implementation of the Gradient Boosted Decision Trees algorithm and it is the leading model for working with standard tabular data.

Advantages:

- It builds trees one at a time, each new tree corrects some errors made by the previous trees, so the model becomes even more expressive.
- Usually perform better than random forest, if using the correct tuning parameters.

Disadvantages:

- It is hard to tune and more prone to overfitting.
- Training takes longer time since trees are built sequentially.

Recommendations for solution:

For our case study, random forest is easy to manipulate. We don't need to do a lot of data preprocessing before using the random forest. The performance of the outcomes is good.

Potential improvement:

Deleting more outliers, select some important features and use h2o package to tune the parameters in the regression models.

VII. Summary

In this case study, we preprocessed our data by converting data types, deleting outliers, visualizing data and selecting variables. Then we split the data into training set and validation set. We use the training data to build two regression models: random forest model and XGBoost model. Then we use validation set to evaluate our model by checking the MAE and RMSE. Along with the prediction we made on the testing dataset and Kaggle score we obtain, we made a conclusion that random forest method is optimal for our case study.

Appendix: R Code for use case study

Please show the R code you generated for the use case study. Please do not show results here, only the code.

1. Code for random forest model:

```
Originaldata <- read.csv("train.csv", fileEncoding = "UTF-8")
data <- Originaldata[, -1]
data$Open.Date <- as.Date(data$Open.Date, "%m/%d/%Y")

ggplot() + geom_point(data = data[data$City.Group == "Big Cities", ],
                      mapping = aes(as.Date("01/01/2015") - Open.Date, y = revenue))
+ geom_point(data = data[data$City.Group != "Big Cities", ], mapping = aes(x = as.Date("01/01/2015")

dataset <- cbind(as.Date("01/01/2015", "%m/%d/%Y") - data$Open.Date, data[, -c(1:2)])
colnames(dataset)[1] <- "opendays"

dataset[, c(4:40)] <- lapply(dataset[, c(4:40)], as.numeric)
dataset$Type <- factor(dataset$Type, levels = c("DT", "FC", "IL", "MB"))

ggplot(data = dataset) + geom_boxplot(mapping = aes(x = City.Group, y = revenue))
+ geom_boxplot(mapping = aes(x = "Total", y = revenue))
ggplot(data = dataset) + geom_density(mapping = aes(x = revenue))

ggpairs(dataset[,c(4:40)])

Q3 <- quantile(dataset$revenue, probs = 0.75)
Q1 <- quantile(dataset$revenue, probs = 0.25)
IQR <- Q3 - Q1
outliers <- Q3 + 1.5 * IQR
dataset <- dataset[dataset$revenue < outliers, ]

set.seed(1)
train.index <- sample(c(1:dim(dataset)[1]), dim(dataset)[1] * 0.6)
valid.index <- setdiff(c(1:dim(dataset)[1]), train.index)
train.df <- dataset[train.index, ]
valid.df <- dataset[valid.index, ]

rf <- randomForest(revenue ~., data = train.df, ntree = 600, importance = TRUE)
print(rf)
plot(rf, main = "Error rates of the model")

varImpPlot(rf, main = "Variable Importance Plot")

valid.pred <- predict(rf, valid.df[, -41])
MAE(valid.pred, valid.df[, 41])
RMSE(valid.pred, valid.df[, 41])

test <- read.csv("test.csv", fileEncoding = "UTF-8")
test$Open.Date <- as.Date(test$Open.Date, "%m/%d/%Y")
test.df <- cbind(as.Date("01/01/2015", "%m/%d/%Y") - test$Open.Date, test[, -c(1:3)])
colnames(test.df)[1] <- "opendays"
test.df[, c(4:40)] <- lapply(test.df[, c(4:40)], as.numeric)

test.pred <- predict(rf, test.df, type = "response")
final = data.frame(ID = test$Id, Prediction = test.pred)
write.csv(final, "skj_predict.csv", row.names = FALSE)
```

2. Code for xgboost model:

```
Originaldata <- read.csv("train.csv", fileEncoding = "UTF-8")
data <- Originaldata[, -1]

city.type <- class.ind(data$City.Group)
dataset <- data.frame(ifbigcity = city.type[, 1])

restaurant.type <- class.ind(data$Type)
colnames(restaurant.type)
dataset$dt <- restaurant.type[, 1]
dataset$fc <- restaurant.type[, 2]

data$Open.Date <- as.Date(data$Open.Date, "%m/%d/%Y")

ggplot() + geom_point(data = data[data$City.Group == "Big Cities", ],
                      mapping = aes(as.Date("01/01/2015") - Open.Date, y = revenue))
+ geom_point(data = data[data$City.Group != "Big Cities", ],
             mapping = aes(x = as.Date("01/01/2015") - Open.Date, y = revenue), shape = 3)

dataset <- cbind(as.Date("01/01/2015", "%m/%d/%Y") - data$Open.Date, dataset, data[, c(5:42)])

colnames(dataset)[1] <- "opendays"

ggplot(data = data) + geom_boxplot(mapping = aes(x = City.Group, y = revenue))
+ geom_boxplot(mapping = aes(x = "Total", y = revenue))

Q3 <- quantile(dataset$revenue, probs = 0.75)
Q1 <- quantile(dataset$revenue, probs = 0.25)
IQR <- Q3 - Q1
outliers <- Q3 + 1.5 * IQR
dataset <- dataset[dataset$revenue < outliers, ]

set.seed(1)
train.index <- sample(c(1:dim(dataset)[1]), dim(dataset)[1] * 0.6)
valid.index <- setdiff(c(1:dim(dataset)[1]), train.index)

dataset$opendays <- as.numeric(dataset$opendays)
train.df <- dataset[train.index, ]
valid.df <- dataset[valid.index, ]

model <- xgboost(data = as.matrix(train.df[, -42]), label = train.df[, 42], nrounds = 1000,
                 max_depth = 4, eta = 0.1, early_stopping_rounds = 10)
m <- xgb.dump(model, with_stats=T)
names <- dimnames(data.matrix(dataset[,-42]))[[2]]
importance_matrix <- xgb.importance(names, model = model)
xgb.plot.importance(importance_matrix)
xgb.plot.tree(model = model, trees = 0)
xgb.plot.tree(model = model, trees = 1)
xgb.plot.tree(model = model, trees = 2)

valid.pred <- predict(model, as.matrix(valid.df[, -42]))
MAE(valid.pred, valid.df[, 42])
RMSE(valid.pred, valid.df[, 42])
```

```
new.train.df <- train.df %>% select(-c(P27, P32, P26, P30, P7, P24, P18, P4,
                                       ifbigcity, P36, P15, fc, dt, P35, P33, P34, P13))
new.valid.df <- valid.df %>% select(-c(P27, P32, P26, P30, P7, P24, P18, P4,
                                       ifbigcity, P36, P15, fc, dt, P35, P33, P34, P13))

new.model <- xgboost(data = as.matrix(new.train.df[, -25]),
                     label = new.train.df[, 25], nrounds = 1000, max_depth = 4, eta = 0.1,
                     early_stopping_rounds = 10)
new.m <- xgb.dump(new.model, with_stats=T)
new.names <- dimnames(data.matrix(new.train.df[,-25]))[[2]]
new.importance_matrix <- xgb.importance(new.names, model = new.model)
xgb.plot.importance(new.importance_matrix)

new.valid.pred <- predict(new.model, as.matrix(new.valid.df[, -25]))
MAE(new.valid.pred, new.valid.df[, 25])
RMSE(new.valid.pred, new.valid.df[, 25])

test <- read.csv("test.csv", fileEncoding = "UTF-8")
test$Open.Date <- as.Date(test$Open.Date, "%m/%d/%Y")

city.type <- class.ind(test$City.Group)
test.df <- data.frame(ifbigcity = city.type[, 1])

restaurant.type <- class.ind(test$Type)
colnames(restaurant.type)
test.df$dt <- restaurant.type[, 1]
test.df$fc <- restaurant.type[, 2]

test.df <- cbind(as.Date("01/01/2015", "%m/%d/%Y") - test$Open.Date, test.df, test[, c(6:42)])
colnames(test.df)[1] <- "opendays"
test.df$opendays <- as.numeric(test.df$opendays)

test.df <- test.df %>% select(-c(P27, P32, P26, P30, P7, P24, P18, P4,
                                 ifbigcity, P36, P15, fc, dt, P35, P33, P34, P13))

test.pred <- predict(new.model, as.matrix(test.df))
final = data.frame(ID = test$Id, Prediction = test.pred)
write.csv(final, "skj_predict.csv", row.names = FALSE)
```