Steps in Predictions of Prices, Guests included and Extra price

1. System Configuration: Local system, 8 Gb RAM, 320 GB Hard Disk
2. R version: 3.3.3, system: x86_64, mingw32

Loading the data:

Data was read to memory using read.csv function. A total of 30000 rows with 54 variables were read into the memory for the training set.

Data Cleaning:

Next step after data load is data clean.

Removing columns having more than 50% NA values

NA values in every column were counted and columns having more than 50% of NA values are removed.

As a result, columns "**Square_feet**" and "**security_deposit**" got removed

Removing columns providing very less or no value

1. "**State**" column is always filled as newyork . As no new information is received from this column, it was removed.
2. "**ID**" column is also removed as it is unique for each row
3. "**Smart_location**" column is combination of city. and state. As we have city and state variables in data, this will not provide much information for forcasting, this removing this.
4. Checking "**Market**" column, I found that it has 29264 "NY" value and 681 NA values out of 30000 rows so that will not be useful, I deleted that column too.

So total 6 columns **Square_feet, security_deposit, State, Smart_location, market and id** are removed.

Converting variables for Numeric or Factor

I checked "**price**", **"extra people"** and **"cleaning fee"** are character because of $ sign and "**host_acceptance_rate**", "**host_response_rate"** bacause of % sign.

I converted them to numeric by removing these characters and taking as.numric.

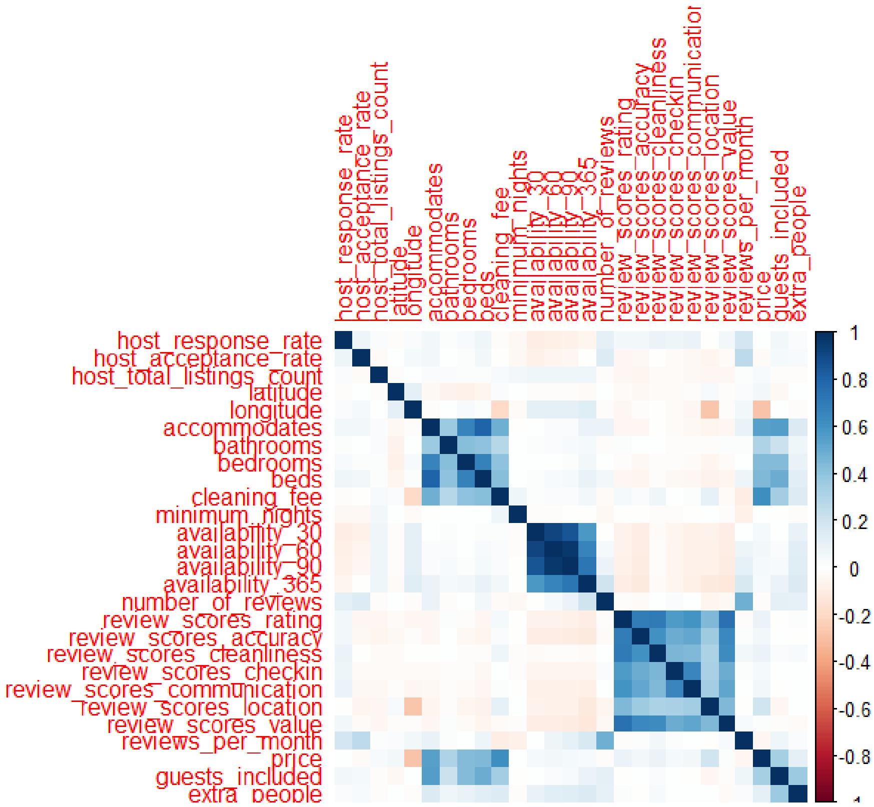
Converted "**zip code**" to factor from numeric.

Then I converted all integers variables to numeric and character variable to factors.

Co-Relation Graph:

I checked correlation of numeric variables and created graph.

I didn't remove any variable from dataset using correlation because there was not very powerful correlation between variables.



Formatting Street, amenities and Host_verification

Now I found that columns “**street**”, “**amenities**” and “**host_varification**” are character variable with many character values in one row. We must divide these columns in to different columns.

Amenities

In amenities I first remove all extra characters and took all uniqe values of column. I create a data frame of all unique values of amenities with 1 or 0 value. 1 if value is in that row and 0 if value is not in that row. A total of 45 number of unique features were generated

Host_verification

A process similar to Amenities was done for Host_verification. A total of 15 variables were generated.1

```
data <- as.character(data)
data=gsub("\\[", '', data)
data=gsub("]", '', data)
data=gsub(",", '', data)
data=gsub(" ", '', data)
dataSplit <- strsplit(data, "")
allWords <- unique(unlist(dataSplit))
dtm <- lapply(allWords, function(amenity){
as.vector(sapply(dataSplit, function(amenities_row) {
return(amenity %in% amenities_row)
}))
})
out <- do.call(cbind, dtm)
```

Street

In street I took first value of all characters of that row. A total of 1 new variables were generated as a result which is only street removing city, state.

I mergeed these all data frames with original data and deleted these 3 columns, which created 105 total coulmns in data.

Reducing factor levels:

1. Convert “**neighbourhood_cleansed**” to lower case by removing spaces to decrease factors.
2. Combined “**cities**” to “other” city if city count is less than 6 by first removing extra characters, spaces. Combine “brooklyn” and similar cities same for other cities.
3. Similarly for “**property_type”** removed spaces and converted to lower case.

Imputing NA values of Numeric variables

After performing above steps, I filled all numerical NA values with mean of that column.

I tried other techniques like Mice & Amenia package , but as they were taking a lot of time, were skipped.

Linear model

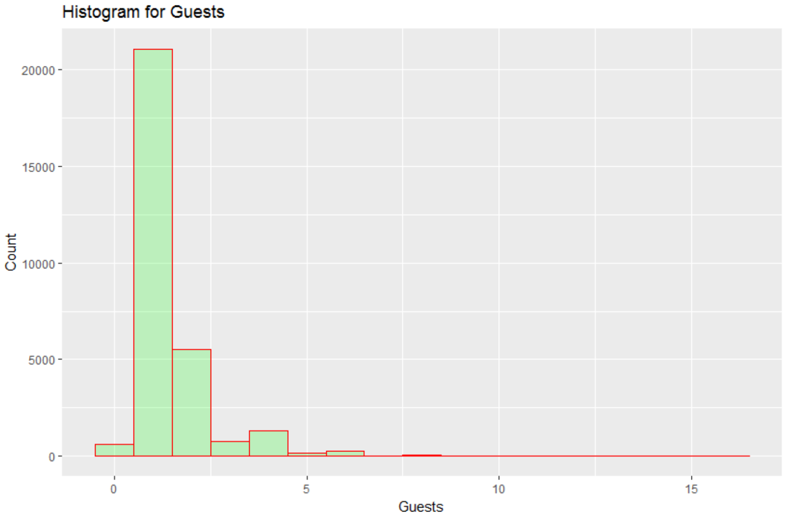
Now I ran linear model on price variable after removing guest_included and extra_people. Linear model took a lot of time and results were also not good, so I decided to do more data cleaning.

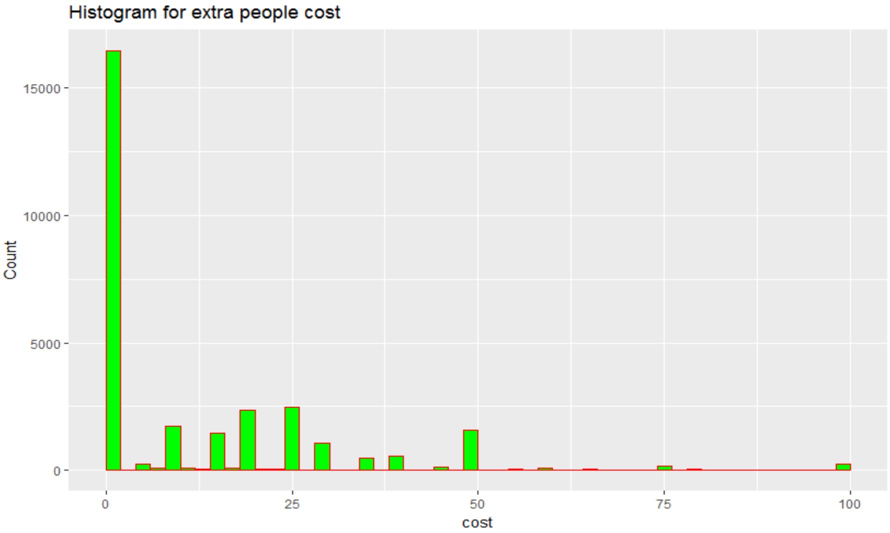
Miscellaneous cleaning:

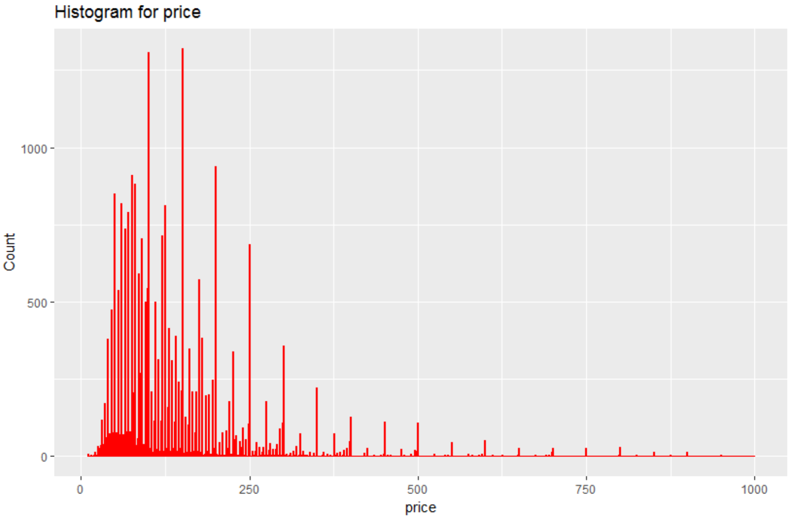
1. I checked all factor variables and found that “**profile pic**” has only 82 rows false out of 30000 rows and rest are true, so I removed this variable.
2. Now I checked some categorical variables has 3 NA rows and all these rows are same, so I removed these 3 rows.
3. I removed rows where price is NA, as there is target value can’t be NA.
4. I found that a new variable “**None**” was generated in the process of expanding amenities column, which was not useful. This column was also removed.
5. Few rows in “**property_type**” is NA so I replaced it with “apartment” because “apartment” has more than 50% value in property_type.
6. “**city**” NA values were replaced with “newyork” as “newyork” has 50% value in city.
7. “**zipcode**” NA values were replaced with 0.

Now we have no NA values in in all columns and total 103 columns in which 100 are independent variables and 3 are dependent variables.

Histograms of all 3 independent variables:







Model on Price Variable

This is a regression problem because predicted variable is continues. For this problem I tried Linear regression, Decision Tree, Random Forest and XGBoost algorithm.

Linear Model

Now I applied linear regression on price variable. It took a lot of time and training & testing error was also high, so I decided to first extend train data to convert categorical data into dummy data.

After **expansion** of data I got 29845 rows with 4055 columns. Now I applied linear regression on this data after removing columns “extra_people” and “guests_included” (From now expanded set for all models and all variables is used for training and testing).

On linear model, with default parameters, training error and testing error was very high, so I decided to move on without any change in this model. I can also try polynomial regression but independent variables are 4000 in number so there will be very high number of variables and my machine will not support that.

```
extand_linear_model= lm(formula = price~ .,data = price_training_setExtended)
summary(xtand_linear_model)
y_pred = predict(extand_linear_model, newdata = price_test_set_Extended )
error_linear_model<- error(y_pred, price_test_set_Extended$price )
```

Decision Tree:

Then I applied decision tree on data. Now training error was very low and testing error was high so that was a problem of overfitting. Then I tried with important variable given by decision tree model and again apply decision tree. Training and testing error didn’t change by big number. Decision tree model gave overfitting results.

I also tweaked multiple parameters to reduce the overfitting, but the results did not change much. So I moved on to Random Forest Model as it is know to not overfit.

```
decision=rpart(formula=price ~., data = price_training_setExtended, method="anova")
pred_decision <- predict(decision, price_test_set_Extended)
error_decisiontree_model<- error(pred_decision, price_test_set_Extended$price )  #testing error
error_decisiontree_training <- error(decision$y, price_training_setExtended$price)  #training error

# Choosing important variable from decision tree
dt_importance <- as.matrix(sort(decision$variable.importance, decreasing = TRUE))
rows<-row.names(dt_importance)
new_data <- select(priceExpanded, one_of(rows))
new_data$price = train$price
```

Random Forest model:

I applied random Forest on this dataset using package ranger with default parameters. Again, training and Testing error was both high. I did parameter tuning and tried random forest with important variables but was not able to lower the error less than 4000. I tried with top 100, 500, 1000 variables and also changed values of **number of tress** from 200-1000. I spend a lot of time on this model and choose to move on. I applied random forest with “**ranger**” package because random forest package was taking a lot of time in run. This model is time consuming and after parameter tuning and variable importance error didn’t change by large amount.

```
random_forest <-  ranger(price ~ ., data = price_training_setExtended,
                         num.trees = 500, num.threads = 8, importance)
rf_predictions <- predict(random_forest, price_test_set_Extended)
error_rf_model<-  error(rf_predictions$predictions, price_test_set_Extended$price ) #test error
error_rf_model_train <- error(random_forest$predictions , price_training_setExtended$price)  #train error


#Taking important variables from random_forest method
rf_importance <- as.matrix(sort(random_forest_extra$variable.importance, decreasing = TRUE))
rows<-row.names(rf_importance)
rows=rows[1:100] #taking top 100 varibles  # i also tried for 200, 500, 1000 variables
```

XGBoost Model:

XGBoost model is very powerful for big dataset, it is very less time consuming and give powerful results.

I first tried XGboost with random parameters and got some results. Then I did parameter tuning on parameters **eta, max_depth, subsample, colsample_bytree, nrounds**.

Explanation of tuning variable:

Eta: Makes the model more robust by shrinking the weights on each step

Max_depth: maximum depth of a tree

Subsample: Denotes the fraction of observations to be randomly samples for each tree

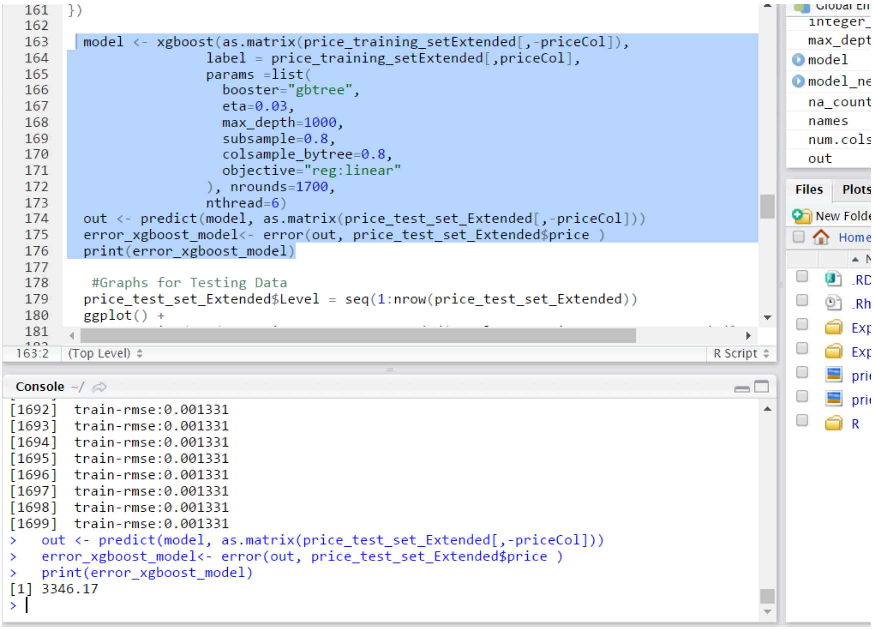
colsample_bytree: Denotes the fraction of columns to be randomly samples for each treenrounds : number of iterations

I run loop on these variables and found that for what values testing error is minimum. I did more parameter tuning, and was able to calculate minimum testing error MSE (mean square error) equal to 3340 approximately.

I did k-fold cross validation test on this model with k=10 and got average error 3390 approximately.

I decide to check error after variable importance with same tuning parameters. I choose 100, 500, 1000, 2000 important variable with help of previous result and again run XGboost model, but I found that error didn't decrease. Error was little higher than previous model. I did parameter tuning in this model also but error was not less than 3400, so I decided to choose model with error 3340 and predicted values from this model. I could have tried more models but because of time and machine constraint I was not able to.

My final testing error(MSE) in this model was 3340 and training error was very low. It is overfitted model and we can solve this model further to remove overfitting.



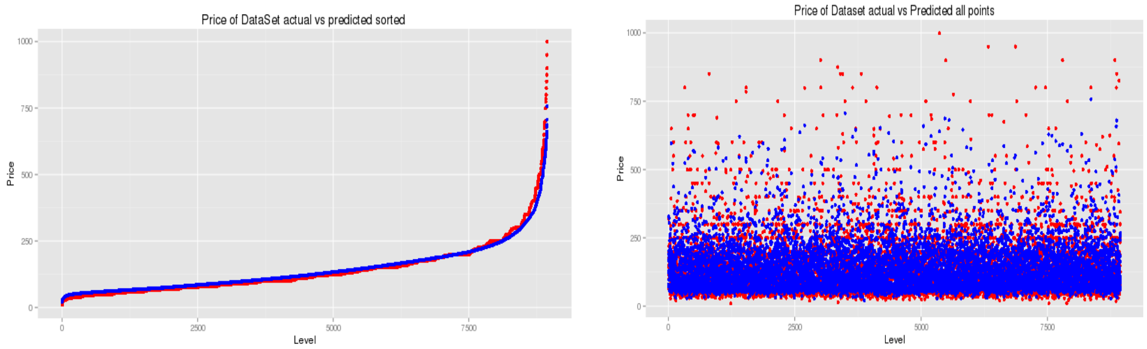
```
})

 model <- xgboost(as.matrix(price_training_setExtended[,-priceCol]),
                label = price_training_setExtended[,priceCol],
                params =list(
                  booster="gbtree",
                  eta=0.03,
                  max_depth=1000,
                  subsample=0.8,
                  colsample_bytree=0.8,
                  objective="reg:linear"
                ), nrounds=1700,
                nthread=6)
 out <- predict(model, as.matrix(price_test_set_Extended[,-priceCol]))
 error_xgboost_model<- error(out, price_test_set_Extended$price )
 print(error_xgboost_model)

  #Graphs for Testing Data
 price_test_set_Extended$Level = seq(1:nrow(price_test_set_Extended))
 ggplot() +
```

```
[1692]  train-rmse:0.001331
[1693]  train-rmse:0.001331
[1694]  train-rmse:0.001331
[1695]  train-rmse:0.001331
[1696]  train-rmse:0.001331
[1697]  train-rmse:0.001331
[1698]  train-rmse:0.001331
[1699]  train-rmse:0.001331
>  out <- predict(model, as.matrix(price_test_set_Extended[,-priceCol]))
>  error_xgboost_model<- error(out, price_test_set_Extended$price )
>  print(error_xgboost_model)
[1] 3346.17
> |
```

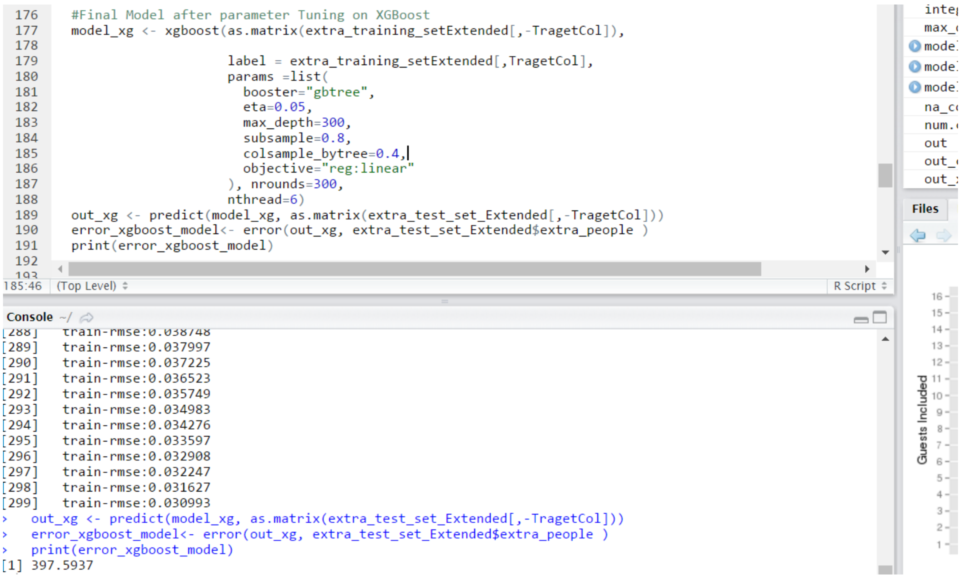
Graphs of actual vs predicted for price



Prediction of extra_prople_cost:

This problem is also a regression problem. I followed same path as I followed in previous model but this time I didn’t take important features from decision tree to calculate error.

My final error was calculated from XGboost model after parameter tuning is 397.59. I was not able to create model by selecting important variables on this feature because of machine and time constraint.

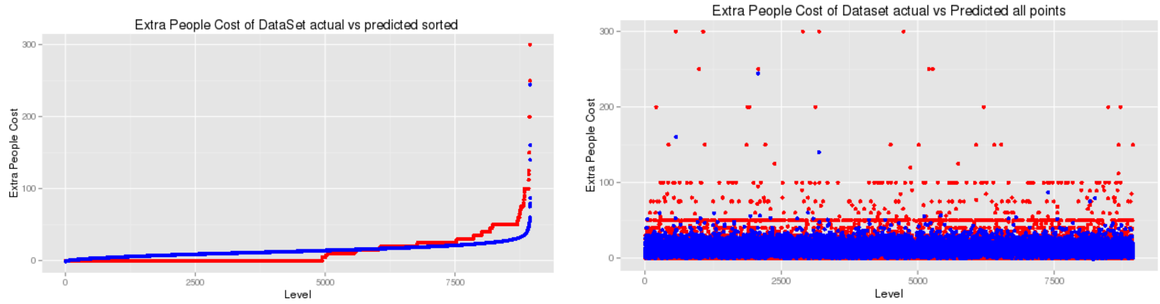


```
#Final Model after parameter Tuning on XGBoost
model_xg <- xgboost(as.matrix(extra_training_setExtended[,-TragetCol]),

                    label = extra_training_setExtended[,TragetCol],
                    params =list(
                      booster="gbtree",
                      eta=0.05,
                      max_depth=300,
                      subsample=0.8,
                      colsample_bytree=0.4,
                      objective="reg:linear"
                    ), nrounds=300,
                    nthread=6)
out_xg <- predict(model_xg, as.matrix(extra_test_set_Extended[,-TragetCol]))
error_xgboost_model<- error(out_xg, extra_test_set_Extended$extra_people )
print(error_xgboost_model)
```

```
[288]   train-rmse:0.038748
[289]   train-rmse:0.037997
[290]   train-rmse:0.037225
[291]   train-rmse:0.036523
[292]   train-rmse:0.035749
[293]   train-rmse:0.034983
[294]   train-rmse:0.034276
[295]   train-rmse:0.033597
[296]   train-rmse:0.032908
[297]   train-rmse:0.032247
[298]   train-rmse:0.031627
[299]   train-rmse:0.030993
>  out_xg <- predict(model_xg, as.matrix(extra_test_set_Extended[,-TragetCol]))
>  error_xgboost_model<- error(out_xg, extra_test_set_Extended$extra_people )
>  print(error_xgboost_model)
[1] 397.5937
```

Graphs of actual vs predicted for extra people cost



Model on guests Included

Values in guests in this column ranges from 0 to 16, so the main confusion was to consider this as regression problem or classification problem.

I decided to calculate 2 errors for this problem.

1. Error(MSE) as in regression: mean of square of (actual values – floor (predicted values))
2. Accuracy as in classification: correct predicted/Total values

Random Forest and SVM:

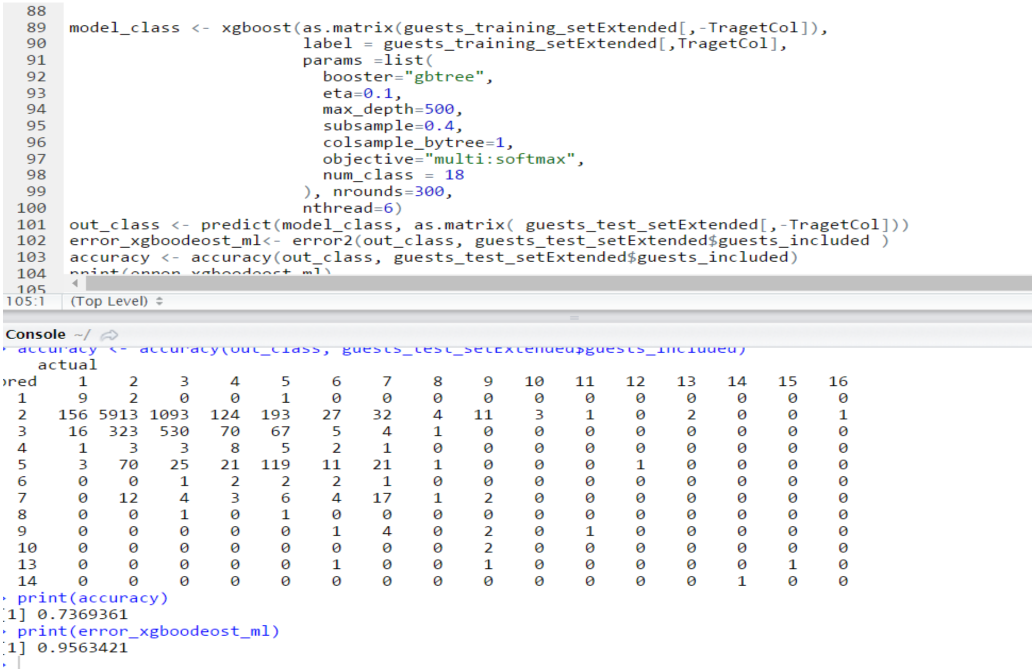
I first started with classification on random forests and calculate error and accuracy. Error was high and accuracy was low so I decided to move on and tried SVM model. SVM gave better accuracy than random forest with less error. I decided to do parameter tuning on SVM but this model was taking very much time. I waited very long to execute that code but didn't get results so I decided to stop this and try another Model.

XGboost:

First I tried XGboost for this problem as classification and regression to compare which model will be better for this type of problem. I found that classification is working better than regression as I got 10% more accuracy in classification than regression and less error, so I decided to work as a classification problem.

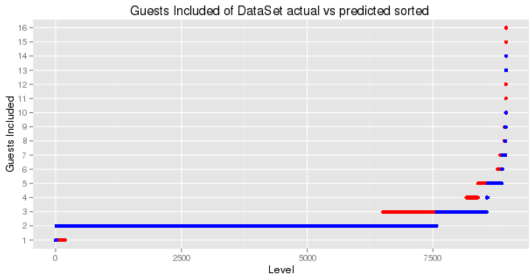
I did parameter tuning to get best results, and after a lot of time a got final results.

My final accuracy was 73.69% and error(MSE) was 0.9563.

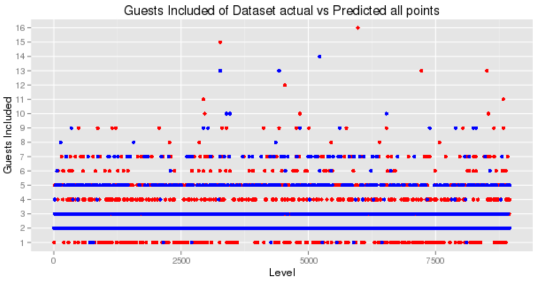


```
model_class <- xgboost(as.matrix(guests_training_setExtended[,-TragetCol]),
                        label = guests_training_setExtended[,TragetCol],
                        params =list(
                          booster="gbtree",
                          eta=0.1,
                          max_depth=500,
                          subsample=0.4,
                          colsample_bytree=1,
                          objective="multi:softmax",
                          num_class = 18
                        ), nrounds=300,
                        nthread=6)
out_class <- predict(model_class, as.matrix( guests_test_setExtended[,-TragetCol]))
error_xgboodeost_ml<- error2(out_class, guests_test_setExtended$guests_included )
accuracy <- accuracy(out_class, guests_test_setExtended$guests_included)
print(error_xgboodeost_ml)
```

```
> accuracy <- accuracy(out_class, guests_test_setExtended$guests_included)
   actual
pred    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16
  1     9    2    0    0    1    0    0    0    0    0    0    0    0    0    0    0
  2   156 5913 1093  124  193   27   32    4   11    3    1    0    2    0    0    1
  3    16  323  530   70   67    5    4    1    0    0    0    0    0    0    0    0
  4     1    3    3    8    5    2    1    0    0    0    0    0    0    0    0    0
  5     3   70   25   21  119   11   21    1    0    0    0    1    0    0    0    0
  6     0    0    1    2    2    2    1    0    0    0    0    0    0    0    0    0
  7     0   12    4    3    6    4   17    1    2    0    0    0    0    0    0    0
  8     0    0    1    0    1    0    0    0    0    0    0    0    0    0    0    0
  9     0    0    0    0    0    1    4    0    2    0    1    0    0    0    0    0
  10    0    0    0    0    0    0    0    0    2    0    0    0    0    0    0    0
  13    0    0    0    0    0    1    0    0    1    0    0    0    0    0    1    0
  14    0    0    0    0    0    0    0    0    0    0    0    0    0    1    0    0
> print(accuracy)
[1] 0.7369361
> print(error_xgboodeost_ml)
[1] 0.9563421
> |
```



Guests Included of DataSet actual vs predicted sorted



Guests Included of Dataset actual vs Predicted all points

That is my final solutions of these problems.

I have attached code and some graphs with this documents.

CODE FILES

1. Data_cleaning.R : All cleaning process is in this file

2. Functions.R : all functions are written in this file
3. Price.R : model on price variable
4. Guests_included.R: Model on guest included.
5. Extra_people.R: Model on extra people cost
6. Prediction.R : data cleaning of prediction data.
7. Final.R: Final prediction of data.

For more better results we can try SVM or ANN on this problem, because of time & machine constraint I was not able to do that.

We could try more parameter tuning and variable importance for better results.

Thank you so much for your time.

Regards

Aditya Modi

9742377204