Final Project

SHOPPING DATA ANALYSIS AND MODELING

Group 1 | STAT-515 | 12/10/2018
Dr. Daniel Carr

Lokesh Merupula
Naga Akhil Venkat Joginpally
Shourya Simha Addepalli

Introduction and Problem Statement

An anonymous retailer published the black Friday sales on Kaggle [1]. It contains customer profile associated with a fake user id along with products purchased by the user and purchase amount. The user profile includes categorical variable – age, gender, occupation, city type, marital status and how long the user has been staying in the current place. Purchase amount seems to be modified by applying a multiplication factor to keep the anonymity of the retailer.

In this report, we attempt to explore the data and come up with models to predict purchase amount with variable that describe customer demographics. A baseline Root Mean Square Error (RMSE) is established with simple average and compared with RMSE of Linear Regression, Lasso Regression and XGBoost.

Data contains several rows for each customer for each product he or she purchased. As product is directly influences the purchase price and is no value to the prediction model. The data has been aggregated at user id level and total mount that the user spent is taken into the models. Below is the summary of the data.

```
'data.frame':	5891 obs. of  8 variables:
 $ User_ID                   : int  1000001 1000002 1000003 1000004 
1000005 1000006 1000007 1000008 1000009 1000010 ...
 $ Gender                    : Factor w/ 2 levels "F","M": 1 2 2 2 2 1 
2 2 2 1 ...
 $ Age                       : Factor w/ 7 levels "0-17","18-25",..: 1 
7 3 5 3 6 4 3 3 4 ...
 $ Occupation                : int  10 16 15 7 20 9 1 12 17 1 ...
 $ City_Category             : Factor w/ 3 levels "A","B","C": 1 3 1 2 
1 1 2 3 3 2 ...
 $ Stay_In_Current_City_Years: Factor w/ 5 levels "0","1","2","3",..: 3 
5 4 3 2 2 2 5 1 5 ...
 $ Marital_Status            : int  0 0 0 1 1 0 1 1 0 1 ...
 $ Purchase                  : int  333481 810353 341635 205987 821001 
379450 234427 796545 593960 2169486 ...
```

Table [1]. Summary of the data that is used in models.

This below is the summary of the purchase price that will be modeled and as it can be seen it ranges from $44,108 to $10,536,783 with a mean of 851752.

```
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max. 
   44108   234914   512612   851752  1099005 10536783 
```

Table [2]. Summary statistics of the dependent variable Purchase amount in Dollars

Data Analysis

Figure [1] shows the difference from data set average for each gender and age range combination. This is generated using "plotly" library in R and pushed to the community account on plot.ly which is available at https://plot.ly/~merupula/3/#/

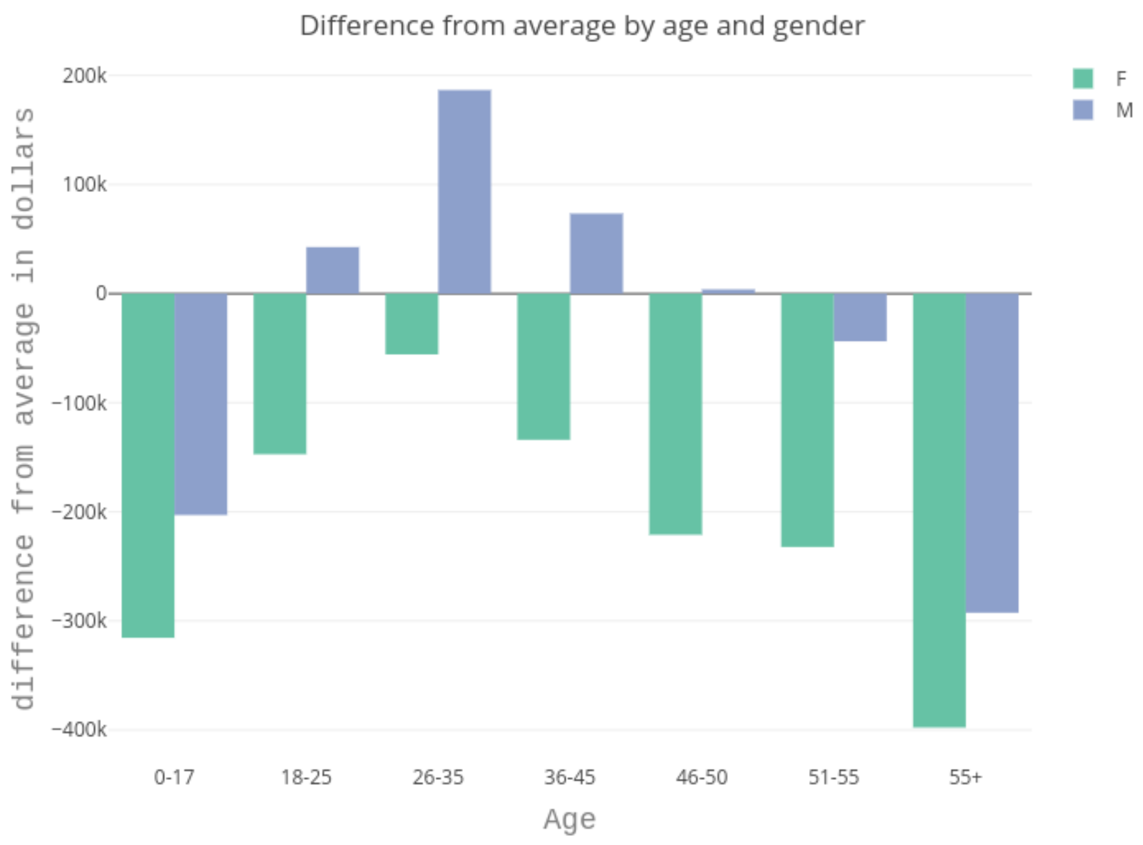


Fig [1]. Plotly graph showing difference from average for each group of gender and age.

It can be observed that male customers of age ranges 18-25, 26-35 and 36-45 spend more than average. Age group 55+ spent less than average irrespective of gender. For further analysis to observe purchase amount as factor of number of years that the customer has been staying at the current place, figure [2] with "plotly" library has been generated using R and pushed to the community account on plot.ly which is available for public access at https://plot.ly/~merupula/5/#/.

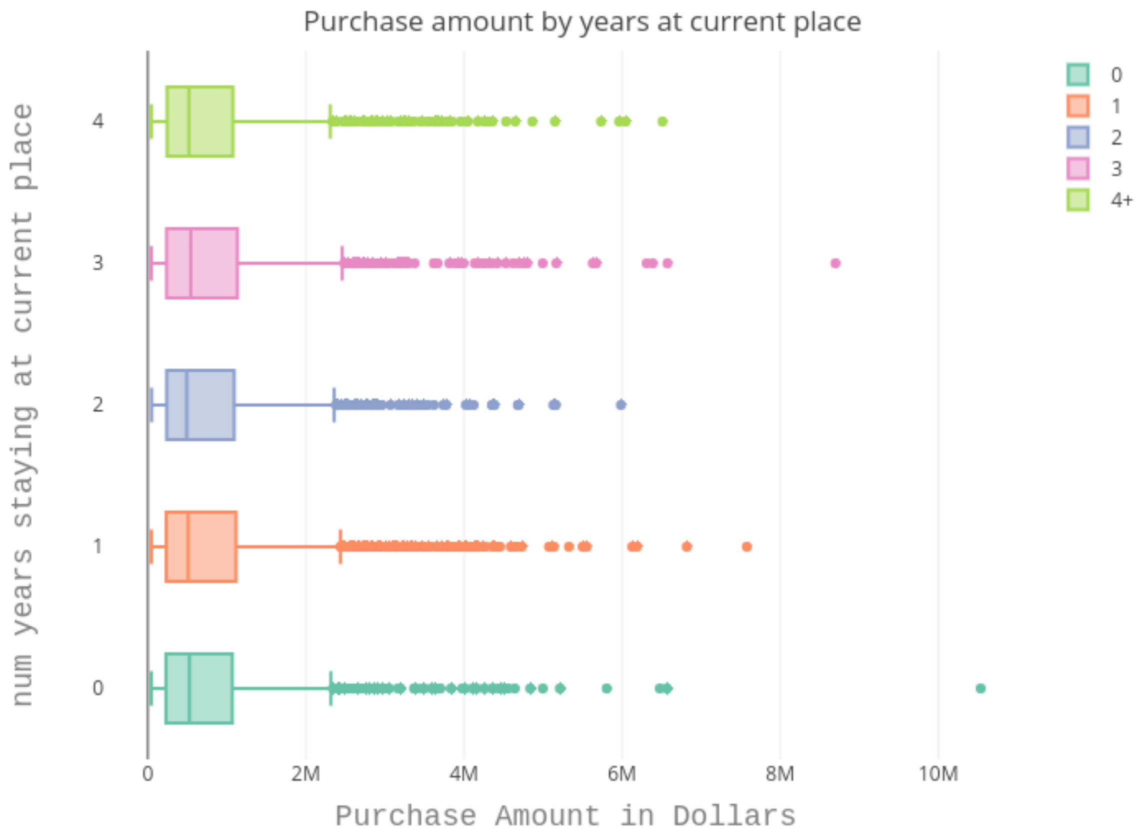


Figure [2]. Box plot with "plotly" showing purchase amount as a factor of number of years stay at current place.

It can be observed that the there is not much variation in purchase amount based on the number of year customer has been staying at the current place.

As further exploration, a series of Chi-Square tests are performed to observe what variables are most correlated to the fact that a customer spends more than average. We created a series of binary column for each customer - if a customer spent more than average, if a customer is above 35 years old and if a customer stayed at current place for more than two years. Following results establish the fact that among age, marital status, number of years stay at current place and gender variable, most correlated to purchase amount are age and gender and the other two do not have significant impact on the purchase amount.

```
data:  chisqData$Gender and chisqData$spentAboveAvg
X-squared = 60.324, df = 1, p-value = 8.047e-15
data:  chisqData$above35 and chisqData$spentAboveAvg
X-squared = 24.202, df = 1, p-value = 8.673e-07
data:  chisqData$Marital_Status and chisqData$spentAboveAvg
X-squared = 4.6906, df = 1, p-value = 0.03033
data:  chisqData$stay2plus and chisqData$spentAboveAvg
X-squared = 0.0013151, df = 1, p-value = 0.9711
```

Table [3]. Chi-Square Results

Establishing Baseline

The simplest model is to apply averages by group which is calculating average for each combination of input variables - age, gender, marital status, occupation, current stay years, city category and use this value as prediction. Calculated RMSE with this approach is 932918 which will be used as baseline to compare performance of other models.

Model 1 - Linear Regression with Cross Validation

Linear regression is a basic and most commonly used predictive technique which take a linear approach to model the relationship of the independent variable and explanatory variables in the data set. With “lm” function provided in R, linear regression has been done on the sales data set we are using and with “CVlm” function from the library “DAAG” cross validation has been done with option of 10 folds.

Figures [3], [4], [5] and [6] below show the diagnostic plots from linear regression which are Residuals vs Fitted, Normal Q-Q, Scale-Location and Residuals vs Leverage respectively.

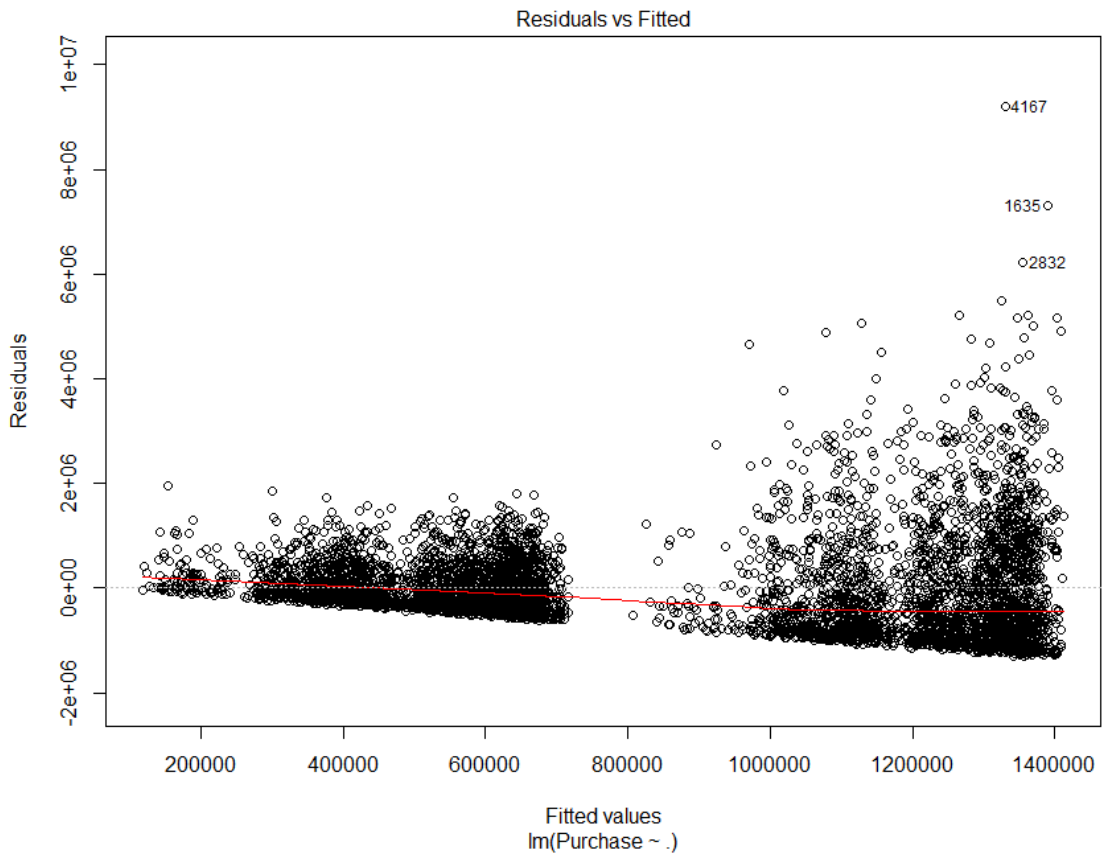


Figure [3]. Linear Regression diagnostic plot – Residuals vs Fitted

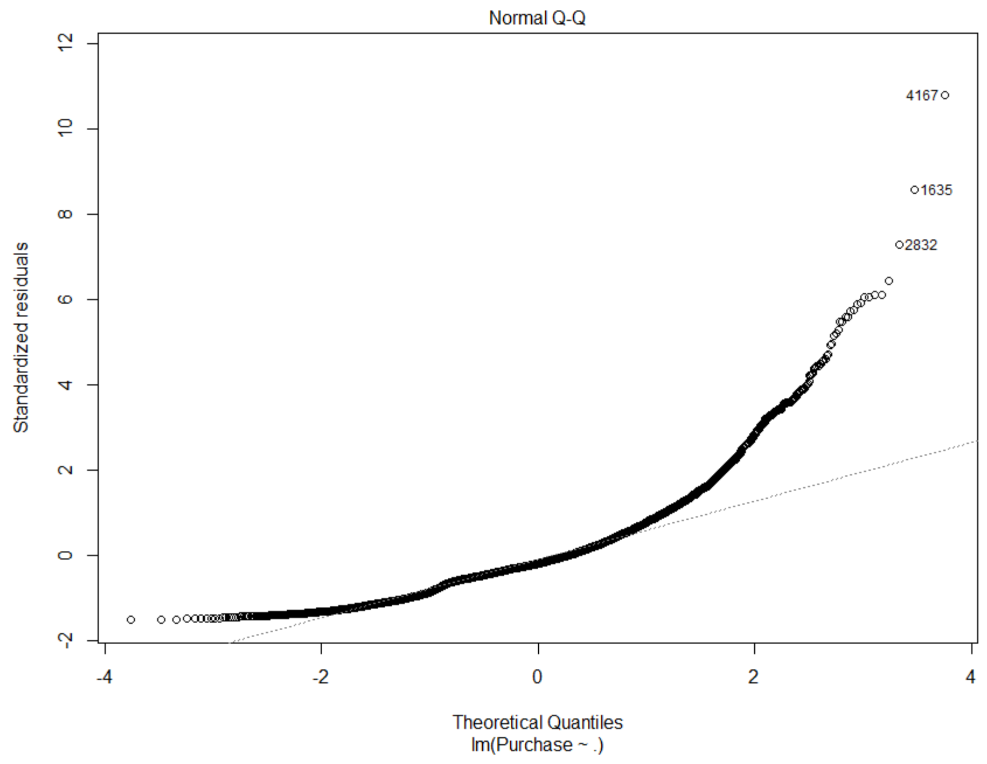


Figure [4]. Linear Regression diagnostic plot – Normal Q-Q

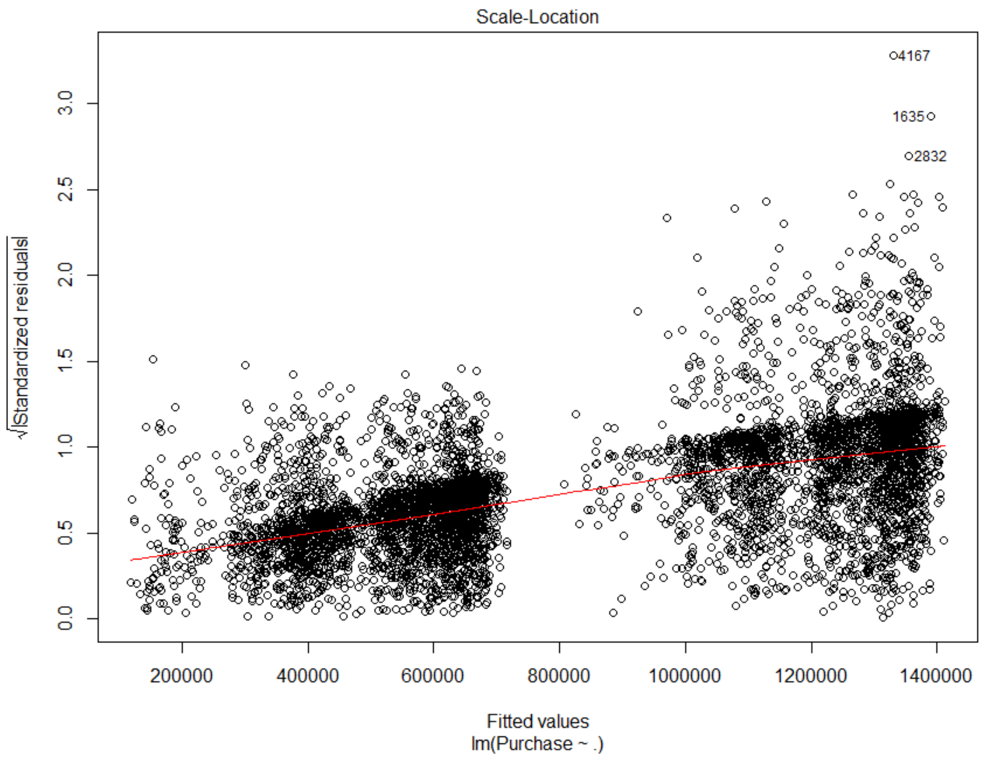


Figure [5]. Linear Regression diagnostic plot – Scale-Location

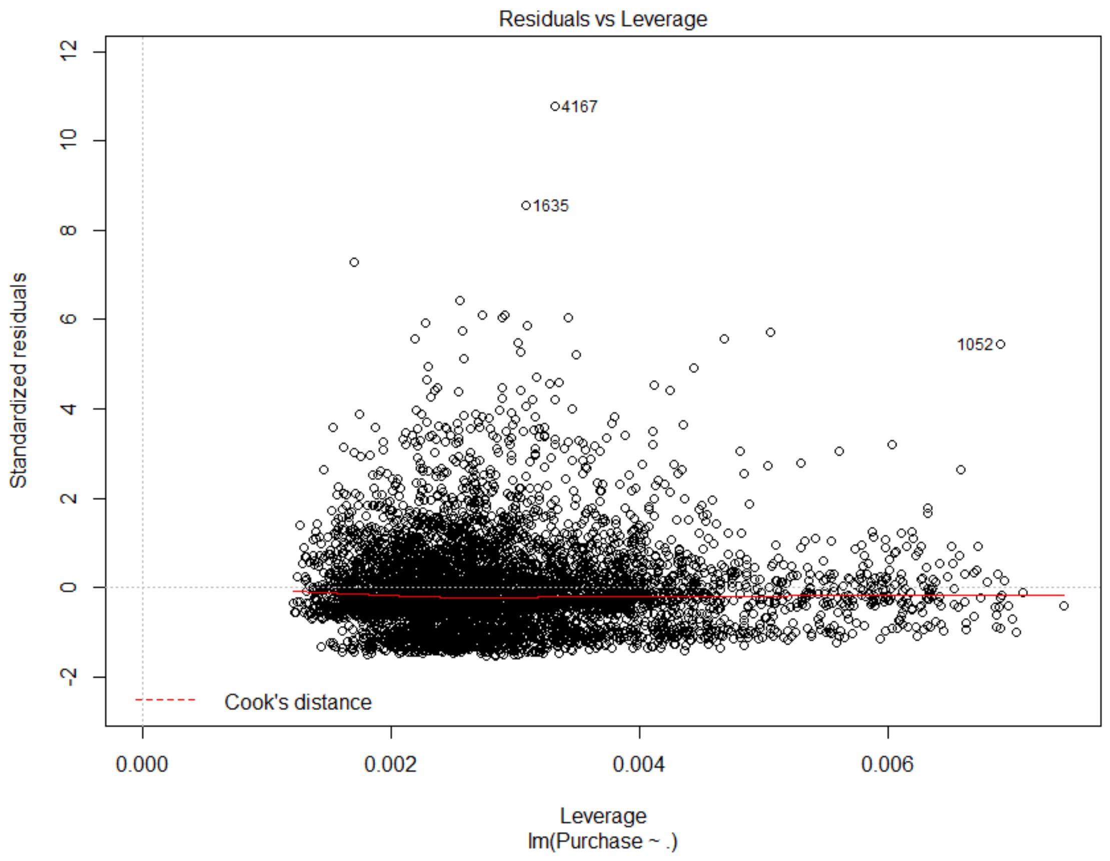


Figure [6]. Linear Regression diagnostic plot – Residuals vs Leverage

MODEL PERFORMANCE ANALYSIS

It can be observed that there is a gap in the purchase price around $700,000 which could be the pricing of products and needs further investigation. Residuals vs Fitted graph in figure [3] helps us determine if residuals have non-linear patterns and it is clear from the graph that residuals in this model do not have non-linear pattern which is a good sign for linear regression. Normal Q-Q plot from figure [4] tells us if residuals are normally distributed if they are skewed. Residuals in this model do not follow a straight line and this tells us out data is skewed. Figure [5] shows Scale-Location plot which is also called Spread-Location plot and tells us if residuals are equally spread along the ranges of predictors and have uniform variance. In our mode, residuals seem to have somewhat uniform variance till $600,000 and later it becomes non-uniform. Residuals vs Leverage from figure [6] shows that there are no observations that are heavily influencing our model.

Besides the diagnostic plots, the calculated RMSE from the residuals in linear model came out to be 854,309 which is better than the baseline model RMSE but not by much.

Below cross validation results which give predictive accuracy of a simple linear regression mode. The final resulting Mean Square (MS) from cross validation is shown below.

```
Overall (Sum over all 589 folds)
      ms
2.72e+12
```

Table [4]. Mean-Square Error from CV Linear Regression.

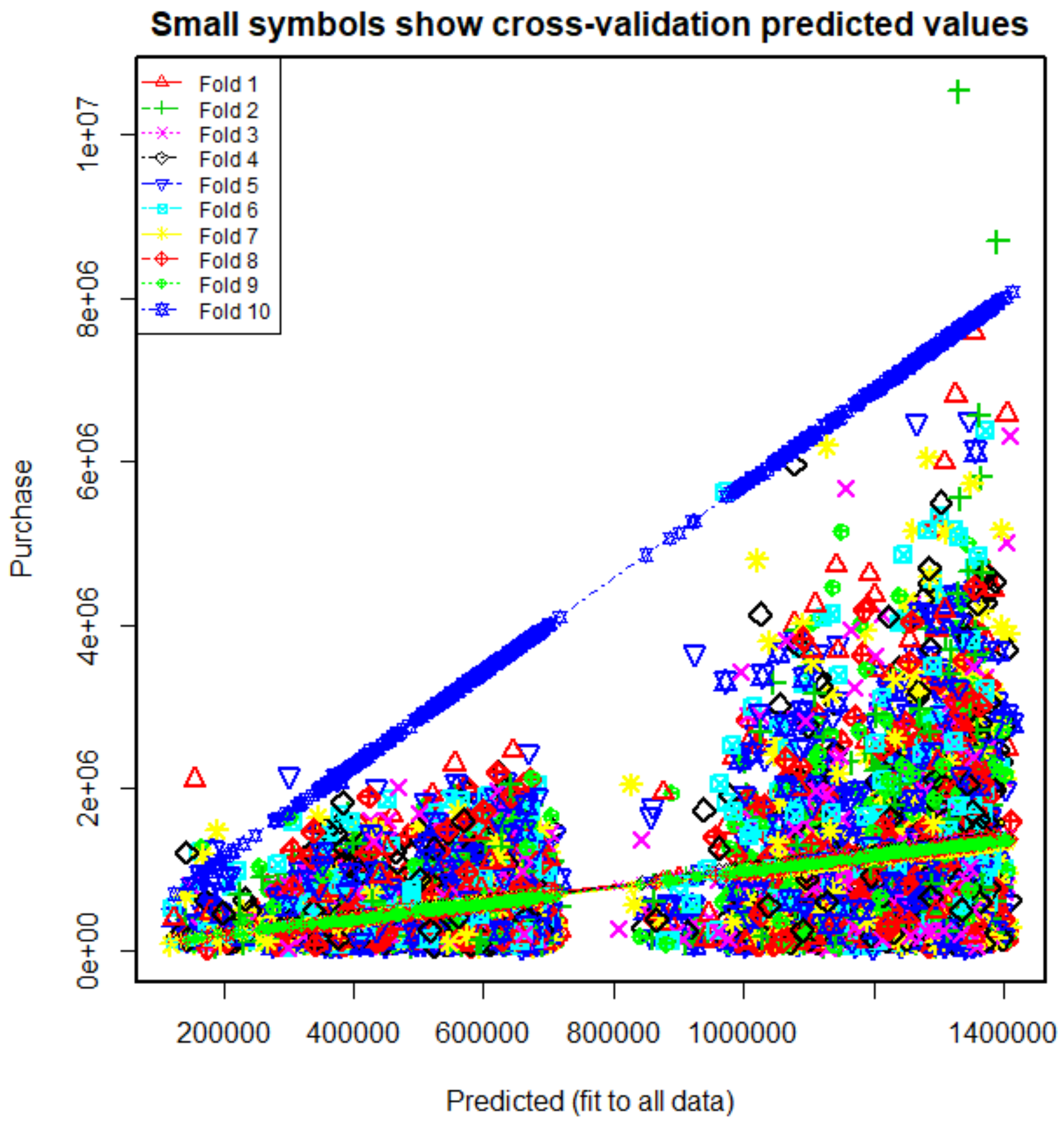


Figure [7] shows the predicted vs observed values from each fold in cross validation.

The above figure [7] shows the predictive vs observed from the cross validation and the Fold 10 has bad performance compared other Folds. Data from that fold needs to be filtered and observed for any anomalies.

Model 2- Lasso Regression

After creating model matrix with the data and creating test and train data set with reproducible random seed, “glmnet” library has been used for Lasso Regression. Minimum lambda identified by the model has been used to predict and calculate RMSE. Below figures [8] and [9] show L1 Norm vs Coefficients and Log-Lambda vs Mean Squared Error (MSE) for the model.

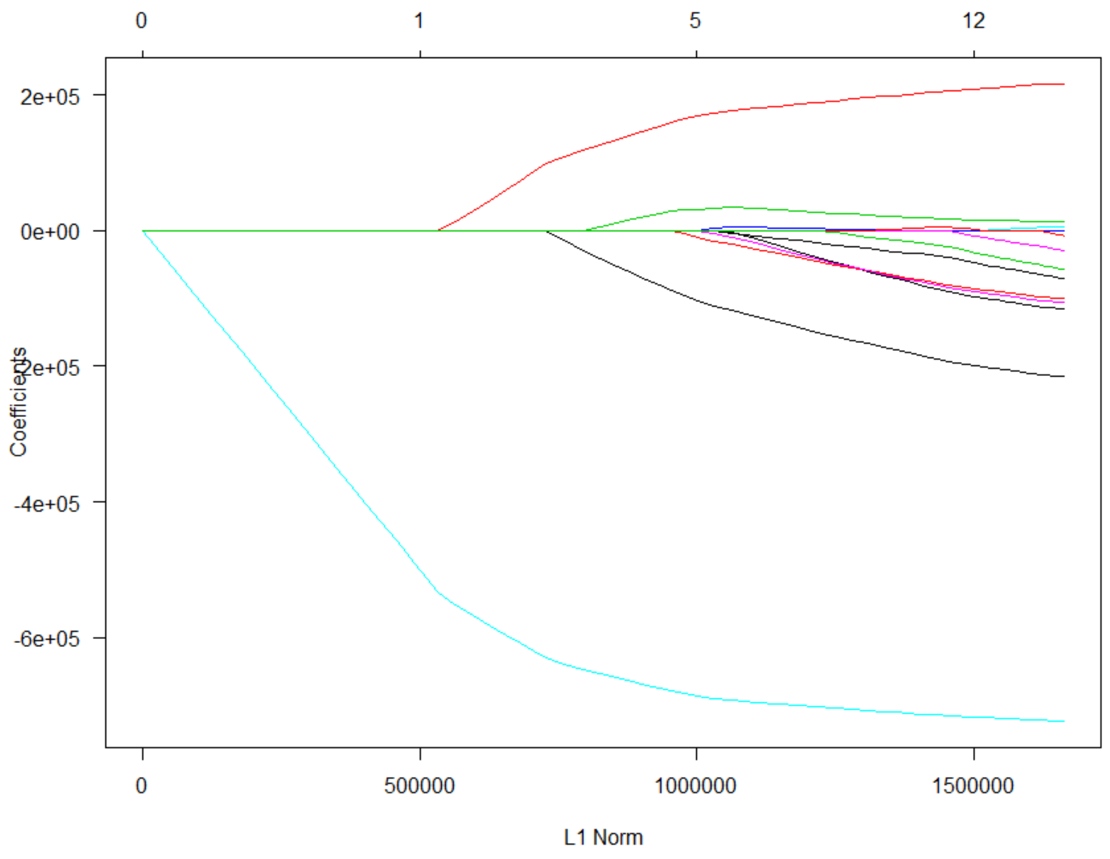


Figure [8]. L1 Norm vs Coefficients

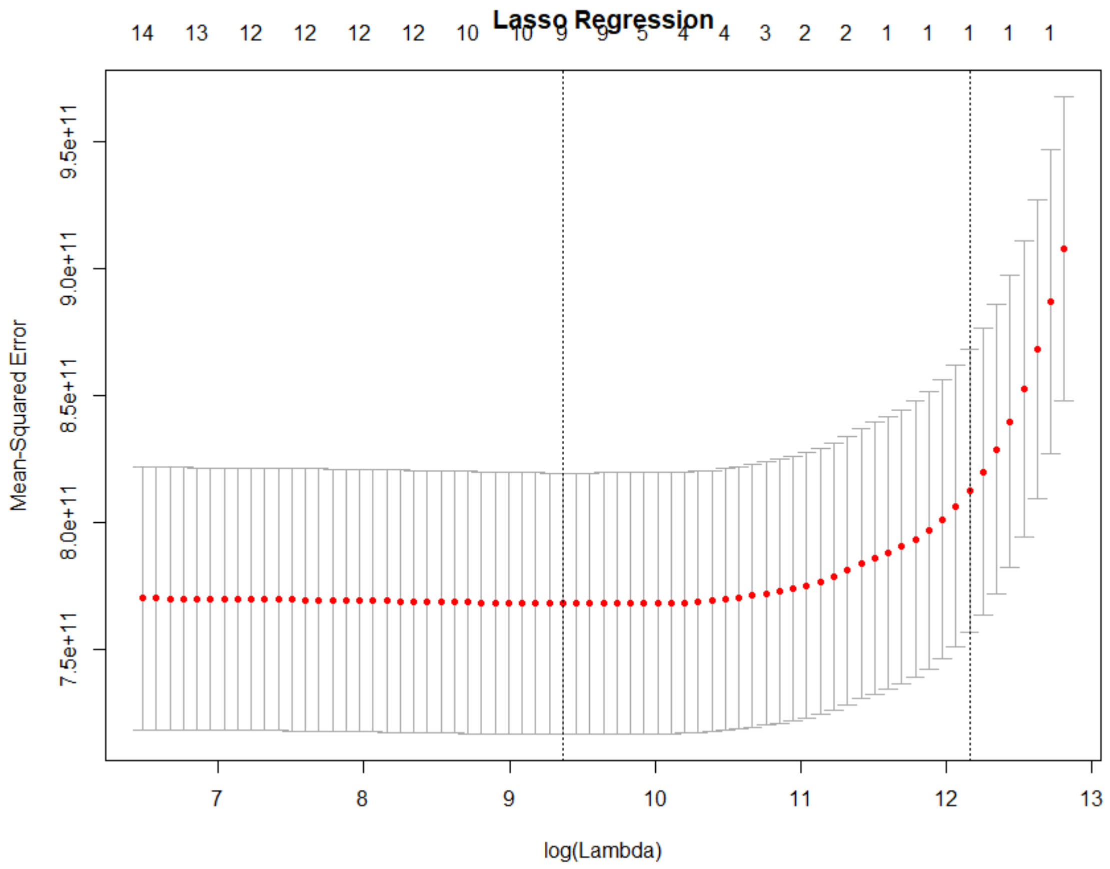
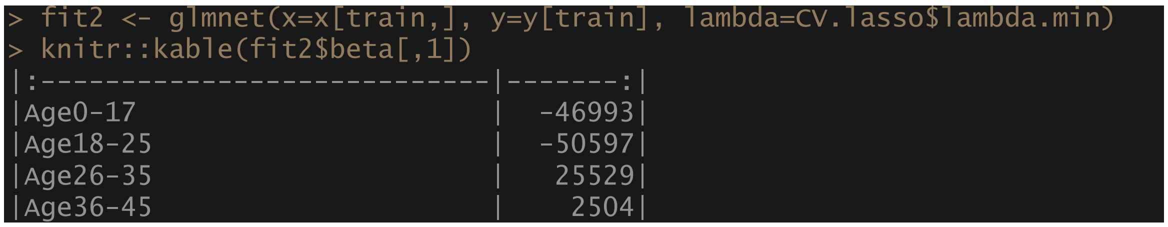


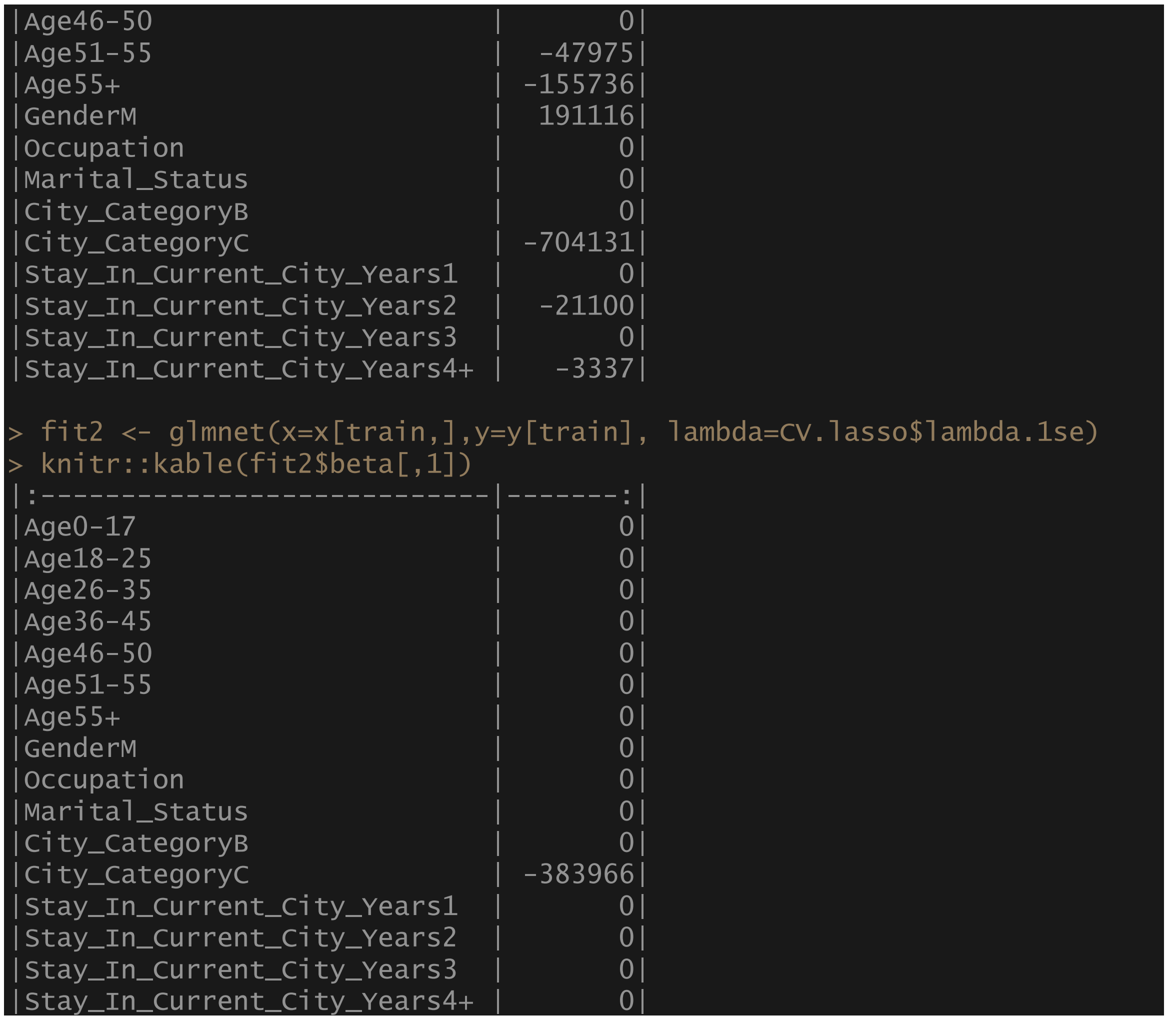
Figure [9]. Log (Lambda) vs Mean Squared Error of Lasso Regression.

At around 1,000,000 L1-Norm, we are left with 4 non-zero coefficients and finally only one coefficient has non-zero below 500,000 L1 Norm. Log Lambda vs MSE plot shows that minimum MSE can be achieved with 9 variables and MSE within 1 standard deviation can be with just one variable.

In order to see what these coefficients are, coefficients with "glmnet" fit with minimum lambda and 1 standard error lambda are printed. As seen below from the beta 1se, City Category C has the highest influence on the model and that variable alone can be used with 1 standard error from MSE. The variables that provide minimum MSE are Age, Gender, City Category and with very little influence, years stayed in current city. Occupation and Marital status have zero influence which is consistent with our observation earlier from Chi-Square tests on spending above average.



```
> fit2 <- glmnet(x=x[train,], y=y[train], lambda=CV.lasso$lambda.min)
> knitr::kable(fit2$beta[,1])
|:-----------------------------|-------:|
|Age0-17                       |  -46993|
|Age18-25                      |  -50597|
|Age26-35                      |   25529|
|Age36-45                      |    2504|
```



```
|Age46-50                    |       0|
|Age51-55                    |  -47975|
|Age55+                      | -155736|
|GenderM                     |  191116|
|Occupation                  |       0|
|Marital_Status              |       0|
|City_CategoryB              |       0|
|City_CategoryC              | -704131|
|Stay_In_Current_City_Years1 |       0|
|Stay_In_Current_City_Years2 |  -21100|
|Stay_In_Current_City_Years3 |       0|
|Stay_In_Current_City_Years4+ |   -3337|

> fit2 <- glmnet(x=x[train,],y=y[train], lambda=CV.lasso$lambda.1se)
> knitr::kable(fit2$beta[,1])
|:---------------------------|-------:|
|Age0-17                     |       0|
|Age18-25                    |       0|
|Age26-35                    |       0|
|Age36-45                    |       0|
|Age46-50                    |       0|
|Age51-55                    |       0|
|Age55+                      |       0|
|GenderM                     |       0|
|Occupation                  |       0|
|Marital_Status              |       0|
|City_CategoryB              |       0|
|City_CategoryC              | -383966|
|Stay_In_Current_City_Years1 |       0|
|Stay_In_Current_City_Years2 |       0|
|Stay_In_Current_City_Years3 |       0|
|Stay_In_Current_City_Years4+ |       0|
```

Table [5]. Coefficients from Lasso Regression Fit

RMSE calculate from prediction from Lasso Regression is 837,459 which clearly shows improvement compared to Linear Regression Model but still does not improve by much.

Model 3 – Extreme Gradient Boost (XGBoost)

"XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable. It implements machine learning algorithms under the Gradient Boosting framework. XGBoost provides a parallel tree boosting (also known as GBDT, GBM) that solve many data science problems in a fast and accurate way" [2].

XGBoost requires numerical and all the character categorical variables are converted to numerical values using R package "caret" stands for Classification and Regression Training [3]. This process is called Hot One Encoding. Training and testing data sets are generated with reproducible random seed and fed to the "trainControl". Below code snippet shows the tuneGrid that was used for XGBoost.

Eta specifies the step size shrinkage to prevent overfitting, Gamma specifies the minimum loss reduction to make a further partition of a leaf node of a tree, min_child_weight minimum sum of weights that a child needs and subsample 0.5 means randomly collects half of data instances to grow tree randomly and this value prevent overfitting [4].

```
xgbGrid <- expand.grid(nrounds = c(100,200),
                  max_depth = c(10, 15, 20, 25),
                  colsample_bytree = seq(0.5, 0.9, length.out = 5),
                  eta = 0.1,
                  gamma=0,
                  min_child_weight = 1,
                  subsample = 1
                 )
```

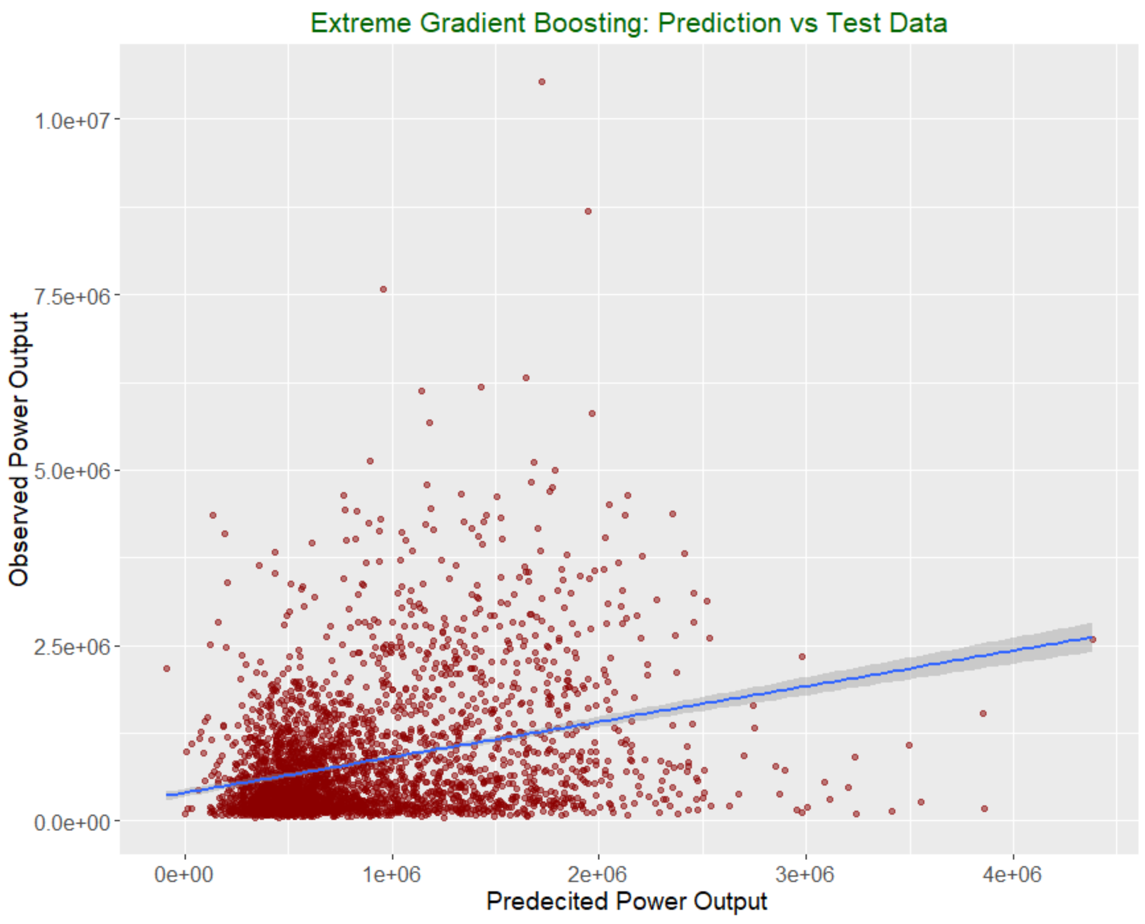


Figure [10]. XGBoost Predicted vs Observed with lm geom_smooth

After prediction, the RMSE calculates is 921643 which is worse than both Linear and Lasso Regression Models. This can be improved by hyperparameter tuning in XGBoost training.

TUNING HYPERPARAMETERS

With changing the parameters below, new RMSE shows improvement from the first iteration but not better than Linear or Lasso Regressions. The new value of RMSE is 857935 and below are the parameters used. This reduction in RMSE comes at the expense of longer running training and more CPU and memory utilization.

```
xgbGrid <- expand.grid(nrounds = c(100,500),  # this is n_estimators in the python code
above

               max_depth = c(5, 10, 15, 20, 25),

               colsample_bytree = seq(0.1, 0.9, length.out = 5),

               ## The values below are default values in the sklearn-api.

               eta = 0.3,

               gamma=0,

               min_child_weight = 1,

               subsample = 1

              )
```

For further improved, one more iteration is done making maximum depth go up to 40 - c(5, 10, 15, 20, 25, 30, 35, 40). This made the training run for extra time and the RMSE came down to 852879 which is better than Linear Regression and still Lasso Regression had better RMSE. Below figure [11] the predicted vs observed for this iteration and it can be notices that the smoothing lm line is little better slope and data is spread around the line better than the first iteration.

NEXT STEPS

In order to achieve optimal performance with XGBoost, hyperparameter tuning is important. Using "makeParamSet" function in package "mlr", we can make a resampling plan and retrain the model with random matrix option. This takes time and resources.

Another option to explore is to enable GPU for XGBoost training. The training and prediction with XGBoost can be accelerated with CUDA-capable GPUs [5].

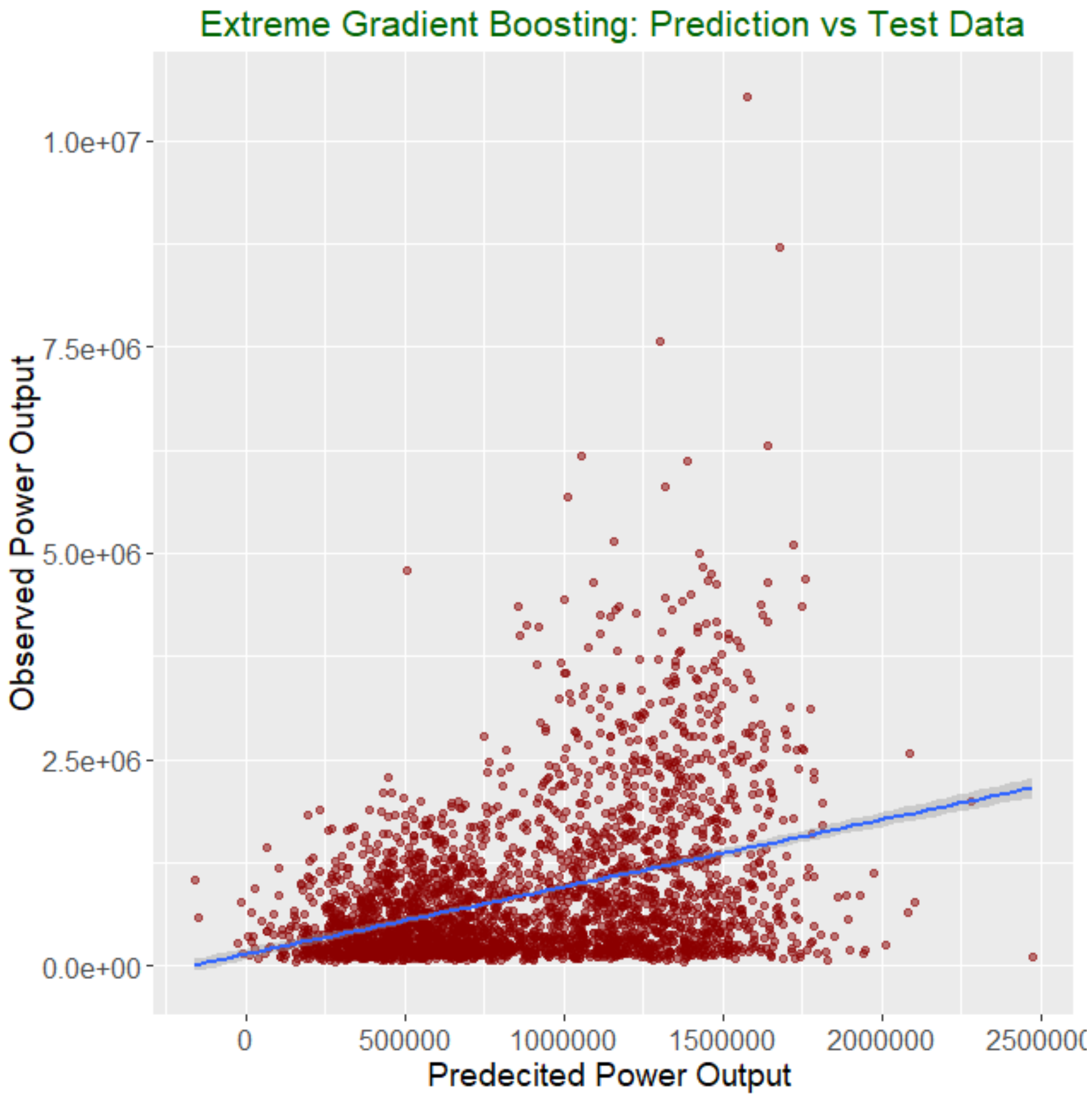


Figure [11]. Hyperparameter tuned XGBoost Predicted Vs Observed.

Hardware and Software

Models were run on Windows Laptop with Intel® Core™ i7-8650U CPU @ 2.11 GHz and 16 GB RAM with 64-bit operating system and x64-bsed processor.

R Studio Version 1.1.456 with R Version R version 3.5.1 (2018-07-02) was used.

R libraries used for this report - “plotly”, “dplyr”, “lmvar”, “DAAG”, “glmnet”, “tidyverse”, “caret” and xgboost”.

References

[1] Kaggle.com. (2018). *Black Friday*. [online] Available at: https://www.kaggle.com/mehdidag/black-friday [Accessed 8 Dec. 2018]..

[2]. Xgboost.readthedocs.io. (2018). XGBoost Documentation — xgboost 0.81 documentation. [online] Available at: https://xgboost.readthedocs.io/en/latest/ [Accessed 8 Dec. 2018].

[3]. Kuhn, M. (2018). The caret Package. [online] Topepo.github.io. Available at: http://topepo.github.io/caret/index.html [Accessed 8 Dec. 2018].

[4]. Learning, M. and steps, H. (2018). *How to use XGBoost algorithm in R in easy steps*. [online] Analytics Vidhya. Available at: https://www.analyticsvidhya.com/blog/2016/01/xgboost-algorithm-easy-steps/ [Accessed 8 Dec. 2018].

[5]. Xgboost.readthedocs.io. (2018). XGBoost GPU Support — xgboost 0.81 documentation. [online] Available at: https://xgboost.readthedocs.io/en/latest/gpu/ [Accessed 8 Dec. 2018].

Appendix

Data Dictionary

User_ID: User – Primary Key

Product_ID

Gender

Age: Age in bins

Occupation

City_Category

Stay_In_Current_City_Years

Marital_Status

Product_Category_1

Product_Category_2

Product_Category_3

Purchase: Purchase amount in dollars

Code

```
salesDataRawTemp <- read.csv('BlackFriday.csv')

#summary(salesDataRaw)

salesDataRaw <- salesDataRawTemp %>%

  group_by(User_ID, Gender, Age, Occupation, City_Category,
Stay_In_Current_City_Years, Marital_Status) %>%
summarise(Purchase=sum(Purchase)) %>%

  ungroup %>% as.data.frame()

summary(salesDataRaw$Purchase,type='Box')


Sys.setenv("plotly_username"="your_user_id")

Sys.setenv("plotly_api_key"="yoUrAPIkey")


normSpent <-  mutate(salesDataRaw,

                spentNorm = Purchase - mean(Purchase))

byAgeGen <- normSpent %>%

  group_by(Age, Gender) %>%

  summarise(avgSpent = mean(spentNorm))

f <- list(

  family = "Courier New, monospace",

  size = 18,

  color = "#7f7f7f"

)

x <- list(
  title = "Age",
  titlefont = f
)
y <- list(
  title = "difference from average purchase amount in dollars",
  titlefont = f
)
q <- plot_ly(data = byAgeGen,
    x = ~Age,
    y = ~avgSpent,
    color= ~Gender,
    #colors = 'Reds',
    type = 'bar',
    legendgroup = "A") %>%
  layout(
    title = "Difference from average by age and gender",
    xaxis = x,
    yaxis = y)

q
api_create(q, filename = "blackFridayPurchaseDevFromAvgByGendAge", sharing='public')

f <- list(
  family = "Courier New, monospace",
  size = 18,
  color = "#7f7f7f"
)
```

```
x <- list(
  title = "Purchase Amount in Dollars",
  titlefont = f
)
y <- list(
  title = "num years staying at current place",
  titlefont = f
)
p2 <- plot_ly(data=salesDataRaw, x = ~Purchase, color = ~Stay_In_Current_City_Years,
type = "box") %>%
  layout(
    title = "Purchase amount by years at current place",
    xaxis = x,
    yaxis = y)
p2
api_create(p2, filename = "blackFridayPurchaseNumYearsAtPlace", sharing='public')
chisqData <-  mutate(salesDataRaw,
                spentAboveAvg = ifelse(Purchase > mean(Purchase), 1, 0),
                above35 = ifelse((Age == '0-17' | Age == '18-25' | Age == '26-35'), 0, 1),
                stay2plus = ifelse((Stay_In_Current_City_Years == '3' |
Stay_In_Current_City_Years == '4+'), 1, 0))
table(chisqData$Gender, chisqData$spentAboveAvg)


chisq.test(chisqData$Gender, chisqData$spentAboveAvg, correct=FALSE)
chisq.test(chisqData$above35, chisqData$spentAboveAvg, correct=FALSE)
chisq.test(chisqData$Marital_Status, chisqData$spentAboveAvg, correct=FALSE)
chisq.test(chisqData$stay2plus, chisqData$spentAboveAvg, correct=FALSE)
```

```
grpByInput <- salesDataRaw %>%

  group_by('Age', 'Gender', 'Occupation', 'Stay_In_Current_City_Years', 'Marital_Status',
'City_Category') %>%

  summarize(meanPurch = mean(Purchase, na.rm = TRUE)) %>%

  ungroup %>% as.data.frame()

model1Dat <- merge(salesDataRaw, grpByInput)

residuals = model1Dat$Purchase - model1Dat$meanPurch

RMSE = sqrt(mean(residuals^2))

cat('The root mean square error of the test data is ', round(RMSE,3),'\n')


lmPur <- lm(data=salesDataRaw, Purchase~. , x = TRUE, y = TRUE)

plot(lmPur)

cv.lm( lmPur, m=5, dots = FALSE, seed=29, plotit=TRUE, printit=TRUE)

library(DAAG)

CVlm(data =salesDataRaw, m=10, form.lm = formula(Purchase ~  . ))

hist(predictedresiduals)

RSS <- c(crossprod(lmPur$residuals))

MSE <- RSS / length(lmPur$residuals)

RMSE <- sqrt(MSE)

RMSE

cat('The root mean square error of the test data is ', round(RMSE,3),'\n')


x <- model.matrix(~ Age + Gender + Occupation + Marital_Status + City_Category +
Stay_In_Current_City_Years -1, dat=salesDataRaw)

y <- salesDataRaw[, 'Purchase']

nr <- nrow(x)

trainingSize <- ceiling(nr/2) # half case for training

set.seed(37)
```

```
train <- sample(1:nr,trainingSize)

x.train <- x[train,]

y.train <- y[train]

head(x.train)

head(y.train)

x.test <- x[-train,]

y.test <- y[-train]

head(x.test)

library(glmnet)

lasso.mod <- glmnet(x[train,],y[train], alpha=1)

plot(lasso.mod,las=1)

plot(lasso.mod, "norm",   label=TRUE)

plot(lasso.mod, "lambda", label=TRUE)

set.seed(1)

CV.lasso=cv.glmnet(x[train,],y[train],type.measure='mse',alpha=1)

plot(CV.lasso,main="Lasso Regression")

CV.lasso$lambda.min

fit2 <- glmnet(x=x[train,],y=y[train], lambda=CV.lasso$lambda.1se)

#print(fit2)

fit2$beta[,1]

lambda.lasso = CV.lasso$lambda.min

lasso.pred = predict(CV.lasso, s = lambda.lasso, newx = x.test)

errL = y.test - lasso.pred

mse=mean(errL^2)

rmse=sqrt(mse)

rmse

library(tidyverse)
library(xgboost)

library(caret)


unique(salesDataRaw$Age)
unique(salesDataRaw$Gender)
unique(salesDataRaw$Occupation)#no need to hot
unique(salesDataRaw$Stay_In_Current_City_Years)
unique(salesDataRaw$Marital_Status) #not needed
unique(salesDataRaw$City_Category)
max(salesDataRaw$Purchase)

#data = as.data.frame(data)
ohe_feats = c('Age', 'Gender', 'Occupation', 'Stay_In_Current_City_Years', 'Marital_Status',
'City_Category')
dummies = dummyVars(~ Age + Gender + Occupation + Stay_In_Current_City_Years +
Marital_Status + City_Category, data = salesDataRaw)
df_all_ohe <- as.data.frame(predict(dummies, newdata = salesDataRaw))
df_all_combined <- cbind(salesDataRaw[,-c(which(colnames(salesDataRaw) %in%
ohe_feats))],df_all_ohe)

salesDataRawXGB <- df_all_combined


str(salesDataRawXGB)

set.seed(100)  # For reproducibility
```

```
# Create index for testing and training data

inTrain <- createDataPartition(y = salesDataRawXGB$Purchase, p = 0.5, list = FALSE)

# subset power_plant data to training

training <- salesDataRawXGB[inTrain,]

# subset the rest to test

testing <- salesDataRawXGB[-inTrain,]


X_train = xgb.DMatrix(as.matrix(training %>% select(-Purchase)))

y_train = training$Purchase

X_test = xgb.DMatrix(as.matrix(testing %>% select(-Purchase)))

y_test = testing$Purchase


xgb_trcontrol = trainControl(

  method = "cv",

  number = 5,

  allowParallel = TRUE,

  verboseIter = FALSE,

  returnData = FALSE

)


xgbGrid <- expand.grid(nrounds = c(100,500),  # this is n_estimators in the python code
above

                       max_depth = c(5, 10, 15, 20, 25),

                       colsample_bytree = seq(0.1, 0.9, length.out = 5),

                       ## The values below are default values in the sklearn-api.

                       eta = 0.3,

                       gamma=0,

                       min_child_weight = 1,
                     subsample = 1
                    )
set.seed(0)
xgb_model = train(
  X_train, y_train,
  trControl = xgb_trcontrol,
  tuneGrid = xgbGrid,
  method = "xgbTree"
)


xgb_train$bestTune
predicted = predict(xgb_model, X_test)
residuals = y_test - predicted
RMSE = sqrt(mean(residuals^2))
cat('The root mean square error of the test data is ', round(RMSE,3),'\n')


options(repr.plot.width=8, repr.plot.height=4)
my_data = as.data.frame(cbind(predicted = predicted,
                            observed = y_test))
# Plot predictions vs test data
ggplot(my_data,aes(predicted, observed)) + geom_point(color = "darkred", alpha = 0.5) +
    geom_smooth(method=lm)+ ggtitle('Linear Regression ') + ggtitle("Extreme Gradient Boosting: Prediction vs Test Data") +
      xlab("Predecited Power Output ") + ylab("Observed Power Output") +
        theme(plot.title = element_text(color="darkgreen",size=16,hjust = 0.5),
         axis.text.y = element_text(size=12), axis.text.x = element_text(size=12,hjust=.5),
         axis.title.x = element_text(size=14), axis.title.y = element_text(size=14))
```

```
model <- xgb.dump(xgb_model, with.stats = T)
```