



PRUDENTIAL DATATHON (Group 8)

-JIAMIN WANG

-HANAN ALSALAMAH

-NAUKA SALOT


-ABHINAV TIWARI



OBJECTIVE


- ▶ To analyze the Prudential Life Insurance dataset and predict the data points in the existing evaluation and sanction the outcomes to significantly amalgamate the process by building the Linear Regression, Decision Tree Regression, SVM Regression and XGBoost linear booster models to conclude the results.

APPROACH

- ▶ Built a predictive model for Prudential Life Insurance by evaluating the risk factor associated with various parameters in an application to accurately classify risk on a scale of 1-8 (8 being the highest) using a more automated approach.
- 



DATA PREPARATION

- ▶ The data contains 110 categorical variables, 13 continuous variables and 5 discrete variables.
 - ▶ Categorical variables contain a finite number of categories. Such as (Product_Info_2)
 - ▶ Discrete variables are numeric variables that have a countable number of values between two values. Such as (Medical_History_10)
 - ▶ Continuous variables are numeric variables that have an infinite number of values between two values. Such as (BMI)
- 

DATA PREPARATION

- The first step was to check for any noise in the data, but after analyzing the dataset was found to be noise-free.
- The second step was to find out the percentage of null values in each column and remove the columns having more than or equal to 70% null data and also remove the id column.

```
# Finding the the percentage of null values in each column
colMeans(is.na(Prudential_Data)*100)
#Removing the columns that have more than or equal 70% of null values;
#and removing unnecessary id column
Prudential_Data <- subset(Prudential_Data, select = -c(Id,Medical_History_15,
Medical_History_24 ,Medical_History_32, Medical_History_10,Family_Hist_5) )|
```

TREATING THE NULL VALUES

- Replaced the null values in each column using the mean of that column.

```
# Replacing the null values in each column with the computed mean of that column
Prudential_Data$Family_Hist_2 <- ifelse(is.na(Prudential_Data$Family_Hist_2),
ave(Prudential_Data$Family_Hist_2, FUN=function(x) mean(x, na.rm = TRUE)),
Prudential_Data$Family_Hist_2)
```

- Checked the sum of NA variables of a specific column to be zero after filling in the mean value.

```
#The sum of NA values for a specific column should be zero
#after filling null values with the computed mean value
sum(is.na(Prudential_Data$Family_Hist_2)) # sum is zero as no more null values
```

- Settled the categorical columns to as a factor to get factors with n levels.

```
#Setting categorical columns to as a factor to get factors with n levels
Prudential_Data$Product_Info_1 <- as.factor(Prudential_Data$Product_Info_1)
Prudential_Data$Product_Info_2 <- as.factor(Prudential_Data$Product_Info_2)
```

DATA TRANSFORMATION

- Creating a function to convert each categorical column to its numeric representation

```
# Creating a function to convert each level of every categorical column to a separate column
convert.fun <- function(Prudential_Data, Attribute){
  for(level in unique(Prudential_Data[[Attribute]])){
    Prudential_Data[paste(Attribute,seq = "_",level)]<- ifelse(Prudential_Data[[Attribute]] == level,1,0)
  }
  return(subset(Prudential_Data,select = -get(Attribute)))
}
```

- Calling the function for Categorical Columns

```
# Calling the function for all categorical columns; If the categorical column has a lot of values,
#then we deal with it as a discrete column (e.g. Product_Info_3, Employment_Info_2, Medical_History_2)
Prudential_Data <- convert.fun(Prudential_Data, "Product_Info_1")
Prudential_Data <- convert.fun(Prudential_Data, "Product_Info_2")
Prudential_Data <- convert.fun(Prudential_Data, "Product_Info_5")
Prudential_Data <- convert.fun(Prudential_Data, "Product_Info_6")
```

- Fitting the linear regression model and taking the significant variables only

```
#Fitting the linear regression model
fit <-lm(Cleaned_Data$Response ~. ,Cleaned_Data)
summary(fit)
#Assuming that the desired significance is 0.05,
#we take only variables/columns with p-value less than or equal 0.05 (significance level)
Final_variables <- data.frame(summary(fit)$coef[summary(fit)$coef[,4] <= .05, 4])
# writing the variables/columns names in a new file
write.csv(Final_variables, file = "Final_variables.csv", row.names = TRUE)
```


LINEAR REGRESSION

- It is used for modeling the relationship between a dependent variable Y and one or more independent variables denoted as X.

PERFORMING LINEAR REGRESSION

- Used the cleaned training data to construct the linear model using `lm()`

```
linear_model_2 <- lm(Response ~., data = Prudential_train_2)
```

```
on 47476 degrees of freedom  
Adjusted R-squared: 0.1738  
176 DF, p-value: < 2.2e-16
```

```
> error_2  
[1] 2.665275
```

- From the summary of the model, removed the columns with less significant p values and redesigned the model focusing on the R-Squared and RMSE values.

```
linear_model_1 <- lm(Response ~ Product_Info_4+Ins_Age+Ht+Wt+Family_Hist_2+Family_Hist_4+  
Medical_History_1+Medical_Keyword_3+Medical_Keyword_9+Medical_Keyword_11+  
Medical_Keyword_12+Medical_Keyword_15+Medical_Keyword_16+Medical_Keyword_18+  
Medical_Keyword_19+Medical_Keyword_25+Medical_Keyword_31+Medical_Keyword_33+  
Medical_Keyword_34+Medical_Keyword_37+Medical_Keyword_38+Product_Info_2 _ A1`+  
`Product_Info_2 _ E1`+`Product_Info_2 _ D4`+`Product_Info_2 _ A7`+  
`Product_Info_2 _ A6`+`Product_Info_2 _ A5`+`Product_Info_2 _ B2`+  
`Product_Info_2 _ A4`+`Product_Info_6 _ 1`+`Employment_Info_3 _ 1`+  
`Employment_Info_5 _ 3`+`InsuredInfo_2 _ 2`+`InsuredInfo_5 _ 1`+  
`InsuredInfo_6 _ 2`+`InsuredInfo_7 _ 1`+`Insurance_History_3 _ 1`+  
`Insurance_History_7 _ 3`+`Medical_History_4 _ 1`+`Medical_History_7 _ 1`+  
`Medical_History_11 _ 3`+`Medical_History_22 _ 2`+`Medical_History_35 _ 1`+  
`Medical_History_38 _ 1`+`Medical_History_39 _ 3`|data= Prudential_train_2)
```

```
on 47439 degrees of freedom  
Adjusted R-squared: 0.3139  
439 DF, p-value: < 2.2e-16
```

```
> rmse(Prudential_train_2$Response, pred)  
[1] 2.823731
```

LASSO REGRESSION

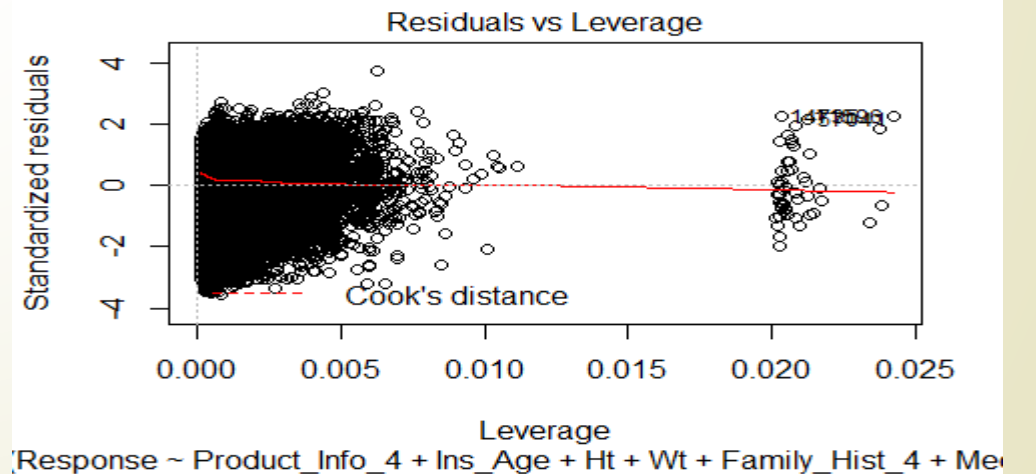
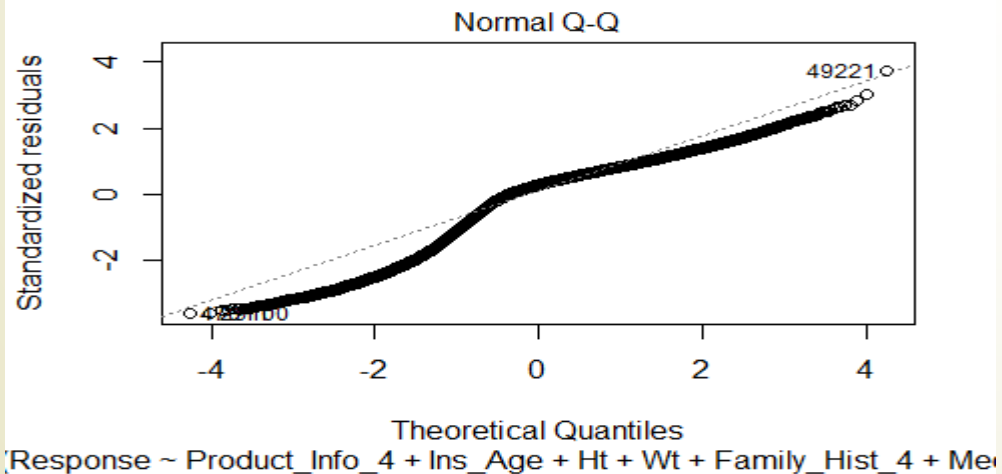
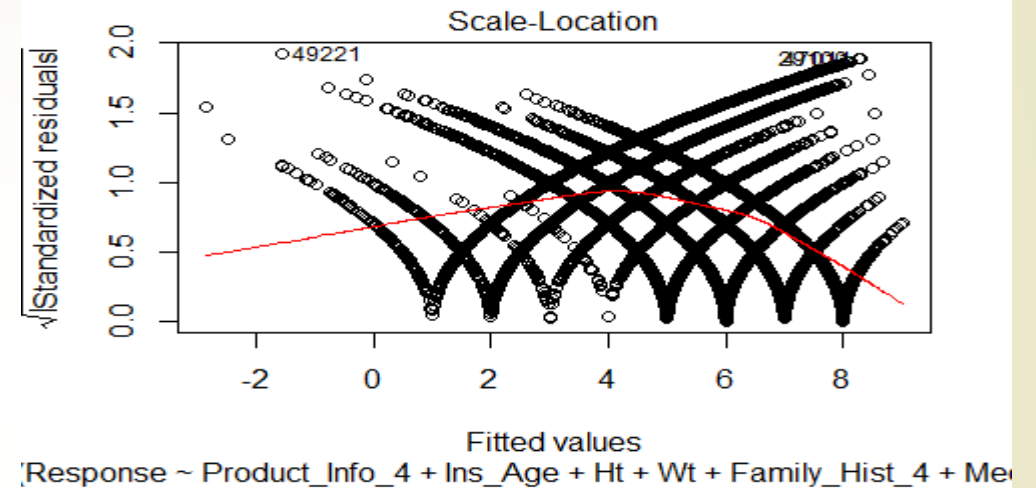
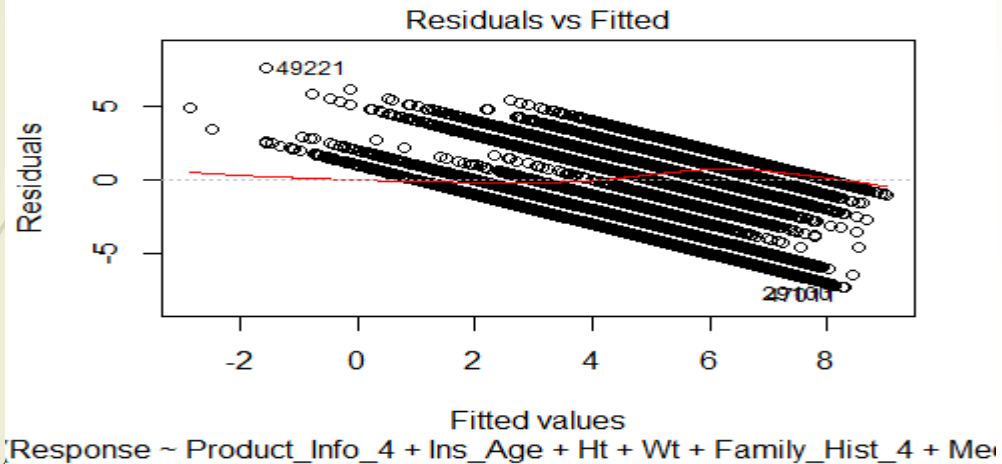
- Applied Lasso Regression, to decrease the RMSE of the model.

```
x<- model.matrix(Response ~ ., Prudential_train_2)[,-65]
y<-Prudential_train_2$Response
grid = 10^seq(10,-2,length=100)
lasso.mode = glmnet(x,y,alpha = 1, lambda = grid)
summary(lasso.mode)
coefficient_lasso<-coef(lasso.mode)
```

- The final value of RMSE is

```
> error_3
[1] 2.336253
```


DATA VIZUALIZATION

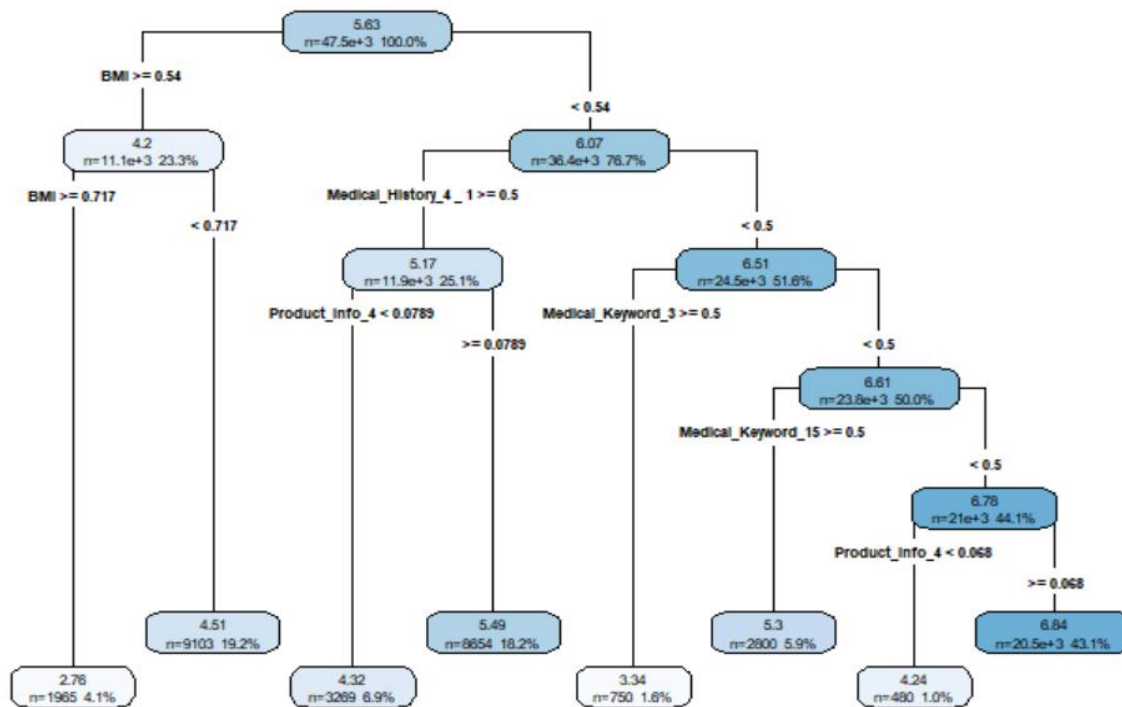


Decision Tree Regression

- Decision Tree is a supervised learning algorithm.
- It uses the ID3 top-down search algorithm to select and split the attributes and build the tree shaped diagram.
- The tree has decision nodes and leaf nodes. The decision node in a tree which corresponds to the best predictor is the topmost node (root node).
- **The steps of performing Decision Tree Regression on prudential dataset:**
 1. Retrieving the final preprocessed data and splitting dataset into 80% train and 20% test datasets.
 2. Fitting the model using `rpart()` on train dataset to train the model.
 3. Printing plots for train data and plot the decision tree
 4. Plot cross-validation results.
 5. Predicting the Response for test data.
 6. Printing the confusion matrix and computing the model accuracy and correlation.
 7. Computing the Root Mean Squared Error (RMSE) between actual and predicted data.
 8. Adding the predicted Response column to the dataset and writing it to a new file.

Performing Decision Tree Regression

```
#Fitting the Tree Regression model on train dataset
tree_reg_model <- rpart(Response~.,method="anova", data=Prudential_train)
#Printing plots for train data
tree_reg_model
summary(tree_reg_model) # detailed summary
rpart.plot(tree_reg_model, type= 4,digits=3, extra = 101 )
```



Performing Decision Tree Regression

```
#Predicting the Response for test data
```

```
Prediction <- predict(tree_reg_model, Prudential_test[-65])
```

```
> confusion_Matrix <- table(Prediction,Prudential_test$Response)
> print(confusion_Matrix)
```

Prediction	1	2	3	4	5	6	7	8
2.76386768447837	111	230	12	0	120	21	3	2
3.33733333333333	53	66	0	0	23	59	2	0
4.24166666666667	15	13	0	0	95	0	0	0
4.32119914346895	133	109	53	70	68	180	75	104
4.5088432384928	313	325	98	10	553	680	241	42
5.29714285714286	105	99	1	3	53	112	346	37
5.48913797088052	219	190	51	165	71	628	253	500
6.84274764438803	271	273	8	36	118	498	730	3231

```
> accuracy <- (sum(diag(confusion_Matrix))/sum(confusion_Matrix)) * 100
> print(accuracy)
[1] 37.01271
```

```
> correlation_accuracy<- cor(Prediction,Prudential_test$Response)
> print(correlation_accuracy)
[1] 0.4964292
```

```
> RMSE <- rmse(Prudential_test$Response,Prediction)
> print(RMSE)
[1] 2.129498
```

SVM Regression

- Support vector machines (SVMs) are a set of supervised learning methods used for classification regression and outlier.
- It is effective in high dimensional spaces.
- Different Kernel function can be specified for the decision function.
- **The steps of performing SVM regression on prudential dataset:**
- 1. Retrieving the final cleaned data and splitting dataset into 80% train and 20% test datasets.
- 2. Tuning the SVM models to get optimal cost and gamma parameter which will be used for creating SVM model. Four kernels are used: radial, polynomial, sigmoid and linear.
- 3. Using the parameters calculated with the tune function and train data to fit SVM model.
- 4. Using the SVM model and test data to predict the Response.
- 5. Printing the accuracy.
- 6. After comparing the Root Mean Squared Error(RMSE) of the above 4 predictions, SVM model with linear kernel is the best one because it has the smallest RMSE value.

SVM Regression

- Screenshots of using tune function to decide the value of parameters:

```
# 1.Using Radial kernel
tune_radial <- tune(svm,Response ~ .,
                   data = Prudential_train2,
                   kernel="radial",
                   ranges=list(cost=10^(-1:2),
                               gamma=c(.5,1,2),
                               scale=F
                                ))

# showing the value of gamma and cost
summary_radial <- summary(tune_radial)
print(summary_radial$best.parameters)
```

```
> print(sumry_sigm$best.parameters)
cost gamma scale
4  100   0.5 FALSE
```

```
> print(summary_radial$best.parameters)
cost gamma scale
3   10   0.5 FALSE
```


SVM Regression

- Screenshots of comparing the value of RMSE to decide with kernel is better:

```
prediction4 <- predict(svm_linear, Prudential_test2[-65],type="class")  
accuracy(prediction4, Prudential_test2$Response)#RMSE value is 2.242062
```

```
> accuracy(prediction3, Prudential_test2$Response)  
      ME      RMSE      MAE      MPE      MAPE  
Test set -0.425 2.531798 2.022 -69.20964 89.44536
```

```
> accuracy(prediction2, Prudential_test2$Response)  
      ME      RMSE      MAE      MPE      MAPE  
Test set -0.5856486 2.517406 1.981792 -71.7298 89.49728
```

```
> accuracy(prediction1, Prudential_test2$Response)  
      ME      RMSE      MAE      MPE      MAPE  
Test set -0.1135444 2.436015 1.978688 -57.0579 81.63623
```

```
> accuracy(prediction4, Prudential_test2$Response)  
      ME      RMSE      MAE      MPE      MAPE  
Test set -0.6455554 2.242062 1.600623 -60.32487 74.64904
```

XGBoost – eXtreme Gradient Boosting

For XGboost, we must set three types of parameters: general parameters, booster parameters and task parameters.

- **General parameters** relates to which booster we are using to do boosting, commonly tree or linear model
- **Booster parameters** depends on which booster you have chosen.
- **Learning Task** parameters that decides on the learning scenario, for example, regression tasks uses different parameters with ranking tasks.

- eta [default=0.3, alias: learning_rate]
 - step size shrinkage used in update to prevents overfitting. After each boosting step, we can directly get the weights of new features. and eta actually shrinks the feature weights to make the boosting process more conservative.
 - range: [0,1]
- gamma [default=0]
 - minimum loss reduction. The larger, the more conservative the algorithm will be.
 - range: [0,∞]
- lambda [default=1, alias: reg_lambda]
 - L2 regularization term on weights, increase this value will make model more conservative.
- alpha [default=0, alias: reg_alpha]
 - L1 regularization term on weights, increase this value will make model more conservative.
- verbose = 0, no message

Grid search for various parameters. Using only 10% of the data.

```
# Doing grid search and first assigning the smallest error value as 100
smallestError <- 100
for (depth in seq(1,10,1)) {
  for (rounds in seq(1,20,1)) {
    for (lambda in seq(0,1,0.1)){
      for (alpha in seq(0,1,0.1)){

# Doing grid search by putting in the 4 given paramters in step values
#to get the least rmse error value
bst <- xgboost(data = as.matrix(trainSet[,predictors]),
label = trainSet[,outcomeName],
max.depth=depth, nround=rounds, lambda= lambda, alpha = alpha,
objective = "reg:linear", verbose=0)
gc()
```

```
[1] "8 16 0 0.6 1.97601345577155"
[1] "8 16 0.1 0.6 1.97477719672225"
[1] "8 16 0.1 0.6 1.97237136663544"
[1] "8 17 0.1 0.6 1.97185417282665"
[1] "8 18 0.1 0.6 1.97079190811744"
[1] "8 19 0.1 0.6 1.97050608566629"
[1] "8 20 0.1 0.6 1.96910708404833"
```

```
smallestError <- 100
for (eta in seq(0,1,0.05)) {
  for (gamma in seq(0,5,0.1)){

# Doing grid search by putting in the 2 given paramters in step values
#to get the least rmse error value
bst <- xgboost(data = as.matrix(trainSet[,predictors]),
label = trainSet[,outcomeName], max.depth=8, nround=20
,lambda= 0.1, alpha = 0.6, objective = "reg:linear", verbose=0,
eta= eta, gamma= gamma )
gc()

# prediction is done and rmse error is calculated
```

Console C:/Users/Abhinav/Desktop/ADS/ADS- Midterm Project/

```
[1] "0.15 0.1 1.99452587821197"
[1] "0.15 0.2 1.99259124859593"
[1] "0.15 1 1.99232608530841"
[1] "0.15 1.1 1.99224798780211"
[1] "0.15 3.5 1.99221794341415"
[1] "0.15 3.6 1.99184334367971"
[1] "0.2 0 1.97755519033372"
[1] "0.2 0.7 1.97695218012457"
[1] "0.25 0.7 1.97688139358517"
[1] "0.25 0.9 1.97564902564105"
[1] "0.25 1 1.97564376770963"
[1] "0.25 1.4 1.97493989382763"
```

```
# Training the model using the values from the grid search
# for optimal parameters
bst <- xgboost(data = as.matrix(trainSet[,predictors]),
              label = trainSet[,outcomeName],
              max.depth=8, nround=20, lambda= 1, alpha = 0.6, eta = 0.25,
              gamma=1.4, objective = "reg:linear", verbose=0)

# Predicting the testset using the model trained
pred <- predict(bst, as.matrix(testSet[,predictors]), outputmargin=TRUE)

# Calculating rmse value of the same
r1 <- rmse(as.numeric(testSet[,outcomeName]), as.numeric(pred))
r1
```

```
# Calculating rmse value of the same
r1 <- rmse(as.numeric(testSet[,outcomeName]), as.numeric(pred))
r1
[1] 1.972631
```

```
# Using booster gblinear
bst <- xgboost(data = as.matrix(trainSet[,predictors]),
              label = trainSet[,outcomeName],
              max.depth=8, nround=20, lambda= 1, alpha = 0.6,
              gamma=1.4, objective = "reg:linear", booster = "gblinear", verbose=0)
pred <- predict(bst, as.matrix(testSet[,predictors]), outputmargin=TRUE)
r2 <- rmse(as.numeric(testSet[,outcomeName]), as.numeric(pred))
r2
print(pred)
```

```
> pred <- predict(bst, as.matrix(testSet[,predictors]), outputmargin=TRUE)
> r2 <- rmse(as.numeric(testSet[,outcomeName]), as.numeric(pred))
> r2
[1] 2.058979
```




Conclusion

- The dataset had **128 features** and by feature engineering, we got **64 features** which significantly affected the response of the predictive model.
- Following are the **RMSE values** for the different predictive models:
 1. Linear Regression : RMSE **2.336**
 2. Decision Tree: RMSE **2.129**
 3. SVM : RMSE **2.242**
 4. XGBoost : RMSE **1.972**
- To conclude based on our results, the best predictive model for this dataset is **XGBoost**.



Future work/recommendations

- ▶ **Data cleaning** is the most important aspect to get the best predictions.
- ▶ We could work on a bigger data set to scale our results and improve **accuracy** of the predictions in future.
- ▶ This in turn could be used by various companies to target **customer-centric outcomes**.