Liberty Mutual Group: Property Inspection Prediction

Final Project: DSA 5103

Team members:

Chanukya Lakamsani, chanukya.lakamsani-1@ou.edu

Akhil Sanjay Potdar, akhilpotdar@ou.edu

Ghaneshvar Ramineni, ghaneshvar@ou.edu

Data Science &Analytics,

University of Oklahoma, Norman

Contents

Executive Summary

Insurance is a means of protection from financial loss. It is a form of risk management, primarily used to hedge against the risk of a contingent or uncertain loss. An entity which provides insurance is known as an insurer, insurance company, insurance carrier or underwriter. A person or entity who buys insurance is known as an insured or as a policyholder. The insurance transaction involves the insured assuming a guaranteed and known relatively small loss in the form of payment to the insurer in exchange for the insurer's promise to compensate the insured in the event of a covered loss. The loss may or may not be financial, but it must be reducible to financial terms, and usually involves something in which the insured has an insurable interest established by ownership, possession, or pre-existing relationship. There are many types of insurance and property insurance is one among them. Property insurance provides protection against most risks to property, such as fire, theft and some weather damage.

In this project, we are predicting the hazard score by taking into consideration the 33 variables that are anonymized and checking the accuracy with the help of normalized Gini index that is calculated (explained below) upon the submission of the predicted values. Here, since it is normalized Gini index, higher the value, higher is the accuracy. Thus, our goal is to achieve an index as high as possible.

In our modelling, since the predictors were all anonymized, feature selection was not extensive. Different models were created with xgboost, random forest, Bayesian linear regression, lasso, and ridge. The data was used after converting the factor columns into numeric data, further transforming the dataframe into a matrix. A second approach using one hot encoding was also made. For visually analyzing the data, correlation matrix plot, histograms and variable importance plots were used.

On comparison of our models, we realized that xgboost has been the most effective method. Our evaluation parameters were the Gini index and root mean squared error (RMSE) values amongst the various others. In conclusion, we were able to predict the Hazard scores with high accuracy as when compared to the winner’s Gini index in the competition. Also an alternative approach to classify the hazard levels is also proposed.

Problem description

Background

A home insurance score is a numerical value or rating that insurance companies use to determine risk. The factors used in the insurance point system are similar to those that make up our credit scores. Insurers use your credit information to help determine the likelihood that you’ll file a claim during the term of your policy.

Kaggle competition

This report is based on a Kaggle competition to predict hazard score based on anonymized variables provided. This competition was held by a Fortune 100 company, Liberty Mutual Insurance that has provided a wide range of insurance products and services designed to meet the customers' ever-changing needs.

To ensure that Liberty Mutual's portfolio of home insurance policies aligns with their business goals, many newly insured properties are inspected. These inspections review the condition of key factors of the property, including things like the foundation, windows, roof, siding, compound and quality of walls. The results of an inspection help the company determine if the property is one they want to insure or not.

In this quest, the task is to predict a transformed score of hazards or pre-existing damages using the given dataset of property information. This will enable Liberty Mutual to more precisely identify homes with high risk and that require additional examination to confirm their insurability.

Prediction Evaluation: (defined in the competition)

Submissions are evaluated on the normalized Gini coefficient. Gini coefficient can be used to check linearity in the model. And we can also rank variable based on their GINI coefficient. A higher Gini coefficient suggests a higher potential for the variable to be useful in a regression.

To calculate the normalized Gini, your predictions are sorted from largest to smallest. This is the only step where the explicit prediction values are used (i.e. only the order of your predictions matters). We then move from largest to smallest, asking "In the leftmost x% of the data, how much of the observed loss have you accumulated?" With no model, you expect to accumulate 10% of the loss in 10% of the predictions, so no model (or a "null" model) achieves a straight line. The area between your curve and this straight line is the Gini coefficient.

There is a maximum achievable area for a perfect model. The normalized Gini is obtained by dividing the Gini coefficient of your model by the Gini coefficient of a perfect model.

The **Gini Coefficient** is to normalize the AUC so that a basic classifier scores 0, and a perfect classifier scores 1. The range of possible Gini coefficient scores is [-1, 1]. While the **Area Under Receiver Operating Characteristic** curve (or AUROC for short) is the summary statistic of the ROC curve chart, the Gini index is the summary statistic of the Cumulative Accuracy Profile (CAP) chart. It is calculated as the quotient of the area which the CAP curve and diagonal enclose and the corresponding area in an ideal rating procedure. The direct conversion between Gini and AUROC is given by: $\text{Gini}=2\times\text{AUROC}-1$. The use of Gini index instead of AUROC curve is just a requirement from the competition's end. The aim of this score is its prediction of the accuracy.

Method of Approach

Data description:

Each row in the dataset corresponds to a property that was inspected and given a hazard score ("Hazard"). You can think of the hazard score as a continuous number that represents the condition of the property as determined by the inspection. Some inspection hazards are major and contribute more to the total score, while some are minor and contribute less. The total score for a property in the sum of the individual hazards. The aim of the competition is to forecast the hazard score based on anonymized variables which are available before an inspection is ordered.

Train data has 50,999 observations with 34 variables (including the target variable – Hazard) and Test data has 51,000 observations with 33 variables for which the target variable is to be predicted. Considering all target variables as output continuous variables since there are more than 40 levels.

Dealing with Ordinal data:

Considering as nominal: analysis like Chisq-test, logistic regression, loglinear models. It also works when we have dependent or independent or both as ordinal data. This method cannot be considered as the ordering is important (which means
Dividing data based on some criteria) as the primary aim is determine the Hazard
Ranking for a given data set. Considering ordinal data as nominal will fail to answer the question of ordering.

Considering as Numeric: considering each category as actual numbers will enable us to more flexible in choice of analysis and also the preserve the information of ordering. The downside of considering ordinal data as numeric is assuming the numerical distance between each category is equal (all Independent variables are equally important in determining the Independent variable).

Problem definition

The goal of this report is to explain the steps and methodologies followed to predict the hazard score and to possibly compare the results (hazard scores) of different types of models applied on the data. This a linear regression problem where the target value or the hazard score values are continuous numbers.

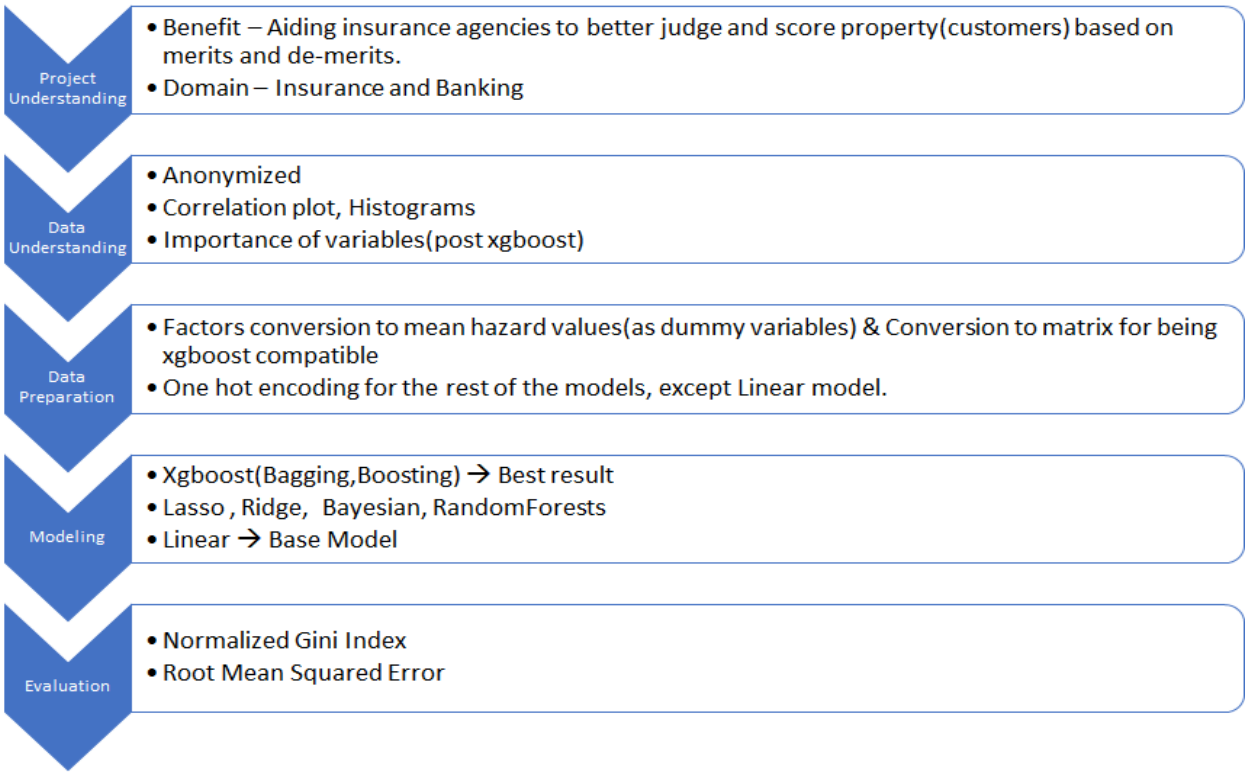


Fig. CRISPDM: Cross-Industry Standard Process Data Mining

Data exploration

It is observed there are no missing values in the data. Since all the variables are anonymous, feature selection can be done using NearZeroVar or pairwise correlation. These methods do not need the knowledge of what the feature is. However, upon doing so we didn't find any increase in the accuracy of our predictions and hence the features were left out so that we could use the full potential of the given data.

Also, as there are 16 predictors having alphabetic data, there isn't much feature engineering that can be done on these. Instead these factors were transformed into numeric data by taking the mean of the hazard score of the respective factor (These values could also be dummy variables, however for meaning engineering we choose the mean hazard score). The dataframe was then transformed into a matrix, so that it is compatible with xgboost function. Similarly, for the other models, we did one hot encoding. A linear model without any feature engineering was used as a base line.

Correlation matrix plot, histograms and variable importance plots were used to visualize the data. The plots are shown below.

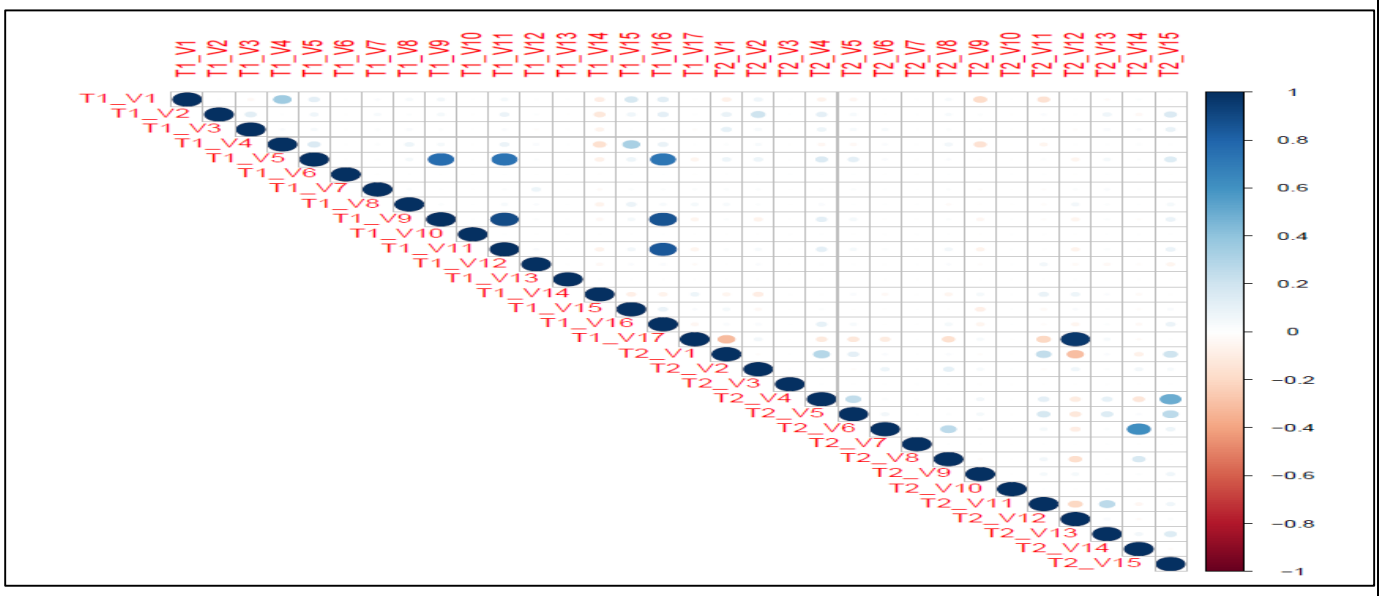


Fig. Correlation plot of the variables

Feature selection was not done based on correlation because boosted trees are immune to multi collinearity. From the above plot, we can observe that: T1_V12 vs. T1_V17 being the strongest, but also further correlations exist between T1_V9 vs. T1_V5, T1_V16 vs. T1_V5 and T1_V9.

From the variable importance plot, we see that the variable T2_V1 is of highest importance while T2_V8 is of the lowest.

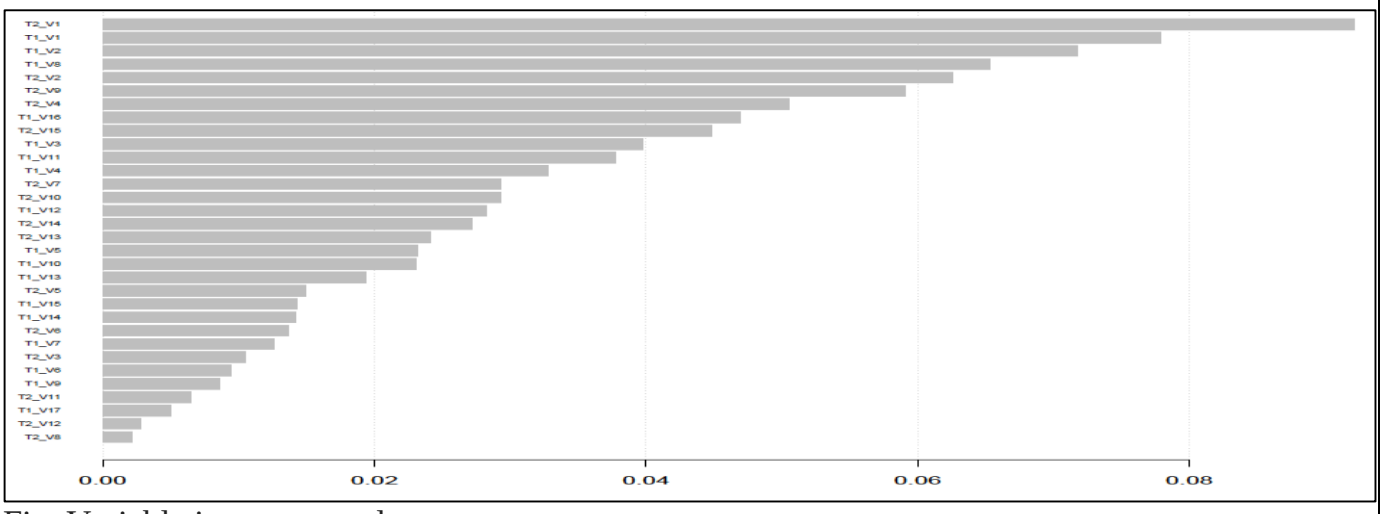


Fig. Variable importance plot

Modelling

We are following evaluation parameters RMSE (root mean square error), relative absolute error (Absolute error: Difference between predicted and exact value), and coefficient of determination (coefficient of determination: How much percentage of train data determines the test data. It is also called R-squared commonly) were generated by implementing cross validation on training data set by converting 75% of the data into a sub training set and the remaining 25% into test set. After which the model was run against the main test data for which the Hazard score was to be predicted. The predictions on the main test data gave us the output, which upon submission (on Kaggle) gave us Gini index.

Higher Gini index indicates a good score. The best score obtained in the competition was 0.397064 (private score). Hence our goal is to approach this value or obtain a better score. In this event, the following models gave us the respective results.

Table 1: Comparison of different models performed

Sl.no	Model Name	Function() / Method	R Package	RMSE value	Gini Index
1	XGBoost	xgboost	xgboost	3.810055	0.383179
2	Bayesian Linear Regression	bas.lm	BLR	3.859	0.326371
3	Random Forest	rf	randomForest	4.03	0.257711
4	Lasso	lasso	lars	10.32	0.13709
5	Ridge	ridge	ridge	7.9841	0.134374
6	Linear	lm	stats	17.6246	-0.00714

The above table is ranked from best to the worst Gini index obtained, as it was the evaluation parameter of the competition. A linear model as a base line was used since it is considered as a regression problem.

Gradient Boosting

In an algorithm we perform bagging for boosted trees, thereby effectively combining the advantages of both. In order to perform boosting, we created a log file that kept the changing configuration of the boosted tree as the loop is repeated. This process was repeated to create 50 bags (models = 5 and repeats = 10). Each with their individual seed, so that we could create reproducible results for each bag.

Model Evaluation Trainset:

Root Mean Squared Error	3.810055
Relative Absolute Error	0.97064
Relative Squared Error	0.908255
Coefficient of Determination	0.091745

Model Evaluation on Test set:

GINI Index: Public Private

Bayesian linear regression

Model Evaluation on Train set

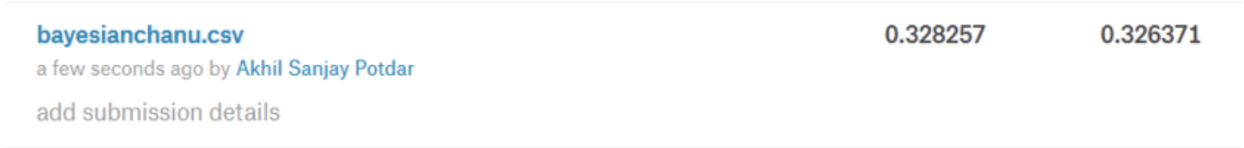
RMSE: 3.859

Relative absolute error: 0.971571

Coefficient of Determination: .097927

Result:

Gini index:



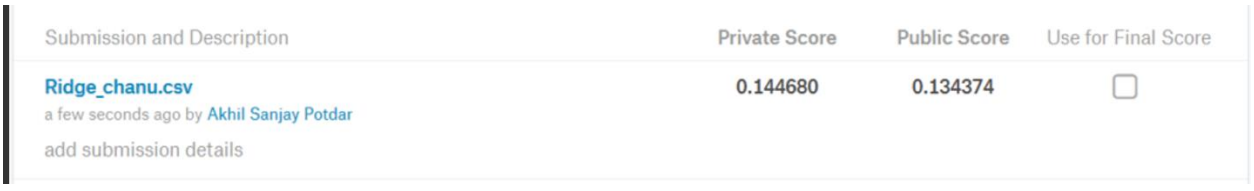
Lasso Using Cross validation, Ridge Regression

Model evaluation Trainset:

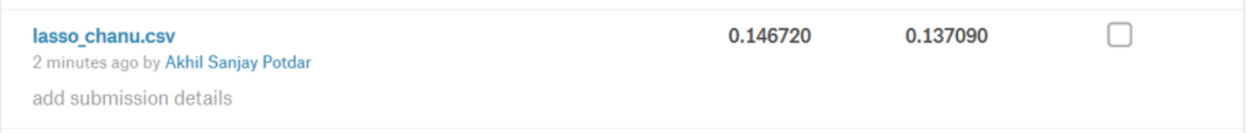
Model	RMSE
Ridge(alpha=0)	7.9841
Lasso(alpha =1)	10.32

Result on Testset:

GINI Index(for Ridge):



GINI Index(for Lasso):



Random Forest Regression:

Model Evaluation on Test data:

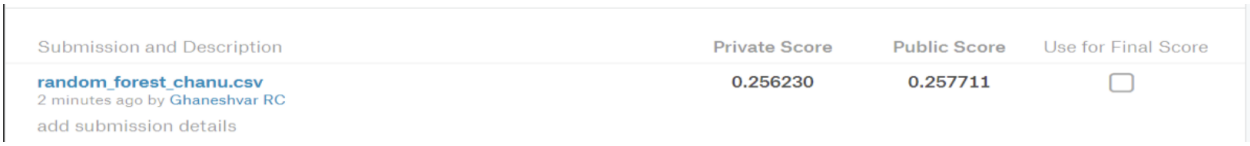
RMSE: 4.03

Relative Absolute Error: 2.88

Relative Squared Error: 1.01

Model Evaluation on Test Dataset:

GINI Index:



Evaluation

Different models were evaluated and tested on the dataset that was given in the Kaggle. Among all the models performed, GINI Index was highest for Gradient boosting regression due to the usage of bagging and boosting (as explained above in Gradient Boosting).

In comparison with Random Forest and Bayesian Linear Regression models were closer to the highest value. The linear model was performed to show the impact of other models.

The main objective was to see how close we could get to the competition winner and we successfully achieved an index of 0.383179 (using gradient boosting) which was short of the winner's index value (0.397064) by a very small margin.

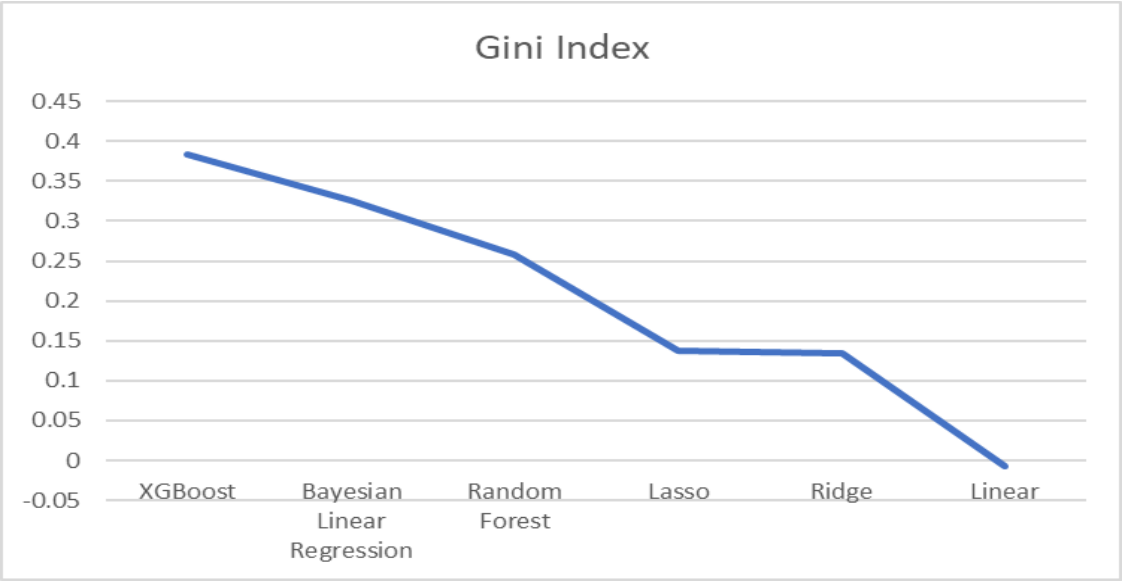


Fig: Graph of Gini index vs type of model

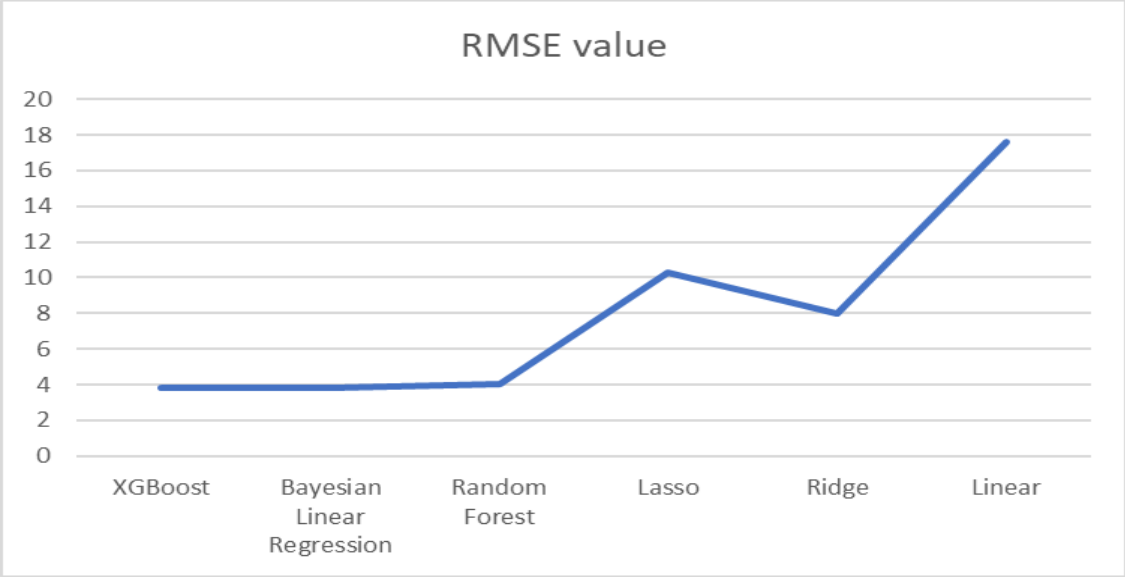


Fig: Graph of RMSE vs type of model

We can observe from the above two graphs that with the decrease in Gini index, the RMSE value increases.

Conclusion

Using this model, we believe that the Property Inspection Prediction can be accurately done for the Liberty Mutual Group. We believe that, if the predictors were not anonymized, better predictions could have been generated. We can say so because data understanding is integral to the CRISPDM procedure of data science. During this endeavor, we as a team

learnt a lot and through this expertise gained, we believe that we are better equipped to face more of such real life data intensive problems.

Furthermore, we would like to thank Dr. Charles Nicolson and Ms. Yunjie Wen for their evergreen support, motivation and insights. The design of the course has helped us to tackle real time data. Also we would like to thank out peer reviewers, Gor Beglarian and Yan Liang for their detailed scrutiny. This entire design of the course project helped us get to these results. Thanking you once again for the opportunity.

Second Approach: Classification

Classification can also be considered if we don't have the question of ordering on validation set.

Data Transformation: Data binning (since there are lot of levels of Target variables, we tend to decrease the number of levels by grouping the levels).

Transforming.
Optical scaling regression: Transforms the ordinal predictor into and interval one so as to maximize the linear effect on the predictors. Why apply only regression on ordinal data? Since the data is continuous, classification will not give good results where regression will do. However, for the sake of easing prediction parameters for faster outputs we are suggesting this approach. This however will not be used for predicting the actual Hazard score but only its bins.

We grouped the data into four levels:

0-10, 10-20,20-30,>30 which are 1,2,3,4. These are four factors.

Random Forest Classifier:

Model Evaluation from confusion matrix:

Accuracy: Accuracy: How often the classifier is correct. We got 0.931 as accuracy which is 93.1 % times model predicts accurate values. Precision: description of the standard variability. Recall : How sensitive is the model (lower the value better the results).

Accuracy : 0.931 ; Precision : 0.91 ; Recall : 0.25

We have tested the model on test dataset we have got the following results:

1^{st} level : 50837 ; 2^{nd} level : 157 ; 3^{rd} level: 4 ; 4^{th} level : 2

Representation of the levels is as follows :

1 : 0 to 10 ; 2 : 11 to 20 ; 3: 21 to 30 ; 4: 31 to 69

References:

i) Kaggle. (2015). Liberty Mutual Group: Property Inspection Prediction, Quantify property hazards before time of inspection.

Retrieved from Kaggle: https://www.kaggle.com/c/liberty-mutual-group-property-inspection-prediction

ii) Wikipedia (2018): Insurance

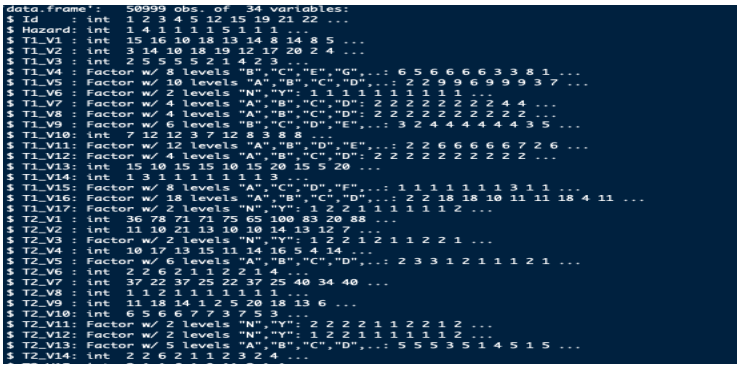
Retrieved from: https://en.wikipedia.org/wiki/Insurance

iii) Wikipedia (2018): Property insurance

Retrieved from: https://en.wikipedia.org/wiki/Property_insurance

Appendix

Structure of data:



```
'data.frame':   50999 obs. of  34 variables:
$ Id    : int  1 2 3 4 5 12 15 19 21 22 ...
$ Hazard: int  1 4 1 1 1 1 5 1 1 1 ...
$ T1_V1 : int  15 16 10 18 13 14 8 14 8 5 ...
$ T1_V2 : int  3 14 10 18 19 12 17 20 2 4 ...
$ T1_V3 : int  2 5 5 5 5 2 1 4 2 3 ...
$ T1_V4 : Factor w/ 8 levels "B","C","E","G",..: 6 5 6 6 6 6 3 3 8 1 ...
$ T1_V5 : Factor w/ 10 levels "A","B","C","D",..: 2 2 9 9 6 9 9 9 3 7 ...
$ T1_V6 : Factor w/ 2 levels "N","Y": 1 1 1 1 1 1 1 1 1 1 ...
$ T1_V7 : Factor w/ 4 levels "A","B","C","D": 2 2 2 2 2 2 2 2 4 4 ...
$ T1_V8 : Factor w/ 4 levels "A","B","C","D": 2 2 2 2 2 2 2 2 2 2 ...
$ T1_V9 : Factor w/ 6 levels "B","C","D","E",..: 3 2 4 4 4 4 4 4 3 5 ...
$ T1_V10: int  7 12 12 3 7 12 8 3 8 8 ...
$ T1_V11: Factor w/ 12 levels "A","B","D","E",..: 2 2 6 6 6 6 6 7 2 6 ...
$ T1_V12: Factor w/ 4 levels "A","B","C","D": 2 2 2 2 2 2 2 2 2 2 ...
$ T1_V13: int  15 10 15 15 10 15 20 15 5 20 ...
$ T1_V14: int  1 3 1 1 1 1 1 1 3 ...
$ T1_V15: Factor w/ 8 levels "A","C","D","F",..: 1 1 1 1 1 1 1 3 1 1 ...
$ T1_V16: Factor w/ 18 levels "A","B","C","D",..: 2 2 18 18 10 11 11 18 4 11 ...
$ T1_V17: Factor w/ 2 levels "N","Y": 1 2 2 1 1 1 1 1 1 2 ...
$ T2_V1 : int  36 78 71 71 75 65 100 83 20 88 ...
$ T2_V2 : int  11 10 21 13 10 10 14 13 12 7 ...
$ T2_V3 : Factor w/ 2 levels "N","Y": 1 2 2 1 2 1 1 2 2 1 ...
$ T2_V4 : int  10 17 13 15 11 14 16 5 4 14 ...
$ T2_V5 : Factor w/ 6 levels "A","B","C","D",..: 2 3 3 1 2 1 1 1 2 1 ...
$ T2_V6 : int  2 2 6 2 1 1 2 2 1 4 ...
$ T2_V7 : int  37 22 37 25 22 37 25 40 34 40 ...
$ T2_V8 : int  1 2 1 1 1 1 1 1 1 1 ...
$ T2_V9 : int  11 18 14 1 2 5 20 18 13 6 ...
$ T2_V10: int  6 5 6 6 7 7 3 7 5 3 ...
$ T2_V11: Factor w/ 2 levels "N","Y": 2 2 2 2 1 1 2 2 1 2 ...
$ T2_V12: Factor w/ 2 levels "N","Y": 1 2 2 1 1 1 1 1 1 2 ...
$ T2_V13: Factor w/ 5 levels "A","B","C","D",..: 5 5 5 3 5 1 4 5 1 5 ...
$ T2_V14: int  2 2 6 2 1 1 2 3 2 4 ...
```

Summary of data:

```
       Id              Hazard          T1_V1           T1_V2           T1_V3      
 Min.   :     1   Min.   : 1.000   Min.   : 1.000   Min.   : 1.00   Min.   :1.000  
 1st Qu.: 25660   1st Qu.: 1.000   1st Qu.: 6.000   1st Qu.: 7.00   1st Qu.:2.000  
 Median : 50977   Median : 3.000   Median : 9.000   Median :14.00   Median :3.000  
 Mean   : 50930   Mean   : 4.023   Mean   : 9.722   Mean   :12.85   Mean   :3.186  
 3rd Qu.: 76268   3rd Qu.: 5.000   3rd Qu.:14.000   3rd Qu.:18.00   3rd Qu.:4.000  
 Max.   :101999   Max.   :69.000   Max.   :19.000   Max.   :24.00   Max.   :9.000  

    T1_V4           T1_V5       T1_V6      T1_V7      T1_V8      T1_V9         T1_V10      
 N      :25112   K      :14138   N:28550   A:  546   A: 1033   B: 2748   Min.   : 2.00  
 B      :12502   A      :10900   Y:22449   B:47982   B:46391   C:  251   1st Qu.: 3.00  
 C      : 4869   H      :10137             C:  345   C: 1547   D:23859   Median : 8.00  
 W      : 3480   C      : 7845             D: 2126   D: 2028   E:21491   Mean   : 7.02  
 E      : 2702   I      : 4297                                 F: 2432   3rd Qu.: 8.00  
 S      : 1051   B      : 3335                                 G:  218   Max.   :12.00  
 (Other): 1283   (Other):  347                                                          
    T1_V11      T1_V12         T1_V13         T1_V14          T1_V15         T1_V16     
 B      :17047   A: 1130   Min.   : 5   Min.   :0.000   A      :45680   I      : 9331  
 H      :15381   B:46900   1st Qu.:10   1st Qu.:1.000   N      : 1879   B      : 8933  
 L      : 7003   C: 1395   Median :15   Median :1.000   C      : 1652   R      : 8339  
 J      : 6197   D: 1574   Mean   :14   Mean   :1.579   D      :  758   K      : 8159  
 A      : 1556             3rd Qu.:20   3rd Qu.:2.000   H      :  524   A      : 2705  
 I      : 1364             Max.   :20   Max.   :4.000   W      :  230   E      : 2599  
 (Other): 2451                                          (Other):  276   (Other):10933  
 T1_V17       T2_V1           T2_V2          T2_V3          T2_V4          T2_V5      
 N:41183   Min.   :  1.00   Min.   : 1.00   N:34548   Min.   : 1.00   A:33845  
 Y: 9816   1st Qu.: 40.00   1st Qu.: 9.00   Y:16451   1st Qu.: 6.00   B:11201  
           Median : 56.00   Median :11.00             Median :10.00   C: 5013  
           Mean   : 57.58   Mean   :12.42             Mean   :10.26   D:  515  
           3rd Qu.: 77.00   3rd Qu.:15.00             3rd Qu.:14.00   E:  412  
           Max.   :100.00   Max.   :39.00             Max.   :22.00   F:   13  

    T2_V6           T2_V7           T2_V8           T2_V9          T2_V10      
 Min.   :1.000   Min.   :22.00   Min.   :1.000   Min.   : 1.00   Min.   :1.000  
 1st Qu.:2.000   1st Qu.:31.00   1st Qu.:1.000   1st Qu.: 6.00   1st Qu.:3.000  
 Median :2.000   Median :34.00   Median :1.000   Median :14.00   Median :4.000  
 Mean   :1.948   Mean   :33.49   Mean   :1.032   Mean   :12.49   Mean   :4.497  
 3rd Qu.:2.000   3rd Qu.:40.00   3rd Qu.:1.000   3rd Qu.:18.00   3rd Qu.:6.000  
 Max.   :7.000   Max.   :40.00   Max.   :3.000   Max.   :25.00   Max.   :7.000  

 T2_V11    T2_V12    T2_V13        T2_V14          T2_V15      
 N:14059   N:41703   A:10260   Min.   :1.000   Min.   : 1.000  
 Y:36940   Y: 9296   B:  514   1st Qu.:2.000   1st Qu.: 1.000  
                     C: 7507   Median :2.000   Median : 2.000  
                     D: 5084   Mean   :2.451   Mean   : 3.484  
                     E:27634   3rd Qu.:3.000   3rd Qu.: 5.000  
                               Max.   :7.000   Max.   :12.000  
```

We see that the data doesn't have any null values

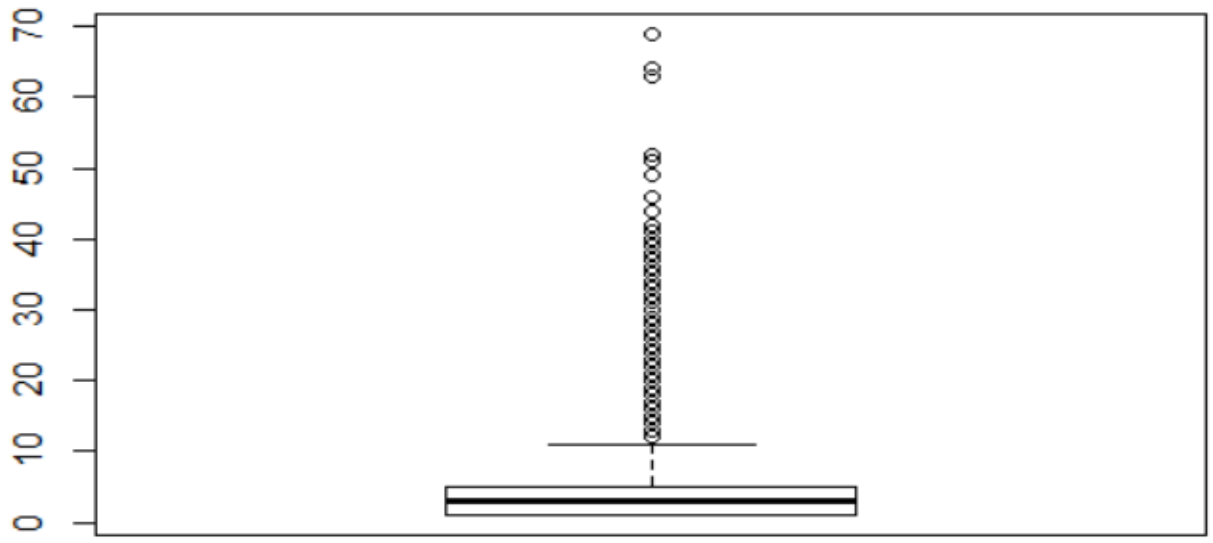


Fig: Boxplot of Hazard variable.

```
# replace factors with level mean or(median) hazard
for (i in 1:ncol(total))
{
  if (class(total[,i])=="factor")
    {
      mm <- aggregate(target~train[,i], data=train, mean)
      levels(total[,i]) <- mm[,2]
      total[,i] <- as.numeric(as.character(total[,i]))
    }
}

train_trans <- total[1:50999,]
test_trans <- total[51000:101999,]
train <- as.matrix(train_trans)
test <- as.matrix(test_trans)
```

Fig. Code for converting factors to numeric data.

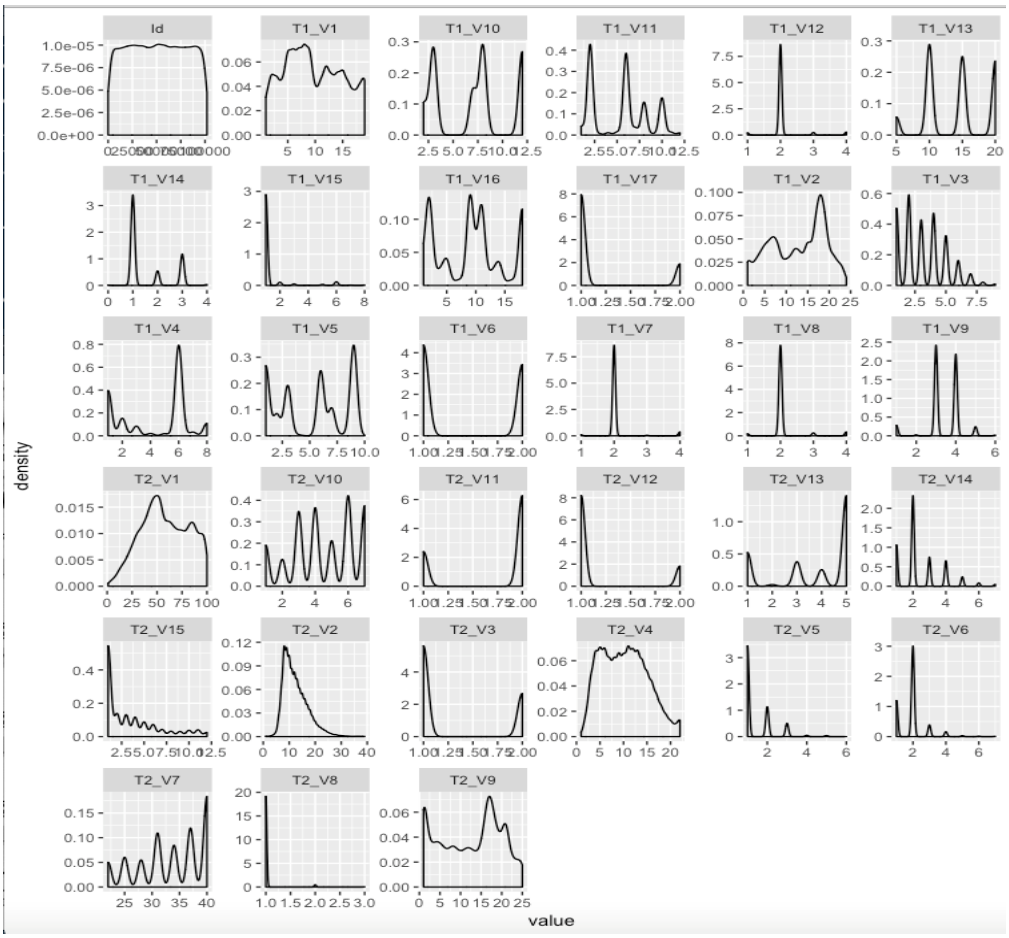


Fig: Understanding the distribution of data

```
# Prediction using 50 models through baging
models <- 5
repeats <- 10
targethat.test  <- rep(0,nrow(test))
for (j in 1:repeats) {
  for (i in 1:models){
    set.seed(i*2+j*2)
    xgboost.mod <- xgboost(data = train, label = target, max.depth = logfile$depth[i],
                           eta = logfile$shrinkage[i], nround = logfile$rounds[i],
                           nthread = 4, objective = "reg:linear", subsample=logfile$subsample[i],
                           colsample_bytree=logfile$colsample.bytree[i], gamma=logfile$gamma[i],
                           min.child.weight=logfile$min.child[i])
    targethat.test  <- targethat.test + predict(xgboost.mod, test)
  }
}
targethat.test <-  targethat.test/(models*repeats)
```

Fig. Code for xgboosted trees through bagging

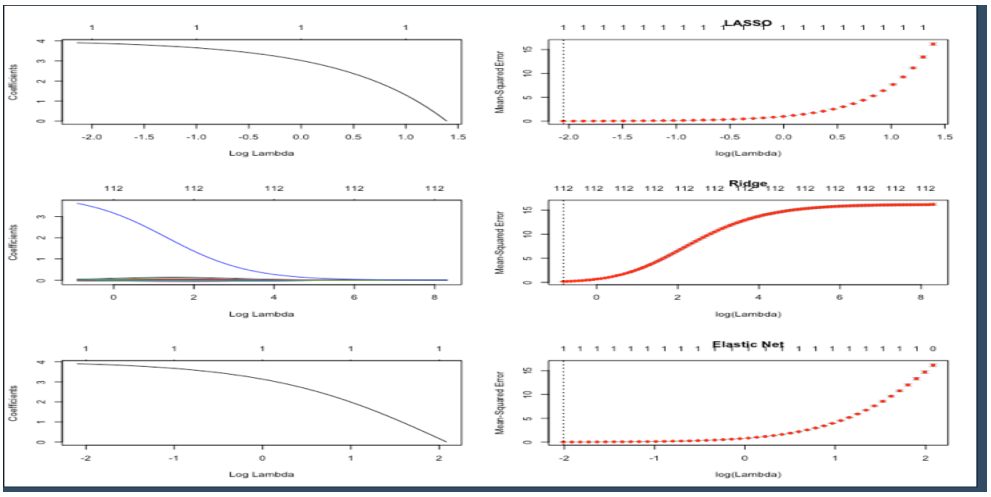


Fig: Plot solution path with respect to lambda values

Converting factor variables to dummies

Code:

```
for (f in cname1){
df_all_dummy = acm.disjonctif(final[f])
final[f] = NULL
dat = cbind(final, df_all_dummy)
}
```

Converting all the variable to numeric:

final <- as.data.frame(sapply(final, as.numeric)

```
# assignment 8 clustering
# the given dataset requires customer segmentation with 18 varaibles which explains the bheaviour of the custc

#reading the dataset in R
#install.packages("data.table")
install.packages("babar")
library(babar)
library(data.table)
library(caret)
library(ggplot2)
library(magrittr)
library(dplyr)
library(mlbench)
library(magrittr)
library(ggplot2)
library(tidyr)
library(PerformanceAnalytics)
library(MASS) # for transformations
library(moments)
library(corrplot)
library(tidyverse)
library(car)
library(cluster)    # clustering algorithms
library(factoextra) # clustering algorithms & visualization
library(dbscan)
library(ade4)
library(dummies)
install.packages('lars')
library(lars)
library(glmnet)
#install.packages('rcompanion')
library(rcompanion)
cc<- read.csv("/Users/chanukya/Desktop/IDA/Project/train.csv")
View(cc)
x <- as.factor(cc$Hazard)

levels(x)
test <- read.csv("/Users/chanukya/Desktop/IDA/Project/test.csv")
ID1 <- test$Id
test$Id <- NULL
str(test)
colnames(test)
ncol(test)
str(cc)
summary(cc)
dat <- cc
Hazard <- dat$Hazard
ncol(dat)


dat$Hazard <- NULL
dat$Id <- NULL
colnames(dat)
final <- rbind(dat,test)

#first lets see how the Varaibles are distributed.
dat %>%gather() %>%ggplot(aes(value)) +facet_wrap(~ key, scales = "free") + geom_density()


#converting all the factor variables to dummies
#library(dummies)
fact <- sapply((dat), is.factor)
fact <- dat[, fact]
cname1 <- colnames(fact)
colnames(dat)
(cname1)
for (f in cname1){
df_all_dummy = acm.disjonctif(final[f])
final[f] = NULL
final = cbind(final, df_all_dummy)
}
colnames(final)
str(dat)
write.csv("/Users/chanukya/Desktop/IDA/Project/dummy.csv")
#converting all the variables as numeric:

dat <- as.data.frame(sapply( final, as.numeric ))
str(dat)
```

```
summary(final_train)
colnames(final_train)
write.csv(final_train,"/Users/chanukya/Desktop/IDA/Project/dummy_train.csv")

final_test <- final[51000:101999,]
write.csv(final_test, "/Users/chanukya/Desktop/IDA/Project/dummy_test.csv")
dim(final)


#modelling.


#Ridge Regression
final_train1 <-as.data.frame( scale(final_train[,-1]))
model.ridge <- lm.ridge( Hazard~ ., data=final_train1)  #ridge regression is run for several values of lambda
qqnorm(model.ridge$coef)
str(model.ridge)

# The optimal lambda is given by

qplot(model.ridge$lambda,model.ridge$GCV )        # plots generalized cross validated error (GCV) vs the tuni
which.min(model.ridge$GCV)   # identifies the value for lambda with the smallest associated GCV
lambda.ridge <- seq(0,10,0.1)[which.min(model.ridge$GCV)]   #save the value of optimal lambda for future use



#Lasso Regression:

model.lasso <- lars(as.matrix(final_train1),Hazard, type="lasso")
plot(model.lasso)
View(final_train1)
predict()

#bayesian regression

set.seed(1234)  ## reproducible sim
lmfit <- lm(Hazard~ ., data=final_train1)
bf<-Bayesfit(lmfit,10000)
t(apply(bf,2,Bayes.sum))

#lasso, Ridge,Elastic net
final_train1$Hazard <- NULL
fit
fit.lasso <- glmnet(as.matrix(final_train1,Hazard, family="gaussian", alpha=1)
fit.ridge <- glmnet(as.matrix(final_train1),Hazard, family="gaussian", alpha=0)
fit.elnet <- glmnet(as.matrix(final_train1),Hazard, family="gaussian", alpha=.5)


# 10-fold Cross validation for each alpha = 0, 0.1, ... , 0.9, 1.0
# (For plots on Right)
for (i in 0:10) {
  assign(paste("fit", i, sep=""), cv.glmnet(as.matrix(final_train1),Hazard, type.measure="mse",
                                            alpha=i/10,family="gaussian"))
}


train1 <- final_train
nrow(final_train)
Hazard <- as.data.frame(Hazard)
train1 <- train1[0:39000,]

test1 <- final_train[39001:50999,]
test1$Hazard <- NULL
fit0 <- glmnet(as.matrix(train1),Hazard[0:39000,], family="gaussian", alpha=0)

yhat0 <- predict(fit0, s=fit0$lambda.1se, as.matrix(test1))
mse0 <- mean((Hazard[39001:50999,] - yhat0)^2)
RMSE <- sqrt(mse0)
RMSE


nrow(final_train1)*.75


cc$  <- cut(train1$Hazard, breaks=c(0,10,20,30,70), labels=c("1","2","3","4"))

table(train1$bins)


# Plot solution paths:
par(mfrow=c(3,2))
```

```
# For plotting options, type '?plot.glmnet' in R console
plot(fit.lasso, xvar="lambda")
plot(fit10, main="LASSO")

plot(fit.ridge, xvar="lambda")
plot(fit0, main="Ridge")

plot(fit.elnet, xvar="lambda")
plot(fit5, main="Elastic Net")


class(final_test)
dim(final_test)
dim(final_train)
#imputation
library(mice)
# Deterministic regression imputation via mice

cc <- mice(out, method = "norm.predict", m = 1)
cc1 <- read.csv("/Users/chanukya/Desktop/IDA/out2.csv")
yhat0 <- predict(fit0, s=fit0$lambda.1se, as.matrix(final_test))
write.csv(yhat0,"/Users/chanukya/Desktop/IDA/Ridge.csv")
yhat1 <- predict(fit1, s=fit1$lambda.1se, as.matrix(final_test))
yhat2 <- predict(fit2, s=fit2$lambda.1se, as.matrix(final_test))
yhat3 <- predict(fit3, s=fit3$lambda.1se, as.matrix(final_test))
yhat4 <- predict(fit4, s=fit4$lambda.1se, as.matrix(final_test))
yhat5 <- predict(fit5, s=fit5$lambda.1se, as.matrix(final_test))
yhat6 <- predict(fit6, s=fit6$lambda.1se, as.matrix(final_test))
yhat7 <- predict(fit7, s=fit7$lambda.1se, as.matrix(final_test))
yhat8 <- predict(fit8, s=fit8$lambda.1se, as.matrix(final_test))
yhat9 <- predict(fit9, s=fit9$lambda.1se, as.matrix(final_test))
yhat10 <- predict(fit10, s=fit10$lambda.1se, as.matrix(final_test))
write.csv(yhat10,"/Users/chanukya/Desktop/IDA/lasso.csv")

#Randome forest regression
set.seed(123)
valid_split <- initial_split(final_train, .8)

# training data
ames_train_v2 <- analysis(final_train)

# validation data
a

rf_oob_comp <- randomForest(
  formula = Hazard ~ .,
  data    = final[0:50999],
  xtest   = final_train,
  ytest   = Hazard$Hazard[]
)

# extract OOB & validation errors
oob <- sqrt(rf_oob_comp$mse)
validation <- sqrt(rf_oob_comp$test$mse)

# compare error rates
tibble::tibble(
  `Out of Bag Error` = oob,
  `Test error` = validation,
  ntrees = 1:rf_oob_comp$ntree
) %>%
  gather(Metric, RMSE, -ntrees) %>%
  ggplot(aes(ntrees, RMSE, color = Metric)) +
  geom_line() +
  scale_y_continuous(labels = scales::dollar) +
  xlab("Number of trees")

#View(cc1)
Hazard$Hazard <- as.factor(Hazard$Hazard)
levels(Hazard$Hazard)
hazard1 <- Hazard$Hazard
hazard1<- as.numeric(hazard1)

breaks <- c(10,20,40,50,60,70)
labels <- c("First", "Second", "Third", "Fourth", "Fifth", "Sixth")
bins <- cut(hazard1,breaks, include.lowest = T, right=FALSE, labels=labels)
hazard1 <- as.numeric(hazard1)
# classification
length(hazard1)
for(i in 1:length(hazard1))
  {
  if(0<hazard1[i]<10){
```

```
  }
  else if(10<hazard1[i]<20){
    hazard1[i] <- "Second"
  }
  else if(30< hazard1[i] <40){
    hazard1[i] <- "Third"
  }
  else if(40<hazard1[i]<50){
    hazard1[i] <- "Fourth"
  }
 else (hazard1[i]>50){
    hazard1[i] <- "Fifth"
  }


}
Hazard$Hazard <- as.numeric(Hazard$Hazard)
Hazard$Hazard[ Hazard$Hazard < 10 & Hazard$Hazard >=0] <- "FIRST"
Hazard$Hazard[Hazard$Hazard < 20 & Hazard$Hazard >=10] <-"Second"
Hazard$Hazard[Hazard$Hazard < 30 & Hazard$Hazard >=20] <-"Third"
Hazard$Hazard[Hazard$Hazard < 40 & Hazard$Hazard >= 50] <-"Fourth"
Hazard$Hazard[ Hazard$Hazard >=50] <-"Fifth"
# w

#classification
cc$Hazard <- cut(cc$Hazard, breaks=c(0,10,20,30,70), labels=c("1","2","3","4"))
final_train$Hazard <- cc$Hazard

ncol(final_train)
write.csv(final_train,"/Users/chanukya/Desktop/IDA/Project/dummy_train.csv")
ncol(final_test)

  set.seed(101)
model.rf=randomForest(Hazard ~ . , data = cc , subset = final_train[0:39001,])
model.rf
plot(model.rf)
pred<-predict(model.rf,tr)
test.err= with(final_test, mean( (Hazard[39001:50999] - pred)^2))
confusionMatrix(pred,Hazard[39001:50999] )
pred1 <- predict(model.rf,final_test)
```

```
###################################XGboost################3

library(readr)
library(reshape2)
library(xgboost)
library(DiagrammeR) #for xgb.plot.multi.trees
library(corrplot)
library(caret)

# Read data files:
train <- read.csv("train.csv")
test <- read.csv("test.csv")

cat(sprintf("Training set has %d rows and %d columns\n", nrow(train), ncol(train)))
cat(sprintf("Test set has %d rows and %d columns\n", nrow(test), ncol(test)))

# extract id
id_test <- test$Id
test$Id <- NULL
train$Id <- NULL
n <- nrow(X)

#No missing values
length(train[is.na(train)])
length(test[is.na(test)])

# extarct target
target <- train$Hazard
train$Hazard <- NULL
boxplot(target) # Has a lot of outliers

# summarize the Hazard or y variable, note that not every number is used (50 Levels only)
summary(target)
length(levels(as.factor(target)))

total=rbind(train,test) #

# Find near zero values
nzv <- nearZeroVar(total, saveMetrics= FALSE)
filteredtotal <- total[, -nzv]
```

```
names(filteredtotal)

# Checking the names of the predictors that are factors
factcol_total=names(total[,sapply(total, is.factor)] )

# replace factors with level mean or(median) hazard
for (i in 1:ncol(total))
{
  if (class(total[,i])=="factor")
  {
    mm <- aggregate(target~train[,i], data=train, mean)
    levels(total[,i]) <- mm[,2]
    total[,i] <- as.numeric(as.character(total[,i]))
  }
}

train_trans <- total[1:50999,]
test_trans <- total[51000:101999,]
train_trans2=cbind(train_trans,target)
write.csv(train_trans2,"train_numeric.csv",row.names=F, quote=FALSE)
train <- as.matrix(train_trans)
test <- as.matrix(test_trans)

#Correlation plot
cormat=cor(train_trans)
corrplot(cormat,type="upper")

logfile <- data.frame(shrinkage=c(0.04, 0.03, 0.03, 0.03, 0.02),
                      rounds = c(140, 160, 170, 140, 180),
                      depth = c(8, 7, 9, 10, 10),
                      gamma = c(0, 0, 0, 0, 0),
                      min.child = c(5, 5, 5, 5, 5),
                      colsample.bytree = c(0.7, 0.6, 0.65, 0.6, 0.85),
                      subsample = c(1, 0.9, 0.95, 1, 0.6))


# Prediction using 50 models through baging
models <- 5
repeats <- 10
targethat.test  <- rep(0,nrow(test))
for (i in 1:repeats) {
# Prediction using 50 models through baging
models <- 5
repeats <- 10
targethat.test  <- rep(0,nrow(test))
for (j in 1:repeats) {
  for (i in 1:models){
    set.seed(i*2+j*2)
    xgboost.mod <- xgboost(data = train, label = target, max.depth = logfile$depth[i],
                           eta = logfile$shrinkage[i], nround = logfile$rounds[i],
                           nthread = 4, objective = "reg:linear", subsample=logfile$subsample[i],
                           colsample_bytree=logfile$colsample.bytree[i], gamma=logfile$gamma[i],
                           min.child.weight=logfile$min.child[i])
    targethat.test  <- targethat.test + predict(xgboost.mod, test)
  }
}
targethat.test <-  targethat.test/(models*repeats)

#xgboost.mod2=xgboost(data=train,label=target,nrounds=10)
write.csv(data.frame(Id=id_test, Hazard=targethat.test),"xgboost_output_1bag.csv",row.names=F, quote=FALSE)

imp=xgb.importance(model=xgboost.mod)
order(imp)
rank(imp)
xgb.plot.importance(imp) #shows the importance of variables w.r.t the following:
"The meaning of the importance data table is as follows:

    The Gain implies the relative contribution of the corresponding feature to the model calculated by taking
    The Cover metric means the relative number of observations related to this feature. For example, if you ha
    The Frequence (frequency) is the percentage representing the relative number of times a particular feature

The Gain is the most relevant attribute to interpret the relative importance of each feature."
xgb.plot.deepness(model = xgboost.mod)
"There is more than one way to understand the structure of the trees, besides plotting them all. Since there a

From the function xgb.plot.deepness, we can get two plots summarizing the distribution of leaves according to
"The upper plot shows the number of leaves per level of deepness. The lower plot shows noramlized weighted cov
```