Final Report

Title: Predicting loss of an insurance claim

All state Kaggle Competition

Team: Model Ensemblers

Name	Net id
Jayachandu Bandlamudi	bandlmd2
Wenke Huang	whuang67
Yubai Yuan	yubaiy2

Link for Datasets: https://www.kaggle.com/c/allstate-claims-severity/data

We have four different R programs and needs to be run independently in this sequence.

1. **Data_exploration.R**
2. **Lasso.R**
3. **Tree_models.R**
4. **Ensemble.R**

Contents:

References:

1) Introduction

Use of statistical machine learning for solving many real-world problems across several domains is so prominent as well as very useful.Insurance domain is one such area, where machine learning can be used to predict 'loss' of an insurance claim by using a set of parameters.For the final project our group choose to work on a real time competition hosted on kaggle platform,majorly this competition is about predicting the 'loss' of an insurance claim.For this project we have two datasets i.e.; training dataset and testing set, so we will train several machine learning algorithms on training set on make predictions on testing set and submit the testing set predictions to kaggle platform to check the performance of our trained models.

Our goal for this project is to predict the (response variable) 'loss' of an claim based on several independent variables. These independent variables includes both continuous and categorical predictor variables, Mean Absolute Error(MAE) is the error metric that we will be minimising during the training process of several models.

Pipeline for the project:

Fig-1:Pipeline for the project

Figure-1 shows the pipeline for the project, at first we explore the datasets in order to get a better sense of data before moving on to actual model training.After data exploration we will try different methods for feature selection also we will create new features from the existing one's.Next step is Learning ,it includes training several models such as LASSO,RF,GBM,XGboost etc. (**Note**:in the learning step ,at first we will do feature selection based on supervised models which have embed variable selection capabilities,and we will use variable selection from this step to generate best set).After the learning step each model performance is evaluated using MAE error metric, in the final step we ensemble a set of best models to obtain much better performing model for final submission.

2) Basic data exploration

In this section we will explore the datasets, as mentioned earlier we have two data sets i.e.; training and testing. Training set has 188318 observations and 132 variables including the response variable('loss') ,testing set has 122546 observations and 131 variables and testing set doesn't have the 'loss' variable.

Both training and testing sets have 116 categorical predictor variables, 14 continuous predictor variables.Also the variables doesn't have any names to associate with insurance domain,so it is

considered as **'Anonymous data'**. There exists one more variable 'id' ,which is redundant in model training because it is just an index to the observations.

2.1 Exploring the response variable and log transformation

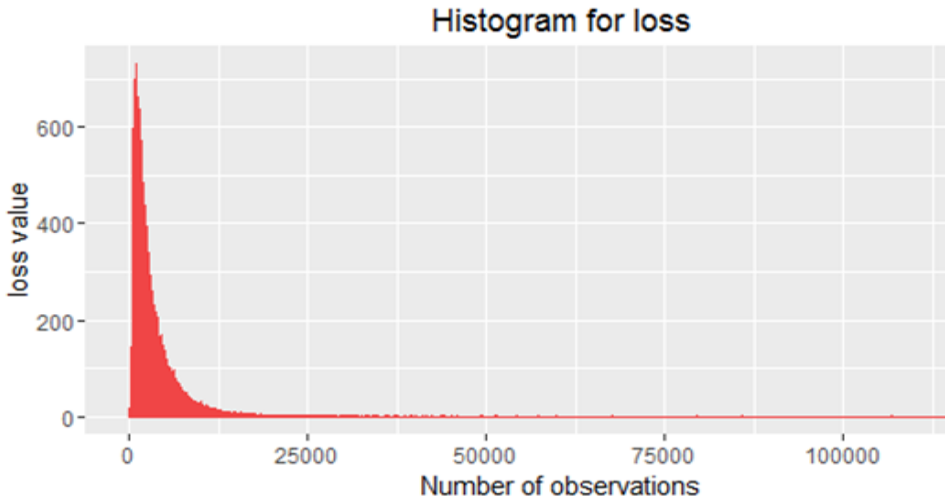


Fig-2 Histogram for loss

In Figure-2, we have the histogram/distributional plot for response ('loss') variable and we observe that 'loss' is right skewed and there seems outliers.In order make 'loss' variable follow normal distribution we apply log trsnaformation on the response variable.

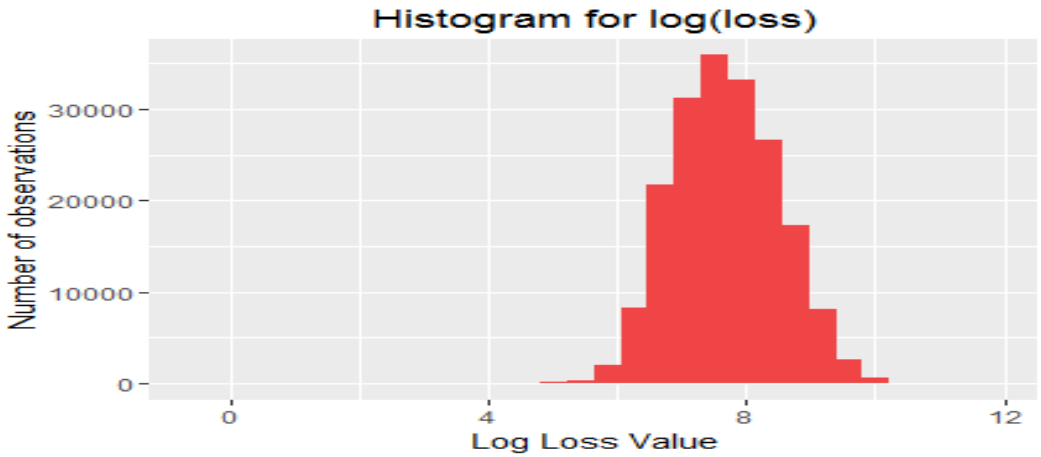


Fig-3 Histogram for log(loss)

From after log transformation 'loss' variable looks 'Normal distributed'. So we consider log(loss) as response variable for all model training purposes. And while making submissions on kaggle we apply exponential transform(inverse of log) on predictions for testing set.

2.2 Exploring Continuous variables in the training dataset

As mentioned earlier we have 14 continuous variables in the training dataset, below figure 5 shows summary of continuous variables .We observe that variables does not have specific name other than just cont1,cont2,.. and all of the continuous variables are min-max scaled in the dataset.

```
'data.frame':   188318 obs. of  14 variables:
 $ cont1 : num  0.726 0.331 0.262 0.322 0.273 ...
 $ cont2 : num  0.246 0.737 0.358 0.556 0.16 ...
 $ cont3 : num  0.188 0.593 0.484 0.528 0.528 ...
 $ cont4 : num  0.79 0.614 0.237 0.374 0.473 ...
 $ cont5 : num  0.31 0.886 0.397 0.422 0.704 ...
 $ cont6 : num  0.718 0.439 0.29 0.441 0.178 ...
 $ cont7 : num  0.335 0.437 0.316 0.391 0.247 ...
 $ cont8 : num  0.303 0.601 0.273 0.318 0.246 ...
 $ cont9 : num  0.671 0.351 0.261 0.321 0.221 ...
 $ cont10: num  0.835 0.439 0.324 0.445 0.212 ...
 $ cont11: num  0.57 0.338 0.381 0.328 0.205 ...
 $ cont12: num  0.595 0.366 0.373 0.322 0.202 ...
 $ cont13: num  0.822 0.611 0.196 0.605 0.246 ...
 $ cont14: num  0.715 0.304 0.774 0.603 0.433 ...
```

Fig-5 summary of continuous variables.

2.3 Exploring Categorical variables in the training dataset

Also we have 116 categorical variables in the training set,and number of levels in the each categorical variables varies .Some of the variables have 2,3,4,5 fewer levels but there exists categorical variables with maximum of 326 levels. Figure 6 shows the sample of categorical variables and different levels are represented by alphabets A,B etc. and when there are more number of levels they are represented with AA,AB..etc.

```
'data.frame':   188318 obs. of  115 variables:
 $ cat1  : Factor w/ 2 levels "A","B": 1 1 1 2 1 1 1 1 1 1 ...
 $ cat2  : Factor w/ 2 levels "A","B": 2 2 2 2 2 2 1 2 2 2 ...
 $ cat3  : Factor w/ 2 levels "A","B": 1 1 1 1 1 1 1 1 2 1 ...
 $ cat4  : Factor w/ 2 levels "A","B": 2 1 1 2 2 1 1 2 2 1 ...
 $ cat5  : Factor w/ 2 levels "A","B": 1 1 2 1 1 1 2 1 2 2 ...
 $ cat6  : Factor w/ 2 levels "A","B": 1 1 1 1 1 1 1 1 1 2 ...
 $ cat7  : Factor w/ 2 levels "A","B": 1 1 1 1 1 1 1 1 1 1 ...
 $ cat8  : Factor w/ 2 levels "A","B": 1 1 1 1 1 1 1 1 1 1 ...
 $ cat9  : Factor w/ 2 levels "A","B": 2 2 2 2 2 2 1 2 2 2 ...
 $ cat10 : Factor w/ 2 levels "A","B": 1 2 2 1 2 1 1 1 2 1 ...
```

Fig-6 Summary of categorical variables.

3) Feature Selection and Engineering

From data exploration , we know that there are 130 predictor variables and close to 190K observations for training the models.Since we have too many predictors to be considered for model training ,so we try to reduce the dimensionality of predictors by using both Unsupervised and supervised machine learning methods as follows.

3.1 Feature engineering using Unsupervised learning

This section is about using unsupervised learning to reduce the dimensions of predictors.Here we used Principal Component Analysis to reduce dimensionality of continuous predictors also we use clustering

method on both continuous,categorical variables with the use of “Gower distance” metric.And the clustering ,PCA results will be used as engineered feature in the future model training.

3.1.1 PCA feature engineering for continuous variables

Since we have 14 continuous predictors for the task, we will first check the correlations among predictor variables and if there exists correlations we will apply principal component analysis on the original features.And the obtained principal components will then be used instead of original features in future model training.

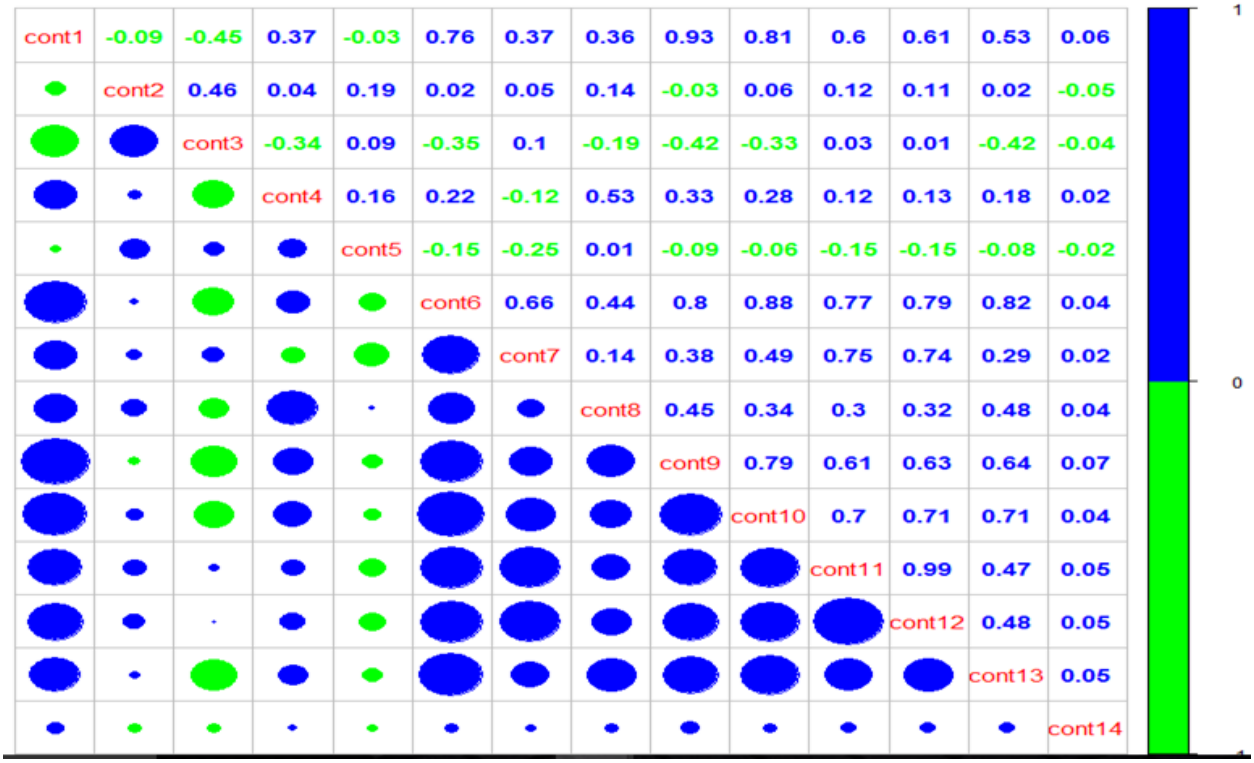


Fig-7 Correlation plot for continuous variables.

In the above figure,we have correlation between all of the continuous variables.In visualization ‘blue’ color represents positive correlation , ‘green’ color represents negative correlation.And the size of the circle represents the value of the correlation.Also we observe the correlation values listed in the upper portion of the plot.So, from the plot we can infer that ‘Cont1’ is correlated with most of the varibales and set of variables such as ‘Cont6’ to ‘Cont13’ are inter correlated with each ohter.We conclude that Principal component analysis (PCA) is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components.

We know that, PCA uses an orthogonal transformation to convert a set of observations that are possibly correlated variables into a set of values of linearly uncorrelated variables called ‘principal components’.And to deal with correlation problem we apply PCA on the original set of continuous variables to obtain principal components. PCA results are shown below.

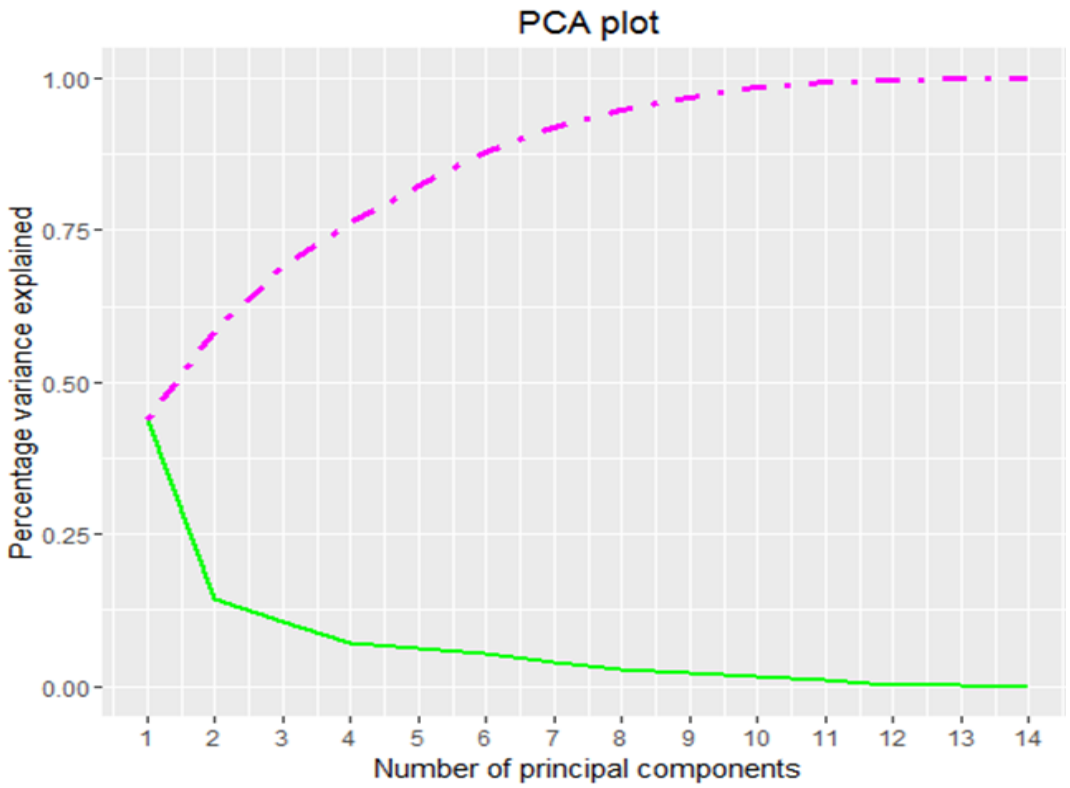


Fig-8: PCA plot for percentage of variance explained.

Figure-8 shows the number of principal components and the percentage of variance explained by the each components represented by 'green' line , cumulative variance explained represented by dotted 'Purple' line. Also, we can observe that first four principal components could explain more than 75 percent of total variation in the original features. So we consider first four principal components in our analysis as best set of continuous predictors and these engineered features will be used in future model training.

3.1.2 Feature engineering based on clustering both continuous, categorical variables

In the previous step, we dealt with continuous predictors but our major challenge is dealing the categorical variables and in this step we will focus on this problem. Here we will try to cluster the observations based on both continuous and categorical predictors for training and testing set together. Since we are using clustering algorithm, we need a distance metric that could handle either continuous or categorical predictors and for this purpose we used 'Gower' distance metric.

Gower distance metric is a similarity measure like Cosine, Jaccard similarity. To calculate the distance measure we no problem with continuous variables but categorical variables needed to be converted into binary or dummy encoded variables. For example, if a categorical variable with two levels 'A', 'B' will be represented by 0, 1 like below

Cat1	Cat1
"A"	"0"
"B"	"1"
"A"	"0"

and if we have more than 2 levels additional dummy variables are created.

We are familiar with the jaccard similarity for binary variables is defined as $\frac{a}{a+b+c}$ here a.b,c,d are defined by using the picture

	cat2B	
cat1B	0	1
0	d	b
1	c	a

[Both 1 = a (similarity),Both 0 =d(dissimilarity)]

Let us assume we have two categorical variables Cat1,Cat2 and the dummy encoded variables are represented by cat1B, cat2B. Here 'a' represents similarity between two variables which means both should have '1' and 'd' represents dissimilarity measure between two variables where both should have 'a', 'b' and 'c' are defined as cross over between the two dummy variables.Similar to Jaccard measure, we can have Gower distance metric as $\frac{2a}{2a+b+c}$ so we put more weight on similarity of variables.

Using the Gower distance metric, we can calculate distance between all of the observations in the dataset. Once we have the gower distance matrix which is of size (n*n where n=no.of observations in a dataset) we try to cluster the observations based on K-means clustering.

Choosing the value for K is very subjective and considering the data size we decided to choose k=3, we tried 4, 5 but results with 3 clusters seems reasonable. So by the end of clustering step we will group all our observations into three different clusters which means every observation is assigned with cluster number 1, 2 or 3. This newly engineered cluster assignment feature will be used as a predictor variable in future model training. Below we have clustering visualization and for ease of understanding we plotted only 5000 sampled instances.

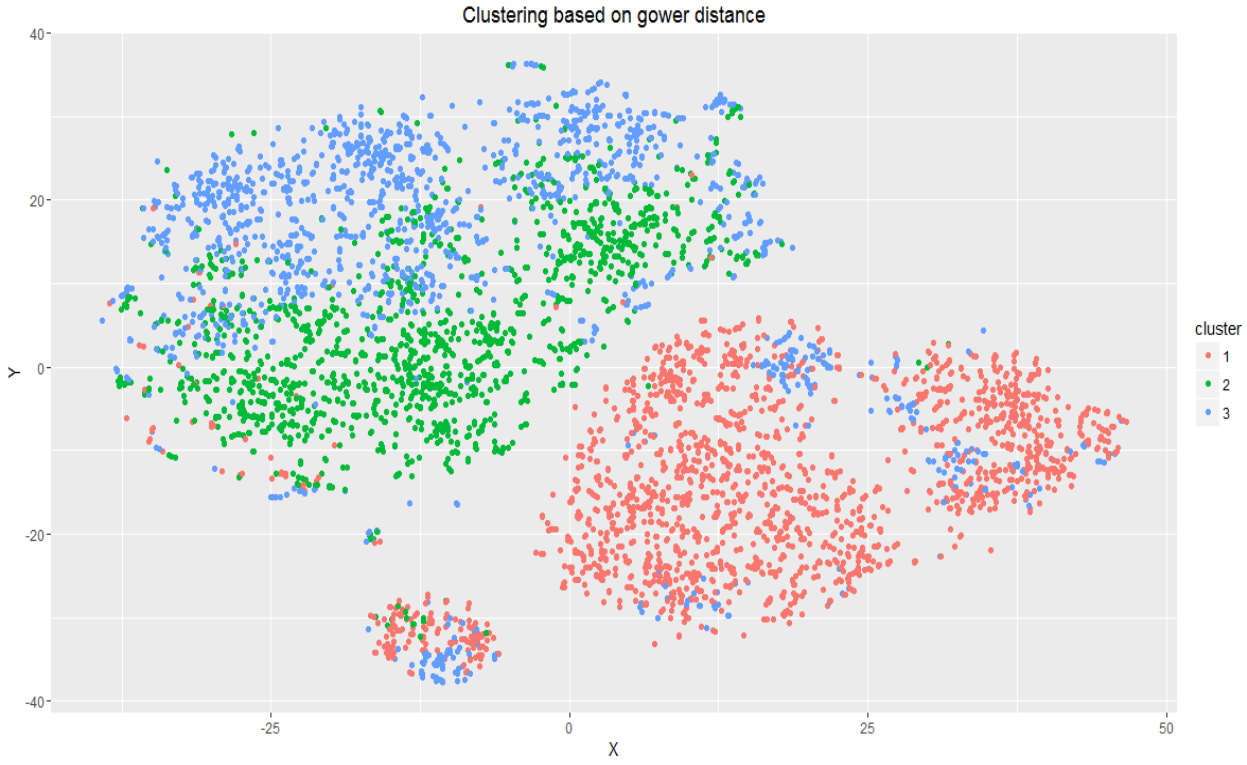


Fig-10: clustering based on gower distance.

Figure 10 shows how cluster assignment is done and we could see that three clusters are indicated by three different colors also there is separation between different clusters.

NOTE: Visualization of clustering was performed in 2-D scatterplot but we know that we have several hundreds of dummy categorical variables. So we may wonder how several hundred dimensions were taken care, for this purpose we used handy neighbourhood embedding method called 't-SNE' for plotting 2-D graph and more details on t-SNE can be found in the appendix section.

3.2 Feature selection using supervised learning models

Earlier we discussed unsupervised way of feature engineering ,now we consider using supervised learning to reduce the dimensions of predictors. Here we will try Sliced Inverse Regression, LASSO and Random Forest three methods to reduce dimensionality of both categorical and continuous predictors.

3.2.1 Feature selection using Sliced Inverse Regression

Besides PCA, we can also consider Sliced inverse regression to reduced the dimension of numeric variables. Unlike PCA, SIR search the directions in the space spanned by numeric variables by the variance of dependent variable Y. The following plot is the percentage variance of covariance matrix

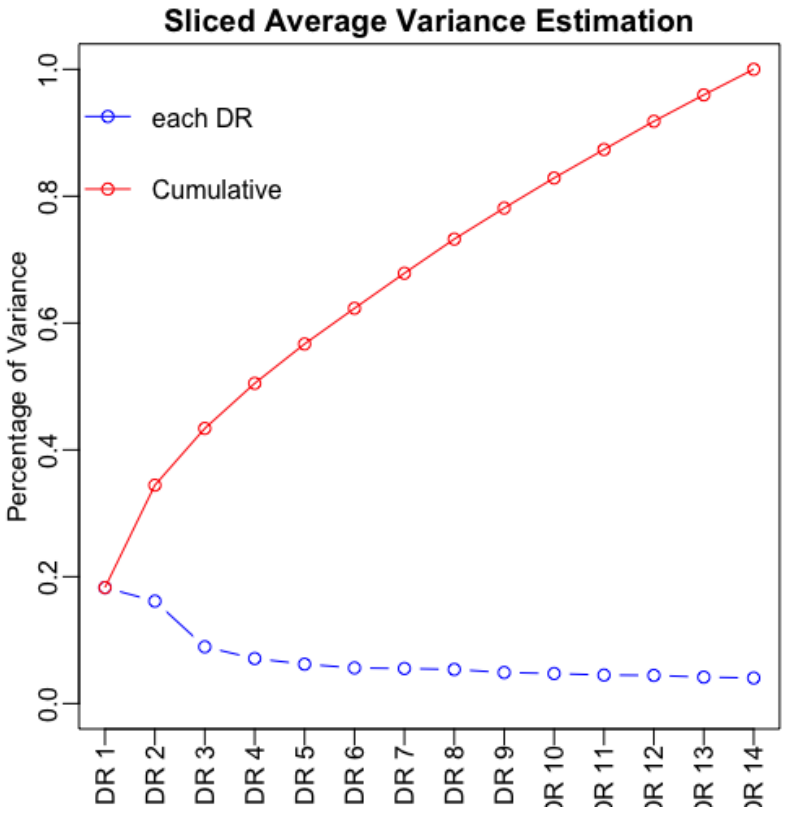


Fig-11 : SIR directions VS Percentage of variance explained plot

explained by its eigenvector (DR) according to the variance of Y. From the plot we see that about 60% variance explained by the first 5 directions. Thus we will use $X * DR1, X * DR2, X * DR3, X * DR4, X * DR5$

as 5 numeric features in the later predict model. ($X$is numeric samples matrix)

3.2.2 Feature selection using LASSO models

LASSO can help us do both model fitting and feature selection by shrinking the coefficients at the same time. Here, 5-fold cross validation and metric of Mean Absolute Error will be used. LASSO does not work well on the categorical variables with too many levels. So, only categorical variables with less than 10 levels and all continuous variables are being included in our model.

We consider using cross validation MAE to choose best lambda(shrinkage parameter). The tendency tells us the less the lambda is, the less the corresponding cross validation MAE is, which also means that the model is more satisfactory. Here, our main goal is to select features which will be used in the following sections.

NOTE: All of the categorical variables are converted as dummy variables by means of sparse matrix representation. In this case, if any one or more than one levels of a categorical variable are selected by our LASSO model, we consider that categorical variable irrespective of levels as important/significant.. For example, we notice that both variable levels cat79B and cat79D will be selected by LASSO while the other levels are shrinked.It makes senese to keep all of the possible levels and cat79 will be considered in best feature selection.

At the end ,we consider following variables to be in best feature set based on the LASSO method for the tuned shrinkage parameter (lambda=0.045) and variables are listed below

Categorical variables selected by using LASSO:

```
"cat1"  "cat2"  "cat4"  "cat5"  "cat6"  "cat7"  "cat9"  "cat10" "cat11" "cat12" "cat13"
"cat23" "cat25" "cat26" "cat27" "cat36" "cat37" "cat38" "cat44" "cat49" "cat52" "cat53"
"cat57" "cat71" "cat72" "cat73" "cat75" "cat76" "cat79" "cat80" "cat81" "cat82" "cat83"
"cat87" "cat95" "cat98"
```

Continuous variables selected by using LASSO:

```
"cont2"  "cont3"  "cont7"  "cont12" "cont14"
```

3.2.3 Feature selection using Trees methods

We know that several tree models can be used for feature/variable selection based on importance of the variables. Here we choose Random Forest and gradient-boosting tree for the feature selection and both of these methods measure the importance of variables in a similar way. However, random forest will calculate the change of prediction accuracy in out-of -bag samples when they are permuted and GBM uses gradient learning rate.

In this step to make our feature selection process more robust, we will make use of several RF, GBM models using different set of tuning parameters. For example we used 20 different RF,GBM models separately and tuning parameters considered using different levels of complexity like ntrees=100+i*50 (i is from 1 to 20), it means to say first RF model will have 150 trees and the 20th RF model will have 1000 trees similarly max_depth parameter is represented by max_depth = 5 + ceiling(i/2) i.e.; max-depth takes values in the range [6 to 15].

The purpose of training several models is that we will obtain feature importances in different settings of tuning parameters. So we compute feature importances from both simple models and more complex

models. After obtaining importance scores from each RF/GBM model we will calculate the average importance to rank features according to their importance. Intuitively, if one variable is important in the majority of 20 models, then we should consider to contain it in the final model. The following 2 plots are the variables relative importance rank obtained by random forest and gradient boosting tree.

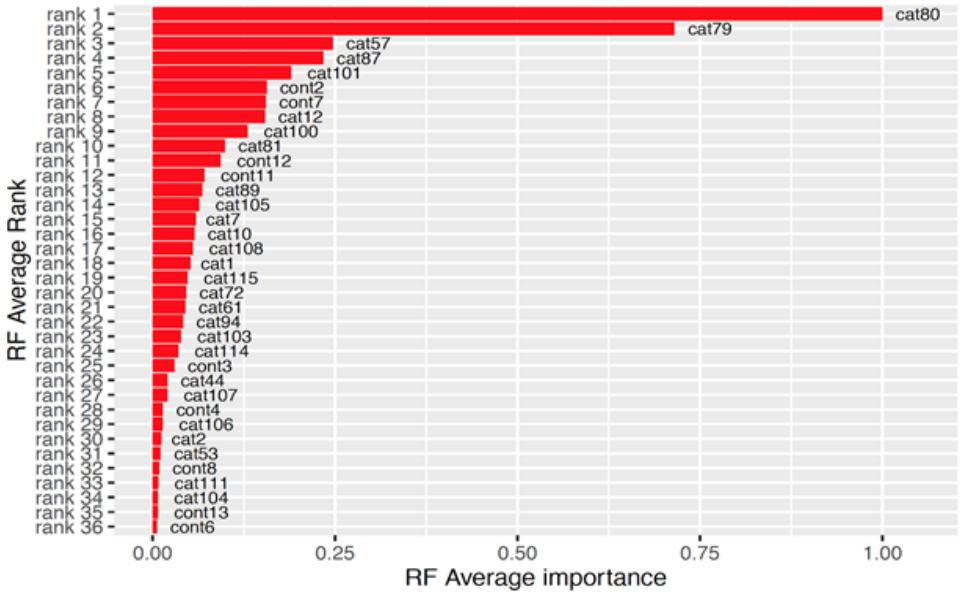


Fig 12: RF models Average Importance.

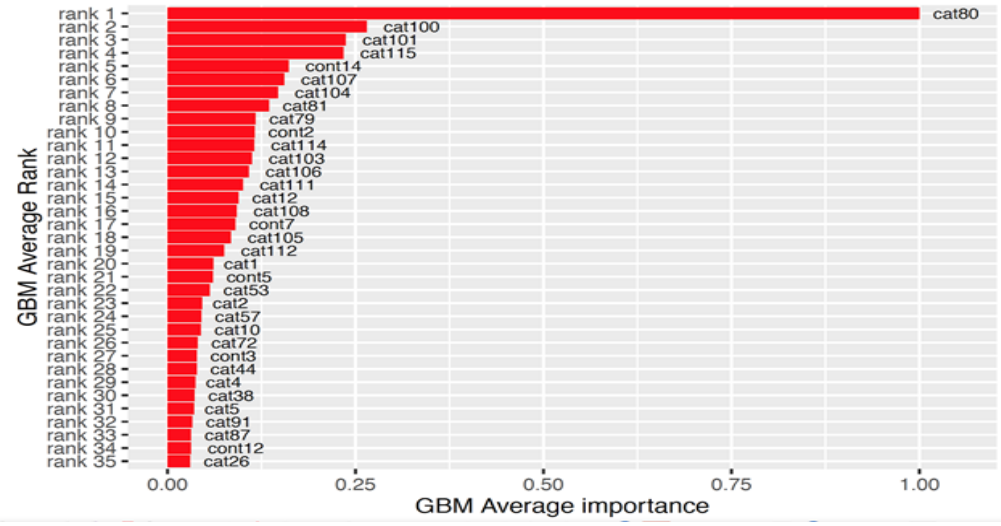


Fig 13: GBM models Average Importance.

From above figures, we observe that some variables like cat80 and cat79 are very important irrespective of complexity of RF/GBM models. By combining the results from above two types of model, we consider following ranked variables as the variables set selected by tree method.

Categorical variables selected by tree based methods:

```
"cat1"   "cat2"   "cat7"   "cat10"  "cat12"  "cat44"  "cat53"  "cat57"  "cat61"  "cat72"
"cat79"  "cat80"  "cat81"  "cat87"  "cat89"  "cat94"  "cat100" "cat101" "cat103" "cat104"
"cat105" "cat106" "cat107" "cat108" "cat111" "cat114" "cat115"
```

Continuous variables selected by tree based methods:

```
"cont2"  "cont3"  "cont4"  "cont6"  "cont7"  "cont8"  "cont11" "cont12" "cont13"
```

Later we will combine part of these variables with other features like principal components and feature obtained by Sliced inverse regression above to train models in next step.

4)Model Training

In this step, we will train several models such as Random Forest, LASSO and Xgboost with different set of best features. And details are as follows,

4.1 Best feature sets from Feature selection and engineering

This section deals with creating different best set of features. We consider 6 different best set features, and methodology is described below. All of the predictors are obtained from the previous feature selection part.

1. set1: All 36 categorical and 5 numerical variables from Lasso, and cluster feature
2. set2: Top 27 important categorical and top 9 continuous variables from tree model, and cluster feature
3. set3: All 36 categorical variables from LASSO, top 4 principle components, and cluster feature
4. set4: Top 27 important categorical variables from tree model , top 4 principle components and cluster feature
5. set5: All 36 categorical variables from LASSO, 5 numeric features corresponding to the 5 most significant direction from SAVE, and cluster feature
6. set6: Top 27 important categorical variables from tree model, 5 numeric from SAVE, and cluster feature.

We use these six different datasets for model training, and we expect these different set of features will help us to obtain different models with best performance and less correlated with each other. So that we could get some benefit in the final stage which is ensemble of models.

4.2 Model training using LASSO with best feature sets

Unlike the previous part, LASSO model is used for model fitting and prediction instead of only feature selection which has been done in the previous part.

Even though the data set is really large, LASSO model can be performed very fast, which means it is possible for us to try to find out the best possible lambda value based on some methods. Here, parameter lambda ranging from 0.001 to 0.03 (30 values in total) will be tuned and cross-validation MAE will be considered as the only metric here to find out our best lambda. We need to notice that here we will not choose and try those lambda values which are greater than 0.03. It is because that our predictors will be shrunk more obviously as lambdas going greater. We have already lost some information in the previous part. It is not a very good idea to keep shrinking more predictors here.

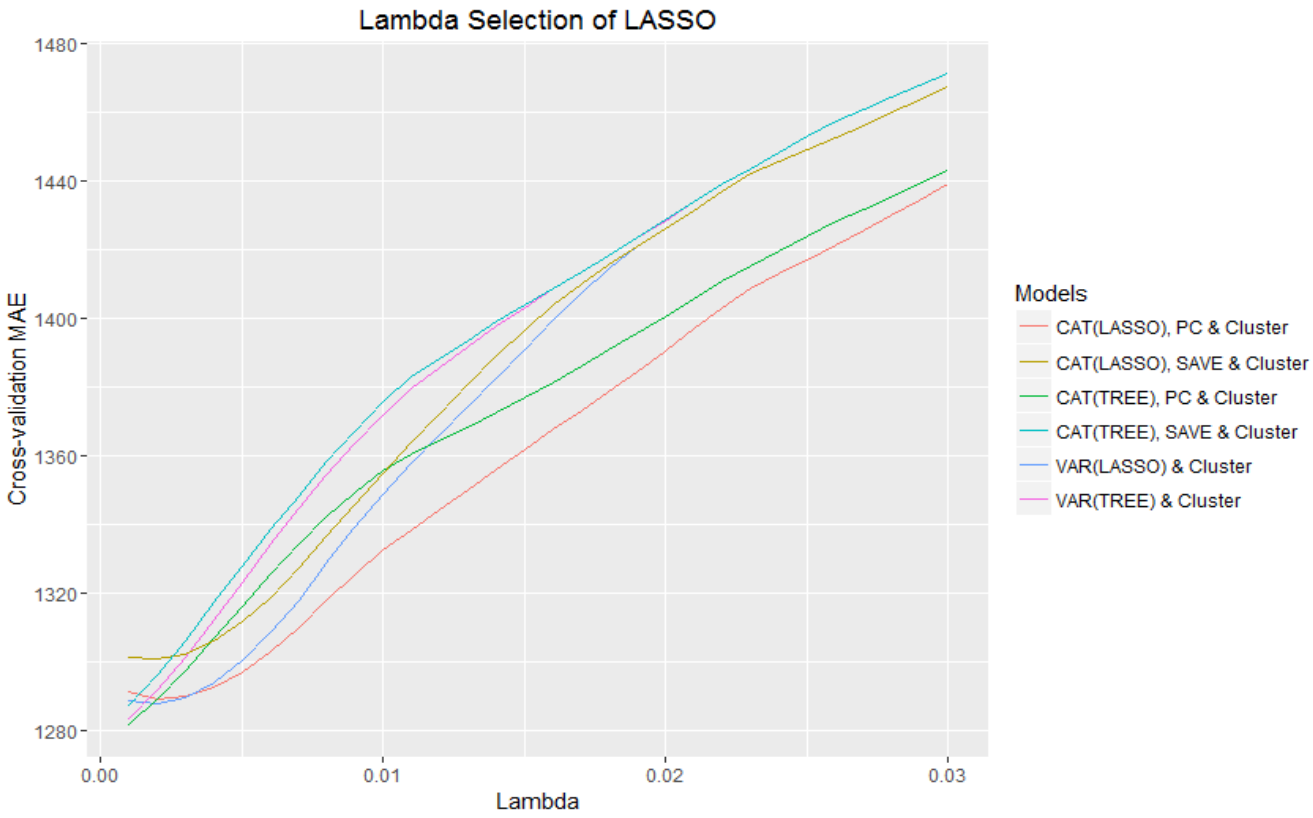


Fig 14: LASSO models

The results of LASSO from different groups of predictors are shown above. We notice that the range of cross-validation MAE for these attempts are from 1280 to 1480 which are basically the same as those values we got from the previous part where we did the feature selection.

We also can get that in these four models, all chosen lambdas are very close to 0, which illustrates that the new models do not be shrunk or just be shrunk only a little bit more, which could be totally ignored. The results we got here are very satisfactory because they did not cause the problems which we were worried about in the previous part.

Then, we will re-fit these four models with their corresponding chosen lambdas based on the whole training data set. The results are listed in the table below. The method number means the corresponding training data set which was used and could be found in part 4.1. Training and Testing MAE will be calculated and performed here.

Model Number	Chosen Lambda	Training MAE	Testing MAE
1	0.002	1288.773	1281.73779
2	0.002	1287.089	1280.07399
3	0.001	1282.538	1279.85560

4	0.001	1280.697	1277.40600
5	0.002	1300.179	1293.38575
6	0.001	1286.542	1284.08014

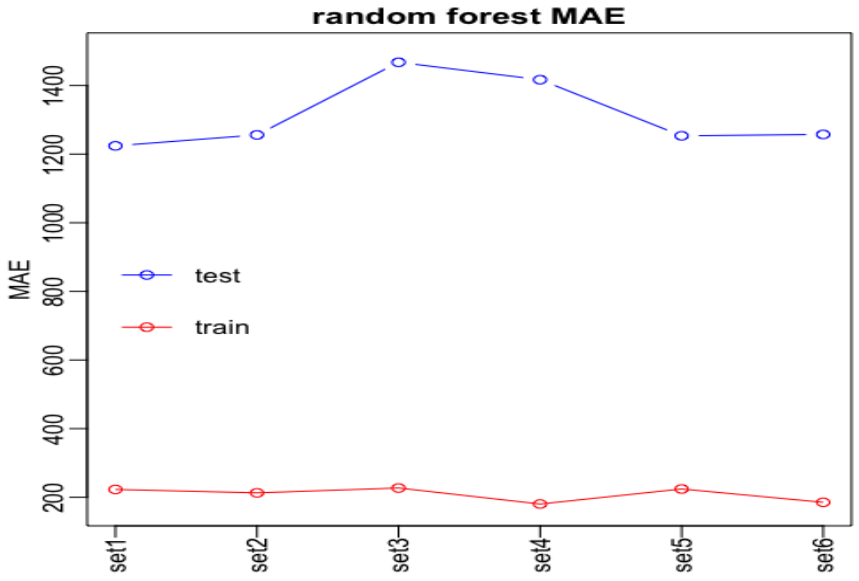
We can see that our Testing MAE are very close to training MAE, and almost all the metric MAE values are around 1280. The training and testing MAE for the fifth model is a little greater than the rest corresponding MAEs, but we still assume the values are tolerable. We think that LASSO model gives us several not bad results, and there is no sign of over- or under-fitting which can be observed from the table above.

4.3 Model training using random forest with best feature sets

In this section, Random Forest model is used for model fitting and prediction. Same as the previous part, we will try to fit models based on those six different training sets we have mentioned at the beginning of this section.

Unlike the previous part 4.2, the Random Forest model takes way much longer time than the model of LASSO, which means that it is impossible for us to tune those necessary parameters and find out the best one with the lowest MAE. If we really choose to do that, the cost will be much greater than the benefit we could get. But to improve the performance will still be our ultimate goal. In order to make it come true, model ensemble will be considered and discussed with detailed information in the next section.

We fit one groups of random forest models using the 6 variables sets which we have mentioned above separately. Here, parameters node size and mtry are fixed as 5 and ⅓ of the predictors numbers, respectively. At the same time, number of trees will be 200 . Here, we only try this very small tuning grid



because we concentrate ourselves on ensemble model instead of the individual model only. The training

Fig 15: random forest MAE.

and testing MAE of rf on 6 variables sets are shown in plot. It can be shown that the random forest suffers overfitting problem. This may result from the number of trees is redundant.

4.4 Model training using xgboost with best feature sets

XGBoost is an algorithm of tree model which is dominating machine learning currently . It is an implementation of gradient boosted decision trees designed for speed and performance.

We fit 2 xgboost model using 6 variables sets separately. The parameter max_depth is set to be a fixed value 10. At the same time, we will try a large number of trees with small learning rate and a small number of trees with large learning rate. In this case, they are set to be 600 and 0.01, 200 and 0.5, respectively. Same as the previous part, a large number of tuning parameters will not be tried, because we do not focus on individual model too much in this case. What we would like to do is to find out whether model ensemble can improve the performance of our final prediction significantly.

Intuitively, the Xgb1 will perform better on training data because the large iteration number and small learning rate and perform worse on test data because overfitting. However, the result is inverse. Xgb1 performs better on test data and worse on train data. The train and test error of Xgb1 and Xgb2 on 6 variables sets are shown following:

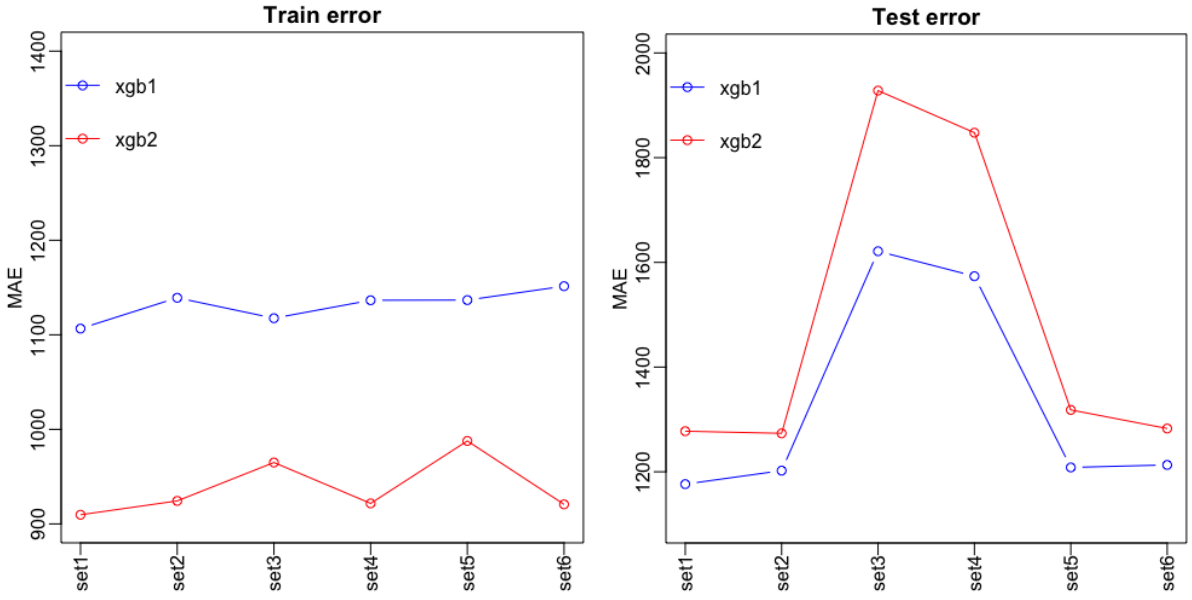


Fig 16: Xgboost Training and test MAE

We can see that, the second with small number of trees and large learning rate performs better on training set. However, their testing performances are not very satisfactory. Particularly, one testing MAE of the third model almost reaches 2000 which is not good at all.

5 Ensemble of models

Ensemble of models or ensemble learning, use multiple learning algorithms to obtain better predictive performance than those obtained from any of the individual learning algorithms alone. A machine

learning ensemble refers to only a concrete and finite set of alternative models. At the same time, it allows the flexible structure to exist among the alternatives. Typical ensemble can be a simple average or a weighted average of different model predictions from individual learners, and this averaging can account for "bias-correction" to achieve much better performing model.

5.1 correlation between models

Here, we focus on finding correlation between several models we trained in the previous step, we know that we have 24 different models using different combinations of algorithms and best-feature sets. correlation plot of 24 predictions from above model on the training data.Lasso model predictions: $\widehat{Y_1}, \ldots \widehat{Y_6}$ according to variables set1 to set6,Random Forest predictions: $\widehat{Y_7}, \ldots \widehat{Y_{10}}$ according to variables set1 to set4,Xgboost 1 predictions: $\widehat{Y_{11}}, \ldots \widehat{Y_{16}}$ according to variables set1 to set6,Xgboost 2 predictions: $\widehat{Y_{17}}, \ldots \widehat{Y_{22}}$ according to variables set1 to set6.

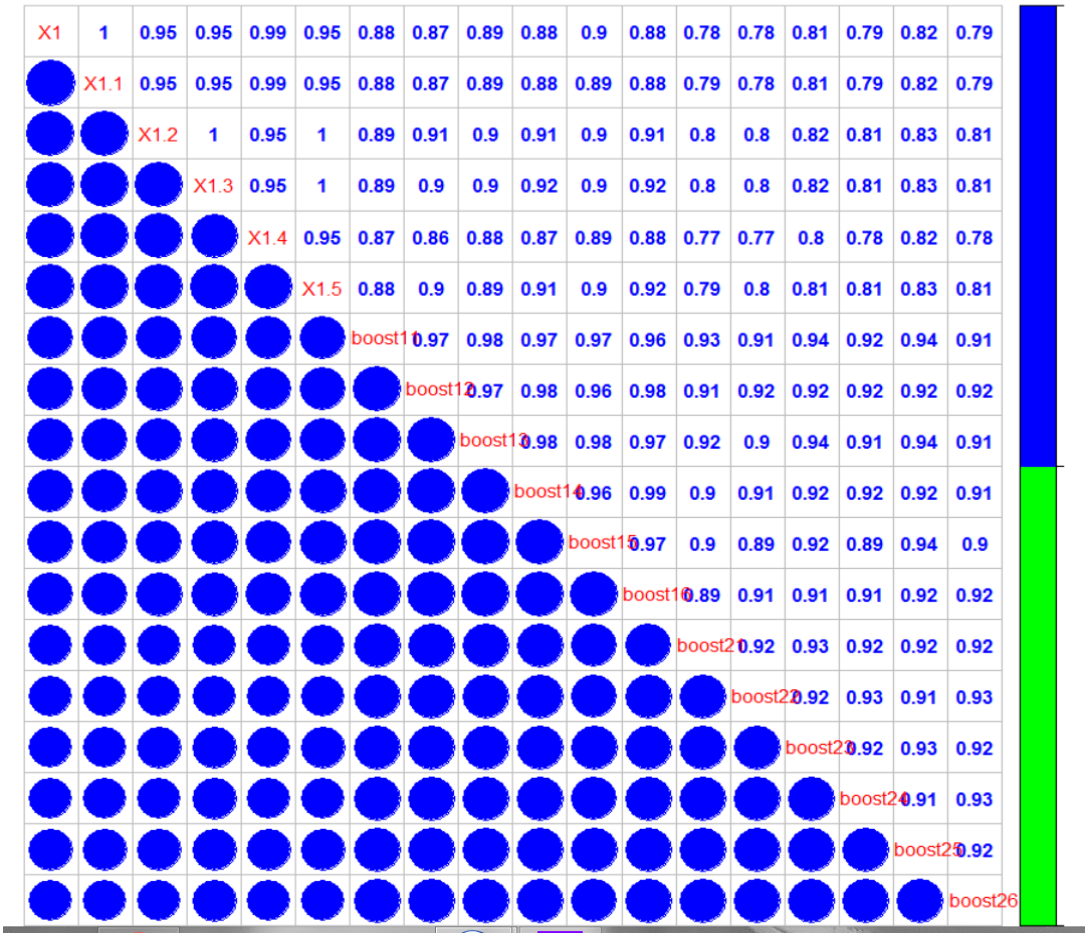


Fig 17: correlation between predictions

In ensembling, we look for best performing model with least correlation between them to improve your model performance and reduce the MAE.But from above correlation plot we have many correlated models so we have to see how ensemble works.

5.2 Simple average

At first we focus on the simple average methods, arithmetic mean. The arithmetic mean is defined as being equal to the sum of the numerical values of each observation divided by the total number of observations which means equal weightage is given to all models.The table below tells us the training and testing MAEs of different models

Test MAE Simple avg ensemble: **1221.62**

5.3 Weighted average using LASSO regression.

Then we focus on the weighted average methods, And weighted average is defined as $\bar{x} = \frac{\sum_{i=1}^{n} w_i x_i}{\sum_{i=1}^{n} w_i}$.

So using this method each model will be assigned different weights ,and here consider LASSO regression to determine the weights. In this LASSO regression we know that some of the predictor variables will be shrunk and for other predictor variables coefficients (here coefficients will represent the corresponding weights) will be non zero.And we use the same method such that all our 24 model predictions on training set as predictor variables, 'loss' as response variable in the LASSO model.

This LASSO model can be treated as meta learning model, and once we obtain testing set predictions for all 24 models we can use LASSO model to predict the 'loss' on the testing set. As the final step, we submit testing set predictions to kaggle to check model performance. And results are below.

Test MAE LASSO ensemble: **1228.44**

It can be shown in the test data, simple average is slightly better than the lasso ensemble of weighted is average.This is because ensemble of models has correlated models.

6) Conclusions:

For this project, we used many statistical learning techniques from class and other states of the art implementations. And the major issue we encountered was relative large dimensions of data, so we first need to modify these features by keeping most part of the information in relatively small number of predictors. In this stage, we take advantage of both supervised learning methods like SAVE, random forest and unsupervised learning methods like PCA and clustering. It turns out the feature engineering actually works well.

The performance of untuned model built on these modified features are not faraway from the best model in Kaggle. After finishing features engineering, instead of building solo complex model, we choose the model ensemble strategy. The ensembled model (meta learning) will improve performance prediction and reduce MAE. This is verified by our results, the ensembled models typically have smaller MAE on test data. However, this promotion is not very significant especially compared to highly increased time consuming. Thus in some situation it is reasonable not to choose model ensemble strategy if we take a balance between prediction accuracy and cost for computation.

7) Appendices:

7.1 t-SNE

The t-distributed stochastic neighbor embedding (t-SNE) is an algorithm which is used to help us reduce the dimensionality. It is a nonlinear dimensionality reduction technique that is particularly well-suited for embedding high-dimensional data into a space of two or three dimensions so that they can be visualized in a scatter plot to give us a clearer understanding. Particularly, it models each high-dimensional object by a two- or three-dimensional point in such a way that similar objects are modeled by nearby points and dissimilar objects are modeled by distant points.

The t-SNE algorithm comprises two basic steps. First, t-SNE constructs a probability distribution over pairs of high-dimensional objects in such a way that similar objects have a high probability of being picked while dissimilar points have a relatively small probability of being chosen. Second, t-SNE defines a similar probability distribution over the points in the low-dimensional map, and it minimizes the Kullback–Leibler divergence between the two distributions with respect to the locations of the points in the map. More details can be found in the links provided.

7.2 Xgboost

Xgboost which started as a research project by Tianqi Chen originally is an open-source software library which provides the Gradient boosting framework which is supported by R and some other programming languages and works on Linux, Windows and macOS. Its goal is to offer a “Scalable, Portable and Distributed Gradient Boosting (GBM, GBRT, GBDT) Library”.More details can be found in the links provided.

References:

1. Friedman, J., Hastie, H., & Tibshirani, R. The Elements of Statistical Learning: Data Mining, Inference, and Prediction. (2nd edition) Springer.
2. James, G., Witten, D., Hastie, T., & Tibshirani, R. An introduction to statistical learning. Springer.
3. van der Maaten, L.J.P.; Hinton, G.E. (Nov 2008). "Visualizing High-Dimensional Data Using t-SNE". Journal of Machine Learning Research. 9: 2579–2605.
4. Polikar, R. (2006). "Ensemble based systems in decision making". IEEE Circuits and Systems Magazine. 6 (3): 21–45. doi:10.1109/MCAS.2006.1688199
5. Rokach, L. (2010). "Ensemble-based classifiers". Artificial Intelligence Review. 33 (1-2): 1–39. doi:10.1007/s10462-009-9124-7
6. t-distributed stochastic neighbor embedding. (2016, November 8). In Wikipedia, The Free Encyclopedia. Retrieved 19:58, December 10, 2016, from https://en.wikipedia.org/wiki/T-distributed_stochastic_neighbor_embedding.
7. Ensemble learning. (2016, October 27). In Wikipedia, The Free Encyclopedia. Retrieved 21:32, December 13, 2016, from https://en.wikipedia.org/wiki/Ensemble_learning.
8. Xgboost. (2016, September 29). In Wikipedia, The Free Encyclopedia. Retrieved 16:05, December 12, 2016, from https://en.wikipedia.org/wiki/Xgboost.