PROJECT REPORT

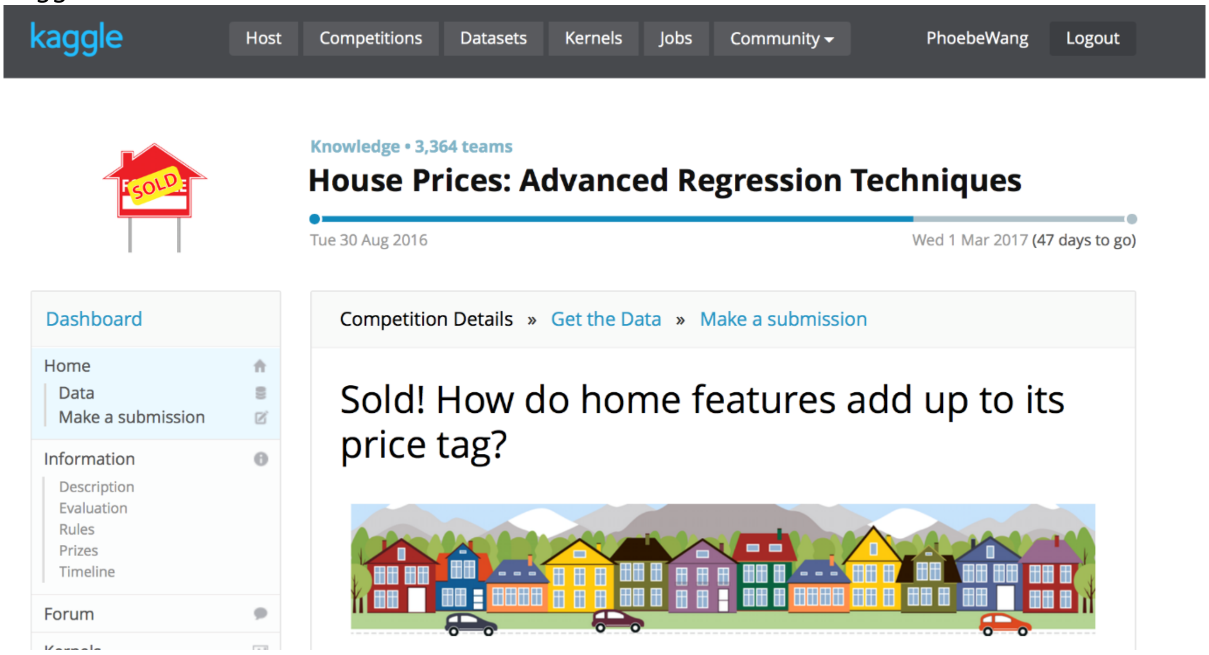
House prices prediction-Kaggle Competition

Yu-Han Chen (陳宥涵)
Sébastien Montella (李胤龍)
Ting-Hsuan Wang (王婷瑄)
Yu-Ching Chen (陳昱瑾)
Group 2
Machine Learning

Contents

I. Introduction

"Living" is our basic need in Maslow's Hierarchy of Needs, therefore is critical to one's life. No matter it's about going to buy a house or simply rent an apartment, a reasonable price is what everyone cares. Generally, the factors affecting the house price are the size of the house, the age of the house, location, building materials, garage etc.
We found our datasets on *Kaggle*, which is a house price prediction competition. We planned to predict the house price using linear regression as a base line and try some other models like neural networks for comparison. At last, we will submit the predict results to *Kaggle*.



House Prices: Advanced Regression Techniques

Sold! How do home features add up to its price tag?

II. Datasets

Kaggle provides 1,460 training data and 1,459 testing data. There are 79 features in the dataset. After understanding the features, we divided these features into three categories: house surroundings (the length of the street, the neighborhood of the house, etc.), the house itself (the building materials, the year that the house was built, etc.), and the selling-related method (the sale condition, the sale type). As for the format of the datasets, there are numerical, categorical, ordinal, and binary.

feature type	amount	feature type	amount
Numerical	37	Ordinal	9
Categorical	31	Binary	2

As we inspected the datasets, we found that there are totally 13,965 NAs, which are approximately 6 percent of our data. We had to cope with the missing values since the training size is small and the number of features are large. Having different thoughts about how to deal with missing values, we decided to do the preprocessing and the prediction separately and compared the results at the end.

III. Our Approach

a) Seb's Method

1. Data Preprocessing

Two main issues needed to be solve. The first one is the missing values in our dataset and the second one is the transformation of categorical data to numerical data because regression package in different languages usually require numerical data format.

i. *Missing Values*

A first analysis was done to determine whether the values marked as NA was true missing value or not. Since some features can't be applied to all examples, it is important to differentiate these NAs to the real missing value which the semantic is close to "value not known". When this first step is done, Seb actually called a package in R, *MICE,* which deals with the missing value by giving those values avoiding sparsity in the data.

ii. *Categorical Data to Numerical*

To overcome the problem of non-numeric data, a simple solution is to create dummies features (features that will have binary value). Of course this method generated about 180 new features, expanding hugely the current dimension (79). Now that all the data are numeric, the model can be applied on the dataset.

iii. *Feature Selection*

The size of the dataset now is over 250 features, which is too much to have small error. A feature selection step can be done in R using the *caret* package. The package orders the features in the dataset in an ascendant order of the influence of the features upon house price. After some tuning, only two subsets was selected: The 150 and 200 most influenced features.

iv. *Data splitting*

2. Model

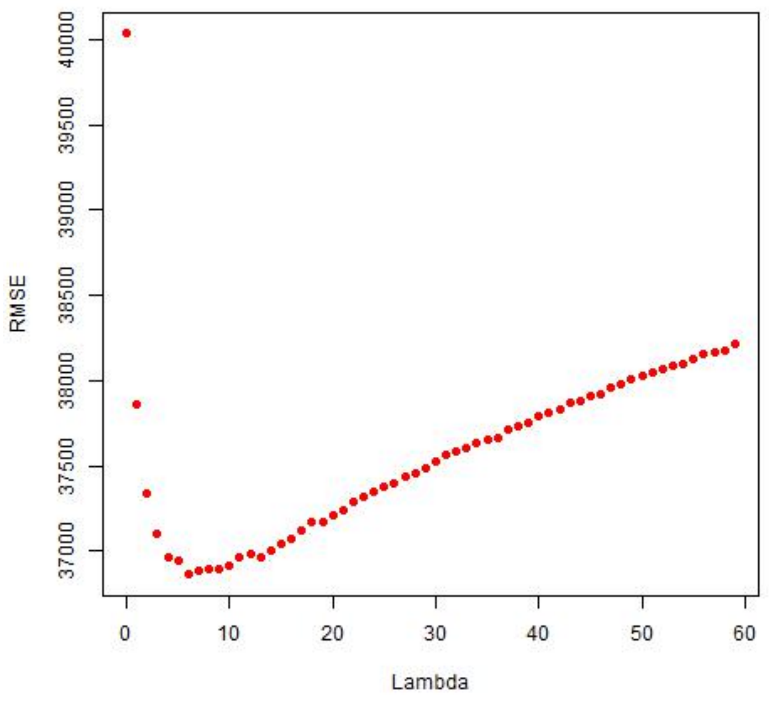
Two linear regression models were trained with the Seb's dataset.

i. *Linear Regression*

A straightforward linear regression was implemented with ease with the *glm* package to determine the houses prices. We tried this model on different dataset size: all features/200 features and 150 features.

ii. *Linear Regression with regularization*

A bit more advanced technique to avoid overfitting is to use regularization. Due to the small size of the dataset and huge number of features, it is important in such case to use a regularization for our model not to over fit our data that is furthermore not of a good quality. After parameter tuning, the best λ found is 7.



3. Result

	All features	200 features	150 features
Linear Regression	41,057.85	37,185.15	37,326.00
Linear Regression + Regularization	36,911.92	36,742.62	44,116.92

Figure 1: RMSE for both linear regression models

We observe after testing our models that the smallest error with Seb's dataset is obtained with 200 features using linear regression with regularization. Indeed, the error reaches about 37,000 dollars, which is a really bad performance to predict houses prices. Let's see other data preprocessing method and models to see if the performance can be improved.

b) Phoebe's Method

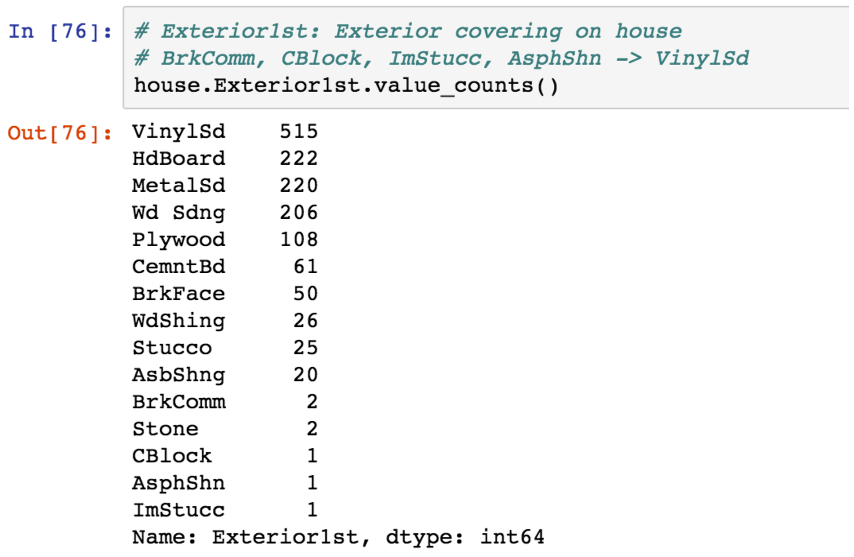
1. Data Preprocessing

i. ***Missing values***

There are two kinds of missing values. One represents "true missing value", and the other simply represents "do not have that feature". For the true missing values, a largest or the most frequent value of that feature is used to replace them. As for the "None" values, the word "None" is used to represent them. For instance, in the "garage" feature, the missing value means that the house does not have a garage. The missing values are then replaced the missing value with "None".

ii. ***Categorical and ordinal data***

Dummies variables are created when there's categorical data. However, there might be a lot of dummies variables. To decrease the number of dummies variables, a value with frequency less than 10 is replaced by the most frequent value of that feature. For instance, in the picture below, we can find that the number of some types in the feature "Exterior1st" is really small. Phoebe then change "Brkcomm", "Stone", "CBlock", "AsphSnn", "ImStucc" to "VinglSd".



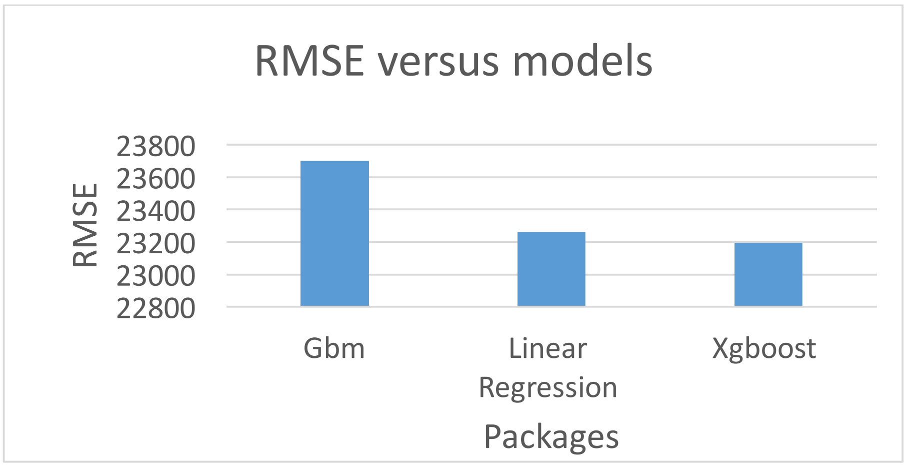
```
In [76]: # Exterior1st: Exterior covering on house
         # BrkComm, CBlock, ImStucc, AsphShn -> VinylSd
         house.Exterior1st.value_counts()

Out[76]: VinylSd    515
         HdBoard    222
         MetalSd    220
         Wd Sdng    206
         Plywood    108
         CemntBd     61
         BrkFace     50
         WdShing     26
         Stucco      25
         AsbShng     20
         BrkComm      2
         Stone        2
         CBlock       1
         AsphShn      1
         ImStucc      1
         Name: Exterior1st, dtype: int64
```

As for the ordinal data (excellent, good, average, etc.), a score of 1 to 5 is used to represent them.

iii. ***Normalization***

Since the quantifier for each feature is different, a normalize process is used in order to train a better model.

2. Model and Result



(GBM: 23701, linear regression: 23262, Xgboost: 23193)

Though the RMSE of these three models are large, Phoebe found an interesting fact that the predicting results of these three are actually quite close.

c) *Julie's Method*

1. Data preprocessing

i. *Missing values*

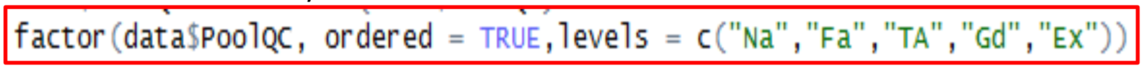
True missing values: Julie uses the package "MICE" in R to deal with them.
Not having certain feature: If a feature is numerical, Julie replaces it with -1. As for the categorical data, Julie uses the word "NULL" to describe it.

ii. *Change the numerical data into more meaningful data*

Some manipulation were done on the age-related numerical data. For instance, when dealing with the year that the house was built, Julie minus the value by the current year. I expected that the younger the house is, the higher the selling price would be.

iii. *Specify the order of ordinal data*

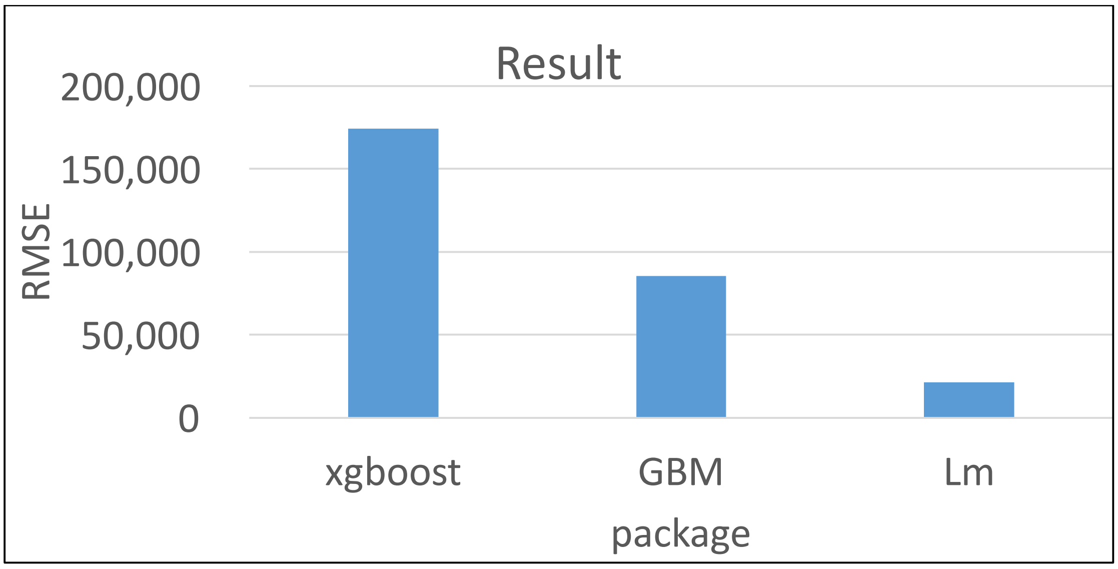
The "factor" in R will turn the ordinal data into numbers. To specify the order of importance, Julie does that manually.



```
factor(data$PoolQC, ordered = TRUE,levels = c("Na","Fa","TA","Gd","Ex"))
```

2. Model & Result

The result of different models are shown in the figure below.
The RMSE of the Xgboost is 174,251, GBM is 85,573, and Lm is 21,396, which performs the best.



d) Joy's Method

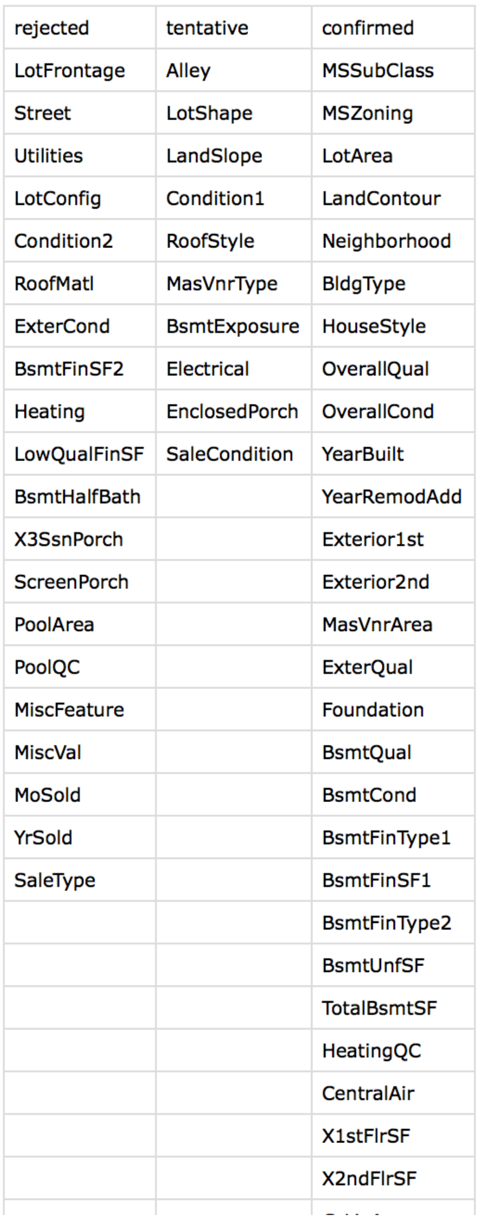
1. Data preprocessing

i. ***<u>Missing values</u>***

Before doing the machine learning process, dealing with the NAs in the datasets is our first priority. All the numerical missing values are substituted by -1, and "Missing" for missing categorical values.

ii. ***<u>Feature selection</u>***

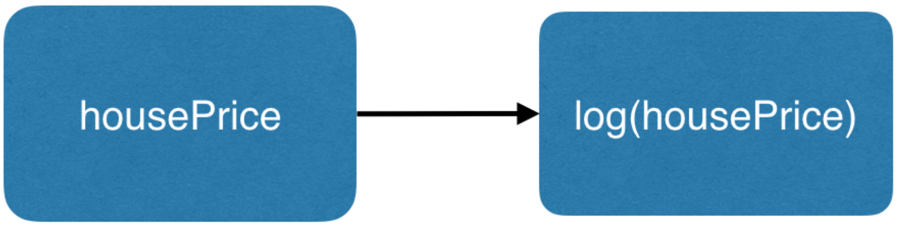
Since the training datasets have only 1460 data with 79 features, feature selection is needed in order to decrease the number of features to get a more accurate result. The package used here is called *"Boruta"* in R. How *Boruta* conducts feature selection is to iteratively compare *Zscores* of attributes with *ZScores* of shadow attributes and split the features into three categorized attributes, which are "Confirmed", "Rejected", and "Tentative" output by *"Boruta".* "Confirmed" represents the attributes that are significantly better than shadows. "Rejected" represents the attributes that have significantly worst importance than shadow ones. The left outs will be categorized to "tentative" attributes.
With the help of the package, 59 confirmed and 9 tentative features are selected.

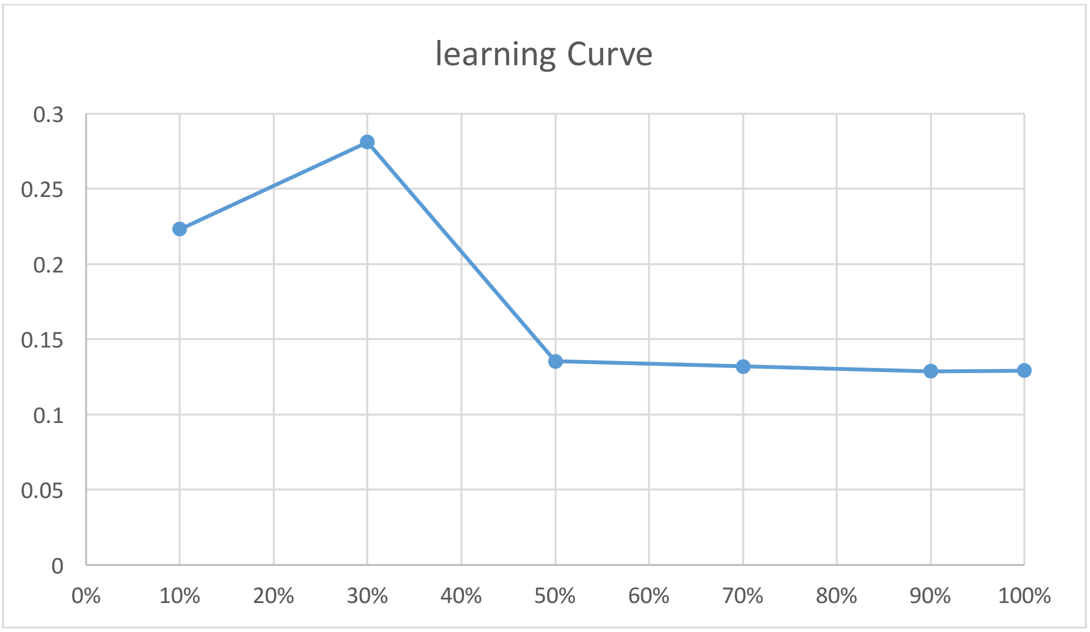


rejected	tentative	confirmed
LotFrontage	Alley	MSSubClass
Street	LotShape	MSZoning
Utilities	LandSlope	LotArea
LotConfig	Condition1	LandContour
Condition2	RoofStyle	Neighborhood
RoofMatl	MasVnrType	BldgType
ExterCond	BsmtExposure	HouseStyle
BsmtFinSF2	Electrical	OverallQual
Heating	EnclosedPorch	OverallCond
LowQualFinSF	SaleCondition	YearBuilt
BsmtHalfBath		YearRemodAdd
X3SsnPorch		Exterior1st
ScreenPorch		Exterior2nd
PoolArea		MasVnrArea
PoolQC		ExterQual
MiscFeature		Foundation
MiscVal		BsmtQual
MoSold		BsmtCond
YrSold		BsmtFinType1
SaleType		BsmtFinSF1
		BsmtFinType2
		BsmtUnfSF
		TotalBsmtSF
		HeatingQC
		CentralAir
		X1stFlrSF
		X2ndFlrSF

iii. *Normalize the target attribute (housePrice)*

Since the variation of the target attribute is too large which will influence the accuracy of the prediction of the model; thus, I took the log of the price in order to normalize it. By doing so, I got a better result with smaller RMSE.





According to the learning curve, it's clear that using the full size performs the best. Therefore, for all the following models, I took the whole dataset for training.

2. Model

i. GBM

GBM is the short of *"gradient boosting machine"*. Gradient boosting is a technique for regression and classification problem, which uses decision tree as a basis. The reason why it is called "gradient boosting" is that it ensembles several weak models algorithmically and use stochastic gradient to iteratively solve the residual.

This model has several characteristics:

- Fast: GBM is relatively faster than other model.
- Competitive with high-end algorithms: GBM uses several algorithms such as random forest.
- Explicitly handles with NA / Scaling or normalization is unnecessary: No need to worry about the NAs or the normalization problem in the data sets, since the package in R do that automatically.
- Handles perfectly correlated independent variables

➔ Parameter tuning

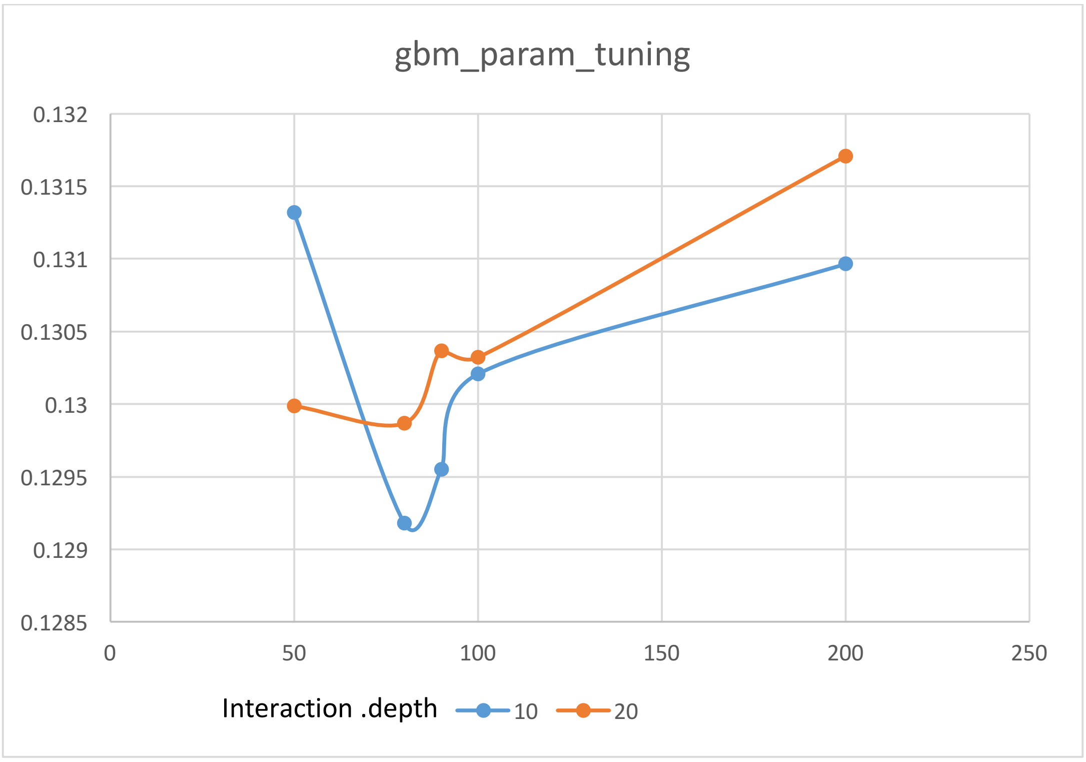
GBM parameters:

n.Trees: Number of trees (the number of gradient boosting iteration).

Interaction.depth: Maximum nodes per tree.

Best parameter combination: *n.trees* = 80, *interaction.depth* = 10.

Validation RMSE: 0.129178

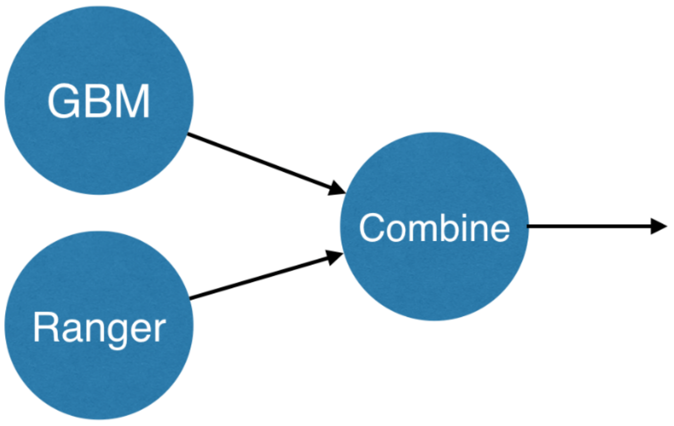


ii. *__Ranger__*

Ranger is a fast implementation of random forest for high dimensional data.
Important parameters:
Mtry: Number of variables randomly sampled as candidates at each split. Note that the default values are different for classification (sqrt(p) where p is the number of variables in x) and regression (p/3).
n.trees: number of trees to grow.
Best parameter combination: *mtry* = p/3, *n.trees* = 500
Validation RMSE: 0.1406389

iii. *__Ensemble__*

The ensemble model is created by taking the output of GBM and Ranger as a neuron to create a neural network.



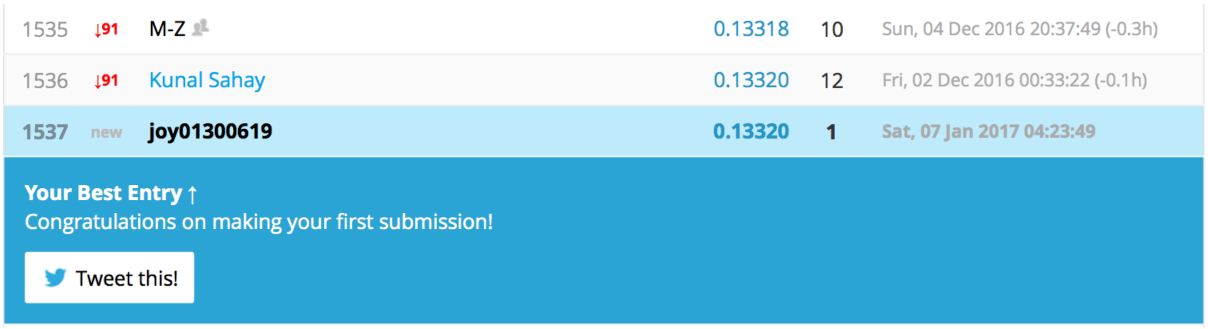
3. Final result

The predictions of different models were submitted to *Kaggle*. The rank and RMSE of different models are listed below:

i. *GBM*

Rank: 1537

RMSE: 0.13320



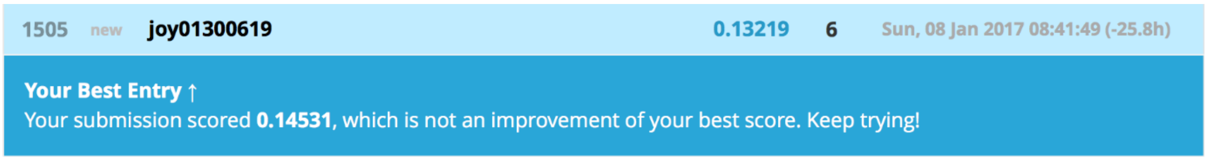
1535	↓91	M-Z	0.13318	10	Sun, 04 Dec 2016 20:37:49 (-0.3h)
1536	↓91	Kunal Sahay	0.13320	12	Fri, 02 Dec 2016 00:33:22 (-0.1h)
1537	new	**joy01300619**	**0.13320**	**1**	**Sat, 07 Jan 2017 04:23:49**

Your Best Entry ↑
Congratulations on making your first submission!

Tweet this!

ii. *Ranger*

RMSE: 0.14531



1505	new	**joy01300619**	**0.13219**	**6**	**Sun, 08 Jan 2017 08:41:49 (-25.8h)**

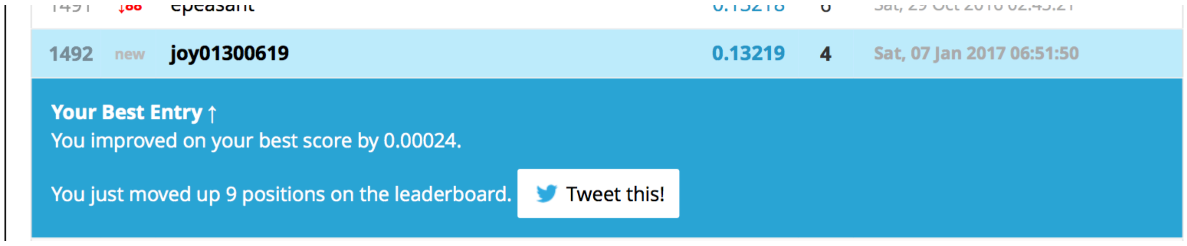
Your Best Entry ↑
Your submission scored **0.14531**, which is not an improvement of your best score. Keep trying!

iii. *Ensemble model*

Rank: 1492

RMSE: 0.13219

The ensemble model performs the best by decreasing the RMSE to 0.13219.



1492	new	**joy01300619**	**0.13219**	**4**	**Sat, 07 Jan 2017 06:51:50**

Your Best Entry ↑
You improved on your best score by 0.00024.

You just moved up 9 positions on the leaderboard. Tweet this!

IV. Conclusion

House price prediction is a well-known machine learning task. In this project, we applied several regression techniques in order to compete with other models trained by other *Kaggle* teams. A significant part for the model to be performant is the amount of the data, and the representation of it. Different preprocessing methods on our dataset were made, and each of them impacted considerably the performance of the model. As an example, our error to predict houses prices fall from 37,000 to 0.13 as we were changing our data preprocessing and our regression techniques. We finally end up by mixing models to reach the lowest error so far. However, our result still need to be improved since one of the best result recorded on Kaggle is 0.03.