THE HONG KONG POLYTECHNIC UNIVERSITY
香港理工大學

COMP5434 Big Data Computing

Topic 2 Book Rating

Name	Student ID
LIU Jingdi	20090712g
HUANG Zheming	20085189g

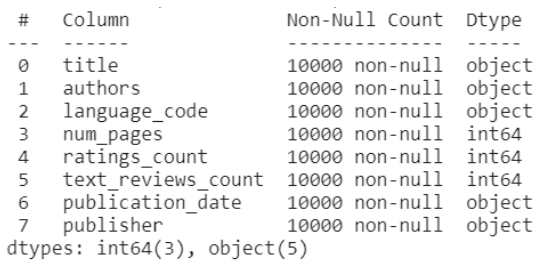
Contents

1. Introduction

The books on google books are always labeled a rating score, which represents its popularity. Predicting the book rating is not a simple work, because many factors may contribute to its popularity. This report will introduce how we deal with the features of a book and make a prediction of its rating score.

2. Data preprocessing

Before data can be used as input for machine learning algorithms, it often must be cleaned, formatted, and restructured — this is typically known as preprocessing. For this dataset, there are no invalid or missing entries we must deal with.

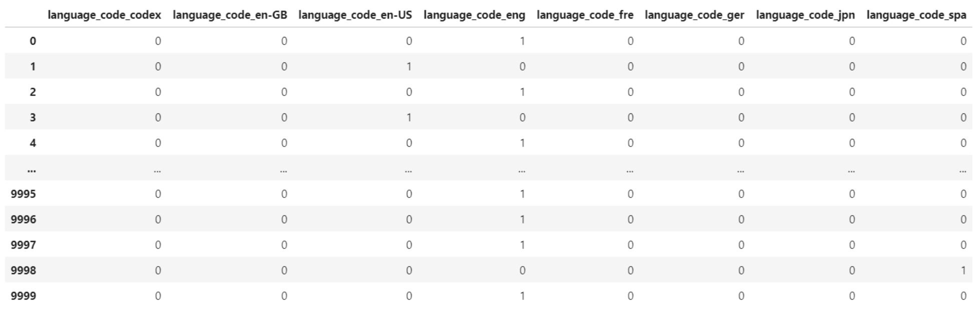


```
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   title               10000 non-null  object
 1   authors             10000 non-null  object
 2   language_code       10000 non-null  object
 3   num_pages           10000 non-null  int64
 4   ratings_count       10000 non-null  int64
 5   text_reviews_count  10000 non-null  int64
 6   publication_date    10000 non-null  object
 7   publisher           10000 non-null  object
dtypes: int64(3), object(5)
```

2.1 Discrete Feature

2.1.1. one-hot

From the above pictures, we can see that not all features are numerical features. For categorical variables, it can be difficult for model to understand, the popular way to deal with them is using one-hot encoding if the variables of text feature are same weighted, such as language_code.

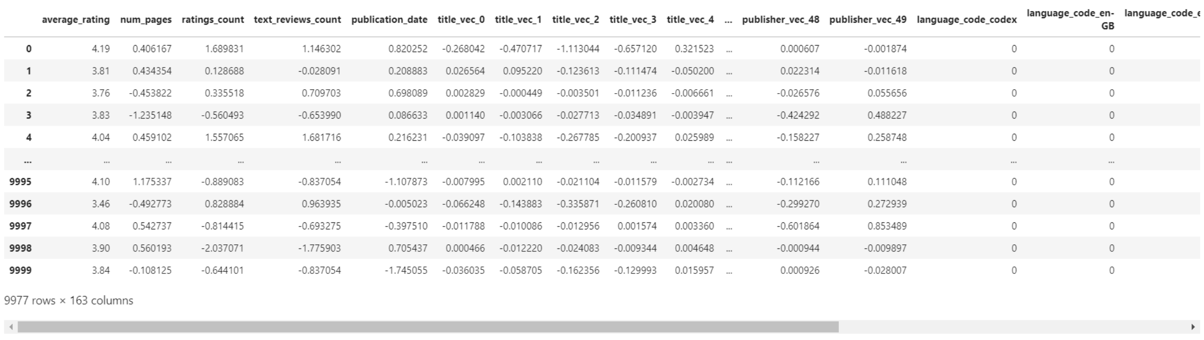


	language_code_codex	language_code_en-GB	language_code_en-US	language_code_eng	language_code_fre	language_code_ger	language_code_jpn	language_code_spa
0	0	0	0	1	0	0	0	0
1	0	0	1	0	0	0	0	0
2	0	0	0	1	0	0	0	0
3	0	0	1	0	0	0	0	0
4	0	0	0	1	0	0	0	0
...	...	...	...	...	...	...	...	...
9995	0	0	0	1	0	0	0	0
9996	0	0	0	1	0	0	0	0
9997	0	0	0	1	0	0	0	0
9998	0	0	0	0	0	0	0	1
9999	0	0	0	1	0	0	0	0

2.1.2. word2vec and tf-idf

But for more complicated text feature like title, we try to deploy word2vec model to describe them in a certain amount of features as we like. The first thing we need to do is splitting sentences into words. And as we learned, more frequent terms in a document are more important, and terms that appear in many different documents are less indicative of overall topic. So we can use if-idf to get the key-words of features(e.g. publishers)

With the help of word2vec and IF-IDF, we can use the vectors to represent the keywords of the text feature. Finally we can use the only 50 extra features to conclude the meaning of one text feature.

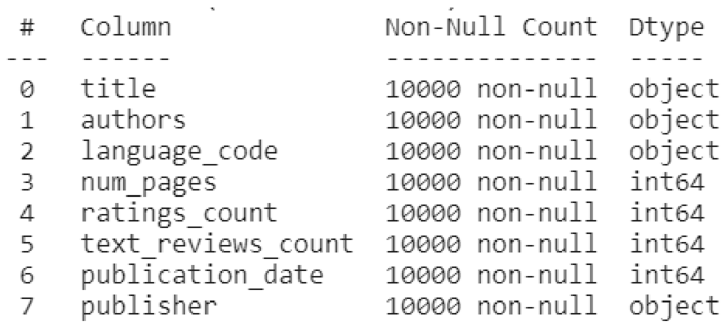


	average_rating	num_pages	ratings_count	text_reviews_count	publication_date	title_vec_0	title_vec_1	title_vec_2	title_vec_3	title_vec_4	...	publisher_vec_48	publisher_vec_49	language_code_codex	language_code_en-GB	language_code_e
0	4.19	0.406167	1.689831	1.146302	0.820252	-0.268042	-0.470717	-1.113044	-0.657120	0.321523	...	0.000607	-0.001874	0	0	
1	3.81	0.434354	0.128688	-0.028091	0.208883	0.026564	0.095220	-0.123613	-0.111474	-0.050200	...	0.022314	-0.011618	0	0	
2	3.76	-0.453822	0.335518	0.709703	0.698089	0.002829	-0.000449	-0.003501	-0.011236	-0.006661	...	-0.026576	0.055656	0	0	
3	3.83	-1.235148	-0.560493	-0.653990	0.086633	0.001140	-0.003066	-0.027713	-0.034891	-0.003947	...	-0.424292	0.488227	0	0	
4	4.04	0.459102	1.557065	1.681716	0.216231	-0.039097	-0.103838	-0.267785	-0.200937	0.025989	...	-0.158227	0.258748	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
9995	4.10	1.175337	-0.889083	-0.837054	-1.107873	-0.007995	0.002110	-0.021104	-0.011579	-0.002734	...	-0.112166	0.111048	0	0	
9996	3.46	-0.492773	0.828884	0.963935	-0.005023	-0.066248	-0.143883	-0.335871	-0.260810	0.020080	...	-0.299270	0.272939	0	0	
9997	4.08	0.542737	-0.814415	-0.693275	-0.397510	-0.011788	-0.010086	-0.012956	0.001574	0.003360	...	-0.601864	0.853489	0	0	
9998	3.90	0.560193	-2.037071	-1.775903	0.705437	0.000466	-0.012220	-0.024083	-0.009344	0.004648	...	-0.000944	-0.009897	0	0	
9999	3.84	-0.108125	-0.644101	-0.837054	-1.745055	-0.036035	-0.058705	-0.162356	-0.129993	0.015957	...	0.000926	-0.028007	0	0	

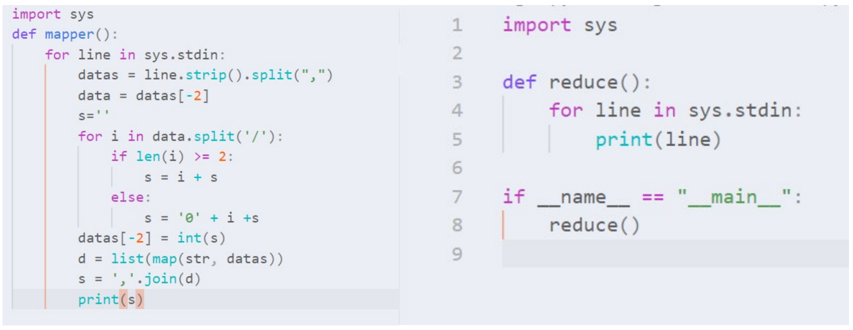
9977 rows × 163 columns

2.1.3. mapreduce

The last thing we are going to deal with is the publication_date, we use mapreduce to convert the datatime from string into int, we just scan each row and then generate the new training dataset.



```
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   title               10000 non-null  object
 1   authors             10000 non-null  object
 2   language_code       10000 non-null  object
 3   num_pages           10000 non-null  int64
 4   ratings_count       10000 non-null  int64
 5   text_reviews_count  10000 non-null  int64
 6   publication_date    10000 non-null  int64
 7   publisher           10000 non-null  object
```



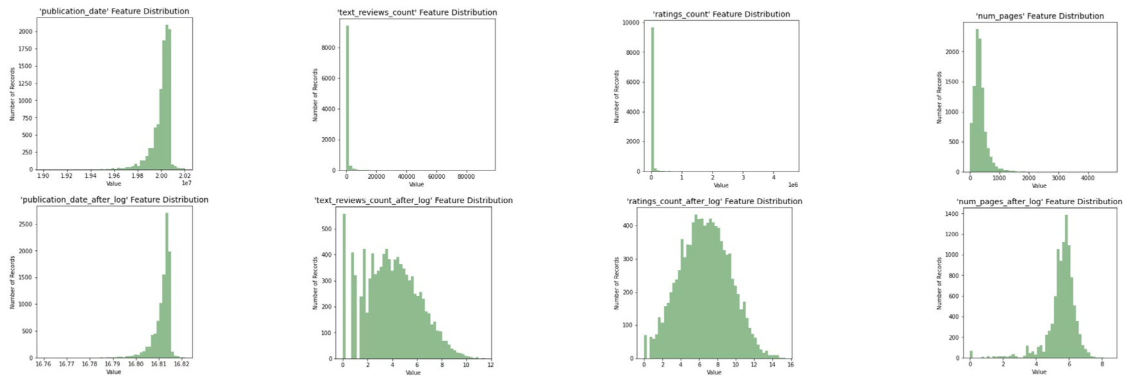
```
import sys
def mapper():
    for line in sys.stdin:
        datas = line.strip().split(",")
        data = datas[-2]
        s=''
        for i in data.split('/'):
            if len(i) >= 2:
                s = i + s
            else:
                s = '0' + i +s
        datas[-2] = int(s)
        d = list(map(str, datas))
        s = ','.join(d)
        print(s)
```

```
import sys

def reduce():
    for line in sys.stdin:
        print(line)

if __name__ == "__main__":
    reduce()
```

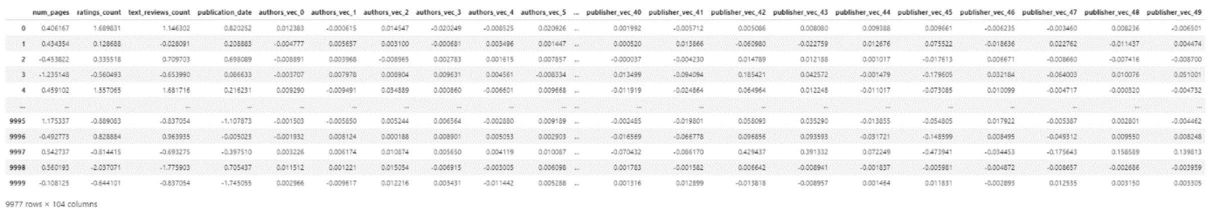
2.2 Numerical feature

Look through the dataset, we found some continuous features may have some very large values, which can be sensitive to the distributions of values and can reduce the performance of each model if it not properly normalized.

For highly-skewed feature distribution such as ‘publication_date’, ‘ratings_count’, ‘num_pages’ and ‘text_reviews_count’, the common practice is using logarithmic transformation on the data so the very large value brings less negative affect on the performance. After using logarithmic transformation, the new features distribution are like this:



We use standardization to normalize the numerical data to accelerate the speed of gradient descent to find the optimal solution. After preprocessing the dataset, the final input dataset is just as the following picture showing. We just drop some rows(we regarded as the ouliers) which rating score is under 2.0. So the final input dataset is 9977 rows and 104 columns.



	num_pages	ratings_count	text_reviews_count	publication_date	authors_vec_0	authors_vec_1	authors_vec_2	authors_vec_3	authors_vec_4	authors_vec_5	...
0	0.406167	1.689831	1.146302	0.820252	0.012383	-0.000615	0.014547	-0.020249	-0.008525	0.020926	...
1	0.434354	0.128688	-0.028091	0.208883	-0.004777	0.005657	0.003100	-0.000681	0.003496	0.001447	...
2	-0.453822	0.335518	0.709703	0.698089	-0.008891	0.003968	-0.008965	0.002783	0.001615	0.007857	...
3	-1.235148	-0.560493	-0.653990	0.086633	-0.003707	0.007978	0.008904	0.009631	0.004561	-0.008334	...
4	0.459102	1.557065	1.681716	0.216231	0.009290	-0.009491	0.034889	0.000860	-0.006601	0.009668	...
...	...	...	...	...	...	...	...	...	...	...	...
9995	1.175337	-0.889083	-0.837054	-1.107873	-0.001503	-0.005850	0.005244	0.006564	-0.002880	0.009189	...
9996	-0.492773	0.828884	0.963935	-0.005023	-0.001932	0.008124	0.000188	0.008901	0.005053	0.002903	...
9997	0.542737	-0.814415	-0.693275	-0.397510	0.003226	0.006174	0.010874	0.005650	0.004119	0.010087	...
9998	0.560193	-2.037071	-1.775903	0.705437	0.011512	0.001221	0.015054	-0.006915	-0.003005	0.006098	...
9999	-0.108125	-0.644101	-0.837054	-1.745055	0.002966	-0.009617	0.012216	0.003431	-0.011442	0.005288	...

	publisher_vec_40	publisher_vec_41	publisher_vec_42	publisher_vec_43	publisher_vec_44	publisher_vec_45	publisher_vec_46	publisher_vec_47	publisher_vec_48	publisher_vec_49
0	0.001992	-0.005712	0.005086	0.008080	0.009388	0.009661	-0.006235	-0.003460	0.008236	-0.006501
1	0.000520	0.013866	-0.060980	-0.022759	0.012676	0.075522	-0.018636	0.022762	-0.011437	0.004474
2	-0.000037	-0.004230	0.014789	0.012188	0.001017	-0.017613	0.006671	-0.008660	-0.007416	-0.008700
3	0.013499	-0.094094	0.185421	0.042572	-0.001479	-0.179605	0.032184	-0.064003	0.010076	0.051001
4	-0.011919	-0.024864	0.064964	0.012248	-0.011017	-0.073085	0.010099	-0.004717	-0.000320	-0.004732
...	...	...	...	...	...	...	...	...	...	...
9995	-0.002485	-0.019801	0.058093	0.035290	-0.013855	-0.054805	0.017922	-0.005387	0.002801	-0.004462
9996	-0.016569	-0.066778	0.096856	0.093593	-0.031721	-0.148599	0.008495	-0.049312	0.009550	0.008248
9997	-0.070432	-0.086170	0.429437	0.391332	0.072249	-0.473941	-0.034453	-0.175643	0.158589	0.139813
9998	0.001783	-0.001582	0.006642	-0.008941	-0.001837	-0.005981	-0.004872	-0.008657	-0.002686	-0.003959
9999	0.001316	0.012899	-0.013818	-0.008957	0.001464	0.011831	-0.002893	0.012535	0.003150	0.003305

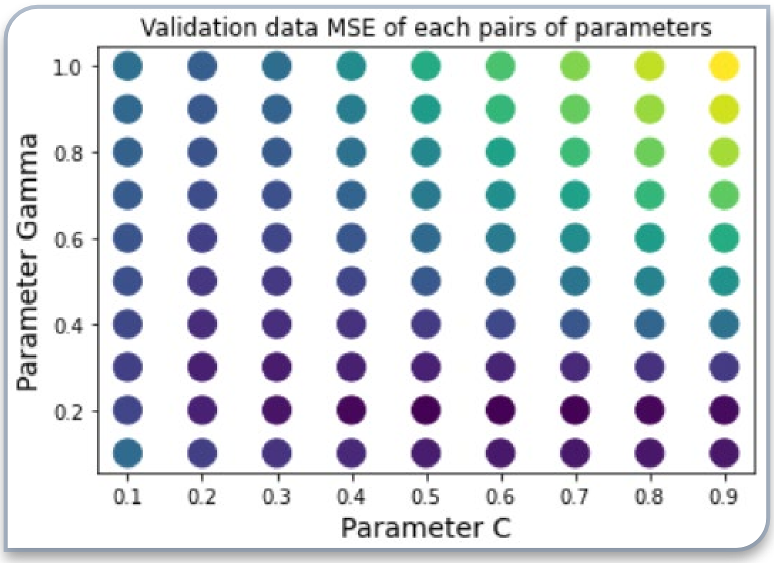
9977 rows × 104 columns

3. Model Training

Before select the models, the training data need to be splited into training dataset and testing dataset, so we use 10% as the testing dataset, and 90% as the training dataset. Then we will split training dataset again and get two parts of data, training data and validation data. In order to evaluate each model, we calculate the "MSE", "R2_score", "Variance_Score" of each model.

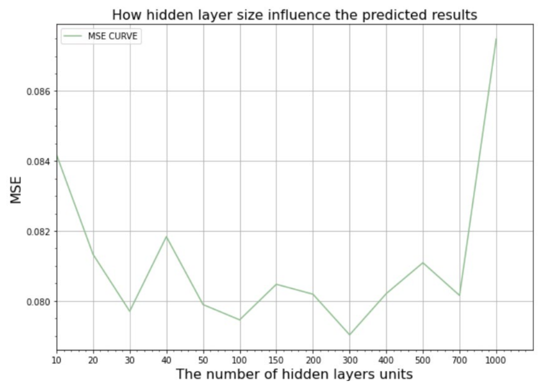
3.1 SVM

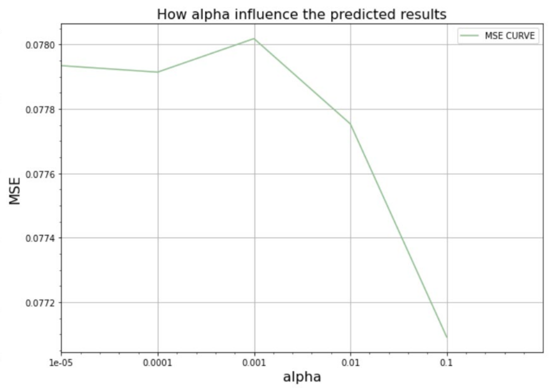
The first model we used was SVR model. Here are the results of different MSE with different parameters. As you can see from the picture, the deeper color is, the result is better. For example, you can see the point(0.2, 0.6) gets a good result. And the best MSE of the SVR model using 3-fold to calculate is **0.0757**



3.2 MLP

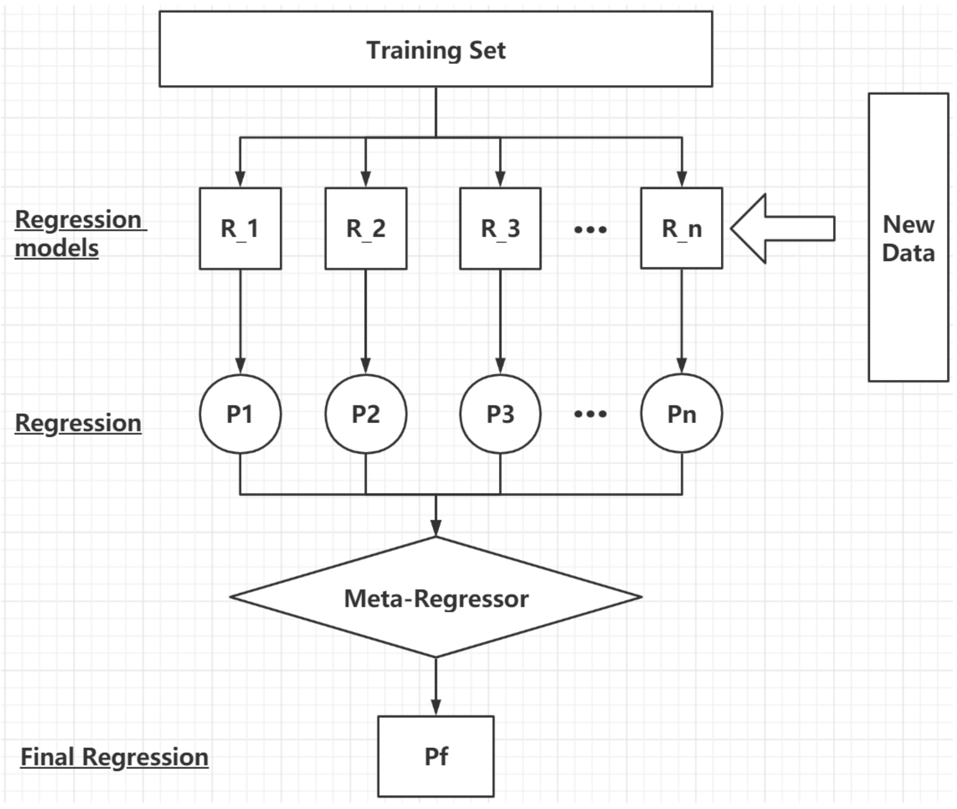
While facing with the large number of features, nurel network may be good to avoid the high variance. MLP is the simple structure to do this job. After testing different number of hidden layer (more hidden layer may leads to overfitting). Here we decide a single hidden layer mlp, and try different number of the units and the L2 penalty. On the left hand picture is the process of adjusting the hyper-parameters. We can see for our validation data, the best number of one hidden layer units is 300, and l2 penalty is 0.1. The best MSE of the **0.0768**





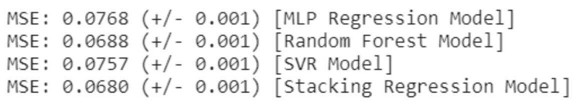
3.3 Stacking

Stacking is a very popular ensemble learning model. It has many different types of models as basic learners and combines them with a meta model. The picture shows that how stacking works. Firstly we get a subset of training data as New Data, then we train each basic regression model and use them to predict the new data. After that we regard the output is new feature X, and the book rating of new data is Y, to train a meta regressor. We choose mlp、svr and randomforest as basic models and we feed the model to stacking model to get a better final regressor. The best MSE of the stacking model is **0.0680**



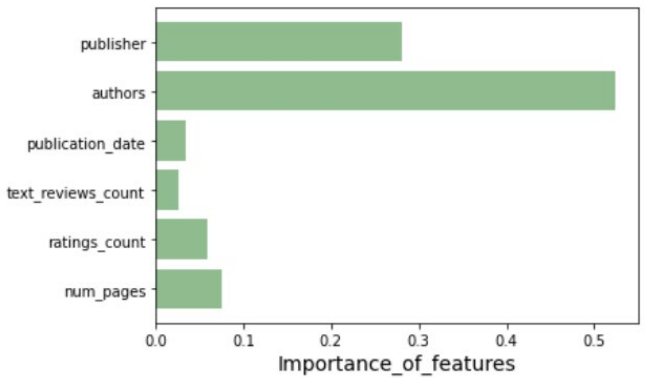
4. Evaluation

Cross-validation and visualization are main methods we use to do our optimization. The Picture one shows the MSE of each model we used. Picture two is the feature importance of each feature. Picture three to picture five show the validation data predicted results——the red is predicted value and green one is true value.

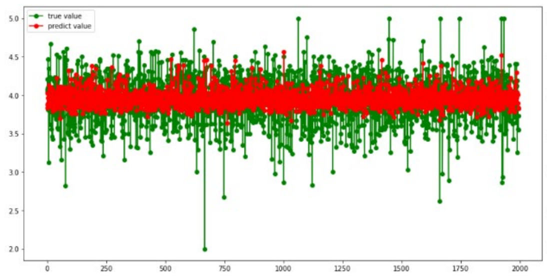


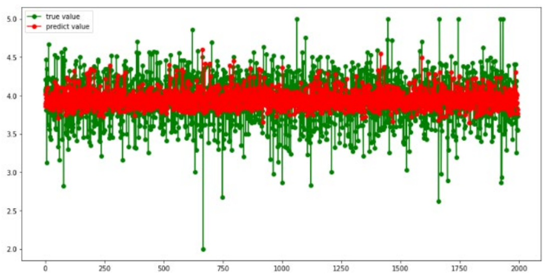
```
MSE: 0.0768 (+/- 0.001) [MLP Regression Model]
MSE: 0.0688 (+/- 0.001) [Random Forest Model]
MSE: 0.0757 (+/- 0.001) [SVR Model]
MSE: 0.0680 (+/- 0.001) [Stacking Regression Model]
```

Picture one: MSE of each model

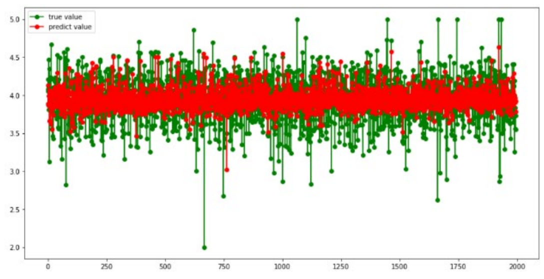


Picture two: Feature Importance





Picture three: SVR model Picture four: MLP model



Picture five: Stacking model

5. Summary

With the help of embedding algorithm (here we use word2vec), we can extract clearer information for our model training. After several training iterations, model selections, and re-processing the training data (such as dropping some irrelevant feature and unimportant feature, adjusting the splitting etc. strategy), we finally find out the best hyper-parameter for each models and the stacking model which has the best performance. As you can see, with the help of cross-validation (3-fold) we can get the best MSE in 0.680 for the validation data.

As an ensemble learning, stacking can easily improve the performance as well as reducing the variance by merging the predicted results from learning basic learners` output. Here the basic learners we used are MLPregressor, SVMregressor and Randomforestregressor. Last but not least, To predicting our project Test_data.csv, we should process our Test_data again in the same way, and use the trained stacking model (stacking for all train_data.csv) to predict it, and out put in csv format

6. References

[1] Manley J L, Tacke R, Hogan B L M, et al. SR proteins and splicing control 1569[J]. Genes & development, 1996..

[2] Manley, James L., et al. "SR proteins and splicing control 1569." Genes & development (1996).

[3] Díaz-Uriarte R, De Andres S A. Gene selection and classification of microarray data using random forest[J]. BMC bioinformatics, 2006, 7(1): 3.

[4] Liaw A, Wiener M. Classification and regression by randomForest[J]. R news, 2002, 2(3): 18-22..

[5] Joachims T. Making large-scale SVM learning practical[R]. Technical Report, 1998.