Projektarbeit

Certified Data Scientist

Exploratory data analysis and prediction using machine learning algorithms

verfasst im Rahmen der EN ISO / IEC 17024-
Zertifizierungsprüfung von

Chenwei Li

02.02.2022

Eidesstattliche Erklärung

Hiermit versichere ich an Eides statt, dass ich die vorliegende Projektarbeit eigenständig und ohne Mitwirkung Dritter angefertigt habe. Quellenangaben wurden entsprechend als solche gekennzeichnet.

____

Ort, Datum, Unterschrift

Content

1 Introduction

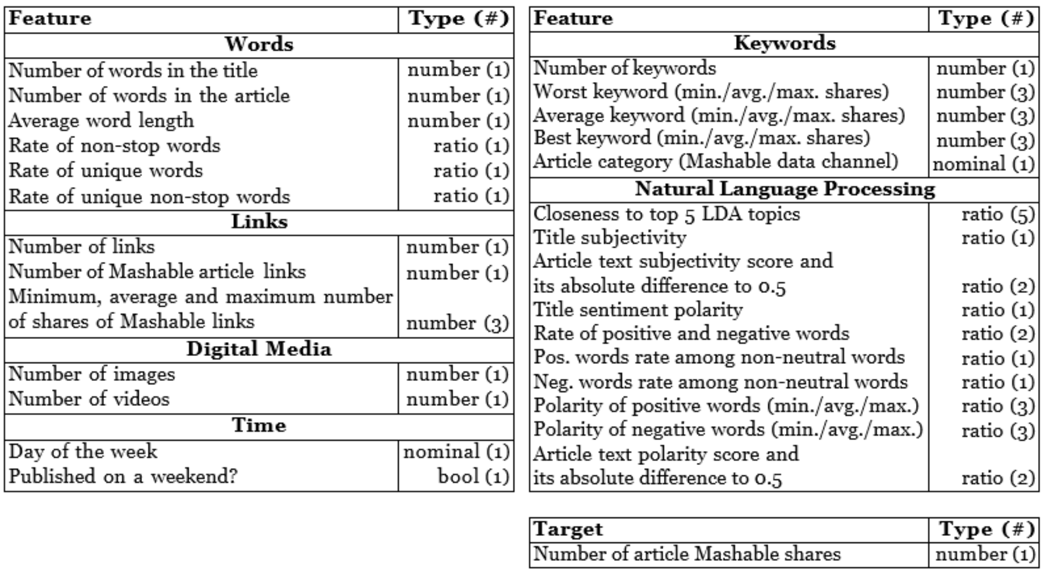
Nowadays with the development of the Internet, most people especially the younger generation prefer to receive information from online news rather than traditional media. Since the analysis of online news popularity can evaluate the value of news before it is published, predicting the popularity has become a new research trend and makes a great contribution to news publishers, recommendation systems or even activists. In this project, the study is mainly focused on the prediction of online new popularity using traditional machine learning algorithms such as Random Forest and SVM. To analyze the data comprehensively and accurately predict the popularity, an exploratory data analysis is performed as the first task. In this task, the correlation between each feature is investigated and a reverse of OneHotEncoder is applied in preprocessing, after then the different distributions of instances are plotted and analyzed according to the popularity. In the second task, a new label is firstly created based on the number of shares (popularity), thus making the popularity prediction as a classification problem. Some typic classification algorithms are first used in this task, and then 4 performance optimization methods are tried. In addition, a brief study of performance analysis and multiclass classification is performed. As the third task, some regression algorithms are performed to predict the number of shares, the impact of different dataset sizes on the training and test error is also investigated. Finally, an advance ensemble learning method stacking is introduced to optimize the performance. The code part of this project is publicly available on Github at https://github.com/hisly/digethic_project.git.

2 Exploratory Data Analysis

2.1 Data Acquisition and Description

The data source of online news popularity is from UCI Machine Learning repository (https://archive.ics.uci.edu/ml/datasets/online+news+popularity) which summarizes a heterogeneous set of features about articles published by Mashable in a period of two years. The number of attributes is 61, of which there are 58 predicted attributes, 2 non-predicted attributes and 1 target field. Number of instances is 39644 and no missing values.

The target feature is “shares” (Number of shares). 2 non-predicted features are “url” (URL of the article) and “timedelta” (Days between the article publication and the dataset acquisition), and the feature “url” can be dropped without doubt. For the” timedelta”, If the days in between is longer, theoretically there would be more shares in the news, therefore it is still necessary to indicate whether this feature is related with shares and will be discussed in later 2.2 chapter. According to the study by **Kelwin Fernandes**[1], who donated this dataset, these predicted Features can be divided into 6 categories as shown in Table 1. Most features are easy to understand, the feature “LDA” is used with **Latent Dirichlet Allocation**[2] algorithm to identify all the news into five top relevant topics and then measure the closeness of every article to such topics. **Kelwin Fernandes**[1] also adopted the **Pattern web mining module**[3] to compute the subjectivity and polarity sentiment analysis. Noted that the features “weekday” and “data channel” were already categorized as one-hot numeric feature, therefore a reverse of OneHotEncoder will be applied for data exploratory analysis.



Feature	Type (#)
Words	
Number of words in the title	number (1)
Number of words in the article	number (1)
Average word length	number (1)
Rate of non-stop words	ratio (1)
Rate of unique words	ratio (1)
Rate of unique non-stop words	ratio (1)
Links	
Number of links	number (1)
Number of Mashable article links	number (1)
Minimum, average and maximum number of shares of Mashable links	number (3)
Digital Media	
Number of images	number (1)
Number of videos	number (1)
Time	
Day of the week	nominal (1)
Published on a weekend?	bool (1)

Feature	Type (#)
Keywords	
Number of keywords	number (1)
Worst keyword (min./avg./max. shares)	number (3)
Average keyword (min./avg./max. shares)	number (3)
Best keyword (min./avg./max. shares)	number (3)
Article category (Mashable data channel)	nominal (1)
Natural Language Processing	
Closeness to top 5 LDA topics	ratio (5)
Title subjectivity	ratio (1)
Article text subjectivity score and its absolute difference to 0.5	ratio (2)
Title sentiment polarity	ratio (1)
Rate of positive and negative words	ratio (2)
Pos. words rate among non-neutral words	ratio (1)
Neg. words rate among non-neutral words	ratio (1)
Polarity of positive words (min./avg./max.)	ratio (3)
Polarity of negative words (min./avg./max.)	ratio (3)
Article text polarity score and its absolute difference to 0.5	ratio (2)

Target	Type (#)
Number of article Mashable shares	number (1)

Table 1. List of attributes by category from the study by **Kelwin Fernandes**[1]

2.2 Data Preparation

2.2.1 Feature Preprocssing

As mentioned previously, the relation between „timedelta“ and target “shares” will be investigated and the plotted figure is shown below. Pearson correlation coefficient is also calculated to verify the result, which is 0.0086. After double verifying, this feature is proved to have no relation with target and can be dropped like “url”.

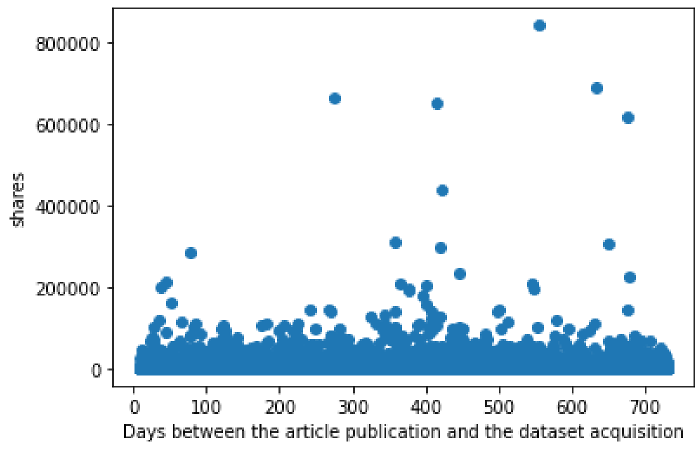


Fig 1. Relation between the timedelta and shares

The high correlation features are also identified with heatmap which is shown in Figure 2, and 7 Features are removed due to the high correlation. At last, a reverse of OneHotEncoder is applied to transfer “weekday” and “data channel” from numeric feature into categorical feature, so that these13 features can be compressed into 2 features. Since the feature “weekday” includes weekend, the feature “is_weekend” is redundant so it is removed.

After feature preprocess, the number of features is reduced from 61 to 40.

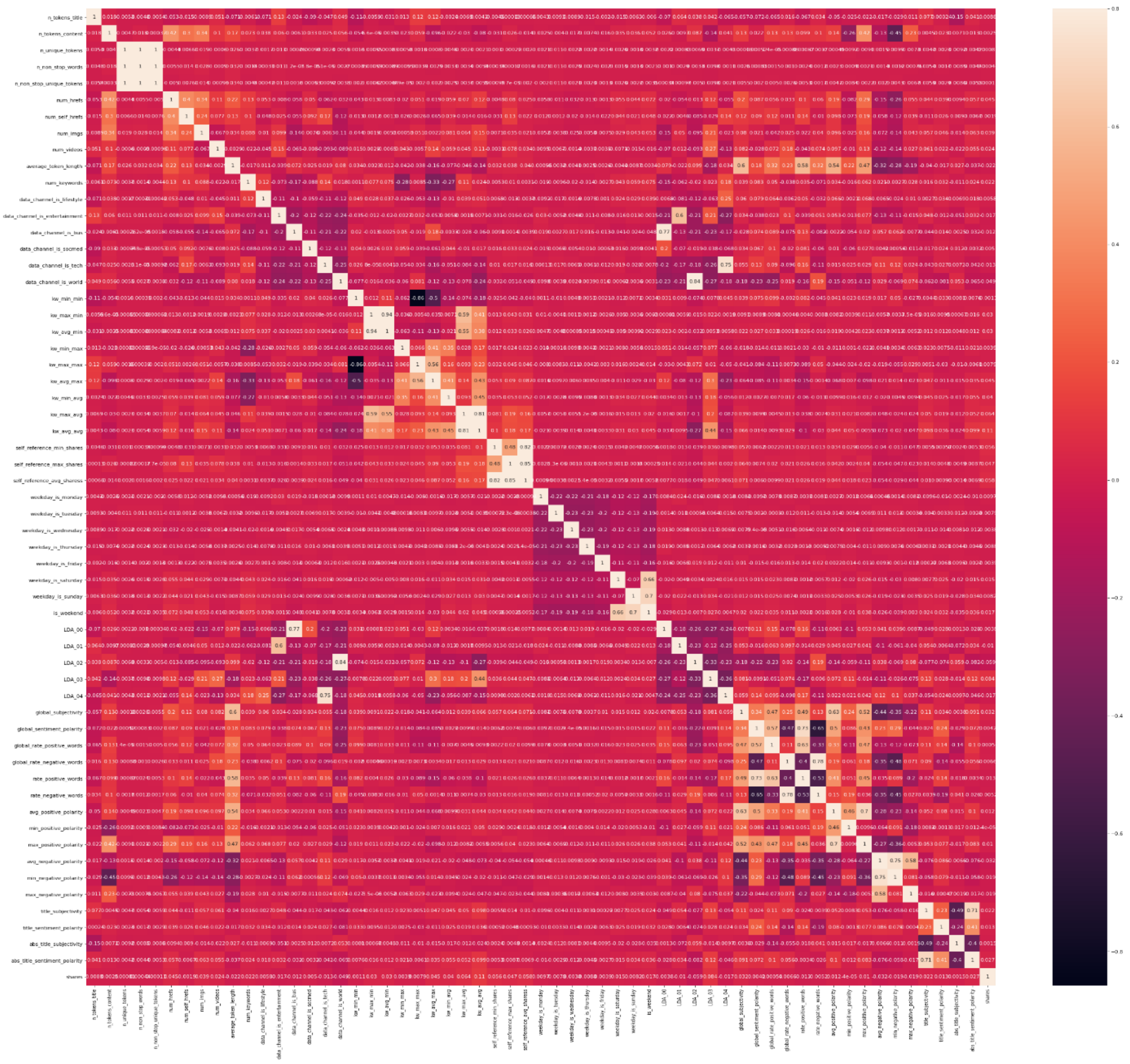


Fig 2. Feature correlation in Heatmap

2.2.2 Instance Preprocessing

The step to check NaN value is omitted because there are no missing values in the data set description, and all the Null values are reasonable. Noted that in feature “n_unique_tokens” the mean value is 0.548 but max value is 701. After an inspection it turns out that this value 701 is wrong and is replaced by 0.701. Finally, the outliers from target are indicated and these 2765 outliers with extremely high shares will be separated from dataset and labeled as tiktok_news. In the later exploratory analysis, they will be compared with the news with normally high shares.

After data preprocess, the number of instances is reduced from 39644 to 36879.

2.3 Data Exploratory Analysis

Because there are 2765 instances in tiktok_news, the same amount of news with lowest and highest shares are also selected out from the processed dataset, which are categorized as bad_news and good_news. The Analysis is performed on the comparison of these two groups namely “good_news vs tiktok_news” and “good_news vs bad_news”.

2.3.1 Data Analysis for Words & Links & Digital Media category

All 3 kinds of News have nearly the same distribution among these 3 categories. The comparison between the good_news and tiktok_news doesn’t show a significant difference. However, the comparison between good_news and bad_news shows that for those features “n_unique_tokens”, “num_hrefs”,” self_reference_avg_shares” and “num_imgs”, the most instances in bad_news have always a higher number than those in good_news. This indicates that the rate of unique words in content is better less than 0.8; the number of links in an article should be less than 200; also the average length of words in the content is better not longer than 200000, otherwise the readers may not have the patience to finish reading; besides, too many images will not make the readers happy, and the maximal image number should be less than 60.

2.3.2 Data Analysis for Weekday & Channel category

The analysis of weekday indicates that the difference between good_news and tiktok_news is too small and can be ignored, but the comparison between good_news and bad_news shows a clearly difference that the count of Saturday and Sunday in good_news is apparent larger than that in bad_news. Since the different working days don’t influence the target “shares” too much, the feature “is_weekend” is an important factor and will be used later for the prediction.

In the channel category, one interesting point is that both tiktok_news and bad_news in data channel “World” and “Entertainment”, so these two features are like a double-edged sword. The rate of „Tech “, „Lifestyle“ and „Social Media“ in bad_news is apparently lower than those in good_news, which means if an article is in these three channels, then the probability of being a bad article will be relatively low.

2.3.3 Data Analysis for Keywords category

In the comparison of “good_news vs tiktok_news”, the feature “kw_avg_avg” shows an apparent difference, tiktok_news has a lower number of averaged keywords. This also happened in the comparison of “good_news vs bad_news” where good_news has a lower number. Therefore, too many keywords in an article are normally not a good thing. In the distribution of feature “kw_min_avg” also verifies this conclusion.

2.3.4 Data Analysis for NPL category

Due to the large number of features in NPL category, the analysis will be performed within small groups of features. The analysis for “LDA” shows that the rate of tiktok_news is clearly higher in LDA topic 0 and lower in LDA topic 3. Therefore, if the content of an article is closer to LDA topic 0, then it has a higher probability of being an excellent article. For the comparison in “good_news vs bad_news”, the rate of LDA topic 1 and topic 2 in good_news is apparently higher and conversely LDA topic 0, topic 3 and topic 4 have a lower rate. It indicates the content of a good article is usually closer to LDA topic 1 and 2. Based on the analysis, it has risks if the content of article is close to LDA topic 0, but topic 1 and 2 are always a good choice.

Another difference occurs in feature “rate_positive_words” and “rate_negative_words”. Compared with bad_news, the article in good_news has always a higher rate of positive words and lower rate of negative words. It is also reasonable because positive things are always popular except in some medical tests.

Since there is no obvious difference in the comparative analysis for other features, so in this project they will not be discussed any more.

3 Classification

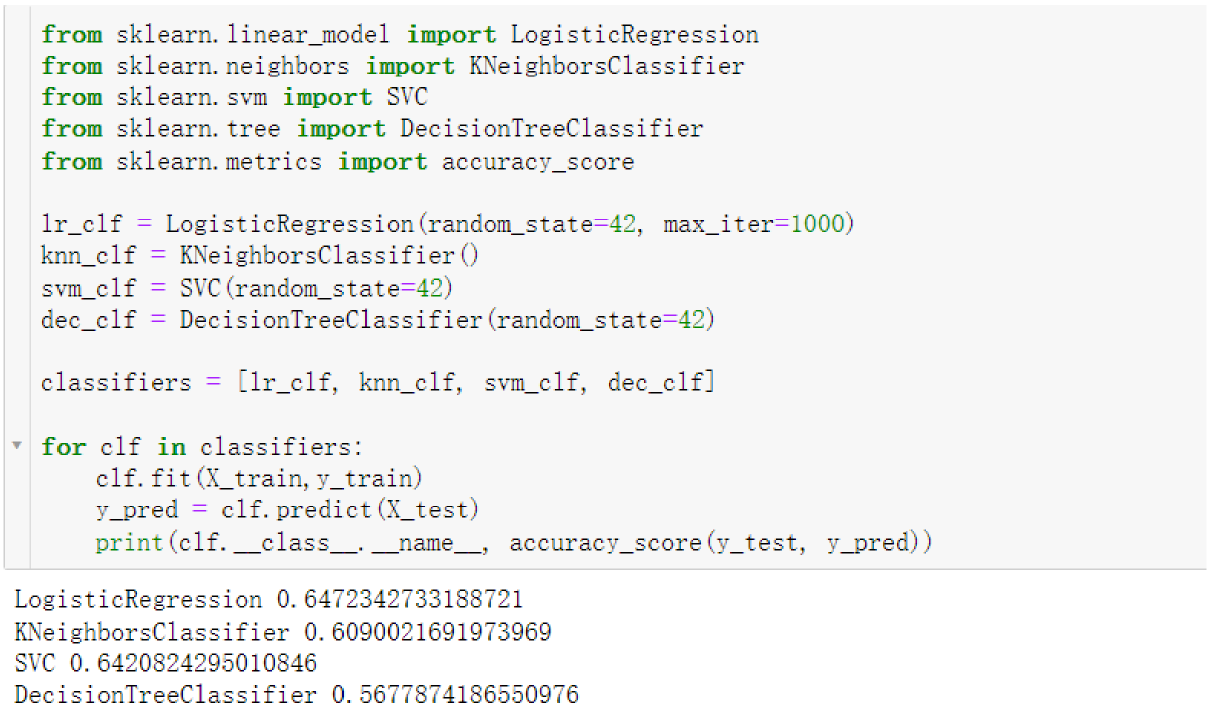
3.1 Data Preprocess

The exploratory analysis of data has shown that the feature “is_weekend” has a larger impact on the target “shares”. Therefore, in the data preprocess for classification task, the other 7 features for different weekdays and those features with high correlation are removed. Besides, the outliers in target are removed since the comparison of “good_news vs tiktok_news” indicates that these two kinds of news have nearly the same distributions.

A new feature “popularity” is created as target based on the number of shares, which makes the task into a classification problem. As we know, a normalization process is not necessary for the tree models, because they don't care about the value of instances, but about the distribution of the instances and the conditional probability. However, normalization or standardization is still necessary for the most algorithms such as SVM or logistic regression. In this project, those classical machine learning algorithms will be used for the prediction, therefore all data are normalized.

3.2 Machine Learning Algorithms

In this project, 4 classical Machine Learning Algorithms are performed for the prediction, which are Logistic Regression, KNN, SVM and Decision Tree. The performance of these algorithms is based on accuracy score first, the specific analysis of the results will be discussed later. With different Algorithms, the accuracies are also very different. Although the Logistic Regression and SVM have better performances, The overall result is still not satisfactory. Therefore, a performance optimization is essential.



```python
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score

lr_clf = LogisticRegression(random_state=42, max_iter=1000)
knn_clf = KNeighborsClassifier()
svm_clf = SVC(random_state=42)
dec_clf = DecisionTreeClassifier(random_state=42)

classifiers = [lr_clf, knn_clf, svm_clf, dec_clf]

for clf in classifiers:
    clf.fit(X_train, y_train)
    y_pred = clf.predict(X_test)
    print(clf.__class__.__name__, accuracy_score(y_test, y_pred))
```

```
LogisticRegression 0.6472342733188721
KNeighborsClassifier 0.6090021691973969
SVC 0.6420824295010846
DecisionTreeClassifier 0.5677874186550976
```

3.3 Performance Optimization

3.3.1 Outliers Handling

As we know, the outliers in features can sometimes contaminate dataset and interfere model training. Therefore, removing the outliers can normally help to increase the performance. In this project these outliers in each feature are replaced with mean values. The comparison of accuracy is shown in Table 2 as blow. As a result, the outliers handling doesn't improve the performance, which indicates that removing outliers isn't effective all time. Hence, this method is not applied in classification task.

	Accuracy without outliers handling	**Accuracy with outliers handling**
LogisticRegression	0.647	0.632
KNeighborsClassifier	0.609	0.577
DecisionTreeClassifier	0.568	0.563
SVC	0.642	0.645

Table 2. Comparison of accuracy with outliers handling

3.3.2 Hyperparameter Tuning

Hyperparameters control model complexity, which can affect the performance of the classifiers. Due to the diversity of datasets, the default hyperparameters in classifiers are normally not the optimal values. Therefore, a hyperparameter tuning is needed to improve the models' performance.

Since the Logistic Regression has simply hyperparameters, in this project the other three basic classifiers will be fine-tuned. For KNN, the best number of neighbors is optimized with a for-loop. In Decision Tree, 2 Hyperparameters "max_depth" and "min_samples_leaf" are fine-tuned using GridSearchCV. Since some hyperparameters in SVM are continuous values, thereby RandomizedSearchCV method is used instead of GridSearchCV for a SVM Classifier. Besides, 2 kinds of kernel “poly” and “rbf” are also attempted to obtain the optimal performance in SVM.

Noted that these hyperparameters have already been adjusted several times before reaching the optimal values. Hence, all the optimal hyperparameters are “lucky” within the set range. As seen in Table 3, the accuracy of classifiers except SVC has been significantly improved after a hyperparameter tuning process, especially for Decision Tree Model. Because if the parameters such as the tree’s maximal depth are not limited, the tree model will be very complex after training and may cause overfitting problem. The SVC model in table 3 is used “poly” kernel with degree=2 instead of “rbf” kernel since it has a slightly higher accuracy.

	Accuracy without Hyperparameter tuning	Accuracy with Hyperparameter tuning
KNeighborsClassifier	0.609	0.622
DecisionTreeClassifier	0.569	0.632
SVC (“poly” kernel)	0.642	0.646

Table 3. Comparison of accuracy with hyperparameter tuning

3.3.3 Ensemble Learning

Ensemble learning can combine the weak models into a strong model thus improving the accuracy of prediction. The most used ensemble methods are bagging and boosting. As a typic bagging method, the random forest classifier is applied and Adaboosting is used as boosting method in this project. Finally, a voting classifier is integrated with above algorithms.

These ensemble learning algorithms have also been performed with hyperparameter tuning process. In voting classifier, both “hard” and “soft” voting are used for a comparison. “hard” voting uses simply the most predicted class as majority rule voting, but “soft” voting is based on the sums of predicted probabilities. Hence, all the basic estimators must be capable of probability-prediction, and the parameter “probability” in SVC model is set True for running the “soft” voting classifier. The results of accuracy are given in table 4. As seen in this table, all the ensemble models have obviously a better accuracy than the single estimators above.

	Accuracy
RandomForestClassifier	0.650
AdaBoostClassifier	0.655
VotingClassifier(hard)	0.651

Table 4. Comparison of accuracy with hyperparameter tuning

3.3.4 Feature Engineering

The last optimization step is Feature Engineering, which is also a common method in machine learning. Since too many features increase time/computational cost and complicate the models, which will lead to a bad prediction and cause overfitting when the corresponding dataset is not large enough. In this Project, two methods are used to investigate the influence of features on performance.

The first method is feature importance. In a tree model, nodes are divided according to the purity (such as GINI or Information entropy gain). The more important a feature is, the better it will increase the purity in a node. Based on this principle, the importance on each feature can be estimated by the tree model. In this project, 11 features with lowest importance are selected out using RandomForestClassifier and a new round training with this extracted dataset is proceeded to compare with the original dataset. The result is shown in table 5, when the number of dropped features are increased, the time cost trends to decrease which is reasonable. However, the accuracy has nearly no change. This indicates the number of features in the dataset is not too large yet to make the model complex, and a feature extraction process is not necessary.

	Accuracy	Time cost
Original Data	0.651	3.95 s
Removing 1 feature	0.650	4.09 s
Removing 6 features	0.652	4.14 s
Removing 11 features	0.650	3.78 s

Table 5. Comparison of accuracy and time cost with feature importance

The second method is dimensionality reduction, and PCA is applied to compress the number of features. As seen in Figure 3, when the dimension number is reduced to 15, but there are only 10% information loss. The most likely reason is that some features still have relatively high correlations or there are some sparse features. The result of comparison is similar with feature importance, and the time cost is less but accuracy is also reduced, which verifies the above inference.

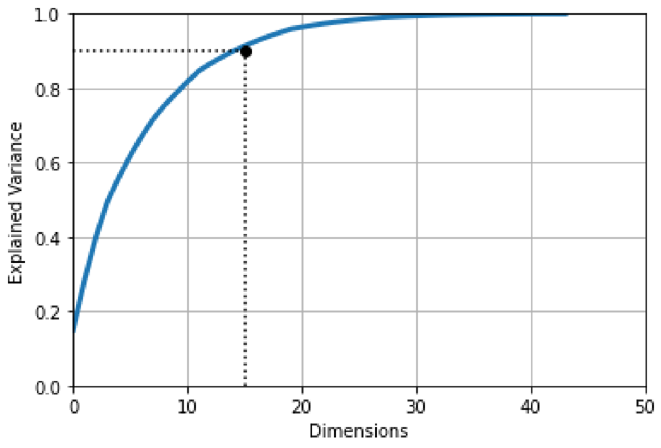


Fig 3. Relation between reduced dimensions and information loss

3.4 Performance Analysis

3.4.1 Precision & Recall

Usually, for binary classification problems, the accuracy cannot explain the real performance of models very well, especially when the dataset is imbalanced. Hence, to verify whether the accuracy can represent the performance of a model, 2 advanced metrics precision (how many percent the positive predicted instances are correct) and recall (how many percent the real positive instances can be predicted) are introduced to investigate the error of models. A confusion matrix is plotted in figure 4 for calculating precision and recall. Furthermore, a combination of these 2 values is calculated as f1_score, which can estimate the performance of models based on both precision and recall.

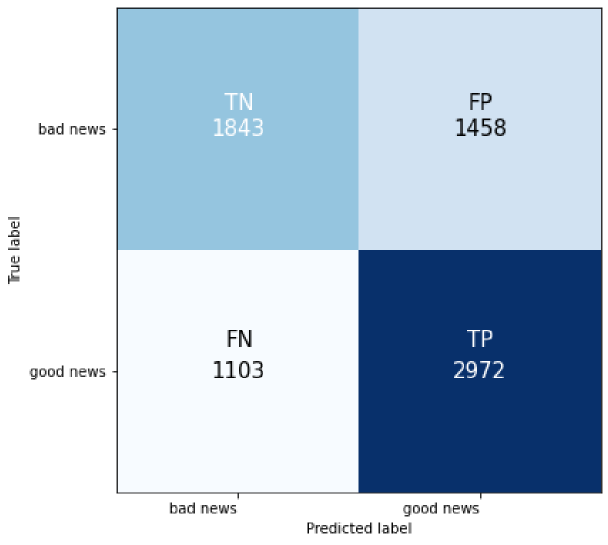


Fig 4. Confusion matrix

As seen in table 6, since these 3 values are close to the accuracy, it indicates that using accuracy to estimate the performance of different algorithms is reasonable.

	Equation	Value
Precision	TP / (TP + FP)	0.671
Recall	TP / (TP + FN)	0.729
F1_score	2 * (Precision x Recall) / (Precision + Recall)	0.699

Table 6. advanced metric values

3.4.2 Threshold Analysis

In Logistic Regression, the prediction by each instance is computed first as probability under the hood. Therefore, 2 kinds of probabilities for positive and negative classes are computed to compare with a threshold. Here I selected the probability for positive class. If it is greater than the threshold, then this instance will be assigned to the positive class (good news); otherwise, labeled as negative class (bad news). A classification threshold value is always default as 0.5, but this value can be manually tuned. Lowering the threshold results the instances are more likely to be predicted as positive and thus increasing the false positive number. According to the equations in table 6, this will reduce precision value and increase recall value. Conversely, increasing the threshold can make precision value higher and reduce the recall. As seen in figure 4, the change of precision and recall with threshold is called Precision / Recall trade-off. It signifies that there is not a perfect threshold value which can increase both precision and recall at same time.

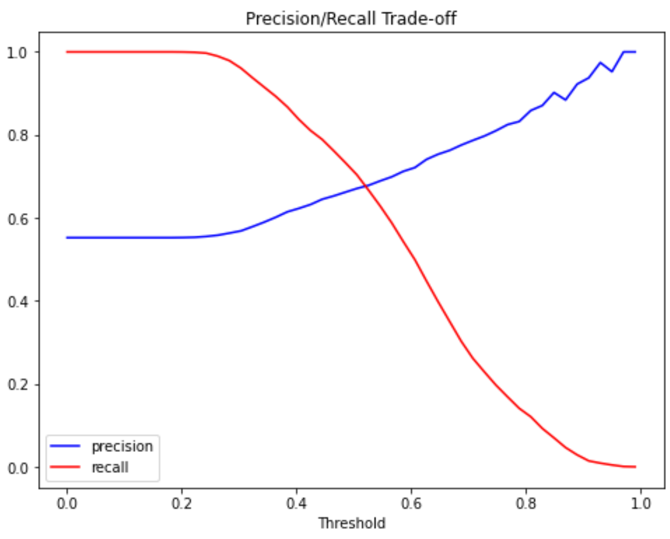


Fig 4. Precision / Recall Trade off

Which threshold is optimal depends on the requirements, and a profit analysis as requirement is performed based on different threshold values. In this project, I assume if an article is predicted as good news and the true label is good news, then the profit is 300 euros, but if true label is bad news, there will be 250 euros loss; if an article is predicted as bad news but the true label is good news, then there are still 50 euros loss. And if true label is also bad, then no gain no loss. Thus, a profit graph for the test set can be plotted based on the variance of threshold. As seen in figure 5, when threshold is around 0.45, we can have the maximal profit.

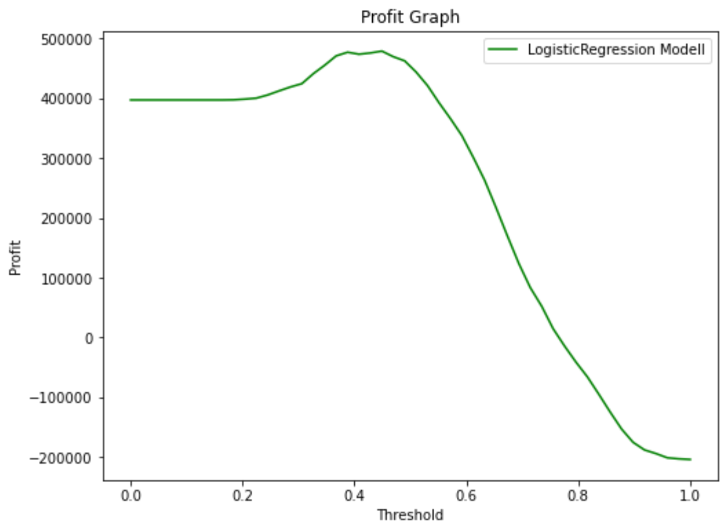


Fig 5. Profit graph

3.5 Multiclass classification

The last task in classification is a multiclass problem. Based on the number of shares, 5 classes are manually labeled which are “bad”, “sufficient”, “satisfying”, “good” and “very good”. At first, a RandomFrorestClassifier is used for this task and the confusion matrix is plotted blow in figure 6, the result shows a high accuracy with Random Forest, which performs better than in binary class task. The only problem is that the instances in “good” and “satisfying” are more easily to be predicted as “sufficient” which leads to a bad performance if we only want the “sufficient” instances. Therefore, another algorithm SVM is also used to solve this problem.

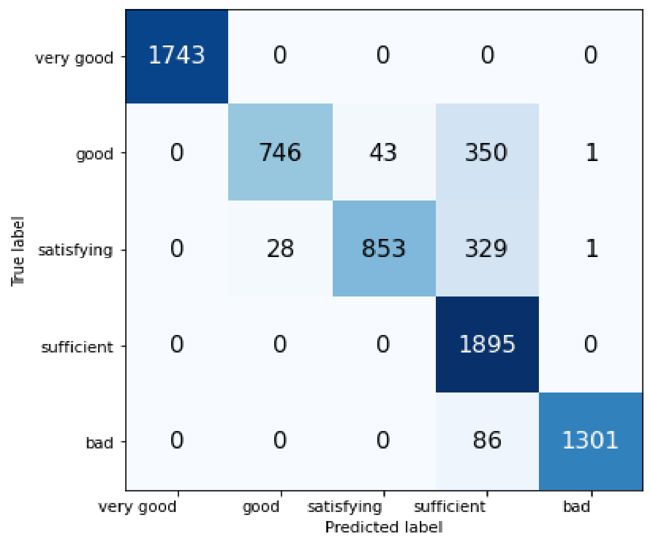


Fig 6. Confusion Matrix (Random Forest)

Some algorithms can natively handle the multiple classes task such as Random Forest, but others such as SVM are only capable of handling binary classes. However, 2 strategies OneVsOne and OneVsRest can help such kind of algorithms to solve multi-classification problems. The default strategy in SVC is changed to OneVsRest with version 0.19 in Scikit-Learn. As shown in figure 7, the original SVC has a very poor prediction accuracy, and almost all the predictions fall on the same label “sufficient”. However, after adding a large regularization parameter C=500, the predicted results are much better even in “sufficient” class. Since the SVC with default parameter has a better performance in binary classification but is overfitting in multiclass prediction. It indicates that these 2 kinds of classification have a big different even with same dataset.

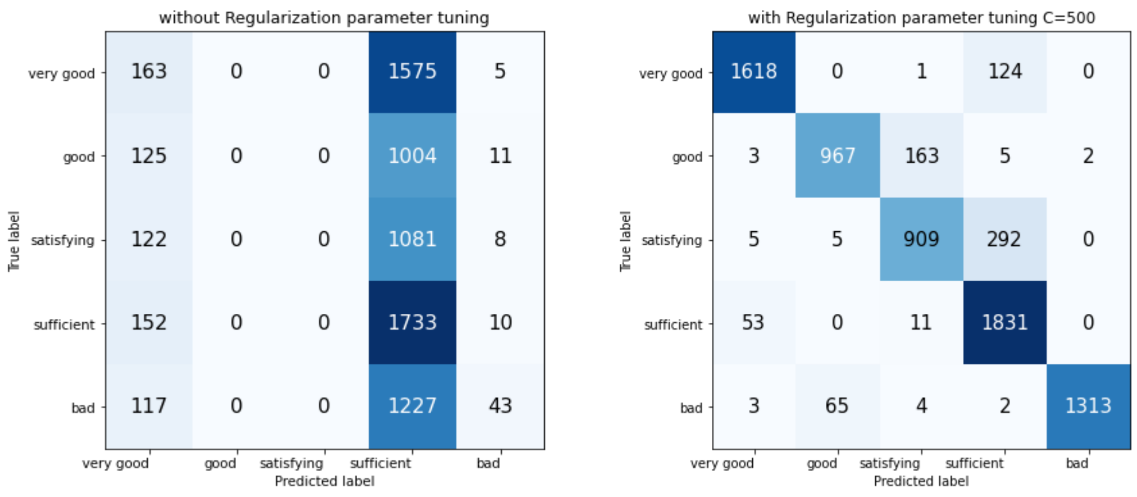


Fig 7. comparison with different regularization (SVC)

As discussed above, the default strategy in SVC is OneVsRest. However, we can still force SVC to use OneVsOne with OneVsOneClassifier. The result in figure 8 shows that there is only a slight difference between these two strategies. In this project, the OneVsRest is preferred, since the OneVsOne strategy needs to train a binary classifier for every pair of classes, which means 10 classifiers are needed to for the classification and thus leads higher time & computational consumption than OneVsRest.

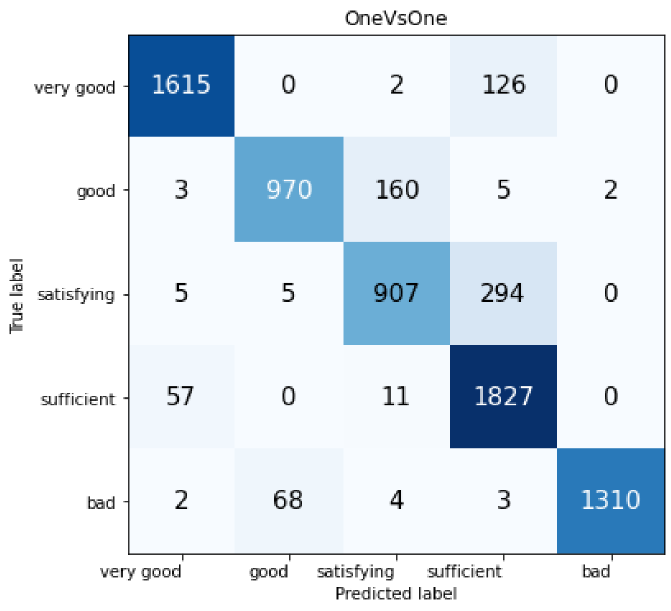


Fig 8. SVC with regularization using OneVsOneClassifier

4 Regression

4.1 Linear Regression

The data preprocess in regression is same as in classification, the only difference is target feature “shares” is used directly for this task. As the most common algorithm in Regression problem, Linear Regression is firstly used for the prediction. The RMSE (Root Mean Squared Error) is used as performance criterion. As seen in table 7, the RMSE is unexpectedly bad, but after removing all the outliers from features, RMSE value is reduced to a reasonable value with 1502.09. It turns out that Outlier’s handling is necessary for regression rather than classification, because when outlier values in some features are extremely high or low, as a result the weights in these features are also abnormal and making the error much larger than expected.

In the former classification task, since the SVC with “poly” kernel doesn’t show a higher accuracy, which implies that the dataset is likely linear. To confirm this guess, a polynomial Regression (degree = 2) is performed and the RMSE is higher than that in linear model. In additionally, for a linear model, the regularization process cannot improve the performance very well, and the RMSE value for Ridge and Lasso regressor in table 7 also verify this guess. Based on the above analysis this data is confirmed to be linear, so linear regressor is suitable for this regression task.

	Before Outliers Removing	After Outliers Removing			
Regressor	Linear	Linear	Polynomial	Ridge	Lasso
RMSE	348164326599.42	1502.09	1507.17	1502.11	1502.18

Table 7. RMSE value with different models

4.2 Influence of Dataset-size on Training & Test Error

The influence of Dataset size on training and test error is also included in this project to investigate how large dataset is needed for the prediction. At beginning 100 instances are used and the training batch includes 1000 instances. In figure 9, it shows the change of train & test error with increased dataset size. As seen in this figure, the train error is quite low at first which means the regressor can fit a small dataset perfectly. On the other side, because the model cannot generalize very well, resulting the test error extremely high. With the increased dataset size, the train error starts to increase but test error is reducing conversely. Both errors have barely any change after the dataset size in increased up to 13000 instances. Hence, when the size of dataset is above 13000 instances, the prediction can already perform very well.

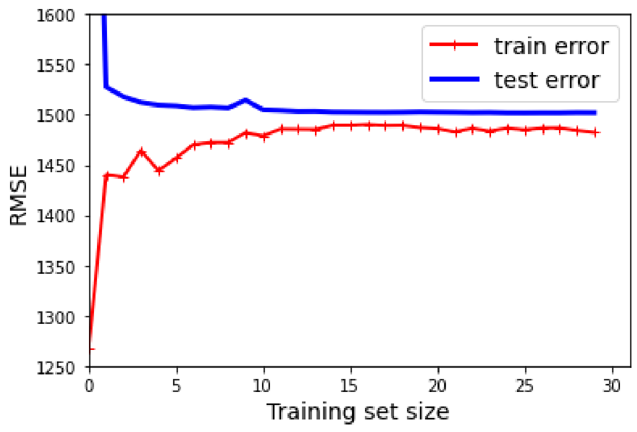


Fig 9. Train & Test Error with increased dataset size

4.3 Hyperparameter Optimization

The hyperparameter optimization is also a essential step for regression. A comparison with different kinds of hyperparameter tuning is investigated. As a result, these RMSE values are shown in table 8: The RMSE is 1513.50 with default parameter. After hyperparameter tuning this value is decreased to 1490.47, another tuning process with those parameters from classification task is also processed and the RMSE has nearly no difference compare with regression task. It indicates that the tuned hyperparameters from classification are also suitable for regression problem and the only influencing factor is dataset.

RandomForestRegressor	RMSE
No Hyperparameter Tuning	1513.50
Tuned Hyperparameter from Classifier	1490.47
Tuned Hyperparameter from Regressor	1489.94

Table 8. RMSE comparison with tuned hyperparameters

4.4 Advance Ensemble Learning: Stacking

To optimize the performance, an advance ensemble learning Stacking is used for regression task. The inspiration comes from a competition on Kaggle [4] (https://www.kaggle.com/c/otto-group-product-classification-challenge/discussion/14335). Stacking can aggregate multiple classification or regression models via meta-classifiers or meta-regression. There are two layers in stacking learning, the first layer also called base level model is trained based on the complete training set, the predictions are aggregated from first layer and are trained as input in the second layer. The second layer as meta model is trained based on these inputs from the base level model and predict a final value.

In this project, 5 Regressors are used for the first layer. As seen in figure 10, it describes the process of stacking in one regressor: during the training process, the training dataset is divided into 5 parts (with K-fold method and k=5) at first. Hence, there are 5 rounds training and in every round training, 4 parts will be used for modeling then predict the last part. After 5 rounds training, the predictions for all 5 parts can be obtained and combined as a new train data for the second layer. For test data because K-fold is not applied, these predictions for test data in every round training are averaged as test data input for the second layer. Thus, for each regressor there will be a new feature (a new column with the same number of instances for training and test data) generated.

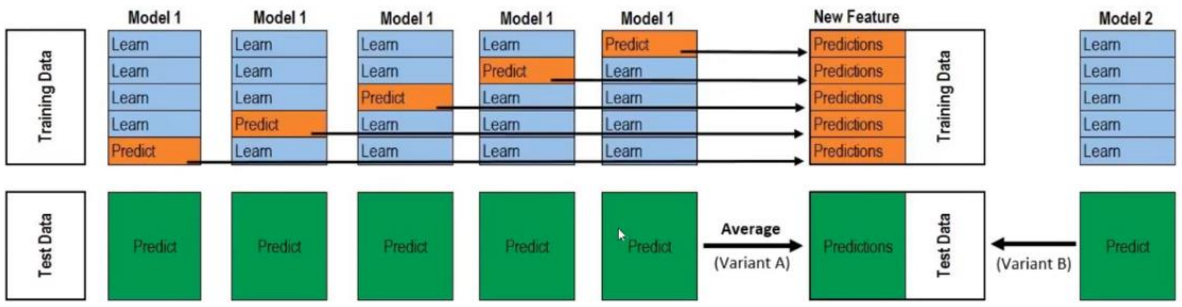
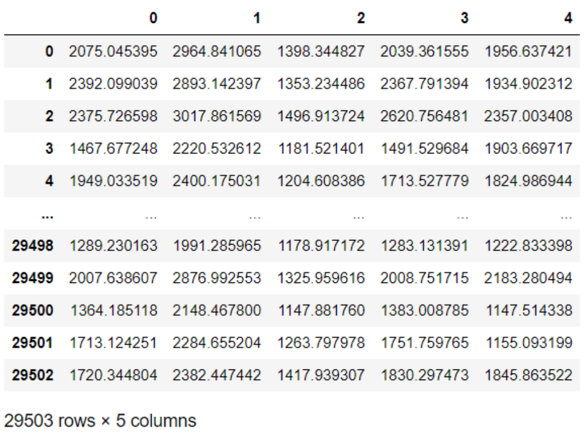


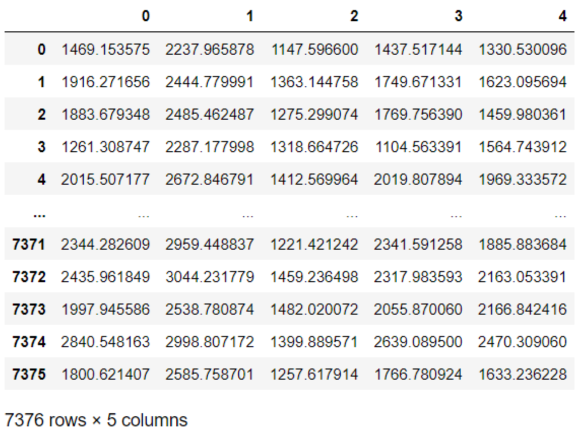
Fig 10. visualized process for stacking [5] (https://www.kaggle.com/getting-started/18153)

Since 5 regressors are applied as first layer, 5 new features are created as input to the second layer. In figure 11, these new features are shown that they have the same number of instances with original dataset, but only 5 columns. The second layer as meta learn is used with a simple regressor. The RMSE with stacking is 1477.24, which is the lowest error. It indicates that as an advance ensemble learning, stacking has normally a better performance. That's why it is the most used method in machine learning competitions.



	0	1	2	3	4
0	2075.045395	2964.841065	1398.344827	2039.361555	1956.637421
1	2392.099039	2893.142397	1353.234486	2367.791394	1934.902312
2	2375.726598	3017.861569	1496.913724	2620.756481	2357.003408
3	1467.677248	2220.532612	1181.521401	1491.529684	1903.669717
4	1949.033519	2400.175031	1204.608386	1713.527779	1824.986944
...	...	...	...	...	...
29498	1289.230163	1991.285965	1178.917172	1283.131391	1222.833398
29499	2007.638607	2876.992553	1325.959616	2008.751715	2183.280494
29500	1364.185118	2148.467800	1147.881760	1383.008785	1147.514338
29501	1713.124251	2284.655204	1263.797978	1751.759765	1155.093199
29502	1720.344804	2382.447442	1417.939307	1830.297473	1845.863522

29503 rows × 5 columns



	0	1	2	3	4
0	1469.153575	2237.965878	1147.596600	1437.517144	1330.530096
1	1916.271656	2444.779991	1363.144758	1749.671331	1623.095694
2	1883.679348	2485.462487	1275.299074	1769.756390	1459.980361
3	1261.308747	2287.177998	1318.664726	1104.563391	1564.743912
4	2015.507177	2672.846791	1412.569964	2019.807894	1969.333572
...	...	...	...	...	...
7371	2344.282609	2959.448837	1221.421242	2341.591258	1885.883684
7372	2435.961849	3044.231779	1459.236498	2317.983593	2163.053391
7373	1997.945586	2538.780874	1482.020072	2055.870060	2166.842416
7374	2840.548163	2998.807172	1399.889571	2639.089500	2470.309060
7375	1800.621407	2585.758701	1257.617914	1766.780924	1633.236228

7376 rows × 5 columns

Fig 11. New features generated by the base level models

5 Conclusion

This paper conducts an exploratory analysis to the dataset “online news popularity” and presents the investigation on this dataset with different machine learning algorithms. Based on the prediction and analysis, the conclusions can be summarized as follows:

1. The instances from outliers in target show no difference against those instances with normal high shares.

2. When the unique words in content, number of links & images and averaged length of words increased, the number of shares is likely to decrease and become bad news. Also try to release news on weekends and the channel is better related to technology, lifestyle and social media. At last, don’t use too many keywords or negative words in the news, and most important is the content of news should be closer to LDA topic 1 and 2 to obtain higher number of shares.

3. In classification, the outliers in features have no negative impact on the prediction or even a positive impact in some algorithms. Hyperparameter tuning process can apparently improve the performance, especially in those algorithms with complex parameters such as decision tree. The ensemble learning classifier has significantly a better performance than a single classifier. The number of features is not large enough to make the model complex, therefore a feature engineering is not necessary.

4. Since the dataset is balanced, the Precision and recall value doesn’t show a difference with accuracy, therefore accuracy can be used to represent the performance of models. There is not a perfect threshold value to make both precision and recall maximal. Therefore, the threshold can only be optimized according to different requirements. For a multiclass classification, RandomForestClassifier can directly be used for the prediction. As a binary classifier, SVC can also handle the multiclass with default strategy “ovr”, note that in multi-class problem, regularization is needed to prevent overfitting.

5. Unlike classification problems, it is necessary to deal with outliers in regression. The dataset is indicated to be linear, as a result, the linear regression model can fit well. Different sizes of dataset have significant effects on both training and testing error. At last, an advance ensemble learning is introduced in regression and the performance is better than any other models.

6 Reference

[1] K. Fernandes, P. Vinagre and P. Cortez. A Proactive Intelligent Decision Support System for Predicting the Popularity of Online News. Proceedings of the 17th EPIA 2015 - Portuguese Conference on Artificial Intelligence, September, Coimbra, Portugal.

[2] Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent dirichlet allocation. Journal of Machine Learning Research 3, 993–1022 (2003)

[3] De Smedt, T., Nijs, L., Daelemans, W.: Creative web services with pattern. In:Proceedings of the Fifth International Conference on Computational Creativity (2014)

[4] Otto Group Product Classification Challenge (https://www.kaggle.com/c/otto-group-product-classification-challenge/discussion/14335)

[5] explanation on stacking (https://www.kaggle.com/getting-started/18153)