Competition 1 Report

組員: 106033211 余孟旅 108062637 馮翔荏 106033233 周聖諺

1. Preprocessing

(1) Select features by using beautifulsoup

Please refer to **feature_selected.ipynb, preprocessing.ipynb** and data cleaning.ipynb for code.

We followed the example code provided by TAs. We clean the HTML tags and some unnecessary part of the raw data with BeautifulSoup4.

After observing the raw data, we discover that there are some special tags in HTML. We extract following features:

a. Dates: Includes Year, Month, Day, Hour, Minute, Second as individual features
b. Is Weekend: indicate whether the day is weekend or not
c. Author: Author of the article
d. Data Channel: It seems to be the category of the article
e. Title: The title of the article, we will convert it into numeric values via TFIDF/ Bags of Words
f. Content: The content of the article, we will convert it into numeric values via TFIDF/ Bags of Words
g. Topics: The tags of the articles.
h. Number of LInks
i. Number of The Images in the Articles
j. Average Words Length
k. Length of Articles (in words)
l. Length of Titles

(2) Word Stemming, Stop-word Removal

Please refer to **stem_compress.ipynb and data cleaning.ipynb** for code.

After fetching the essence of the raw data, we applied **word stemming(Porter Stemming) and stop-word removal** on the **contents, titles and topics** with NLTK.

2. Convert text into numeric values and dimension reduction

(1) Convert text to numeric values

Please refer to **clean_topic.ipynb and stem_compress.ipynb** for code.

Here we used both **IFIDF and Bags of Words** to count the frequency of the words of title, contents and topics. We've tried **100, 1000, infinite sizes of vocabulary bags**.

As for categorical data such as author and data channel, we encode them with LabelEncoder, we've tried both **naive encoding and one-hot encoding**.

(2) Dimension reduction

Please refer to **clean_topic.ipynb and stem_compress.ipynb** for code.

Topics, contents and titles are features fetched from raw data which includes several important tags related to content. However, to feed this feature to the model, we need to transform it into numerical values. Like other word-based features (e.g., title and content) , we use **LDA (Latent Dirichlet Allocation)** and **t-SNE** to reduce dimension because of its high dimension after performing one-hot encoding. We've also tried **LLE(Locally Linear Embedding)** and **LOBPCG (Spectral Embedding in Sklearn)**, but due to the lack of memory and time, we don't convert the matrix successfully.

Also, we've tried different parameters of LDA decomposition. We compressed the origin matrix into 2, 10 and 100 dimensions. However, it seemed, that it doesn't matter too much. t-SNE also gives the same result, although t-SHE is better than LDA a little bit.

3. Model Used

For all models, we use cross validation with 5 folds to validate the accuracy of the model.

(1) Random Forest

Please refer to **new_forest.ipynb** for code.

We try not to use all features, although it turns out that using all features got better validation accuracy. Besides, we also try mixing random forest with other models by assembling.

(2)AdaBoost,DecisionTreeClassifier ,KNeighborsClassifier

Please refer to **test.ipynb and model_tune_on_tfidf_tsne_ohe.ipynb** for code.

We've tried training somel models we have learned in the class on the standard data set and using feature_importances_ function to check the importance of all features thus we could select which features we need .The final result shows the adaboost model has the highest accuracy.

(3)Logistic Regression, Ridge Classifier

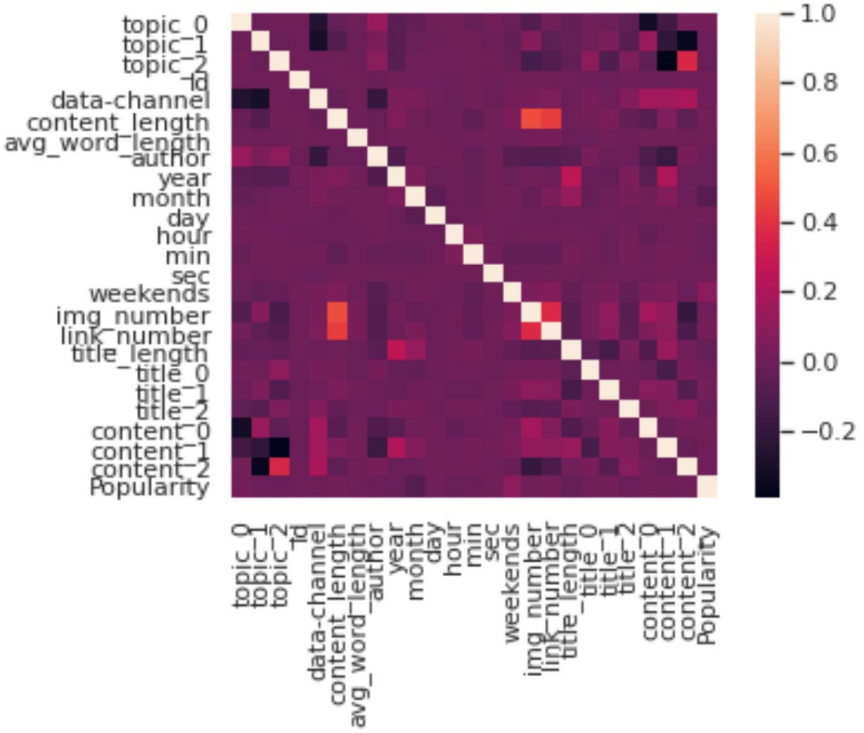
Please refer to **model_tune_on_tfidf_tsne_ohe.ipynb** for code.

It is surprising that simple linear models have good performance(about 55%) close to complex models(lose < 1%) like Random Forest, Multi-Layer Perceptron. However, we don't know why.

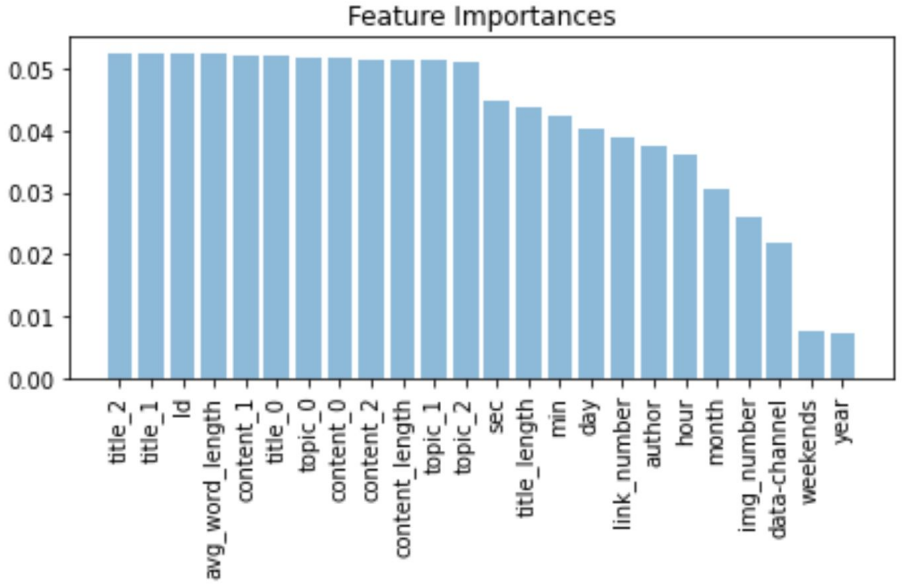
4. Conclusion

As we mentioned above, we've tried as many models and features as we can. However, we still don't achieve good performance in the competition.

About the reasons why we don’t achieve good performance, we draw the covariance matrix and feature importance plot as follows. As the charts show, the covariance of popularity and other features are close to 0 which means the features don’t give too much information.



As for feature importances, the highest feature only provides 0.05 explanation for the model.



TOP 3 Fetures: title_2, title_1, Id

As a result, we guess that bad features selection may be the reason why we cannot achieve good performance.