DS 310 Project #1: Clickbait Thumbnail Classification

Kaggle Team Name: Sean + Yiyun

Member Names: Sean Klavans, Yiyun Gong

Table of Contents

Data Preprocessing

After reading in the training data and test data, we have a column of description data that we want to perform text classification on. To pre-process the description column, we first lowercase all text data, then we remove the punctuation, remove all the special characters, remove all single characters, remove all single characters from starting and substitute multiple spaces with a single space. After cleaning up the text of the description column for both training and test datasets, we seperate the datasets into Y_train which contains the actual results of training data, X_train which contains the description column of training data, and X_test which contains the description column of test data.

Feature Engineering with Explanation

We first label encode the target variable by transforming categorical data of boolean type True/False in the dataset into numerical values 1/0 which can help fit the model. Then we use TF-IDF method from sklearn.feature_extraction.text to turn text documents into numerical feature vectors with fit transform. We turn this result into an array to create a dense array from it. TF-IDF method provides us with the term frequency and inverse document frequency for the text. Finally, we transform X_train and X_test to vectorized train_X_Tfidf and test_X_Tfidf with the vectorizer we created with the TF-IDF method. After the vectorization, we have the row number, unique integer number of each word in the first row, and the calculated TF-IDF score of our predictors.

Explanation of the Chosen ML Model with Rationale

For our machine learning model, we decided to use the multinomial Naive Bayes model. Throughout our different attempts at data pre-processing and feature engineering to test various models on, we eventually settled on text classification as a method to base our model around. We had an inquiry that there was potentially valuable information that could be learned from the descriptions and comments, so we went in that direction. After testing a few models on our model-ready data, we found that Naive Bayes provided the strongest results. We believe this worked well because we treated each distinct word as a feature, and since there was a large number of features, the simplicity of Naive Bayes and the assumption of independence established it as a strong classifying model. Essentially, our model was based around the concept of "What is the conditional probability that a certain word is in the description given the class"? (Clickbait or Non-Clickbait). Interestingly enough, we ended up not using any of the comment data in our machine learning model because the description data proved to be sufficient.

Hyperparameter Settings

TfidfVectorizer() function from sklearn.feature_extraction.text:

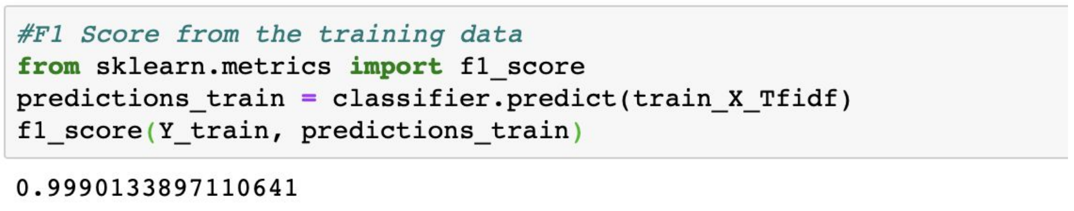
- min_df = 3
- sublinear_tf = True (Apply sublinear tf scaling)
- norm = 'l2'(default)
- ngram_range = (1,2)

cross_val_score() function from sklearn.model_selection:

- cv = 10

Since we have relatively large training data set, we decide to use k=10, while the value for k is fixed to 10, the result in a model skill estimate with low bias a modest variance.

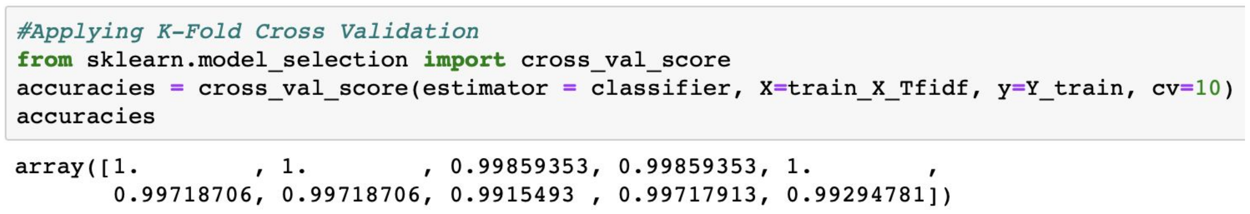
Performance Evaluation in Training



```
#F1 Score from the training data
from sklearn.metrics import f1_score
predictions_train = classifier.predict(train_X_Tfidf)
f1_score(Y_train, predictions_train)
```

```
0.9990133897110641
```

The best F1 score from the training data is 0.999.

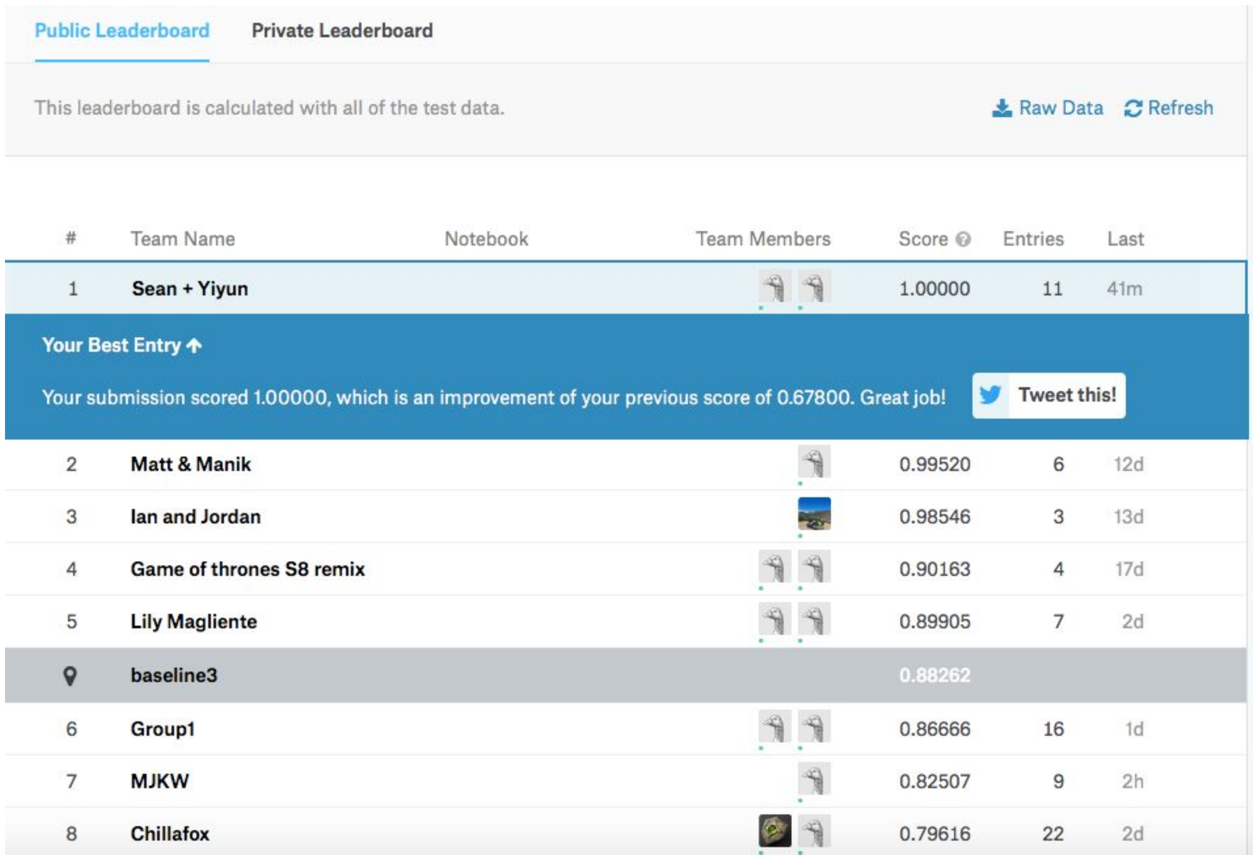


```
#Applying K-Fold Cross Validation
from sklearn.model_selection import cross_val_score
accuracies = cross_val_score(estimator = classifier, X=train_X_Tfidf, y=Y_train, cv=10)
accuracies
```

```
array([1.        , 1.        , 0.99859353, 0.99859353, 1.        ,
       0.99718706, 0.99718706, 0.9915493 , 0.99717913, 0.99294781])
```

While we performing K-Fold Cross Validation on the training set, we split the data into 10 groups and these are the results from these 10 groups.

Screenshot of the Leaderboard



This leaderboard is calculated with all of the test data.

#	Team Name	Notebook	Team Members	Score	Entries	Last
1	Sean + Yiyun			1.00000	11	41m

Your Best Entry

Your submission scored 1.00000, which is an improvement of your previous score of 0.67800. Great job!

#	Team Name	Notebook	Team Members	Score	Entries	Last
2	Matt & Manik			0.99520	6	12d
3	Ian and Jordan			0.98546	3	13d
4	Game of thrones S8 remix			0.90163	4	17d
5	Lily Magliente			0.89905	7	2d
	baseline3			0.88262		
6	Group1			0.86666	16	1d
7	MJKW			0.82507	9	2h
8	Chillafox			0.79616	22	2d

Appendix

Bedi, Gunjit. “A Guide to Text Classification(NLP) Using SVM and Naive Bayes with Python.”

Medium, Medium, 9 Nov. 2018, https://medium.com/@bedigunjit/simple-guide-to-text-classification-nlp-using-svm-and-naive-bayes-with-python-421db3a72d34.

Malik, Usman. “Text Classification with Python and Scikit-Learn.” Stack Abuse, Stack Abuse, 27 Aug. 2018, https://stackabuse.com/text-classification-with-python-and-scikit-learn/.

“Sklearn.feature_extraction.Text.TfidfVectorizer¶.” Scikit, https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html.

Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.