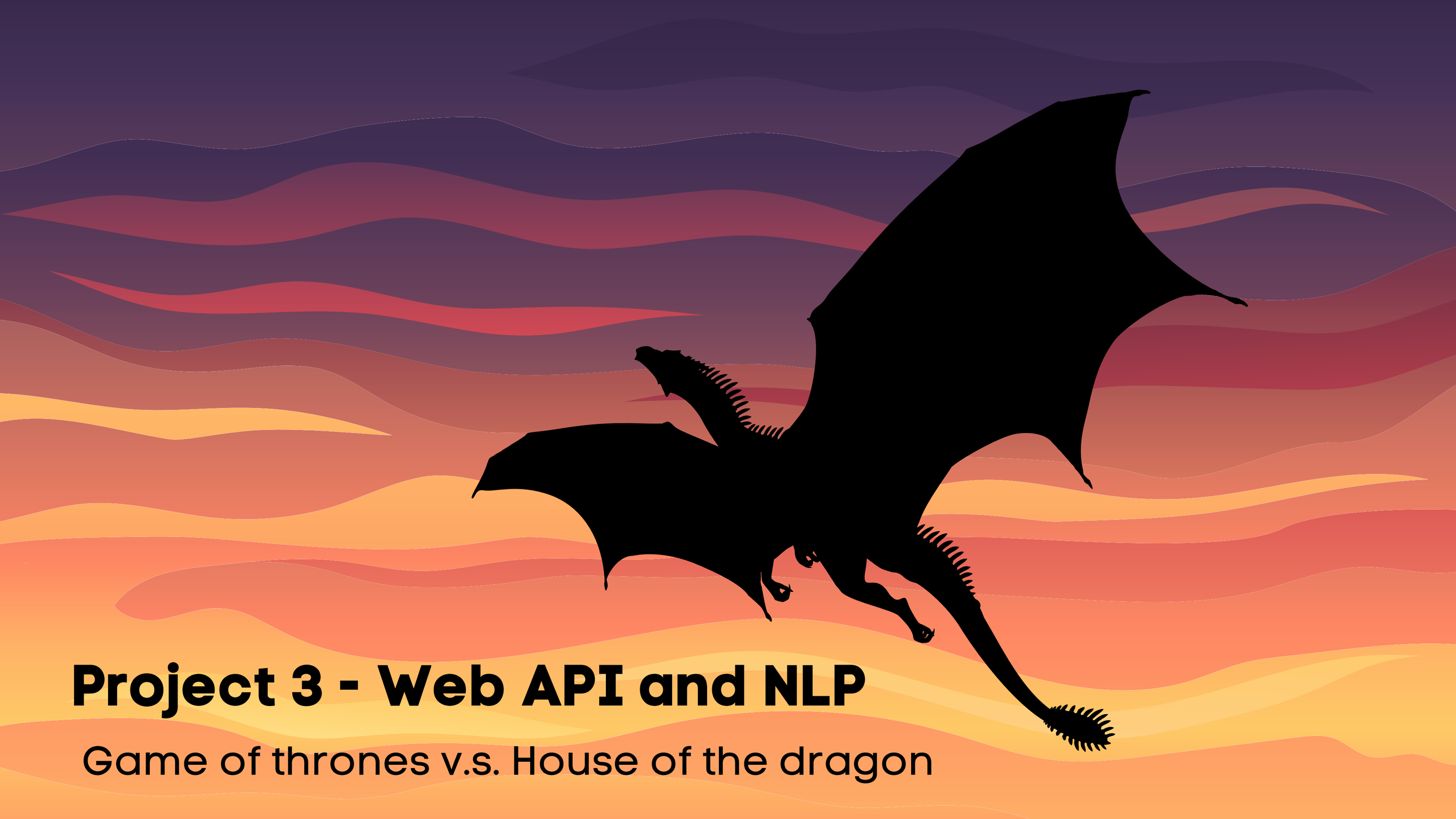


Project 3 - Web API and NLP

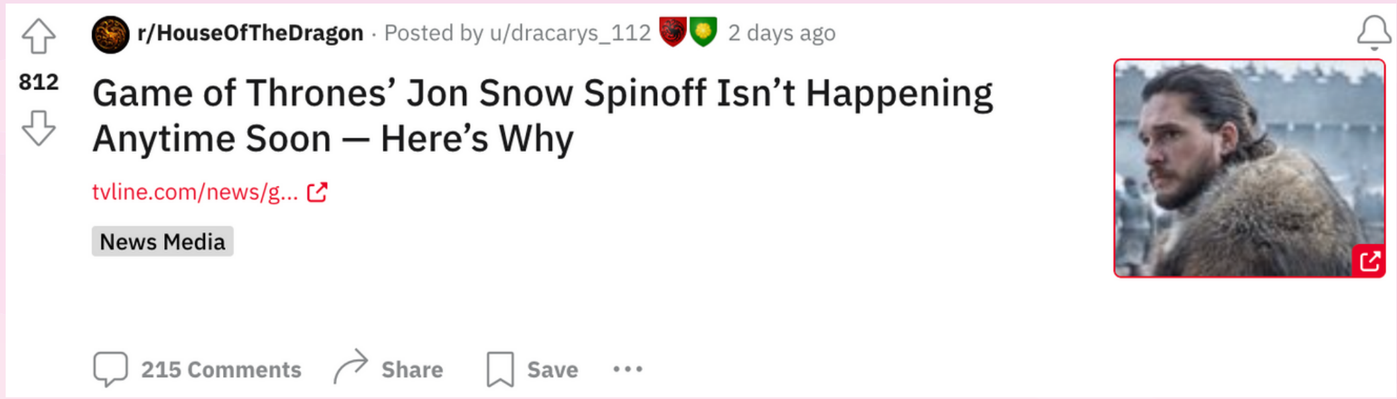
Game of thrones v.s. House of the dragon

Background

One of the greatest TV series of all time, "Game of thrones". The serie finale was in 2019. To continue on its success, the studio release the prequel "House of the dragon" in 2022. Game of thrones subreddit used to be on fire during the show was on air and the popularity has dropped once the time goes by.The same popularity happened to House of the dragon too.

Problem Statement

Because these two shows are basically in the same universe. Many of Game of thrones fans sometime create the post in House of the dragon subreddit, due to it is more recent and the forum is more active. As the moderator of these two subreddits, I want to develope tool to check if post is in the right subreddit.

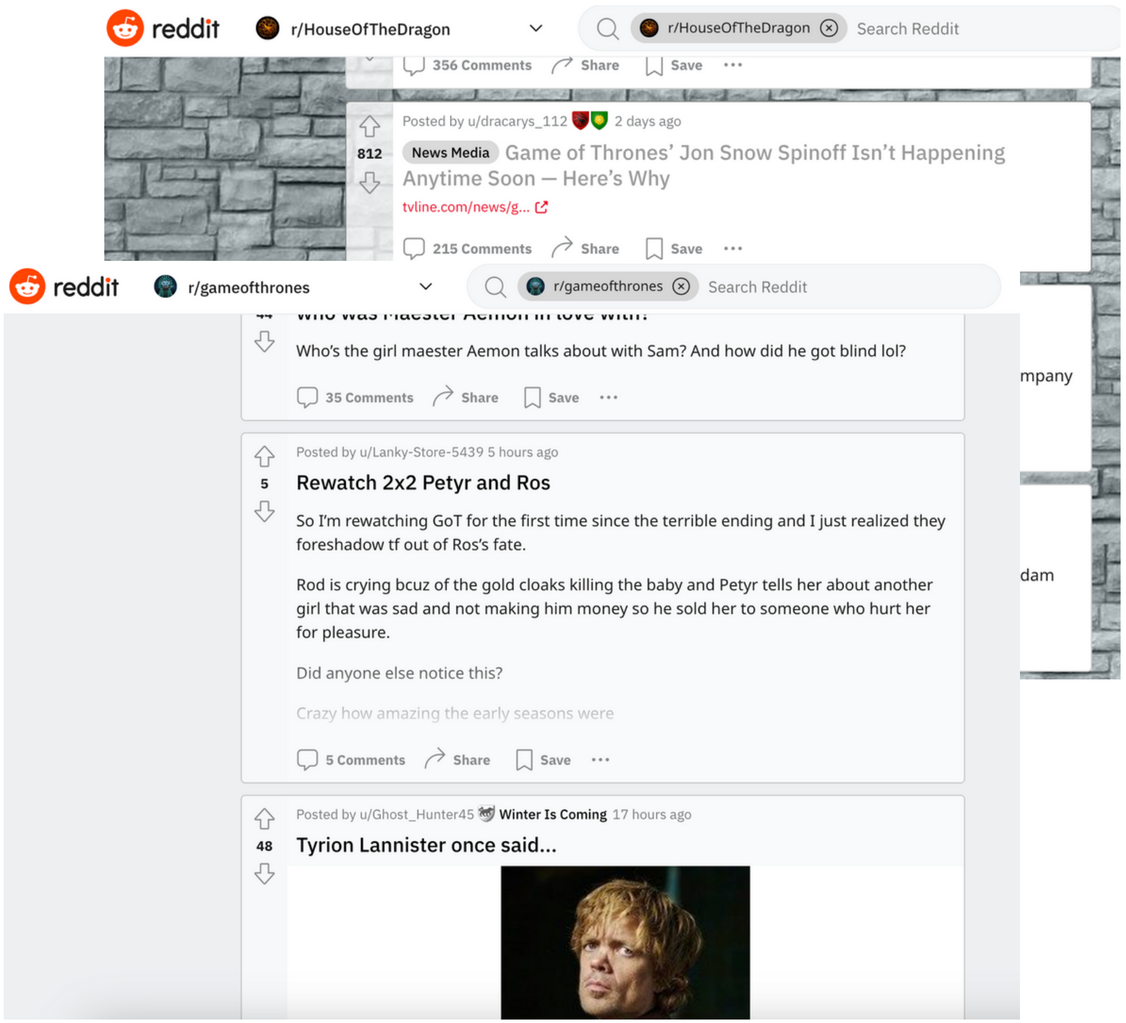


Approach


Collection
Cleaning
Exploratory Data Analysis
Model Building
Model Deployment

Data collection

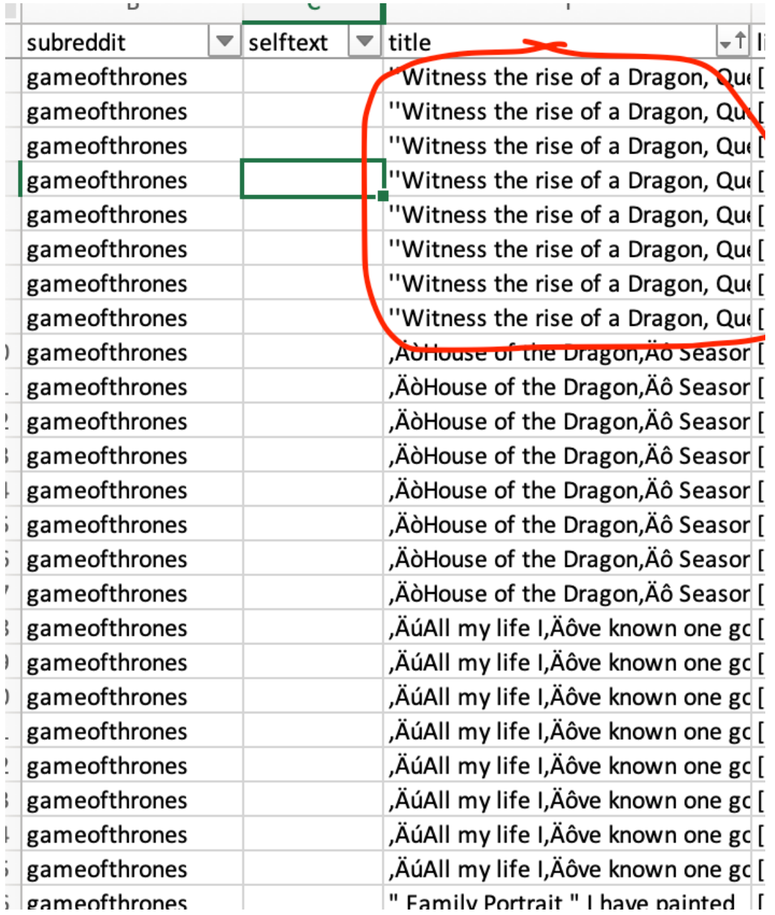
The data is collected via reddit API. The goal is to obtain around 7,5000 post for each subreddit, total 15,000 posts. The data is collected in a list of dictionary. Then convert to .csv file, so they're easily transfer and used in the next process.



Data cleansing

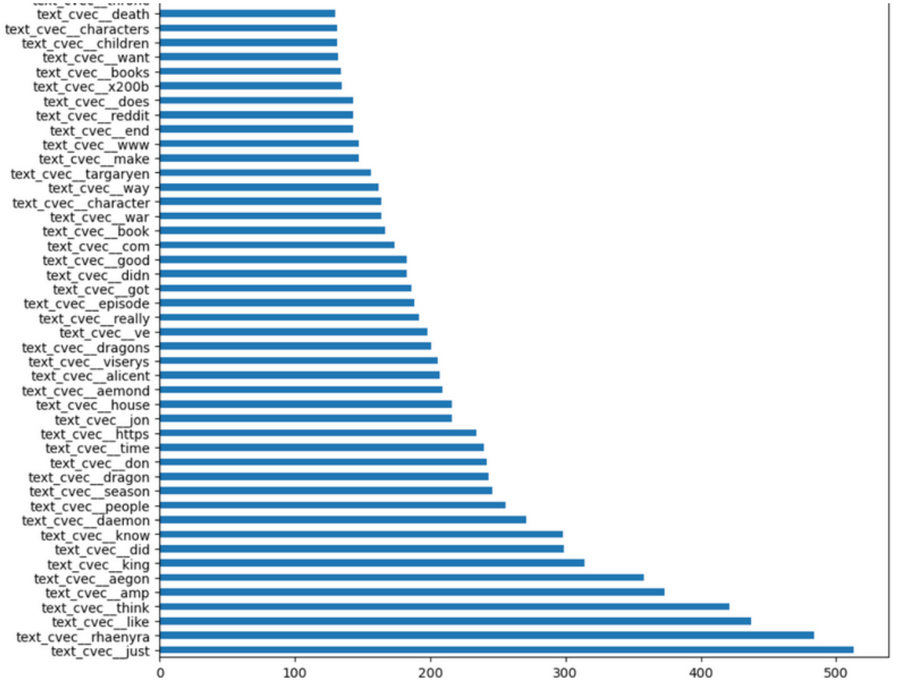
14,702 posts from both subreddits were collected. Unfortunately 12,808 of them were duplicated. and another 702 of them has no content of the post. Ending up we only have 1,191 posts left.

Column	Data type	Description
Subreddit	Boolen	1 is Game of thrones, 0 is House of the dragon
Title	string	Title of the post
Seltext	string	Content of the post



EDA

Baseline score is 50%
and I applied CounterVectorized and TfidVectorized to extract features from both title and selftext column. Then list the word with frequency more than 30 times. Then extract the unique names of each shows. Keep them as my custom stop word to be added to English stop word



Model building and optimization

Strategy:

Step 1

Use CounterVectorized and TfidVectorized together with pipeline to 2 set of 7 models:

- Logistic Regression
- kNN Classification
- Naive Bayes
- Bagging
- AdaBoost
- Support Vector Classification

Select 2 base on F1 score, since we need balance of sensitivity and specificity

Step 2

The 2 models will be further optimized by adding custom stop word, and Gridsearch on the best paramenter for of the model. Then select the model with best performance

Model building and optimization.... Result

	Test Accuracy	Train Accuracy	Test Precision	Train Precision	Test Sensitivity	Train Sensitivity	Test f1_score	Train f1_score
LogisticRegression_C	0.775168	0.982083	0.746988	0.975664	0.832215	0.988789	0.787302	0.982183
KNeighborsClassifier_C	0.637584	0.772676	0.643357	0.76129	0.61745	0.793722	0.630137	0.777168
NaiveBayes_C	0.798658	0.970885	0.820144	0.970852	0.765101	0.970852	0.791667	0.970852
Bagging_C	0.738255	0.978723	0.700565	0.975501	0.832215	0.982063	0.760736	0.978771
RandomForestClassifier_C	0.765101	1.0	0.730994	1.0	0.838926	1.0	0.78125	1.0
AdaBoostClassifier_C	0.711409	0.807391	0.652174	0.732993	0.90604	0.966368	0.758427	0.833656
support vector_C	0.775168	0.970885	0.732955	0.952586	0.865772	0.991031	0.793846	0.971429
LogisticRegression_T	0.785235	0.965286	0.767296	0.95207	0.818792	0.979821	0.792208	0.965746
KNeighborsClassifier_T	0.718121	0.833147	0.721088	0.857831	0.711409	0.798206	0.716216	0.826945
NaiveBayes_T	0.795302	0.969765	0.809859	0.964523	0.771812	0.975336	0.790378	0.9699
Bagging_T	0.724832	0.978723	0.716129	0.977629	0.744966	0.979821	0.730263	0.978723
RandomForestClassifier_T	0.755034	1.0	0.72619	1.0	0.818792	1.0	0.769716	1.0
AdaBoostClassifier_T	0.711409	0.819709	0.661538	0.745267	0.865772	0.970852	0.75	0.843233
support vector_T	0.785235	0.99888	0.757576	1.0	0.838926	0.997758	0.796178	0.998878

Step 1 result

AdaBoost model with CounterVectorized is the one with the least overfitting and SVC with TfidVectorized is the one with the best f1 score on test data

Model building and optimization.... Result

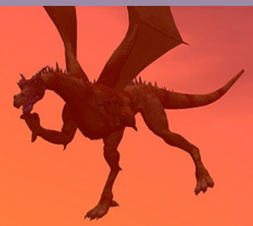
	Test Accuracy	Train Accuracy	Test Precision	Train Precision	Test Sensitivity	Train Sensitivity	Test f1_score	Train f1_score
AdaBoostClassifier_C	0.818792	0.996641	0.777778	0.993318	0.892617	1.0	0.83125	0.996648
support vector_T	0.684564	0.783875	0.629108	0.698587	0.899329	0.997758	0.740331	0.821791

Step 2 result

AdaBoost model score are improved, but it is more overfitting than prior optimization

SVC model score are lower, but now as bad, and the overfitting is much better

Conclusion



SVC model is my selection

The total test data of 298 posts, the model prediction performance:

- Accuracy for subreddit is 78%
- Precision is 63%
- Sensitivity is 90%
- F1 score is 74%

It is better than baseline