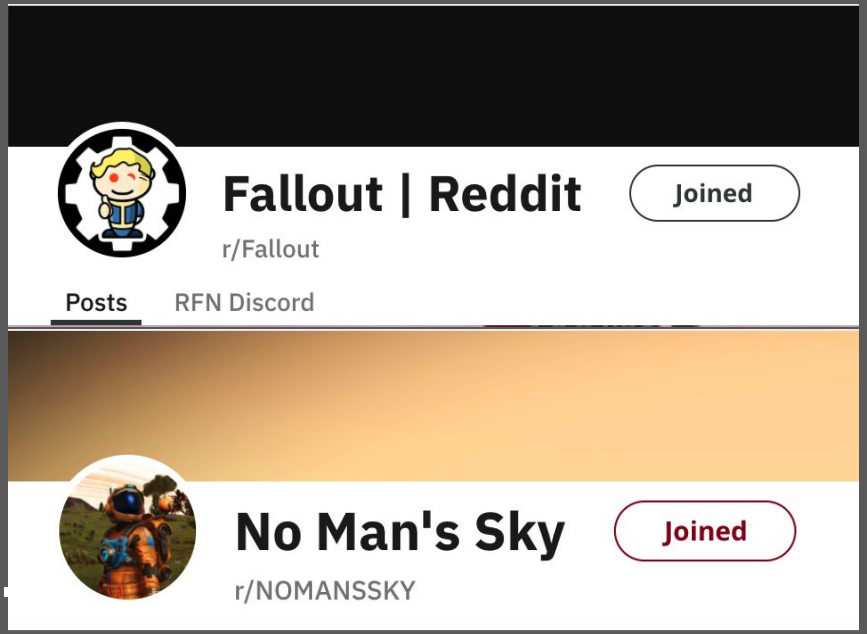
NLP and Classifying the Source of Reddit Publications

Problem Statement

With the datasets I've gathered, is it possible to accurately predict document classification, with the origin of the document being either the Fallout or No Man's Sky subreddits. Is there enough strength in the relationships between word choice in the Fallout subreddit to distinguish it from the No Man's Sky subreddit, and vice versa?

The Source: 2 Subreddits



https://www.reddit.com/r/Fallout/

https://www.reddit.com/r/NOMANSSKY/

Data Collection

Out of the 3 pathways I had, I chose to use PushShift since it works well with the reddit API and seemed like the straightforward way to aggregate the data I needed.

I started with day windows that were too large and missed data, and with n values that weren't sizable enough to gather the amount of data I needed. I added a print statement in the query_pushshift function to print the shape of the shape of the temp dataset, in order to see live results of the extraction. I then adjusted day window and n value until I was confident that I would be able to gather enough data. From the Fallout sub, I gathered over 14,000 submissions, and did the same for the No Man's Sky dataset. I wanted to make sure I had more than enough data to work with. I decided to focus on the selftext over the title, since I believe from the distributions of word counts that there would be more meaningful data found there.

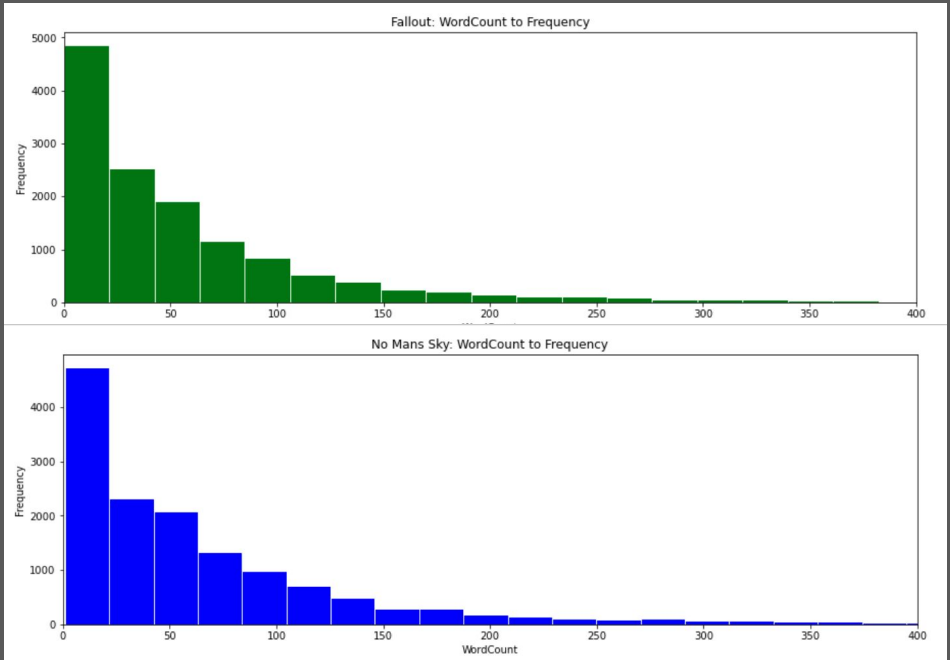
Data Cleaning

There were many decisions that had to be made in order to prepare the corpus for processing and classification. One of the main decisions I had to make was what to do with the significant amount of missing or null values, as well as the [deleted] or [removed] strings. One step further, there was a significant amount of undesired punctuation and strange characters.

I began by joining the two datasets together so I would only have to clean once. I used value counts to identify the most frequent documents and this is what led me to find a significant number the deleted and removed text.

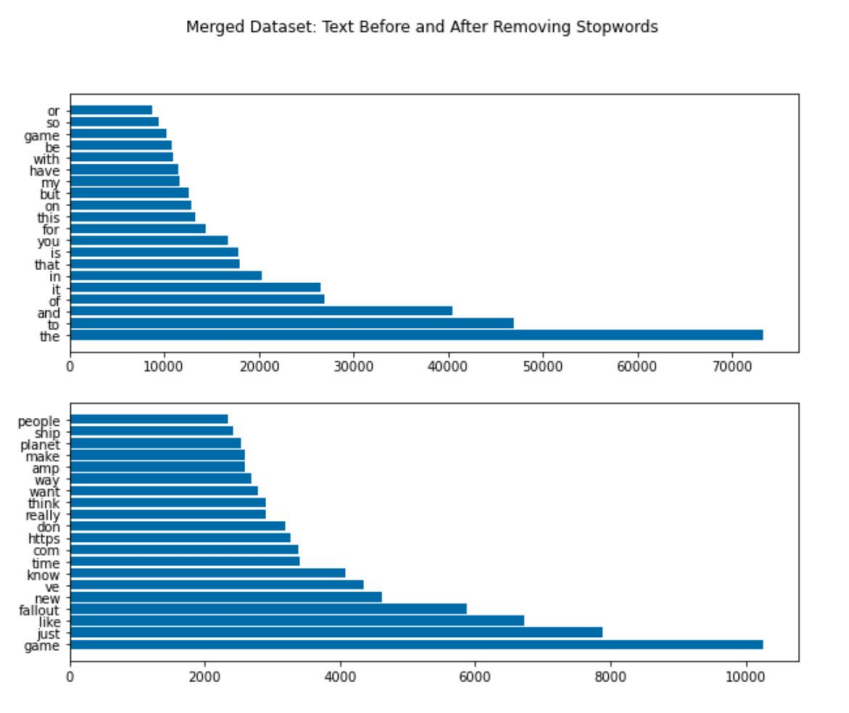
I mapped a function to the features title and selftext to substitute np.nan for the instances of undesired text. This allowed me to easily handle removing specific documents that seemed problematic. Then I used Regexp to further clean the data, and created a tokenized feature. I then created a lemtokenized feature that was lemmatized. This made testing original text against Regexp cleaning and Lemmatizaton much easier.

EDA



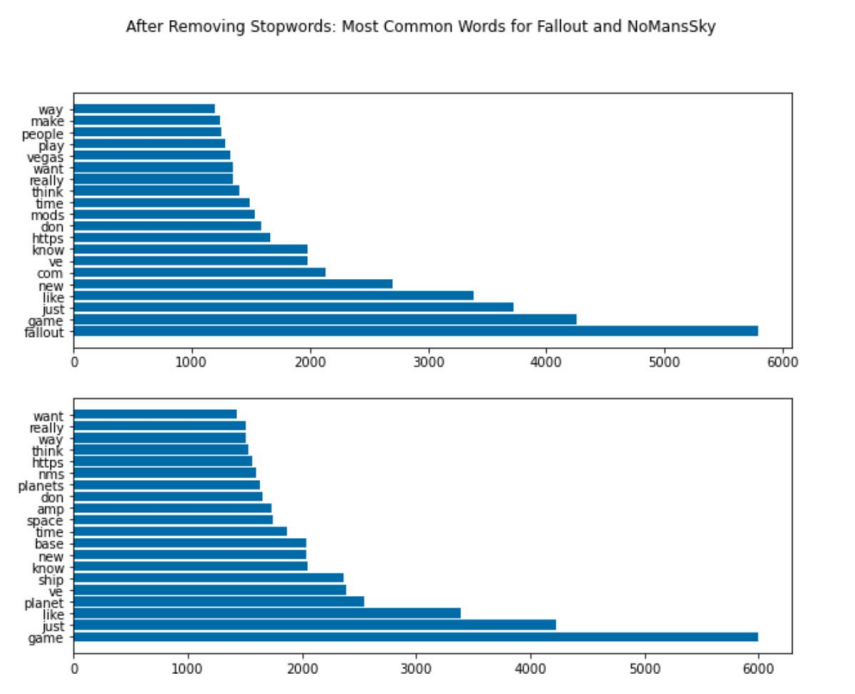
A majority of word count per post distribution shows that most users, regardless of subject matter, tend to write brief posts. This led me to explore actual word counts and average length of posts. I considered removing the numerous below 10 word posts, but decided against it. This data came to me the way it did, and I wanted to see how the classifiers would respond.

More EDA



Merged Dataset word count of most common words, before and after the removal of common stopwords. These words add no meaning or value really and this explains how important it is to remove stopwords from your documents.

Even More EDA



This visualization displays the most common words for the Fallout subreddit and the No Man's Sky subreddit after common stopword removal. I used this chart to inform me what stop words I still needed to remove.

Classifier Selection

Multinomial Naive Bayes

-Pros - Very fast / assumes all features are independent / works well with hyper dimensional data (text)

-Cons - Assumed feature independence reduces accuracy for speed

Logistic Regression models

-Pros - Most common binary classification algorithm, coefficients in a logistic regression model are interpretable, quick to fit and fast to generate predictions with.

Cons - Assumption of linearity between the dependent variable and the independent variables.

Tree Models

-Pros - Interpretability, Doesn't make assumptions, Regression and Classification

-Cons - Prone to overfit, Frequency / Use

Bag Classifier Ensemble model

-Pros - Wisdom of the crowd for decision trees / tends to outperform single decision trees

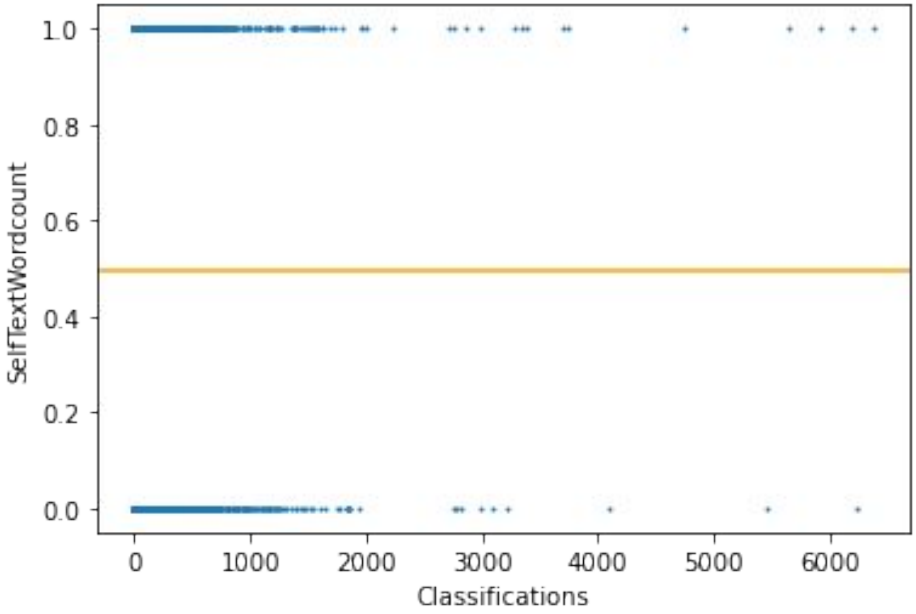
-Cons - Loss of Interpretability / Computationally Expensive

Baseline Accuracy

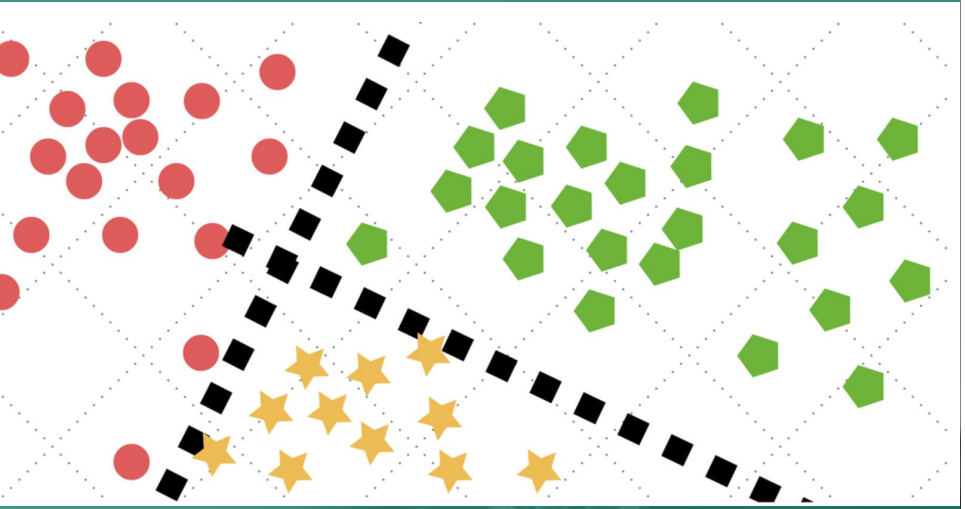
The baseline accuracy for the distribution of my classes were 0.5156, since that is the plurality class. Any model that doesn't perform significantly better than this baseline is relatively useless.

```
Baseline Accuracy is Predicting the Plurality Class

0    0.51563
1    0.48437
Name: subreddit, dtype: float64
```



Classifiers



Regarding the many types of classifiers, I implemented:

4 Multinomial Naive Bayes
2 Logistic Regression models
2 Tree Models
1 Bag Classifier Ensemble model.

Some models performed significantly better in some ways than in others.

(Higher scores / higher variance)
(Lower scores / reasonable variance)

I used Pipeline and Gridsearch for every model I assembled.

Vectorizers

Regexp(tokenizer)

CountVectorizer()

TfidfVectorizer()

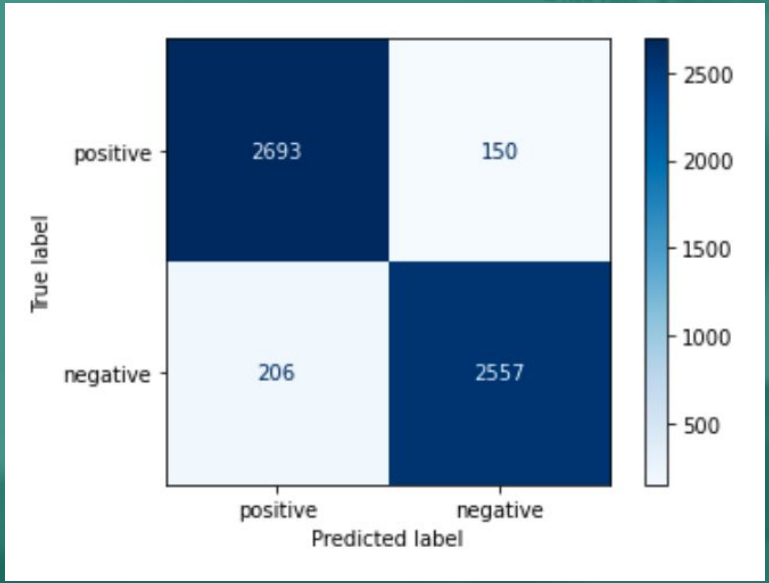
I tested out the Regexp for data cleaning and tokenizing, and then assembled the docs back into strings.

I tested CountVectorizer against TfidfVectorizer, and all the arrangements of selftext, Regexp cleaned, and Regexp cleaned and lemmatized.

I also cleaned the data with the clean strings function to test it out.

	selftext	subreddit	cleantext	cleantexttokenized	cleantexttokenizedlem	tokenized	lemtokenized
0	Idk but the whole "replace humans with synths ...	1	idk replace humans synths mysterious institute...	idk replace humans synths mysterious institute...	idk replace human synths mysterious institute ...	idk but the whole replace humans with synths f...	idk but the whole replace human with synths fr...

MultinomialNB



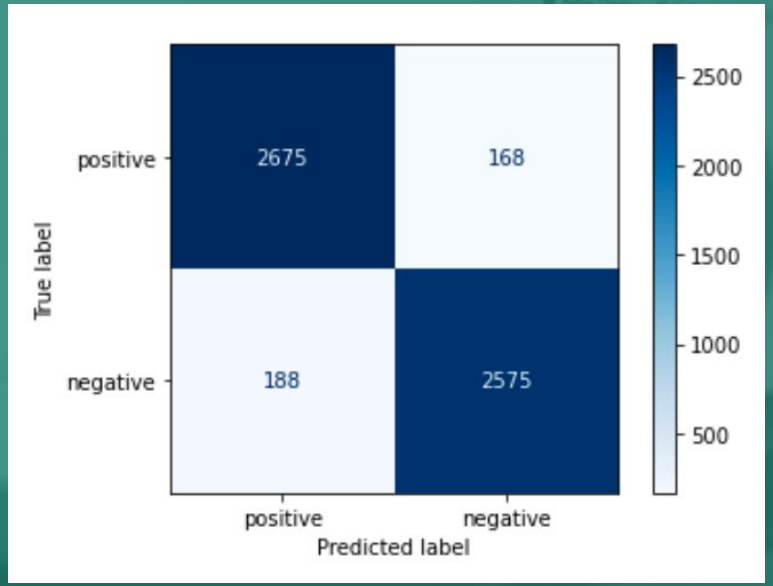
Naive Bayes Pipe Grid
CountVectorizer with Selftext

Best Scores:

Grid Search Best Score: 0.9341
Grid Search CV Train Score: 0.9411
Grid Search CV Test Score: 0.9364
Grid Search CV Sensititivity: 0.9199
Grid Search CV Precision: 0.9449
Grid Search CV F1: 0.9322
Grid Search CV Roc Auc: 0.9855
Specificity: 0.9472
TPR: 0.9199
FPR: 0.0527

These are cross validated scores.

MultinomialNB



Naive Bayes Pipe Grid
CountVectorizer with Selftext
Cleaned by RegExTokenized

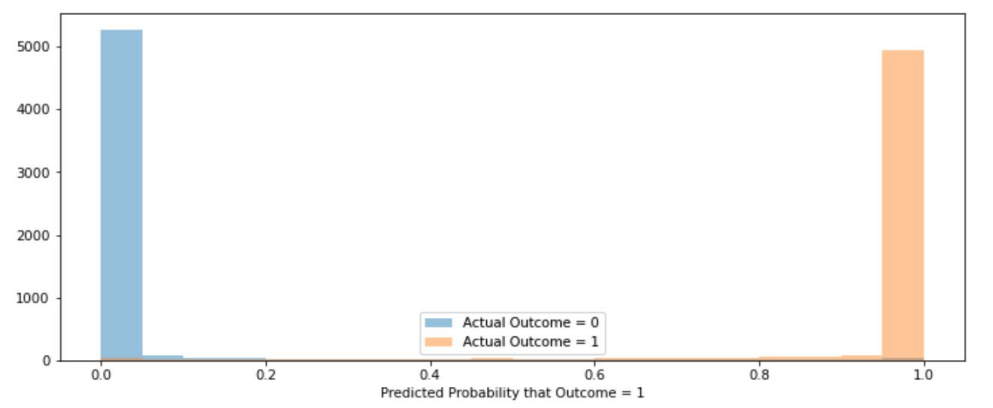
Besides the numerous hyper parameters to select from, I tested a range of other inputs as well.

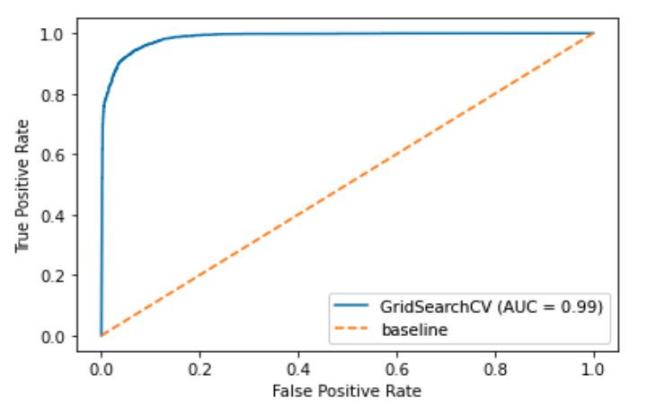
Variables:

Self Text()
RegexpTokenizer Text()
WordNetLemmatizer()
CountVectorizer()
TfidfVectorizer()
Numerous Hyper Parameters

MultinomialNB Classifier in Depth

Excellent ROC Curve, and the distplot for Actual Outcomes shows that there is nearly 0 overlap, visually demonstrating the models ability to separate the two classes. ROC AUC score is 0.9866.





Naive Bayes Model Evaluation

Comparison to Baseline Accuracy: Interpretation of Model 1
Baseline Accuracy for Plurality Class is 0.506953. Model 1 Naive Bayes significantly outperformed the Baseline Accuracy of 0.506953. Model 1 Naive Bayes scores at the top of the list with the highest performance.
For this model I used the CountVectorizer() in order to compare to TfidfVectorizer(). I used selftext to compare original data as X versus cleaned and lemmatized data as X. The cleaned data performed best in the classifier.

Model 1 is relatively low bias since the train accuracy and test accuracy scores are pretty great. The model is picking up on meaningful data and assembling proper relationships. Accuracy of 0.9364 on cross validation test is satisfactory. Model 1 has almost no variance since it predicts consistently between test and train scores. This shows the model is correctly adapted to the training data. The larger the difference between the train cross validation score and the test cross validation score, the more positive one can be that the model is high variance. Since there is only a slight difference between the scores,, I am confident the model doesn't show signs of major bias or a significant level of variance. All the other classification scores are satisfactory..

Other Model Evaluation

Compared to baseline accuracy, all of my classifiers outperformed the baseline. The Naive Bayes models consistently did better than the rest of the models I tested.

The logistic regression models weren't too far behind at around 0.90 cross validated accuracy score. These models were a bit more susceptible to variance, with 0.96 train score, and 0.915 test score for the first logreg model, and 0.97 train score and 0.92 on the test score. Here is a perfect illustration of the Bias-Variance trade off. Generally speaking, they are inversely related, meaning that increasing variance lessens bias, and vice versa. This model has some bias, and is overfitting to the training data and doesn't perform as well on the testing data.

The single decision trees scored suboptimally across the board, however they were significantly low variance, since the difference between cross validated train (0.714 and test scores(0.719) is hardly noticeable.

The bagging classifier performed similarly to the decisions trees, which would make plenty of sense since they learn off of decision trees errors. Scores were in the low 70s, some bias, unnoticeable variance due to the tight range of scores. Predictions are consistently in the same range, they just aren't great predictions.

Summary / Evaluation

Through a thorough and methodological approach, I believe that I have managed to build a classifier that can accurately predict the origin of text posts and state what subreddit the text came from. I tested a total of 9 model variations, but the main classifiers I tested were Multinomial Naive Bayes through Pipe / Gridsearch (4), Logistic Regression through Pipe / Gridsearch (2), Decision Tree through Pipe / Gridsearch (2), and (1) Bagging Classifier Ensemble through Pipe / Gridsearch. My objective was to accurately predict the origin of a submission and assign it to either the Fallout subreddit with 1, or the No Man's Sky subreddit with 0, with above 0.90% confidence. With the first Multinomial Naive Bayes classifier, I reached 0.94 and 0.93 for the cross validated train score and the cross validated test score.

Recommendations

As for the rest of the classifiers, generally speaking they all performed rather well. I was disappointed to see the decisions trees perform suboptimally, but still, every model managed to outperform the baseline accuracy being the plurality class in this case.

Through the implementation of these models above, and the exploration of the many hyperparameters assigned to each model, I would strongly suggest moving forward with the Multinomial Naive Bayes classifier through Pipeline / Gridsearch. The hyperparameters below served me well.

Best Estimator: Pipeline(steps=[('cvec',CountVectorizer(max_df=0.85, max_features=7000, min_df=3)),('nb', MultinomialNB())])

Sources

Classifiers Photo - by Mohammed Hemayed | Oct 6, 2018 | Mohammed Hemayed, The Data School