# Authorship Identification using Reuter_50_dataset

CMPE 255 Fall 21 Team project Report
San Jose State University, CA


Team 8:

| | |
|---|---|
| Adarsh Narasimha Murthy | 014952275 |
| Anuhya Gankidi | 015897323 |
| Deepak Vellore Karunamoorthy | 014655628 |
| Sai Harsha Anirudh Garre | 015218996 |



## Abstract

Authorship identification is to identify the author of an article based on the attributes describing their writing style. In this project, we try to find suitable techniques and models that we can apply at Article level from the Reuter_50_50 dataset to find the author of a given text. Methods to clean the dataset, Pre-processing functions and vectorization are applied on the data. Then different classification algorithms and models that are applied on the dataset are evaluated to get the best suited algorithm for Author identification.


## Introduction:

The goal of this project is to extract information from Reuter_50_50's C-50 dataset that contains 50 Authors, which has over 50000 attributes, to recognize patterns in texts. Two Approaches, Feature Engineering and Feature Selection are implemented. Stylometric features, top topics and geographic features are scaled and fed to supervised classifier in the first approach. In Approach 2, each of the training and test batches has 2500 texts, 50 per author, that are nonoverlapping. For training and test datasets, Term Frequency -Inverse Document Frequency is applied for every author. K-nearest neighbor, Logistic Regression, Naïve Bayes Classifier, Support Vector Machine, Random Forest, Bagging and AdaBoost on the dataset. A comparison of the findings of each of these models is carried out using various analytical measures, and a future scope for the project's implementation is offered.

[Source code: https://github.com/Adarsh3thy/CMPE255-Project]

## Approach 1

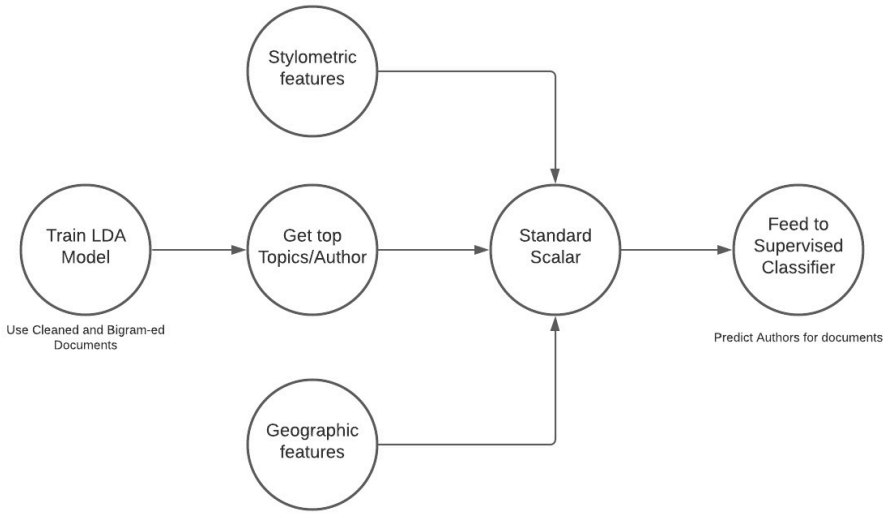
**Stylometry**:
The quantitative analysis of literary style using computational distant reading tools is known as stylometry.[1] It is based on the observation that authors tend to write in a similar, recognizable, and distinct style.
We used the library https://github.com/neomatrix369/nlp_profiler.git@master to generate the stylometric features, in addition to our own functions to extract the below features:

| Stylometric characteristics | |
|---|---|
| sentences count | noun phrase count |
| characters count | English characters count |
| repeated letters count | Non-English characters count |
| spaces count | syllables count |
| chars excl spaces count | sentiment polarity score |

| | |
|---|---|
| repeated spaces count | sentiment polarity |
| whitespaces count | sentiment polarity summarized |
| chars excl whitespaces count | sentiment subjectivity score |
| repeated whitespaces count | sentiment subjectivity |
| count words | sentiment subjectivity summarized |
| duplicates count | spelling quality score |
| emoji count | spelling quality |
| repeated digits count | spelling quality summarized |
| whole numbers count | ease of reading score |
| alpha numeric count | ease of reading quality |
| non alpha numeric count | ease of reading summarized |
| punctuations count | Average Sentence length |
| repeated punctuations count | Yules Characteristic K |
| stop words count | dates count |
| Average Word Frequency Class | Brunets Measure W |
| Shannon Entropy | Syllables Count |

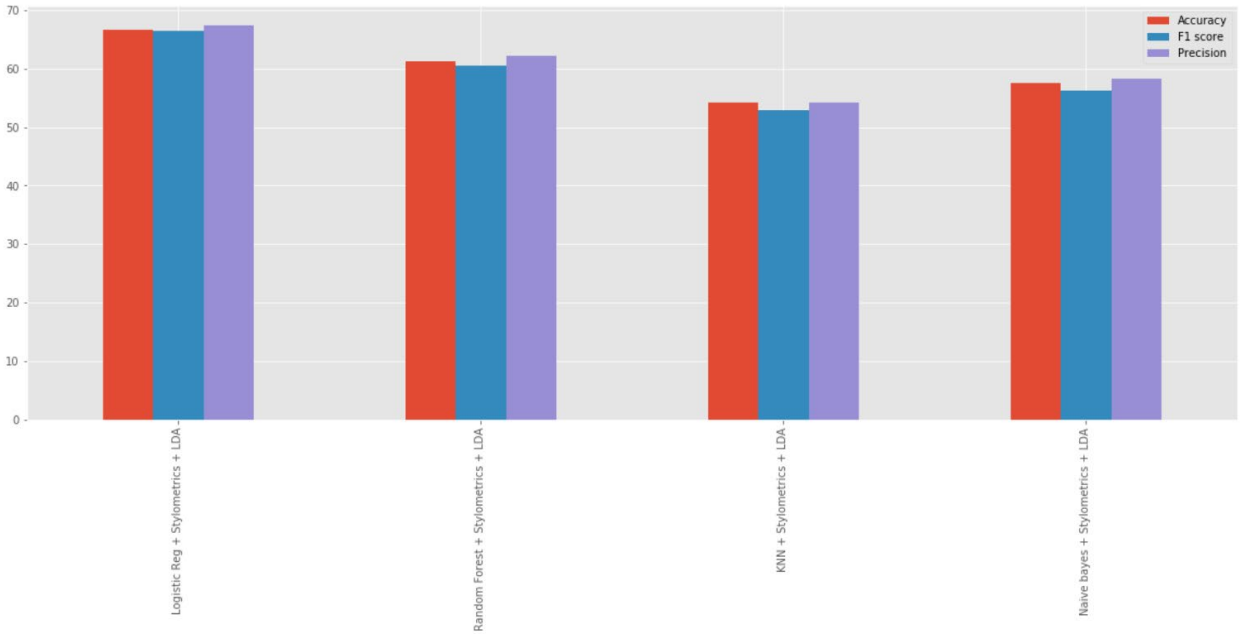
**Latent Dirichlet Allocation (LDA):**
Latent Dirichlet Allocation technique states that each document is a combination of topics. Once LDA is applied on the data, words that make the topic surface. Cleaned and bi-gramed Documents are sent to the train the LDA Model which generate hot topics per each author that are fed to the classifier as one of the three inputs.

**Geographic Features**: The dataset consists of news articles, and journalists tend to report based on a specific geographic area. Hence, we have taken around 100 geographic locations also as an input to supervised classifier. These inputs, stylometry features, top topics and geometric features are then sent to standard scalar before they are fed to a supervised classifier.




We experimented with KNN, Naïve Bayes, Logistic Regression and Random Forest. Logistic Regression gave the best accuracy of 67%. The details about each of the algorithm is available in page 6.

| | Logistic Reg + Stylometrics + LDA | Random Forest + Stylometrics + LDA | KNN + Stylometrics + LDA | Naive bayes + Stylometrics + LDA |
|---|---|---|---|---|
| **Accuracy** | 66.76 | 61.32 | 54.24 | 57.60 |
| **F1 score** | 66.51 | 60.54 | 52.99 | 56.30 |
| **Precision** | 67.39 | 62.24 | 54.26 | 58.33 |




## Approach 2

## Data Preprocessing:

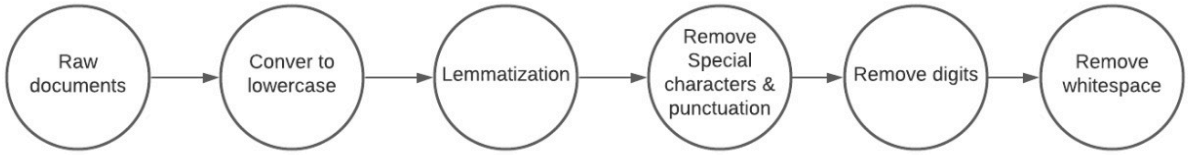
The following preprocessing steps are taken to clean the raw data.

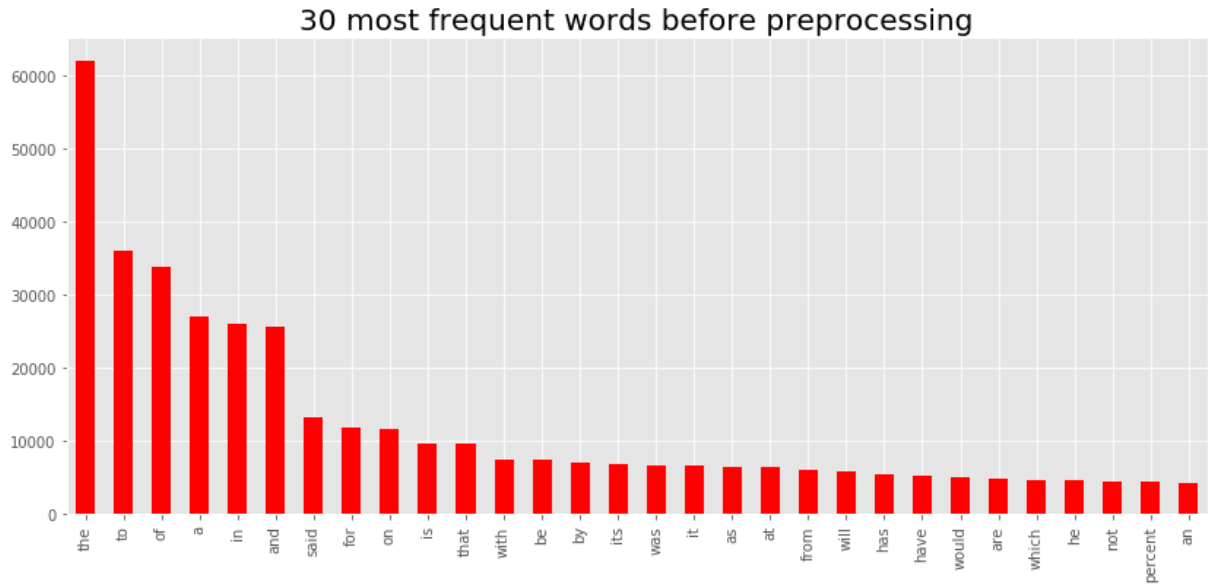
1. Remove white space and convert to lowercase: Remove all extra white spaces in the document and convert all upper-cased characters to lower case.

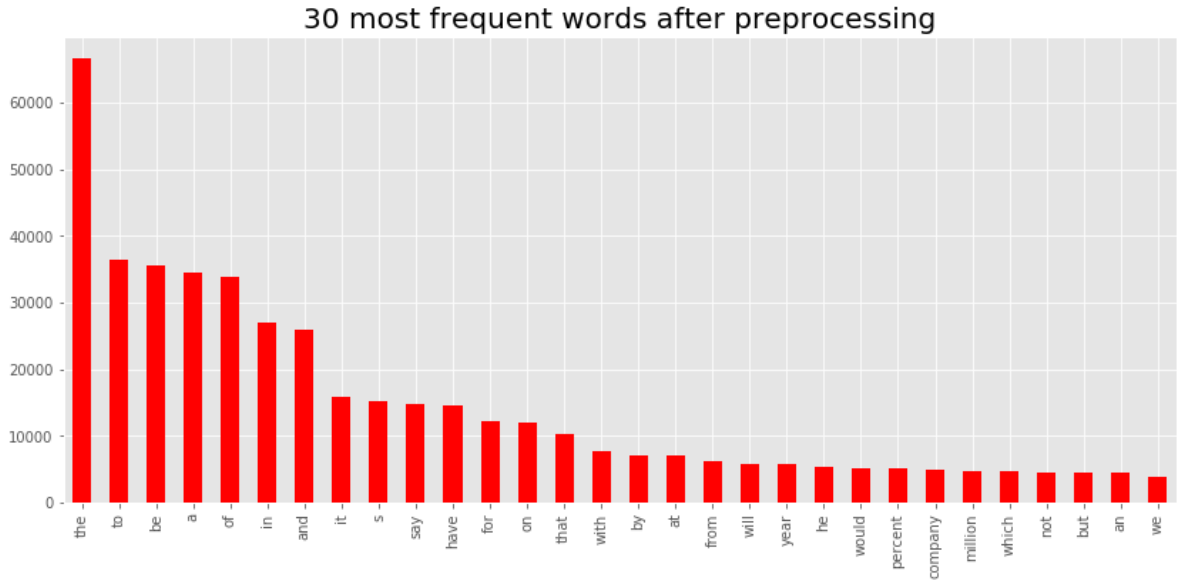
2. Remove Punctuation marks, special characters, and digits: Remove all punctuation marks, special characters, and numbers from text since do not contribute to movie sentiment. The punctuation list from string library is used here

3. Stop words: Retaining the stop words increased the accuracy of the text by 1%. Hence, they have been retained.

4. Lemmatization: Lemmatization is the process of converting a word to its base form. (Eg: liked to like). Stemming is another way obtaining the base word; however, stemming does not provide meaningful base forms always (eg: caring is converted to car). Hence, lemmatization is performed on the text corpus using Wordnet library along with positional tags




1. **Bag of Words:** In a text corpus, the Bag of Words (BoW) model represents the frequency of word occurrences. The sequence in which words appear in the text is irrelevant to the bag of words; it just cares about which words appear in the text. The text in our data frame will be converted into a bag of words

model, which will include a sparse matrix of numbers. Each word's number of occurrences will be counted and printed. For this, we'll utilize the Scikit-learn library's count vectorizer.

Hyperparameters:

| | |
|---|---|
| Vectorizer | CountVectorizer |
| Ngrams | Unigrams, bigrams, and trigrams |
| max_features | 10000 |

2. **TF- IDF TF-IDF (term frequency-inverse document frequency):** is a statistical measure that assesses the relevance of a word to a document in a set of documents. Each word is given a score to indicate its prominence in the document. Term Frequency, Inverse Document Frequency is abbreviated as TF-IDF. The TF-IDF calculates the importance of a word in a document and across the full corpus. The count of each word in a document divided by the total number of words in the document is called term frequency.

   **(Number of times the term w appears in a document) / TF(w) (total number of words in the document)**

   The IDF is a metric for determining the importance of a word based on its frequency throughout the corpus. It determines the importance of a word in the corpus.

   **IDF(w) = log (total number of documents divided by number of documents containing w)**

   Finally, we multiply these two components – TF and IDF – to get TF-IDF.

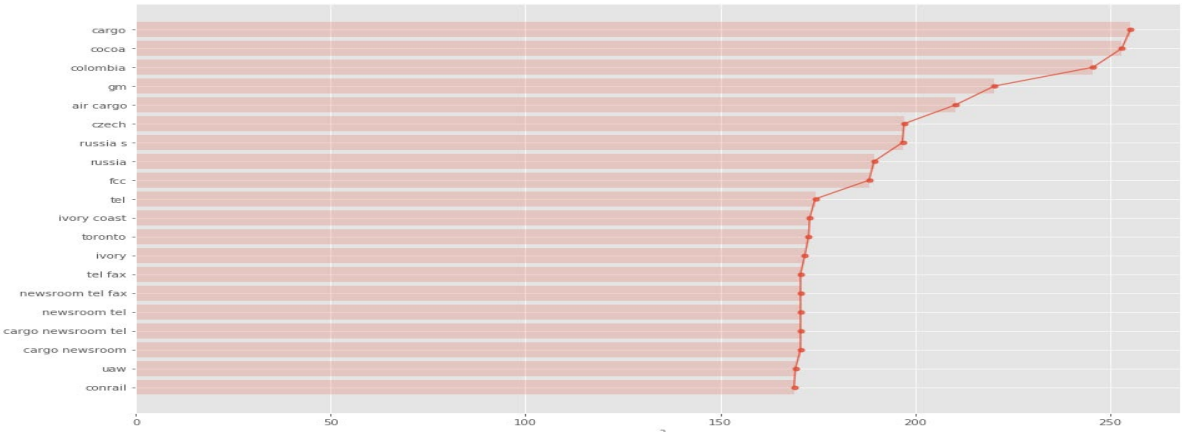
   **IDF(w) x TF(w) = TF-IDF(w) (w)**

   We considered unigrams, bigrams and trigrams for TF-IDF

   Hyperparameters:

| | |
|---|---|
| norm | L2 |
| Ngrams | Unigrams, bigrams, and trigrams |
| max_features | 15000 |
| Analyzer | word |

3. **Chi Square**: The chi-squared statistic assesses the degree of independence between a feature (in this case, a single term inside a review) and its classification (whether the reviews are positive or negative). If a feature has a high chi-squared score when compared to other features, it is useful for class prediction. For example, below are the top useful features in the given dataset:



## Dimensionality Reduction (SVD)

Dimensionality reduction is the process of reducing the amount of input variables for a predictive model.
SVD: We use singular value decomposition TF-IDF vector to reduce the number of dimensions to 3000 from 10000. However, this did not yield the best accuracy, but it did reduce the computational time required to predict while maintaining the same accuracy as the vectorizer.
We considered 2000 n_components for SVD

## Classification Algorithms:

We implemented below algorithms on the dataset to identify the author and evaluated each of them using measures: Accuracy, precession, and F1. For all the algorithms the optimal hyperparameters were found using GridSearchCV

### Classical Algorithms

1. **K-Nearest Neighbor:**
   The KNN algorithm is a versatile classification algorithm. This will be used as a benchmark for other algorithms. We will use KNN with the default distance metrics (Euclidean), Manhattan and cosine using scikit learn. The optimal k value was 30.

| | cosine | euclidean | manhattan |
|---|---|---|---|
| **Count Vectorizer** | 41.92 | 35.44 | 13.84 |
| **TF-IDF** | 63.16 | 63.16 | 15.44 |
| **TF-IDF + CHI2** | 61.6 | 61.08 | 34.24 |
| **SVD** | 62.8 | 62.24 | 11.48 |

F1 Score:

| | cosine | euclidean | manhattan |
|---|---|---|---|
| **Count Vectorizer** | 42.67 | 36.21 | 13.99 |
| **TF-IDF** | 62.08 | 62.08 | 17.96 |
| **TF-IDF + CHI2** | 60.66 | 60.51 | 36.45 |
| **SVD** | 61.73 | 61.13 | 9.04 |

Precision:

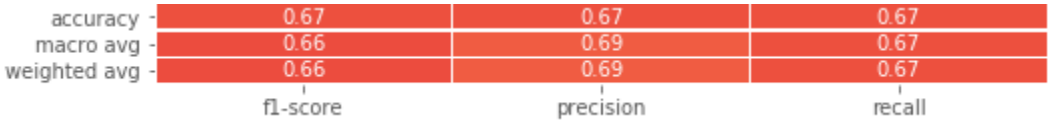
| | cosine | euclidean | manhattan |
|---|---|---|---|
| **Count Vectorizer** | 54.34 | 52.85 | 50.44 |
| **TF-IDF** | 64.98 | 64.98 | 50.89 |
| **TF-IDF + CHI2** | 62.7 | 64.2 | 64.57 |
| **SVD** | 64.28 | 64.26 | 27.87 |

As is evident above, TD-IF and TF-IDF along with SVD have the best accuracy, precision and recall.

2. **Naïve Bayes**

Based on the training data for each authorship class, this classifier creates a probabilistic model for that class. Then it calculates and multiples the probabilities among all authors to determine who is the most likely author. Some of the word probabilities may yield zero, to avoid this we will use the Laplace correction parameter. Naive Bayes assumes strong (naive) independence assumptions between the features. Hence, it's not appropriate to apply it on SVD dataset. We will apply it only on TF-IDF vector.

There could be some words with zero probability, so offset we have used the Laplace parameter by setting alpha to 1.0 in MultinomialNB()

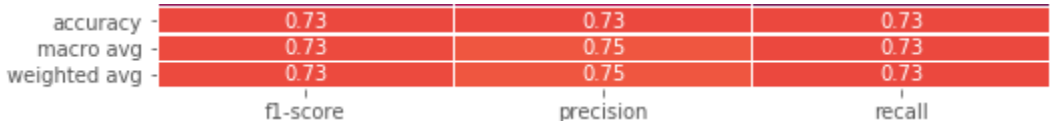



3. **Support Vector Machine**

SVM tries to find a line that separates the instances of two classes by the widest margin. SVM can cope with datasets containing many attributes and does not require feature selection. SVM has been shown to give very high levels of accuracy.

Hyperparameters:

| | |
|---|---|
| probability | True |
| kernel | sigmoid |
| gamma | 1 |
| C | 100 |

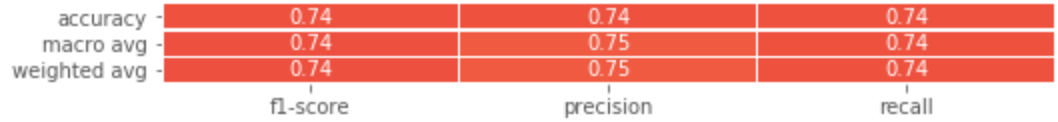



4. **Logistic Regression**

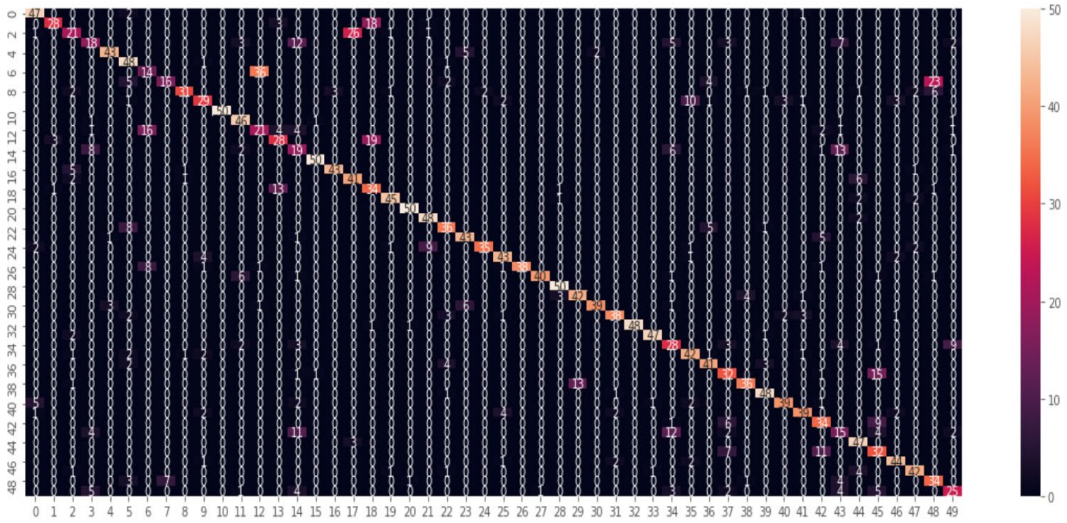
The input variables are mapped to categorical response/dependent variables using a logistic function in logistic regression. Unlike Linear Regression, this method returns a probability between 0 and 1. Logistic Regression, on the other hand, evaluates the probability of a binary result rather than predicting it. Here, the optimal parameters for Logistic regression have been determined using GridSearchCV.

HyperParameters:

| Name | Value |
|---|---|
| Penaly | 'l2' |
| Max_iter | 100 |
| C | 1000 |
| Solver | Newton-cg |




Confusion Matrix for LR:



## Ensemble Methods

1. **Random Forest**

   Random forest creates training datasets with replacement using bootstrapping, and it also chooses a set of features (without replacement) to optimize randomization on each training dataset. The square root of the total number of features is usually used to determine the number of features to be selected.

   Hyperparameters:

| | |
|---|---|
| n_estimators | 100 |
| max_depth | 50 |
| Max_features | 1000 |


2. **Bagging**

   Bootstrap Aggregating - "Bagging" combines various estimators that utilize the same algorithm but have been trained on distinct subsets of the data.

   Hyperparameters:

| Name | Value |
|---|---|
| base_estimator | base_model |
| n_estimators | 50 |
| n_jobs | 3 |

| | | | | |
|---|---|---|---|---|
| accuracy | | | 0.64 | 2500 |
| macro avg | 0.66 | 0.64 | 0.63 | 2500 |
| weighted avg | 0.66 | 0.64 | 0.63 | 2500 |

3. **AdaBoost**

Boosting employs progressive learning, which is an iterative procedure aimed at reducing the preceding estimator's mistakes. It is a sequential strategy in which each estimator improves predictions by relying on the prior estimator. For each n estimator, AdaBoost uses the whole training dataset with a few key changes.

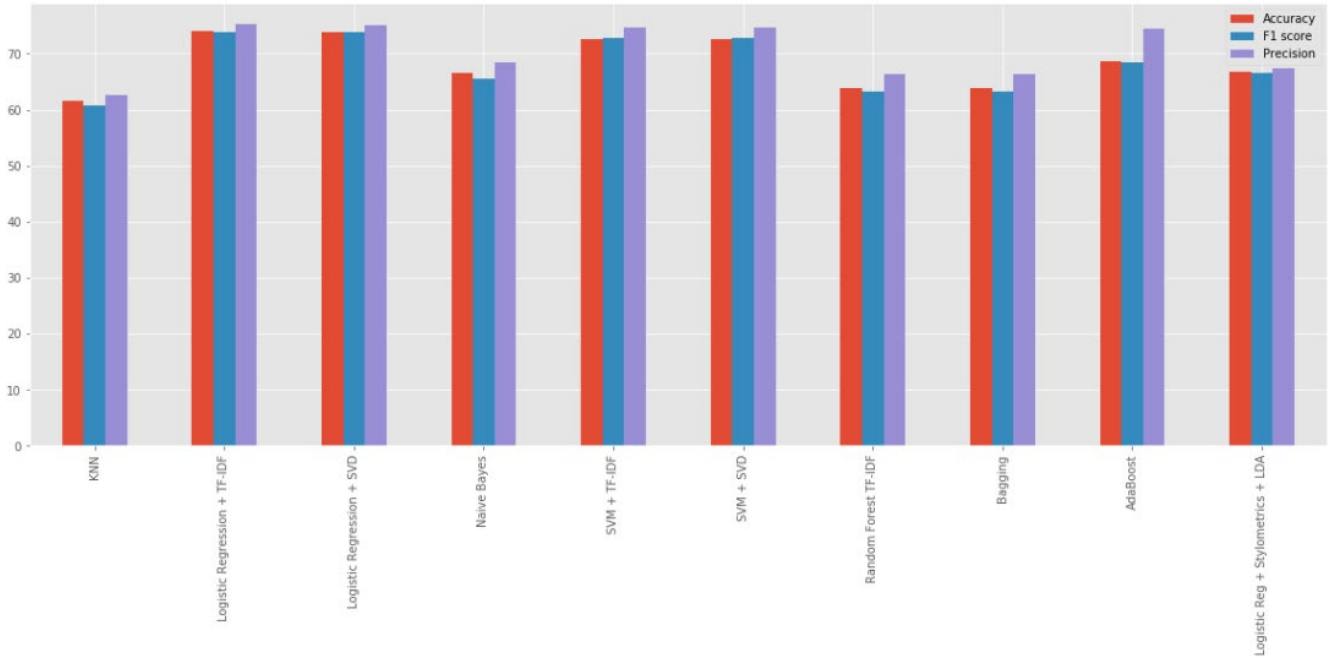
Hyperparameters:

| Name | Value |
|---|---|
| base_estimator | base_model |
| n_estimators | 100 |
| random_state | 42 |

| | | | | |
|---|---|---|---|---|
| accuracy | | | 0.69 | 2500 |
| macro avg | 0.75 | 0.69 | 0.68 | 2500 |
| weighted avg | 0.75 | 0.69 | 0.68 | 2500 |

Comparison between all the implementations from Approach 1 and 2:

| | KNN | Logistic Regression + TF-IDF | Logistic Regression + SVD | Naive Bayes | SVM + TF-IDF | SVM + SVD | Random Forest TF-IDF | Bagging | AdaBoost | Logistic Reg + Stylometrics + LDA |
|---|---|---|---|---|---|---|---|---|---|---|
| Accuracy | 61.60 | 73.96 | 73.88 | 66.52 | 72.56 | 72.60 | 63.84 | 63.84 | 68.52 | 66.76 |
| F1 score | 60.66 | 73.83 | 73.74 | 65.52 | 72.68 | 72.72 | 63.21 | 63.21 | 68.40 | 66.51 |
| Precision | 62.70 | 75.26 | 75.14 | 68.50 | 74.54 | 74.58 | 66.28 | 66.28 | 74.51 | 67.39 |



## Conclusion

- The best accuracy and F1 score of 75% were achieved through Logistic Regression using TF-IDF as feature selection.
- Applying SVD on TF-IDF did not improve the accuracy of Logistic Regression, though it made the computation faster.
- Using Ensemble methods with Logistic Regression also did not improve the accuracy beyond 75%
- Support Vector Machine provided the next best accuracy of 73%.
- We also tried applying deep learning techniques [6] like CNN, LSTM and BART, but due to the high computational requirements of these methods, the results were either unable to converge or provided very low accuracy.

## References:


[1] https://programminghistorian.org/en/lessons/introduction-to-stylometry-with-python
[2]https://www.researchgate.net/publication/221655968_NGram_Feature_Selection_for_Authorship_Identification
[3] https://www.researchgate.net/publication/323631302_Chat_biometrics
[4] https://towardsdatascience.com/a-machine-learning-approach-to-author-identification-of-horror-novels-from-text-snippets-3f1ef5dba634
[5] https://www.researchgate.net/publication/221246703_Visualizing_Authorship_for_Identification
[6] https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1174/reports/2760185.pdf