Project name: Horror Author Recognition
Group: Horror Text Miners

- David Medina (F11115117)
- Kevin Da Rosa (F11115108)
- Jaime Colman (F11115107)

Dataset chosen: Spooky Author Identification

1. Introduction

The following document is a guideline through our process of analyzing the dataset, performing text classification on it, and finding the best performing model. The task is to determine the author of a given text based on its vocabulary, and writing style. The process involves steps, such as: preprocessing, feature extraction, TF-IDF, and text classification.

2. Analyzing the Dataset

The labeled dataset in train.csv, is a collection of text from various authors, such as: Edgar Allan Poe (EAP), H.P. Lovecraft (HPL), and Mary Wollstonecraft Shelley (MWS). The file consists of 19579 rows, and 3 columns, in format of | id | text | author |. The distribution of texts by author is: EAP has 7900, HPL has 5635, and MWS has 6044.

3. Preprocessing the Data

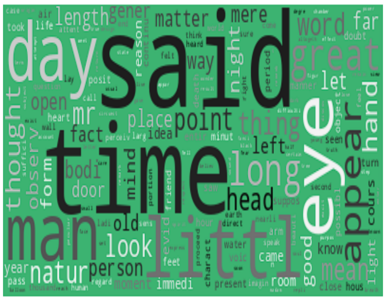
Before any feature extraction, or text classification technique is applied, the texts in the 'text' column, must be preprocessed. First, lower case the texts, and tokenize by using the 'nltk' library. Second, we remove stopwords, the stopwords removed are those in the 'gensim' library. Also, all words of less than 3 character length are ignored. Then, we remove apostrophes, perform stemming by using the PorterStemmer, and remove punctuations.

4. Feature Extraction

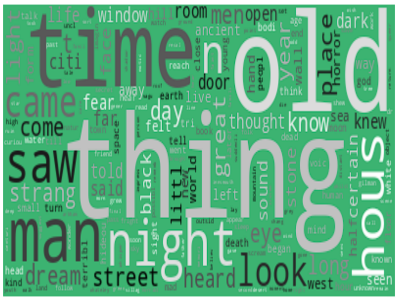
We believe that each author has its own vocabulary and way of writing, and that we can use those characteristics to classify them. So, to capture the characteristics of each text, two features are extracted: term frequency and the sentiment analysis score of each sentence.

a) **Term Frequency:** calculates the frequency of each word in a text through TF-IDF. This approach captures the importance of each word within the corpus, and also, helps to normalize the impact of high frequency occurring words, thus, providing a more balanced representation. The resulting vocabulary size of TF-IDF is 14232. To better visualize that, we decided to show WordClouds of term frequency for each author. As you can see, MWS uses positive words such as: love, life, heart, hope, etc. That clue led us to add Sentiment Analysis as another feature.

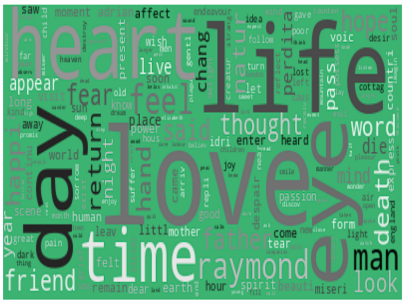
EAP



HPL



MWS



b) **Sentiment Analysis:** perform sentiment analysis on each text row to capture the emotion of the author. For the analysis, the SentimentIntensityAnalyzer from 'nltk' library was used, where a score greater than 0.05 is labeled as 'positive', in between 0.05 and -0.05 is labeled as 'neutral', and lesser than -0.05 is labeled as 'negative'.

5. Data Partition

All preprocessed texts from train.csv are split into two sets: one for training the model, and another for validation. The split is 80% training, and 20% validation. This results in a matrix of 15663 texts/rows, and 14240 features/columns, that will be input to train the models.

6. Evaluation Metrics

To evaluate the effectiveness of the model, we used two metrics:

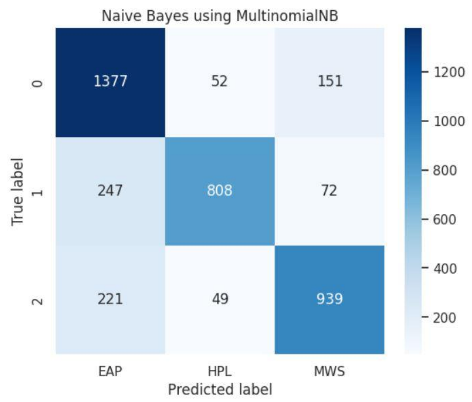
a) **Cross-validation (CV):** is a technique to evaluate performance where data is split into multiple k-subsets. Data is splitted as described in part 5., and this process iterates on different subsets to provide more generalization ability to the model. The higher the score of Cross-validation, the more accurate the model is. We used a 5-fold CV.

b) **Confusion matrix:** is a metric that provides breakdown of the model performance. By providing true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN), when labeling the authors of a text. From this matrix, further metrics are derived, such as precision, recall, and F1-score. All this is done with the 'sklearn' library.

7. Text Classification Models

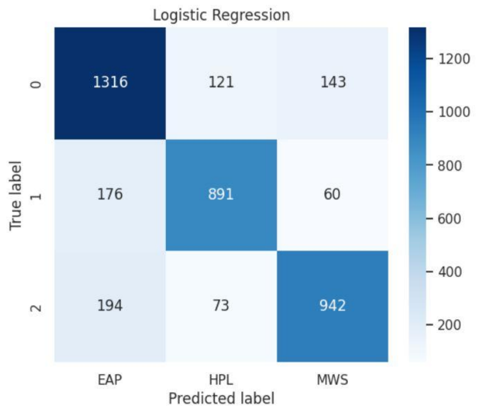
The following models used the 'sklearn' library with their default parameter values.

a) **Multinomial Naive Bayes:** is a probabilistic classification model that assumes independence between features, and estimates the conditional probabilities of each class given the features. Results:



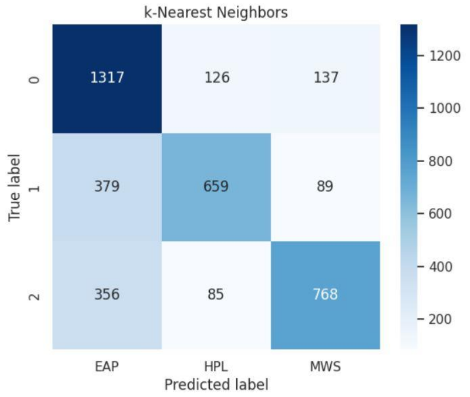
- CV accuracy: 0.798
- Micro average of …
  - Precision: EAP(0.75), HPL(0.89), MWS(0.81)
  - Recall: EAP(0.87), HPL(0.72), MWS(0.78)
  - F1-score: EAP(0.80), HPL(0.79), MWS(0.79)
- Macro average of …
  - Precision: 0.81
  - Recall: 0.79
  - F1-score: 0.80

b) **Logistic Regression:** is a linear classification model that predicts the probability of an instance belonging to a particular class. It models the relationship of input variables and multiclass target labels using a logistic function. Results:



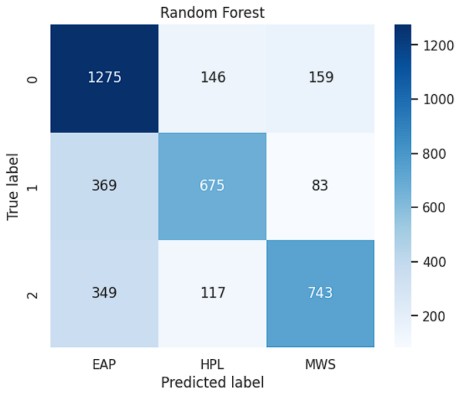
- CV accuracy: 0.804
- Micro average of …
  - Precision: EAP(0.78), HPL(0.82), MWS(0.82)
  - Recall: EAP(0.83), HPL(0.79), MWS(0.78)
  - F1-score: EAP(0.81), HPL(0.81), MWS(0.80)
- Macro average of …
  - Precision: 0.81
  - Recall: 0.80
  - F1-score: 0.80

c) **k-NN:** is a non-parametric method that classifies instances based on their similarity to k nearest neighbors in the training set. parameter n_neighbors=500, and also, without the Sentiment Analysis feature. Because, Sentiment Analysis increases the space features, and the data points become more sparse (curse of dimensionality). Result of k-NN without Sentiment Analysis:



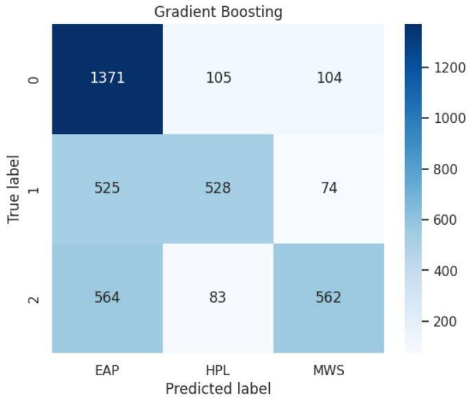
- CV accuracy: 0.703
- Micro average of …
    - Precision: EAP(0.65), HPL(0.76), MWS(0.76)
    - Recall: EAP(0.83), HPL(0.58), MWS(0.64)
    - F1-score: EAP(0.73), HPL(0.66), MWS(0.70)
- Macro average of …
    - Precision: 0.72
    - Recall: 0.69
    - F1-score: 0.70

d) Random Forest: consists of a collection of decision trees where each tree is constructed using a random subset of training data and a random subset of the features. The final prediction is an average of all individual tree predictions. Result:



- CV accuracy: 0.692
- Micro average of …
    - Precision: EAP(0.64), HPL(0.72), MWS(0.75)
    - Recall: EAP(0.81), HPL(0.60), MWS(0.61)
    - F1-score: EAP(0.71), HPL(0.65), MWS(0.68)
- Macro average of …
    - Precision: 0.70
    - Recall: 0.67
    - F1-score: 0.68

e) Gradient Boosting: combines weak learners, generally decision trees, into a strong predictive model. Builds an additive model by iteratively training new weak learners to correct the mistakes made by previous ones. Result:



- CV accuracy: 0.629
- Micro average of …
    - Precision: EAP(0.56), HPL(0.74), MWS(0.76)
    - Recall: EAP(0.87), HPL(0.47), MWS(0.46)
    - F1-score: EAP(0.68), HPL(0.57), MWS(0.58)
- Macro average of …
    - Precision: 0.69
    - Recall: 0.60
    - F1-score: 0.61

8. Conclusion

After trying several Traditional Machine Learning Models, we found out that Logistic Regression was the most accurate in terms of text classification with a CV score of 0.804. We got the best result after adding the feature of Sentiment Analysis for all models, except for k-NN, where the resultant accuracy was significantly worse. We think this may be due to the curse of dimensionality. So, you should be careful about the features you add, and how they affect your particular model. A possible improvement over this project would be to apply hyper parameter tuning to each model, to find the best parameters for best performance.