# Spooky Author Identification

## Purpose, Motivation, and Description

"It was a dark and stormy night!" These words may come from many or may be similar to one. In the spirit of Halloween, the second capstone will see which three popular horror story authors wrote certain text pieces.

The purpose is identification. The algorithm is to train and transform a natural language processing model using different algorithms and parameters. This model will derive from a training set that has classified the text and associated author. From there, it will work with a test set, records of text, to guess which author wrote it. This is a classical supervised learning algorithm, classifying text for multiple classes.

The approach is to use different tools and parameters. Over practice, it will start with the basic natural language processing techniques, using Naive Bayes and Logistic Regression, to classify and predict. But it will move towards advanced concepts such as random forest, hyperparameter tuning, and maybe XGBoost.

## Stakeholders

The business problem is to identify the appropriate author to an unknown text. This will explore appropriate classification and specify time and accuracy to show which methods were able to perform the task best.

This will lure linguists, communication experts, and people interested in classifying unknown text with their source. Using natural language processing to categorize information by text may inspire creating complicated methods to obtain the same data.

## Data

### Acquisition

The data source comes from Kaggle.com (https://www.kaggle.com/c/spooky-author-identification) , which hosted a $25K competition to create learning algorithms to classify and predict the author. They were downloadable as three separate CSV files. The train and test files were the most important; the other was an sample submission.

### Management

The **training dataset** has three attributes for 19,579 observations or documents (19,579 x 3).

- id (six-character id, 'id4021', primary index, text)
- author (three-character abbreviation, three choices: 'EAP', 'HPL', 'MWS'*, text, target response)
- text (text, see exploratory analysis for more metrics, target data)

The **test dataset** has two attributes for 8,392 documents (8,392 x 2).

- id (six-character id, 'id4021', primary index, text)
- text (text, see exploratory analysis for more metrics, test data)

## Cleansing

Originally the data is clean. No missing values were present and the classes in the training set are properly identified. However, the text does contain unnecessary words that are used once or take up a lot of computation. Cleaning the text is essential. Also, a new variable, length, was computed from the text, to serve as part of exploratory data analysis. In modeling, stop_words (or the lack thereof) played a role in parameter tuning.

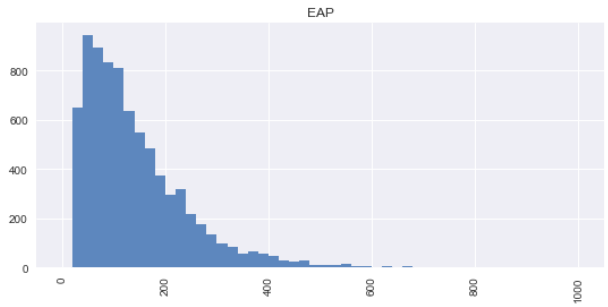
## Exploratory Data Analysis

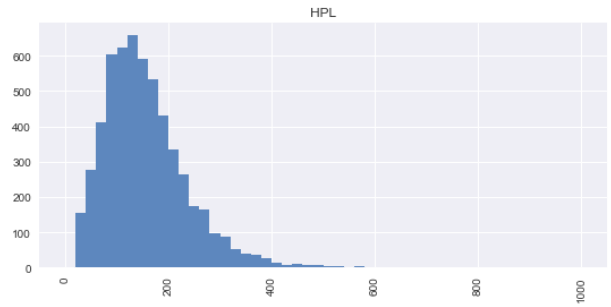
Before diving into the machine learning algorithm, what does the data look like? Below are count and length stats.

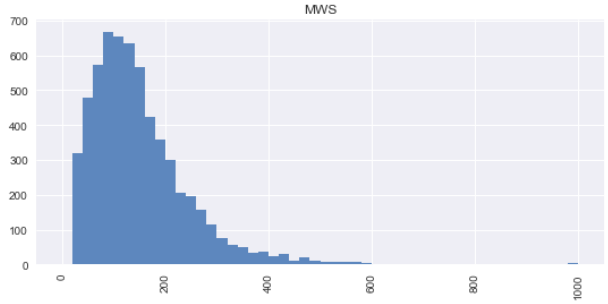
| Author | TOTAL | EAP | HPL | MWS |
|---|---|---|---|---|
| **count** | 19,579 | 7,900 | 5,635 | 6,044 |
| **mean** | 149 | 142 | 156 | 152 |
| **std** | 107 | 106 | 82 | 126 |
| **min** | 21 | 21 | 21 | 21 |
| **25%** | 81 | 68 | 98 | 84 |
| **50% / median** | 128 | 115 | 142 | 130 |
| **75%** | 191 | 186 | 196 | 192 |
| **max** | 4,663 | 1,533 | 900 | 4,663 |

Quickly, EAP has the most prose in the training set; prose by Edgar Allen Poe may appear more. However, the training set is uniformly distributed because the maximum count difference between EAP and HPL is about 2,265. While length plays an trivial role, no prose will be smaller than 21 words. The average words per prose is 149, with Mary Shelley having 4,663 words in one prose. This provides an idea as to how the features (or grams) will grow.

Below is a visualization representation of the length frequency per author. Horizontally is the length of the prose. Vertically is the frequency that the prose had the respected word. HP Lovecraft's prose has a less steep ascent than Edgar Allen Poe's work but over 80% of the author prose will occur within 250 words. This is sufficient to provide a lot of features and combinations to make a prediction. This may make our algorithms compute faster.




## Feature Selection

As mentioned, the training set had only three variables whereas the testing had two. The choice of features are:

- The training data X is the text prose
- The target response y is the authors
- The testing data X_test is the text prose from test.csv

The text trains the model, creating the necessary features and 'bag of words.' This model will predict an author who wrote an arbitrary prose. Please note the testing data does not have any indication as who wrote this, so the metrics alongside a comparison of probabilities will help determine whether the test would do well.

## Modeling

### Strategy

And with text classification, passing words will not work, so it must undergo a transformation into a document-term matrix (or DTM). In addition, which classifier to use and what metrics to look at? For this milestone report, the focus was on two experiments:

The first experiment focuses on the following:

- Split the training set into a training group and a test group
- Vectorize the training set
- Tune on the algorithms for the best parameters
- Fit and predict with the best parameters
- Evaluate the metrics
- Identify the author using the model

The second experiment is slightly different. Rather than splitting the data first, it will tune to find the best parameters for the chosen algorithms, vectorize second, and then split the data.

- Tune on the algorithms for the best parameters
- Vectorize the training set
- Split the training set into a training group and a test group
- Fit and predict with the best parameters
- Evaluate the metrics
- Identify the author using the model

This report focused on one vectorizer and two classifiers.

- vectorizer: CountVectorizer()
- classifiers: MultinomialNB(), LogisticRegression()

This provided FOUR trials.

- Experiment 1 with MultinomialNB()
- Experiment 1 with LogisticRegression()
- Experiment 2 with MultinomialNB()
- Experiment 2 with LogisticRegression()

## Parameter Tuning

Tuning parameters were in GridSearchCV, with CountVectorizer() and one classifier. The parameters were:

| SCI-KIT LEARN | Parameters* |
|---|---|
| `CountVectorizer()` | `max_df = [0,1,2,3,4,5]` When building vocabulary ignore terms that have a document frequency strictly lower than the given threshold.<br><br>`ngram_range = [(1,1), (1,2), (1,3), (2,2), (2,3), (3,3)]` (The lower and upper boundary of the range of n-values for different n-grams to be extracted. All values of n such that min_n <= n <= max_n will be used.) |
| `MultinomialNB()` | `alpha=[0.05, 0.1, 1.0, 2.0]`<br><br>additive smoothing parameter |
| `LogisticRegression()` | `C=[0.05, 0.1, 1.0, 2.0]`<br><br>Inverse of regularization strength, small C larger regulation; smoothing parameter |

* Descriptions come from sci-kit learn documentation.

For the CountVectorizer, stop_words and ngram_range were important, as they may tune to better performance and get better accuracy. The classifiers have smoothing parameters that may get better accuracy. Also note that to go through five to ten parameters would increase the processing time so to focus on two significant parameters was essential. The parameters 'binary' and 'stop_words' were best defaults when cross validated.

For each of the experimental runs, the best parameters are shown below.

| | Wall time | Best score | Vectorizer | Classifier |
|---|---|---|---|---|
| **Experiment 1 w/ Multinomial NB** | 16min 15s | 0.844 | vect__ngram_range: (1, 2)<br>vect__max_df: 0.5 | nb__alpha: 0.1 |
| **Experiment 1 w/ Logistic Regression** | 22min 20s | 0.812 | vect__ngram_range: (1, 1)<br>vect__max_df: 0.75 | logreg__C: 1.0 |
| **Experiment 2 w/ Multinomial NB** | 6min 25s | 0.859 | vect__ngram_range: (1, 2)<br>vect__max_df: 0.5 | nb__alpha: 0.1 |
| **Experiment 2 w/ Logistic Regression** | 7min 56s | 0.814 | vect__ngram_range: (1, 1)<br>vect__max_df: 0.75 | logreg__C: 2.0 |

## Evaluation

### Metrics

Evaluation took two forms: first was the performance metrics such as accuracy score and confusion matrix. The other was the identification based on the model. Here, the metrics of interest were the confusion matrix and the classification report. Below are the metrics based on experiment and run:

**Confusion Matrix**

**Experiment 1**

**MultinomialNB()**

| | | y_pred_class | | |
|---|---|---|---|---|
| | | EAP | HPL | MWS |
| y_true | EAP | 1670 | 132 | 113 |
| | HPL | 104 | 1209 | 53 |
| | MWS | 201 | 68 | 1345 |

**LogisticRegression()**

| | | y_pred_class | | |
|---|---|---|---|---|
| | | EAP | HPL | MWS |
| y_true | EAP | 1689 | 226 | 220 |
| | HPL | 114 | 1095 | 81 |
| | MWS | 172 | 88 | 1210 |

**Experiment 2**

| | | y_pred_class | | |
|---|---|---|---|---|
| | | EAP | HPL | MWS |
| y_true | EAP | 1946 | 8 | 21 |
| | HPL | 8 | 1394 | 7 |
| | MWS | 6 | 4 | 1501 |

| | | y_pred_class | | |
|---|---|---|---|---|
| | | EAP | HPL | MWS |
| y_true | EAP | 1953 | 7 | 15 |
| | HPL | 16 | 1389 | 4 |
| | MWS | 40 | 6 | 1465 |

The second experiment showed better prediction between the predicted class and the true class. Please note that the second experiment used the entire training set and because the original set was separated between training and testing, it is synonymous with it training and testing on itself. So versus itself it is doing great. Yet the real concern is whether it will do well to predict against the testing set. The first experiment is more realistic in dealing with the testing set because it is not testing on itself. It took about 70% of the testing set, trained on it, and tested against the other 30% of the training set. Therefore, fears of overfitting are creeping with the second experiment. Yes, the second experiment covers more features and has a bigger 'bag of words' but may creep of overfitting.

**Classification Report:**

| Experiment 1 with MultinomialNB() | | | | |
|---|---|---|---|---|
| | **Precision** | **Recall** | **F1 score** | **Support** |
| **EAP** | 0.85 | 0.87 | 0.86 | 1,915 |
| **HPL** | 0.86 | 0.89 | 0.87 | 1,366 |
| **MWS** | 0.89 | 0.83 | 0.86 | 1,614 |
| **Avg/ Total** | 0.86 | 0.86 | 0.86 | 4,895 |

| Experiment 1 with LogisticRegression() | | | | |
|---|---|---|---|---|
| | **Precision** | **Recall** | **F1 score** | **Support** |
| **EAP** | 0.86 | 0.79 | 0.82 | 2,135 |
| **HPL** | 0.78 | 0.85 | 0.81 | 1,290 |
| **MWS** | 0.80 | 0.82 | 0.81 | 1,470 |
| **Avg/ Total** | 0.82 | 0.82 | 0.82 | 4,895 |

| Experiment 2 with MultinomialNB() | | | | |
|---|---|---|---|---|
| | **Precision** | **Recall** | **F1 score** | **Support** |
| **EAP** | 0.99 | 0.99 | 0.99 | 1,975 |
| **HPL** | 0.99 | 0.99 | 0.99 | 1,409 |
| **MWS** | 0.98 | 0.99 | 0.99 | 1,511 |
| **Avg/ Total** | 0.99 | 0.99 | 0.99 | 4,895 |

| Experiment 2 with LogisticRegression() | | | | |
|---|---|---|---|---|
| | **Precision** | **Recall** | **F1 score** | **Support** |
| **EAP** | 0.97 | 0.99 | 0.98 | 1,975 |
| **HPL** | 0.99 | 0.99 | 0.99 | 1,409 |
| **MWS** | 0.99 | 0.97 | 0.98 | 1,511 |
| **Avg/ Total** | 0.98 | 0.98 | 0.98 | 4,895 |

The classification reports on the four trials confirm fear of overfitting for the second experiment. Because Experiment 2 is self-training and self-training, it would do well. But once again, it is testing against an unknown set. Therefore, experiment 1 seems best for dealing with the unknown training set.

## Identification

Since showing the result for 8,392 values is monotonous, below shows the distribution of testing results per trial.

| | **EAP** | **HPL** | **MWS** |
|---|---|---|---|
| **Experiment 1 w/ Multinomial NB** | 3,292 | 2,305 | 2,795 |
| **Experiment 1 w/ Logistic Regression** | 3,630 | 2,197 | 2,565 |
| **Experiment 2 w/ Multinomial NB** | 3,383 | 2,211 | 2,798 |
| **Experiment 2 w/ Logistic Regression** | 3,626 | 2,215 | 2,554 |

## Discussion (in lieu of Findings)

After performing four different trials, it seems that the first setup (split and vectorize) may provide a lower accuracy but more realistic handling. Even looking at the identification between the first experiment and the second experiment showed minimal difference. MWS had the lowest variability between the experiments (notice how 2,795 test proses were identified as MWS in the first experiment with multinomial NB and 2,798 in the second experiment). However, it seems that each trial collectively identified each text prose differently.

## Remaining Work

Several deliverables must be provided by December 22:

1. Code
2. Report
3. Slide deck

To continue with Capstone 2, the following will be explored:

- Apply **random forests** to dataset
- Look into **XGBoost** for dataset
- Find other tools or algorithms worth exploring for this capstone

## References

The original sources for performing machine learning for investing will come from:


"Spooky Author Identification: Share code and discuss insights to identify horror authors from their writings." Kaggle.com. Published 10/25/2017. Retrieved 12/07/2017. URL: https://www.kaggle.com/c/spooky-author-identification.

R. Tatman. "Beginner's Tutorial: Python". Kaggle.com. Published 10/25/2017. Retrieved 12/07/2017. URL: https://www.kaggle.com/rtatman/beginner-s-tutorial-python.

S. Dane. "Intermediate Tutorial: Python". Kaggle.com. Published 10/25/2017. Retrieved 12/07/2017. URL: https://www.kaggle.com/sohier/intermediate-tutorial-python/.

K. Markham. "Machine Learning with Text in sci-kit learn". PyCon Portland, 05/28/2016. Published 06/08/2016. Retrieved 12/08/2017. URL: http://bit.ly/2yPaYB4.

(optional) S. Bird, E. Klein, and E. Loper. *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit.* Published URL: http://www.nltk.org/book/.