**DATA MINING AND PATTERN RECOGNITION**

**Spooky Author Prediction**

****

PROJECT BY:

**Chirag Padsala (805613197)**

**Hirva Shah (805654761)**

**Nikita Tank (805703204)**

**Nishant Rathi (805602687)**

**Shantanu Mishra (893361006)**

**CPSC - 483**

**Fall 2017**

**Prof. Kenytt D. Avery**

**Department of Computer Science**

**California State University, Fullerton**

Table of Content

[1 Introduction 4](#_Toc501039817)

[1.1 Key Points in a Text Classifier 4](#_Toc501039818)

[2 Dataset Description 5](#_Toc501039819)

[2.1 Train Dataset 5](#_Toc501039820)

[2.2 Test Dataset 6](#_Toc501039821)

[3 Approach 7](#_Toc501039822)

[3.1 Data Pre-processing 7](#_Toc501039823)

[3.1.1 Feature Engineering: 7](#_Toc501039824)

[3.2 Programming Environment, libraries and tools used 8](#_Toc501039825)

[3.2.1 Programming Environment: 8](#_Toc501039826)

[3.2.2 Library: 8](#_Toc501039827)

[3.2.3 Tools: 8](#_Toc501039828)

[4 Algorithms 9](#_Toc501039829)

[4.1 eXtra Gradient Boosting 9](#_Toc501039830)

[4.2 AdaBoost with Decision Tree 10](#_Toc501039831)

[4.3 Multinomial Naive Bayes 10](#_Toc501039832)

[4.4 Support Vector Classifier 11](#_Toc501039833)

[4.5 Random Forest Classifier: 12](#_Toc501039834)

[4.5.1 Why random forest algorithm? 12](#_Toc501039835)

[4.6 Result obtained on the test datasets 13](#_Toc501039836)

[5 Confusion Matrix and ROC Curve 14](#_Toc501039837)

[5.1 Confusion Matrix: 14](#_Toc501039838)

[5.2 ROC (Receiver Operating Characteristic) Curve: 15](#_Toc501039839)

[6 Introduction of Word Cloud (Tag Cloud) 17](#_Toc501039840)

[7 Kaggle Submissions 19](#_Toc501039841)

[8 Kaggle Result 20](#_Toc501039842)

[9 Achievements and conclusion 21](#_Toc501039843)

[9.1 Analysis 21](#_Toc501039844)

[9.2 Achievements 21](#_Toc501039845)

[10 Conclusion 22](#_Toc501039846)

[11 References 22](#_Toc501039847)

List of Figures

[Figure 1 Sample category tree 3](#_Toc501039891)

[Figure 2 Train.csf file sample 4](#_Toc501039892)

[Figure 3 Input data count by authors 5](#_Toc501039893)

[Figure 4 Sample Test data 5](#_Toc501039894)

[Figure 5 XGBoost with kernel='HP' 8](#_Toc501039895)

[Figure 6 AdaBoost with Decision Tree 9](#_Toc501039896)

[Figure 7 MNB with alpha=0.1 10](#_Toc501039897)

[Figure 8 SVC with Penalty parameter 10](#_Toc501039898)

[Figure 9 Random Forest 11](#_Toc501039899)

[Figure 10 Random Forest 13](#_Toc501039900)

[Figure 11 Multinomial Naive Bayes 13](#_Toc501039901)

[Figure 12 Grid Search CV on SVC 13](#_Toc501039902)

[Figure 13 Adaboost with Decision Tree 13](#_Toc501039903)

[Figure 14 XGBoost 13](#_Toc501039904)

[Figure 15 Confusion Matrices of various classifiers 13](#_Toc501039905)

[Figure 16 Linear Regression ROC 14](#_Toc501039906)

[Figure 17 Multinomial NB ROC 14](#_Toc501039907)

[Figure 18 XGBoost ROC 14](#_Toc501039908)

[Figure 19 Random Forest ROC 14](#_Toc501039909)

[Figure 20 Decision Tree ROC 15](#_Toc501039910)

[Figure 21 Adaboost on Decision Tree ROC 15](#_Toc501039911)

[Figure 22 Word cloud Image ( Train Dataset) 17](#_Toc501039912)

[Figure 23 Word Cloud Image (Test Dataset) 17](#_Toc501039913)

# Introduction

Text classification modules are used to classify information, that is, assign a category to a text, also known as tagging. It assigns one or more classes to a document according to their content. A category is a label and categories are structured in a hierarchical category tree. For example, the following category tree shows categories related to retail products:

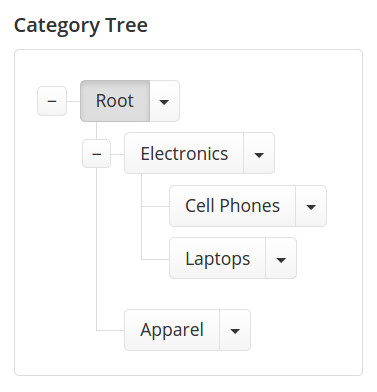


Figure 1 Sample category tree

A machine learning classifier learns to assign the corresponding category to a text by using mathematical models and algorithms that learn to associate a input (text) to the corresponding output (label). The competition dataset contains text from works of fiction written by spooky authors of the public domain: Edgar Allan Poe, HP Lovecraft and Mary Shelley. The data was prepared by chunking larger texts into sentences using Core NLP's MaxEnt sentence tokenizer.

## Key Points in a Text Classifier

Text classification modules are constructed taking into account the following key aspects:

|  |  |
| --- | --- |
| A Category Tree | A Machine Learning Model |
| Some Training Samples | Some Performance Measures |

# Dataset Description

The Dataset contains text from books of fiction written by authors of the public domain: Edgar Allan Poe(EAP), HP Lovecraft(HPL) and Mary Wollstonecraft Shelley(MWS). The data was prepared by chunking larger texts into sentences using CoreNLP's MaxEnt sentence tokenizer.

* Dataset contains 2 files. Train.csv-training set, test.csv-the test set.
* Train.csv: file which contains few statements with its author’s name and unique id.
* Test.csv: File with text for which author needs to be predicted.
* Labels:
  + Id: unique identifier for each sentence
  + Text: Statements from horror story books
  + Author(Classes): Author names EAP, HPL, MWS.
* Sample output: Prediction of a statement to be related to particular author from three given classes.

## Train Dataset

Total training data - 19580

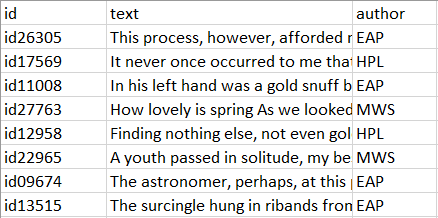


Figure 2 Train.csf file sample

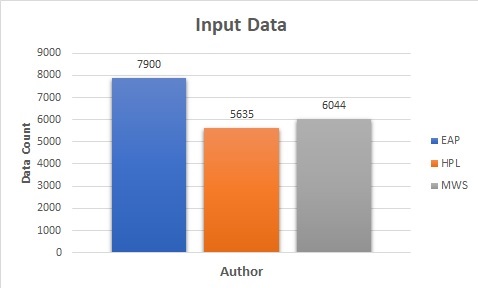


Figure 3 Input data count by authors

## Test Dataset

Total records 8393

Predict Probabilities for Author

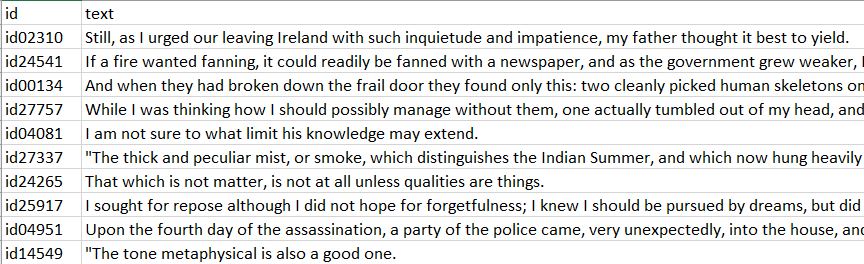
****

Figure 4 Sample Test data

# Approach

## Data Pre-processing

Applied label encoder to get numeric values for classes (here, author).

### Feature Engineering:

Applied multiple stemming algorithms and lemmatization algorithm which are mentioned below:

Stemming Algorithms used:

1. Poster Stemming
2. Snowball Stemming

Lemmatization Algorithm:

1. Word Net Lemmatizer:

Stemming helps to identify root of the word and group multiple words related to the same root example: (run, running, ran, runs) is grouped into single word (run), thus decreasing the word count and increasing the accuracy. Lemmatization is used to find the relative words e.g.., hot -> warm. Though it is mostly used in search engines, voice commands etc. Poster stemming mostly slice down the end part of the word, while snowball stemmer is language specific and performs better compared to poster.

After stemming we had a new set of vocabulary with less but effective words for text classification.

Applied TF-IDF to create a count Vocabulary Matrix which shows importance of the word in the whole document. Here,

1. TF(t) = (Number of times term t appears in a document) / (Total number of terms in the document).
2. IDF(t) = loge(Total number of documents / Number of documents with term t in it).
3. By selecting different parameters in tf\_idf, accuracy will vary.
4. Added defined stop words as well as our own stop words.
5. Applied transform function to normalize the count vector.
6. As a final result, we got single word with its frequency as a vocabulary in which each word will be a single feature.

Applied K-fold cross validation with different values of k=4,5,6.

## Programming Environment, libraries and tools used

### Programming Environment:

Python 3.6, Anaconda

### Library:

NumPy, SciPy, XGBoost, Sklearn, pandas

### Tools:

Jupyter Notebook

# Algorithms

## eXtra Gradient Boosting

* Boosting: Boosting is a [machine learning ensemble](https://en.wikipedia.org/wiki/Ensemble_learning) [meta-algorithm](https://en.wikipedia.org/wiki/Meta-algorithm) for primarily reducing [bias](https://en.wikipedia.org/wiki/Supervised_learning#Bias-variance_tradeoff), and also variance in [supervised learning](https://en.wikipedia.org/wiki/Supervised_learning).
* Gradient Boosting : **Gradient boosting** is a [machine learning](https://en.wikipedia.org/wiki/Machine_learning) technique for [regression](https://en.wikipedia.org/wiki/Regression_(machine_learning)) and [classification](https://en.wikipedia.org/wiki/Classification_(machine_learning)) problems, which produces a prediction model in the form of an [ensemble](https://en.wikipedia.org/wiki/Ensemble_learning) of weak prediction models, typically [decision trees](https://en.wikipedia.org/wiki/Decision_tree_learning). It builds the model in a stage-wise fashion like other [boosting](https://en.wikipedia.org/wiki/Boosting_(meta-algorithm)) methods do, and it generalizes them by allowing optimization of an arbitrary [differentiable](https://en.wikipedia.org/wiki/Differentiable_function) [loss function](https://en.wikipedia.org/wiki/Loss_function).
* XGBoost: XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable. It implements machine learning algorithms under the [Gradient Boosting](https://en.wikipedia.org/wiki/Gradient_boosting) framework. XGBoost provides a parallel tree boosting (also known as GBDT, GBM) that solve many data science problems in a fast and accurate way.

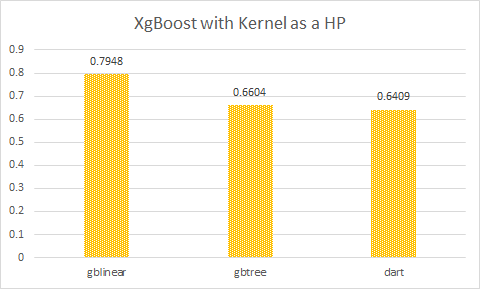


Figure 5 XGBoost with kernel='HP'

## AdaBoost with Decision Tree

It can be used in conjunction with many other types of learning algorithms to improve performance. The output of the other learning algorithms ('weak learners') is combined into a weighted sum that represents the final output of the boosted classifier. AdaBoost is adaptive in the sense that subsequent weak learners are tweaked in favor of those instances misclassified by previous classifiers. AdaBoost is sensitive to noisy data and [outliers](https://en.wikipedia.org/wiki/Outlier). In some problems, it can be less susceptible to the [overfitting](https://en.wikipedia.org/wiki/Overfitting_(machine_learning)) problem than other learning algorithms. The individual learners can be weak, but if the performance of each one is slightly better than random guessing, the final model can be proven to converge to a strong learner.

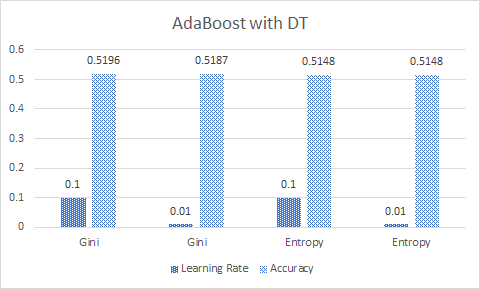


Figure 6 AdaBoost with Decision Tree

## Multinomial Naive Bayes

The multinomial Naive Bayes classifier is suitable for classification with discrete features (e.g., word counts for text classification). The multinomial distribution normally requires integer feature counts. However, in practice, fractional counts such as tf\_idf may also work. Naive Bayes classifiers are highly scalable, requiring many parameters linear in the number of variables (features/predictors) in a learning problem. [Maximum-likelihood](https://en.wikipedia.org/wiki/Maximum-likelihood_estimation) training can be done by evaluating a [closed-form expression](https://en.wikipedia.org/wiki/Closed-form_expression), which takes [linear time](https://en.wikipedia.org/wiki/Linear_time), rather than by expensive [iterative approximation](https://en.wikipedia.org/wiki/Iterative_method) as used for many other types of classifiers.

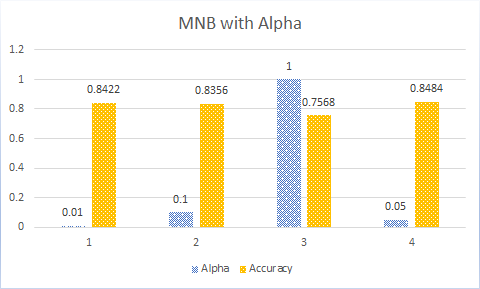


Figure 7 MNB with alpha=0.1

## Support Vector Classifier

Support Vector Machine” (SVM) is a supervised machine learning algorithm which can be used for both classification or regression challenges. However, it is mostly used in classification problems. In this algorithm, we plot each data item as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a coordinate. Then, we perform classification by finding the hyperplane that differentiate the two classes very well.

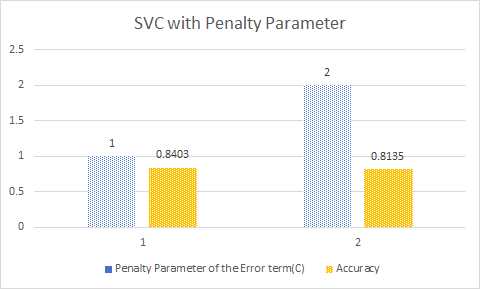


Figure 8 SVC with Penalty parameter

## Random Forest Classifier:

Random forest algorithm is a supervised classification algorithm. As the name suggest, this algorithm creates the forest with a number of trees. In general, the more trees in the forest the more robust the forest looks like. In the same way in the random forest classifier, the higher the number of trees in the forest gives the high accuracy results.

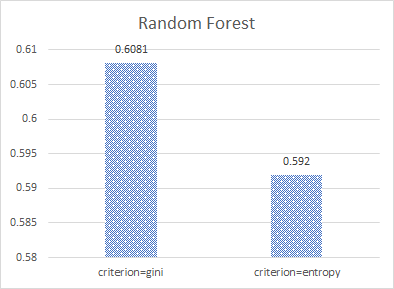


Figure 9 Random Forest

### Why random forest algorithm?

* The same random forest algorithm or the random forest classifier can use for both classification and the regression task.
* Random forest classifier will handle the missing values.
* When we have more trees in the forest, random forest classifier won’t overfit the model.
* Can model the random forest classifier for categorical values also.

Because of the above-mentioned advantages, we applied that algorithm on spooky author identification dataset.

We used various parameters to test spooky author identification dataset.

* n\_estimator: The number of trees in the forest. We fried different value of n\_estimator such as 10, 100, 1000 to see the result.
* Criterion: The function to measure the quality of a split. Supported criteria are “gini” for the Gini impurity and “entropy” for the information gain. Note: this parameter is tree-specific.
* max\_features: The number of features to consider when looking for the best split. we achieved 0.6081 accuracies for Gini criterion and got 0.592 accuracies for entropy criterion.

## Result obtained on the test datasets

|  |  |  |
| --- | --- | --- |
| **Algorithm** | **Accuracy** | **AUC** |
| Random Forest | 0.66 | 0.875 |
| MultinomialNB | 0.82 (alpha = 0.8) , 0.84 (alpha = 1.0) | 0.956 |
| SVC with GridSearchCV | 0.80 (kernel = linear) | 0.940 |
| Adaboost on Decision Tree | 0.51 (learning\_rate=0.1, n\_estimators=10) | 0.692 |
| xgboost | 0.78 (booster='gblinear', max\_depth=5) | 0.919 |

# Confusion Matrix and ROC Curve

## Confusion Matrix:

A confusion matrix is a table that is often used to **describe the performance of a classification model** (or "classifier") on a set of test data for which the true values are known. The confusion matrix itself is relatively simple to understand, but the related terminology can be confusing. Below is the confusion matrix, we got for our classifiers on the validation set.

|  |  |  |
| --- | --- | --- |
| Figure 10 Random Forest | Figure 11 Multinomial Naive Bayes | Figure 12 Grid Search CV on SVC |
| Figure 13 Adaboost with Decision Tree | Figure 14 XGBoost |  |

Figure 15 Confusion Matrices of various classifiers

## ROC (Receiver Operating Characteristic) Curve:

The ROC curve is created by plotting the [true positive rate](https://en.wikipedia.org/wiki/True_positive_rate) (TPR) against the [false positive rate](https://en.wikipedia.org/wiki/False_positive_rate)(FPR) at various threshold settings. The true-positive rate is also known as [sensitivity](https://en.wikipedia.org/wiki/Sensitivity_(tests)), [recall](https://en.wikipedia.org/wiki/Precision_and_recall#Definition_(classification_context)) or probability of detection in [machine learning](https://en.wikipedia.org/wiki/Machine_learning). The false-positive rate is also known as the [fall-out](https://en.wikipedia.org/wiki/Information_retrieval#Fall-out) or probability of false alarm and can be calculated as (1 − [specificity](https://en.wikipedia.org/wiki/Specificity_(tests))). The ROC curve is thus the sensitivity as a function of [fall-out](https://en.wikipedia.org/wiki/Information_retrieval#Fall-out). In general, if the probability distributions for both detection and false alarm are known, the ROC curve can be generated by plotting the [cumulative distribution function](https://en.wikipedia.org/wiki/Cumulative_distribution_function) (area under the probability distribution from to the discrimination threshold) of the detection probability in the y-axis versus the cumulative distribution function of the false-alarm probability on the x-axis.

|  |  |
| --- | --- |
| Figure 16 Linear Regression ROC | Figure 17 Multinomial NB ROC |
| Figure 18 XGBoost ROC | Figure 19 Random Forest ROC |

|  |  |
| --- | --- |
| Figure 20 Decision Tree ROC | Figure 21 Adaboost on Decision Tree ROC |

# Introduction of Word Cloud (Tag Cloud)

A tag cloud (word cloud, or weighted list in visual design) is a visual representation of text data, typically used to depict keyword metadata (tags) on websites, or to visualize free form text. Tags are usually single words, and the importance of each tag is shown with font size or color. This format is useful for quickly perceiving the most prominent terms and for locating a term alphabetically to determine its relative prominence. If you want a stunning visualization format to highlight important textual data points, word cloud can make dull data sizzle and immediately convey crucial information.

**What are Word Clouds?**

Word clouds (also known as text clouds or tag clouds) work in a simple way: the more a specific word appears in a source of textual data (such as a speech, blog post, or database), the bigger and bolder it appears in the word cloud. It basically counts the occurrence of words and change format of that word based on word count.

**Where Word Clouds Excel for Businesses**

* Finding customer pain points — and opportunities to connect.
* Understanding how your employees feel about your company.
* Identifying new SEO terms to target.

I used word cloud on spooky author dataset to visualize the predicted result on the test dataset. We can conclude from the word cloud that the result we achieved on the test data set is accurate.

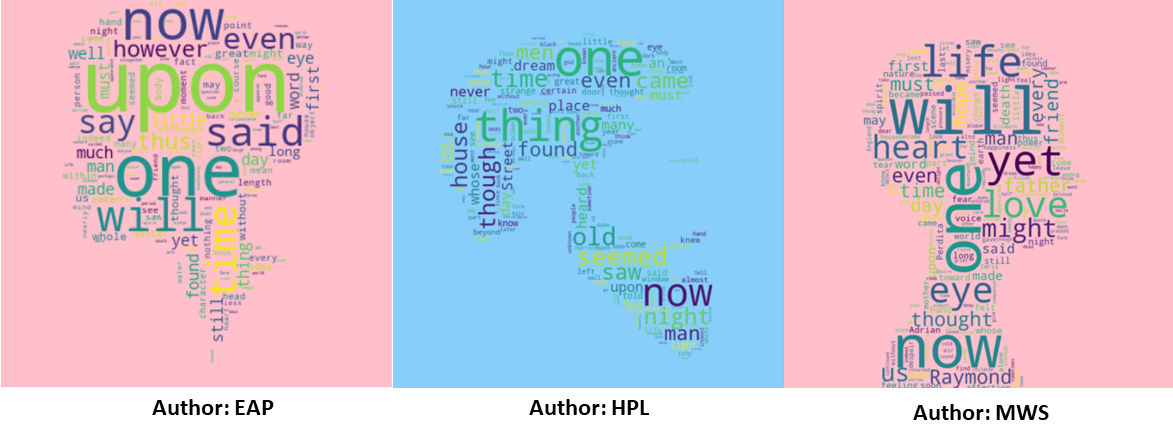


Figure 22 Word cloud Image ( Train Dataset)



Figure 23 Word Cloud Image (Test Dataset)

# Kaggle Submissions

# Kaggle Result

|  |  |  |
| --- | --- | --- |
| **Algorithm** | **Accuracy** | **Kaggle Score** |
| Random Forest | 0.66 | 0.76 |
| MultinomialNB(Alpha=0.8) | 0.82 | 0.36 |
| MultinomialNB (Alpha= 1.0) | 0.84 | 0.35 |
| SVC with GridSearchCV | 0.80 | 0.41 |

# Achievements and conclusion

## Analysis

* Data set analysis and feature extraction
  + Cross validation technique:
  + To measure the accuracy, divided train data into train and validation data with 80-20% ratio. Trained the model with training data and measured the accuracy by predicting author on validation data.
  + K-fold Cross Validation: For different values of k measured accuracy.
  + As a final result, applied all possible algorithms to get maximum accuracy. (Here: MultinomialNB, Random Forest, GridSearchCV, SVC, XGBoost) and used full train data for training the model and predicted probability on test data.

## Achievements

* Understood the importance of different parameters
  + Learning Rate(alpha)
  + Validation percentage
  + Stop Words
  + Bag of words
* Preprocessing of training data improves the prediction.
* Used boosting method to improve the performance.

# Conclusion

In this project, we found the best combination of features and parameters for text classification. We added few parameters and improved boosting algorithm which further improved execution speed and performance of existing model(algorithm). We did cross validation on linear regression, decision tree, random forest, support vector classifier and multinomial Naive Bayes and found the accuracy of Naïve Bayes as the best in cross validation. However, after submitting the submission file on kaggle we got better score for support vector classifier with linear kernel.

# References

* <https://www.kaggle.com/c/spooky_author_identification>
* <https://www.kaggle.com/arthurtok/spooky-nlp-and-topic-modelling-tutorial>
* <https://anaconda.org/>
* <http://scikit-learn.org>
* <https://www.lfd.uci.edu/~gohlke/pythonlibs/>