Computer Science
School of Informatics, Computing, and Engineering
Indiana University, Bloomington, IN, USA
Data Mining B565 Fall 2017

Kaggle Competitions: Author Identification Statoil/C-CORE Iceberg Classifier Challenge

Abhishek Babuji[1]*, Nirad Ranjan Parhi[2]*, Sneha Nyayapati[3]*

Executive Summary

Keywords
Natural Language Processing — Image Classification — Data Mining

[1] *Data Science, School of Informatics, Computing, and Engineering, Indiana University, Bloomington, IN, USA*
[2] *Data Science, School of Informatics, Computing, and Engineering, Indiana University, Bloomington, IN, USA*
[2] *Data Science, School of Informatics, Computing, and Engineering, Indiana University, Bloomington, IN, USA*
***Corresponding author**: ababuji@iu.edu
***Corresponding author**: nparhi@iu.edu
***Corresponding author**: slnyayap@iu.edu

Contents

1 Introduction **1**
1.1 Author Identification . . . 2
1.2 Statoil/C-CORE Iceberg Classifier Challenge . . . 2
2 Datamining **2**
2.1 Data Preprocessing . . . 3
2.2 Mining, Interpretation, and Action . . . 3
C 4.5 • Naive Bayes • K-means clustering • k-Nearest Neighbor algorithm • PageRank • CART • Expectation Maximization • Apriori • Support Vector Machines • Adaboost
3 Author Identification: Full Problem Description **7**
3.1 Data Analysis . . . 8
Number of Excerpts by Author • Number of Excerpts containing Dialogues by Author • Most commonly used words by Author: (WORDCLOUDS and BARGRAPHS)
3.2 Preprocessing . . . 10
3.3 Methods . . . 11
k-Nearest Neighbor • Multinomial Naive Bayes • Logistic Regression
3.4 Evaluation Metric . . . 13
3.5 Hardwares, Softwares and Packages . . . 13
3.6 Results . . . 14
3.7 Author Identification - Data Mining Summary . . . 17
3.8 Future Work . . . 19
4 Iceberg: Full Problem Description **19**
4.1 The Data and its Significance . . . 19
4.2 Details of the dataset . . . 19
4.3 Exploratory Data Analysis . . . 19
4.4 Feature Selection . . . 23
4.5 Methods . . . 23
Logistic Regression • Convolutional Neural Networks
4.6 Hardwares, Softwares and Packages . . . 24
4.7 Results . . . 25
Acknowledgments **25**
References **25**

1. Introduction

- The following sections give an explanation to how the various steps in Data Mining was used to explore solutions to two `Kaggle competitions`, namely; Spooky Author Identification and Statoil/C-CORE Iceberg Classifier Challenge. To gain access to the datasets for the respective two competitions, one must create an account at `Kaggle.com`, read the rules and accept the Terms and Conditions agreement. An example of what a submission must look like is posted on the competition website.
- Given a problem statement, the various steps involved in Data Mining such as getting the data, followed by cleaning and enriching the data, integrating and transforming the data so that it is suitable for fitting them to appropriate Machine Learning models which would then help mine patterns from them and using these patterns to make predictions on unseen data and validating them using the appropriate evaluation metrics would help in creating a valid solution to the problem statement.

1.1 Author Identification

From the Data Description for the Spooky Author Identification competition at `Kaggle`, 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. Our objective is to accurately identify the author of the sentences in the test set given the labelled examples in the training set.

We are given multiple texts from these three authors labelled `EAP`, `MWS` and `HPL`. We must use the information in the texts corresponding to each of three authors and mine patterns from it.

A text is simply a passage from one of the author's works:

> Finding nothing else, not even gold, the Superintendent abandoned his attempts; but a perplexed look occasionally steals over his countenance as he sits thinking at his desk.

from HP Lovecraft, *The Transition of Juan Romero*. The text may contain punctuation, conversations between people within the story, different diacritic symbols[1]

The text will not be fed in directly. The texts will be convert into a *bag of words*[2] model which will have information about the ranking of each word appearance in a particular authors text. There are multiple ways to rank the words. The most common method is word frequency.

Once patterns have been mined after tuning the parameters of the respective Machine Learning algorithms, the pattern is fit on a test set and evaluated for correctness. There are different evaluation metrics used to test for correctness; for the sake of this competition, we will be considering Accuracy. Given a set of predictions and their true labels, accuracy is an indicator of what percentage of our test set, the prediction was the same as the true label.

1.2 Statoil/C-CORE Iceberg Classifier Challenge

Statoil/C-Core Iceberg classification challenge is an image recognition challenge provided by `Kaggle`. The challenge is to primarily detect drifting icebergs in areas such as the offshores of the East Coast of Canada. These icebergs present threats to various navigation activities in such areas.

In order to reduce the impending damages from these icebergs, many companies are using various monitoring methods in these areas to oversee the environmental conditions and icebergs present in these water bodies. But there are many such areas where these tools and techniques fail to perform due to harsh weather conditions. So, the next feasible method in such conditions is monitoring through a satellite.

The clients who have provided this challenge in `Kaggle` are Statoil and C-CORE. Statoil is an energy based company operating worldwide. The containers and ships belonging to this company move in very difficult water bodies and harsh weather conditions. So they have tied up with C-CORE to provide satellite monitoring of its assets in various geographic locations. C-CORE is a company which uses satellite data and computer based surveillance systems to assist companies like Statoil.

2. Datamining

- What is datamining?

  Data Mining is an iterative process over a dataset where; given a problem statement, the dataset is subsetted, transformed, enriched and corrected to answer the questions posed by the problem statement by identifying relationships and associating new meaning to the data.

- What does it yield?

  The steps followed in Data Mining may yield new relationships, produces information (answers) that is either consistent or inconsistent with, or unrelated to the problem statement.

- What are the general steps?

  - Data Collection and Storage
  - Data Enrichment
  - Cleaning
  - Integration and Transformation
  - Mining
  - Validation and Evaluation
  - Make Actionable

- What is the difference between clustering and classification?

  In classification, some relationships between data may already be mentioned before hand. For example, a given set of data may already be labelled to be belonging to a particular category. The goal in classification is to understand how these relationships are built and by doing so, categorize new, never before seen data; given example data with labels.

  In clustering, no existing relationships are specified. The role of clustering is to mine patterns and identify any relationships that may exist and then accordingly

[1] A diacritic mark is added to a letter to indicate a special pronunciation. Examples are the diaereses in the borrowed French words *naïve* and *Noël*, which show that the vowel with the diaeresis mark is pronounced separately from the preceding vowel; the acute and grave accents, which can indicate that a final vowel is to be pronounced, as in *saké* and poetic *breathèd*; and the cedilla under the 'c' in the borrowed French word *façade*, which shows it is pronounced *s* rather than *k*.

[2] A *bag of words* is a representation where, all the unique words in the corpora are split separately. All the text is now represented as a multiset of its words. Each word is now a feature. We can then decide how to attribute weights to each of these features; for e.g. the number of times the words has appeared could be considered its weight.

group items into categories. The number of categories are unknown. The categories will be defined by the patterns and natural groupings that may exist in the data which must be discovered from scratch.

- What is a loss function?

  A loss function is a way to numerically represent the inaccuracy of a Machine Learning algorithm.

$$\frac{1}{n}\sum_{i=1}^{n}(y_i-\hat{y}_i)^2$$

  is a loss function or cost function commonly known as the Mean Squared Error.

  y_i represents the actual label (numerical or categorical).

  $\hat{y}_i$ represents the predicted or estimated label (numerical or categorical).

$$\sum_{i=1}^{n}(y_i-\hat{y}_i)^2$$

  finds the sum of the square of the difference between the actual label and the predicted label from $i=1$ to $i=n$. This is called the Squared Error term.

$$\frac{1}{n}\sum_{i=1}^{n}(y_i-\hat{y}_i)^2$$

  gives the Mean of the Squared Errors.

2.1 Data Preprocessing

- What are the steps?

  The steps range from loading the data to and from the database, reading it in a format that a programming language can understand, understanding the structure of the data; the datatypes of each tuple or column, transformation of the data from one structure to another (such as a sentence to a vector space), dealing with missing values and handling outliers, taking numerical summaries that may help explain correlations and causation.

- What is the general load (time, space, $) for preprocessing

  The general time and space complexity for preprocessing varies on the type of operations and transformations being performed.

- What challenges does each step present?

  Since nothing in data mining is instructed, every step in preprocessing requires experience and gumption. Understanding whether to maintain a numerical variable as numerical type and not converting it to a categorical type. Missing value imputations; debating whether to keep or remove the data, understanding the difference between outliers and anomalies, trying to ascertain the type of transformation that is best suited for the purpose, creating new variables from old variables that may summarize many variables and dimensionality reductions are some of the challenges presented in the steps involved in data preprocessing.

2.2 Mining, Interpretation, and Action

- Briefly discuss the top 10 algorithms.

2.2.1 C 4.5

The C 4.5 is an extension of the ID3 Decision Tree algorithm by Ross Quinlan. The C4.5 does not differ much from the working of ID3 apart from providing a few improvements over ID3 which is probably why it was one of the top 10 algorithms.

`STEP 1`. Check for the following:

i. If all the observation in the data belong to the same class, create a leaf node to the decision tree picking that class. This is known in ID3 as a pure subset.

ii. If any unseen class is found, create a decision node higher up the tree using the expected value[1].

`STEP 2`. For each attribute a, find the normalized information gain ratio from splitting on a.

Since Information Gain is heavily biased towards tests with numerous outcomes[1], using the Gain Ratio overcomes the disadvantage of using Information Gain; by dividing information gain by the information provided by the test outcomes. This is an improvement over the ID3 where only information gain was being considered and not the gain ratio.

`STEP 3`. Let `t` be the attribute with the highest normalized information gain.

`STEP 4`. `t` now becomes one of the parent nodes.

`STEP 5`. Repeat `STEP 1 to 4` on the subsets obtained by splitting on `t`, and add those nodes as children of their respective parent.
The C4.5 found a way to handle continuous data unlike ID3. During the inception of the Decision Trees,

a major problem Decision Trees suffered from is the presence of missing values. In C4.5 missing values are simply ignored in the calculation of information gain and entropy.
The C5.0 which also ended up being created is faster, more memory efficient and went on to become a more robust Decision Tree algorithm than the C4.5.

2.2.2 Naive Bayes

The reason why the algorithm is called 'Naive' is because it assumes that all attributes are independent. The presence of a particular feature is not affected by or caused by the presence or absence of another feature or attribute is what it assumes. And this assumption is considered *Naive*.

The reason why it is in the top 10 algorithms is because despite the assumptions being Naive; in practice, in many real life applications, it performs very well.

The main **disadvantage** of Naive Bayes is that, it can not take into account the relationships and interaction between features (e.g., it can not understand that although you may love chocolate milk as a whole, but you hate chocolate separately and milk separately).

Naive Bayes perform best on all the data where the 'Naive' assumption can hold true; data where features are independent and it performs poorly on data where features are dependent.

Consider the example dataset in Figure 1. Let us say, we get a new

	Color	Origin	Type	_Stolen?
0	Red	Export	Sports	Yes
1	Red	Export	Sports	No
2	Red	Export	Sports	Yes
3	Yellow	Export	Sports	No
4	Yellow	Imp	Sports	Yes
5	Yellow	Imp	SUV	No
6	Yellow	Imp	SUV	Yes
7	Yellow	Export	SUV	No
8	Red	Imp	SUV	No
9	Red	Imp	Sports	Yes

Figure 1. Naive Bayes: example dataset

The dataset contains 4 columns, namely `Color, Origin, Type` and the label `_Stolen?`, where `Yes` in the column means, that row represents the features of a car that is `Stolen` and `No` means `Not Stolen.`
Let us assume our problem statement was to classify an incoming example X with features `Color = Red, Type = SUV, Origin = Export` as `Stolen` or `Not Stolen` using Naive Bayes. We calculate:

$$P\left(X \,\middle|\, yes\right) \times P(yes) \text{ and } P\left(X \,\middle|\, no\right) \times P(no)$$

The unclassified sample X has the features `Color = Red, Type = SUV, Origin = Export.`

We can calculate $P\left(X \,\middle|\, yes\right) \times P(yes)$ as

$$P\left(Red \,\middle|\, yes\right) \times P\left(SUV \,\middle|\, yes\right) \times P\left(Export \,\middle|\, yes\right) \times P(yes)$$

Similarly, we can calculate $P\left(X \,\middle|\, no\right) \times P(no)$ as

$$P\left(Red \,\middle|\, no\right) \times P\left(SUV \,\middle|\, no\right) \times P\left(Export \,\middle|\, no\right) \times P(no)$$

If $P\left(X \,\middle|\, yes\right) \times P(yes) > P\left(X \,\middle|\, no\right) \times P(no)$, then we classify the sample X has having the label *Yes*.

If $P\left(X \,\middle|\, yes\right) \times P(yes) < P\left(X \,\middle|\, no\right) \times P(no)$, then we classify the sample X has having the label *No*.

2.2.3 K-means clustering

The K-means clustering is a top 10 algorithm probably because it overcomes the disadvantages of the hierarchical agglomerative clustering in that, K-means clustering algorithm is computationally tractable and works well with large datasets. This was a major breakthrough in clustering techniques at the time. K-means clustering can now be used as a pre-clustering method that further enhances the capabilities of other advanced clustering methods.

Steps to perform K-means clustering:

`STEP 1`: Start by picking 'k', the number of clusters.

`STEP 2`: Create and initial cluster, by choosing 'k' random points. This means, that we initially now have 3 clusters each of which has a centroid.

`STEP 3`: For each point, place the point into the cluster whose current centroid is nearest to it. (Keep in mind that overtime we add a point to the cluster, there is a chance that the centroid of that cluster changes) To measure nearness, use a similarity metric such as Euclidean Distance.

`STEP 4`: After all points are assigned, update the location of the centroids of the k clusters.

`STEP 5`: Reassign all the points to their closest centroid. This may lead to movement of points between clusters.

`REPEAT STEPS 3 to 5` until centroids do not change and points stop moving between the clusters.
Some of the **disadvantages** of K-means clustering is that selection of an optimum 'k' is hard, selection of

initial centroids is random and so there are possibly many cluster solutions.

2.2.4 k-Nearest Neighbor algorithm

We are given a set $\left\{(a_1,b_1),(a_2,b_2)...(a_n,b_n)\right\}$.
Each of the a is a point in 2 dimensional space.
a_1 is defined by coordinates (x_1,y_1).
a_2 is defined by coordinates (x_2,y_2) and so on.
When a new unclassified a_{new} appears, we compute the distance of a_{new} and each of the a in the set. We then take the k number of closest a for which distance(a_{new}, a) is minimum. Assign a_{new} to the majority of the class that the k closest a belong to.
The main **disadvantage** of kNN is that choosing the parameter k can be tricky. The algorithm is computationally very expensive for large datasets. The main **advantage** of kNN is that the data points need not be linearly separable.

2.2.5 PageRank

PageRank is an algorithm developed by Larry Page used by Google Search to webpages.
The formula for the rank of a page i can be given by

$$r_i = (1-d) + d \sum_{p_j \in B_i} \frac{r_j}{l_j}$$

where

p_i: A particular page i

r_i: Rank of p_i

l_i: Number of outgoing links of page p_i

B_i: The set of pages linking to page p_i

Consider the network drawn as shown in Figure 2.

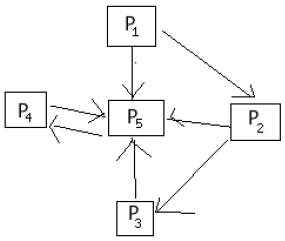


Figure 2. PageRank: example network

For the above network;

$l_1 = 2, B_1 = \left\{\right\}$

$l_2 = 2, B_2 = \left\{p_1\right\}$

$l_3 = 1, B_3 = \left\{p_2\right\}$

$l_4 = 1, B_4 = \left\{p_5\right\}$

$l_5 = 1, B_4 = \left\{p_1, p_2, p_3, p_4\right\}$

Using the formula for r_i

$r_1 = \left(1-d\right) + d\left(0\right)$

$r_2 = \left(1-d\right) + d\left(\frac{r_1}{2}\right)$

$r_3 = (1-d) + d\left(\frac{r_2}{2}\right)$

$r_4 = (1-d) + d\left(\frac{r_1}{1}\right)$

$r_5 = (1-d) + d\left(\frac{r_1}{2} + \frac{r_2}{2} + \frac{r_3}{1} + \frac{r_4}{1}\right)$

Taking a person who is currently browsing the web; the probability, at any step, that the person will continue is call the damping factor d. The damping factor is normally set to 0.85.
The **disadvantages** of PageRank is that it will rank pages based on the set of pages linking to a page, so a new page no matter how relevant and how informative for the search query that has no incoming link will not be ranked in preference to an older page with less relevance to the query.

2.2.6 CART

CART stands for classification and Regression Trees. This is a supervised machine learning technique developed by Leo Breiman, Jerome Friedman, Richard Olshen, and Charles Stone. This technique is used with decision trees to perform both classification as well as regression. The type of operation operation performed depends on whether the dependent variable is categorical or numerical. For categorical, we use classification and for numeric we use regression.

The decision tree that is formed uses a number of rules which are based on the attributes used in the modeling data set. Following are the general steps:

`STEP 1`: Initially, rules based on attribute values are selected to get the best split to segregate the input dataset based on the dependent variable.
`STEP 2`: Then, after a rule is selected, a node is split into two, and the same process is applied on the child node.
`STEP 3`: The splitting will stop, when the algorithm decides that the gain on the dataset is not improving.

After the tree is constructed using CART, each terminal node of the tree represents a new rule. Every observation in our dataset must fall into one terminal node.

CART uses Gini impurity as the criteria to segregate data while constructing a decision tree. It is a measure of how often a randomly chosen element from the set would be labelled incorrectly, if the element was randomly labelled according to distribution of labels in the subset.

Gini Impurity of a node t for a binary target is given as:

$$G(t) = 1 - p(t)^2 - (1 - p(t))^2$$

$p(t)$ is the relative frequency of class 1 in the node. The information gain that is generated by the splitting of the parent P into children L and R is given by:

$$I(P) = G(P) - qG(L) - (1-q)G(R)$$

The primary **advantage** of CART is that it works for both; nominal and categorical data and it does both classification and regression.

2.2.7 Expectation Maximization

Expectation Maximization is a technique in data mining used for clustering and knowledge discovery. This is an iterative method to find maximum likelihood and maximum a posteriori estimates of various parameters in different statistical models.

This model basically has two steps - Expectation and Maximization. In the Expectation step, the algorithm creates a function for the expectation of log-likelihood and in the Maximization step, it computes the parameters for maximizing the log-likelihood calculated in the previous step. These parameters are now used for the next E-step.

If we have a statistical model, which generates a set of observed data **X**, set of missing values **Z**, an unknown parameter vector Θ and a likelihood function given as $L(\Theta;X,Z) = p(X,Z|\Theta)$, the Maximum Likelihood Estimate is given by

$$L(\Theta;X) = p(X|\Theta) = \int p(X,Z|\Theta)dZ$$

The EM steps are:
Expectation(E Step):

$$Q(\Theta|\Theta^{(t)}) = E_{Z|X,\Theta^{(t)}}[LogL(\Theta;X,Z)]$$

Maximization(M Step):

$$\Theta^{(t+1)} = argmaxQ(\Theta|\Theta^{(t)})$$

The **applications** for EM is in data clustering, computer vision, natural language processing etc.

2.2.8 Apriori

The underlying machine learning method for Apriori algorithm is Association rule learning. Apriori algorithm is used to mine frequent itemsets and corresponding association rules.
Association rule learning helps us establish relationships between variables in a dataset. Let S=$\{s_1, s_2, s_3, ..., s_k\}$ be a set of k attributes called items and D=$\{t_1, t_2, ..., t_n\}$ be the set of transactions.
t_i represents a unique transaction ID for every row in the database and S represents a subset of itemsets.
A rule can be defined as an implication, $X -> Y$ where X and Y are subsets of S; $(X,Y$ is a subset of $S)$, and they have no elements in common.

The real-life application of this algorithm can be interpreted as - If a person buys a product X and Y, he is very likely to buy a product Z. This helps business owners predict their sales and customer choices.

Pros:
1. It is a very easy to implement algorithm.
2. It works well on large datasets.

Cons:
1. It works on finding candidate rules, which can be computationally expensive sometimes.

2.2.9 Support Vector Machines

Support Vector Machines are supervised learning models that finds hyperplanes to classify the dataset into two classes.
A hyperplane is a linear function, similar to a line equation $y = mx + c$.

Support Vectors are the data points that are nearest to the hyperplane and any alteration or removal of these points would also impact the position of the hyperplane.

It is similar to the implementation of C4.5, but SVMs do not use Decision Trees. SVMs are capable of projecting the data into higher dimensions that helps best figure out the hyperplanes for classification of data.

The task of classification depends upon a 'margin', which is basically the distance between the hyperplane and the nearest data point. The greater the margin between the hyperplane and any data point, the better the chances for correct classification.

Apart from the linear classification using a hyperplane, SVMs perform a kernel trick that maps the inputs into high-dimensional feature space. This way SVMs are capable of performing non-linear classifications as well.

The applications of SVMs include classification of images, text categorization, hand-written character recognition etc.

Pros:
1. Works well on small sets of data.
2. Good level of accuracy can be obtained with SVMs.

Cons:
1. Training time spikes with larger datasets.
2. When the data is noisy and the classes seem to overlap, SVMs are not very efficient.

2.2.10 Adaboost

Adaboost is a boosting algorithm that constructs a classifier. Boosting is an ensemble method that takes a multiple machine learning algorithms and combines them. It aims at taking a group of weak learners and combining them to create one strong learner.

Adaboost takes a sample of the training set and checks how accurate each learning model is. Any misclassified examples are assigned a weight so that there are higher chances of these samples being picked in the next run.

The best learner is weighted and added into the ensemble . At the end of multiple rounds, what we are left with is a group of weighted learners repeatedly trained on the misclassified examples. This way we can find the best learner.

Pros:
1. A very simple and easy-to-implement algorithm.
2. No tuning of parameters required.
3. We do not need to have any prior knowledge about the weak learners.

Cons:
1. The weak classifiers can lead to overfitting.
2. It is seen to be impacted greatly by noise.

- Does data mining tell us what to do?

  Data mining does not tell you what to do. It is a messy process that requires a lot of experience and gumption. Understanding how to wrangle the data in a suitable format, clean data when its messy and answer questions about what to do with missing data where there never is a perfect answer is what data mining is all about.

- What are some new types of problems in data mining?

  Some of the new types of problems in data mining include but are not limited to:
  1. Learning how to handle high dimensional data.
  2. Mining user interactions (finding ways to measure and capture human interactions)
  3. Handling never before seen types of data (complex data)
  4. Accurately predicting what a user query in a Search Engine is exactly trying to ask for.

3. Author Identification: Full Problem Description

Problem Statement: Given a corpora of excerpts from three writers namely, Edgar Allen Poe, HP Lovecraft and Mary Wollstonecraft Shelley; to learn from, build a good Machine Learning model that identifies patterns in the way the three authors write and make predictions on which of the three authors would have written an unseen excerpt/sentence.

Inputs: The input to the different Machine Learning models will be a bag of words representation. Different combinations of weighting schemes such as Term Frequencies and Term Frequency - Inverse Document Frequencies along with n-gram[2] models will be fed to the Machine Learning models. A description of the weighting schemes used and the n-gram feature vectors will be described below. The choices of the Machine Learning models will be described in the **Methods** section.

1. Bag-of-unigrams:
Consider a sentence `'data mining is an ocean of unfathomable depth'`.
The bag of words[3] uni-gram representation of the above sentence will be, `['an', 'data', 'depth', 'is', 'mining', 'ocean', 'of', 'unfathomable']`
Each unique word or punctuation if it does exist in the sentence will now be considered a feature.

2. Bag-of-bigrams:
Another way a feature vector can be created from a sentence is by constructing bi-grams[3] instead of uni-grams. Instead of treating each word as a feature, a bi-gram of words model creates a sequence of two adjacent words as a single feature. For the sentence `'data mining is an ocean of unfathomable depth'`, the bi-gram feature vector will be, `['data mining', 'mining is', 'is an', 'an ocean', 'ocean of', 'of unfathomable', 'unfathomable depth']`

3. Bag of uni-grams and bi-grams together:
Instead of using each type of n-gram[3] separately, another input would be the inclusion of both uni-grams and bi-grams together as one feature vector. For the sentence `'data mining`

`is an ocean of unfathomable depth'`, the bi-gram feature vector will be, `['an', 'an ocean', 'data', 'data mining', 'depth', 'is', 'is an', 'mining', 'mining is', 'ocean', 'ocean of', 'of', 'of unfathomable', 'unfathomable', 'unfathomable depth']`

Feature Weighting: To understand the different feature weighting schemes being used in this project, we will consider the bag of uni-grams model. There are many different ways of weighting each words. The following two ways which are being used for this project is described in detail below.

- **Term Frequencies**
  As the name suggests, a simple way to weigh the importance of a word is by simply counting the number of occurrences of the word in each sentence.

docID	text
1	this was good
2	this was very good
3	this was bad
4	this was very bad
5	this was very bad very bad

Figure 3. An example corpora to demonstrate Weighting Schemes

  Consider the corpora above in Figure 3. Once all sentences including their punctuation (if any) is converted to a BoW and we used Term Frequencies alone as a weighting scheme, below is Figure 4 that helps visualize how each word in from each sentence in each *docID* has been weighted.

	bad	good	this	very	was	docID
0	0	1	1	0	1	1
1	0	1	1	1	1	2
2	1	0	1	0	1	3
3	1	0	1	1	1	4
4	2	0	1	2	1	5

Figure 4. Term Frequency Weighting Scheme

  As seen in Figure 4, each term/word is a feature. For each *docID*, the weight of the term is simply the number of times the term has appeared in that *docID*.

  A possible problem with this weighting approach is that, each term irrespective of how commonly it appears in all the documents is given a weight based on simply the number of times it appears. A lot of these words which appear in almost all documents may be considered noise. One way to deal with noise is using another different weighting scheme that has been explained next.

- **Term Frequency - Inverse Document Frequency (TF-IDF)**

  For a term i in document j
  $w_{i,j} = \mathrm{TF}_{i,j} \times \log \frac{\text{Total no. of documents}}{\text{Number of documents containing the term } i}$

  where,
  $w_{i,j}$: weight of a term i in document j
  $\mathrm{TF}_{i,j}$: no. of occurences of term i in document j
  The formula above is used to weight each term/word appearing in each document. So if Figure 3 represents the corpora, then the TF-IDF weighting scheme will produce a weighting for each term as seen in Figure 5.

	bad	good	this	very	was	docID
0	0.00	0.77	0.45	0.00	0.45	1
1	0.00	0.65	0.38	0.54	0.38	2
2	0.70	0.00	0.50	0.00	0.50	3
3	0.58	0.00	0.41	0.58	0.41	4
4	0.67	0.00	0.24	0.67	0.24	5

Figure 5. TF-IDF Weighting Scheme

  Comparing the two waiting schemes by looking at Figure 5 and Figure 4, it is seen from TF-IDF weighting scheme on Figure 5 that commonly appearing words such as `this` is given a low score in TF-IDF weighting scheme than in Term Frequency weighting scheme.

3.1 Data Analysis

The original data is a csv file. The csv file has 3 columns, namely, `id`, `text`, `author`.

	id	text	author
0	id26305	process however afforded means ascertaining di...	EAP
1	id17569	never occurred fumbling might mere mistake	HPL
2	id11008	left hand gold snuff box capered hill cutting ...	EAP
3	id27763	lovely spring looked windsor terrace sixteen f...	MWS
4	id12958	finding nothing else even gold superintendent ...	HPL

Figure 6. First 5 records of the csv file

The `id` column is simply an identifier for a given row of text.

The `text` column contains an excerpt from a book; the author of which is labelled in the next column.

The `author` is the column that represents the name of the author that wrote the excerpt. `EAP` is used to represent the author Edgar Allan Poe, `MWS` is used to represent the author Mary Wollstonecraft Shelley and `HPL` is used to represent the author HP Lovecraft.

In total, there are `19579` texts. There are no missing values in the data. Some text may contain dialogues between fictional characters in the excerpt, some text may contain diacritical marks[1]

3.1.1 Number of Excerpts by Author

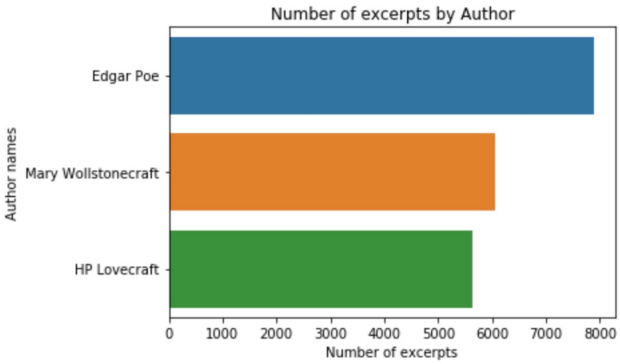


Figure 7. Number of Excerpts by Author

Out of 19579 excerpts there are:

1. 7900 excerpts from the author `Edgar Allan Poe`.

2. 5635 excerpts from the author `HP Lovecraft`.

3. 7900 excerpts from the author `Mary Wollstonecraft Shelley`.

3.1.2 Number of Excerpts containing Dialogues by Author

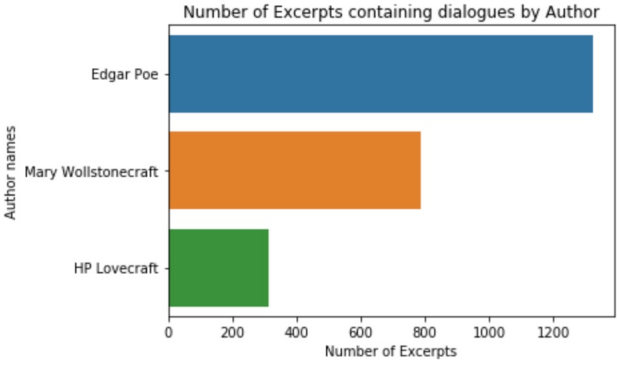


Figure 8. Number of Excerpts containing Dialogues by Author

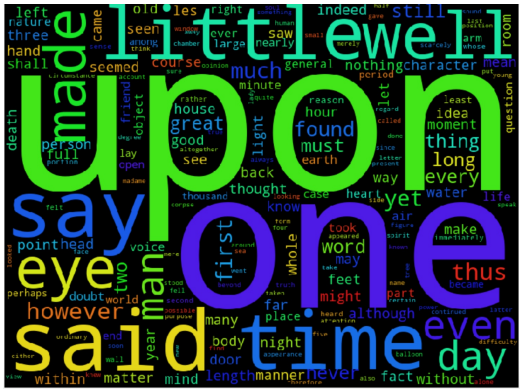
1. Out of 7900 excerpts from the author `Edgar Allan Poe`, 1326 contain dialogues.

2. Out of 5635 excerpts from the author `HP Lovecraft`, 313 of them contain dialogues.

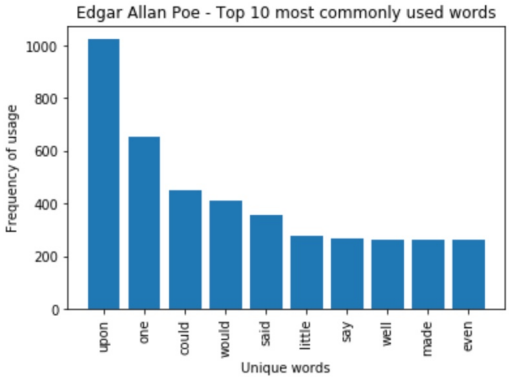
3. Out of 7900 excerpts from the author `Mary Wollstonecraft Shelley`, 788 of them contain dialogues.

3.1.3 Most commonly used words by Author: (WORDCLOUDS and BARGRAPHS)

The following photographic representations will help understand the most used words by each of the three authors in our dataset.

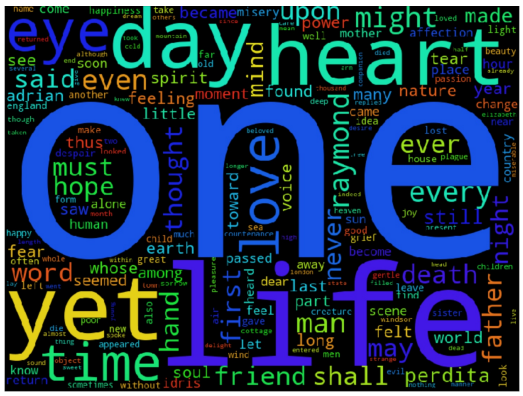


(a) Edgar Allan Poe WORDCLOUD

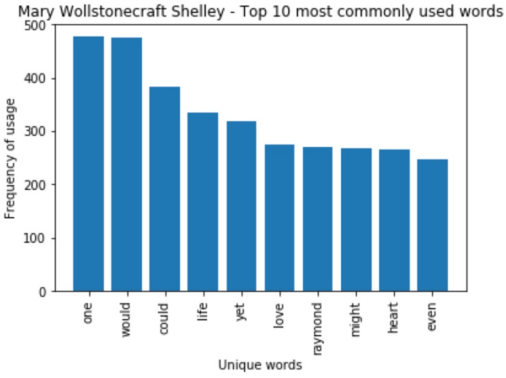


(b) Top 10 most commonly used by EAP

Figure 9. Most commonly used words by Edgar Allan Poe

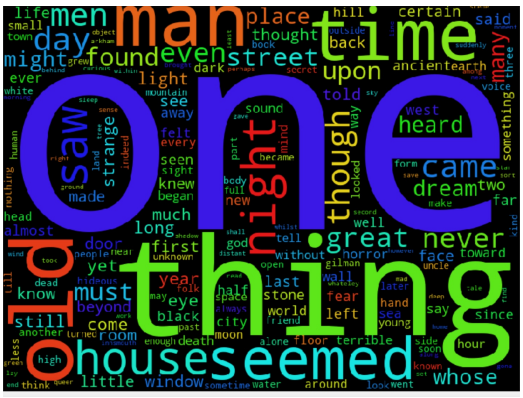


(a) Mary Wollstonecraft Shelley WORDCLOUD

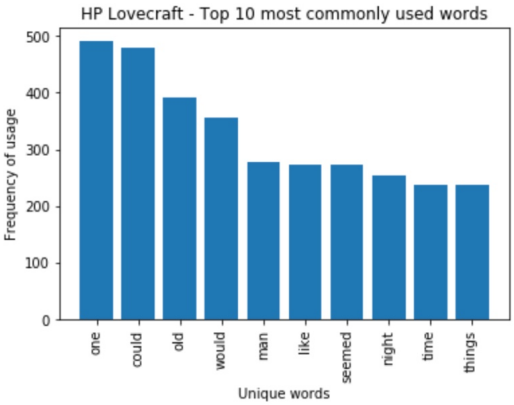


(b) Top 10 most commonly used by MWS

Figure 10. Most commonly used words by Mary Wollstonecraft Shelley



(a) HP Lovecraft WORDCLOUD



(b) Top 10 most commonly used words by HPL

Figure 11. Most commonly used words by HP Lovecraft

3.2 Preprocessing

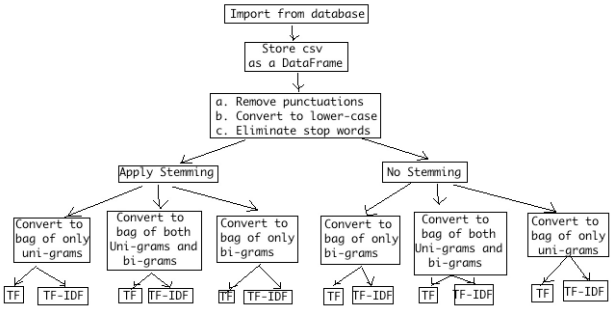


Figure 12. Preprocessing

`STEP 1`: Read data from the database to a DataFrame.

`STEP 2`:
a. Remove punctuation of all text.
b. Convert all text to lowercase.
c. Eliminate stop words.

`STEP 3: WITH STEMMING`
a. Convert to bag of only uni-grams.
b. Convert to bag of only bi-grams.
c. Convert to bag of uni-grams and bi-grams together.

`STEP 4:`
On `a`, `b`, and `c` from `STEP 3`,
a. Compute `TERM FREQUENCY WEIGHTING SCHEME.`
b. Compute `TF-IDF WEIGHTING SCHEME.`

`STEP 5:` Repeat from `STEP 3` to `STEP 4` but this time `WITHOUT STEMMING.`
From Figure 12, it can be seen that 12 different array of bag of words with different weighting schemes and different n-gram models are obtained.

`STEP 6:`
Split each of the 12 different bag of words array into 66.6% and 33.3% split for training and testing.

`STEP 7:`
Fit the training set of each of the 12 arrays of bag of words to each of the machine learning models in the next section.

3.3 Methods

3.3.1 k-Nearest Neighbor

In kNN algorithm[4], a test instance is assigned the same class as that of the majority of its nearest neighbours. To measure 'nearness' we use a distance metric called the Euclidean distance. A brief, high-level overview[4] of the kNN algorithm has been illustrated in Figure 13 to get an overall idea.

```
INPUT: 'K' number of neighbors, unseen
       observation 'x', Similarity
       Distance Metric (Euclidean),
       Training data.

OUTPUT: Class label with maximum
        probability
        for the observation 'x'


STEP 1: Compute Similarity between 'x'
        and the entire training dataset.
        Commonly used Similarity Metric
        is the Euclidean Distance.

STEP 2: A = 'K' points in the
        training set that are closest
        to 'x'.

STEP 3: Find conditional probability
        of each class (The number of
        points in set 'A' with the
        corresponding class label.)

STEP 4: Set the unseen document 'x'
        to that class where majority
        of the 'K' neighbors in its
        set 'A' have the maximum
        probability.
```

Figure 13. kNN algorithm - A high-level overview

When an instance whose class is unknown is presented for evaluation, the algorithm computes its K closest neighbors, and the class is assigned by voting among those neighbors. For multiple classes, a majority voting takes place.

Reason for usage:

- Based on simple intuition, kNN looks at similarity in terms of nearness in distance. Two pieces of text that are similar are likely written by the same author.
- Classes do not have be separable by a straight line.

Challenges and Disadvantages

- If authors have many commonly used words in sentences, that poses a major issue, since the similarity between two sentences that have sentences common to both the authors may be hard to classify correctly.

3.3.2 Multinomial Naive Bayes

Naive Bayes is a probabilistic machine learning method. The probability of a document d being in class c is computed[5] as

$$P(c|d) \propto P(c) \prod_{a \leq k \leq n_d} P(t_k c)$$

where $P(t_k c)$ is the conditional probability of term t_k occurring in a document of class c. We interpret $P(t_k|c)$ as a measure[6] of how much evidence t_k contributes that c is the correct class. $P(c)$ is the prior probability of a document occurring in class c. If a document's terms do not provide clear evidence for one class versus another, we choose the one that has a higher prior probability. $\langle t_1, t_2, ..., t_{n_d} \rangle$ are the tokens in d that are part of the vocabulary[7] we use for classification and n_d is the number of such tokens in d. In text classification, our goal is to find the best class for the document. The *best* class in NB classification is the most likely or *maximum a posteriori*[5] (MAP) class c_{map}.

$$c_{\text{map}} = \arg\max_{c \in \mathbb{C}} \hat{P}(c|d) = \arg\max_{c \in \mathbb{C}} \hat{P}(c) \prod_{1 \leq k \leq n_d} \hat{P}(t_k|c).$$

Figure 14. Maximum a posteriori

Below is a step by step example of how text is classified using Multinomial Naive Bayes.

Consider a corpus as show in Figure 15.

	text	label
0	Chinese Beijing Chinese	yes
1	Chinese Chinese Shanghai	yes
2	Chinese Macao	yes
3	Tokyo Japan Chinese	no

Figure 15. An example corpora to demonstrate Multinomial Naive Bayes for Text Classification

Before trying to apply the Naive Bayes classifier, convert this into a Bag of Words *BoW* (uni-gram).

	beijing	chinese	japan	macao	shanghai	tokyo	label
0	1	2	0	0	0	0	yes
1	0	2	0	0	1	0	yes
2	0	1	0	1	0	0	yes
3	0	1	1	0	0	1	no

Figure 16. *BoW* of the example corpora in Figure 10

$$P\left(word \,\middle|\, label\right) = \frac{\text{Count}(word, label) + 1}{\text{Count}(word) + |V|}$$

Count$(word, label)$: No. of times the *word* has appeared in all documents belonging to the class *label*.

Count$(word)$: Total number of words including duplicates that appear in all documents belonging to the class *label*.

$|V|$: No. of unique words in the vocabulary.

In our case $|V| = 6$

Given the above formula,

$$P\left(Chinese \,\middle|\, yes\right) = \frac{5+1}{8+6} = \frac{3}{7} = 0.42857142857142855$$

In the above fashion, we find the conditional probabilities of all the unique words given all the classes.

```
P( beijing |  no ) = 0.111111111111
P( chinese |  no ) = 0.222222222222
P( japan |  no ) = 0.222222222222
P( macao |  no ) = 0.111111111111
P( shanghai |  no ) = 0.111111111111
P( tokyo |  no ) = 0.222222222222
P( beijing |  yes ) = 0.142857142857
P( chinese |  yes ) = 0.428571428571
P( japan |  yes ) = 0.0714285714286
P( macao |  yes ) = 0.142857142857
P( shanghai |  yes ) = 0.142857142857
P( tokyo |  yes ) = 0.0714285714286
```

Figure 17. Prior probabilities of all the words in the corpora

Now let us say a new sentence `'Chinese Chinese Chinese Tokyo Japan'` needs to be classified to either the label *yes* or *no*.
Transform the incoming sentence again into a Bag of Words.

	chinese	japan	tokyo
0	3	1	1

Figure 18. Bag of Words for a test case

$$P\left(yes \,\middle|\, sentence\right) = P(yes) \times P\left(chinese \,\middle|\, yes\right)^3 \times P\left(japan \,\middle|\, yes\right) \times P\left(tokyo \,\middle|\, yes\right)$$

$$P\left(no \,\middle|\, sentence\right) = P(no) \times P\left(chinese \,\middle|\, no\right)^3 \times P\left(japan \,\middle|\, no\right) \times P\left(tokyo \,\middle|\, no\right)$$

The *sentence* belongs to that class *label* which satisfies

$$\texttt{max}\left[P\left(yes \,\middle|\, sentence\right), P\left(no \,\middle|\, sentence\right)\right]$$

$$P\left(yes \,\middle|\, sentence\right) = \frac{3}{4} \times 0.4285^3 \times 0.07142^1 \times 0.07142^1 = 0.0003$$

$$P\left(no \,\middle|\, sentence\right) = \frac{1}{4} \times 0.2222^3 \times 0.2222 \times 0.2222 = 0.0001$$

$$\texttt{max}(0.0003, 0.0001) = 0.0003$$

Therefore the *sentence* `'Chinese Chinese Chinese Tokyo Japan'` is classified as label *yes*.

It is also important to note that, if a sentence from a test set contains a word which is not seen in the vocabulary of training set, then that word is assigned a very low probability

Reason for usage

- Naive Bayes is known to be a standard baseline machine learning model for comparison of other machine learning models.

3.3.3 Logistic Regression

A stark difference from Naive Bayes, which generates the conditional probability $P\left(y \,\middle|\, x\right)$ using Bayes rule is that Logistic Regression tries to estimate $P\left(y \,\middle|\, x\right)$ directly[5]. Consider binary classification labels, *yes* or a *no* and each example is represented by a feature vector x. The intuition is to create a mapping between x to a real number, such that a highly positive number means x is likely to belong to class *yes* and a highly negative number means means x is likely to belong to class *no*. This is obtained by the operation $\theta^T x$ (the inner product). To bring it down to a probability between range 0 and 1; a *sigmoid function* is used.
It is often to see it represented as $\theta_0 + \theta^T x$.

$$p\left(y = yes \,\middle|\, x\right) = \sigma(\theta^T x)$$

$$\sigma(\theta^T x) = \frac{1}{1+\exp(-\theta^T x)}$$

$$p\left(y \,\middle|\, x\right) = \frac{1}{1+\exp(-y\theta^T x)}$$

Logistic regression can be extended to multiple classes. Suppose there are K classes. Each class K will have its corresponding θ_k, which maps x to its corresponding $\theta^T x$.

$$p(y = k|x) = \frac{\exp(\theta_k^\top x)}{\sum_{i=1}^{K} \exp(\theta_i^\top x)}.$$

Figure 19. Probability for a given class K

Finding the optimum θ can be done by maximizing the *conditional log likelihood* of training data. This is also commonly referred to as conditional **maximum likelihood estimation**[8].

To avoid over-fitting. A regularization term is added to the objective function.

L2 regularization is a quadratic function of the weight values.

L1 regularization is a linear function of the weight values.

$$\max_{\theta} \sum_{i=1}^{n} \log p(y_i|x_i, \theta).$$

Figure 20. Maximum Likelihood Estimation

Reason for usage

- While Naive Bayes is a generative classifier that is known for generating the data x from class y, Logistic Regression is a discriminative classifier that discriminates among the different possible values of the class y.

3.4 Evaluation Metric

Although the Kaggle competition specifies the usage of multiclass log loss as the evaluation metric, for this academic submission, we have considered the use of Accuracy as the metric for model performance. Accuracy is given by the percentage of predicted classes whose predictions match the true label.

3.5 Hardwares, Softwares and Packages

```
HARDWARE: MacBook Pro (Retina, 13-inch,
Early 2015)
OPERATING SYSTEM: macOS Sierra
Version 10.12.6
PROGRAMMING LANGUAGE: Python 3.6.2 ::  Anaconda
custom (64-bit)
```

`scikit-learn v 0.19.1`[9]

- `sklearn.feature_extraction.text.CountVectorizer`
- `sklearn.feature_extraction.text.TfidfVectorizer`
- `sklearn.metrics.accuracy_score`
- `sklearn.preprocessing`
- `sklearn.decomposition`
- `sklearn.model_selection`
- `sklearn.pipeline`
- `sklearn.naive_bayes.MultinomialNB`
- `sklearn.model_selection.train_test_split`
- `sklearn.linear_model.LogisticRegression`
- `sklearn.neighbors.KNeighborsClassifier`

`nltk v 3.2.5`[10]

- `nltk.stem.porter.PorterStemmer`
- `nltk.tokenize.word_tokenize`

`matplotlib v 2.1.0`[11]

- `matplotlib.pyplot.plt`

`pandas v 0.21.1`[12]

`numpy v 1.13.0`[11]

`wordcloud v 1.3.1`[13]

- `wordcloud.WordCloud`
- `wordcloud.STOPWORDS`

,

3.6 Results

Figures 21, 22 and 23 illustrates the combination of word inflection removals, n-gram feature vectors and weighting schemes that yielded the top 4 accuracy for the methods Logistic Regression, Naive Bayes and kNN respectively.

WORD INFLECTION REDUCTION	n-GRAM FEATURE VECTOR	WEIGHTING SCHEME	ACCURACY
WITHOUT STEMMING	ONLY UNIGRAM	COUNT	0.813680
WITHOUT STEMMING	UNI AND BIGRAM	TFIDF	0.806716
WITHOUT STEMMING	UNI AND BIGRAM	COUNT	0.806097
WITHOUT STEMMING	ONLY UNIGRAM	TFIDF	0.803002

Figure 21. Best of Logistic Regression

WORD INFLECTION REDUCTION	n-GRAM FEATURE VECTOR	WEIGHTING SCHEME	ACCURACY
WITHOUT STEMMING	UNI AND BIGRAM	COUNT	0.836738
WITHOUT STEMMING	ONLY UNIGRAM	COUNT	0.835036
WITHOUT STEMMING	UNI AND BIGRAM	TFIDF	0.827298
WITHOUT STEMMING	ONLY UNIGRAM	TFIDF	0.822191

Figure 22. Best of Naive Bayes

WORD INFLECTION REDUCTION	n-GRAM FEATURE VECTOR	WEIGHTING SCHEME	ACCURACY
WITH STEMMING	ONLY UNIGRAM	COUNT	0.444909
WITH STEMMING	UNI AND BIGRAM	COUNT	0.437790
WITHOUT STEMMING	ONLY UNIGRAM	COUNT	0.435314
WITHOUT STEMMING	UNI AND BIGRAM	COUNT	0.413959

Figure 23. Best of kNN

The following will be 20 other graphs that helps visualize changes in accuracy with changes in combinations of method used, n-gram feature vector and weighting scheme.

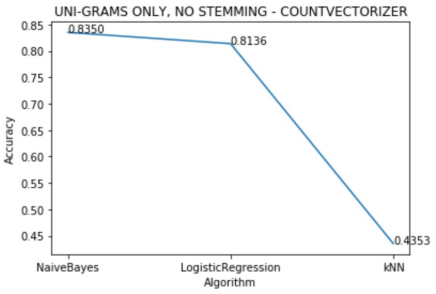


Figure 24. Changes in Accuracy with different models: UNI-GRAMS ONLY, NO STEMMING - COUNTVECTORIZER

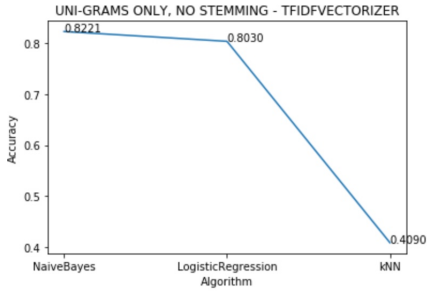


Figure 25. Changes in Accuracy with different models: UNI-GRAMS ONLY, NO STEMMING - TFIDFVECTORIZER

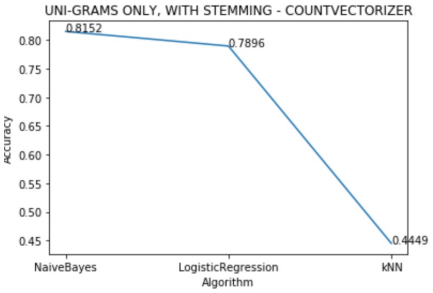


Figure 26. Changes in Accuracy with different models: UNI-GRAMS ONLY, WITH STEMMING - COUNTVECTORIZER

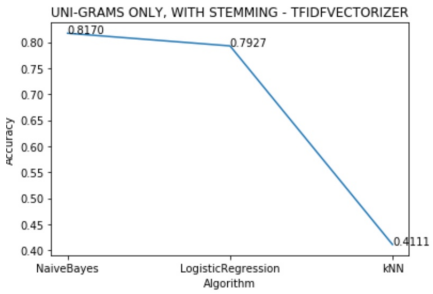


Figure 27. Changes in Accuracy with different models: UNI-GRAMS ONLY, WITH STEMMING - TFIDFVECTORIZER

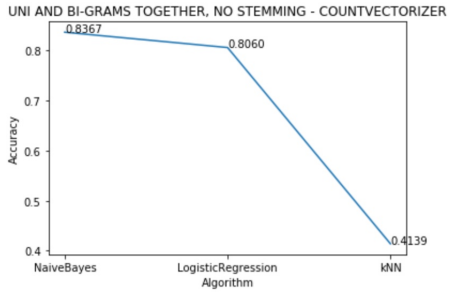


Figure 28. Changes in Accuracy with different models: UNI AND BI-GRAMS TOGETHER, NO STEMMING - COUNTVECTORIZER

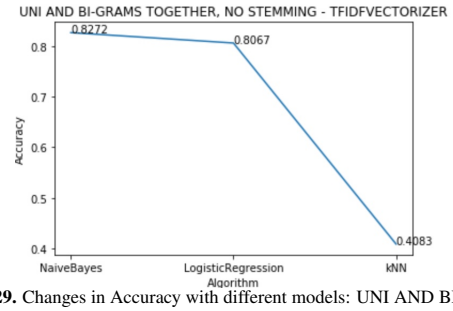


Figure 29. Changes in Accuracy with different models: UNI AND BI-GRAMS TOGETHER, NO STEMMING - TFIDFVECTORIZER

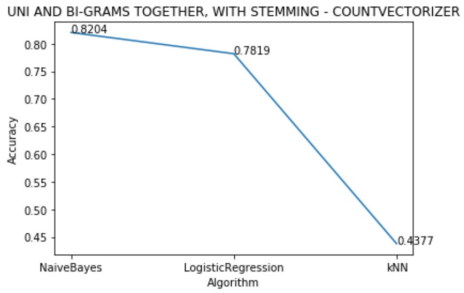


Figure 30. Changes in Accuracy with different models: UNI AND BI-GRAMS TOGETHER, WITH STEMMING - COUNTVECTORIZER

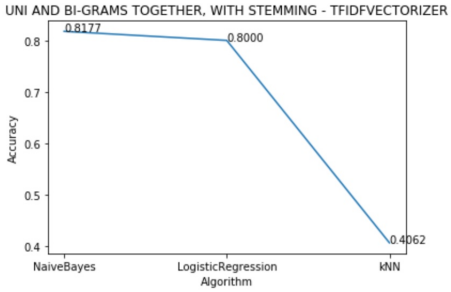


Figure 31. Changes in Accuracy with different models: UNI AND BI-GRAMS TOGETHER, WITH STEMMING - TFIDFVECTORIZER

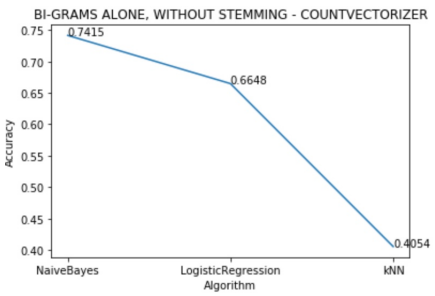


Figure 32. Changes in Accuracy with different models: BI-GRAMS ALONE, WITHOUT STEMMING - COUNTVECTORIZER

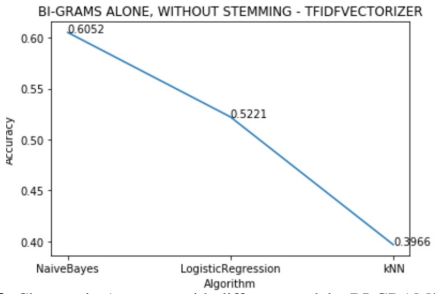


Figure 33. Changes in Accuracy with different models: BI-GRAMS ALONE, WITHOUT STEMMING - TFIDFVECTORIZER

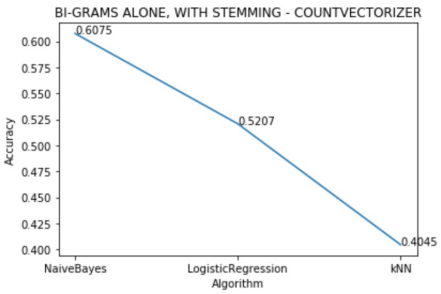


Figure 34. Changes in Accuracy with different models: BI-GRAMS ALONE, WITH STEMMING - COUNTVECTORIZER

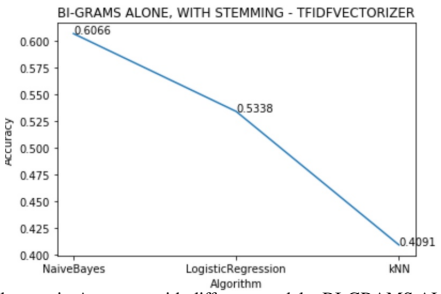


Figure 35. Changes in Accuracy with different models: BI-GRAMS ALONE, WITH STEMMING - TFIDFVECTORIZER

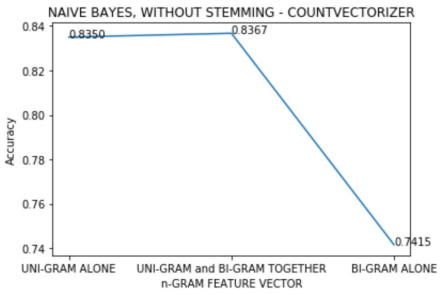


Figure 36. Changes in Accuracy of NAIVE BAYES, WITHOUT STEMMING - COUNTVECTORIZER: UNI-GRAM ALONE, UNI-GRAM and BI-GRAM TOGETHER, BI-GRAM ALONE

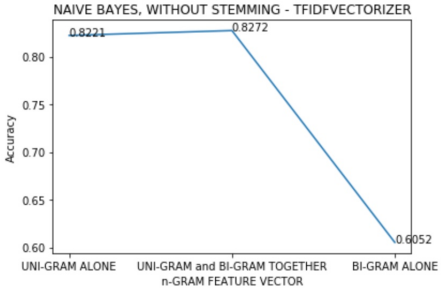


Figure 37. Changes in Accuracy of NAIVE BAYES, WITHOUT STEMMING - TFIDFVECTORIZER: UNI-GRAM ALONE, UNI-GRAM and BI-GRAM TOGETHER, BI-GRAM ALONE

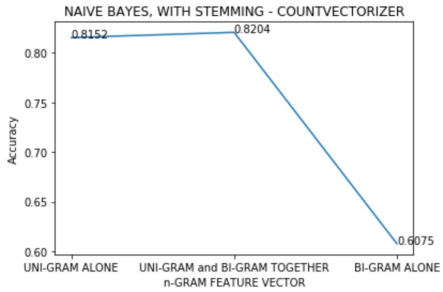


Figure 38. Changes in Accuracy of NAIVE BAYES, WITH STEMMING - COUNTVECTORIZER: UNI-GRAM ALONE, UNI-GRAM and BI-GRAM TOGETHER, BI-GRAM ALONE

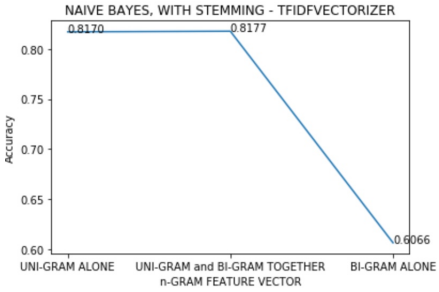


Figure 39. Changes in Accuracy of NAIVE BAYES, WITH STEMMING - TFIDFVECTORIZER: UNI-GRAM ALONE, UNI-GRAM and BI-GRAM TOGETHER, BI-GRAM ALONE

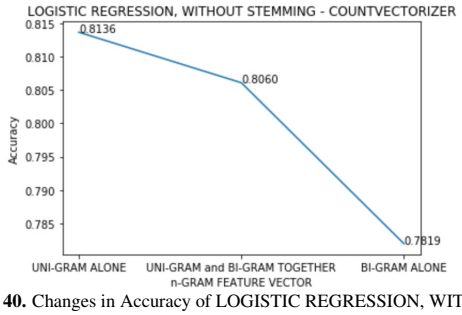


Figure 40. Changes in Accuracy of LOGISTIC REGRESSION, WITHOUT STEMMING - COUNTVECTORIZER: UNI-GRAM ALONE, UNI-GRAM and BI-GRAM TOGETHER, BI-GRAM ALONE

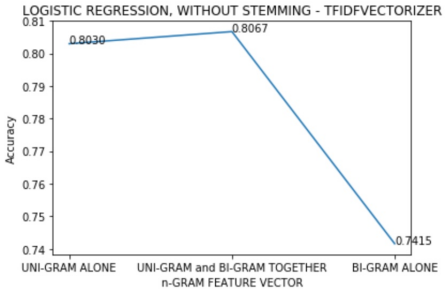


Figure 41. Changes in Accuracy of LOGISTIC REGRESSION, WITHOUT STEMMING - TFIDFVECTORIZER: UNI-GRAM ALONE, UNI-GRAM and BI-GRAM TOGETHER, BI-GRAM ALONE

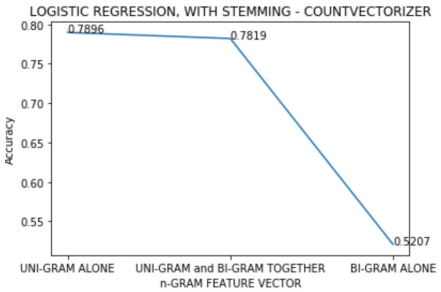


Figure 42. Changes in Accuracy of LOGISTIC REGRESSION, WITH STEMMING - COUNTVECTORIZER: UNI-GRAM ALONE, UNI-GRAM and BI-GRAM TOGETHER, BI-GRAM ALONE

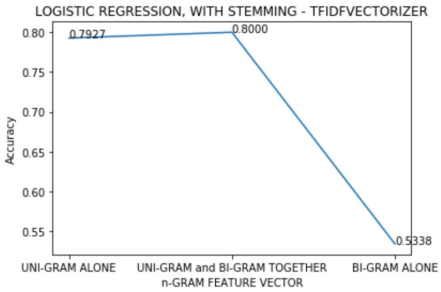


Figure 43. Changes in Accuracy of LOGISTIC REGRESSION, WITH STEMMING - TFIDFVECTORIZER: UNI-GRAM ALONE, UNI-GRAM and BI-GRAM TOGETHER, BI-GRAM ALONE

- Do you deem this successful?

  The accuracy score of the best model is pretty decent (83.6%). However, Naive Bayes is known to be a base

model for text classification to compare other text classification algorithms with, so it is to be expected that Naive Bayes would perform very well for text categorization. No novel approach was tried.

- What do the results suggest?
  - The models that produce the best accuracy score are the ones where the words are not stemmed. Stemming tends to destroy a unique aspect of each feature/word that made it important.
  - A combination of uni-grams and bi-grams tend to give the highest accuracy. In other words, more useful information about an author is being captured when we consider uni and bi-grams together.
  - Consider a bi-gram as a single word, just like stemming masks a lot of information contained in the individual words.

- What were challenges?

  The sheer number of features that were being generated when we considered uni-grams and bi-grams together made many machine learning methods very slow.

3.7 Author Identification - Data Mining Summary

Figure 44 describes a flowchart of the entire process of Data Mining for the project Spooky Author Identification.

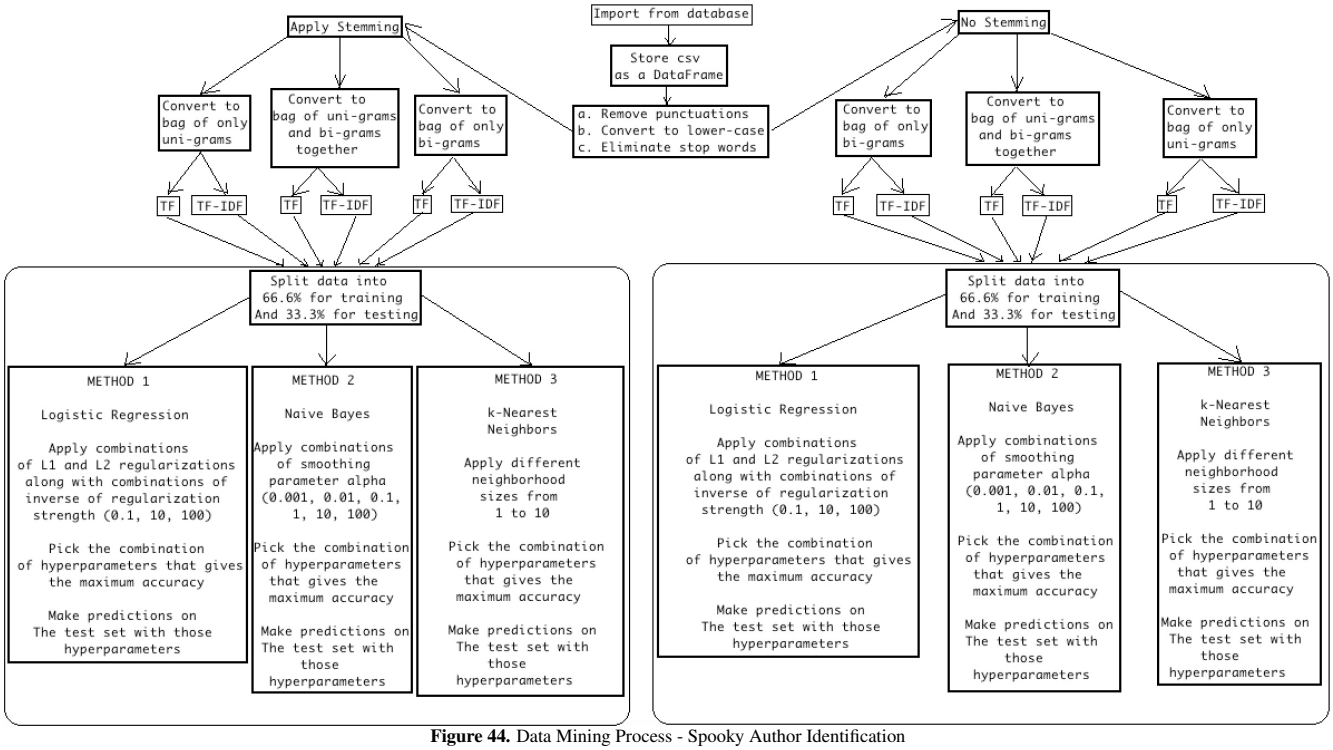


Figure 44. Data Mining Process - Spooky Author Identification

3.8 Future Work

The datamining process done for Spooky Author identification is a standard way to approach any problem in Natural Language Processing. More categorical features such as `'Publishing date of the book containing the excerpt'` can be scraped from appropriate websites that have the information in the database. One of the challenges encountered in scraping the `'Publishing date of the book containing the excerpt'` is that, the books were published in the 1990s, and as a result, in 2017, works from authors have been consolidated into a collection.

It is hard to decipher `'Published on'` information since this information is not indexed and hence hard to wrangle them. Some of the works of the three authors are hidden deep in the internet because they are very old. Apart from performing Stemming, the bag of words[3] features can be constructed by performing Lemmatization[3].

Other machine learning models such as Decision Trees, AdaBoost and approaches such as Latent Dirichlet Allocation can be attempted. Models apart from Vector Space models can be put to use.

4. Iceberg: Full Problem Description

4.1 The Data and its Significance

In this challenge, we work on the satellite image data provided by Statoil to predict whether a target sensed through satellites is a `ship` or an `iceberg`. The early detection of icebergs in the proximity of ships will lead to safe working condition for the people on these ships and it will reduce the operation costs due to damages done to ships by these icebergs. The satellites used for capturing these images are located around 600 kilometres above earth. The group of satellites used for monitoring are called the Sentinel constellation. The satellite orbits the Earth 14 times a day and captures the images. These satellites are equipped with C-Band radars that are able to capture image in darkness, rain, cloud and various other weather conditions.

These satellites radars emits signal on towards earth and captures the echo after it bounces off some object. This reflected data is converted into an image. So, any object will appear more brighter in an image compared to its surroundings because it reflects back more energy. The issue in image recognition occurs because an iceberg is also a solid object and it reflect back almost same amount of energy as a ship does.

The radar cannot tell the difference between a ship and an iceberg. Hence, the features in the image need to be properly analyzed for features like shape, size, brightness etc. The water bodies surrounding the target can also be analyzed to make better predictions. Factors such as speed of wind affects the back-scatter of the oceans and other water bodies.

Another factor which affects the images that is captured by the radars is that they are side looking. This means that images captured by the satellite are not perpendicular images but images taken at an angle. The radar takes images from a horizontal plane as well as a vertical plane. When we discuss about the dataset, we will discuss about data from these 2 channels *i.e.* `HH(transmit/receive horizontally)` and `HV(transmit/receive vertically)`. These channel data can be combined to produce good quality images for making predictions.

4.2 Details of the dataset

We have been provided with two datasets i.e. train.json and test.json. The data is provided in `json` format as opposed to the standard `csv` format which is more popular in Kaggle competitions. These files consists of details about the images captured by the satellites.

Details of the fields in the dataset:

- `id` : id of the image
- `band_1` : this is the captured image data converted into flattened format. The image consists of 75×75 pixels, so after flattening, this field has `5625` elements. The value in these fields are float numbers with their unit in decibels. They have a significant physical meaning with respect to the image. This field represents the radar backscatter by HH(transmit/receive horizontally) polarization.
- `band_2` : This field represents radar backscatter by HV(transmit/receive vertically) polarization.
- `inc_angle` : This field represents the inclination angle at which the image was taken.
- `is_iceberg` : This is the target variable details present in the training.json file. It has two possible values : 0 and 1. 0 indicates a `ship` and 1 indicates an `iceberg`.

4.3 Exploratory Data Analysis

We will now look at the data and try to make some sense out of it. We will try to figure out those attributes that should be considered as a part of the dataset while we try to train our models, so that we obtain the highest accuracy.

The `train.json` files contains the training data. It contains **1604** rows of data and **5** columns. There are a number of missing values in the field `'inc_angle'`. We will process them properly. The `test.json` data file consists of **8424** rows of data, each containing **4** fields. There are no missing values in the test set.

[3] Lemmatization is a method of grouping together the grammatical representation forms of a word so they can be analyzed as a single item. For *e.g.* the words *happy, happier happiest* is reduced to *happy*.

	band_1	band_2	id	inc_angle	is_iceberg
0	[-27.878361, -27.15416, -28.668615, -29.537971...	[-27.154118, -29.537888, -31.0306, -32.190483,...	dfd5f913	43.9239	0
1	[-12.242375, -14.920305, -14.920363, -12.66633...	[-31.506321, -27.984554, -26.645678, -23.76760...	e2538fd	38.1562	0
2	[-24.603676, -24.603714, -24.871029, -23.15277...	[-24.870956, -24.092632, -20.653963, -19.41104...	58b2aaa0	45.2859	1
3	[-22.454607, -23.082819, -23.998013, -23.99805...	[-27.889421, -27.519794, -27.165262, -29.10350...	4cfc3a18	43.8306	0
4	[-26.006956, -23.164886, -23.164886, -26.89116...	[-27.206915, -30.259186, -30.259186, -23.16495...	271f93f4	35.6256	0

Figure 45. Snapshot of training set

```
Data columns (total 5 columns):
band_1          1604 non-null object
band_2          1604 non-null object
id              1604 non-null object
inc_angle       1604 non-null object
is_iceberg      1604 non-null int64
```

Figure 46. Details of training set

	band_1	band_2	id	inc_angle
0	[-15.863251, -15.201077, -17.887735, -19.17248...	[-21.629612, -21.142353, -23.908337, -28.34524...	5941774d	34.966400
1	[-26.0589694977, -26.0589694977, -26.058969497...	[-25.7542076111, -25.7542076111, -25.754207611...	4023181e	32.615072
2	[-14.1410999298, -15.0642414093, -17.375520706...	[-14.745639801, -14.5904102325, -14.3626976013...	b20200e4	37.505433
3	[-12.167478, -13.706167, -16.54837, -13.572674...	[-24.32222, -26.375538, -24.096739, -23.8769, ...	e7f018bb	34.473900
4	[-23.3745937347, -26.0271816254, -28.121963501...	[-25.7223434448, -27.0115776062, -23.149162292...	4371c8c3	43.918874

Figure 47. Details of training set

```
band_1          8424 non-null object
band_2          8424 non-null object
id              8424 non-null object
inc_angle       8424 non-null float64
```

Figure 48. Details of training set

Now, let us try to see if the values in the target values are balanced. A balanced target variable ensures that our predictions are good. A skewed target variable set might lead to incorrect predictions.

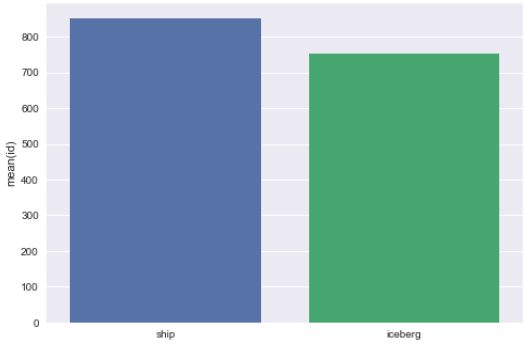


Figure 49. Comparison between target labels

As we can see, the number of ships and icebergs in the dataset are almost evenly distributed. Here we see a direct correlation in the rows in which '`inc_angle`' is null and its target value. An absence of the '`inc_angle`' value classifies the image as a ship, in each and every case. We will try to handle this correlation in the next section.

Now, let us try to visualize the images given in the records.

Visualizing iceberg images from `band_1`:

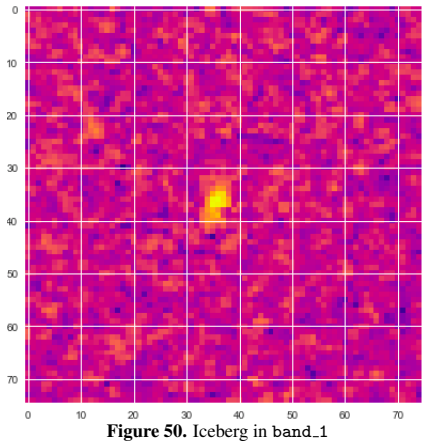


Figure 50. Iceberg in `band_1`

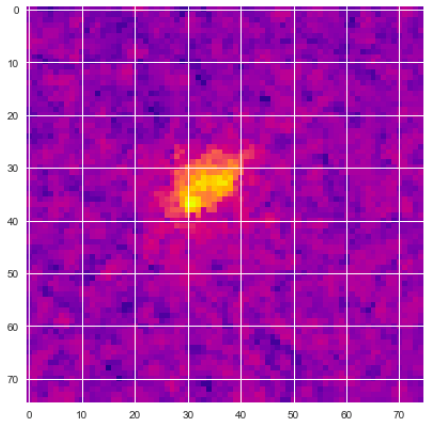


Figure 51. Iceberg in `band_1`

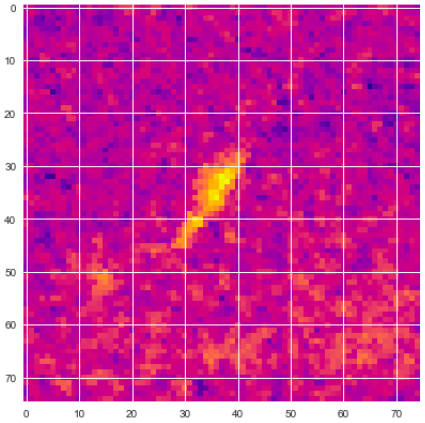


Figure 52. Iceberg in `band_1`

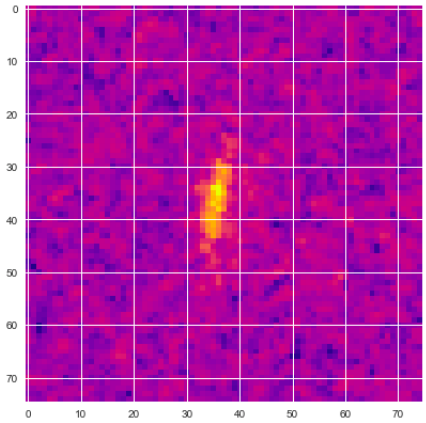


Figure 53. Iceberg in `band_1`

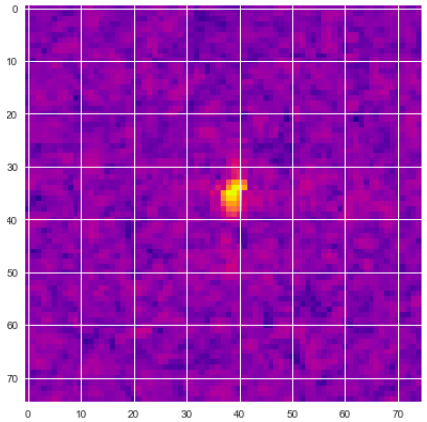


Figure 56. Ship in `band_1`

Visualizing ship images from `band_1`:

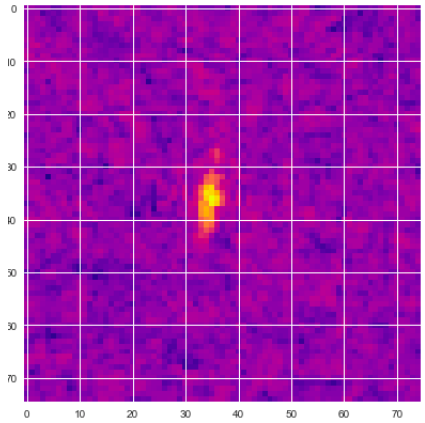


Figure 54. Ship in `band_1`

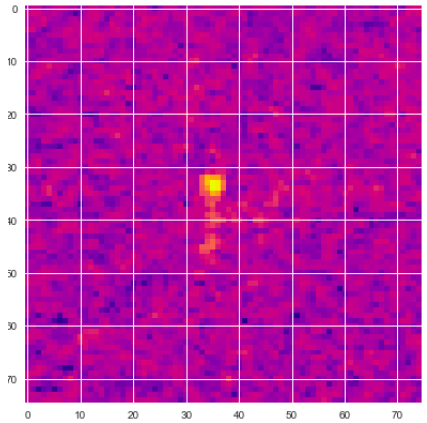


Figure 57. Ship in `band_1`

Visualizing iceberg images from `band_2`:

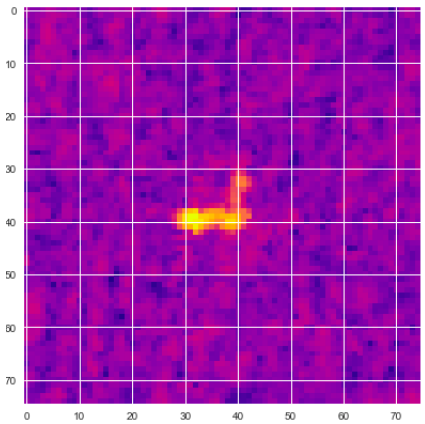


Figure 55. Ship in `band_1`

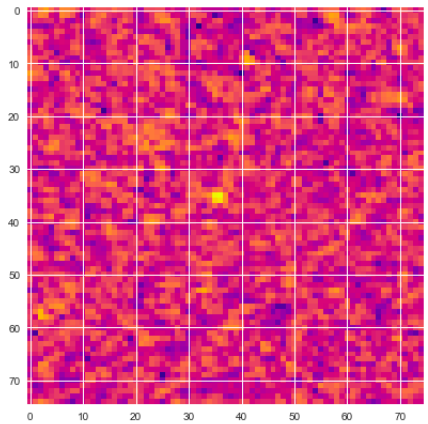


Figure 58. Iceberg in `band_2`

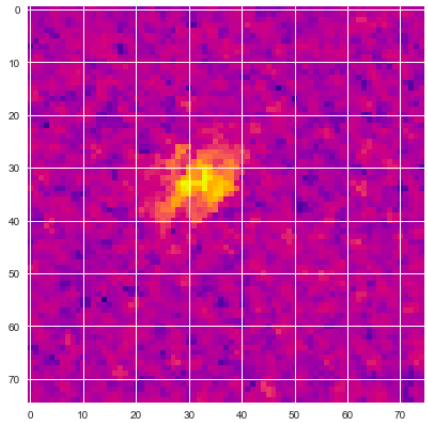


Figure 59. Iceberg in band_2

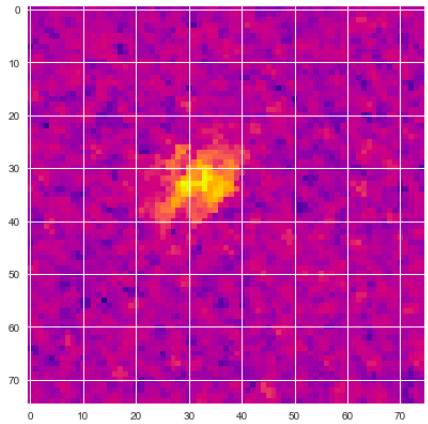


Figure 60. Iceberg in band_2

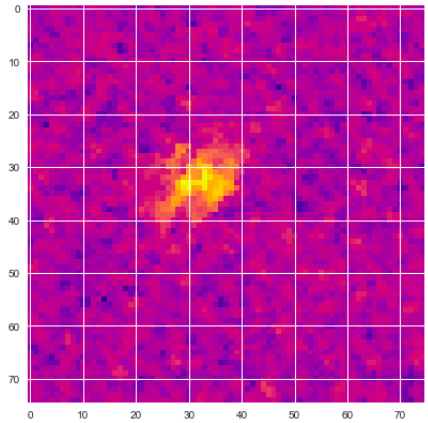


Figure 61. Iceberg in band_2

Visualizing ship images from band_2:

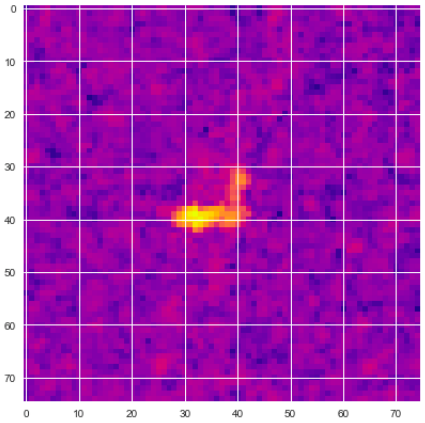


Figure 62. Ship in band_2

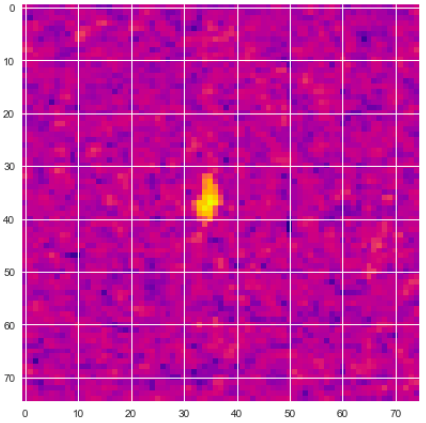


Figure 63. Ship in band_2

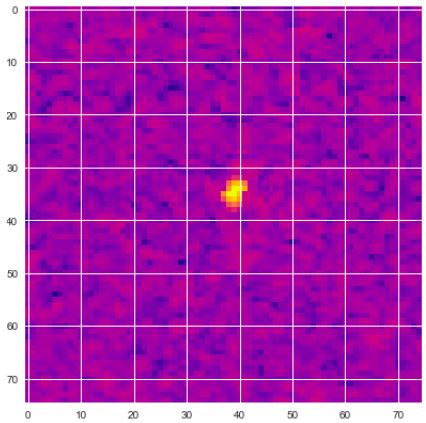


Figure 64. Ship in band_2

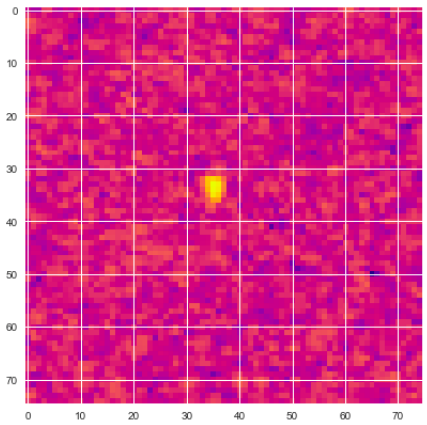


Figure 65. Ship in band_2

As we can see, in some of the images, we can see the object clearly and in some others, the objects are not clearly distinct from the background. Now, let us dive into feature selection. We will try to extract some features from the training set and figure out which of the attributes should be used for training the classifiers.

4.4 Feature Selection

The features obtained after steps mentioned in this section are used for Logistic Regression.

- Before doing feature extraction, we replace the null values in the 'inc_angle' field to NA
- Find various statistical parameters from the two bands i.e. band_1 and band_2. The parameters that we extract from both these band are :
  - minimum value for each band
  - maximum
  - median
  - standard deviation
  - mean
  - 25th percentile
  - 75th percentile
  - 50th percentile

After finding these values, we try to find correlation between them, to check which features are highly correlated with each other. Following plot shows the correlation between various features:

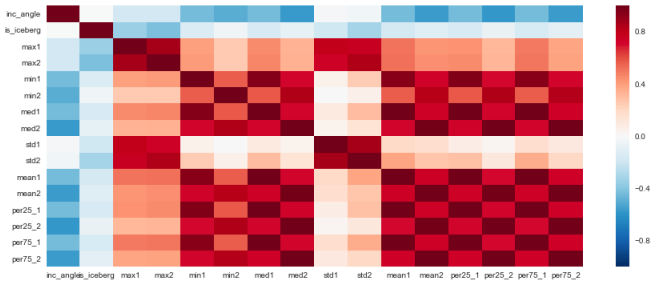


Figure 66. Plot showing correlation

Looking at the above correlation plot, we will now try to find the most important features in our new dataset. We will reject the features which have a high correlation with each other.

- max1 and max2 seem to have high correlation. So, we will consider max1 and drop max2
- min1 has high correlation with med1, mean1, per25_1, per75_1. So we will keep min1 and drop the rest
- min2 is retained
- med2 is retained but mean2, per25_2, per75_2 are dropped.
- std1 and std2 are retained in the dataset.

The retained features in this step are used for building the Logistic Regression model.

4.5 Methods

4.5.1 Logistic Regression

Logistic Regression[14] is a classification algorithm which is used for binary or multiclass classification where the target variable is categorical in nature. The Logistic Regression model tries to predict the probability of a dependent variable based on a number of independent variables.

At the core of Logistic Regression is the logistic function:

$$\delta(t) = \frac{e^t}{(e^t+1)} = \frac{1}{(1+e^{-t})}$$

where t is any real input. t can be expressed as combination of explanatory variables.

$$t = \beta_0 + \beta_1 x$$

The logistic function can now be represented as:

$$F(x) = \frac{1}{1+e^{-(\beta_0+\beta_1 x)}}$$

Reason for usage:
Image classification is usually done using neural networks. But Logistic Regression seems like a good choice too. In this challenge, there are only two possible target variables. Hence, Logistic Regression may be good with binary classification problems.

4.5.2 Convolutional Neural Networks

One of the models used for the Iceberg problem is Convolutional Neural Networks[15]. CNNs have proven to be a good choice when it comes to automatic image classification[16].

The top layer features of CNN are used in classification, but might not have enough information to predict a given image accurately.

In a convolutional layer, the neurons specifically look for certain features and on finding those, they produce high activation.

Activation Function

The activation function used here is the Rectifier activation function. The unit that employs the rectifier is called Rectifier Linear Unit, `ReLU`.

The rectifier is a function that is defined as the positive part of its argument:

`Output = max(0, x)`, x being the input to the neuron.

Feature Scaling

Since the CNN layer is majorly based in finding features, it is important to have the data on the same scale, as different scales for different features will affect the ability of the model to learn and predict. Once the features are standardized, it ensures that all features are represented on an equal scale.

The data in the columns $band_1$ and $band_2$ are a 75×75 matrix of pixel values and standardizing these features is important for this problem. The re-scaling method we have used here is :

`x' = (x - min(x)) / max(x) - min(x)`
where x is the original value and x' is the normalized value.

Part of the code where rescaling is done :

```
b1_rescale = (band_1 - band _1.min()) /
(band_1.max() - band_1.min())
```

```
b2_rescale = (band_2 - band_2.min()) /
(band_2.max() - band_2.min())
```

Keras Model Details

Model - Sequential

The Sequential model is a stack of layers, all linear. A model can be created by specifying the input and output shape, activation function, parameters for the layers, `batch_size` etc.

We can always add different layers to the model with the `.add()` method.

Layers - Dense, Flatten, Activation, Dropout

Dense: Takes the activation function as an argument, creates a kernel which is basically a weight matrix and calculates the output using the two parameters. No bias has been included in this calculation.

Dropout: A regularization technique used in neural networks where randomly selected neurons are dropped, which means their contribution towards the activation of downstream neurons is disabled. This can be very easily implemented in `Keras` with the help of the Dropout function in the Layers. We have taken a value of 0.3 for all the layers of this CNN.

Flatten: Used to flatten the input.
Activation: The activation function used here is the Rectifier activation function. The unit that employs the rectifier is called Rectifier Linear Unit, `ReLU`.

Callbacks - EarlyStopping, ModelCheckpoint

Keras gives us the flexibility to understand the internal states of the model during training. This is done using a set of functions called 'Callbacks'. In our model we have used two such functions.

EarlyStopping: We would not want our model to run after a point where we stop seeing any improvement. EarlyStopping helps us monitor specific quantities, measure changes in the quantity that can be called as an improvement. We are monitoring `'val_loss'` with the mode set to **min**, since we want to stop training when the loss has stopped decreasing.

ModelCheckpoint: Similar to what EarlyStopping does in monitoring quantities, but it checks and saves the model state after every epoch. The number of epochs initially used was 10, which gives us an average accuracy of 74.10%

Optimizer - Adam

The Adam optimizer is a first-order gradient-based optimizer that employs stochastic objective functions. It demands very little memory requirements and is very efficient computationally.

The values of the parameters used for the optimizer are listed below :

Learning rate:
`lr = 0.01`

4.6 Hardwares, Softwares and Packages

```
HARDWARE: MacBook Pro
OPERATING SYSTEM: macOS El Capitan
Version 10.11.6
PROGRAMMING LANGUAGE: Python 2.7.13 ::  Anaconda
4.3.0 (64-bit)
```

`scikit-learn v 0.18.1`[9]

- `sklearn.model_selection`
- `sklearn.linear_model.LogisticRegression`
- `sklearn.metrics`
- `sklearn.cross_validation.cross_val_score`
- `sklearn.cross_validation.train_test_split`
- `sklearn.metrics.log_loss`

`numpy v 1.13.3`[17]

`seaborn v 0.7.1`[18]

`matplotlib v 2.0.0`[11]

- `matplotlib.pyplot.plt`

`pandas v 0.19.2`[12]

`Keras`[19]

- `Model : Sequential`
- `Layers - Dense, Flatten, Activation, Dropout`
- `Callbacks - EarlyStopping, ModelCheckpoint`
- `Optimizer - Adam,`
- `Backend - Tensorflow`

4.7 Results

Average Accuracy for **CNN**:

Epochs	Dropout	Validation Split	Avg Accuracy
10	0.3	0.25	74.10%
20	0.3	0.25	79.04%
30	0.3	0.25	81.70%

Average accuracy for **Logistic Regression** after performing 10 fold cross-validation :

Run	Accuracy%
1	0.72222222
2	0.70186335
3	0.70807453
4	0.73125
5	0.0.7125
6	0.66875
7	0.7125
8	0.725
9	0.66875
10	0.55625

Average Accuracy is calculated to be 0.690716011042

Precision/Recall Details for Logistic Regression:

	precision	recall	f1-score	support
0	0.66	0.66	0.66	242
1	0.66	0.66	0.66	240
avg / total	0.66	0.66	0.66	482

Figure 67. Details of Precision and Recall

The average accuracy of CNN is $\sim$**80%** whereas the average accuracy of Logistic Regression is $\sim$**70%**. The difference in accuracy is expected as CNN performs better with image classification. The accuracy of Logistic Regression can be further improved by doing better feature selection.

Acknowledgments

We would like to thank Professor Mehmet Dalkilic and Marcin Malec for their guidance and support in completing this project.

References

[1] Patrick Pantel Peter D. Turney. J.r. quinlan. *Kluwer Academic Publishers, Boston*, 1:81–106, 1986.

[2] Robert L. Mercer Vincent J. Della Pietra Jenifer C. Lai Peter F. Brown, Peter V. deSouza. Class-based n-gram models of natural language. *IBM T. J. Watson Research Center*.

[3] Patrick Pantel Peter D. Turney. From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research*, 37(11).

[4] Robert Tibshirani Hastie, Trevor and Jerome H. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2001.

[5] Prabhakar Raghavan Christopher D. Manning and Hinrich Schütze. *Introduction to Information Retrieval*. 2008.

[6] Jingbho Zhu Le Zhang and Tianshun Yao. An evaluation of statistical spam filtering techniques. *ACM Transactions on Asian Language Information Processing*, 3(4), 2004.

[7] Hae-Chang Rim Sang-Bum Kim, Kyoung-Soo Han and Sung Hyon Myaeng. Some effective techniques for naive bayes text classification. *IEEE Transactions on Knowledge and Data Engineering*, 18(11), 2006.

[8] Dan Jurafsky and James H. Martin. *Speech and Language Processing*. 2001.

[9] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[10] Edward Loper Bird, Steven and Ewan Klein. *Natural Language Processing with Python*. 2007.

[11] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing In Science & Engineering*, 9(3):90–95, 2007.

[12] Wes McKinney. Data structures for statistical computing in python. In Stéfan van der Walt and Jarrod Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 51 – 56, 2010.

[13] Andreas C. Müller. Wordcloud. *MIT*, 2011.

[14] Wendy A. Bergerud. *Introduction to Logistic Regression Models*. BRITISH COLUMBIA.

[15] Jianxin Wu. Introduction to convolutional neural networks. 2017.

[16] Geoffrey E. Hinton Alex Krizhevsky, Ilya Sutskever. Imagenet classification with deep convolutional neural networks.

[17] Eric Jones, Travis Oliphant, Pearu Peterson, et al. SciPy: Open source scientific tools for Python, 2001–. [Online; accessed ¡today¿].

[18] Michael Waskom.

[19] François Chollet.