Data Science
School of Informatics, Computing, and Engineering
Indiana University, Bloomington, IN, USA
Data Mining B565 Fall 2017
All the work herein is solely ours

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

Rishab Nagaraj[1], Vivek Vikram Magadi[2], Pramod Duvvuri[3]

Executive Summary The problems under consideration for this project have been taken from https://www.kaggle.com/competitions. Both problems involve large data-sets that need to be processed before any classification or identification can be performed on them.
As a solution, Data Mining works as the system must be able to pick out patterns in the data that will 'teach' it what to look for during prediction. The two problems under review here are:

1. Author Identification - Model a system that will be able to predict whether an excerpt from a story was written by Edgar Allan Poe, Mary Shelley or HP Lovecraft.
2. Iceberg Classification - Model a system that will be able to identify whether an item (detected by satellite) is either a ship or an iceberg.

Keywords
Classification — Natural Language Processing — Data Mining — Machine Learning

[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*
[3] *Data Science, School of Informatics, Computing, and Engineering, Indiana University, Bloomington, IN, USA*

Contents

1. Introduction

- The problems have been taken from https://www.kaggle.com/competitions. Some challenges even have cash prizes that depend on the problem's complexity and/or urgency. The challenges under consideration for this project are the Author Identification and the Iceberg Classifier Challenge.
- These challenges involve a data-set containing a large amount of data. As such, the data needs to be processed and so, Data Mining is always the first step to solving these problems. At the most basic level, these problems

can be reduced to patterns. The system needs to be trained to be able to recognize these patterns and match them to a label through the use of a classifier. This is what Data Mining is all about i.e., making sense of unintelligible data.

- 1. The Author Identification problem has excerpts of texts from each of the three authors who are identified as 'EAP', 'HPL' and 'MWS'. In order to gauge which excerpt belongs to which author, we need to identify patterns in writing styles. This project has been implemented using the Naive Bayes Classifier and the data has been processed using the Bag of Words model. Other techniques were also used and are elaborated upon in this paper.

  2. The Iceberg Classification problem has flattened image data - $band_1$ and $band_2$. Each band has 75×75 pixel values that are respresented as float values with the unit dB. Band 1 and Band 2 are signals characterized by radar backscatter produced from different polarizations at a particular incidence angle. The polarizations correspond to HH (transmit/receive horizontally) and HV (transmit horizontally and receive vertically).

1.1 Author Identification

This project aims to parse through excerpts from stories written by three different authors, namely Edgar Allen Poe, HP Lovecraft and Mary Shelley and then given an excerpt, the system will be able to predict which author wrote the excerpt [18].
The data here has three columns - 'id', 'text' and 'author'. 'id' is the ID number of each excerpt, 'text' is the excerpt and 'author' is a three letter code that identifies the author i.e., EAP, HPL or MWS.
Goodness here is quantified as:

1. Train/Test splits - We use different train/test splits in our project [1]. We have various train/test splits in our project. When the data-set is split into a 70-30 ratio we usually mean we reserve 70% of the data for training our model and the remaining 30% of the data testing/predicting, this allows us to know the overall accuracy of our model. We use the *sklearn.model_selection.train_test_split* implementation in our project for the train/test splits.

2. Accuracy - The accuracy of our model is measured using the *sklearn.metrics.accuracy_score* metric. This accuracy score is the sum of correct predictions divided over the sample size.

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

> Once upon a midnight dreary, while I pondered, weak and weary, Over many a quaint and curious volume of forgotten lore— While I nodded, nearly napping, suddenly there came a tapping, As of some one gently rapping, rapping at my chamber door. "'Tis some visitor," I muttered, "tapping at my chamber door— Only this and nothing more."

from Edgar Allen Poe's, *The Raven*. The project aims to train the system such that it can assign an author to the above excerpt accurately depending on the word frequency from which we can obtain the probability that that word belongs to a certain author.

1.2 Statoil/C-CORE Iceberg Classifier Challenge

This project deals with icebergs drifting off the Canadian East Coast that could potentially pose threats to navigation and local activities in the area. Current methods of minimizing the risk posed by icebergs are aerial reconnaissance and shore based support monitors. These methods are less effective in areas prone to harsh weather. This leaves satellite imagery as the only other option. Hence this project deals with satellite radar data.
The data consists of 2 bands of radar signals emitted with a shift in the angle of incidence. The main task is to classify whether the given signalled data is an iceberg or a ship.
Here, we quantify goodness as:

1. Train/Test splits - We use a 50/50 split in our project i.e., we reserve 50% of the data for training the model and 50% for testing/predicting. This allows us to know the overall accuracy of our model. We use the *sklearn.model_selection.train_test_split* implementation in our project to split the dataset.

2. Accuracy - The accuracy of our model is measured using the *sklearn.metrics.accuracy_score* metric. This accuracy score is the sum of correct predictions divided over the sample size.

2. Data Mining

- What is Data Mining?
  Data Mining is essentially pattern recognition. It is the process where algorithms are used on data-sets to extract patterns. The final aim is to use these patterns and be able to predict future results.

- What does it yield?
  The outcome of performing Data Mining is that we are left with a system that, given data, can predict outcomes. For example, feeding weather data of Bloomington for the last 10 years into a Data Mining algorithm gives us a system that can quite accurately predict tomorrow's weather with a minimal error rate.

[1] Splitting the dataset into training and testing data allows for testing the accuracy on a partition of data for which the expected class is already known

- What are the general steps?
  The entire Data Mining process has 3 basic steps:
  1. Preprocessing - This is the most time consuming part of the process and involves acquiring the data as well as cleaning and enriching the data to get it into the required format.
  2. Mining - This is where the algorithms are executed in order to make sense out of the data and is the so called 'easiest' step.
  3. Interpretation & Validation - This is the step where the user tries to make sense of the data, test it to check the model's accuracy, visualize it, etc.
- What is clustering *vs.* classification?
  Clustering is a technique where similar objects are grouped together in order to see what relation they have with one another
  Classification is a technique where the objects are to be assigned to pre-defined labels in order to know how similar each object is to one of the pre-existing classes.
- What is a loss function?
  A loss function represents some 'cost' incurred by the system which shows by how much the system's predicted value differs from the actual observed values that we obtain for a given prediction.

2.1 Data Preprocessing

- What are the steps?
  1. Data Cleaning - Handle missing values, noisy data, outliers and inconsistent data, correct spelling mistakes, incorrect data and basically get the data in the right format.
     eg: Age stored as 9999, addresses stored as 'Circle', 'Cle' or 'Cir', random spikes in audio samples or radar data.
  2. Data Enrichment - Combining the data at hand with external sources to make more sense of it
     eg: Pincode given but not city, can enrich the data at hand by adding a variable for city to make it easier to locate.
  3. Data Transformation - Normalization and other methods to get the data in the format required to perform analysis on.
- What challenges does each step present?
  1. Data Cleaning - Data Entry Errors, Inconsistent Data, Contradictory Data. There is no standard determining the correct format these variables should be in. Different people can follow different standards.
  2. Data Enrichment - Creating a system to pull data from various sources and then. Can cause problems if a user has to parse data from different sources as each source will have to be linked to the enriching system.
  3. Data Transformation - Incorrectly formatting data can lead to broken code eg: Changing a currency symbol from $ to some other currency can change the outcome of some business prediction model if it equipped to work differently with $ and other currencies.

2.2 Mining, Interpretation, and Action

- Briefly discuss the top 10 algorithms [1].
  1. C4.5 - This algorithm not only generates classifiers as decision trees, but can also construct classifiers as rulesets.
     C4.5 uses the Divide-and-Conquer algorithm to grow the tree given a set S of cases
     (a) If all cases belong to one class or if S is small, the tree can be represented as a leaf with the highest frequency class as a label.
     (b) Else, pick a single attribute test with at least two outcomes. This test is now at the root with one branch for each outcome of the test. Partition S into subsets according to each case's outcome and recursively apply this procedure to all subsets.

     The process is completed by selecting a subset of simplified rules for each class. These subsets are ordered such that the error on the training cases is minimized and a default class is selected. The leaves of the pruned decision tree are generally much more than the rules of the final ruleset. The main disadvantage of using rulesets is the amount of memory space and CPU time required.
  2. k-Means - This algorithm iteratively partitions a dataset into k clusters. The algorithm is initialized by picking k real-valued points as the initial k "centroids". Techniques for selecting these seeds include random sampling from the dataset. Then the algorithm iterates between two steps till convergence:
     (a) Data Assignment - All points are assigned to the closest centroid. Ties are arbitrarily broken.
     (b) Relocation of means - Every cluster's representative point is relocated to the mean of all of its data points.

     Convergence occurs when assignments no longer change.
     The default measure of closeness is the Euclidean distance which guarantees finite iterations. k-means' greedy nature implies that convergence is only to a local optimum.

3. SVM - Support Vector Machines offer among the highest accuracy and robustness.
   The aim is to find the best classification function to differentiate between members of two classes in the training data. A hyperplane exists that splits the classes. Due to there being many such hyperplanes, SVM guarantees that the best function is obtained by maximizing margins between the classes i.e., the shortest distance between the closest data points to a point on the hyperplane.
   The widest margin hyperplanes give the best generalization and allow for the best classification of current and future data.
4. Apriori - This algorithm finds high frequency itemsets using candidate generation. It is characterized by "If an itemset is not frequent, it's supersets are not frequent either". The algorithm assumes that items are sorted alphabetically. The algorithm scans the database and searches for the most frequently occurring itemsets of size 1 by incrementing count for each item and collecting those that satisy the minimum support requirement. It then iterates over the following steps:
   (a) Generate C_{k+1} candidates of frequent itemsets of size $k+1$, from the list of frequent itemsets of size k.
   (b) Scan the DB and calculate support of each candidate of the frequent itemsets.
   (c) Add the itemsets that satisfy minimum support requirement to F_{k+1}

   C_{k+1} is generated from F_k by the following:
   (a) Join - Generate R_{K+1}, the first candidates of frequent itemsets of size $k+1$ by taking the union of the two frequent itemsets of size k, P_k and Q_k that have the first k1 elements in common.
   (b) Prune - Check if all itemsets of size k in R_{k+1} are frequent and generate C_{k+1} by removing those that do not from $Rk+1$. This is because any subset of size k of C_{k+1} that is not frequent cannot be a subset of a frequent itemset of size $k+1$.

   The algorithm then finds all the candidates of the frequent itemsets included in transaction t. It calculates frequency only for those candidates by scanning the database. The algorithm scans the database at most k_{max+1} times when the maximum size of frequent itemsets is set at k_max. It achieves good performance by reducing the size of candidate sets, however, when there are a large number of frequent itemsets, large itemsets, or very low minimum support, it suffers from the cost of generating a huge number of candidate sets and scanning the database continuously. In fact, it is necessary to generate 2^{100} candidate itemsets to obtain frequent itemsets of size 100.
5. EM - Finite mixture distributions provide a flexible approach to modeling and clustering of data observed on random phenomena. We look at normal mixture models which are used to cluster continuous data and estimate the density function. These can be fitted by maximum likelihood via the Expectation Maximization Algorithm.
6. PageRank - Search ranking algorithm that uses hyperlinks on the internet. This is the algorithm that Google was built on.
   PageRank gives a ranking of web oages where the value is computed for each page and does not depend on user queries. In essence, it interprets a hyperlink from page x to y as a vote and counts number of votes towards page quality. It also analyzes the page that casts the vote. Votes from **important** pages ware weighted higher.
7. AdaBoost - This algorithm employs ensemble learning with a solid theoretical foundation, accurate predictions and is simple. (Ensemble learning deals with methods that use multiple learners to solve a problem.)
   (a) The algorithm assigns equal weights to all the items in the training set.
   (b) From the training set and the weight distributions, the algorithm generates a weak learner
   (c) Uses training set to test the weak learner and increases weights of incorrectly classified examples. An updated weight distribution is obtained.

   AdaBoost then uses the new distribution and the training set to generate a new weak learner and repeats the above steps T times where T is the number of learner rounds specified by the user.
   AdaBoost can **boost** a weak learner that performs just better than random guess into a strong learning algorithm.
8. kNN - A sophisticated version of the rote classifier (memorizes the training set and classifies only if attributes of test object exactly match a training example). k-Nearest Neighbors finds a group of k objects in the training set closest to the test object. Assignment is done based on predominance of a particular class in this neighborhood. The three key elements of this approach are - set of labeled objects, distance metric and k. To classify an unlabeled object, distance to labeled objects is calculated, its k nearest neighbours are identified and those class labels (majority class) determine the unlabeled object's label.
   If k is too small, result can be sensitive to noise

and if k is too large, neighborhood might include noise from other classes. Determining class labels by majority can also be an issue if the nearest neighbors vary widely in their distance and the closer neighbors more reliably indicate the object class. This can be avoided by weighted voting i.e., weight each object by distance from the node.
kNN is easy to implement and understand and despite its simplicity, it performs very well. Its error rate is actually bounded above by twice the Bayes error under certain assumptions.

9. Naive Bayes - Naive Bayes is easy to construct, so can be readily applied to large data sets. It is easy to interpret, so anyone can understand the classification it makes. It might not be the best classifier but is usually robust. It is a method of Supervised Classification where a rule is constructed for a set of objects belonging to known classes such that future objects can be assigned to a class.
Assume 2 classes, 1 and 0, and use the training set to construct a score such that the scores are associated with the right classes. This is achieved by by comparing the score with some threshold. $P(i|x)$ is the probability that an object with measurement vector x belongs to class i and can be decomposed as $f(x|i)P(i)$ where $f(x|i)$ is a conditional distribution of x for class i's objects and $P(i)$ is the prior probability of class i. These are the values we need to estimate to produce classifications. $P(i)$ can be estimated from from the proportion of class i objects of the training data while to estimate $f(x|i)$, the algorithm assumes the components of x are independent and then estimates the univariate distributions of each class separately.
10. CART - The CART (Classification and Regression Trees) decision tree is a binary recursive partitioning algorithm capable of processing continuous data as both targets and predictors. Data is handled raw i.e., no binning. Trees are grown to a maximal size and then pruned back to the root via cost-complexity pruning. The next split to be pruned is the one contributing least to the overall performance of the tree on training data. This procedure produces trees that are constant for all orderings thereby preserving predictor attribute transformations. CART is intended to produce sequence of nested pruned trees, all of which are optimal. The **honest** tree is identified by evaluating the predictive performance of every tree in the pruning sequence. Tree performance is always measured on independent test data or via cross validation and tree selection proceeds only once this evaluation is complete. If no test data exists and cross validation has not been performed, CART cannot specify which tree in the sequence is best. CART includes automatic class balancing, automatic missing value handling and allows for cost-sensitive learning, dynamic feature construction and probability tree estimation. The algorithm shows how cross validation can be used to assess performance for every tree in the pruning sequence.

- Does Data Mining tell us what to do?
Data Mining finds patterns in data. Nothing in the world is random. Everything has some inherent order to it whether we can comprehend what it is or not. Data Mining extracts these hidden patterns from data to answer questions about business practices or research or just general day to day life. It doesn't tell us what to do but it can predict the probability of certain events occurring and gives us the power of foresight by which we can make informed decisions and decide what we want to do.
For example, a beverage company wants to increase prices of a certain product of theirs but want to know how the increase would affect the sales of the product. Data Mining would help the company understand if increasing the price is a good option and if so by how much.
- What are some new types of problems in data-mining?
    1. Security and Privacy concerns - Every single digital move we make can be watched or accessed by anyone. Google, Facebook, phone companies, the government, etc have access to your digital identity.
    2. Cost - Procuring and maintaining software that can process large datasets and run algorithms efficiently will require significant costs.
    3. Poor Data Quality - Data is almost never in a usable form. Some preprocessing or transformation needs to be done to get it into a form we can use. Preprocessing is a very time-intensive process.
    4. Large datasets - In order to effectively use large datasets, need to have very good hardware, scalable and efficient algorithms, maybe splitting the dataset and work on it in batches, etc.

3. Author Identification: Full Problem Description

Author prediction has become a major topic in the field of Natural Language Processing these days. It helps us identify the most likely author of articles, news, messages,text and documents [18].

In this project, we try to predict the author of excerpts from horror stories by Edgar Allan Poe, Mary Shelley, and HP Lovecraft. The task at hand is to classify each line of text, i.e. predict the author, given a line of text. We tackle this problem

using the dataset provided by Kaggle, we try to predict the author using various machine learning models. The input for this project is the training dataset, which contains three attributes,the id, the text and the author of the text. The output in this project would be a single value i.e. the author of the text given a line of text as the input. Ideally, we would like to try different types of classifiers in our predictive model to see the accuracy/error score. We would split the input dataset and use it to train and test our model.

3.1 Data Analysis

The dataset contains three attributes,the id, the text and the author of the text. We use different splits to identify which split gives us the best accuracy, the splits we have used are 80/20, 70/30, 60/40, 50/50. There are no missing values in the dataset. There are 19579 lines in the input dataset. We add a new attribute called "author_label" and map the values in the column "author" to EAP = 0, HPL = 1, MWS = 2. The below bar-chart represent the frequency of the author in the training dataset. Before we fit our predictive model we need to do some preprocessing our input dataset, that is we need to transform the input dataset before we use a machine learning algorithm to train and predict the author. The next section in this document will discuss about the different transformation techniques used by the authors in this project.

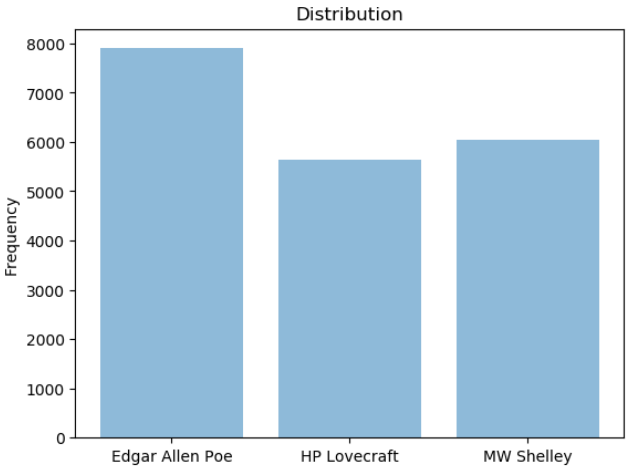

Figure 1. Author Frequency in Training Dataset

3.1.1 Transformations

The input text needs to be transformed before it can be used for training and testing, this is because most machine learning algorithms expect the input data to be in a numerical format. Hence, we need to transform our raw input text. This is done by using a *text feature extraction* method called *"The Bag of Words"* representation.

We use *sklearn.feature_extraction.text.CountVectorizer* [4] implementation of the bag of words model in our project. Most input text are not of the same fixed length, it would be difficult for an algorithm to handle this variable length problem, the bag of words representation solves this issue for us. The bag of words representation does this by doing the following:

3.1.2 Working of Bag of Words

- **Tokenizing** the input text is converted into tokens by using punctuation's and white-spaces as the delimiters.
- **Counting** the occurrences of each token is counted in each line of text
- **Normalizing** and weighting with diminishing importance tokens that occur in the each line.

A corpus of the text can thus be represented by a matrix with one row per line and one column per token (e.g. word) occurring in the corpus.

We call vectorization the general process of turning a collection of text into numerical feature vectors. This specific strategy (tokenization, counting and normalization) is called the Bag of Words or *"Bag of n-grams"* representation. The matrix represents the word occurrences while completely ignoring the relative position information of the words in the line of text.

We typically use a small subset of the words in the corpus, the resultant output matrix will have a lot of zeroes. This output matrix is a sparse matrix of type *scipy.sparse.csr_matrix*. This sparse matrix is used for faster algebraic operations.

Another method we employ to normalize the tokens is called *Term Frequency Inverse Document Frequency - (TFIDF)* . In this method, the number of occurrences of a word is multiplied by it's idf component, so that words with lesser significance such as the articles "a", "an", "the" do not shadow the frequencies of some rarer and more interesting words. This is achieved by a tf-idf transform. The below equations give a brief overview of math involved the TFIDF transformation

$$tf-idf(t,l) = tf(t,l) * idf(l)$$
$$idf(l) = log(\frac{1+n}{1+df(l,t)})$$

Where n is equal to the number of lines of text and $df(l,t)$ is the number of lines that contain the term t. The resultant tf-idf vectores are normalized using the euclidean norm. We use the *sklearnfeature_extraction.text.TfidfTransformer* implementation to transform our text for suitable usage by a classifier. We also tried multiple transformations on the input text data. That is we implemented the Bag of Words and TFIDF using a *Pipeline* [7] in our project.

We have used the *sklearn.pipeline.Pipeline* implementation to combine the above transformation on our data.

3.1.3 Natural Language Processing

Natural language processing (NLP) [2] is a field of computer science, artificial intelligence concerned with the interactions between computers and human languages, and, in particular, concerned with programming computers to fruitfully process large natural language data. It is very common to perform NLP before we train our predictive model. We use the built in NLTK [3] for our preprocessing [5] to remove some commonly used words as mentioned in TFIDF process, these common words are usually referred to as *stop words.* Another

common method as a part of NLP preprocessing is to perform *stemming* . Stemming refers to the process of reducing derived words. Example: ”running” is reduced to ”run”.

After we have transformed our input text, we have to choose an appropriate machine learning algorithm for our model. There are many classifiers available for us in machine learning. The next section of this document discusses the various methods we have used to predict the author.

3.2 Methods

The below flow diagram best describes the process we have adopted in this project.

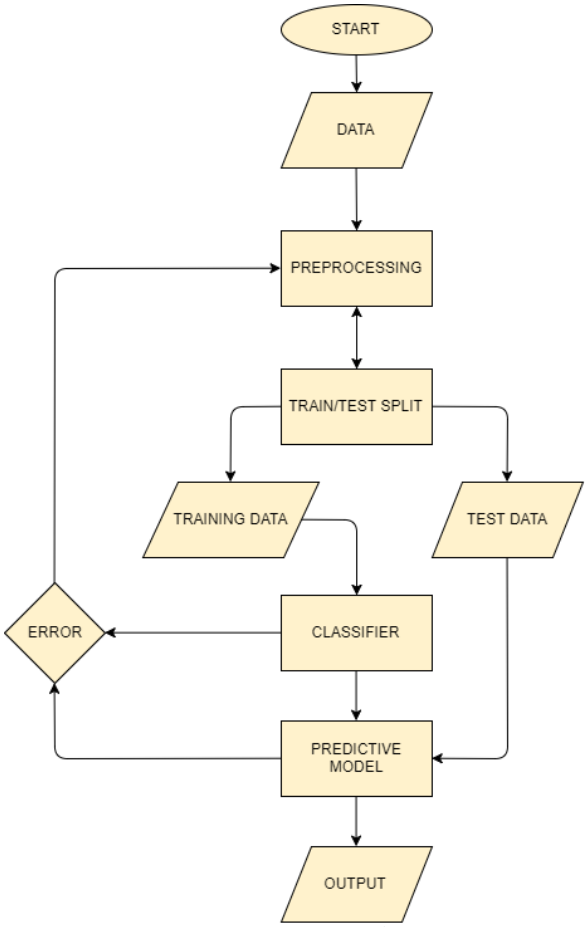

Figure 2. Process Flow

These are the various predictive machine learning models we have used for the task of author prediction after we have transformed the data.

3.2.1 Multinomial Naive Bayes

Multinomial Naive Bayes (MNB) [9] is a simple classifier that uses the Bayes Theorem and assumes independence between the features and is used for multinomially distributed data, since the data is represented in sparse matrix of word vectors this would be ideal for such scenarios. We shall use *sklearn.naive_bayes.MultinomialNB* implementation in our project. We have used various train/test splits to analyze the accuracy of our model. The same has been represented in terms of error in the plot below. The highest accuracy we got was when we used the 80/20 train-test split which gave us an error of 0.1542 and 90/10 split gave us an accuracy of 0.1425 (rounded to nearest decimal point)

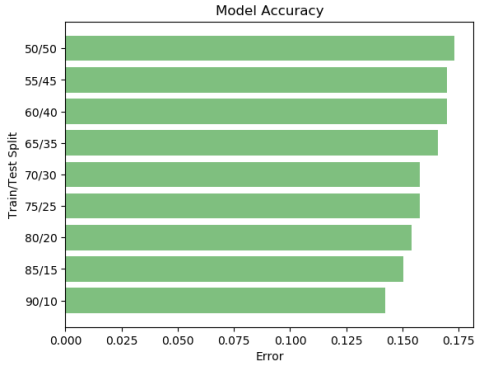

Figure 3. Error Comparison in Train/Test Splits

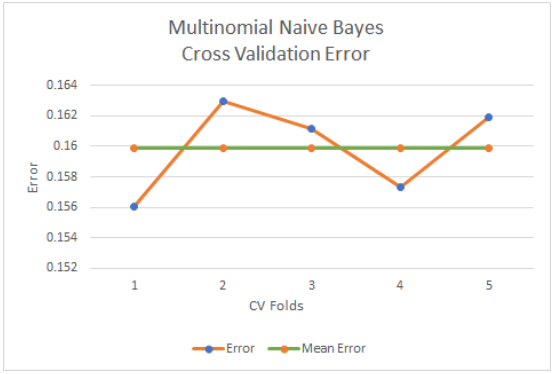

Figure 4. Cross Validation Result

3.2.2 Support Vector Machines (SVM)

Support Vector Machines [13] are predictive models in machine learning that are used for *Supervised Learning*. When the output variable is labelled it is called supervised learning, in our scenario we are the author name which is the label. SVMs are highly efficient in performing linear classification. SVMs are also very helpful in text classification, hence we have chosen to use them in our project.

An SVM uses a hyperplane or a set of hyperplanes to perform the task of classification. We have used two different implementations of SVMs in our project, *sklearn.svm.SVC* and *sklearn.linear_model.SGDClassifier* for author prediction. We shall also implement SVMs with Stochastic Gradient Boosting in our project, this method is discussed in the next sub-section.

3.2.3 Stochastic Gradient Boosting

Stochastic Gradient Descent (SGD) [14] is a simple technique for using linear classifiers under convex loss functions such as (linear) SVMs and Logistic Regression. The efficient application of SGD in large-scale and sparse machine learning

problems in recent times has made it popular. SGD has been widely used in NLP and text classification. This is reason why we have chosen SGD for this text classification problem. Also SGD is very easy to implement. For the parameters we use *elasticnet* for the penalty as it might bring sparsity to the model. We also use 100 as the value for the parameter max_iter. The below plots show the accuracy of our model when SGD is used for author prediction, we have used cross validation:

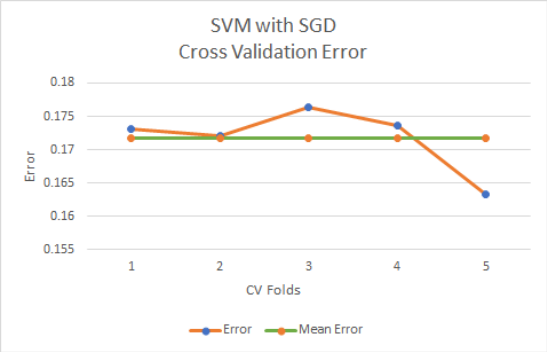

Figure 5. Model Accuracy with SGD with cv = 5

The below plot shows the error with cross validation, when the number of folds is equal to ten.

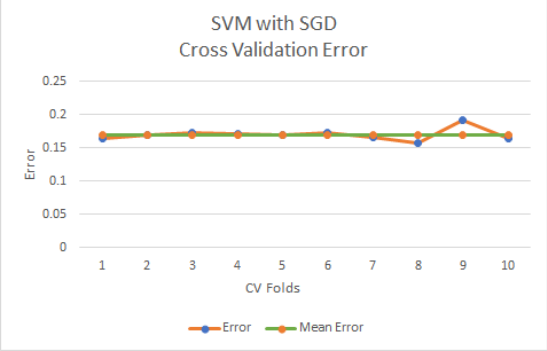

Figure 6. Model Accuracy with SGD with cv = 10

The model accuracy is show in the plot below for various train/test splits:

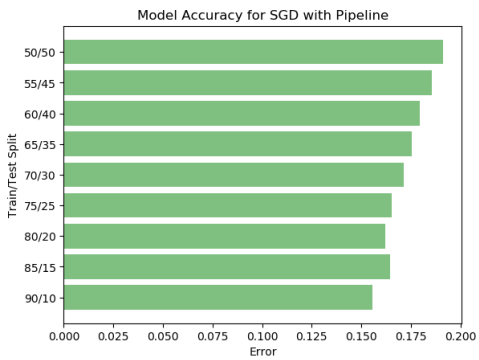

Figure 7. Train/Test Splits Comparison for SGD Classifier

3.2.4 Logistic Regression

Logistic Regression [12] is a machine learning algorithm that is used for regression and classification, it uses the natural logarithm function to find the relationship between the independent variable and the target variable, it uses test data to find the coefficients. The function can then predict the future results using these coefficients in the logistic equation. We use the *sklearn.linear_model.LogisticRegression* [12] implementation in our project.

3.2.5 Random Forest

Decision Trees are versatile Machine Learning algorithms that can perform both classification and regression tasks, and even multi-output tasks. They are very powerful algorithms, capable of fitting complex datasets. A Decision Tree is the fundamental concept behind a *Random Forest* [15].

In general, aggregation of answers tends to give a better result, same technique used in machine learning is called ensemble learning. Instead of using a single classifier we use a group of classifiers or predictors, these group of classifiers usually tend to give a better prediction. When a group of decision trees are trained on random subset of the training data we get better results. Such groups of decision trees or ensemble of decision trees are known as a "Random Forest". We use *sklearn.ensemble.RandomForestClassifier* [15] implementation in our project.

3.2.6 Bagging

We can always use a group of diverse classifiers in our predictive model. Another approach would be to just use one training algorithm and train it on different random subsets of the training dataset. When such different samples of the training data is used with replacement to train our model the process is known as *Bagging* [16]. The bagging process is very popular and frequently used in ensemble learning.

3.2.7 Voting Classifiers

This is another application of ensemble learning. The outputs of individual classifiers are aggregated and the class which gets the most votes is the prediction. When it is a majority vote, it is known as a *hard voting*. In most scenarios, the voting classifier always achieves a higher accuracy. We use *sklearn.ensemble.VotingClassifier* [17] implementation in our project. The below plots show the accuracy in different voting classifier we have used:

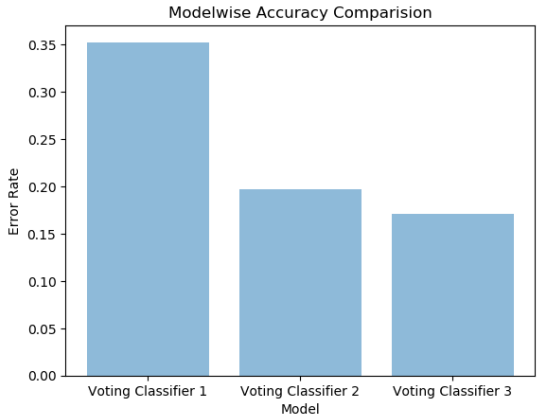

Figure 8. Model Accuracy in Voting Classifiers

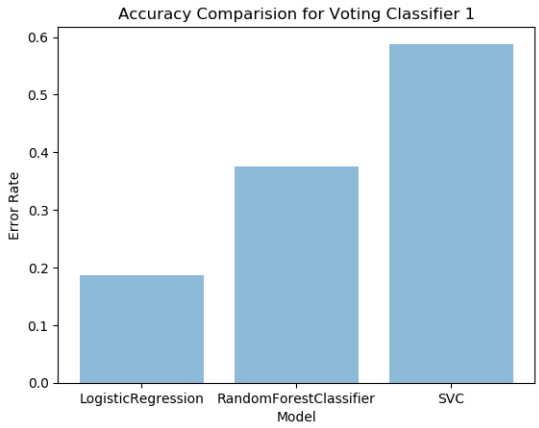

Figure 9. Model Accuracy in Voting Classifier 1

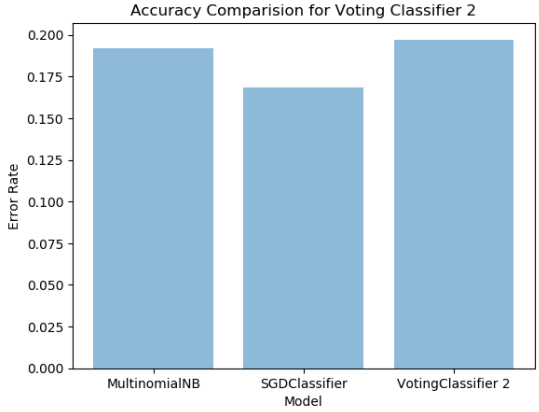

Figure 10. Model Accuracy in Voting Classifier 2

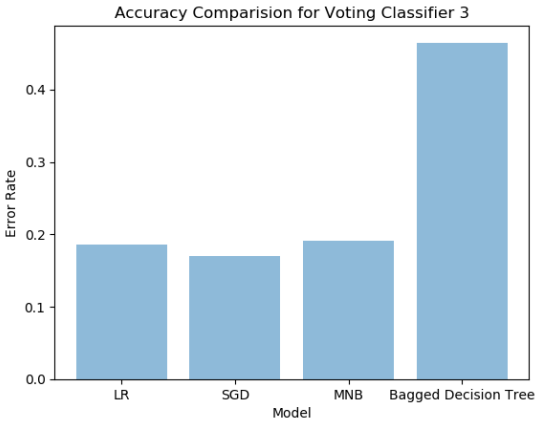

Figure 11. Model Accuracy in Voting Classifier 3

Among the different voting classifiers we have used, Voting Classifier 3 has given us the least error rate, so we use the Voting Classifier 3 and perform cross validation to check the foldwise error rate and average error rate. The results of the cross validation shall be discussed in the **Results** section. BaggedDecisionTree error rate has been plotted to compare it with the group of classifiers in Voting Classifier 3. The group of classifiers in Voting Classifier 3 are : **LinearRegression Classifier, SGD Classifier and MultinomialNaiveBayes Classifier**.

3.3 Tools & Infrastructure

The programs were run on the SICE server **Sharks**. The tables below contain the tools and infrastructure we have used.

Table 1. Table for Infrastructure

Name	Type	Specification
Sharks	Server	32 Cores
Python	Platform	Version 3.6.2
Linux	OS	Red Hat Enterprise
PuTTY	SSH Tool	Version 0.70

Table 2. Table for Packages

Package	Version
Scikit-Learn [6]	0.18.1
Numpy [10]	1.11.3
Pandas [11]	0.21.0
Matplotlib [25]	1.5.1

3.4 Results

The algorithm which has consistently given good scores on different train/test splits and random subsets is the *Multinomial Naive Bayes (MNB)*, it gave us an accuracy of 80-82 % initially. So, a significant amount of time was spent to tune the parameters and pre-process the input text data to obtain a higher accuracy score. After the preprocessing had been done we obtained an accuracy score of 85.75 % for the Multinomial Naive Bayes (MNB) which we believe was it's best. Also the

SVM wtih SGD gave us good accuracy scores in the range of 80-84 %.

From the plots, we observe that all the other models such as the Random Forest, Bagged Decision Tree, SVM perform poorly on the task of author prediction and give us a higher error rate which we believe is due to overfitting on the training data. We have implemented each model with and without cross validation to understand how the model performs. We hoped implementing cross validation would certainly not overfit the model but the error compairision plots indicate that overfitting is still present in our models. The main challenge was in preprocessing the input text data, we have tried a few other methods which gave us accuracy scores below 50%. the best algorithm the MNB was agreed upon after implementing lots of classifiers which performed poorly on the data. The best performer among the group classifiers has been the *Voting Classifier 3*.

The below plots show the results of the cross validation we have performed using the Voting Classifier 3.

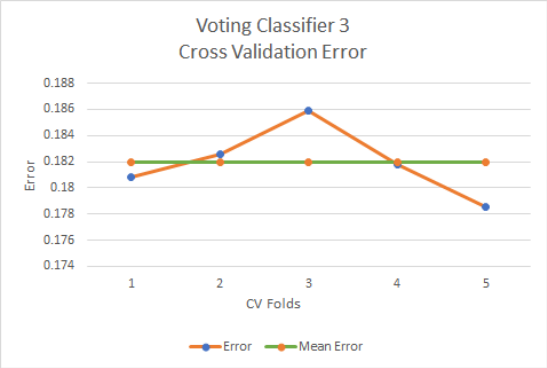

Figure 12. Cross Validation in Voting Classifier 3

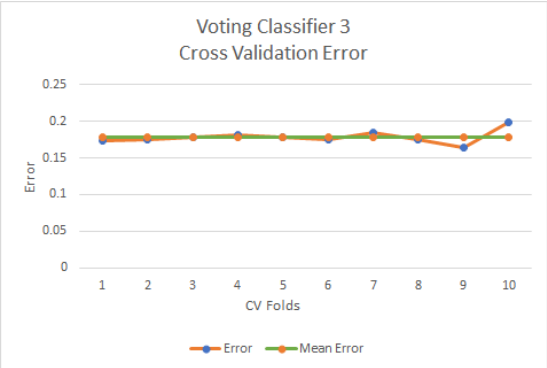

Figure 13. Cross Validation in Voting Classifier 3

The authors consider this project to be a success, but there is always scope to decrease the overall error rate and increase the accuracy of the model we have chosen to use.

3.5 Summary and Future Work

The main objective of the author identification problem is to identify the author of the text. We have used Natural Language Processing (NLP) techniques to preprocess the text data and used different classifiers/algorithms in machine learning to predict the author of the text. The accuracy/error rates of our models have been represented visually in the methods and results sections.

In the future, we would like to try different preprocessing methods or explore the area of preprocessing more, we were unable in implementing/trying all the different preprocessing techniques for NLP successfully in our project this time. The small changes we make in preprocessing can help in reducing the error rate thereby increasing the overall accuracy of our model. Perhaps a deeper understanding of Natural Language Processing might help us achieve higher accuracy in the future. Such understanding might help us fine tune the parameters better of our predictive models.

4. Iceberg: Full Problem Description

As mentioned in the Introduction Section, the data consists of objects represented by two bands of signals along with the shift in the angle of incidence of each object. These objects can be viewed visually as images. The task here is to classify each object as a ship or an iceberg. Since the data depicts radar signal values and not the actual images themselves, the data when visually represented contains a lot of distortions. Also external factors such as winds affect the quality of the image. Low winds generate a darker background whereas high winds generate a brighter background. Such instances have their own applications in the domain of detecting hurricanes by searching for wind patterns but are beyond the scope of this paper. The classification method used in binary. Therefore the predicted output is either 0 or 1 indicating whether the classified object is a ship or an iceberg. The initial data is represented in the following format: the features used for the training data for each of the instances are 75x75 images created by each of the bands; the shift of the bands represented by the angle of incidence and the target feature indicating whether the object is an iceberg or not.

Different transformations on the training data have led to different models (more about it in methods section). For all the approaches a Convolutional Neural Network (CNN) is used as the algorithm detects patterns based on the proximity of pixels that is two pixels close to each other are said to exhibit a stronger relationship when compared to pixels that are far apart from each other.

4.1 Data Analysis

The dataset contains 1604 images each of which is represented by 2 bands of 75x75 dimensions. The training/testing split taken on the dataset is 50/50. That is 802 images shuffled in random order are used for training and the model is tested on the rest 802 images. There are some missing values in the dataset. These are only present in the angle of incidence feature. There are 133 such instances and they are all imputed with 0.

4.1.1 Transformations

The data of each of the bands for each instance is represented by a list containing 5625 values. This is reshaped into a Two Dimensional Array of 75x75 dimensions where each position reflects the pixel value of the image. The values in the 2D array are stored as type float. For humans, it is easier to differentiate the brightness between different RGB bands separately. The same logic is applied to Convolutional Neural Networks as well and hence we create a new band by taking the average of the 2 bands. The bands depict each of the image channels as follows:

- Band 1 = Red Channel
- Band 2 = Green Channel
- Average of Band 1 and Band 2 = Blue Channel

Later in the results section we shall see how this approach of stacking the bands on top of each other compare against considering gray-scale images of each of the bands individually.

The radar signal strength values are in measured in decibel(dB) format and most of the values are negative. The condition of brightness being directly proportional to the signal values still holds good. That is if the pixels denoting the object have values around -9 then the object's surrounding(i.e. the ocean) would have values in the range of -27 and so on indicating that the backscatter signal reflecting from the ocean's surface travels a distance greater than that of the iceberg/ship object.

For easy interpretability and enhancing the difference between the object and it's surroundings, the data is transformed in the following way for every band of all the images:

- Inverse the sign of all the values
- Subtract each pixel value from the maximum value present on the particular row.

After taking the inverse sign of all the values, the surroundings of the object will have higher magnitude values when compared to that of the actual object itself. In the previous paragraph where we considered the object to have signal value of around -9 would now have a value of 9 and that of it's surroundings would be 27. Indicating now, lower the value, greater the distance of signal to travel from the satellite. The second step of the transformation just enhances the differentiation by considering the highest point in the inverse transformation from which the signal can be reflected off ie the signal that has to travel the least distance (which is the maximum value) and subtracting each of the pixel values from this value. This once again transforms the lowest point(which belonged to the object) to be the highest point. In general the object can differentiate itself by a few more values from it's surroundings. The scope of the maximum is confined to the particular row the pixel value is present in. The 3D plots are developed using the plotly[19] library in python.The Before and after 3D image Transformations are as follows:

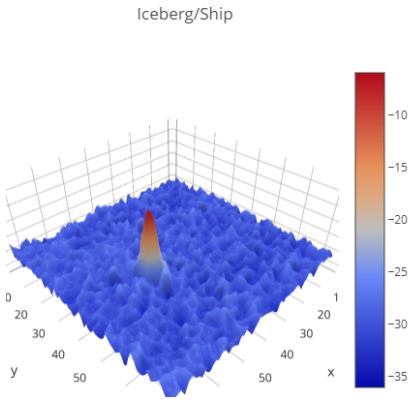

Figure 14. 3D example of a ship before transformation

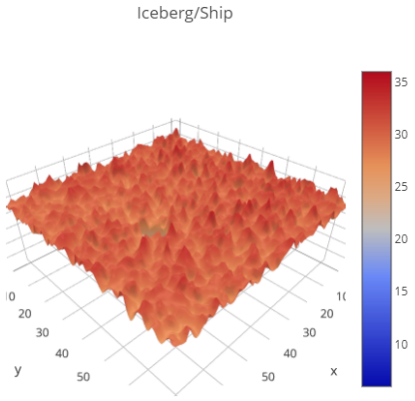

Figure 15. 3D example of a ship after inversing the sign

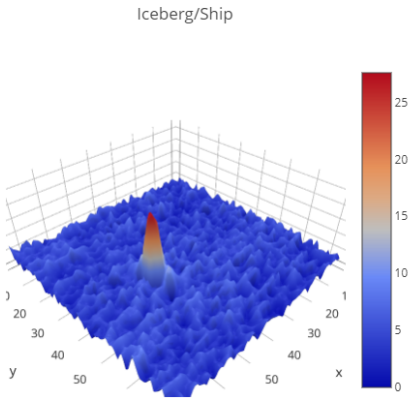

Figure 16. 3D example of a ship after transformation

The band considered over here is the new band formed by taking the average of the two existing bands. The X and

Y axis grids depict the pixel position and the Z value depicts the signal strength at the pixel position. From Figures 1 and 3 it is seen that the original structure is still retained but there are minute differences which can be seen in the color band of the ocean. In Figure 1 the intensity of blue is on average light which indicates a higher value on the color band whereas the intensity of blue in the After image is darker which indicates a lower value of the surrounding region on the color band without change in the highest point of the image. Figure 2 shows that after inversing the sign, the object is now a depression in the middle overshadowed in the image by it's surroundings. Then we normalize the data. Normalization is done by just dividing the each pixel value by the square root of the sum of squares of values in the row ie L2 normalization. The equation is given as follows:
Normalized pixel value =

$$x/\sqrt{\sum_{i=1}^{n} y_i^2}$$

where x represents the particular pixel value.
y_i represents the every pixel value present in the same row as that of x.

For some of the models, the L1 normalization is considered. The formula is given as follows:
Normalized pixel value =

$$x/\sum_{i=1}^{n} y_i$$

where x represents the particular pixel value.
y_i represents the every pixel value present in the same row as that of x.

This concludes the initial transformation of the image which is common to all the methods that are described in the methods section.

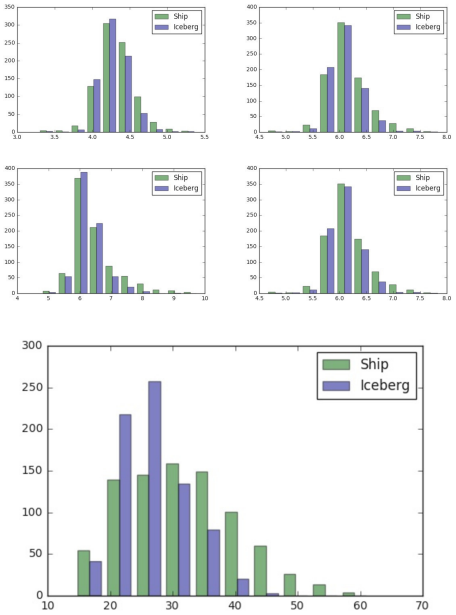

Figure 17. Summary Statistics of created Band 1: From Left to Right- 25 percentile,75 percentile,mean,median and max(the last plot)

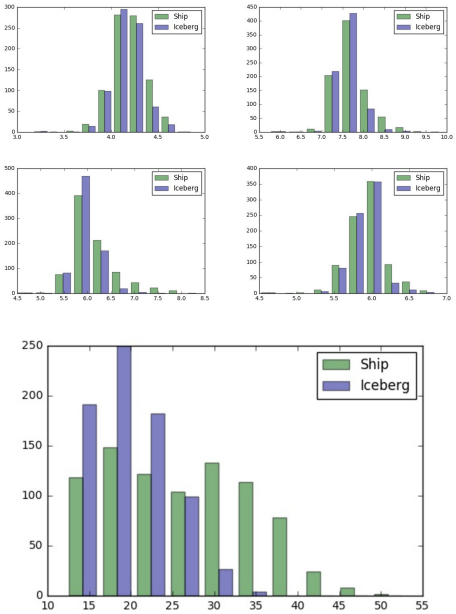

Figure 18. Summary Statistics of created Band 2: From Left to Right- 25 percentile,75 percentile,mean,median and max(the last plot)

4.1.2 Summary Statistics

Summary Statistics is taken just before normalizing the data. The summary statistics of the bands portray the 25 percentile, 50 percentile , 75 percentile , mean, median and max values of each of the transformed bands.

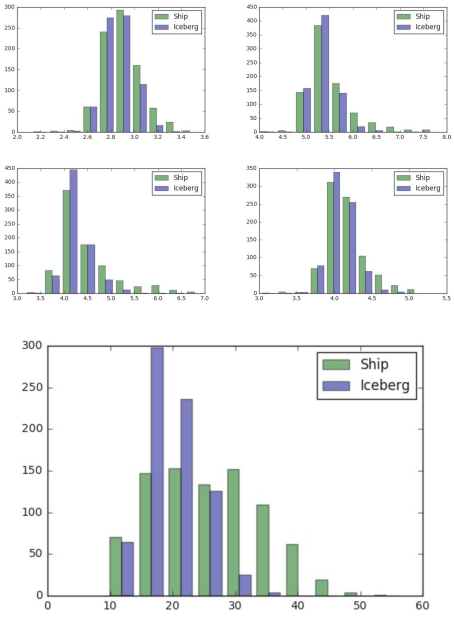

Figure 19. Summary Statistics of created new band: From Left to Right- 25 percentile,75 percentile,mean,median and max(the last plot)

From the above summary statistics it is clear that the distribution of ship's pixel's values are more spread out. That is, the icebergs values are higher and have a lot more peaks when compared to the ships which makes sense. This is most effectively portrayed in the maximum summary statistic of each band.

4.2 Methods

The flowchart of the models is showin in Figure 20:

- Benchmark Model
  This model consists of running a Convolutional Neural Network over all the three bands separately and then deciding which is the best band to perform furthur transformations on.

  Convolutional Neural Network Summary[20] : A Convolutional Neural Network detects certain patterns in an image which helps the neural network to classify the image as a particular class. This is achieved by detecting simple patterns in stages and then combining them to form more complex patterns that resemble the object being classified in successive layers. In the final layers layers when the template of each object is created from the respective image, the weighted sum is taken and the prediction is made. A Convolutional Neural Network consists of 3 layers:

  - Convolutional Layer
  - Pooling Layer
  - Fully Connected/Dense Layer

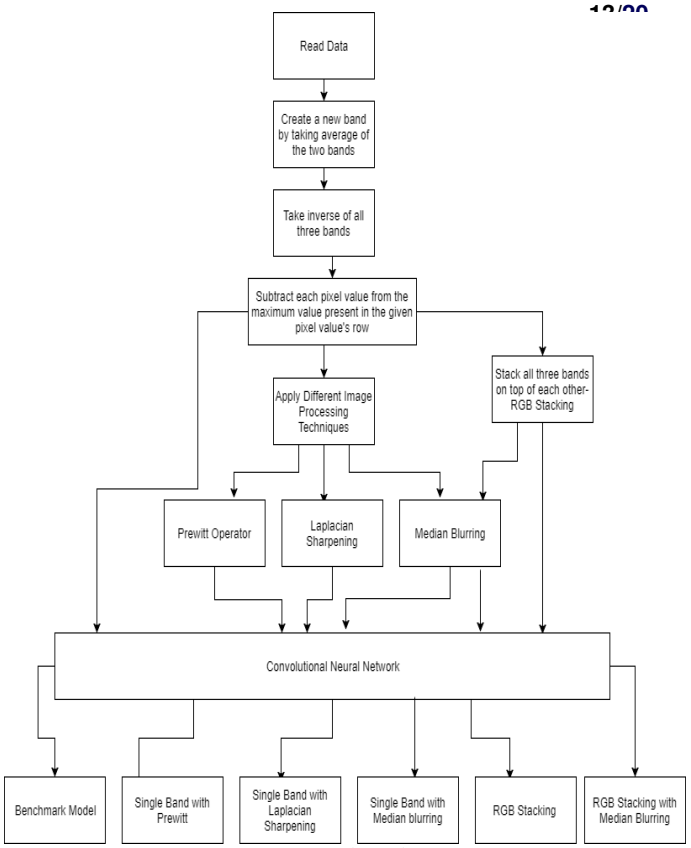

Figure 20. Flowchart

In the convolutional layer, the input is an image of certain dimensions [2]. A convolutional layer is formed by running a kernel of certain dimensions over it. The size of the kernel must always be smaller than the size of the image. Typically kernels of area 3x3, 5x5, 7x7 etc are chosen. The depth dimension of the kernel is always equal to the depth of the image. So for a color image the size of the kernel would be 3x3x3 -over here the last dimension indicates the depth. Initially the kernel is assigned random weights which get updated during the back-propogation step. We now superimpose this kernel over the left top corner of the image and the dot product of the two matrices(ie sum of element wise multiplication between the two matrices). Once we are done with the multiplication, the kernel then slides towards the right by a particular value. This value is known as stride. Doing so creates a set of new matrices known as feature maps. The number of feature maps will be equal to the number of kernels used. Each kernel will have it's own set of weights trying to detect different patterns in the image. After all the pixel values in the initial image is covered by the kernel the resulting size of the feature map(output produced by the convolutional layer) will always be lesser than that

[2] if the number of dimensions is two then it is a grayscale image, if the number of dimensions is three then it is a color image consisting of three channels-RGB

of the original image size. Assuming that your image has equal number of rows and columns and the stride value is 1, when a kernel of NxN dimensions reaches the end of the image it would have produced a feature map of size:

(number of rows in image - N + 1)*(number of columns in image - N + 1).

If you would like to maintain the size of the image throughout the layers, then the feature maps are zero padded. Meaning to the difference in the border sizes between the feature map and the original image, pixel values of 0 are padded. This concludes the end of a single convolutional layer.

Once the image passes the convolutional layer, a non linear activation function is applied. This is because if only linear operations are performed a single perceptron with change in weights can perform the task. But in convolutional neural networks researchers have found out that the Rectified Linear Units(ReLu) is far better as it is able to train the network quickly without resulting in a significant decrease in accuracy[21]. The function is as follows:

$$f(x) = max(0,x) \tag{1}$$

Hence there are no negative activation's after the ReLu function is applied.

Moving on to the Pooling layer, in this layer a kernel of a particular size is applied on the image and summary statistics of that particular area is calculated. The most commonly used pooling function is the maxPooling where the maximum of the kernel size area superimposed on the image is taken. Typically a 2x2 kernel size is used. An image passes through multiple Convolutional and Pooling layers before it gets to the Fully Connected/Dense Layer.

In the Fully Connected Layer just as the name suggests, every feature of the previous layer is connected to every feature in the Fully Connected Layer. The output of a Fully Connected Layer(considering the softmax approach is applied) is just a vector of probabilities of the image belonging to each class. The sum of this vector is always one.

To the applied transformations discussed in section 4.1.1 we now run a simple Convolution Neural Network. We have used the keras[22] library in python which uses tensorflow or theano APIs as the backend to implement the Network. Model Architecture of simple neural network is as follows :

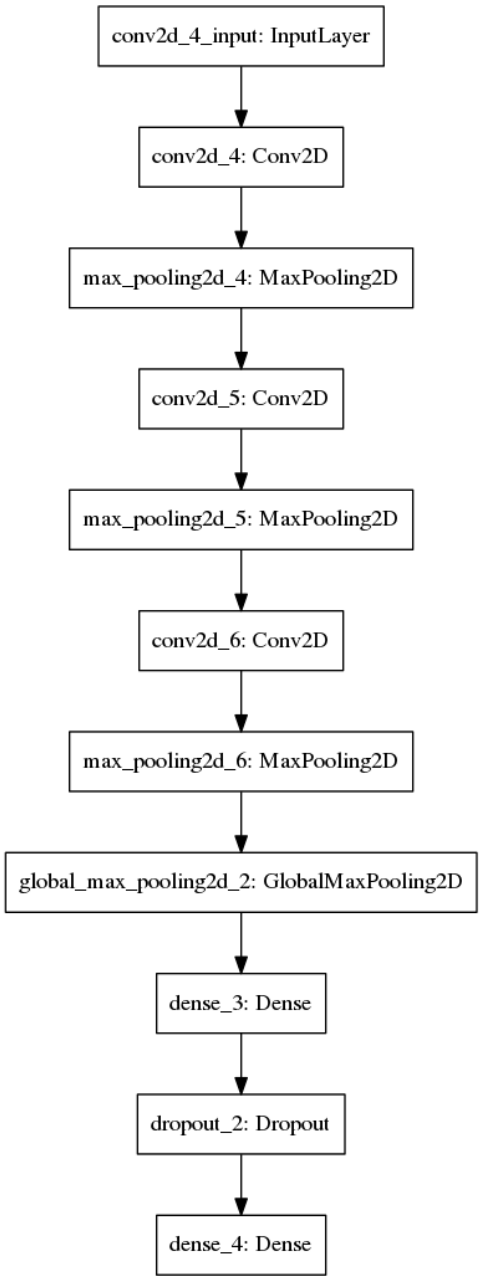

Figure 21. Benchmark model Architecture

Parameter Values:
Type of Model: Sequential
First Convolutional Layer -Layer conv2d4:

- Number of Filters : 32
- Kernel Size : 3x3
- Activation Function : "ReLu";Rectified Linear Units
- Input Dimension of single image : 75x75x1
- Padding: "Same" ;ie retains the same size

First Max Pooling Layer-maxpooling2d4:

– max pooling of kernel pool size (2,2)

Second Convolutional Layer-Layer conv2d5:

– Number of Filters : 32

– Kernel Size : 3x3

Second Max Pooling Layer-maxpooling2d5:

– max pooling of kernel pool size (2,2)

Third Convolutional Layer-Layer conv2d6:

– Number of Filters : 32

– Kernel Size : 3x3

Global Max Pooling Layer-globalmaxpooling2d2 First Dense Layer:dense3:

– Number of Units :256

– Activation : ”ReLu”; Rectified Linear Units

Dropout Layer-dropout2 :

– rate:0.5

Second Dense Layer-dense4:

– Number of units:2

– Activation : ”Softmax”

Compile with the following parameters:

– loss=’binarycrossentropy’,

– optimizer=’adam’,

– metrics=[’accuracy’]

Fit the model with the following parameters:

– batchsize=32

We shall now take a look at some of the transformed images. These are gray scale images but we use the ”hot” spectrum (which basically maps the values onto a single color) parameter present in matplotlib[23] package.

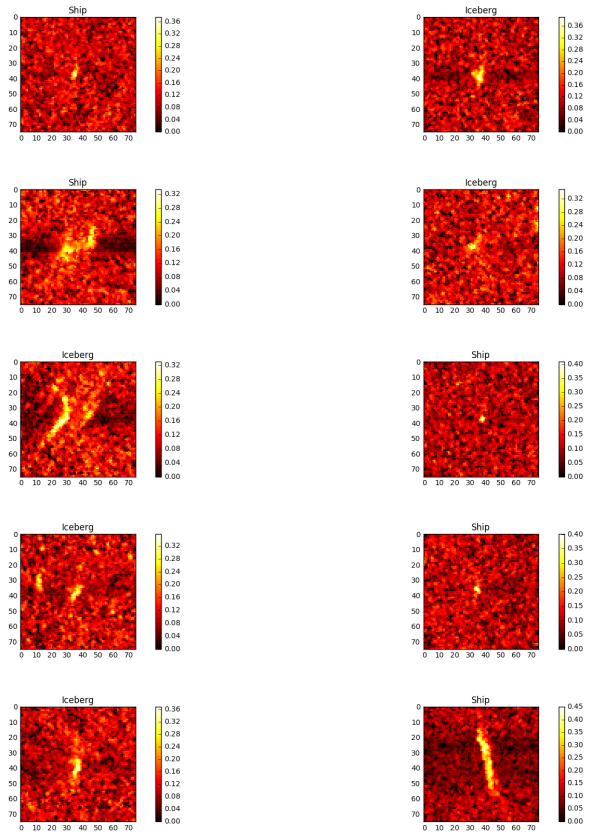

Figure 22. Pictures of images selected randomly after the transformation.

The Convolutional Neural Network was run separately for all three bands. The new band created from the average gave the best results. Hence rest of the models are tested out with this band.

- Model 2 Prewitt Operator[24]:
  In the upcoming part of this section in Model 3 we have taken the Laplacian Sharpening transformation which is a second order derivative(more on second order and first order derivates discussed in the next model). In this model we consider a First Order Derivative for sharpening. In general, the first order derivative operators are very sensitive to noise and produce thicker edges. There are plenty of First Order Derivatives, but in this approach the Prewitt Operator is used. The operator calculates the gradient of the image intensity at each point. The image gradient is directional change in intensity and color of the image. In simple terms this is the change in the pixel’s values to it’s surroundings from high to low or vice versa. This gives the direction of the largest possible increase from light to dark and the rate of change in that direction. The result therefore shows how abruptly or smoothly the image changes at that point, and therefore how likely it is that part of the image represents an edge, as well as the edge’s

orientation. The operator has 2 filters in vertical and horizontal directions. We have considered the vertical filter for this approach which is given as follows:

$$\begin{bmatrix} -1 & 0 & 1 \\ -1 & 1 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

Scipy[25] package in python used for implementation.

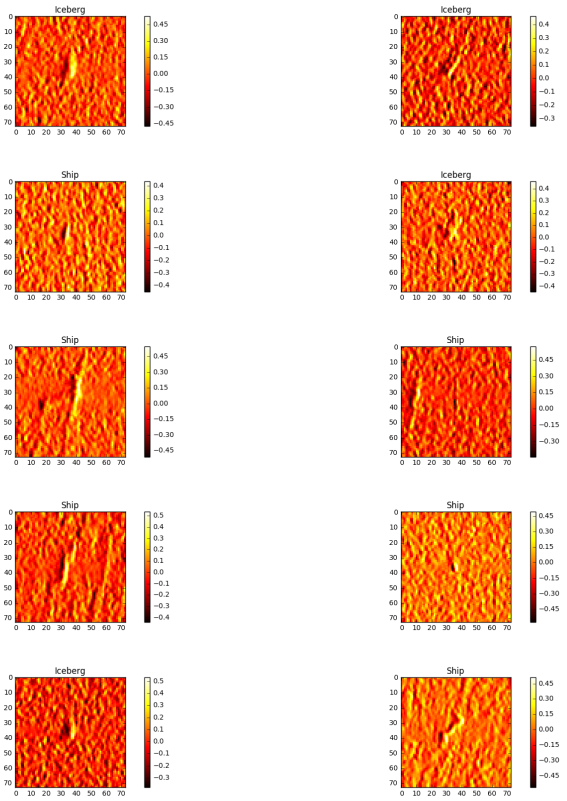

Figure 23. Randomly selected Transformed Images using the Prewitt Operator

- Model 3: Laplacian Sharpening[26]
  This model consists of an image processing techinque known as Laplacian Sharpening which is applied to the already transformed data. The key of using sharpening techniques is to differentiate the edges from it's surroundings(spatial differentiation). Typically used in edge detection.
  An image is a discrete function therefore the traditional definition of a derivative cannot be applied. Hence, a suitable operator have to be defined such that it satisfies the main properties of the first derivative:
  - it is equal to zero in the regions where the intensity is constant
  - it is different from zero for an intensity transition
  - it is constant on ramps where the intensity transition is constant

  The natural derivative operator is the difference between the intensity of neighboring pixels (spatial differentiation). The derivative along the X axis would be as follows:

  $\partial f/\partial x = f(x+1) - f(x)$

  Similarly, the second derivative along the X axis can be defined as:

  $\partial f/\partial x = \mathrm{f}\,(\mathrm{x}+1) - \mathrm{f}\,(\mathrm{x}) - (\mathrm{f}\,(\mathrm{x}) - \mathrm{f}\,(\mathrm{x}-1))$
  $= \mathrm{f}\,(\mathrm{x}+1) - 2\mathrm{f}\,(\mathrm{x}) + f(x-1)$

  This operator satisfies the following properties:
  - it is equal to zero where the intensity is constant
  - it is different from zero at the begin of a step (or a ramp) of the intensity
  - it is equal to zero on the constant slope ramps

  Usually sharpening filters make use of second order operators as it is more sensitive to variations that first order operators. One such filtering technique is using the Laplacian. The intuition behind going along with this technique is to increase the sharpness of edges so that the network would be able to figure out patterns in the edges of the objects. This is implemented by just running the following kernel over each of the images:

0	1	0
1	−4	1
0	1	0

Figure 24. Laplacian Kernel Used.

  Implementation done using scipy package in python. To this transformation the same Convolutional Neural Network architecture that is described in the Benchmark Model is applied.

- Model 4: Median Blurring[27]
  The concept of Median Blurring is obtained from a wide range of Smoothing techniques. Smoothing is to create an approximation function that captures the important patterns in images while discarding noise. The intuition

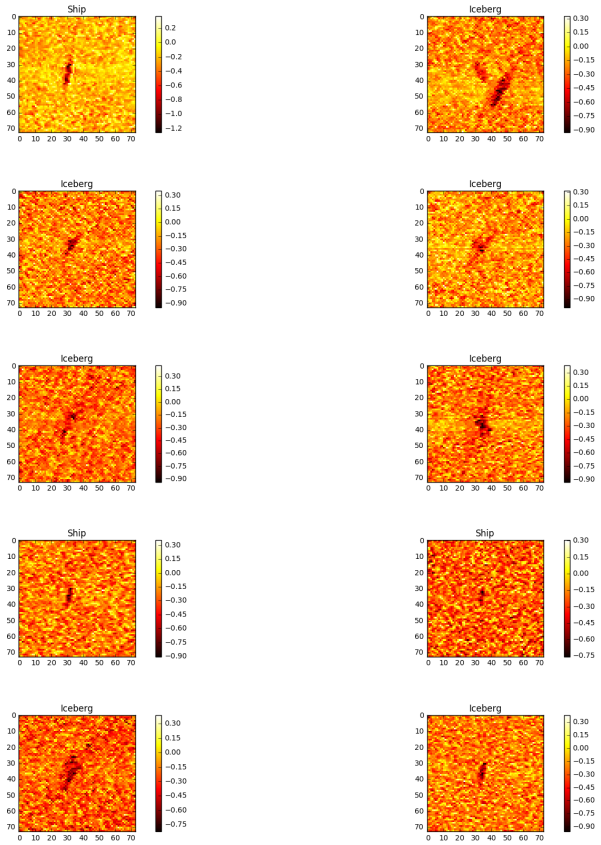

Figure 25. Randomly selected images after Laplacian Sharpening.

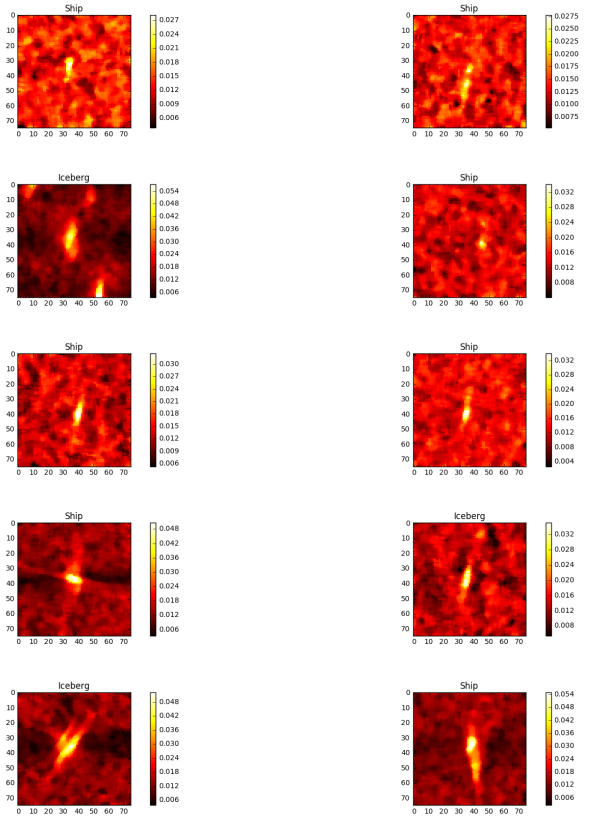

Figure 26. Median Blurred images.

of applying this technique was that we noticed a lot of noise present in the images(the object's surroundings) and to reduce the intensity of the noise we decided to apply a blurring filter. The blurring filter used in this approach is the median filter. It works by mapping a kernel of a particular size(say 5x5) onto the image and then the median value is picked to represent the the particular cell size in the image. It should be noted that blurring removes the noise present in the image by decreasing the sharpness. But this was just a trade off we decided to go with.

The transformed images can be seen in Figure 25.

To this above transformation, the same Convolutional Neural Network Architecture as described earlier under the Benchmark model part of this section is applied.

- Model 5 RGB Stacking We would like to see whether the network would detect different patterns when color was attributed to the images. Therefore in order to represent our input image as a color image, the three bands are stacked on top of each other.

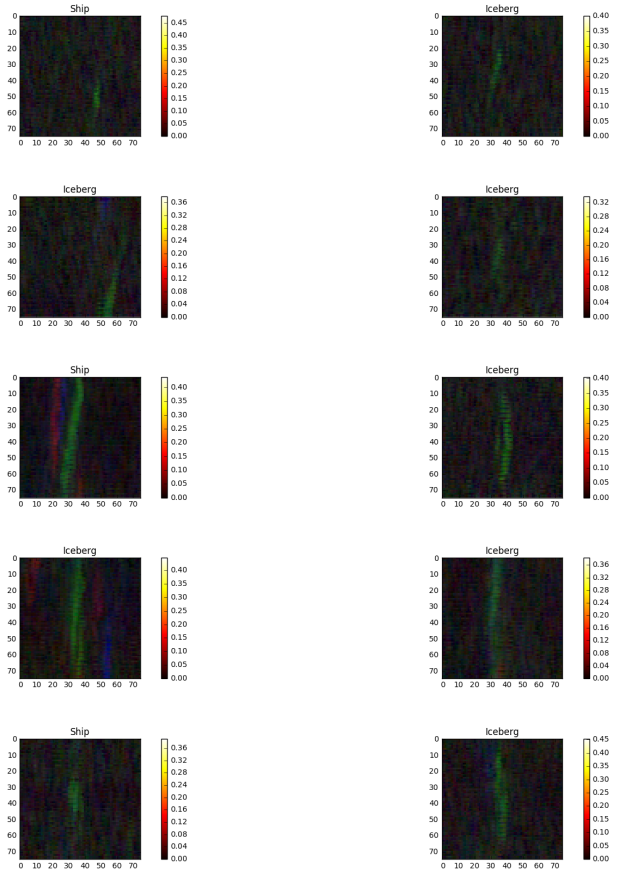

Figure 27. RGB Stacked images

The input dimensions of the image in the First Convolutional Layer changes to 3x75x75.

- Model 6 RGB Stacking with Median Blur There is a slight change in the type of normalization. Instead of going ahead with L2 normalization, this model uses the L1 normalization.

  To the previous model, the concept of Median Blurring is extended. As described in the results section, the concept of median blurring yields better result than the benchmark model when applied to the new band. We decided to try out the median blurring approach on all three bands before we stack them on top of each other.

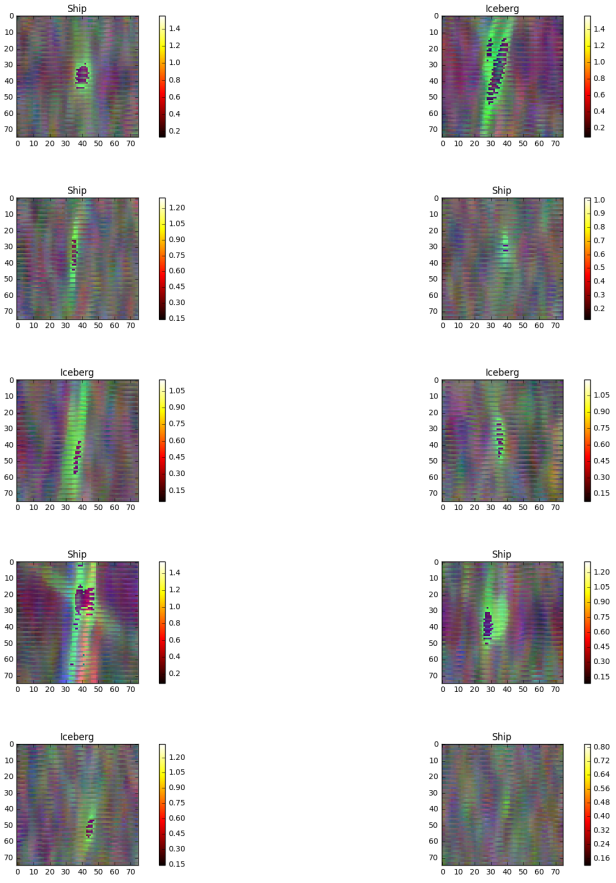

Figure 28. RGB Stacked images after Median Blur

The above images are multiplied by a scale of 30 for visualization. Without the multiplication, the images would not bring out the different color schemes as their values were too low and the end result was a pitch dark image.

The input dimensions of the image in the First Convolutional Layer changes to 3x75x75.

4.3 Tools and Infrastructure

Table 3. Table for Infrastructure

Name	Type	Specification
Sharks	Server	32 Cores
Python	Platform	Version 2.7.13.final.0
Anaconda	Python Distribution Platform	4.3.30
Jupyter Notebook (Part of Anaconda)	Interactive Web Framework	1.0.0
Linux	OS	Red Hat Enterprise
PuTTY	SSH Tool	Version 0.70

Table 4. Table for Packages

Package	Version
json	2.4.0
numpy	1.11.3
pandas	0.21.0
plotly	2.2.2
matplotlib	1.5.1
scikit-learn	0.18.1
keras	2.0.8
opencv	3.1.0
scipy	0.19.0

4.4 Results

We first run the Convolutional Neural Network on all the three transformed bands separately and then select the band which gives the best testing accuracy to be our benchmark model. The table is as follows:

Table 5. Testing Accuracies of the three bands

Band Name	Testing Accuracy
Band 1	79.86%
Band 2	82.17%
Average of Band 1 and Band 2	83.54%

Now we select the new formed band of averages to be the data used for our benchmark model. This transformed band is the data supplied to the Laplacian Sharpening,Prewitt Operator and Median Blurring models as well.

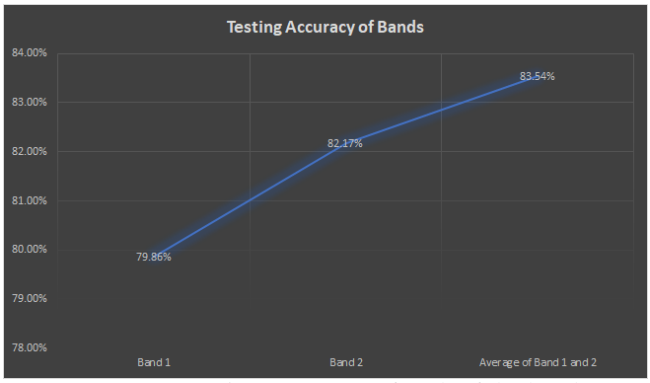

Figure 29. Testing Accuracy of each of the bands

Over a certain set of iterations, the training and testing accuracies of the best results are recorded. Usually towards the end of the iterations, the model starts overfitting and the testing accuracies decreases a lot. The Training and Testing accuracies of each of the models are as follows:

Table 6. Training and Testing Accuracy

Model	Training	Testing	Iterations
Benchmark	96.88%	83.54%	100
Laplacian Sharpening	92.64%	83.67%	100
Prewitt Operator	87.91%	80.55%	100
Median Blurring	88.65%	85.41%	300
RGB Stacking	89.90%%	88.53%	100
RGB Stacking with Median Blurring	92.77%	89.4%	100

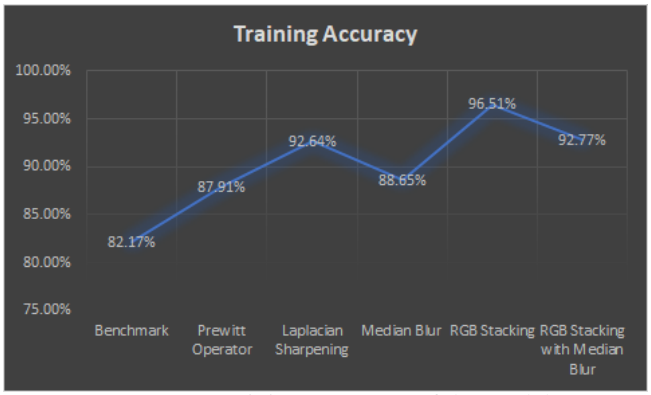

Figure 30. Training Accuracy of the models

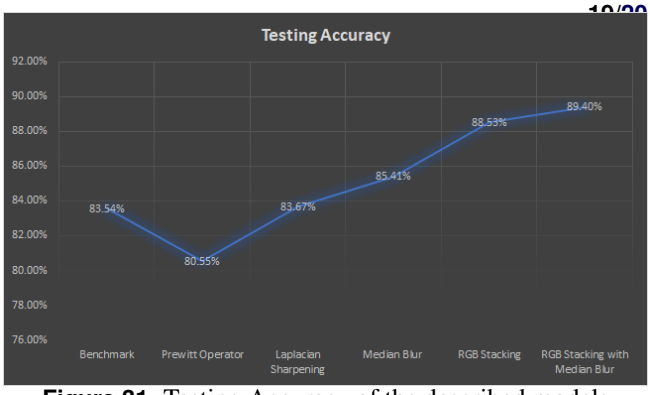

Figure 31. Testing Accuracy of the described models

From the results we can say that the majority of our models were better than the Benchmark model that we had created. Three image transformation techniques were used on a single band out which the only technique that seemed to differentiate itself from the Benchmark model was Median Blurring. Taking the First Order X-Derivative(Prewitt Operator) looked good visually where the object in the image can be seen as a 3 Dimensional object but the results were not better than that of the benchmark. We expected the Second Order X-Derivative(Laplacian Sharpening) to do better than the First Order Derivative as described in the Models Section. This was true but the result was similar to that of the benchmark model. As mentioned above, the Median Blurring approach was better. From this we can conclude that our data is more suited for smoothing techniques rather than sharpening. For the next two models, all the three bands are considered and the results are much better when compared to the single band transformation models. As suspected, the neural network detects certain patterns when the image is represented on a color scale than when represented on grayscale. Combining our two best approaches-Median Blurring and RGB Stacking led to the creation of our last and best model which is about 6 percent away from the benchmark.So the results produced by this project are successful.

4.5 Future Work

Exploring image thresholding(canny edge detection) in detail, histogram of gradience and see if wavelet transformation can be applied to this project.
Also, we have used a simple convolutional neural network. There are many ways in which the network could be tuned by adding more layers and accordingly adjusting the parameters. Our next step would be to try out approaches such as transfer learning by using pre trained models such as VGGNet16, ResNet50 etc.

Acknowledgments

The authors wish to thank Prof Dr. Mehmet Dalkilic for his feedback and suggestions. We would also like to thank Marcin Malec (AI Data Mining) for his help. The authors would like

to thank Kaggle for providing us with the challenge and the data.

References

[1] Top 10 Algorithms in Data Mining
http://www.realtechsupport.org/UB/CM/algorithms/Wu_10Algorithms_2008.pdf

[2] Natural Language Processing:
https://en.wikipedia.org/wiki/Natural_language_processing

[3] Natural Language Toolkit:
http://www.nltk.org/

[4] Count Vectorizer:
http://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html

[5] Preprocessing using NLP:
https://nlp.stanford.edu/IR-book/html/htmledition/dropping-common-terms-stop-words-1.html

[6] Scikit Learn Documentation:
http://scikit-learn.org/stable/documentation.html

[7] Pipeline:
http://scikit-learn.org/stable/modules/generated/sklearn.pipeline.Pipeline.html

[8] Naive Bayes:
https://en.wikipedia.org/wiki/Naive_Bayes_classifier

[9] Multinomial Naive Bayes:
http://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html

[10] NumPy Documentation:
http://www.numpy.org/

[11] Pandas Documentation:
https://pandas.pydata.org/

[12] Logistic Regression:
http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

[13] Support Vector Machines (SVM):
http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

[14] SVM with Stochastic Gradient Descent (SGD):
http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDClassifier.html

[15] Random Forest Classifier:
http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

[16] Decision Tree with Bagging:
http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.BaggingClassifier.html

[17] Voting Classifier:
http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.VotingClassifier.html

[18] Author Identification using Deep Learning:
https://web.stanford.edu/class/cs224n/reports/2760185.pdf

[19] Plotly Documentation
https://plot.ly/python/

[20] ImageNet Classification with Deep Convolutional Neural Networks:
https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolut
pdf

[21] Rectified Linear Units Improve Restriced Boltzmann Machines
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.165.6419&rep=rep1&type=pdf

[22] Keras Documentation
https://keras.io/

[23] Matplotlib Documentation
https://matplotlib.org/1.5.1/contents.html

[24] Prewitt Operator
https://en.wikipedia.org/wiki/Prewitt_operator

[25] Scipy Documentation
https://docs.scipy.org/doc/scipy/reference/release.0.19.0.html

[26] Laplacian Sharpening
https://bohr.wlu.ca/hfan/cp467/12/notes/cp467_12_lecture6_sharpening.pdf

[27] Open CV Documentation
https://docs.opencv.org/2.4/modules/imgproc/doc/filtering.html?highlight=medianblur#medianblur

[28] The Book
Hands-On Machine Learning with Scikit-Learn and TensorFlow