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

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

Hankyu Jang[1+*], Sunwoo Kim[2+*], Tony Lam[3+*]

Executive Summary The Kaggle competition we are competing in is the Author Identification and Iceberge Classification Challenge. The Author Identification challenge requires users to retrieve excerpts of text by authors, train a predictive model given the data, and be able to classify new text excerpts to the correct belonging authors. It is a supervised learning datamining problem in the natural language processing domain. The solution is achieved with a neural network approach called the Word2Vec model with the skip-gram approach on a fourgram excerpt dataset, which achieved testing results of approximately 80 percent using 5-fold cross-validation.
The Iceberge Classification Challenge requires to classify an image whether it is an iceberg or a ship. It is also a supervised learning datamining problem. Among experiments with different classifiers, the Random Forest model shows 86 percent accuracy when trained and tested on the training set using 10-fold cross-validation. CNN was able to achieve 90 percent validation accuracy when trained and tested on 10-fold cross-validation.

Keywords
Word2Vec — CNN — Random Forests

[1] *Data Science, School of Informatics, Computing, and Engineering, Indiana University, Bloomington, IN, USA*
[2] *Computer Science, School of Informatics, Computing, and Engineering, Indiana University, Bloomington, IN, USA*
[3] *Informatics, School of Informatics, Computing, and Engineering, Indiana University, Bloomington, IN, USA*
[+] *These authors contributed equally to this work*
[*] **Corresponding authors**: hankjang@indiana.edu, kimsunw@indiana.edu, tjlam@indiana.edu

Contents

1 Introduction **1**
1.1 Author Identification . . . 2
1.2 Statoi/C-CORE Iceberg Classifier Challenge . . . 2
2 Datamining **2**
What is Data Mining • What Does Data Mining Yield • General Steps of Data Mining • Clustering vs. Classification • Loss Function
2.1 Data Preprocessing . . . 3
2.2 Mining, Interpretation, and Action . . . 4
3 Author Idenfication: Full Problem Description **6**
3.1 Data Analysis . . . 6
Statistics
3.2 Methods . . . 7
Algorithm • Background • Challenges • Softwares • Hardwares • Input Structure • Presentation
3.3 Results . . . 8
3.4 Summary and Future Work . . . 9
4 Iceberg: Full Problem Description **9**
4.1 Data Analysis . . . 10
4.2 Data Pre-processing . . . 10
4.3 Methods . . . 11
Algorithm • Background • Softwares • Hardwares • Structure • Convolutional Neural Networks Implementation
4.4 Results . . . 13
Convolutional Neural Network Results • Challenges
4.5 Summary and Future Work . . . 15
References **15**
Appendices **17**
A Appendix Appendix A: Dropout Experimentations for Author Identification **17**
B Appendix Appendix B: Models for Statoil/C-CORE Iceberg Classifier Challenge **19**

1. Introduction

In the age of information, data mining has become a driver of intellectual advancement and discovery. Data mining, the process of discovering patterns to be used to build predictive models, has become widespread across many disciplines ranging from finance to health-care[1]. This concept of learning from data is not a new concept but may arguably date back to the works of Turing and his paper of computable numbers [2], or even further to the works of Bayes and the development of Bayes' theorem [3]. Given the explosion of data and the advancement of computational tools and algorithms, data mining has seen a surge of emergence in a wide range of applications and used to draw classifications in data types such as text and images.

In the following paper, we will discuss our participation in two competitions hosted by Kaggle: Spooky Author Identification[4], and Statoil/C-Core Iceberg Classifier Challenge[5]. Kaggle is a platform used by various companies and organizations to host data science competitions which uses community participation to solve data-oriented problems[6].

The Spooky Author Identification Competition is a competition initiated by Kaggle with the goal of using the provided collection of labeled training text to build a model used to classify unlabeled text to their corresponding authors. In this competition, the collection of texts were obtained from a variety of novels written by spooky authors: Edgar Allan Poe, HP Lovecraft, and Mary Shelley. [something about data provided] The competition can be found at the following webpage: https://www.kaggle.com/c/spooky-author-identification.

The Iceberg Classifier Challenge is a competition initiated by Statoil, an international energy company, and aims to utilize C-CORE satellite data to correctly classify maritime objects between icebergs and ships. The purpose of this competition is to develop a machine learning approach to accurately classify and discriminate against icebergs to improve nautical operations. The contributions of this competition will be used to improve existing infrastructure utilized to protecting the lives of maritime workers as well as improve the cost of safety operations. [something about data provided] The competition can be found at the following webpage: https://www.kaggle.com/c/statoil-iceberg-classifier-challenge.

1.1 Author Identification

The problem to be solved is to predict the author to which a given excerpt belongs to. The data given consists of thousands of excerpts $S_{a_1}, S_{a_2}, S_{a_3}, \cdot$, and the corresponding author label defined as $A = \{a_1, a_2, a_3, \cdot\}$.

The excerpt data is given as such,

> 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*. To build a model from this input, the data is preprocessed and split into four types of formats: bigrams, trigrams, fourgrams, and word tokens.

The goodness of the prediction can be quantified by the number of correct predictions over the total number of data samples. The prediction output is given as the probability to which the excerpt belongs to each of the authors as such, $f(A|t,S) = \{p_{a_1}, p_{a_2}, p_{a_3}\}$. Since it is a strict categorical classification problem, the loss can be quantified as the categorical cross entropy.

1.2 Statoi/C-CORE Iceberg Classifier Challenge

In some areas, such as offshore of the East Coast of Canada, drifting icebergs are threats because they hamper navigation and activities in these areas. It's especially difficult to monitor the movements of these icebergs due to the harsh weather in these remote areas. However, there's an option to monitor them regardless of the harsh weather: satellite.

Statoil[1] has worked with C-CORE[2], and provided satellite data that could be used to train models that can classify icebergs from ships. In the field of data mining, this problem is a binary classification problem. The features are flattened image of data, which consists of 5625 elements per band, and the incidence angle of the image. Training data has a label of `is_iceberg` which is 1 if iceberg, or 0 otherwise.

We can apply classification algorithms to the dataset. Before that, we need to process the data set. There are `na`'s in the feature `is_iceberg`, hence we need to handle missing values. Due to the high-dimensionality of the features, we can reduce the dimension by applying dimension reduction algorithms such as PCA. We may also have to scale the features and compile the two bands together in use for CNN.

2. Datamining

2.0.1 What is Data Mining

Data Mining is a process of iterating over (usually large amounts of) data to (1) answer questions, (2) explore and discover relationships, (3) annotate data. Data Mining is often a method which can help provide supporting information but is not directly used to solve problems.

2.0.2 What Does Data Mining Yield

In regards to what data mining does, data mining can yield answers to questions in a sense that it provides the necessary information which will allow sensible judgment if the data is consistent, inconsistent, related or unrelated to the problem statement. Additionally, data mining may yield correlations, anti-correlations, or lack thereof, suggest relationships. Data mining is also a useful tool to annotate and make sense of data through methods such as classification.

[1]Statoil is a Norwegian-based energy company with operations in more than 30 countries.

[2]C-CORE is a Canadian research and development (R&D) corporation that creates value in the private and public sectors by undertaking applied research and development, generating knowledge, developing technology solutions and driving innovation.

2.0.3 General Steps of Data Mining

There are nine general steps to data mining. The first step, arguably the most important step, is a construction of the problem statement. The problem statement identifies the purpose or goal in what the data miner is trying to achieve. From there we can begin to understand the question and its necessities. The second step, we then need to obtain the data. Data acquisition is often difficult as we must find data which may be informative to answer our problem statement. Additionally often the data you wish to receive is not readily accessible for a variety of reasons such as privacy, proprietary, and data is valuable.

Once the data is obtained, it must go through various stages of pre-processing steps. The third step is data enrichment, which usually entails enhancing the raw data in some way. Also, the fourth step is data cleaning, where non-sensible, missing, or irrelevant data may be removed. The fifth step of integration, which often may be necessary if data is collected or sample from multiple sources. Given multiple sets of data, integration is necessary to join the data sets together. Depending on the model used, the sixth step of transformation is often necessary. The transformation stage can be regarded as modifying the data such as normalization or transforming into categorical data.

After the data has been preprocessed, it can be applied to the models which will help extrapolate correlations and make sense of the data. This step of the process can be considered the mining stage. Often our models do not directly tell us results, and we need to make sense of the model outputs, this stage is a validation and interpretation stage. Lastly, once we have constructed our model, we must apply the model and utilize its results.

2.0.4 Clustering vs. Classification

If we don't have a prior knowledge of where the classes belong to, in other words, if samples are unlabelled then this problem is a clustering problem. Famous clustering algorithms are K-means and Expectation Maximization. Because we have no prior knowledge of how many clusters there are, we need to try clustering the data points with different values of k, where k stands for the number of clusters.

If the samples are labeled (discrete), we consider this problem as a classification problem. We construct a model using the existing data set, then classify the unlabeled data points following the model. Famous classification algorithms are Decision Trees and Support Vector Machines. Decision boundary could be either linear or very complex if kernel tricks are used to transform the data to a higher dimensional space.

2.0.5 Loss Function

In the Author Identification and Iceberg Classification competition, as well as many others, Kaggle uses a Logarithmic Loss function (log-loss) as the metric to evaluate competition submissions. The goal for classifiers is to minimize the loss function, which compares the prediction to the ground truth. The minimization of the loss function is effectively the maximization of the classification accuracy. Kaggle defines their log-loss function [7] as follows:

$$logloss = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{i,j} log(p_{i,j}) \tag{1}$$

Where N is the number of observations, M is the number of class labels, log is natural log, where y = 1 if i=j and 0 otherwise, and p is the predicted probability that i is in class j.

The log-loss function classifies the scores in a degrading logarithmic function, where the log-loss function provides large penalty scores for classifiers which incorrectly classify. Because submission is not just a binary answer of yes or no and requires a probability of assigning to available labels, it would be to the advantage of the algorithm to be partially right versus entirely wrong as the log loss function exponentially increases the penalty for wrong classifications.

2.1 Data Preprocessing

- What are the steps?
  In preprocessing, we need to check about missing values. If there are a small number of missing values in a large amount of data, we can simply get rid of the rows entries with missing values. If missing values are condensed in some feature vectors, then we can remove those features. However, there are times where we need to fill in these missing values. A simple method is to take the mean or median value of the corresponding feature vector. However, handling missing values, we should experiment them with many different results of the preprocessed data sets.
  We may need to change the representation of certain feature from continuous variable to a discrete or a logical value before using some learning techniques, such as decision tree. If this is the case, we can automatically bin the data sets with some logic, but many times we need to draw histograms at different intervals.
  If data is high-dimension, meaning that there are more features compared to entries, we may have to reduce the dimension of the dataset. There are two methods in reducing the dimensions: (1) feature selection and (2) latent features. Feature selection method is simply selecting features that are relevant to the learning task using domain knowledge, mutual information, prediction accuracies, etc. Latent features are generating some linear or non-linear combination of features to provide a better representation of the dataset. PCA is one of the famous linear dimension reduction technique, and Isomap or Laplacian eigenmaps are examples of non-linear dimension reduction techniques, also called manifold learning.
- What is the general load (time, space, $) for preprocessing

Let there be n samples in d feature space. To handle the missing values, we should at least go through the whole data set. If we are simply removing the rows or columns with missing entries, then the time complexity would be O(nd), where space complexity would be linear. However, if we are applying some logic to fill in the missing entries, or reducing the dimension of the dataset using latent features, the complexities would vary a lot according to the method we are using.

- What challenges does each step present?
  There's no right or wrong way in handling the missing data. In practice, a feature with many missing entries may be found as a critical feature for the data set. Also, the result of filling in missing entries with values depends on the specific data set. Hence, it's challenging to choose the way of handling the missing data. Also, experimenting with many preprocessed datasets is time-consuming.
  When changing a continuous feature to a discrete or a logical value, it is hard to determine where to split the values. If a histogram shows us obvious points that differentiate block of points from one another, we can use those points as the splitting points. However, in practice, it is hard to find the right binning for features.
  Lastly, in dimensionality reduction, you need to choose which method to use, the dimension to project the data on, parameters, and techniques underlying the method. To choose the number of dimensions, you can try projecting the feature space to many different dimensions and then run learning techniques separately on the dimension reduced data. This procedure is challenging because best parameter set learned from each learning technique is dependent on that specific data set. Hence, it may not be feasible to try all the configurations of parameter sets per data set.

2.2 Mining, Interpretation, and Action

- Briefly discuss the top 10 algorithms.
  The top 10 algorithms listed below is from the paper "Top 10 algorithms in data mining"[8].

  1. C4.5

     There are many systems that generate classifiers. Among those, C4.5 constructs classifiers expressed as decision trees. C4.5 is capable of constructing more comprehensible rulesets.

     C4.5 utilizes divide-and-conquer[3] The algorithm for constructing the decision tree. It follows the simple rule. Assume a set S of cases are given. It chooses a test based on attributes with more than one outcomes. Grow the tree with each possible outcomes as the root, then recursively, partition S into its corresponding subsets. If a number of cases in S get smaller than some threshold, or all the outcomes in S belong to the same class, the leaf of the tree is labeled with the most frequent class in S.

     Along the procedure, C4.5 uses two heuristics: (1) Information Gain to minimize the total entropy of the subsets, (2) default gain ratio to divide the information gain using the information by the test outcomes. C4.5 could be used in either numeric or nominal features. For numeric attribute, a threshold could be used to split the values in a feature into discrete value sets. To overcome the overfitting problem, pruning algorithm is used.

  2. k-Means

     the k-means algorithm simply uses an iterative method to partition a data set into k different clusters. k-Means is one of the representation learning methods, meaning that training and testing are done at the same time.

     After initializing the k centroids either randomly or with some logic given by the user, it follows the two steps to cluster the data sets: (1) Assign data point to its closest centroid, (2) Relocation of the centroids. There are different methods in this procedure, such as running through step (1) for all the data sets and then relocate the centroids, which is Lloyd's algorithm. Other methods, such as exchange algorithm, include performing step (1) and step (2) one by one until the convergence.

     At some point, the assignments of points would not change, which implies the convergence. Euclidean distance is usually used as the measure of distance. With different initialization of the centroids, k-Means sometimes result in different clustering results.

  3. SVM (Support Vector Machines)

     Support Vector Machines is a classification algorithm with solid theoretical foundation and not sensitive to the number of dimensions in the dataset. When the dataset S is linearly separable, Support Vector Machines finds a hyperplane that best separates the two classes which maximize the sum of the margins between the two classes.

     Most of the times, the dataset would have noises that hamper the perfect classification. In this case, Support Vector Machines handles the noisy data with the idea of "soft margin." Support Vector Machines introduces a slack variable that takes account of the misclassified points in the training set. The costs from the slack variables are later used to modify the objective minimization function.

     Also, training data may not be linearly separable. In this case, kernel tricks are applied to the

[3]Divide and conquer an algorithm design paradigm in computer science that divides a problem into sub-problems. These sub-problems are conquered and then later combined to solve the original problem.

dataset to enable it to be linearly separable. Support Vector Machines does functional mapping of the original data set to a different space. For classification problems with more than two classes, Support Vector Machines can be extended by selecting one class as the positive class, others as negative then apply the learning technique. This is one-against-all method and Support Vector Machines applies this method to all classes to find many separating functions.

4. Apriori

   Apriori algorithm finds frequent itemsets to derive association rules. There's an assumption of the dataset in before using the algorithm: data set should be sorted in lexicographic order. Apriori goes through the database and look up for frequent itemsets of size 1, increase the count, then collect items which satisfy the minimum support requirement. Next, it (1) generates candidates of frequent itemsets of size k+1, (2) calculate the support of each candidate of frequent itemsets in the database, (3) look for itemsets that satisfy minimum support requirement, then add the itemsets, to extract frequent itemsets.

   Unlike many pattern finding algorithms developed in machine learning research community, Apriori boosted data mining research. One of the outstanding improvement of Apriori is that eliminating candidate generation by using FP-growth.

5. EM (Expectation-Maximization)

   Expectation-Maximization utilizes mixture models to estimate the underlying density function. The algorithm has two steps, (1) E-step that takes the expectation of a log likelihood of the mixture model, and (2) M-step updates estimates of the mixing proportion, mean vector, and covariance. These EM steps are repeated until the variation of the parameters is less than some threshold.

6. PageRank

   PageRank is a search ranking algorithm that uses hyperlinks on the Web. Apart from looking at the sheer number of votes from page x to page y as a vote by page x for page y, it analyze pages that casts the vote. PageRank uses the idea of ranking used in social networks. In this way, PageRank produces a static ranking that is independent from search queries.

   The Web is regarded as a directed graph, where each webpage corresponds to a node, and the hyperlinks correspond to edges. PageRank scores a page (prestige score) according to the importance of pages it's pointing and pages it's being pointed by.

7. AdaBoost

   The AdaBoost algorithm is an ensemble method that utilizes a base learning algorithm, then generate many other weak learners to be later used in majority voting for the final classification. AdaBoost is simple, has a solid theoretical foundation and performs well when tested in many different domains.

   AdaBoost first assigns equal weights to training data. AdaBoost calls the base learning algorithm to the data set and the distribution of the weights, then generate a base (weak) learner `h`. `h` is tested by training examples, and the weights get updated; if there are incorrectly classified examples, the weights will increase. From these, another weak learner is generated. This procedure is repeated for `T` times, and the final classification is done by majority vote from `T` different learners.

8. kNN (k-nearest neighbor)

   The k-nearest neighbor algorithm simply finds k nearest samples of a specific object, then assigns a label to the object based on the majority vote. In short, labels of the k nearest neighbors determine the label of the object.

   There are some issues when using k-nearest neighbor. If small k is used, then the classification would be vulnerable to noises. However, if large k is used, then it may include many points from other classes. Also, a majority vote can be a problem if neighbors vary according to their distances. In this case, weights can be used in the voting that corresponds to the distance from the object and samples.

9. Naive Bayes

   Naive Bayes is one of the simple probabilistic classifiers that has its foundation on Bayes' theorem. The task is to estimate the posterior probability of which cluster the specific object belongs to given the feature vectors. This probability can be calculated by using prior, likelihood, and the evidence. Naive Bayes doesn't have complicated parameters to tune with, which makes it a simple classifier. However, independence assumptions among features should be met to use this classifier.

10. CART (Classification and Regression Trees)

    CART decision tree builds decision trees in a way that is different from a previously mentioned C4.5 algorithm. Trees grow to their maximal depth and then pruned back to the root. In this procedure, features that contribute least to the overall performance is pruned, and each decision tree is stored. The performance of these trees is tested on the test data set only which is different from other learning techniques.

CART is simple to use because all you need is to feed the raw data to the CART decision tree. CART automatically balances the classes and handles missing values. CART also reports attribute importance ranking that could be used to interpret the model.

- Does datamining tell us what to do?
  Data mining doesn't tell us what to do. However, we can get an idea of what to do when we are given a data mining problem to solve. If we are given a problem to solve, unless we are using CART which we just need to provide the raw data for analysis, we need to first process the data to a format to be used in the analysis. Then, we choose appropriate algorithm based on our knowledge of different learning techniques and try many different sets of parameters to fit the model to find a good working parameter set.
- What are some new types of problems in data mining?
  We are living in an era of big data, where massive data with hidden asset underlying them lies around us. Data mining research nowadays tries to incorporate these aspects. Some f the new types of problems in data mining include developing more efficient algorithms for massive data sets, privacy-preserving methods, and new machine learning approaches[9].

3. Author Idenfication: Full Problem Description

Author Identification problem requires users to predict the names of authors given excerpts from their writings. It is a multiclass supervised learning problem which trains on a training dataset and tests on a separate test dataset. The training is conducted on excerpt data and on the labels, which are the abbreviations of the three authors.

To avoid overfitting and find the optimum bias-variance tradeoff, during training the training dataset will be split into training and validation datasets to consistently check for validation accuracy. After training, the testing will be conducted on the learned model, and the accuracy will be derived. The loss is computed as a categorical cross entropy, and the accuracy will simply be as follows:

$$Accuracy = \frac{NumCorrect}{Total}$$

3.1 Data Analysis

The provided training and test data are as follows. They each have three columns, the first being the unique id of the excerpt, second the excerpt itself, and third the author of the excerpt. The unique IDs are discarded as the IDs for the training, and test data points will all be non-identical, which means there will be no pattern to learn from. The excerpts are then separated from the labels into two sets of files, the X excerpt dataset and $ytrain.txt$, the labeled dataset. The excerpt dataset is preprocessed before being written to the file. The punctuations are removed. Also, the stop words are removed initially but kept later due to good performance results. All words are turned into lower cases and also lemmatized, which acts as a form of normalization. Furthermore, the excerpt dataset is processed into four separate datasets. It is split into a bigram, trigram, and fourgram character models, and also a word dataset.

3.1.1 Statistics

Words The words sorted by the frequency can be visualized in a histogram. The top ten words for the entire excerpt of all three authors is visualized in Figure 1.

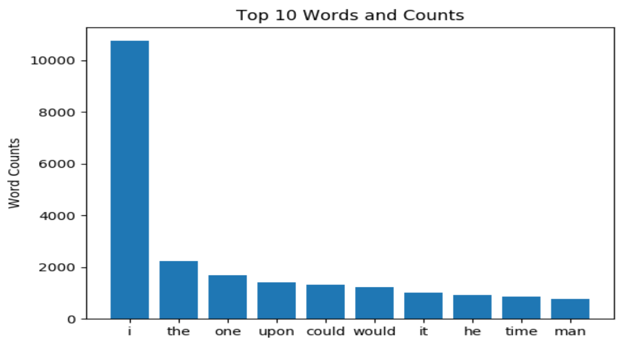


Figure 1. Top ten words by frequency

For HP Lovecraft, the top ten words is shown in Figure 2. It can be observed that the top 10 words are somewhat similar.

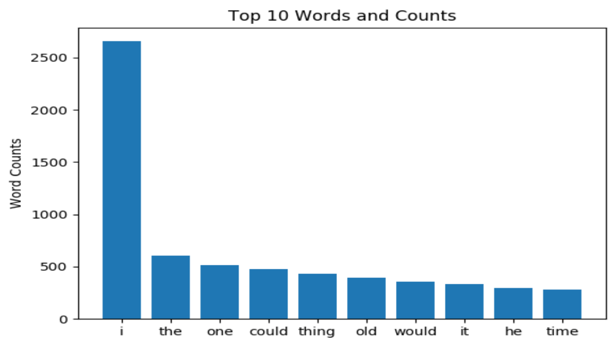


Figure 2. HP Lovecraft: Top ten words

For Edgar Allen poe, the top ten words is shown in Figure 3. It is also similar to the top 10 words, but there are certain words such as **but** or **upon** that distinguishes his excerpts with that of HP Lovecraft.

Finally for Mary Wollstonecraft Shelley, the top ten words is shown in Figure 4. Besides the extremely common words such as **I** or **the**, it can be seen that there are significantly unique words used by Shelly such as **yet** and **heart**.

The counts of the words for the entire excerpt and in each of the authors' total excerpts are shown in Table 1. Immediately from the table, it can be noticed that Edgar Allen Poe

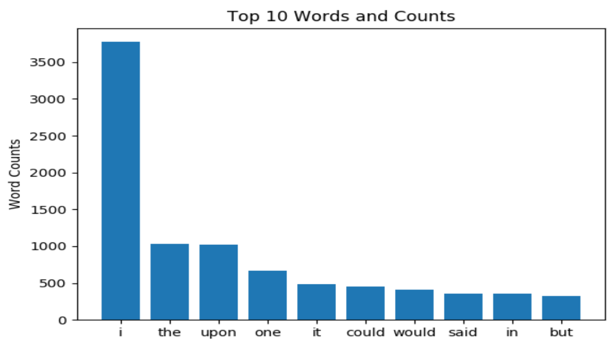


Figure 3. Edgar Allen Poe: Top ten words

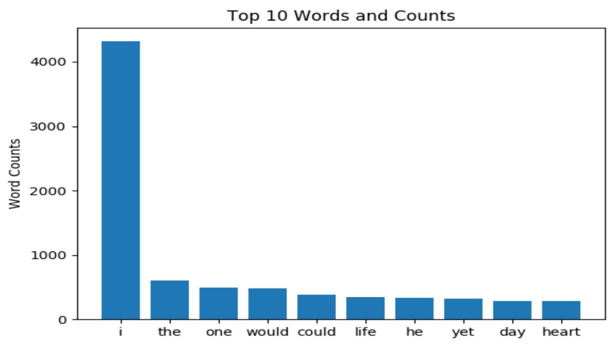


Figure 4. Mary Shelley: Top ten words

used the word **upon** significantly more than the other two authors (1025 as opposed to 186 and 200).

3.2 Methods

3.2.1 Algorithm

The algorithm chosen is the Word2Vec with memory cells, such as the LSTM (Long Short Term Memory) cell. The reason for choosing this is it is an efficient way to convert words in our dataset into numeric vectors which can then be inputted into our models. Furthermore, the Word2Vec algorithm uses the skip-gram approach. The skip-gram approach can maintain the relationships each word has with other words by estimating the probability of other words appearing close to each word.

As an overview of the structure, the input dataset, which is one hot vector, is linearly transformed into an embedding layer, which is then fed into a softmax output layer. The embedding layer is created with dimension 300 for the feature vectors. The embedding matrix is created with the four types of inputs: bigrams, trigrams, fourgrams, and words. To enhance the learning, memory cell layers are added after the embedding layer.

3.2.2 Background

Yoav Goldberg and Omer Levy's paper, Word2Vec explained, provides the mathematical foundation and conceptual information on the Word2Vec model with the skip-gram approach [10]. Goldberg and Levy explain the skip-gram model and also a negative sampling approach, which is shown to be more effective although not implemented in our model. The authors also go in depth in the sampling steps where one is encouraged to subsample and prune the rare-words.

Table 1. Word Count by Author

	Word	Total	HPL	MWS	EAP
1	i	10746	2656	4316	3774
2	the	2241	603	604	1034
3	one	1676	516	489	671
4	upon	1411	186	200	1025
5	could	1316	480	383	453
6	would	1241	357	475	409
7	it	1011	335	194	482
8	he	917	299	330	288
9	time	869	279	276	314
10	man	778	279	242	257
11	day	743	197	288	258
12	thing	725	433	71	221
13	but	724	158	245	321
14	yet	715	165	318	232
15	eye	707	157	284	266
16	said	704	140	208	356
17	even	701	192	248	261
18	a	654	172	184	298
19	might	629	172	269	188
20	old	616	392	85	139
21	in	616	158	107	351
22	like	613	273	167	173
23	u	611	98	272	241
24	life	608	144	349	115
25	first	602	142	211	249
26	must	594	186	212	196
27	night	586	260	175	151
28	thought	576	154	230	192
29	never	570	193	175	202
30	made	565	136	166	263

Another resource on Word2Vec with skip-grams is Improving Lexical Embeddings with Semantic Knowledge by Mo Yu and Mark Dredze [11]. Yu and Dredze focus on extending the model to capture the desired semantics in a neural language model. Word2Vec is one of the models explained in the paper, and the authors go on to explain the thought process in parameter estimation.

Also, Google's Semi-supervised Sequence Learning by Andrew Dai and Quoc Le dives into the LSTM cell with the Word2Vec model [12]. This paper delves into the usage of LSTM Recurrent Neural Networks (RNN) that can achieve good performance with pre-trained models. It is shown that pretraining stabilizes the learning in recurrent networks.

3.2.3 Challenges

The challenges to this method were in two main parts: the pretraining the Word2Vec language model and structuring the optimal neural network architecture.

Strangely, the model seemed to perform similarly without the pre-trained model and starting with a non-initialized embedded matrix. This was a demotivating result; nonetheless, it was still used since it provided approximately a 3 percent increase in accuracy when the model was used.

Regularizing to avoid overfitting was one of the main challenges. The training error at times for the fourgram model could reach up to 99 percent accuracy, yet the validation accuracy showed approximately 70 percent.

3.2.4 Softwares

The main programming was done in Python 2.7. To preprocess, the libraries used were the built-in string and nltk (v3.2.5). The string library was used to remove punctuations and lower case the words. nltk's stopword corpus and WordNetLemmatizer was used to initially remove stopwords and lemmatize the words respectively. The Word2Vec model was created through the gensim package (v3.1.0). The training step was done with numpy, sklearn (v0.19.0), and mainly keras (v.2.1.2). The sklearn is used to encode the authors names into numerical integers and for the kfold validation. The keras is applied to tokenize the text into sequences and more importantly build the training model.

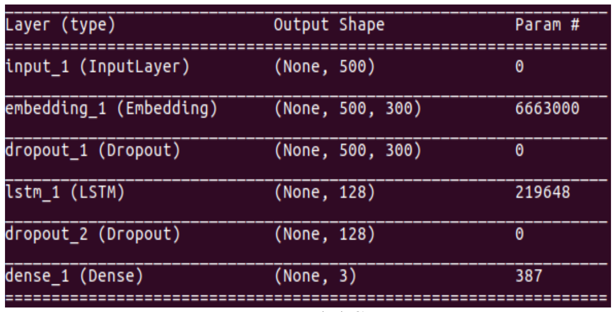
3.2.5 Hardwares

Linux kernel was the operating system for the system, with 4.4.0-62-generic machine hardware. Architecture is x86_64, 40 CPU, 2 socket, 10 core per socket, and 2 threads per core. Model name is Intel(R) Xeon(R) CPU E5-2630 v4 @ 2.20GHz and there are 4 GPU Titan X cards.

3.2.6 Input Structure

The structure of the data is four excerpt datasets (bigrams, trigrams, fourgrams, words) each with 19579 data samples and the label dataset also of the respective 19579 samples [13]. The feature of the unique ID is removed since the uniqueness provides no special predictive qualities.

The structure of the model is comprised of an input, embedding, dropout, LSTM, dropout, and a dense layer. The input layer accepts the excerpt data and outputs a dimension of a designated maximum sequence length, which is 500. It is processed with a embedding matrix to create the next embedding layer, which outputs a 500 by 300 dimension as a result [14]. The first dropout layer also takes 500 by 300 dimensions as input and outputs the same dimension. Then the LSTM layer, with 128 hidden nodes, takes the input from the dropout layer and outputs a 128 dimension vector. The second dropout layer also takes 128 dimension as input and outputs the same dimension. Finally, the dense layer (fully-connected) accepts the 128 dimension as input and produces a one hot vector of dimension 3, each point representing the author. An example illustration of the model structure is shown on Figure 5.



```
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 500)               0
_________________________________________________________________
embedding_1 (Embedding)      (None, 500, 300)          6663000
_________________________________________________________________
dropout_1 (Dropout)          (None, 500, 300)          0
_________________________________________________________________
lstm_1 (LSTM)                (None, 128)               219648
_________________________________________________________________
dropout_2 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_1 (Dense)              (None, 3)                 387
=================================================================
```

Figure 5. Model Structure

3.2.7 Presentation

On the given model and data structure as mentioned previously, the training was done on the four types of excerpt datasets. The following results are conducted with 0.2 dropout rate, default optimizer parameters (0.001 learning rate and no decay), and 10 epochs and 5 fold cross validation.

For the word excerpt, the training and validation accuracy is shown in Figure 6a. The training and validation error is also shown in Figure 6b.

It can be clearly seen that there is significant overfitting at epoch 2 and regularization needs to be done. The training error reaches almost perfect percentage but the validation is approximately at 70 percent.

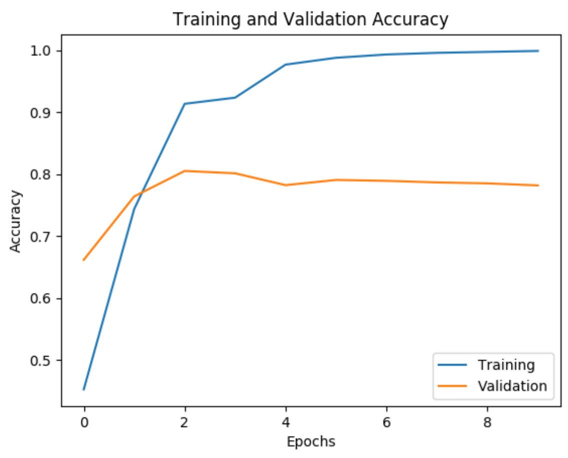
Several regularization strategies were employed. Bias regularization, kernel regularization, recurrent regularization, dropout, recurrent dropout, and batch normalization [15]. For the training, elastic net regularization was used. The bias regularization seemed to have not hampered the learning but not helped, the kernel regularization stopped all learning, and recurrent regularization decreased the accuracy by a few percentages. The bias regularized results can be seen in Figure 7.

The recurrent dropout with 40 percent gave promising results as well but also seems to have difficulty going above 80 percent accuracy [16]. The error rate begins to go back up after three epochs. Batch normalization also did not seem to produce good results. Hence, it was decided to continue using the bias regularization. It was decided after looking at Figure 7 that the error rate has not exactly flattened and could train for a few more epochs. The results can be seen below in the results section.

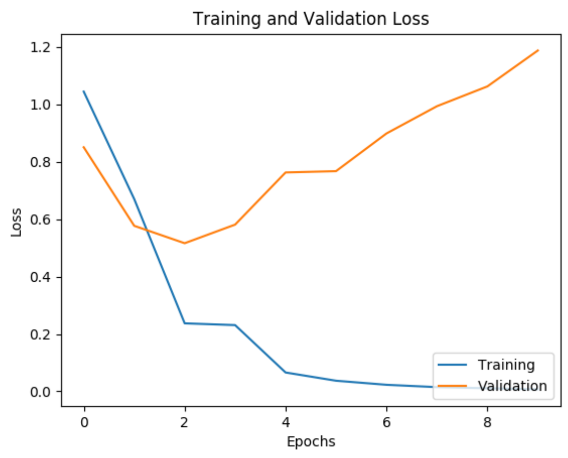
3.3 Results

The results for all four excerpt formats for the same experiment parameters as above are shown in Table 2. The accuracy are shown in percentages.

The results seem successful for the training as almost 100 percent accuracy is achieved. The testing accuracy is deemed to require some regularization to prevent overfitting. It can be seen that the word model performs better than the n-gram models. This was an unexpected result because the character model was expected to perform better. Another surprising result is that the accuracies did not differ too much when the excerpts were trained with and without the pretrained

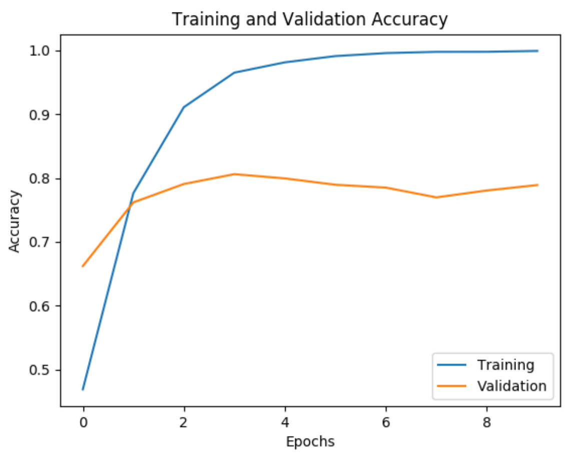


(a) Accuracy

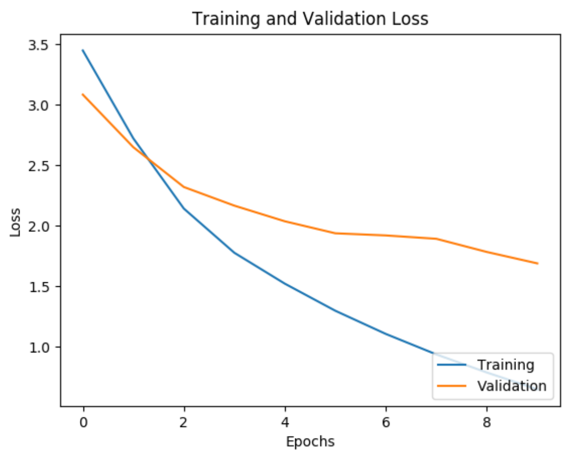


(b) Loss

Figure 6. Training and Testing Results



(a) Accuracy



(b) Loss

Figure 7. Bias Regularized Results

Word2Vec models. For fourgram and word excerpts there was approximately a 3 percent improvement when the model was used, which is promising. However, it is surprising that for the bigram excerpt the performance was better when the model was not used.

Furthermore, the dimension of the feature vector did not seem to produce a different result. This seemed strange since when the dimension was 50 it produced a similar performance than when the dimension was 300.

Further experimentation was done to improve the training testing gap in accuracy and loss. The experiment was focused on the dropout rate together with the bias regularizer with elastic net. The dropout rates experimented were 0.0, 0.2, 0.3, and 0.5. Figures 12, 13, 14, and 15, show respectively the results of the dropout rate (Appendix A).

Experimentation on Dropout Rate 0.5 seemed to have the most continuing decrease in the validation loss; hence, more experimentation with more epochs was done. The results are shown on Figure 16 (Appendix A). It can be seen that although the loss seems to continuously decrease, the accuracy does not reflect this behavior in the validation dataset.

3.4 Summary and Future Work

The project objective was to predict the correct author to the given excerpt data. With a Word2Vec model, embedding layer, and an LSTM layer, the training prediction was able to achieve almost 100 percent and testing 80 percent. Regularization was applied with aims to increase the validation error, although it was minimal. In the future, I would try different models (character and word model combined or word n-gram models) and different network architectures (deeper with more layers or other memory cells such as GRU) [17].

4. Iceberg: Full Problem Description

Drifting icebergs are threats in some areas because they hamper navigation and activities in these areas. You can monitor the movements of these icebergs, but the harsh weather in these remote areas make it extremely difficult to do so. As an

Table 2. Training and Testing Accuracies

Excerpt and Model	Training	Testing
Bigram With Model	68.26	63.41
Bigram Without Model	66.12	62.41
Trigram With Model	87.35	74.64
Trigram Without Model	88.24	73.62
Fourgram With Model	96.33	79.24
Fourgram Without Model	94.82	76.79
Word With Model	99.80	78.84
Word Without Model	99.45	75.80

alternative to observing this iceberg from the ground, Statoil had worked with C-CORE to provide satellite data that could be used to train models that can classify icebergs from ships.

We've fed the training data set as an input to classification models, then applied the models to the test datasets. The output is the probability of the test images of being an iceberg. We've tried five different learning techniques, and have done several different ways of processing the data for the analysis.

4.1 Data Analysis

There are two datasets: the training set and test set. There are 1604 images in the training set. Out of those, 753 images are labeled as an iceberg, and 851 images are labeled as a ship. Each image is represented in 5625 elements from `band_1`, 5625 elements from `band_2`, which are flattened images of data, and `inc_angle`, the incidence angle of the image. In summary, there are 1604 images in total with 11,251 features. For the test set, only the label was missing.

The raw data was in a .json format. We first read the data into Pandas data frame using Python. We simply converted the data frame into a .csv file which could be used directly in the analysis. Then, we made different versions of data sets to be used in further experiments.

By reshaping bands into 75x75 pixel matrices, and normalizing dB frequency to a 0 to 1 scale, we can visualize the bands as a picture. This allowed us to inspect the data before working with the data visually.

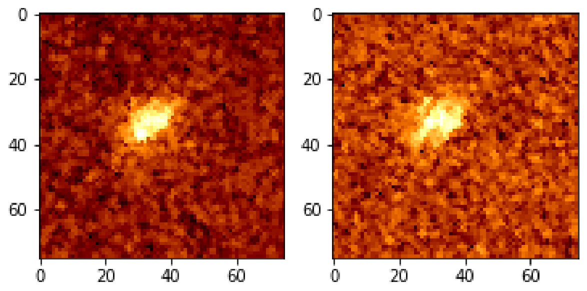


Figure 8. Example visualization of iceberg. (Left) Band1 picture, (Right) Band2 picture.

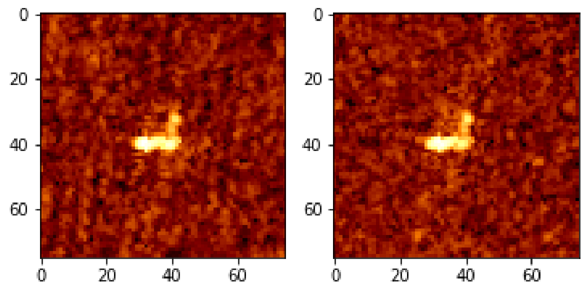


Figure 9. Example visualization of ship. (Left) Band1 picture, (Right) Band2 picture.

4.2 Data Pre-processing

1. Eliminate feature `inc_angle`

   `inc_angle` had 133 missing entries out of 1,604 entries. All these rows were in class `ship`. Because there should not be any missing values to feed in as input to classification models, we simply removed this feature. The shape of the data set is (1604, 11250).

2. Remove rows with `na` in the feature `inc_angle`

   As an alternative method to the previous one, we removed the entries with missing values in the feature `inc_angle`. In this way, there are 753 entries classified as an iceberg, and 718 entries classified as a ship. The shape of the data set is (1471, 11251).

3. Merge two bands together and Remove rows with `na` in the feature `inc_angle`

   Here, we merged two bands (11250 elements) into one band (5625 elements). Because the two bands are in log scale, we took exponential value for each element, and then added the value of the two elements (one each in band1 and band2). We took the log of the new band. Also, we applied the technique above to remove rows with `na` in the feature `inc_angle`. The shape of the data set is (1471, 5626).

4. PCA dimension reduction on the processed data set (Removed rows with `na` in the feature `inc_angle`)

   Used PCA in `sklearn.decomposition` package to reduce the features. In this procedure, we reduced the features of the bands only. After the dimension is reduced, we added `inc_angle` feature to the dimension reduced data set.

   - Number of dimensions: 100, Shape of the data set: (1471, 101)
   - Number of dimensions: 50, Shape of the data set: (1471, 51)

5. Decibel normalization

Decibels(dB) were normalized for bands to fit on a 0 to 1 scale.

$$X_{Normi} = \frac{(X_i - min(X))}{(max(X) - min(X))} \quad (2)$$

6. Re-Shape Bands for CNN Input

   Scaled bands were reshaped to 75x75 matrices. Band1 and Band2 images were merged into single matrices of size 75x75x2 to create a dual channel image. This was repeated for all training (1604) and testing (8424) inputs.

Here are some summary statistics of the features of the processed data set (Remove rows with `na` in the feature `inc_angle`). band1_1 is the first element of the band1

-	inc_angle	band1_1	is_iceberg
count	1471.000000	1471.000000	1471.000000
mean	39.268707	-21.328999	0.511897
std	3.839744	4.651750	0.500028
min	24.754600	-34.342476	0.000000
25%	36.106100	-24.601152	0.000000
50%	39.501500	-21.547363	1.000000
75%	42.559100	-18.249153	1.000000
max	45.937500	-0.409181	1.000000

4.3 Methods

4.3.1 Algorithm

1. k-nearest neighbor

   k-nearest neighbor is in top 10 algorithm, which is explained in the above section [2.2 Mining, Interpretation, and Action]. The k-nearest neighbor algorithm simply finds k near-est samples of a specific object, then assigns a label to the object based on the majority vote. In short, labels of the k nearest neighbors determine the label of the object.

2. SVM

   SVM is also in the Top 10 algorithm, where we explained in [2.2 Mining, Interpretation, and Action]. When the dataset S is linearly separable, Support Vector Machines finds a hyperplane that best separates the two classes which maximize the sum of the margins between the two classes. It also uses the concept "soft margin" to handle the noise in the data, and use kernel tricks that projects the data set into space where the data could be linearly separated in that transformed space.

3. Random Forest

   Random Forest is an ensemble learning method that is well suited for classification that creates many Decision Trees and then let them vote for the final classification. Usually, the more trees you build, the better result Random Forest gives.

4. Neural Networks

   Neural Networks is a Machine Learning algorithm that feeds in mostly the raw data and let the complicated structure of hidden layers, neurons, and activation functions to train the classification model. The method is different from kernels used in SVM, because, kernel tricks are used to preprocess the data to choose the kernels manually. However, Neural Networks automatically trains the model by updating weight matrices in steps for forward propagation and backward propagation.

5. Convolutional Neural Networks

   Convolutional Neural Networks are Neural Network with specialized connectivity structure. Convolutional Neural Networks stack multiple stages of feature extractors. Here, higher stages compute more global and more invariant features. There are four main components to a Convolutional Neural Network: (1) Convolutional Layer, (2) Nonlinear Activation Function (e.g., ReLu), (3) Pooling Layer, and (4) Fully-Connected Layer.

4.3.2 Background

We used four different classification algorithms on data sets created in 4 steps in [4.1 Data Analysis]. Those are k-nearest neighbor, SVM, Random Forest, Neural Networks, and Convolutional Neural Networks. There are reasons for choosing above algorithms. Reason for using k-nearest neighbor is to use as a baseline due to its simplicity in modeling. We used SVM with different kernel tricks because the problem we are solving is binary classification, and SVM is a learning technique that exactly does binary classification. Also, we used Random Forest because there were two algorithms in top 10 algorithms used in data mining that utilizes Decision Tree, hence voting from many Decision Trees seemed to make sense for a better classification. Also, Neural Networks is the most popular algorithm used in Machine Learning, and due to the high-dimensionality of the features, we used Neural Networks which is qualified for complex data sets.

Apart from the above four classification algorithms, we also used Convolutional Neural Networks. The reason for using Convolutional Neural Networks is because it is widely used especially in Computer Vision. Since our data set is images, we thought Convolutional Neural Networks would give us the best training and testing accuracy of the data set.

4.3.3 Softwares

- Python v3.6.3
- Keras v2.0.9
- Tensorflow-gpu v1.4.0
- numpy v1.13.3
- pandas v0.21.0
- matplotlib 2.1.0
- scikit-learn v0.19.1

4.3.4 Hardwares

- Personal Desktop
  System Requirements Processor: AMG Phenom(tm) II X6 1055T Processor 2.81 GHz 6-Core
  Installed memory (RAM): 16.0GB
  System type: 64-bit Operating System, x64 based processor
  GPU: NVIDIA GeForce GTX 1050 Ti
  Operating System: Windows 10 Pro

- Tank Server
  model name: Intel(R) Xeon(R) CPU E5-2680 v3 @ 2.50GHz
  cpu cores : 12
  MemTotal: 528082800 kB
  MemFree: 66912068 kB

- Karst Server
  model name : Intel(R) Xeon(R) CPU E5-2650 v2 @ 2.60GHz
  cpu cores : 8
  MemTotal: 66078368 kB
  MemFree: 5558776 kB

We mostly used Python for this problem. We used the scikit-learn library for the k-nearest neighbor, SVM, Random Forest, and Neural Networks. We also used Pandas for pre-processing, and Numpy to use matrix configuration of data sets for speeding up the analysis.

We used `sklearn.decomposition` library to apply PCA on the data set. For drawing plots, we used `matplotlib`. For CNN we used Keras with a Tensorflow back-end to build the model.

4.3.5 Structure

- Feature `inc_angle` removed - Data structure for k-nearest neighbor, SVM, Random Forest, and Neural Networks: 1604 x 11250

  Model structure for this configuration is in the appendix, Table 20 (a).

- Rows with `na` in `inc_angle` removed - Data structure for k-nearest neighbor, SVM, Random Forest, and Neural Networks: 1471 x 11251

  Model structure for this configuration is in Table 20 (b).

- Merged two bands together, and removed rows with `na` in `inc_angle` removed - Data structure for k-nearest neighbor, SVM, Random Forest, and Neural Networks: 1471 x 5626

  Model structure for this configuration is in Table 20 (c).

- PCA dimension reduction - Data structure for k-nearest neighbor, SVM, Random Forest, and Neural Networks: 1471 x 51

  This data set had 1471 images where rows with `na` in the feature `inc_angle` is removed. It has 51 features in total: 50 features after applying PCA on the bands, and 1 feature for `inc_angle`. Model structure for this configuration is shown in Table 3.

Table 3. SVM, Random Forest, KNN, NN Model 4 Architecture

Support Vector Machines
- **kernel:** radial basis function
- **C:** 512.0
- **gamma:** 1.9073486328125e-06

Random Forest
- **criterion:** gini index
- **number of trees:** 1024
- **min split nodes:** 2

k Nearest Neighbors
- **number of neighbors:** 3
- **weight metrics:** uniform

Neural Networks
- **hidden layers:** (64, 16, 64)
- **alpha:** 0.015625
- **activation function:** identity
- **solver:** adam

- Final structure for CNN

  1604x75x75x2 Dual Channel Pictures (n pictures, width, height, depth)

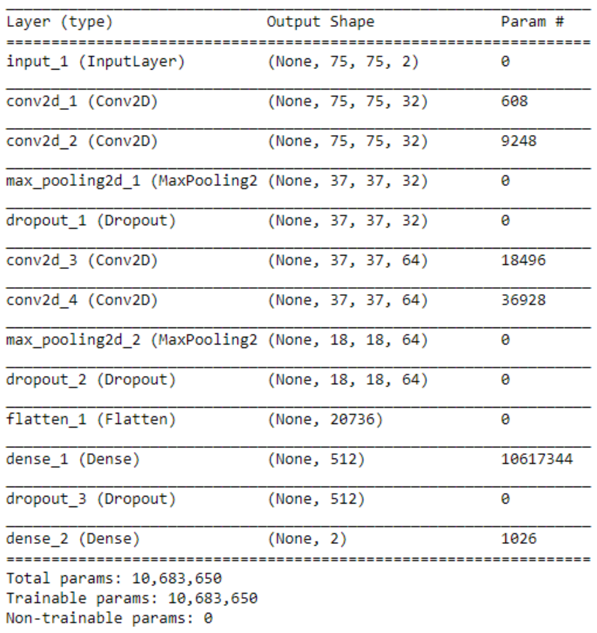
4.3.6 Convolutional Neural Networks Implementation

Using the pre-processed dual channel image we built a model of the convolutional neural network to classify the pictures. For this project, we tested tgree CNN architectures with different hyper parameters.

The first model was built based on a basic CNN model described by Velickovic [18]. The model was built with a batch size of 32, which defines the number of samples propagated through the network per batch. The CNN also was implemented with a 3x3 convolutional filter, and 2x2 pooling filter. We had a convolutional depth of 32 kernels or neurons in the first convolutional layer, and 64 kernels in the second convolutional layer. We had two dropout layers first with the probability of 0.2 and the second with 0.65. Lastly, we had a fully connected layer consisting of 512 fully connected neurons. All convolutional layers used ”same” padding method and a ”relu” activation function. The model consisted of a total of 10,683,650 parameters. A summary of the model can be found in Table 4 below.

The second model we tested, was based off a model described by Chollet[19]. Model two had a smaller batch size of 32. The CNN also was implemented with a 3x3 convolutional filter, and 2x2 pooling filter. This model consisted of three convolutional layers, the first two with 32 kernels and the last convolutional layer with 64 kernels. There is a single drop out layer with a 0.5 drop out the probability and a fully connected layer of 512 neurons. Much like model 1, model 2 also used

Table 4. Convolutional Neural Network Model 1 Architecture



```
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 75, 75, 2)         0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 75, 75, 32)        608
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 75, 75, 32)        9248
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 37, 37, 32)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 37, 37, 32)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 37, 37, 64)        18496
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 37, 37, 64)        36928
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 18, 18, 64)        0
_________________________________________________________________
dropout_2 (Dropout)          (None, 18, 18, 64)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 20736)             0
_________________________________________________________________
dense_1 (Dense)              (None, 512)               10617344
_________________________________________________________________
dropout_3 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_2 (Dense)              (None, 2)                 1026
=================================================================
Total params: 10,683,650
Trainable params: 10,683,650
Non-trainable params: 0
```

a relu activation function. A summary of the model can be found in Table 5 below.

For our third model we modified model 2 to contain a batch size of 16, while keeping all other hyperparameters the same. A summary of the model can be found in Table 6 below.

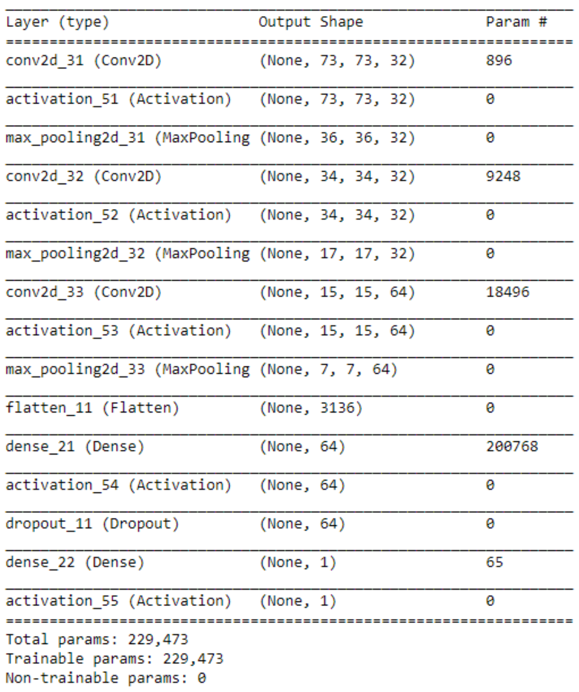
All three CNN models used a 10 fold cross-validation method. The number of epoch's for training were decided based on the accuracy and error rates to determine when the model began to become over fit and to determine the optimal epoch to end training.

4.4 Results

Figures were drawn from 10-fold cross-validation on the training set. The boxplots describe 10 different accuracies for each classifier within the 10-fold cross-validation. Figure 21 (a) in the appendix is our first experiment with the processed data with the feature `inc_angle` removed because it had missing values. For the next experiment, we removed the rows with missing values other than the feature. The experiment in this setting is depicted in Figure 21 (b). Figure 21 (c) is the experiment with using the data set where we merged two bands. Then, we performed dimension reduction on the two bands, reducing the dimension to 50. The performance of all classifiers significantly improved after reducing the dimension of the dataset. The result of this configuration is shown in Figure 10.

Accuracy was tested on the 10-fold cross-validation on the training set. Out of four different experiments with a different way of preprocessing the data set, the best classifier was Random Forest classifier when modeled using dimension reduced data. This result was surprising, linearly projecting the high dimension features to the planes described the feature space better than the using the whole dimension.

Table 5. Convolutional Neural Network Model 2 Architecture



```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_31 (Conv2D)           (None, 73, 73, 32)        896
_________________________________________________________________
activation_51 (Activation)   (None, 73, 73, 32)        0
_________________________________________________________________
max_pooling2d_31 (MaxPooling (None, 36, 36, 32)        0
_________________________________________________________________
conv2d_32 (Conv2D)           (None, 34, 34, 32)        9248
_________________________________________________________________
activation_52 (Activation)   (None, 34, 34, 32)        0
_________________________________________________________________
max_pooling2d_32 (MaxPooling (None, 17, 17, 32)        0
_________________________________________________________________
conv2d_33 (Conv2D)           (None, 15, 15, 64)        18496
_________________________________________________________________
activation_53 (Activation)   (None, 15, 15, 64)        0
_________________________________________________________________
max_pooling2d_33 (MaxPooling (None, 7, 7, 64)          0
_________________________________________________________________
flatten_11 (Flatten)         (None, 3136)              0
_________________________________________________________________
dense_21 (Dense)             (None, 64)                200768
_________________________________________________________________
activation_54 (Activation)   (None, 64)                0
_________________________________________________________________
dropout_11 (Dropout)         (None, 64)                0
_________________________________________________________________
dense_22 (Dense)             (None, 1)                 65
_________________________________________________________________
activation_55 (Activation)   (None, 1)                 0
=================================================================
Total params: 229,473
Trainable params: 229,473
Non-trainable params: 0
```

We think our result is successful because we learned that different way of preprocessing gives us the different result of the classification. Especially, when we reduced the dimension of the dataset, we could train and test the model magnitudes faster compared to using the whole feature set.

The results suggest that although the data set was complicated, after reducing dimension using PCA to find 50 basis vectors that describe the data set well, Random Forest built a model that could classify iceberg and ship correctly by over 85 percent.

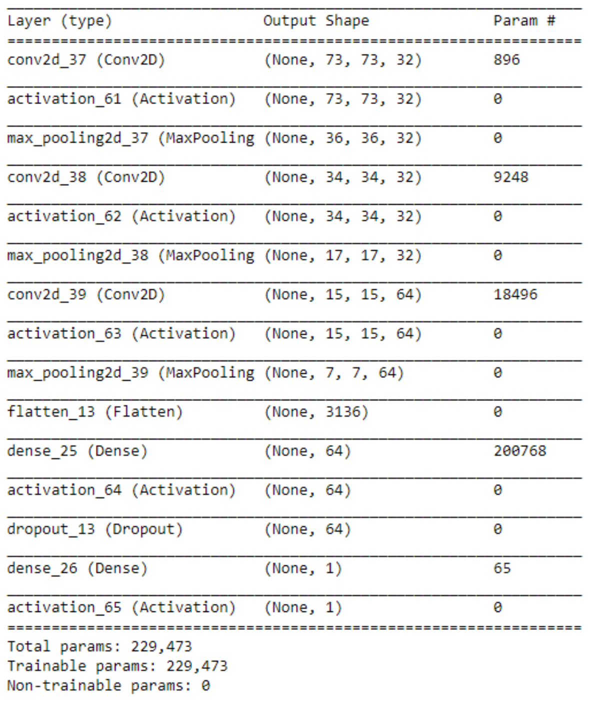
4.4.1 Convolutional Neural Network Results

Using a 10 fold cross-validation method, we were able to observe approximately 80 percent validation accuracy rate with Model 1 at 25 epochs. Model 1 had an approximate validation loss of about 29 percent. Figures for this model is shown in Appendix Figure 18. Considering the relative simplicity of the model, we can say the model performed relatively well as it outperformed most of our other methods.

Improving upon Model 1, we were able to achieve approximately 87.8 percent validation accuracy rate with Model 2 at epoch 35. The model to had an approximate validation loss of 24.5 percent at epoch 35. Figures for this model is shown in Appendix Figure 19. We can see in Figure 19 that the validation loss begins to become larger than the testing loss, which is an indication that our model is beginning to become over-fit. Thus we stopped our training at epoch 35.

Our most successful model was Model 3, which achieved approximately 90 percent validation accuracy rate at epoch

Table 6. Convolutional Neural Network Model 3 Architecture



```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_37 (Conv2D)           (None, 73, 73, 32)        896
_________________________________________________________________
activation_61 (Activation)   (None, 73, 73, 32)        0
_________________________________________________________________
max_pooling2d_37 (MaxPooling (None, 36, 36, 32)        0
_________________________________________________________________
conv2d_38 (Conv2D)           (None, 34, 34, 32)        9248
_________________________________________________________________
activation_62 (Activation)   (None, 34, 34, 32)        0
_________________________________________________________________
max_pooling2d_38 (MaxPooling (None, 17, 17, 32)        0
_________________________________________________________________
conv2d_39 (Conv2D)           (None, 15, 15, 64)        18496
_________________________________________________________________
activation_63 (Activation)   (None, 15, 15, 64)        0
_________________________________________________________________
max_pooling2d_39 (MaxPooling (None, 7, 7, 64)          0
_________________________________________________________________
flatten_13 (Flatten)         (None, 3136)              0
_________________________________________________________________
dense_25 (Dense)             (None, 64)                200768
_________________________________________________________________
activation_64 (Activation)   (None, 64)                0
_________________________________________________________________
dropout_13 (Dropout)         (None, 64)                0
_________________________________________________________________
dense_26 (Dense)             (None, 1)                 65
_________________________________________________________________
activation_65 (Activation)   (None, 1)                 0
=================================================================
Total params: 229,473
Trainable params: 229,473
Non-trainable params: 0
```

30, and had an approximate validation loss of 27.6 percent. Graphical depiction of the model during training and testing can be seen in Figure 11. As of December 14, 2017, this model was able to achieve a Kaggle submission score of 0.2632, placing our team ranking 1675th out of 2378 teams.

4.4.2 Challenges

Since we were experimenting with five different classifiers and with many data sets that were preprocessed differently from each other, finding good working parameters for each classifier per data set was the most challenging. Especially for SVM, kernel tricks and the learning rate alpha affected a lot on the accuracy of the training and testing.

For Convolutional Neural Networks the challenges mainly stemmed from the preprocessing of the data, as there needed to be substantial reprocessing for the inputs to fit into the

Table 7. Results Summary Table. Training and Testing Accuracies

Model	Training	Testing
SVM	85.38	83.46
Random Forest	85.38	85.25
KNN	79.33	77.37
NN	75.39	73.98
CNN Model 1	81.89	79.97
CNN Model 2	92.25	87.80
CNN Model 3	89.90	90.34

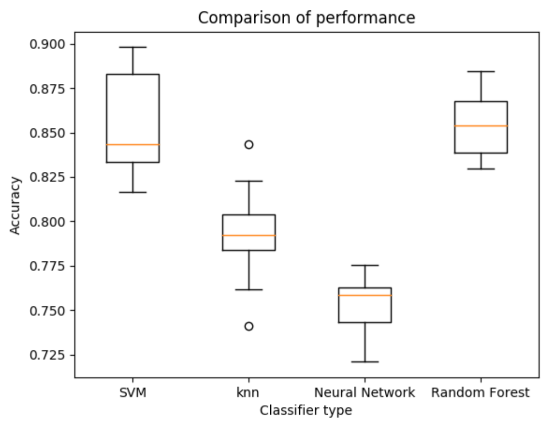
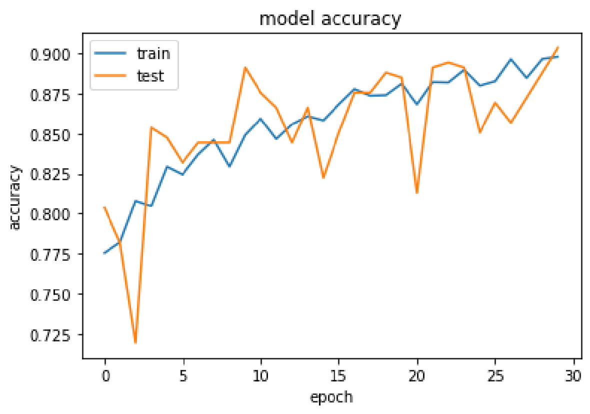
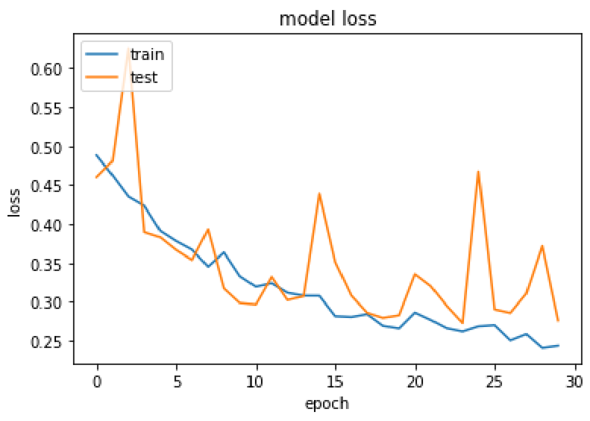


Figure 10. Comparison of the models on data set where dimension is reduced using PCA



(a) Model 3 Accuracy Graph



(b) Model 3 Loss Graph

Figure 11. (a) Y-axis is the model accuracy, X-axis is the epoch iteration. (b) Y-axis is the model loss, X-axis is the epoch iteration. Blue line indicates the model results on training dataset, Orange line indicates model results on testing set. Model was validated by 10 fold cross-validation.

model. In most cases of Convolutional Neural Networks, the images are usually in a single channel or triple channel RBG (Red-Blue-Green) input formats. As the provided inputs only contained two bands, we tried various methods to create a third band, such as finding the differences between the two available bands. Our attempts, however, were unsuccessful in that the introduction of a third band did not help in the accuracy of our model. We also tried merging two bands into a single band on a logarithmic scale, this however also did not improve accuracy. In addition to these two methods, we also tried reducing the noise in the picture, however, the classifier was able to classify images better with noise versus reduced noise. Thus for the analysis we stuck with two-channel scaled unfiltered images. Sample figures of these attempts can be found in Appendix Figure 17.

For all methods, a challenge which exceeded our expertise was image transformation and utilizing incidence angle to correct the decibel bands. We explored various methods for correction but failed to execute and thus decided to proceed and remove this feature from our analysis.

There was another big challenge when dealing with the data set. Test data was too large for us to preprocess it. Even the memory for the tank server was not enough. Hence, we used another supercomputer karst. By using karst, we were able to preprocess the large-scale data set.

4.5 Summary and Future Work

For this project, we experimented with five different configurations of the data set. Four of those data sets were trained and tested on four different classifiers (kNN, SVM, Neural Networks, Random Forests), and one of the models was used for CNN. As a result, CNN was able to achieve 90 percent accuracy where other algorithms reached up to 86 percent accuracy when trained and tested on 10-fold cross-validation.

In the future, we will try different combinations of dimension reduction algorithms and learning techniques. We've tried PCA, which is a linear embedding for reducing the dimension of the dataset. We can try nonlinear embedding of the data set using Isomap or Laplacian Eigenmaps. Also, we can try more combinations of parameters for each learning techniques.

In particular, the models of CNN we used were relatively simple models compared to modern CNN architectures. Given the time, our future work we would like to implement some modern CNN architecture which may be able to push classification accuracy higher than 95 percent.

Another improvement we can help to explore is the preprocessing of our data. In our analysis, we disregarded the incidence angle which may have vastly influenced our result. We are aware that incidence angle can affect the intensity of the satellite decibel readings and we believe that the integration of this parameter is necessary to push the classification accuracy further.

In conclusion, we believe given the constraints we were successful in building a classifier which was able to classify icebergs and ships fairly accurately. We were able to score a Kaggle submission score of 0.2632 which placed us at 1675th out of 2378, beating more than half of the competing teams. Given the log-loss scoring method used in this competition, classification error can have an exponential impact on submission scores. Thus in our future work, we hope to reduce our classification error through various alternative data preprocessing methods, and the exploration of different CNN architecture and hyperparameters.

References

[1] Hian Chye Koh and Gerald Tan. Data mining applications in healthcare. *Journal of Healthcare Information Management*, 19(2):64–72, 2011.

[2] Alan Turing. On computable numbers, with an application to the entscheidungsproblem. *Proceedings of the London mathematical society*, 1937.

[3] Mr. Bayes and Mr. Price. An essay towards solving a problem in the doctrine of chances. by the late rev. mr. bayes, f. r. s. communicated by mr. price, in a letter to john canton, a. m. f. r. s. *Philosophical Transactions (1683-1775)*, 53:370–418, 1763.

[4] https://www.kaggle.com/c/spooky-author identification.

[5] https://www.kaggle.com/c/statoil-iceberg-classifier challenge.

[6] https://www.kaggle.com/host.

[7] https://www.kaggle.com/wiki/LogLoss.

[8] Xindong Wu, Vipin Kumar, Quinlan J. Ross, Joydeep Ghosh, Qiang Yang, Hiroshi Motoda, Geoffrey J. McLachlan, Angus Ng, Bing Liu, Philip S. Yu, Zhi Hua Zhou, Michael Steinbach, David J. Hand, and Dan Steinberg. *Top 10 algorithms in data mining*, volume 14. 2008.

[9] https://research.google.com/pubs/DataMiningandModeling.html.

[10] Yoav Goldberg and Omer Levy. word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method. *CoRR*, abs/1402.3722, 2014.

[11] Mo Yu and Mark Dredze. Improving lexical embeddings with semantic knowledge. In *ACL*, 2014.

[12] Andrew M Dai and Quoc V Le. Semi-supervised sequence learning. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 3079–3087. Curran Associates, Inc., 2015.

[13] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 649–657. Curran Associates, Inc., 2015.

[14] Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Weinberger. From word embeddings to document distances. In *International Conference on Machine Learning*, pages 957–966, 2015.

[15] Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. Recurrent neural network regularization. *CoRR*, abs/1409.2329, 2014.

[16] Stanislau Semeniuta, Aliaksei Severyn, and Erhardt Barth. Recurrent dropout without memory loss. *CoRR*, abs/1603.05118, 2016.

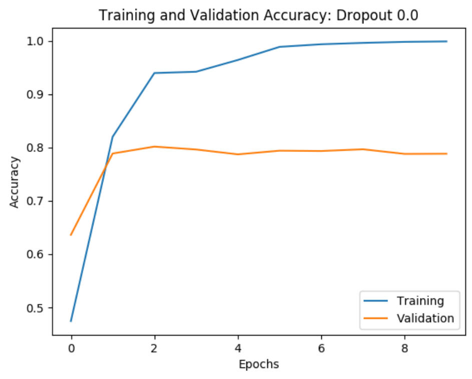
[17] Younes Samih, Suraj Maharjan, Mohammed Attia, Laura Kallmeyer, and Thamar Solorio. Multilingual code-switching identification via lstm recurrent neural networks. *EMNLP 2016*, page 50, 2016.

[18] Peter Velickovic. Deep learning for complete beginners: convolutional neural networks with keras.

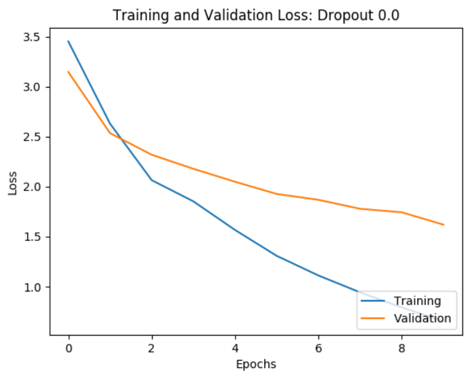
[19] Francois Chollet. Building powerful image classification models using very little data.

Appendices

1. Appendix A: Dropout Experimentations for Author Identification

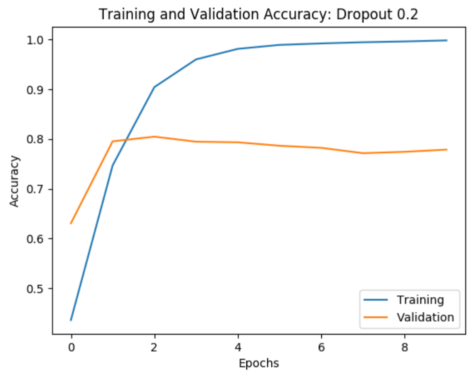


(a) Accuracy

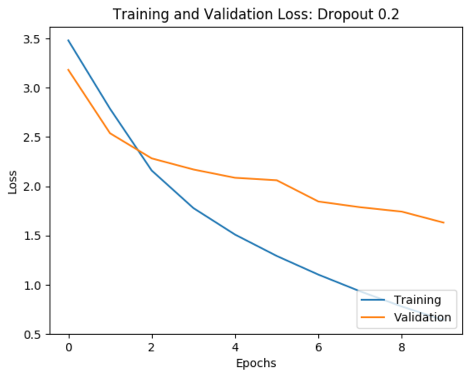


(b) Loss

Figure 12. Dropout Rate Experimentation: 0.0

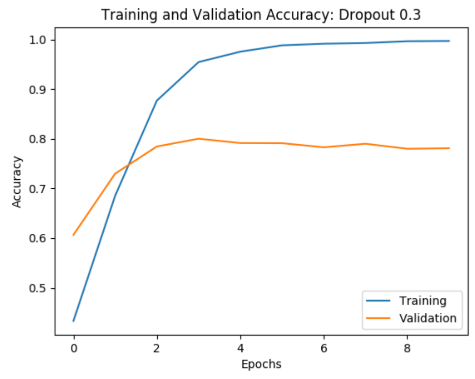


(a) Accuracy

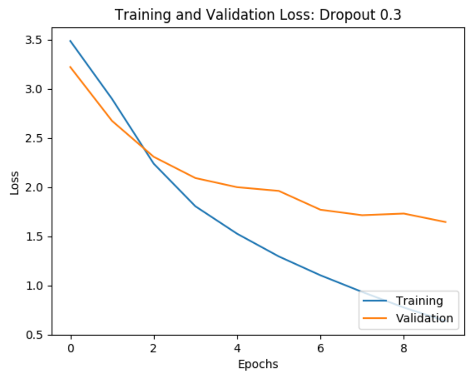


(b) Loss

Figure 13. Dropout Rate Experimentation: 0.2

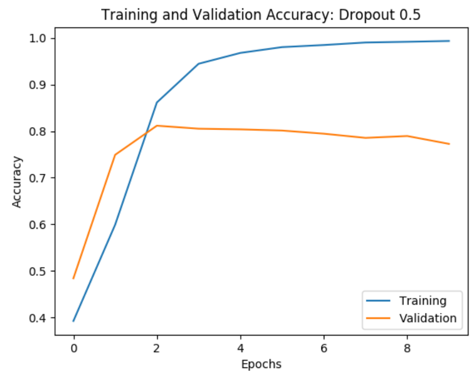


(a) Accuracy

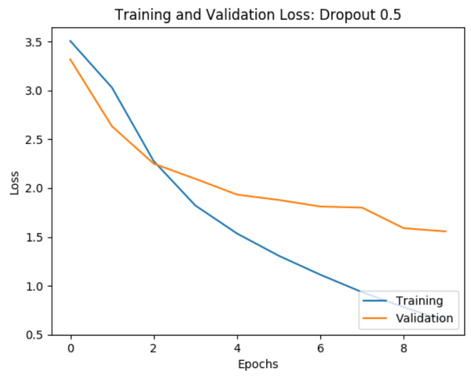


(b) Loss

Figure 14. Dropout Rate Experimentation: 0.3

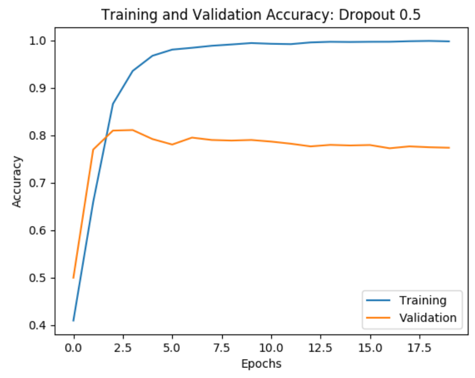


(a) Accuracy

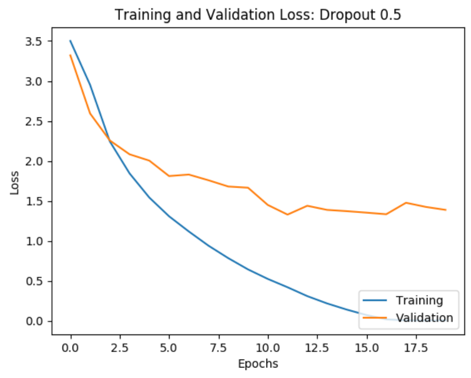


(b) Loss

Figure 15. Dropout Rate Experimentation: 0.5



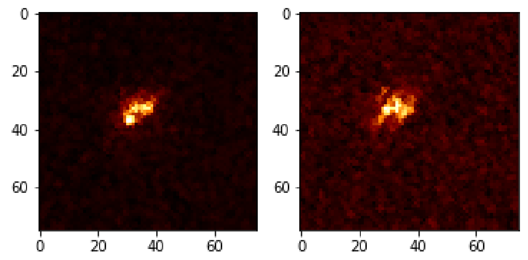
(a) Accuracy



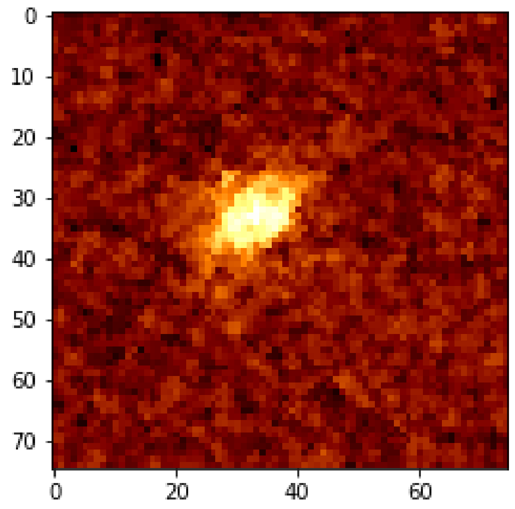
(b) Loss

Figure 16. Dropout Rate 0.5 with 20 Epochs

2. Appendix B: Models for Statoil/C-CORE Iceberg Classifier Challenge

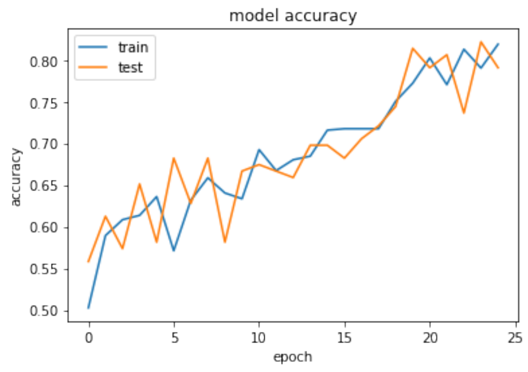


(a) Transformed Picture. Removed noise using power ratio scaling.

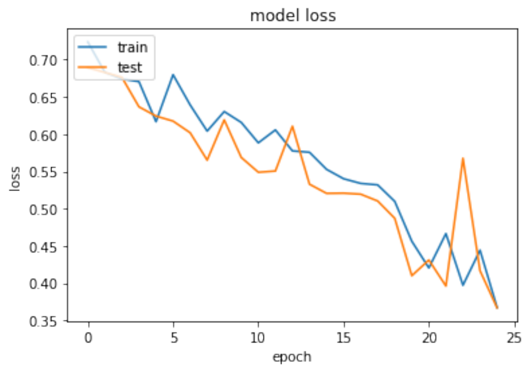


(b) Transformed Picture. Merged band into single channel.

Figure 17. Various Data Preprocessing Transformations. The following figures depict various transformation methods attempted which were unsuccessful in increasing model accuracy.

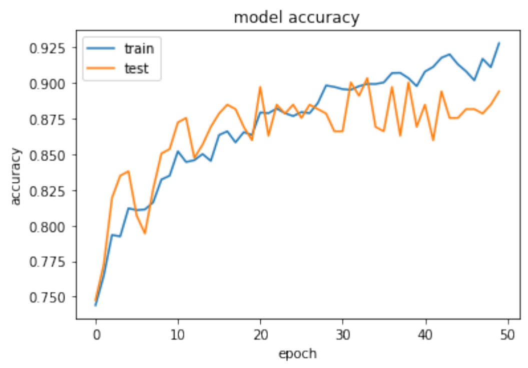


(a) Accuracy

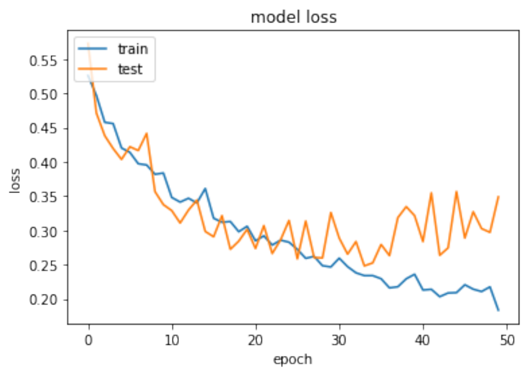


(b) Loss

Figure 18. (a) Model 1 Accuracy Graph. Y-axis is the model accuracy, X-axis is the epoch iteration. (b) Model 1 Loss Graph. Y-axis is the model loss, X-axis is the epoch iteration. Blue line indicates the model results on training dataset, Orange line indicates model results on testing set. Model was validated by 10 fold cross-validation.

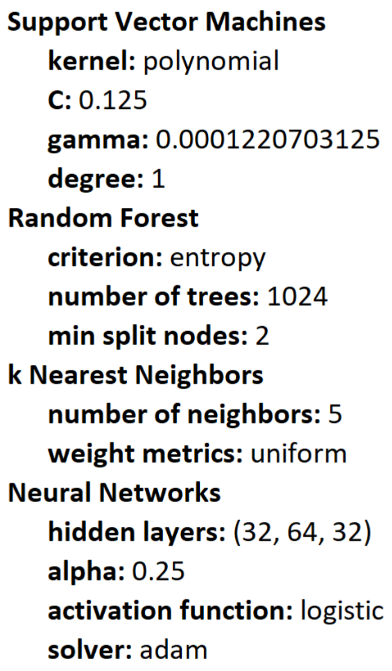


(a) Accuracy



(b) Loss

Figure 19. (a) Model 2 Accuracy Graph. Y-axis is the model accuracy, X-axis is the epoch iteration. (b) Model 2 Loss Graph. Y-axis is the model loss, X-axis is the epoch iteration. Blue line indicates the model results on training dataset, Orange line indicates model results on testing set. Model was validated by 10 fold cross-validation.



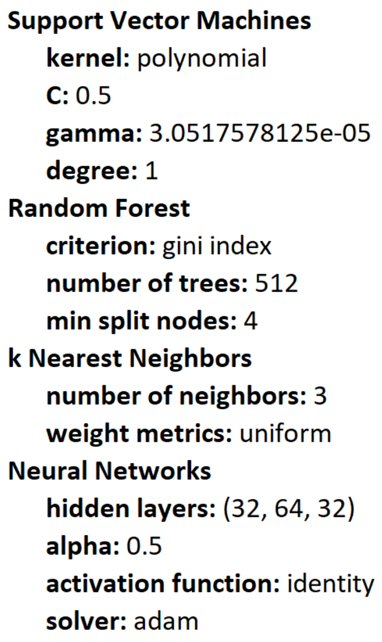
Support Vector Machines
- **kernel:** polynomial
- **C:** 0.125
- **gamma:** 0.0001220703125
- **degree:** 1

Random Forest
- **criterion:** entropy
- **number of trees:** 1024
- **min split nodes:** 2

k Nearest Neighbors
- **number of neighbors:** 5
- **weight metrics:** uniform

Neural Networks
- **hidden layers:** (32, 64, 32)
- **alpha:** 0.25
- **activation function:** logistic
- **solver:** adam

(a) Model1



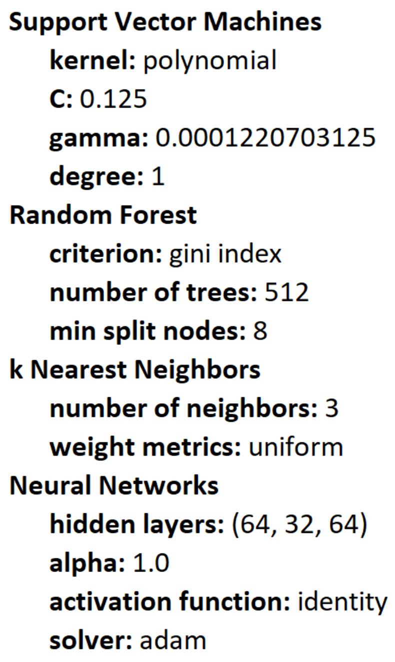
Support Vector Machines
- **kernel:** polynomial
- **C:** 0.5
- **gamma:** 3.0517578125e-05
- **degree:** 1

Random Forest
- **criterion:** gini index
- **number of trees:** 512
- **min split nodes:** 4

k Nearest Neighbors
- **number of neighbors:** 3
- **weight metrics:** uniform

Neural Networks
- **hidden layers:** (32, 64, 32)
- **alpha:** 0.5
- **activation function:** identity
- **solver:** adam

(b) Model2



Support Vector Machines
- **kernel:** polynomial
- **C:** 0.125
- **gamma:** 0.0001220703125
- **degree:** 1

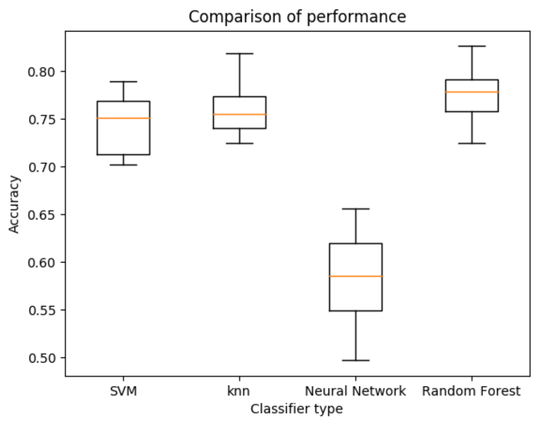
Random Forest
- **criterion:** gini index
- **number of trees:** 512
- **min split nodes:** 8

k Nearest Neighbors
- **number of neighbors:** 3
- **weight metrics:** uniform

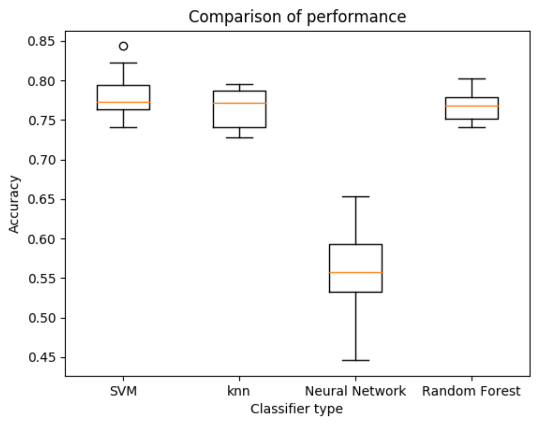
Neural Networks
- **hidden layers:** (64, 32, 64)
- **alpha:** 1.0
- **activation function:** identity
- **solver:** adam

(c) Model3

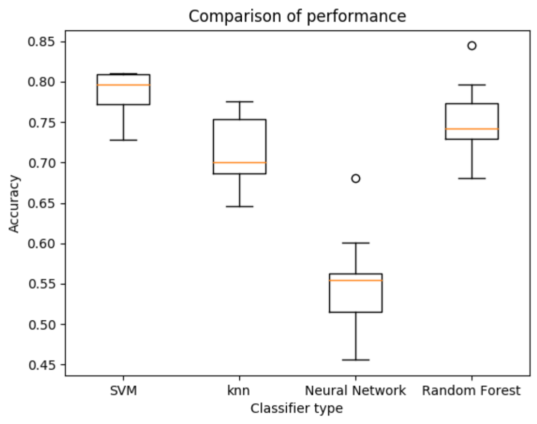
Figure 20. SVM, Random Forest, KNN, NN Model Architectures on three differently preprocessed data sets: (a) feature `inc_angle` removed, (b) rows with `inc_angle` removed, (c) two bands are merged together, and rows with na in the feature `inc_angle` are removed



(a) Model1



(b) Model2



(c) Model3

Figure 21. Comparison of the models on three differently preprocessed data sets: (a) feature `inc_angle` removed, (b) rows with `inc_angle` removed, (c) two bands are merged together, and rows with na in the feature `inc_angle` are removed