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

Tony Ha[1], Ronak Shah[2], Vatsal Jatakia[3]

Abstract
Kaggle is a platform for aspiring data scientists to hone their skills and create predictive models for companies to solve their problems. It is a community of statisticians and data scientists who come togther to solve the given problem and don't just restrict themselves to solving the problem but also share their solutions with others and even participate in discussions to help the others. We have attempted to become a part of this community by participating in the 'Spooky Author Identification' where we have to identify the author given a text from one of the works of that particular author and the 'Statoil/C-CORE Iceberg Classifier Challenge' where we are given an image in the form of band signals and we need to create a model to identify it as a 'ship' or an 'iceberg'. We have tried to use both, the conventional approaches like SVMs and Logistic Regression as well as relatively new algorithms like Convoluted Neural Networks and Artificial Neural Networks in order to see how the performance varies and understand the procedure of implementing these algorithms and also understand what kind of preprocessing each of these approaches require.

Keywords
Keyword1 — Keyword2 — Keyword3

[1] *Computer 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] *Computer Science, School of Informatics, Computing, and Engineering, Indiana University, Bloomington, IN, USA*
***Contact**: Tony Ha (juhha@indiana.edu), Ronak Shah (ronashah@indiana.edu), Vatsal Jatakia(vjatakia@iu.edu)

Contents

1. Introduction

Kaggle is a platform for predictive modeling and analytics competitions in which statisticians and data miners compete to produce the best models for predicting and describing the datasets uploaded by companies and users. This crowdsourcing approach relies on the fact that there are countless strategies that can be applied to any predictive modelling task and it is impossible to know beforehand which technique or analyst will be most effective[10] The problems are presented by companies in the form of competitions where statisticians and data scientists compete with each other to create the best model for solving the problem. It is a great starting line for budding data scientists as the various kernels provide a perspective to approach the problem and other experienced data scientists share their solutions and analysis which provides a vision for attacking the problem and also widens the vision of the various new algorithms and methods available for solving the problem compared as against the conventional methods used since a long time.

Data mining is the process of analyzing large amounts of data in an effort to find correlations, patterns, and insights.These insights that we get from the data, can be used to make further predictions, solve real life problems, get business insights. Kaggle follows a crowdsourcing approach which relies on the fact that there are countless strategies that can be applied to any predictive modeling task and it is impossible to know beforehand which technique or analyst will be most effective. Since, the purpose of the Kaggle competition is to get the best model that makes the most accurate prediction, two things play a major role in this scenario: domain knowledge and preprocessing. If we have the necessary domain knowledge, we would then be able to pre-process the data into a simpler form without affecting the correctness or the meaning of the actual data, hence Data Mining plays a major role here.

1.1 Author Identification

The Author Identification is a classification problem where the competition dataset contains text from works of fiction in horror by authors of the public domain: Edgar Allan Poe, HP Lovecraft and Mary Shelley. The objective is to accurately identify the author of the sentences from the given text in the test set. The dataset contains 19575 instances of 3 attributes- id, text, author. The id is a unique identifier for each sentence; the text is a set of sentences written by one of the authors and author field contains the name of the author who wrote that text. The prediction consisted of the probability of every sentence in the test data to be in each of the 3 classes of the authors, which in turn quantifies the goodness of the model.

1.2 Statoi/C-CORE Iceberg Classifier Challenge

The Iceberg classifier is a classification problem where the competition dataset contains the iceberg and the ship images data taken by satellite and numerical data for angle of projection taken for image. The objective is to accurately identify the iceberg from a ship. The dataset contains 3 input variables and a binary class variable:
band1: This has 5625 elements, which are the flattened array for 75 x 75 pixels, as numbers and each number corresponds to polarization by HH channel(transmit/receive horizontally)
band2: This also has same size of pixels as numbers and the number corresponds to polarization by HV channel (transmit horizontally and receive vertically).
Inc_angle: The angle represents the angle of projection from satellite to the object.
Is_iceberg: 0 for ship and 1 for iceberg.

The prediction consists of the probability of every image in the test data of being an iceberg, which in turn quantifies the goodness of the model.

2. Data Mining

Data mining is a field of intersection of computer science and statistics used to discover patterns in the information bank. The main aim of the data mining process is to extract the useful information from the dossier of data and mold it into an understandable structure for future use. There are different process and techniques used to carry out data mining successfully. Data mining uses artificial intelligence techniques, neural networks, and advanced statistical tools (such as cluster analysis) to reveal trends, patterns, and relationships, which might otherwise have remained undetected.

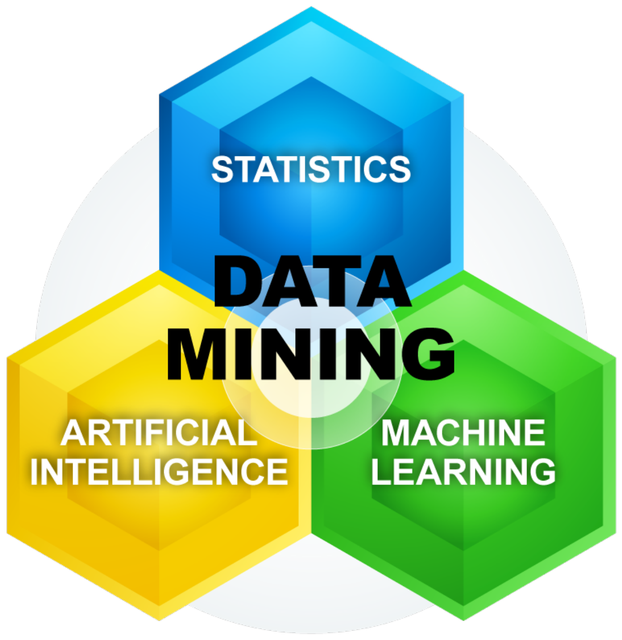


Figure 1. Data Mining as an intersection of 3 fields

Consider a mountain of data (rocks) and using data mining yields the nuggets of insight (gold).It does not give us a solution to the problem at hand, but it helps in getting insights about the data and understanding it better which would help towards getting a solution. It gives patterns and knowledge from large amounts of data, that can be used to determine future insights, make predictions etc.

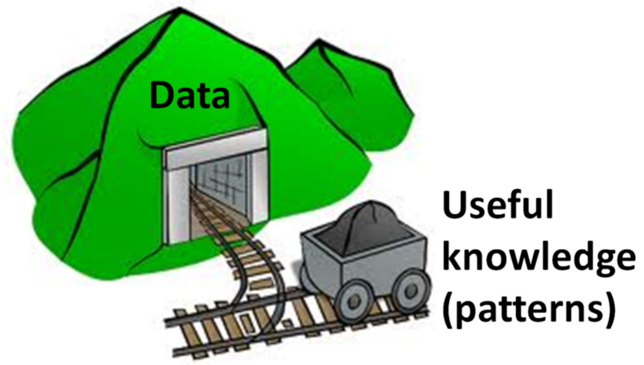


Figure 2. Yields of Data Mining

Below are the 8 steps in the data mining process:

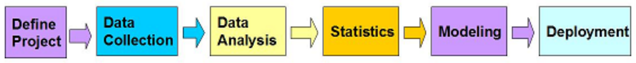


Figure 3. Steps in Data Mining

1.Defining the problem
This defines the objective of the whole data mining process. This is usually considered the most difficult step of the entire process since a good domain knowledge is required for defining the problem correctly and fully.

2. Collecting the data
There are various ways of collecting the data like documents, past records or even sensors. Ideally, the data should be enough in both depth and width such that the problem can be solved and all the variations be captured. But this is very difficult to achieve and hence we should have atleast enough data that could help us answer the general cases of the test data. [1]

3. Preparing the data
When the data to be processed is huge, it is very difficult and time-consuming to load the data every time we are training. Hence, a good practice is to let the entire data reside on a database and just connect the program to this database through which you only need to load the data once and then you can freely access this data anytime.

4.Pre-processing
The raw data that you will get can never be used directly. It may contain missing data or system or human errors that need to be addressed before the data can be processed. Also, sometimes the form of data that we have needs to be converted in a different form which the model can understand better or even just accelerate the process. Hence, pre-processing the data is a very crucial step of data mining on which the entire prediction depends.

5.Selecting of Model and parameters
We need to select the model considering the type of data that we have. Some algorithms work better when the data is numeric while some work better with continuous data, while some provide the flexibility of working great with either of those. Hence, the selection of algorithm and parameters is dependent entirely on the type of data we have.

6. Training and testing
The model selected in the previous step is trained using the training data available and then it is tested using the test data.

7. Iterating to produce different models
One model is never enough to conclude that the results you have obtained are optimal. Hence, we need to try various models in order get the best model which could be used for the test data.

8. Evaluating the final model:
The best model is selected based on the estimated accuracy. This model is then used for future predictions. [2]

Every machine learning problem can be classified into 2 broad categories: Classification and Clustering. Classification is a supervised learning technique used to assign per-defined tag to instance on the basis of features. So classification algorithm requires training data. Classification model is created from training data, then classification model is used to classify new instances. Clustering is an unsupervised technique used to group similar instances on the basis of features. Clustering does not require training data. Clustering does not assign per-defined label to each and every group. It is mainly used to find some patterns in the data during preprocessing of data. After training the model, we need to estimate how good it is, which is why we use the loss function. The loss function is how you're penalizing your output. Loss functions for classification are computationally feasible loss functions representing the price paid for inaccuracy of predictions in classification problems. Loss functions determine how poorly the model did and in the context of backpropagation and neural networks, they also determine the gradients from the final layer to be propagated so the model can learn. For example, the Author Identification problem uses the log loss function, which is given as:

$$\text{logloss} = \frac{-1}{N}\sum_{i=1}^{N} y_{ij} log(p_{ij})$$

where N is the number of observations in the test set, M is the number of class labels (3 classes), log is the natural logarithm, y_{ij} is 1 if observation i belongs to class j and 0 otherwise, and p_{ij} is the predicted probability that observation i belongs to

[1]depth here corresponds to number of instances and width corresponds to the number of columns

[2]Cross Validation is one such process where we break the train data into k parts and each of these k parts acts as the test set while the remaining parts are used for training and the best model(model which gives the least error) is selected.

class j. S

2.1 Data Preprocessing

Data Preprocessing as explained in the previous secton, is a very crucial step of Data Mining. It is further broken down into the following steps: 1. Data cleaning : In this step, the raw data is cleaned to get the clean data that can be used for further processing. Errors like missing values, errors and other noises are removed.

Challenges: Lacking attribute values, lacking certain attributes of interest, missing values. We can handle this by smoothing noisy data, identify or remove outliers, and resolve inconsistencies.

2. Data integration : Data from various sources is combined so that the user gets a unified view.

Challenges: The data is present in multiple databases, data cubes, or files. We need to integrate them, so that we can perform further steps on our data

3. Data transformation: In this process, we standardize the data so that any user can comprehend it. It is usually converted from the format of the source system to that of the destination system. For example, if the data contains the date as DD/MM/YYYY at the source, but the destination expects it as MM/DD/YYYY then it is transformed into the required format.

Challenges: The variables in data are in different units. We need to use methods such as normalization and aggregation.

4. Data reduction: Data can be reduced using various techniques like Principle Component Analysis, where we only consider a part of the data that is contributing the most to the variance.

Challenges: There are many variables in data, that does not help much in classifying the target variable. Methods such as Principal component analysis can be applied for dimensionality reduction.

5. Data discretization: This is an optional step in Data preprocessing where the continuous data is converted into factors by binning, histograms, etc.

Challenges: Data required by our model requires nominal attributes. Methods such binning can be used for replacing numerical attributes with nominal ones.

General load for time: Data pre-processing takes maximum amount of time of around 80% to 85% of the total time of a Data Scientist. This also depends on the type of data and different preprocessing steps applied on the data. For example, if we have large amount of missing values in our data, then it will be time consuming to handle them, data with many variables may take more time as we need to analyze various combinations in order to find a good correlation.

General load for space: This again depends on the type of data that we have. Generally, we reduce the size of the original data, as not all the variables are necessary for building our model and in case of time series data, if we do not want to look at the data at a granular level (say minutes), then we can aggregate them(say into a day) and then use that data for our analysis. Thus, we reduced the size of our data, but again this depends on the type of data that we have.

2.2 Mining, Interpretation, and Action

Top 10 Algorithms:

1. **C4.5:**

   A suite of algorithms for classification problems in machine learning and data mining.
   Type: Decision tree and ruleset classification algorithm.
   Works best with: Both continuous and discrete valued data attributes.
   Given a set S of cases, C4.5 first grows an initial tree using the divide-and-conquer algorithm by using two heuristic criteria to rank possible tests for split: information gain and the default gain ratio.
   Ruleset Classifiers: Uses a list of rules of the form "if A and B and C and ... then class X", where rules for each class are grouped together. A case is classified by finding the first rule whose conditions are satisfied by the case; if no rule is satisfied, the case is assigned to a default class.
   Disadvantages of C4.5: Requires high CPU time and memory.

2. ***k*-Means:**

   Simple iterative method to partition a given dataset into a user specified number of clusters, k.
   Type: Iterative clustering algorithm.
   Algorithm Steps:

   1. Decide the number of clusters K

   2. Randomly assign each data point to a cluster

   3. Compute cluster centroids

   4. Re-assign each point to the closest cluster centroid

   5. Re-compute cluster centroids.

   6. Repeat steps 4 and 5 until no improvements are possible

   Termination: When there will be no further switching of data points between two clusters for two successive repeats, it will mark the termination of the algorithm if not explicitly mentioned.
   Limitations: Difficult to predict the number of clusters and rescaling our datasets (normalizing or standardizing) will completely change results.

3. **Support vector machines:**

   Support Vector Machine (SVM) is primarily a classifier method that performs classification tasks by constructing hyperplanes in a multidimensional space that separates cases of different class labels. SVM supports both regression and classification tasks and can handle multiple continuous and categorical variables. For categorical variables a dummy variable is created with case values as either 0 or 1. Thus, a categorical dependent variable consisting of three levels, say (A, B, C), is represented by a set of three dummy variables: A: 1 0 0, B: 0 1 0, C: 0 0 1. To construct an optimal hyperplane, SVM employs an iterative training algorithm, which is used to minimize an error function.

Disadvantages: Difficulty in choosing the values of parameters in SVM and difficulty in choosing the best kernel function in SVM and very much time consuming.

4. **Apriori algorithm:**

   Apriori is an algorithm for frequent item set mining and association rule learning over transactional databases. It proceeds by identifying the frequent individual items in the database and extending them to larger and larger item sets as long as those item sets appear sufficiently often in the database. The frequent item sets determined by Apriori can be used to determine association rules which highlight general trends in the database. This has applications in domains such as market basket analysis. Disadvantages: Assumes transaction database is memory resident and requires many database scans

5. **EM algorithm:** The EM is a coordinate descent algorithm in which there are two sets of variables. The first is the parameters of the model. The second are the distribution of “natural” latent variables. So, for phase 1, you optimize the parameters assuming the latent variables distribution is fixed. for phase 2, you optimize the latent variables distribution’s likelihood assuming parameters are fixed.

   Disadvantages: Slow convergence and inability to provide estimation to the asymptotic variance-covariance matrix of the maximum likelihood estimator (MLE).

6. **PageRank** PageRank is a link analysis algorithm and it assigns a numerical weighting to each element of a hyperlinked set of documents, such as the World Wide Web, with the purpose of ”measuring” its relative importance within the set. The algorithm may be applied to any collection of entities with reciprocal quotations and references. The numerical weight that it assigns to any given element E is referred to as the PageRank of E and denoted by PR(E). Other factors like Author Rank can contribute to the importance of an entity.

   Disadvantages: PageRank scores are not calculated at the time of search – this would be too expensive and slow. PageRank cannot handle queries containing natural language and information outside of keywords.

7. **Adaboost** AdaBoost is a popular boosting technique which helps you combine multiple “weak classifiers” into a single “strong classifier”. A weak classifier is simply a classifier that performs poorly, but performs better than random guessing. AdaBoost can be applied to any classification algorithm, so it’s really a technique that builds on top of other classifiers as opposed to being a classifier itself. Disadvantages: AdaBoost can be sensitive to noisy data and outliers.

8. ***k*-NN: k-nearest neighbor classification** The k-NN algorithm assumes that an unclassified item can be classified by looking at k of its already-classified, nearest neighbors and finding out which class the largest number of them fall into. The reason for choosing the nearest neighbors to the unknown item is that those are the items which are most similar to it in the attributes used to map the items, so the chances are higher that it belongs to the same class.

9. **Naive Bayes** It is a classification technique based on Bayes’ Theorem with an assumption of independence among predictors. Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature. Naive Bayes model is easy to build and particularly useful for very large data sets. Along with simplicity, Naive Bayes is known to outperform even highly sophisticated classification methods.

   Disadvantages: It assumes every feature is independent, which isn’t always the case.

10. **CART** The CART decision tree is a binary recursive partitioning procedure capable of processing continuous and nominal attributes both as targets and predictors. Data are handled in their raw form; no binning is required or recommended. Trees are grown to a maximal size without the use of a stopping rule and then pruned back (essentially split by split) to the root via cost-complexity pruning.

Data mining can be considered like mining a mountain of rocks for gold.It does not give us a solution, but it gives us insights about the data and helps us understand it better so that we can work towards getting a solution. It gives patterns and knowledge from large amounts of data, that can be used to determine future insights, make predictions etc. With the evolution of technology, there has been a new set of problems that we counter in the field of data science:

1. Efficiency and scalability of data mining algorithms: To effectively extract information from a huge amount of data in databases, data mining algorithms must be efficient and scalable.

2.Parallel, distributed, and incremental mining algorithms: The huge size of many databases, the wide distribution of data, and the computational complexity of some data mining methods are factors motivating the development of algorithms that divide data into partitions that can be processed in parallel.

3.Issues relating to the diversity of database types: Handling of relational and complex types of data: Specific data mining systems should be constructed for mining specific kinds of data. Mining information from heterogeneous databases and global information systems: Local- and wide-area computer networks (such as the Internet) connect many sources of data, forming huge, distributed, and heterogeneous databases.

These are some of the problems that the researchers are working on solving.

3. Author Identification: Full Problem Description

- The Author Identification is a classification problem where the competition dataset contains text from works of fiction in horror by authors of the public domain: Edgar Allan Poe, HP Lovecraft and Mary Shelley. The objective is to accurately identify the author of the sentences in the test set.
- The data was prepared by chunking larger texts into sentences using CoreNLP's MaxEnt sentence tokenizer. The input is a dataset containing sentences from the works of the above mentioned authors. The output expected is basically a set of three probabilities of each sentence in the test sentence belonging to the work of the above mentioned authors. We have tried various text analysis methods like Term Frequency-Inverse Document Frequency (TF-IDF), Document-Term Matrix and even Sentiment Analysis. For purposes of training we tried a wide range of methods ranging from conventional methods like Naive Bayes, Logistic Regression to advanced techniques like Support Vector Machines and Artificial Neural Networks. The accuracy is used as a measure of evaluation of the model.

The dataset contains 19575 instances of 3 attributes- id, text, author. The id is a unique identifier for each sentence; the text is a set of sentences written by one of the authors and author field contains the name of the author who wrote that particular text. The entire process from understanding the data to cleaning and then pre-processing is described in detail below.

3.1 Data Analysis

- Describe the data in full detail–from its raw form to the transformation
- Provide summary statistics and relationships

The dataset contains 19575 rows of 3 attributes- id, text and author. The data in its raw form was "dirty" in the sense that it had missing punctuation marks, spelling errors and even misplaced words, hence the data needed to be cleaned before processing it.

3.1.1 Data Cleaning

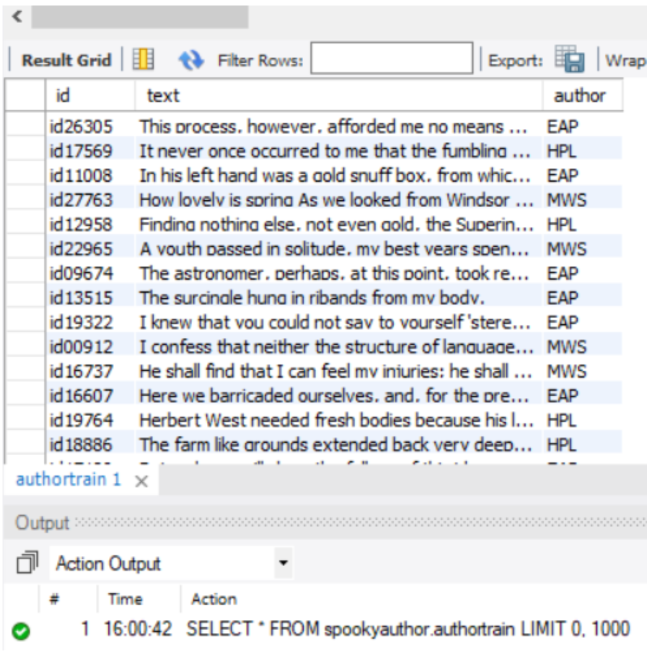
For cleaning the data, we made use of the *tm* package in R. We first create a 'corpus' which is basically a vector containing all the sentences. Using the *tm_map*(), we clean the corpus by

- removing all the punctuation marks
- converting the sentences into lower case
- removing the spaces between words
- removing the numbers
- removing stop words which are very commonly used in any kind of text and hence won't contribute much to the variance of the data

Thus, the data obtained at the end of this process is clean which can now be used for further processing.

3.1.2 Data Pre-processing

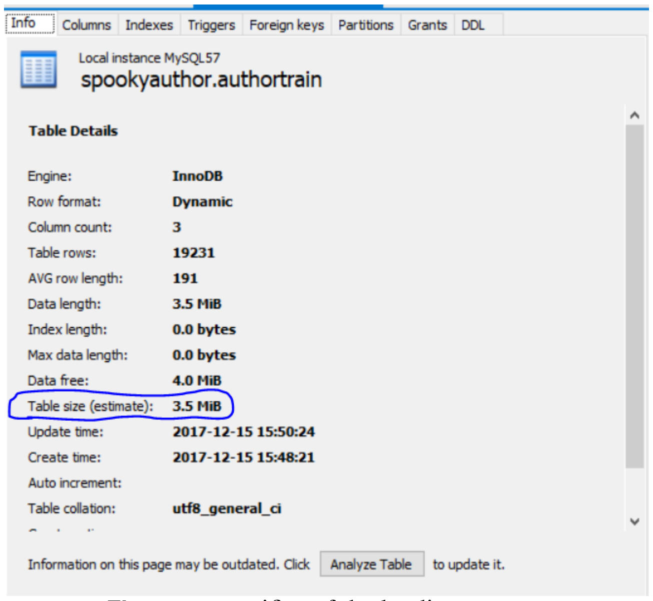
The data was first put into a database so that we get independence from loading the data everytime.



id	text	author
id26305	This process, however, afforded me no means ...	EAP
id17569	It never once occurred to me that the fumbling ...	HPL
id11008	In his left hand was a gold snuff box, from whic...	EAP
id27763	How lovely is spring As we looked from Windsor ...	MWS
id12958	Finding nothing else, not even gold, the Superin...	HPL
id22965	A youth passed in solitude, my best years spen...	MWS
id09674	The astronomer, perhaps, at this point, took re...	EAP
id13515	The surcingle hung in ribands from my body.	EAP
id19322	I knew that you could not say to yourself 'stere...	EAP
id00912	I confess that neither the structure of language...	MWS
id16737	He shall find that I can feel my injuries; he shall ...	MWS
id16607	Here we barricaded ourselves, and, for the pre...	EAP
id19764	Herbert West needed fresh bodies because his l...	HPL
id18886	The farm like grounds extended back very deep...	HPL

```
#   Time      Action
1   16:00:42  SELECT * FROM spookyauthor.authortrain LIMIT 0, 1000
```

Figure 4. Train data in the database



Local instance MySQL57
spookyauthor.authortrain

Table Details

Engine:	InnoDB
Row format:	Dynamic
Column count:	3
Table rows:	19231
AVG row length:	191
Data length:	3.5 MiB
Index length:	0.0 bytes
Max data length:	0.0 bytes
Data free:	4.0 MiB
Table size (estimate):	3.5 MiB
Update time:	2017-12-15 15:50:24
Create time:	2017-12-15 15:48:21
Auto increment:	
Table collation:	utf8_general_ci

Information on this page may be outdated. Click Analyze Table to update it.

Figure 5. specifics of the loading process

There are 3 basic pre-processing approaches that we have tried:

- **Document Term Matrix (DTM):** Document Term Marix as the name suggests is a matrix containing the rows equal to the number of instances and columns containing the words that are a remainder of the above cleaning process.Here, each instance is treated as a separate document. The row values are 1's and 0's where 1 denotes that the word is present in the document and 0 represents that it is not.
  We further reduce the columns in the matrix by removing the sparse terms in the matrix which are simply the terms that are occuring seldom compared to the other terms and hence do not contribute much to the variance. For removing the sparse terms, the $removeSparseTerms()$ method is used, wherein we can set the sparcity level ranging from 0 to 1. We have tested the models for 2 levels of sparsity i.e, 0.9 and 0.999 and concluded that the higher the sparsity, the better the results. At sparcity 0.9 we got a set of 735 words, while at sparcity of 0.999 we obtained a set of 2498 words. The various models were then applied on the DTM obtained which was converted into a binary dataset and saved as a .csv file for future use.

- **Term Frequency- Inverse Document Frequency (TF-IDF):** In simple terms, TF-IDF is a numerical statistic that reflects how important a word is to the document[1]. The value of TF-IDF increases proportionally to the number of times a word appears in a document, but it is often offset by the frequency of the word in the corpus.

  The $term frequency$ corresponds to the number of times a particular word appears in every document. It is mathematically formulated as:

  $tf(f,d) = 1 + log(f_{t,d})$ [2]

  where f is the frequency of term t in document d

  The $inverse\ document\ frequency$ is a measure of how much information the word provides, that is, whether the term is common or rare across all documents. It is the logarithmically scaled inverse fraction of the documents that contain the word, obtained by dividing the total number of documents by the number of documents containing the term, and then taking the logarithm of that quotient. Mathematically, idf is formulated as :

  $idf() = log\frac{N}{\{d \in D : t \in d\}}$ [2]

  where D is the set of documents

- **Sentiment Analysis and WordClouds:** The third technique for analysis we used was Sentiment Analysis using the 'rSentiment' package in R. The purpose of this analysis was to understand better the style of writing of each of these authors. The following plot gives the sentiments of the words used by the authors, what we observed was that Edgar Allen Poe and HP Lovecraft have a similar writing style except that HP Lovecraft uses a bit more negative words compared to Edgar while the number of positive words are more in case of Edgar which shows that Edgar is not aggressive in his writing approach as HP Lovecraft. Mary Shelly on the other hand uses much more negative words which shows the level of spookiness in her writing is much more than the other two authors.

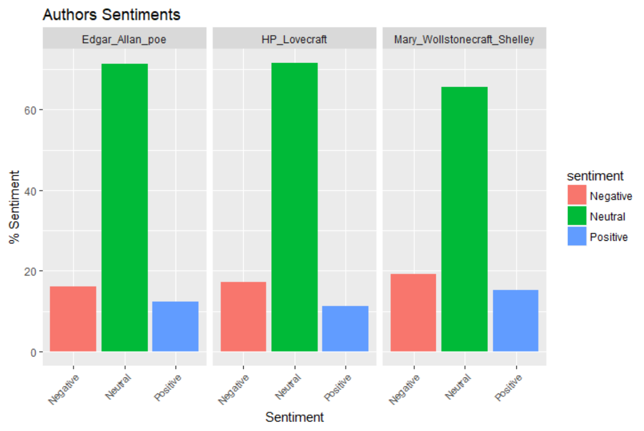


Figure 6. Sentiment Analysis

Extending this analysis further, the wordclouds below give a better perspective on the choice of words by the three authors under consideration. The wordclouds reaffirm the prior result about the sentiments of the author's writing styles.

The wordcloud for Edgar Allen shows much more frequent use of words in the Positive cloud with most frequent use of words like "well", "great", "found" followed by "good", "like" and "matter", while the Negative cloud only includes most frequently used words

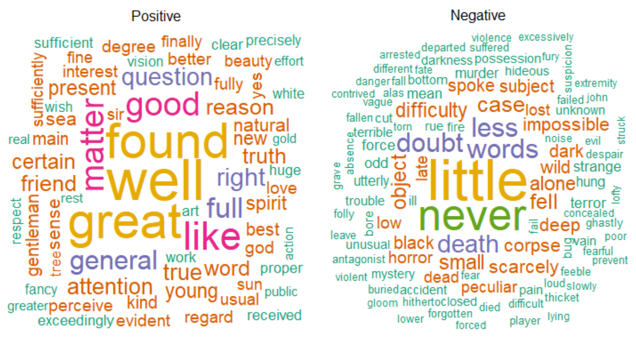


Figure 7. Wordcloud for Edgar Allen Poe

like ”little” and ”never” followed by ”less”, ”doubt” and ”words”. This aligns with the observation from the sentiment analysis from the plot above.

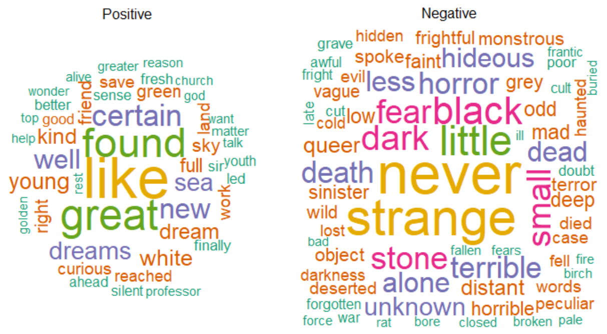


Figure 8. Wordcloud for HP Lovecraft

The wordcloud for HP Lovecraft suggests the use of more frequently used words in the negative cloud than the positive cloud which again turns out to be consistent with our observation earlier. The most frequently used words in the positive sense are ”like”, ”certain” and ”great” followed by ”new”, ”sea”, ”dreams”, ”well” and ”certain”. The most frequently used in the negative sense are ”strange”, ”never” followed by ”little”, ”dark”, ”black”, ”small”, ”fear” in the negative cloud.

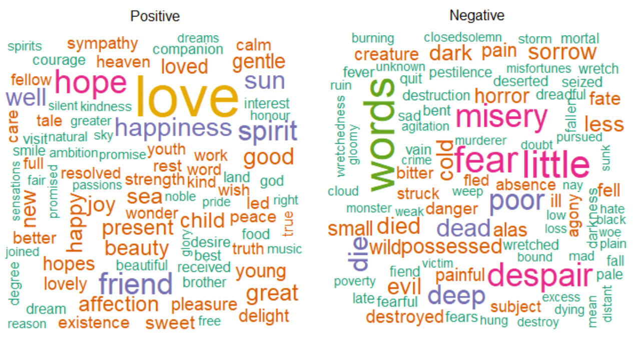


Figure 9. Wordcloud for Mary Wollstonecraft Shelly

The wordcloud for Mary Shelly is a bit difficult to interpret as she uses a wide range of words with relatively less repetitions compared to the other two authors. But it is still evident that the number of negative words by her are more frequent than the negative words. The most frequent positive words are ”hope” and ”love” followed by ”friend”, ”spirit” and ”happiness”. The most frequently used negative words are ”words”, ”fear”, ”little”, ”misery”, ”despair”, ”horror” followed by ”die”, ”deep”, ”dead”. Overall these words describe the aggressiveness of the author which again shows consistency with the observations from the sentiment analysis plot obtained in Fig 1.

3.2 Methods

- Discuss the algorithm you’ve chosen, *e.g.*, why you chose it
- Provide some background material on your method that shows you are well-acquainted with it
- Challenges to your method
- What software and hardware did you use, packages, *etc.*
- Final structure of data after preprocessing
- Present training and testing as some combination of text and visualizations

1. **Using TF-IDF:** After finding the TF-IDF indexes for each of the documents the training and testing dataset using the ’tidytext’ library in R, we considered the product of TF and IDF to get a single value that will be used for prediction. Once this dataset containing the product of TF and IDF was combined with the corresponding labels was then used for creating the classification models and used for further processing.

   (a) **Logistic Regression (LR):** Logistic regression is a statistical method for analyzing a dataset in which there are one or more independent variables that determine an outcome[3]. The outcome is measured with a dichotomous variable (in which there are only two possible outcomes) predicting

either 1/0, T/F, etc. [3]. The reason behind this dichotomous prediction is the use of sigmoid function which is mathematically represented as :
$y = \frac{1}{1+e^{-x}}$
where x is the training example.

The reason for using this algorithm is its speed. It can process a good amount of records in a very short duration of time which provides a good starting step for judging the correctness of preprocessing and the approach in general. The major part of this implementation was on the Macbook (intel i5, 8GB RAM)and 'glmnet' was used for implementing Logistic Regression and as mentioned before 'tidytext' was used for TF-IDF calculation. Due to the speed of the logistic regression, this was done using the personal computer itself. The final output after prediction was a set of 3 probabilities for each of the authors and it was coerced into a dataframe with tuple ids and saved as a .csv file so that it matches with the submission file that is expected by Kaggle.

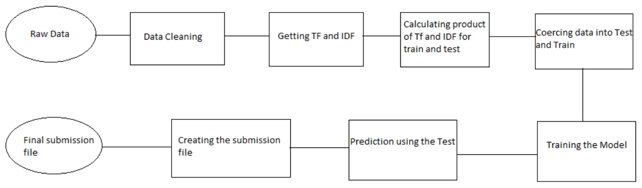

Raw Data
Data Cleaning
Getting TF and IDF
Calculating product of Tf and IDF for train and test
Coercing data into Test and Train
Training the Model
Prediction using the Test
Creating the submission file
Final submission file

Figure 10. Logistic Regression procedure

The above diagram gives an overview of the Logistic Regression procedure. We have first cleaned the data using the cleaning method explained in the earlier section and then the cleaned data is used to calculate the TF and IDF values for each document(text in the training and testing sets) which are multiplied to get a single value. This value is combined with the label to form a clean training set. The test data is also transformed in a similar fashion and turned into a dataframe. These data frames are then made available to the model for training. After the training, this model is then used for prediction and the final predictions are then coerced into a final dataframe along with the ids for each of the examples from the testing set. This dataframe is then saved as .csv file which forms the final submission for Kaggle.

(b) **Naive Bayes (NB)**
The main reason for using Naive Bayes is its speed and simplicity of training. There is no special optimization needed to train a Naive Bayes model and it can work well for both the cases ,i.e, when the assumption of Naive Bayes which is independence of the various predictors holds true as well as when it does not. The background of Naive Bayes has been described in detail in the following section.

The challenge here was again the vastness of the data and the training speed of the model, and it was making the RStudio crash, hence we had to use the Burrow Server by SOIC. This helped us achieve a good speed up. Naive Bayes for TF-IDF was again implemented on the Macbook Pro (i5, 8GB RAM) and the training was on the Burrow Server as mentioned above. The 'e1071' library was used for training the model and 'tidytext' for obtaining the Tf and IDF values. The final output after prediction was a set of 3 probabilities for each of the authors and it was coerced into a dataframe with tuple ids and saved as a .csv file so that it matches with the submission file that is expected by Kaggle.

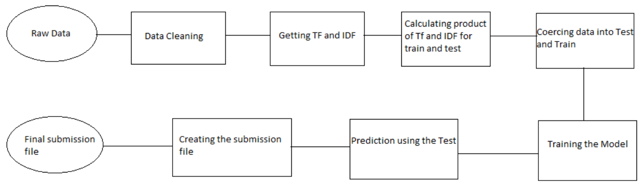

Raw Data
Data Cleaning
Getting TF and IDF
Calculating product of Tf and IDF for train and test
Coercing data into Test and Train
Training the Model
Prediction using the Test
Creating the submission file
Final submission file

Figure 11. Naive Bayes procedure

The diagram above gives an overview of the procedure followed right from processing the initial data to obtaining the final submission file. The raw data from the train and test first needed to be cleaned which was done using the cleaning process explained in the earlier section. Once the clean datasets was obtained, they were used to calculate TF and IDF values. After geting the TF and IDF values, the product was then calculated for each document and stored along with the label for each document. This was the new training set obtained which would be used for further processing. Similar preprocessing was required for the test set as well. After obtaining these transformed sets, the Naive Bayes classifier was trained using the 'e1071' package and this model was then used for testing. The final output was then used to create a submission file by appending the tuple ids and making a Kaggle.

2. **Using the DTM:** From the raw data, we obtained the DTM using the procedure described in section 3.1 and

then obtained a matrix consisting of binary data.Using this matrix we tried the following methods:

(a) **Naive Bayes (NB)**
Naive Bayes is a straightforward and powerful algorithm which is used for classification tasks[4]. It can be used for both categorical and continuous data. Naive Bayes classifier assumes that all the features are unrelated to each other. Presence or absence of a feature does not influence the presence or absence of any other feature[4]. It is based on the Bayes Rule which is mathematically represented as :

$P(A|B) = \frac{P(B|A)*P(A)}{P(B)}$
It basically calculates the probability of a test example being in all the possible target classes which is called the 'posterior probability', and the example finally ends up falling into the class with the highest probability.
There are various types of Naive Bayes Classifiers like Gaussian Naive Bayes, Multinomial Naive Bayes, Bernoulli Naive Bayes, etc.

The biggest challenge was the speed of processing the data due to the vastness of the data. Even at the sparsity level of 0.999, there were 2498 words to be processed, which took a while to run in RStudio. Hence, we tried to run our codes on Burrow server provided by the SOIC, which improved the speed to some extent. The second challenge was the estimation of accuracy of the model. The words in the DTM of the train and test were different, hence there was filtering needed to be done in order to get the same set of words for both the train and test sets. The data that we had contained a lot of errors like missing punctuation marks, missing words, misspelled words, wrongly placed words, wrongly placed punctuation marks, etc. Hence, there was an intensive cleaning of data required before it could be used further for processing.
The initial pre-processing was done on individual laptops namely Dell Inspiron i7559 (Intel i7-6700 HQ, 16GB RAM), Dell Inspiron 7000 (Intel i7-7700 HQ, 12GB RAM) and Apple Macbook(Intel i5, 8GB RAM). The training and testing was done on the Burrow Server.
The final output was a set of probabilities of the particular text being from the works from one of the three authors under consideration. The test ids were attached too form the final submission file as per the format described on Kaggle.com and written down as a .csv file which was used for submission.

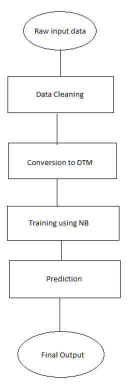


Figure 12. Naive Bayes procedure

The figure above gives a brief overview of the procedure followed from pre-processing the input data to getting the final output. The raw data from both the test and train sets was cleaned and converted to a DTM. The common terms were found in both the train and test DTMs. The final DTMs were then used in the Naive Bayes Model for training. After training the classifier, it was used for prediction which provided the final output that was transformed into a .csv submission file for Kaggle.

(b) **Support Vector Machine(SVM)**
Formally, a support vector machine constructs a hyperplane or set of hyperplanes in a high- or infinite-dimensional space, which can be used for classification, regression, or other tasks like outliers detection[5]. The SVM is another very powerful tool for classification and works on binary and numeric data. It can be extended to categorical attribute values by some preprocessing like one-hot encoding, etc. SVMs typically work with data that is linearly separable through hard margins, but it can also handle data that is not linearly separable using the soft margins. Mathematically, hard and soft margins can be represented as:
Hard-margin: $\vec{w}.\vec{x} - b = 0$[5]
Soft-margin: $\frac{1}{n}\sum_{i=1}^{n} max(0, 1 - y_i(\vec{w}.\vec{x} - b)) + \lambda||\vec{w}^2||$[6]
where λ term is the trade-off between increasing the margin-size and ensuring that the datapoint lies on the correct side of the margin[6]. One of the main reasons for selecting SVM was its ability to provide the best accuracy when the data is continuous or binary.

Like NB, the vastness and the processing speed were a great challenge. We tried to increase the speed by trying to reduce the number of rows that are recurring, but it was not a good idea as the overall accuracy was affected, hence we tested by keeping the same set of words. SVM takes a long time for training and in this case it was even slower due to the total number of columns and rows huge. Also, the words in the DTM of the train set and the test set were not the same, hence getting the same set of words turned out to be very complicated and was leading to a substantial loss in the variance, hence we did not make a submission on Kaggle using SVM, but only evaluated the model based in the overall accuracy.

The SVM was implemented using the 'e1071' package in R. The hardware used was the same that was used for the Naive Bayes. The following plot gives an overview of the approach we followed for obtaining the final output.

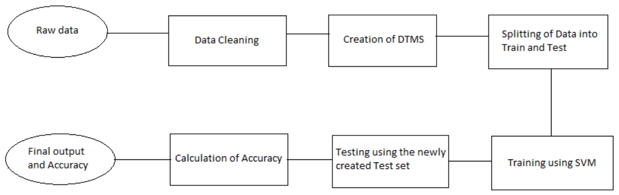

Raw data
Data Cleaning
Creation of DTMS
Splitting of Data into Train and Test
Training using SVM
Testing using the newly created Test set
Calculation of Accuracy
Final output and Accuracy

Figure 13. SVM procedure

The raw data was first taken and cleaned in a similar fashion as explained above for the Naive Bayes procedure. After cleaning, the DTM containing 2498 words was obtained. The no. of rows was the same as the train data which is 19575 rows. This was then split into 2 parts for evaluation of the model. The first 10,000 points were considered as the training set and the remaining 9575 points were put into the testing set. The SVM model was then trained using the newly created train set. After over 3 hours of training, the test set was used for testing this model and the new labels were decided based on the 'author' label having the highest probability. The new labels were then compared to the true labels to obtain the accuracy which was simply the ratio of the matching labels to the total length of the no. of test rows.

3. **Artificial Neural Networks (ANN)**

The idea of ANNs is based on the belief that working of human brain by making the right connections, can be imitated using silicon and wires as living neurons and dendrites[7].ANNs are composed of multiple nodes, which imitate biological neurons of human brain[7]. The neurons are connected by links and they interact with each other[7]. The nodes can take input data and perform simple operations on the data[7]. The result of these operations is passed to other neurons[7]. The output at each node is called its activation or node value. Each link is associated with weight[7]. ANNs are capable of learning, which takes place by altering weight values[7].

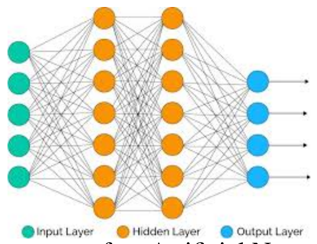


Figure 14. Structure of an Artificial Neural Network[8]

For the purposes of this problem, we make use of the FastText library which is a library created by the Facebook Research Team for efficient learning of word representations and sentence classification. FastText contains only 3 layers:
1. Input Layer: The input consists of the documents[9]
2. Average Pooling Layer: The word features are averaged to form good sentence representations.[9] 3. Hierarchical Softmax layer: The softmax function is often used in the final layer of a neural network-based classifier to improve the computational complexity[9].

We have done this implementation in Python using the 'keras' library. The dimensions of the word vector is 20 and the optimizer used is 'adam'. Inputs are words and bi-grams (as suggested by Marcin)from the document. The training was completed in 5 minutes, which was the least time required among the various approaches we used. Due to the great speed, noGPU servers were required and the implementation was done successfully on the CPU of the computer itself. The main challenge was figuring out the implementation of the algorithm, but other than the speed and the vastness of the dataset was taken care of by FastText itself. The final output was again a set of probabilities for each author and the tuple ids were appended to the prediction and saved as .csv as per the requirements of Kaggle.

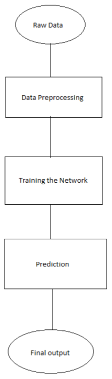


Figure 15. FastText procedure

The process starts with the cleaning of the raw data. For the purposes of cleaning, we have removed the words that are less frequent, cut down long sentences, removing the stopwords and converting the sentence to lower case. After obtaining the clean datasets(cleaning applied to both train and test), the data is made available to the first layer of the network. After training, the model is now used for prediction which is then saved to a file as per the requirements of Kaggle.

3.3 Results

- Dispassionately describe your results both quantified and qualified
- Do you deem this successful
- What do the results suggest
- What were challenges

Using the above approaches, we have first split the training dataset into test and train in order to validate the model. The data was divided in the ratio of 70% and 30% testing. After training the model on 70% of the training data we tested on the remaining 30% and calculated the accuracy.
For the test data, the author for which the highest probability was predicted was assigned as the label. The accuracy was simply the ratio of the labels that match the actual labels to the total number of instances in the test data. Once, we got a good accuracy on this data, we then trained the model on the entire train data and then used this model on the actual test data. The following table summarizes the results for each of the approaches.

Preprocessing: TF-IDF

Algorithm	Accuracy	Kaggle Score
Logistic Regression	0.81	0.35
Naive Bayes	0.792	0.52

Preprocessing: DTM

Algorithm	Accuracy	Kaggle Score
SVM	0.723	N/A
Naive Bayes	0.68	1.4
ANN	0.85	0.74

The reason for writing N/A in the Kaggle score for SVM is that the DTM generated for the test and train were quite different and contained only a few common words which I feel does not define the actual dataset, hence we just found the accuracy for the model. From the above considered approaches, the best accuracy was provided by ANN, but it may have been overfitting since the score that we got was 1.08 which is very less. The best score we achieved was 0.52 which was using Logistic Regression.

Considering our inexperience in this field, we got to learn a lot about the various algorithms, their working, their implementation and the results we get from these algorithms. We got a closer look at the various libraries available in Python and R and how they work in order to give us the predicted results. Hence, we believe that we were successful qualitatively, but not quantitatively as the scores we obtained weren't the best and there is great scope for improvement considering the various other advanced techniques available for solving domain-specific problems.

3.4 Summary and Future Work

- Briefly summarize project and outcome
- What would you do differently in the future?

The Spooky Author was a classification problem, but it was different in terms of the result expected. Instead of the usual binary classification, here we had to predict the probability of a text belonging to any of the three authors with some likelihood. We tried a variety of methods ranging from the traditional Naive Bayes, Logistic Regression and SVM to the advanced ANN algorithm. We tried various preprocessing techniques like DTM and TF-IDF as well which gave us insights into how the textual data can be converted to numeric data for processing. The best result we got was using Logistic Regression on the data that was preprocessed using TF-IDF. The other approach we used was DTM where we faced processing time as a great challenge. We also tried hands on Sentiment Analysis which provided us with better understanding of the data, but were not able to implement it for making a submission.

Hence, the future work would comprise of firstly getting more domain knowledge, in terms of how the sentences are framed and what is the thought process. Once we understand the thought process, we can construct ways to breakdown sentences for better understanding of the data. We would like to get a better understanding of Sentiment Analysis and use it to obtain concrete data that can be used for training a model. We would like to improve our ANN model to obtain a better score. We would also like to explore Deep Learning techniques that would be suitable for the problem. This leaves us with much more options to explore and get a better model that would help us get a good ranking on the Kaggle Leaderboard.

4. Iceberg: Full Problem Description

Problem Description (Input and Output)

The main problem of iceberg project is to differentiate between the iceberg and the boat given satellite image data and numerical data for angle of projection taken for image. Here are the actual images[10] of iceberg and boat.

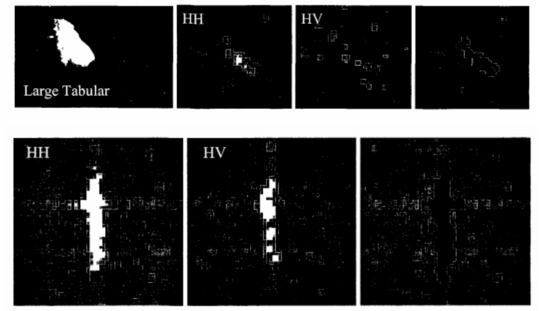


Figure 16. Iceberg (above) and boat (below) satellite image

To differentiate between the iceberg and the boat, the competition holder provides 3 variables (inputs):

- *band1*: list-type data. This has 5625 elements, which are the flattened array for 75 x 75 pixels, as numbers and each number corresponds to polarization by HH channel (transmit/receive horizontally)
- *band2*: list-type data. This also has same size of pixels as numbers and the number corresponds to polarization by HV channel (transmit horizontally and receive vertically).
- *inc_angle*: numerical data. The angle represents the angle of projection from satellite to the object (iceberg or boat).

Unlike normal pixel values, which has 8-bit integer to represent the intensity of color, the pixel values for *band1* and *band2* can be negative and even float type. This is because the number values they represent is the intensity of energy in *dB*.

Other than variables as inputs, The competition holder also provides target label:

- *is_iceberg*: binary value (0 or 1). 0 for boat and 1 for iceberg.

However, the output of our solution is not binary values. Instead, our output is the decimal numbers as percentage representing the probability of begin iceberg. Here is what our output:

- *prob* (p):p is the probability of the label is iceberg and $1-p$ is the probability of the label is boat.

Training and Testing

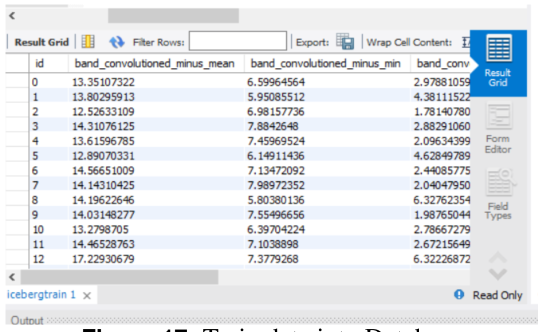
For our prediction models, we use logistic regression, support vector machines (SVM), and convolutional neural network (CNN). And for goodness function to check the quality of our prediction, we use the log-loss function. Here is the log-loss function we use:

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

Note that y_{ij} is 1 when the observed label i is the iceberg and 0 when it is boat. The reason why we use this function instead of simple percentage accuracy is because we want to see how many times our programs predicts right, but we want to see how well it does.

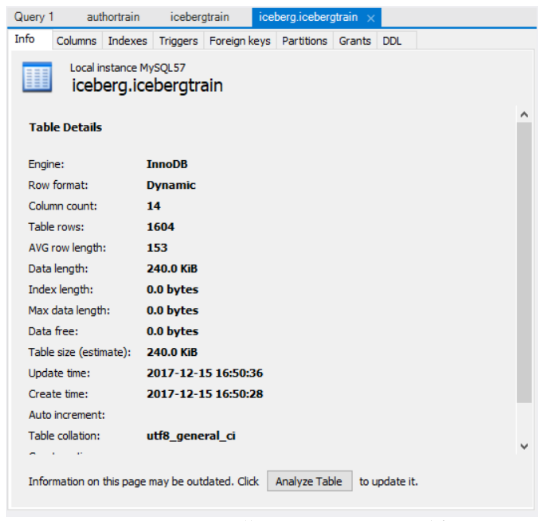
4.1 Data Analysis

First, we put the data into a database (See Figure 17 and 18).



id	band_convolutioned_minus_mean	band_convolutioned_minus_min	band_conv
0	13.35107322	6.59964564	2.97881059
1	13.80295913	5.95085512	4.38111522
2	12.52633109	6.98157736	1.78140780
3	14.31076125	7.8842648	2.88291060
4	13.61596785	7.45969524	2.09634399
5	12.89070331	6.14911436	4.62849789
6	14.56651009	7.13472092	2.44085775
7	14.14310425	7.98972352	2.04047950
8	14.19622646	5.80380136	6.32762354
9	14.03148277	7.55496656	1.98765044
10	13.2798705	6.39704224	2.78667279
11	14.46528763	7.1038898	2.67215649
12	17.22930679	7.3779268	6.32226872

Figure 17. Train data into Database



Local instance MySQL57
iceberg.icebergtrain

Table Details

Engine:	InnoDB
Row format:	Dynamic
Column count:	14
Table rows:	1604
AVG row length:	153
Data length:	240.0 KiB
Index length:	0.0 bytes
Max data length:	0.0 bytes
Data free:	0.0 bytes
Table size (estimate):	240.0 KiB
Update time:	2017-12-15 16:50:36
Create time:	2017-12-15 16:50:28
Auto increment:	
Table collation:	utf8_general_ci

Information on this page may be outdated. Click Analyze Table to update it.

Figure 18. Loading Process Specifics

Before analyzing the data, we came up with several assumptions and expectations about the data.

1. We assumed that the values of bands are different from iceberg and boat because the boat is made of metallic materials and iceberg is made of ice. the values for bands are *dB* values (energy received). We assumed that the numbers of energy received from iceberg are different from the numbers of energy received from boat.
2. We assumed that the shapes of object are different from iceberg and boat. For example, boat would have oval

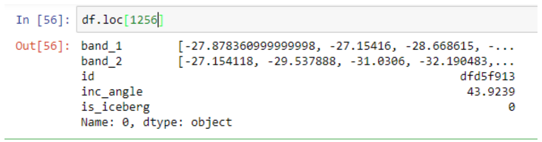
shape and would be uniform but iceberg could be any shape and would be less uniform.

3. We assumed that the boat images would have more uniform values on the object because the material and the density of material for boat would be more uniform than the iceberg.

Our data analysis is based on these assumptions and expectation.

Simple Data Exploration

To understand band1 and band2, we created 2-d image for some incidents of band1 and band2 (See Figure 19).



```
In [56]: df.loc[1256]

Out[56]: band_1         [-27.878360999999998, -27.15416, -28.668615, -...
         band_2         [-27.154118, -29.537888, -31.0306, -32.190483,...
         id                                                      dfd5f913
         inc_angle                                                43.9239
         is_iceberg                                                     0
         Name: 0, dtype: object
```

Figure 19. raw data of incidence (index: 1256)

The data-set includes id, 3 variables, and 1 target label. As mentioned earlier, *band1* and *band2* represents image. Here is the image we created from one of example in *band1* (see Figure 20):

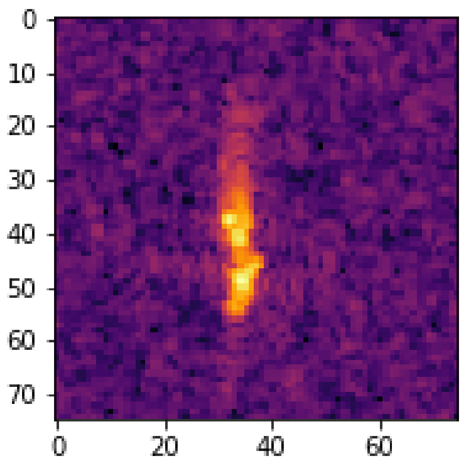


Figure 20. example image in 2-d (band1, idx:1256, boat)

This example image (Figure 20) is 75 x 75 pixel image that each pixel has values (*dB* value). There are total two pixel-image data (*band1* and *bans2*) for each incident. The points on the brighter area have higher values and the other points on the darker area have lower values. For better understanding, we plotted 3-d image as well where x and y are point locations and z (height) is the value on the point. We used same data as the one we used earlier (Figure 20). Here is the image figure (See Figure 21).

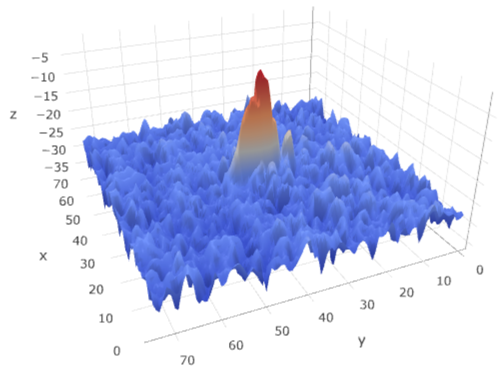


Figure 21. example image in 3-d(band1, idx:1256, boat)

The brighter points in 2-d image corresponds to the red points where z value is high in 3-d image.

Numerical Analysis

Since the given inputs are list-type data, we generated other features to analyze the data. First, we produced simple statistical summary for individual incidences. For this, we picked 9 random indexes of iceberg incidences. Here is the statistical summary and histogram generated in Python (see Figure 22 and Figure 23).

	1575	1420	361	1012	1227	629	1159	1501	410	1596
count	5625.00	5625.00	5625.00	5625.00	5625.00	5625.00	5625.00	5625.00	5625.00	5625.00
mean	-17.50	-15.59	-21.12	-10.94	-26.55	-20.04	-21.52	-21.56	-23.54	-13.88
std	4.08	2.63	2.96	2.86	2.50	2.62	2.29	4.70	2.41	2.43
min	-27.73	-26.53	-31.62	-22.76	-37.41	-32.31	-33.25	-33.08	-35.21	-24.01
25%	-19.77	-17.19	-22.89	-12.73	-28.23	-21.79	-22.86	-24.17	-25.11	-15.46
50%	-17.95	-15.52	-21.17	-11.04	-26.52	-19.97	-21.54	-22.37	-23.39	-13.77
75%	-16.15	-13.98	-19.58	-9.31	-25.10	-18.33	-20.11	-20.29	-21.96	-12.30
max	17.96	1.84	2.91	3.54	-11.01	-0.24	-3.49	8.11	-4.23	-4.66

Figure 22. Statistics of band values (iceberg)

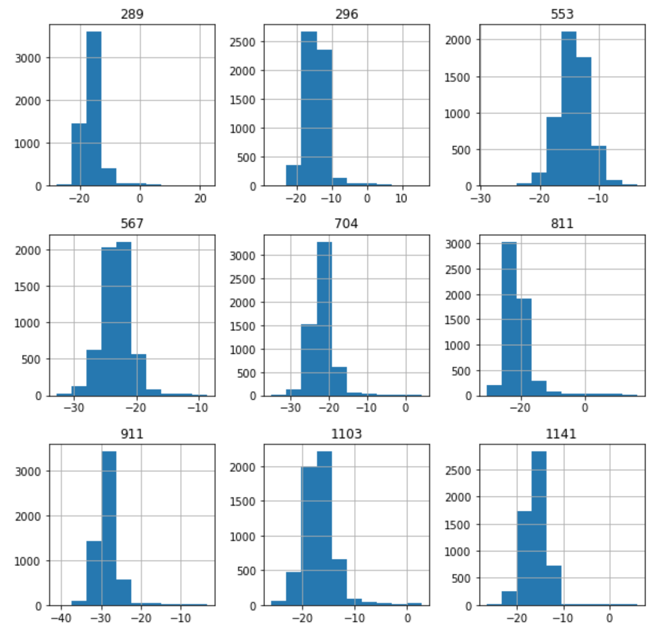
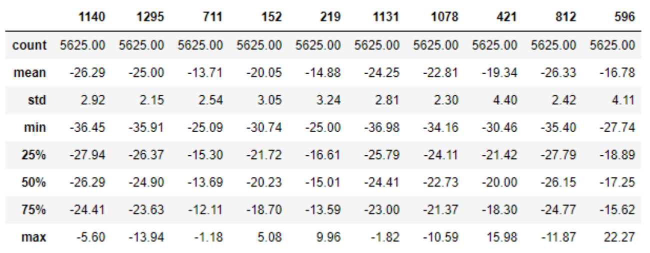


Figure 23. Histogram of band values (iceberg)

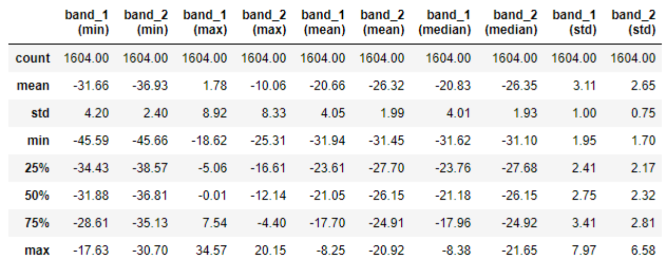
And here is statistical summary for 9 randomly picked boat incidences(see Figure 24 and Figure 25).



	1140	1295	711	152	219	1131	1078	421	812	596
count	5625.00	5625.00	5625.00	5625.00	5625.00	5625.00	5625.00	5625.00	5625.00	5625.00
mean	-26.29	-25.00	-13.71	-20.05	-14.88	-24.25	-22.81	-19.34	-26.33	-16.78
std	2.92	2.15	2.54	3.05	3.24	2.81	2.30	4.40	2.42	4.11
min	-36.45	-35.91	-25.09	-30.74	-25.00	-36.98	-34.16	-30.46	-35.40	-27.74
25%	-27.94	-26.37	-15.30	-21.72	-16.61	-25.79	-24.11	-21.42	-27.79	-18.89
50%	-26.29	-24.90	-13.69	-20.23	-15.01	-24.41	-22.73	-20.00	-26.15	-17.25
75%	-24.41	-23.63	-12.11	-18.70	-13.59	-23.00	-21.37	-18.30	-24.77	-15.62
max	-5.60	-13.94	-1.18	5.08	9.96	-1.82	-10.59	15.98	-11.87	22.27

Figure 24. Statistics of band values (iceberg)

By these statistics and histograms, numerical values in bands seem to be inconsistent; the range of numbers are different from each other. This tells us that the absolute values of bands themselves cannot be used for prediction model. Therefore, we generated overall statistical summary of statistics of every incidence to see the relationships between statistical values. This includes min, max, mean, median, and standard deviation. Here is the overall statistics produce in python (see Figure 26).



	band_1 (min)	band_2 (min)	band_1 (max)	band_2 (max)	band_1 (mean)	band_2 (mean)	band_1 (median)	band_2 (median)	band_1 (std)	band_2 (std)
count	1604.00	1604.00	1604.00	1604.00	1604.00	1604.00	1604.00	1604.00	1604.00	1604.00
mean	-31.66	-36.93	1.78	-10.06	-20.66	-26.32	-20.83	-26.35	3.11	2.65
std	4.20	2.40	8.92	8.33	4.05	1.99	4.01	1.93	1.00	0.75
min	-45.59	-45.66	-18.62	-25.31	-31.94	-31.45	-31.62	-31.10	1.95	1.70
25%	-34.43	-38.57	-5.06	-16.61	-23.61	-27.70	-23.76	-27.68	2.41	2.17
50%	-31.88	-36.81	-0.01	-12.14	-21.05	-26.15	-21.18	-26.15	2.75	2.32
75%	-28.61	-35.13	7.54	-4.40	-17.70	-24.91	-17.96	-24.92	3.41	2.81
max	-17.63	-30.70	34.57	20.15	-8.25	-20.92	-8.38	-21.65	7.97	6.58

Figure 26. Simple statistics on *band*1 and *band*2

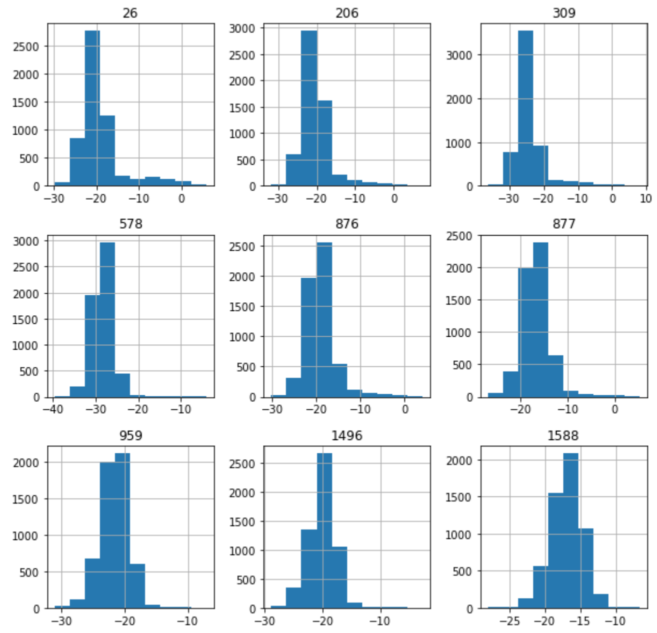


Figure 25. Histogram of band values (boat)

And here is the *correlation heatmap*[3] to produce better visual representation (See Figure 27).

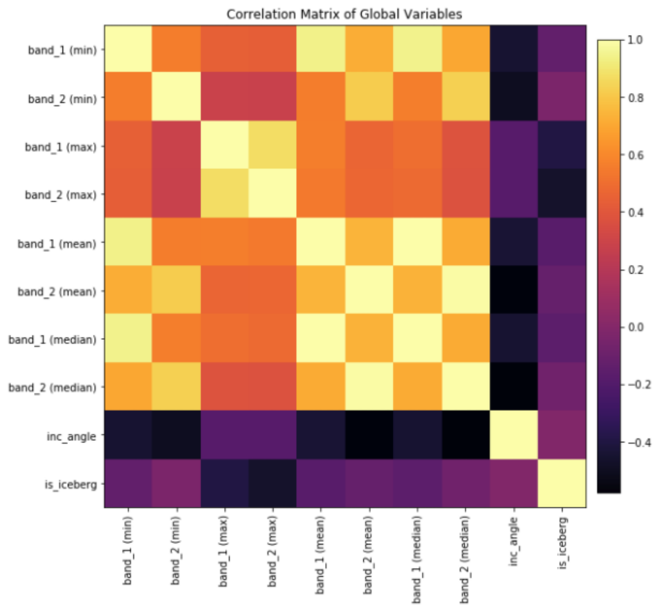


Figure 27. Correlation heatmap

The Figure 27 suggests that the statistics of *band1* and *band2* are highly correlated. This tells us that we can use merged value for band1 and band2. So we decided to merge band1 and band2 by summation as a new feature. Furthermore, to count incidence angle as well, we used the formula below

[3]For correlation heatmap, we used Python and its library matplotlib.pyplot to produce heatmap for correlation matrix.

to combine band values and angle:

$$band_{combined} = \frac{(band1 + band2)}{sin(\theta)} + \frac{(band1 + band2)}{cos(\theta)}$$

After this, we created features that have statistical values (max, min, and mean) of combined band. Now it seems like we can test our first assumption with our new features. To check this, we created summary table (Table 1 and Table 2).

	band (max)	band (min)	band (mean)
count	753	753	753
mean	-17.479160	-62.192641	-47.946987
std	9.977320	4.784894	4.689332
min	-43.822179	-74.665081	-60.648848
25%	-24.759941	-65.462388	-50.978926
50%	-18.502506	-62.561561	-48.209898
75%	-10.369021	-59.423179	-45.259184
max	12.528949	-45.790468	-32.838524

Table 1. Summary for iceberg

	band (max)	band (min)	band (mean)
count	851	851	851
mean	-3.489599	-60.755302	-46.117835
std	19.139787	6.342582	6.320192
min	-46.518556	-78.232979	-63.395427
25%	-19.318966	-65.253326	-50.640522
50%	-5.712788	-60.699135	-45.919064
75%	13.228916	-55.704962	-41.073443
max	54.729166	-42.761624	-30.329713

Table 2. Summary for boat

By looking at the tables (Table 1 and Table 2), the max values between boat and iceberg seems different between iceberg and boat. Here is the histogram representation of the values (Figure 28 and Figure 29).

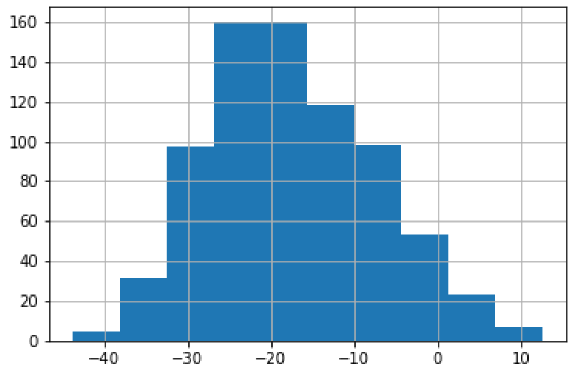


Figure 28. Histogram of max values (iceberg)

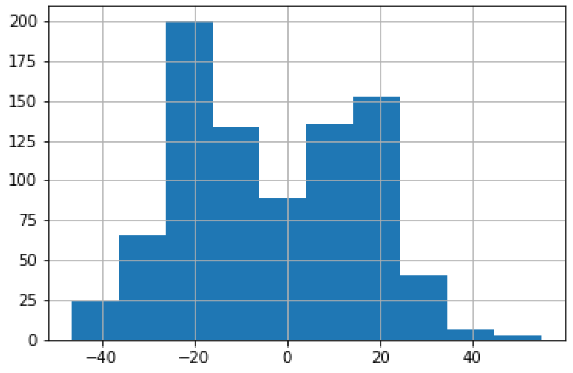


Figure 29. Histogram of max values (boat)

It seems like the boat images have higher max values. So we counted number of each boat and iceberg incidences that are higher than some arbitrary threshold (see Table 3).

Value of threshold	Number of iceberg	Number of boat
0	386 (90.61%)	40 (9.39%)
1	379 (92.44%)	31 (7.56%)
2	368 (93.88%)	24 (6.12%)
3	354 (95.68%)	16 (4.32%)
4	341 (97.43%)	9 (2.57%)

Table 3. Count by max

The table suggests that the max value can differentiate between boat and iceberg more than 90% for top 426 incidences in max values. Our first assumption seems to be working, however, only for about 25% of total data. To expand this idea, we looked some example images of these incidences. Here is images (we picked 4 random data that its max is greater than 4 - See Figure 30).

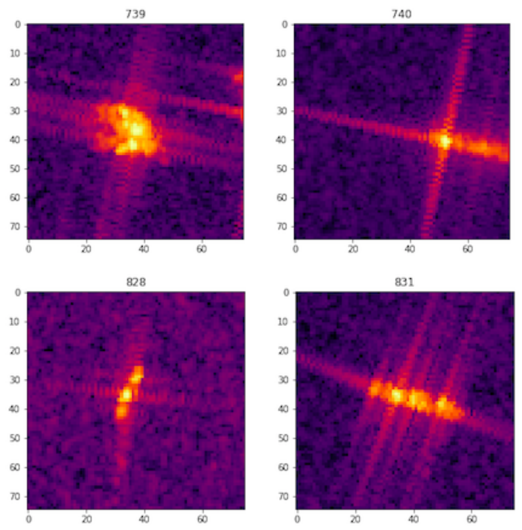


Figure 30. Images of incidences that has max value greater than 4

It seems like theses incidences have pattern; there is/are crossing(s) line. To check the pattern of images, we tried different approaches: peak analysis.

Peak Analysis

The images (Figure 30) suggests that there might be different patterns in images between iceberg and boat like our second assumption says. To check the second and third assumption about the shape of the object, we thought that the coordinate locations for peaks might be helpful. If our second assumption is right, the peak coordinates of boat images would make some obvious shape while the iceberg does not. And if our third assumption is right, the peak values would be more consistent. To extract the peak values and coordinates, we used the algorithm (see Algorithm 1).

Algorithm 1: Get Peaks

```
Data: $arr$, list-type data that have 5625 numeric values
Result: 1. $idx$, list of peak indexes, 2. $val$, list of peak
        values, and 3. $count$, number of peaks
1  $idx \leftarrow$ empty set
2  $val \leftarrow$ empty set
3  $count \leftarrow 0$
4  $nlargest_{50} \leftarrow$ 50 largest values in $arr$
5  for $i$ in $nlargest_{50}$ do
6      if $i$ is peak then
7          $idx \leftarrow idx \cup i$
8          $val \leftarrow val \cup arr[i]$
9          $count \leftarrow count + 1$
10 Return $idx$, $val$, $count$
```

Transformation of image (using convolution[4])

Since the given data is image-type data and values are not concise and the patterns are not clearly shown, we transformed our image using convolution. We used convolution algorithm (See Algorithm 2).

Because some of the values for bands are negative, we first mapped values into certain range - 0 to $(max - min)$. By this mapping, all the values become positive. With mapped band values, we tried convolution of images. For convolutional kernel, we used:

$$\frac{1}{28} \times \begin{vmatrix} 3 & 3 & 3 \\ 3 & 1 & 3 \\ 3 & 3 & 3 \end{vmatrix}$$

The reason why we are using this kernel is because we want to emphasize on the neighboring pixels so that the pattern of images become more emphasized at the same time. After performing convolution with this kernel, we expected the newly created images would be more concise and have more

Algorithm 2: Convolution Algorithm

```
Data: 1. $Img$, $n \times n$ pixel image $n$
      2. $kernel$, convolutional kernel
      3. $x$, number of convolutions
Result: $(n-x) \times (n-x)$ pixel image
1  $Img' \leftarrow$ empty image of same size with $Img$
2  for $i \leftarrow 0$ to $x$ do
3      for $j \leftarrow 0$ to $length(Img)$ do
4          for $t \leftarrow 0$ to $length(Img[j]$ do
5              $matrix_{jt} \leftarrow$ 3x3 matrix around $Img[j][t]$
6              $Img'[j][t] \leftarrow matrix_{jt} * kernel$
7              ***Note: '*' is dot-product in matrix
8  Return
   $Img'[n : length(Img) - 2n][n : length(Img[0] - 2n)]$
```

obvious pattern. Here is the difference of image before and after the convolution (see Figure 31).

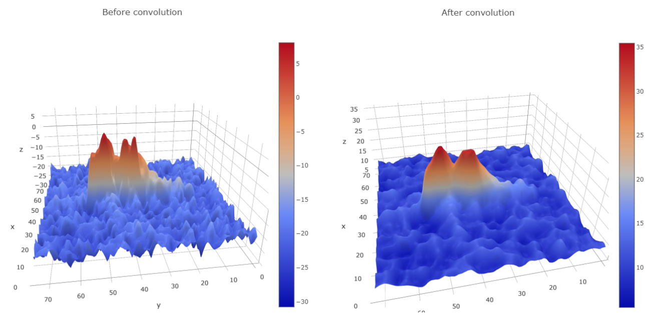


Figure 31. Before and After of Convolution (used idx: 1256)

The newly created image after convolution is blurred and more concise. With this newly created image, the pattern is more easily readable.

4.2 Methods

For prediction model, we used logistic regression, support vector machines (SVM), and convolutional neural network (CNN). For logistic regression and SVM, we followed the same process as earlier project.

4.2.1 Logistic Regression

Logistic regression is the algorithm to use for predicting percentage values for categorical target label. Since the algorithm follows the similar process as linear regression, the algorithm is based on assumption that the input data has linearity. However, unlike the linear regression, the output of the prediction is based on the Sigmoid function $f(x) = \frac{1}{1-e^x}$, not based on the linear function, so that it is bound to specific range (0 to 1).

For logistic regression, we tested 2 predictions by 1. features created by original image and 2. features created by converted image by convolution. For the features, we used statistics we created from ***Data Analysis*** section. For test, we

[4]Image kernel (convolution) is used for blurring, sharpening, embossing, edge detection, and more. This is done by dot-product operations (in matrix) to each pixel of images.

divided given data into 90% of train set and 10% of test set (we picked data sets randomly). To check the quality of the prediction, we used both accuracy and log-loss.

4.2.2 Support Vector Machines (SVM)

Support Vector Machine (SVM) is the algorithm that is used for categorical target label and regression model. The advantage of using SVM is that because it reduces the dimension of given input data by dividing features by kernels, it works efficiently for data that has large number of dimensions and for data that has non-linearity.

For SVM, we also tested 2 predictions by 1. features created by original image and 2. features created by converted image by convolution. For the features, we the same features as features used in logistic regression. For test, we divided given data into 90% of train set and 10% of test set (we picked data sets randomly). To check the quality of the prediction, we used both accuracy and log-loss.

4.2.3 Convolutional Neural Network (CNN)

Convolutional neural network (CNN) is the algorithm widely used in image recognition. It is using neural network algorithm input images. The expected input data (layer) for CNN is multi-dimensional array as an image. For training, just like other neural network algorithm, CNN converts the input layers into other hidden layers and finally produce output layers (See Figure 32[11]).

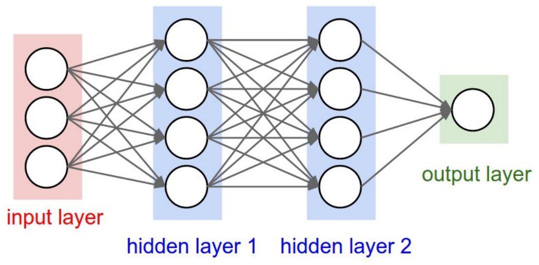


Figure 32. General process of neural network

The output layers have the same size with unique target labels. For our project, the output layers would have the size of 2 (iceberg or boat). For the converting process of input layers into hidden layers, there are steps and types of layers that the algorithm converts: 1. convolutional layer, 2. pooling (sub-sampling) layer, 3. flattening layer, and 4. output layer (See Figure 33).

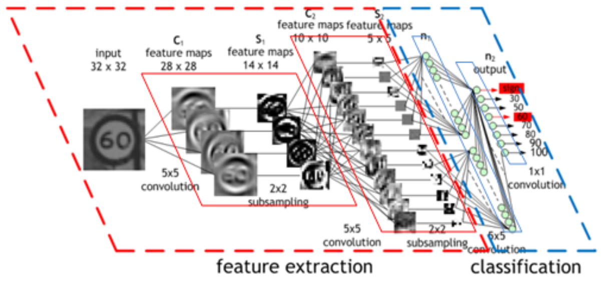


Figure 33. Overall process of CNN

In process of converting input data into convolutional layer, the algorithm uses specific kernels (as we chooses our kernel in ***Data Analysis - Transformation of image***). The specific kernels that the algorithm uses should convert the pixel values of images so that certain features or properties become more emphasized. Our kernel is used for blurring and making more concise the image. However, a learner can choose any other kernels depending on their purpose; there are kernels to emphasize edges, sharpening, and other types of filterings.

For CNN, we used 2 convolutional layers of 32 feature-detecting kernels, 2 pooling layers, and 1 flattening layer. For specific values for convolutional kernels, we used default kernel for *Keras* library in Python.

4.2.4 System Specification

Commonly used system specification:

- OS/ version: macOS High Sierra/ version 10.13.1
- Processor: 1.6GHz Intel Core i5
- Memory: 8 GB 1600 MHz DDR3
- Programming platform: Python
- Libraries:
  Visualization:
  1. matplotlib
  2. plotly

  Dealing with data:
  1. pandas
  2. numpy

  Splitting train/test data:
  1. sklearn.model_selection

For logistic regression[12]:

- Building prediction model:
  1. sklearn.linear_model.LogisticRegression

For SVM[5]:

- Building prediction model:
  1. sklearn.svm

For CNN:

- Building prediction model: Keras[13]

1. keras.preprocessing.ImageDataGenerator
2. keras.models.Sequential
3. keras.layers.Conv2D, MaxPooling2D, Dense, Dropout, Input, Flatten, Activation
4. keras.layers.GlobalMaxPooling2D
5. keras.layers.normalization.BatchNormalization
6. keras.layers.merge.Concatenate
7. keras.models.Model
8. keras.initializers
9. keras.optimizer.Adam
10. keras.callbacks.ModelCheckpoint, Callback, EarlyStopping

4.3 Results

For our result, we used both accuracy and log-loss function to check the quality of the algorithm. We first generated those values with using the training set by splitting train and test set. Then we picked the best algorithm for the submission. The final submission on the actual test set was done with CNN and the score was 0.344.
For accuracy, we simple used:

$$p(x) = \frac{1}{N}\sum_{i=1}^{N} p_i$$

For log-loss, we used:

$$loss(x) = -\frac{1}{N}\sum_{i}^{N}\sum_{j}^{N} y_{ij} log(p_{ij})$$

4.3.1 Logistic Regression

Here is the table of quality of the logistic regression (See Table 4):

	Accuracy	Log Loss
original data	77.64%	0.482
convolutioned data	80.12%	0.415

Table 4. Result: Logistic Regression

4.3.2 Support Vector Machine

Here is the table of quality of the SVM (See Table 5):

	Accuracy	Log Loss
original data	78.88%	0.429
convolutioned data	81.36%	0.406

Table 5. Result: SVM

4.3.3 Convolutional Neural Network

Here is the table of quality of the CNN (See Table 6):

	Accuracy	Log Loss
original data	89.85%	0.315

Table 6. Result: CNN

For CNN, we did not use our convolutioned image because the algorithm already have convolution step. Because CNN preformed the best for testing the quality, we used CNN to submit our prediction on the test set. The final score we have on Kaggle was 0.344.

4.3.4 Overall Result

Overall, our algorithms predicted labels with 85% accuracy and 0.4 in log-loss function. The result suggests that our algorithm predicts right label, 0 or 1, with 80% accuracy but the confidence rate of predicting right label as percentage value, it is not performing as well as our expectation. Considering the quantity of predicting the right label, we think our algorithm performs moderately okay. However, considering the quality of predicting the right label as percentage, we think our algorithm should do better performance. For our submission on test set, the rank produced by our submission on Kaggle did not show in top 100. So we do not deem our work is as successful as we wished.

The challenging part of this project was the given data was not usual data we were dealing before. It was pixel-image and the values of pixels are not 8-bit integer like normal pixel-image. Because of this, we tried many approaches such as peak analysis, number analysis, mapping values, convolution process, and etc.

4.4 Summary and Future Work

As mentioned in ***Result - Overall Result***, the best result of our algorithm (using CNN) was 89.85% accuracy and 0.315 in log-loss. For further development, we could:

1. try to increase of layers for CNN - because we couldn't do it for time-consumption issue.
2. use different software such as ***ImageNet***[14] or ***Microsoft Azure Computer Vision***[15].
3. try modified version of CNN[16].
4. expand further approaches such as trying different kernel and producing statistics based on that.

Citations and Subsubsection

This level should be used sparingly, but can provide a sense to the reader of the outline and, therefore, relationship of information. You'll need to have at least 10 citations. Here is an example. This work was began by Micky Mouse[17]. Here's a way to provide bullet points:

Word Definition

Concept Explanation

Idea Text

Acknowledgments

We would like to thank to our professor, Dr. Mehmat Dalkilic and Marcin Malec for guiding us throughout the project. We would also like to thank School of Informatics and Computing for providing us with the resources of Burrow on which we ran most of our models.

References

[1] Ullman J.D. Rajaraman, A. Data mining - mining of massive datasets. pages 1–17, 2011.

[2] Pavel Senin. Term frequency - inverse document frequency statistics. https://jmotif.github.io/sax-vsm$_{s}ite/morea/algorithm/TFIDF.html, 2016. Accessed: 2017-12-01$.

[3] Logistic regression. `https://www.medcalc.org/manual/logistic_regression.php`. Accessed: 2017-12-01.

[4] Rahul Saxena. How the naive bayes classifier works in machine learning. http://dataaspirant.com/2017/02/06/naive-bayes-classifier-machine-learning/, 2017. Accessed: 2017-11-30.

[5] Svm documentation (scikit-learn). `http://scikit-learn.org/stable/modules/svm.html#regression`. Accessed: 2017-11-24.

[6] Article: Support vector machine. `https://en.wikipedia.org/wiki/Support_vector_machine`. Accessed: 2017-12-01.

[7] Artificial intelligence - neural networks. `https://www.tutorialspoint.com//artificial_intelligence/artificial_intelligence_neural_networks.htm`. Accessed: 2017-12-02.

[8] `https://www.google.com/imgres`. Accessed: 2017-12-02.

[9] Piotr Bojanowski Tomas Mikolov Armand Joulin, Edouard Grave. Bag of tricks for efficient text classification. *Cornell University Library*, 3, 2016.

[10] Carl Howell. Iceberg and ship detection and classification in single, dual and quad polarized synthetic aperture radar. *Memorial University of Newfoundland*, pages 45–46, 2008.

[11] Andrej Karpathy. Convolutional neural networks for visual recognition. `http://cs231n.github.io`. Accessed: 2017-11-20.

[12] Logistic regression documentation (scikit-learn). `http://scikit-learn.org/stable/modules/linear_model.html#logistic-regression`. Accessed: 2017-11-24.

[13] Keras documentation. `https://keras.io/layers/convolutional`. Accessed: 2017-11-30.

[14] Imagenet. `http://www.image-net.org`. Accessed: 2017-12-10.

[15] Microsoft azure computer vision. `https://docs.microsoft.com/en-us/azure/cognitive-services/computer-vision/home`. Accessed: 2017-12-10.

[16] Xi Chen Diederik P. Kingma Tim Salimans, Andrej Karpathy. Pixelcnn++: Improving the pixelcnn with discreatized logistic mixture likelihood and other modifications. *conference paper at ICLR 2017*, 2017.

[17] Kaiser Von Butterhausen and U. C. Bear. When to get food from the trash: optimization. *Canubus*, 10:2–5, 2017.