

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

Ashay Sawant^{1*}, John Stein^{2*}, Samanvitha Pradhan^{3*}

Executive Summary This report summarizes two datamining projects undertaken by the authors. The first project attempts to classify ‘scattered pages’ from works of fiction written by Edgar Allan Poe, HP Lovecraft, and Mary Wollstonecraft Shelley according to their author. Various approaches were explored, including Naive Bayes, Decision Trees, Maximum Entropy. The best-performing approach was achieved by Maximum Entropy. The second project attempts to classify Synthetic Aperture Radar images according to whether the image has captured an iceberg or not. We have used two approaches including Convolution Neural Networks and Histogram of Oriented Gradients to process the image in a way to obtain custom features. Finally, we performed regression on custom features using Logistic Regression, Random Forest and XGBoost Algorithms. The best-performing approach was achieved by Logistic Regression using custom features, with a Log Loss of 0.3682.

Keywords

NLP — Natural Language Processing — SAR — Image Recognition — HOG — Histogram of Oriented Gradients — Logistic Regression — Random Forest — XGBoost

¹ Data Science, School of Informatics, Computing, and Engineering, Indiana University, Bloomington, IN, USA

*Corresponding author: ahsawant@iu.edu

² Computer Science, School of Informatics, Computing, and Engineering, Indiana University, Bloomington, IN, USA

*Corresponding author: jodstein@iu.edu

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

*Corresponding author: sspradha@iu.edu

Contents

1	Introduction	1
1.1	Author Identification	2
1.2	Statoil/C-CORE Iceberg Classifier Challenge	2
2	Datamining	2
2.1	Data Preprocessing	3
2.2	Mining, Interpretation, and Action	4
3	Author Identification: Full Problem Description	5
3.1	Data Analysis	5
3.2	Methods	5
3.3	Results	6
3.4	Summary and Future Work	6
4	Iceberg: Full Problem Description	6
4.1	Data Analysis	7
4.2	Methods	8
4.3	Results	12
4.4	Summary and Future Work	13

Acknowledgments

14

References

14

1. Introduction

The two projects presented herein (Spooky Author Identification and Iceberg Classifier Challenge) are the results of the combined efforts of the authors to meet the corresponding challenges published by Kaggle. [1] [2] Kaggle.com is a site dedicated to the Data Science community where entities can post problems and challenges to be taken on by Data Science academics and enthusiasts. Readers can find more details on each of these challenges at Kaggle’s website ¹ in the Competitions section.

Data Mining (DM) has three broad areas of application: answering questions, exploring and discovering relationships, and annotation. Kaggle competitions, including those described in this paper, are a convenient way for an entity with a requirement in one of these application spaces to draw from a large community of willing problem solvers to find effective

¹<https://www.kaggle.com/>

and efficient solutions to satisfy their requirement. Spooky Author Identification is considered a ‘Playground Competition’, meaning the underlying motivation is to encourage increased participation in the Kaggle community and potentially spur innovative advances in to standing challenges in Natural Language Processing. In the case of Iceberg Classification, the underlying requirement is for Statoil to accurately and efficiently identify icebergs that could threaten their sea-based natural resources operations. In both cases, DM presents an opportunity to discover useful relationships between written word and their author or between SAR imagery and the target being imaged, to enable classification accuracy and efficiency well beyond human performance.

1.1 Author Identification

The Author Identification problem is to accurately classify whether a given statement of text (an excerpt) is from one of the Horror(Spooky) Authors namely, Edgar Allen Poe, H P Lovecraft, and Mary Wollstonecraft Craft. The data has three classes as “EAP”, “HPL”, and “MWS”. The training set has three columns, one identifier, one column of text, and one column as the class label with one of the three classes. The excerpts are anywhere from a quote to a description. The classifier must calculate the probability that and map the text to one of the authors. The test data set has an identifier and a text per row and the trained classifier must classify this set. Kaggle evaluates the goodness of the classifier using the below stated Log Loss function. A good classifier must have a low log loss.

$$LL = -\frac{1}{N} \sum_i \sum_j y_{ij} \log(p_{ij})$$

1.2 Statoil/C-CORE Iceberg Classifier Challenge

The Iceberg Classifier Challenge is to accurately classify whether a Synthetic Aperture Radar (SAR) image contains an iceberg or something that is not an iceberg, which makes it a binary classification problem. The data provided includes an 75×75 image that represents horizontally polarized backscatter intensity (band_1), a 75×75 image that represents vertically polarized backscatter intensity (band_2), and the angle of incidence between Earth-normal and the transmit beam (inc_angle). Training data also contains the class label (is_iceberg). The classifier is to calculate the probability that the image contains an iceberg (1) or not (0). Kaggle evaluates the goodness of the classifier using Log Loss, shown below.

$$LL = -\frac{1}{N} \sum_{i=1}^N t_i \log(y(x_i)) + (1 - t_i) \log(1 - y(x_i))$$

The best classifiers will have Log Loss scores near zero. Receiver Operator Characteristics (ROC) was also used to evaluate the True Positive vs False Positive behavior of the classifier. High-performing classifier will have ROC curves

that resemble a step function and with an area under the curve (AUC) near 1.0.²

2. Datamining

- What is DM?
DM seeks to inform decision making by answering questions about, exploring or discovering relationships in, or providing annotation of data.
- What does it yield?
DM yields information that is not possible or practical to directly observe, based on data that is practical to observe. This information may be predicted trends in human behavior, perceived relationships or clusters within high-dimension data, or a prediction of future events. In all cases, the yielded information is deemed valuable to someone who is then enabled to make decisions based on that information.
- What are the general steps?
 - The first (and most difficult) step in DM is developing the problem statement: What decision or action is the DM effort required to inform or enable, and what must the DM effort yield in order to satisfy that requirement?
 - The second step is data acquisition. Acquisition of data is non-trivial. It can be difficult, expensive, and in some cases even prohibited or restricted by law. Therefore, the miner must be intentional about pursuing the acquisition of data that is both needed (from the problem statement) and available, whether by manual observation, survey, or automated sensing/recording.
 - The third (and most time consuming) step in data-mining is data pre-processing. The data must be analyzed, including identifying unknown, missing, or outlier records, examining attributes and their domains, and visualizing the data for first-order behaviors, trends, and distributions. The data must then be integrated and/or cleaned, including dealing with unknown, missing, or bad data, enriching the data from other sources, and/or transforming the data to work better in the algorithm.
 - The fourth step is to actually mine the data (i.e. answer questions, explore or discover relationships, or annotate). The methodology, assumptions, models, and parameters of how the mining is performed can depend on the objective, the data itself, and the miner themselves.
 - The fifth and final step in DM is to interpret the output of the mining step and either validate that it meets the stated objective/requirement of the

²ROC finds its origins in World War 2 as radar operators were evaluated on their ability to distinguish real targets from false targets and clutter. [3]

problem statement, or determine that changes to the pre-processing or mining steps are warranted and start again with a modified approach.

- What is clustering vs. classification?

Classification refers to the task of predicting the label of some unobserved target attribute given a set of observed attributes for some record. The label is generally a single member of a finite list of possible labels. Clustering refers to the task of identifying or discovering relationships among data. In other words, are there properties or measures for which subsets of data can be interpreted as similar or dissimilar in a meaningful way? Clustering aims to discover these ‘clumps’ or similar data as well as the properties or measurements from which the similarity can be determined. It is useful to think of Classification as a supervised learning task because it is notionally possible to produce a training set with labeled records, and Clustering as an unsupervised learning task because the number and meaning of the discovered clusters is not known apriori (if they were, they could be labeled and it would be called Classification).

- What is a loss function?

A loss function is a function which attempts to assign a real value to some undesirable property of an intermediate outcome during the mining step. This value (known as cost or loss) may represent error, variance, or bias of a model, clustering, or classifier. The actual value of loss is usually not meaningful. Instead, the goal is generally to minimize the loss over some parameter or variable during an optimization step.

2.1 Data Preprocessing

- What are the steps, and what challenges does each present?

The first step in data preprocessing is data analysis. Data analysis can include examining histograms, bar/pair-plots, frequency tables, etc. The goal is to get familiar with the data attributes and their domains, frequency of values, trends and distributions, and first-order behaviors. From the analysis, the two primary remaining tasks are selection and modification. Selection (i.e. sampling and attribute selection) refers to deciding which subset of records and/or which subset of attributes for each record to use for the mining task. Modification (i.e. aggregation, dimensionality reduction, discretization, and transformation) refers to changing existing, creating new, or transforming, attributes and their values.

- Sampling refers to choosing a subset of the total data from which to develop the analysis technique - often because working on the entire dataset would result in more resource investment than it would improve the performance of the technique. The challenge with sampling is that, especially for high-dimensionality data, the data data can

quickly become sparse, resulting in a greater potential for over-fitting due to the small data size and high dimensionality (degrees of freedom).

- Attribute Selection refers to selecting which attributes to include in the analysis and which to exclude, based on some preliminary analysis. Manual attribute selection (or de-selection) is challenging because humans can only perceive data in two, three, or perhaps four dimensions. One can only reasonably expect to act on the most obvious of attribute properties without impacting the analysis in unexpected ways.
- Aggregation refers to manually grouping several attributes or attribute values together, preferably while preserving some higher-level relationship among those being grouped. Aggregation can be relatively straightforward in some cases, but in other cases it can be very difficult to group attributes or values in a way that recognizes the essential common property of its members.
- Dimensionality Reduction refers to the creation of fewer new attributes which are combinations of the many old attributes. Two common methods include Principal Component Analysis (PCA) and Neural Networks. The primary challenge with dimensionality reduction is that the meaning of the original attributes are all but lost as the data are mapped to the new, reduced feature space.
- Discretization refers to the assignment of values from a continuous-valued attributes into one of finite-many bins.
- Transformation refers to the mapping of old attribute values to new attribute values - generally reversible.

Finally, data with unknown, erroneous, or missing values must be handled. Some of these handling techniques may be implemented in the mining algorithm(s) themselves, but some may be performed prior to the mining step. Unknown/missing data handling is more challenging for some algorithms (i.e. regression) than it is for others (i.e. decision trees). Some algorithms need each record to have good data, or they will simply not work. Other algorithms can be made to handle unknown/missing data in a reliable way and achieve good performance. Still other algorithms can operate on records having missing data with no extra effort. The primary challenge, especially for high-dimensionality data, is that the impact of the handling technique can be different for every attribute, and must be understood and chosen carefully.

- What is the general load (time, space, \$) for preprocessing?
Computationally, preprocessing may consume $O(n)$ or

$O(n \log m)$ time where n is the number of records to process and m may be search steps if matching, inclusion, or sorting operations are required. Preprocessing may consume anywhere from constant space to $O(n)$ space, if the records are processed one at a time and stored back into their original location, or copies are made. Practically speaking, preprocessing can consume upwards of 80% of the labor hours, and therefore 80% of the funding, of the overall DM effort. That is because it is the most human-intensive step of the process. Automation of the preprocessing step, although possible, is of limited benefit if the DM problems vary significantly in size, scope, data representation, purpose, etc. The ability to predict the preprocessing impacts of data across problem spaces is indeed an analytics problem unto itself.

2.2 Mining, Interpretation, and Action

Top Ten Algorithms in DM Below is a brief overview of the top 10 algorithms in DM. [4]

- **C4.5:**
C4.5 is an algorithm that can be used for decisions trees or rulesets. For trees, it essentially builds a decision tree by recursively splitting the data on attributes until the data is fully explained by the tree. It then prunes the tree by calculating a binomial error estimate at each of the leaves, and comparing that with the error that would be introduced by collapsing to the parent node, and repeating until complete. For rulesets, it similarly generates rules representing every complete path through the notional unpruned tree (i.e. the rules generated fully explain the data). Then, C4.5 discards conditions in the generated rules to minimize the binomial error estimate using hill-climbing techniques. Finally, in both cases, a default rule is created in case data do not satisfy any of the remaining rules.
- **K-Means:**
K-Means naively assigns K centroids to the space shared by the data, and then assigns each data point to the centroid that is closest (according to some distance function). Next, the algorithm computes new locations for the existing centroids based on that data points that are assigned to them. Then the data points are reassigned, if they are now closer to a different centroid. The process continues until convergence, which is guaranteed, and the centroids are taken to be the resulting clusters.
- **Support Vector Machines:** SVMs attempt to find some hyperplane that fully separates the data into their proper classes, so that the hyperplane can be used to estimate classes for future data. It chooses the hyperplane with the maximum distance from the nearest data. Many times the data is not linearly separable. Two primary approaches have been adopted in such cases: (a) the introduction of a slack variable can be used to assign a penalty to any hyperplane that incorrectly classifies some of the training data, and the task becomes to minimize that penalty, and (b) the data can be kernelized, or mapped onto a new space, which is linearly separable using traditional SVM.
- **Apriori:** The Apriori algorithm was developed to efficiently find frequent sets. It relies on the truth that if an item is not frequent in some set, then any superset containing that item is also not frequent in that set. Starting with singletons, candidate frequent subsets are generated from frequent subsets of lower cardinality, and then eliminated, thereby eliminating the need to check for frequency on subsets that could not be frequent.
- **Expectation Maximization:**
EM is a more general case of K-Means. For any probabilistic model, the parameters for that model are initialized to some starting value, and then the model is then used to cluster the training data. In many cases, the clustering is performed via soft-assignment, meaning each data point is assigned a probability of belonging to each cluster. Based on the new clustering, parameters for the model are recalculated to best fit the data assigned to each. Then the data are reassigned, and the process repeats until convergence (to some threshold) is achieved.
- **Page Rank:**
Page Rank is intended to find relative importance of hub nodes in a complex network, commonly applied to web pages on the Internet. Essentially each page is assigned a rank value that is the sum of all contributions of pages that link to it. Each incoming contribution is the contributing page's rank value divided by the number of outgoing links it has. In this way, the highest page ranks are achieved when a page has many links to it that come from pages with high rank and few outgoing links.
- **Adaboost:**
Adaboost is a type of ensemble learning algorithm that uses voting to create strong learners from multiple weak learners. Essentially, the algorithm picks a single (best) learner by which to classify training (labeled) data. It then assigns a weight to each training sample, lower for those that were correctly classified and higher for those that were not. It then selects a new (best) feature and similarly redistributes the weight to misclassified examples. After the last weak learner is used, each learner is then weighted according to the weighted error they would have using the final weights on the misclassified examples in each learner. The final weighted weak learners represents the new stronger ensemble learner.
- **K Nearest Neighbors:**
KNN is a relatively simple algorithm that finds the K

nearest neighbors to each data points and either clusters that data point according to whichever cluster was most frequent in the neighbors, or clusters that data point according to a weighted (by distance) vote from the nearest neighbors.

- **Naive Bayes:**

Naive Bayes classifiers essentially assume that all the features belonging to a dataset are independent of each other, given the class. Essentially, joint probability distributions are created (rather efficiently using the chain rule, due to the independence assumption) to provide a purely probabilistic estimate of class, based on the training data.

- **Classification and Regression Trees (CART):**

CART is an umbrella term encompassing several variations on building trees for use with categorical or real-valued data. In general, they work by building binary classification trees using intelligent selection criteria for choosing attributes to split on (such as information gain or the Gini impurity), and then perform pruning of the resulting tree (or stopping criteria) by minimizing a loss which contains a complexity penalty element and a misclassification (or error, in the real-valued case) element.

Despite DM's many benefits, the algorithms do not solve problems or tell users what to do. It is up to the human Dataminer to clearly understand the problem space, define what they need from the DM solution, exercise the DM process, interpret and validate the results and limitations of the results, and then use the results to inform their action.

New types of problems in DM include finding more stable decision-tree algorithms, decomposition of complex models into ensembles of simpler models, and improvement in feature selection techniques. [4] Additional (albeit tangential) topics related to DM include security/privacy implications of DM, the convergence of DM and Internet of Things (IoT), and novel applications of DM.

3. Author Identification: Full Problem Description

Edgar Allen Poe, H P Lovecraft, and Mary Wollstonecraft Craft are three popular Horror Fiction authors of their times and whose works have stood the test of time. Poe's Raven or Shelly's Frankenstein are works which have defined the genre for centuries. In the Spooky Author Challenge, the aim is build a classifier using a training set which contains excerpts from works by the three authors. The training dataset consists of three variables, an identifier, a string of text which is the excerpt and a class label column. There are 19,560 records in it. The unlabeled test set consists of 8,394 records and the Classifier must label it as "EAP", "HPL" or "MWS" as its solution

3.1 Data Analysis

Any data which is found has to be cleaned and made machine readable. The text data is the only component here. The strings are excerpts from various works. It can be noticed that some are descriptive sentences such as narration while some have dialogs in the text. They are all strings varying lengths. One of the text's features which stands out is that most of the data from MWS are predominantly in first person it is primarily sourced from Frankenstein. There are number of such instances of the data can be seen here. The following steps were used:

- CSV: As the training and test files were CSVs, when the data was loaded onto MySQL, the comma failed as a delimiter, and had to be processed to make it ' ' as this not part of regular sentence structure.
- Removing of all the punctuations: For the purposes of this assignment, the punctuations were ignored and hence removed as part of pre-processing.
- Stemming and Stopwords: The words were stemmed and stopwords were eliminated before the text was used for training and testing the data as part of pre-processing.

3.2 Methods

There are multiple models available for text processing. The models which were considered for Naïve Bayes', Decision Trees, SVMs and Maximum Entropy Models.

- *Naïve Bayes*: Though Naïve Bayes' is a relatively simple approach, like Occum's Razor, a simple approach and fruitful approach. Naïve Bayes' approach assumes that each feature is independent of one another. They are independently considered, and a probability is assigned to it. This is highly scalable in nature. It follows the following conditional probability method to classify

$$p(C_k|x_1, x_2, \dots x_n)$$

$$P(C_k|x) = \frac{p(C_k)p(x|C_x)}{p(x)}$$

Naïve Bayes works particularly well for textual data and hence gives highly reliable output. Python's NLTK Naïve Bayes' classifier was utilized here. This was trained on all the records of the training set and was made to classify the entire test set.

- *Decision Trees*: Decision Trees or Classification Trees uses various features which is learnt by the classifier to form nodes. Generally, a binary decision is made such as True or False in order to traverse the tree and there are class labels assigned as the root nodes. The path taken from root node to the leaf node serves are the decision made to classify the input as particular class. Decision Trees are easily interpretable by humans owing to their visual nature and hence are highly useful. Python's NLTK Decision Tree classifier was utilized

here. Cross Validation was utilized here, and was trained on sample of the records of the training set and was made to classify the entire test set. The dataset was too large and there were memory errors which occurs when the training set was larger than 5000 records.

- **SVM:** Support Vector Machines are classifiers used for supervised learning for when the given set of data is linearly separable using a hyperplane of required dimension. SVMs aim at obtaining the best possible solution for the linear separation by attempting to obtain a partition which yields the largest margin. SVMs are useful in classifying larger data sets as they rely only the support vectors to gauge the hyperplane. This is a well renowned tool for text classification.

Python's NLTK and sklearn's SVC classifier was utilized here. It was trained on sample of the records of the training set and was made to classify the entire test set.

- **Maximum Entropy:** Maximum Entropy implies that classifiers utilize the most uniform class which meets all the constraints. This has the advantage being applicable to any probability distribution. This classifier unlike Naïve Bayes doesn't assume independence of features. Hence, it uses contextual information to obtain information for classification. But this method faces an optimization issue which are improved using an iterative scaling algorithm such as the GIS (Generalized Iterative Scaling) or the IIS (Improved Iterative Scaling) for lambda parameters to be eliminated in the algorithm.

Python's NLTK Maxent classifier was utilized here. Cross Validation was utilized here, and was trained on sample of the records of the training set and was made to classify the entire test set. The dataset was too large and there were memory errors which occurs when the training set was larger than 5000 records.

```

and=None? ..... "MWS"
she=None? ..... "MWS"
  It=None? ..... "MWS"
    on=None? ..... "MWS"
    on=1? ..... "EAP"
    on=2? ..... "EAP"
  It=1? ..... "HPL"
    is=None? ..... "HPL"
    is=1? ..... "EAP"
  It=2? ..... "EAP"
she=1? ..... "MWS"
  from=None? ..... "MWS"
    moments=None? ..... "MWS"
    moments=1? ..... "EAP"
  from=1? ..... "EAP"
she=4? ..... "MWS"
and=1? ..... "HPL"
to=None? ..... "HPL"
  the=None? ..... "EAP"
    with=None? ..... "EAP"
    with=1? ..... "MWS"
    with=2? ..... "HPL"

```

Figure 1. Partial Decision Tree

3.3 Results

The classifiers each had certain drawbacks as stated above and the results are as follows.

- Naïve Bayes performed well with an accuracy rate of 0.910465294448 from the classifier. As stated above this is simple yet a powerful solution for classification.
- Decision Trees under cross validation gave extremely high accuracy rate of 0.978. This may seem like a perfect classifier, but it most certainly is skewed. There is a high chance of overfitting which has occurred in this classifier implying that the test data hasn't varied much from the training data. Hence, it is not an optimal solution.
- SVMs has produced a really low accuracy rate of 0.406 and this was cross validated, but the score failed to improve.
- Maxent classifier produced accuracy 0.944 with cross validation of 5000 records and 0.789 with 2000 records per iteration.

Remember, we're interested in the journey, so simply because an approach failed doesn't mean failure if you discuss the failure!

3.4 Summary and Future Work

The Spooky Author Detection project was intended for the classifier to identify Spooky Authors namely, Edgar Allen Poe, H P Lovecraft, and Mary Wollstonecraft Craft. The data has three classes as "EAP", "HPL", and "MWS". Excerpts from the authors literary work used to train the classifiers to learn the features of the authors. The most reliable results were obtained from the Naïve Bayes' classifier despite being a simple approach. Maxent classifier obtained better results by improving the cross validation. This project can be deemed a success, despite the obvious gaps in the results mainly because the data was real world data which was worked on. This yielded acceptable results. The handling and management of data was highlighted through this effort. The potential work can be done using Neural Networks. As stated there are nuances in the data set which were left unexplored for the scope of the current work done, like Shelly's work being primarily in First person, Shelly inspiring Poe and Lovecraft's work which may be reflected in the data as homage etc. Using NN such prospects can be explored, and better results might be obtained.

4. Iceberg: Full Problem Description

Statoil, an energy company with oil and natural gas operations based largely in the Norwegian Continental Shelf region, has an operational need to track the presence of icebergs that may present a threat to the safety and efficiency of those operations. [5] The company has therefore posed the following iceberg classification problem to the data science community. Statoil partners with C-CORE (a Canadian R&D corporation) to obtain access to Synthetic Aperture Radar (SAR) imagery data

from the Sentinel-1 satellite constellation, and the challenge is to classify the object in each image as an iceberg or non-iceberg (presumably a ship, which is not likely to pose a threat to operations). A labeled training set (1604 records) and an unlabeled test set (8424 records) is provided, and the task is to classify each image as containing an iceberg (`is_iceberg=1`) or not (`is_iceberg=0`). [2]

4.1 Data Analysis

Each record contains three components (besides an identifier):

- `band_1` - 75×75 pixel image representing horizontally-polarized backscatter intensity from the horizontally-polarized C-band radar transmission
- `band_2` - 75×75 pixel image representing vertically-polarized backscatter intensity from the horizontally-polarized C-band radar transmission
- `inc_angle` - angle of incidence (between the transmission path and Earth-normal at the target object)

Initial Analysis Some simple statistical analysis on the image data reveals that maximum and mean values in both bands are meaningful contributors to the classification task, as shown in Figure 2.

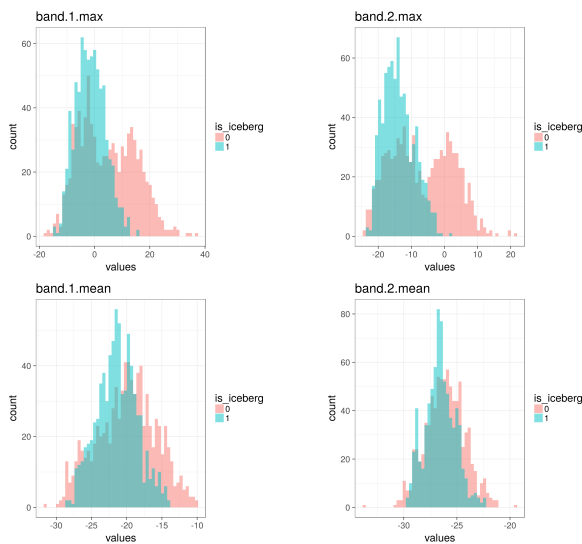


Figure 2. Band 1 and 2 Max and Mean

Further Analysis The overlap areas in the histograms represent those records that will cause difficulty for a classification model. Additional features are required for good performance. The following functionality was developed to enable visual analysis as well as experimentation with new useful features:

- Incidence angle has a perceived impact on backscatter intensity; see Figure 3. In this approach, simple linear regression was used to fit a line to background intensity as a function of incidence angle. The intercepts for this line were used to normalize all the images to the mean

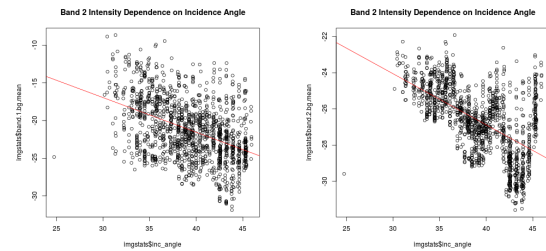


Figure 3. Intensity Dependence on Incidence Angle

incidence angle. Those images whose incidence angle were not defined (NA) were not altered (i.e. assumed to have mean incidence angle). The adjusted data is shown in Figure 4.

- Simple convolution matrix filtering was implemented to achieve some rudimentary image processing. Specifically, 3×3 kernels were used for smoothing, sharpening, edge detection, and gradient. [6] These filters were used to automatically produce images during visual analysis to aid in new feature development. Examples are shown in Figure 5. Notably, the plots for band 2 indicate potential higher-mode dependency, but that was not explored further.
- A simple center-finding function was implemented to calculate the sum of a sliding window over the image, and the position which yields the largest sum is taken to be centered on the object of interest (OoI). Several of the features are calculated solely from a fixed-size mini-matrix centered on that position. That mini-matrix is herein referred to as the ‘target’. The intent was yield feature values that are more influenced by target pixels than by non-target pixels.
- Histogram of Oriented Gradient (HOG) descriptors provide excellent performance compared to other feature sets for human detection[7]. Hence, we decided to observe the performance of algorithm on Iceburg detection. The technique counts the number of occurrences of gradient orientation in localized portions of an image. The descriptors are computed on a dense grid of uniformly spaced cells and uses overlapping local contrast normalization for improved accuracy. We have used ‘HOG’ function from the package **OpenImageR** to compute HOG descriptors for both `band_1` and `band_2` images. We got the 9-point descriptor for each `band_1` and `band_2` image.

Hypotheses From visual analysis and research in the domain area [8] [9] [10], several hypotheses were developed which drove new feature experimentation.

- Ships are relatively sharp and angular, whereas icebergs are smooth and irregular, so differences in surface

backscatter intensities (mean, max, variance) may yield class information.

- Volume backscatter properties of ice and metal [10] are different, so comparisons between band 1 and band 2 may yield some class information.
- Icebergs (being made of frozen water) may have some observable characteristics that are similar to the surrounding water, whereas ships may not. Therefore, comparison of the OoI to the background seawater may yield class information.
- The vast majority of iceberg volume is below the ocean's surface, and the angle of the ice as it protrudes from the water's surface is likely to be off-normal, whereas that of ships is likely to be near-normal. Therefore, gradient and edge-detection approaches near the object-to-ocean transition area may yield class information.
- Incident angle is likely to have an effect on both the intensity and polarity of the return signal. [11]
- We have computed Histogram of Oriented Gradients(HOG) descriptor for both positive and negative observations. By Positive observation, we mean the observation for which response 'is_iceberg' is 1 and Negative observation is the one for which reponse 'is_iceberg' is 0. Intuition behind using HOG is to convert 75×75 image into a set of numeric values, where each value can be used as separate feature. By this phenomenon, we are able to extract 9 features per image, i.e. for $band_1 = 9$ and for $band_2 = 9$. This processed data is used to train regression model and predict test results.

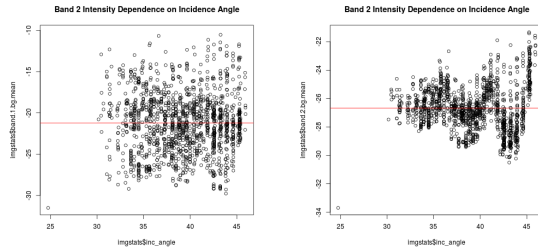


Figure 4. Normalized Intensity

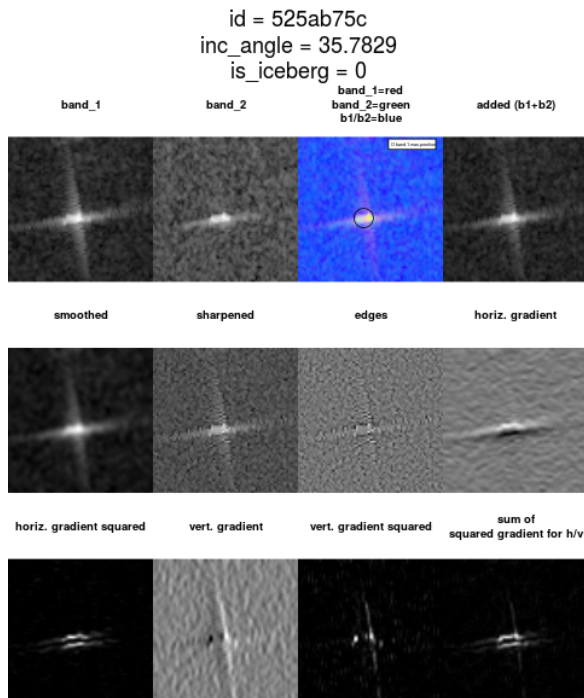


Figure 5. Image Filtering

4.2 Methods

Model Selection Several classification methodologies were considered for this classification task: Convolution Neural Networks (CNN), Logistic Regression (LgR), Random forest, XGBoost. CNN methods have been shown to exhibit good performance on image recognition tasks. [12] [13] Two challenges with Neural Networks are that (a) the more naive examples tend towards high-complexity and so require large data sets to achieve high accuracy and low variance, and (b) the hidden nature of the internal neurons pose challenges to applying useful domain knowledge to the classification problem. They work particularly well in applications where they learn telling features of a visible object within the hidden layers. In the context of this task, the size, shape, and orientation of the OoI is not believed to be as consistent and would be difficult to normalize, which would be highly desirable for a CNN approach. CNNs, therefore, were not implemented. [14]

LgR can be thought of as a special case of a Neural Network which has a single neuron that uses a logit function as its output. Since the single neuron is at the outer layer, its inputs and outputs are known. This makes it straightforward

to apply some domain knowledge to the raw data in order to provide inputs (features) that are thought to contain class information. Additionally, the low complexity of the model makes it ideal for smaller training sets, such as the one provided by Kaggle. [15] [16] The Logistic Regression Algorithm is shown in Algorithm 1. For this project, the authors used R's built-in `glm` function.

Logistic Regression Essentially, LgR uses an initial guess of feature weights to transform the data feature vector into a value between 0 and 1, representing the probabilities that each record belongs to class 1. A likelihood function and its derivative (with respect to the weights) are calculated, representing the curve of the likelihood that the weights are optimal over the domain of W . The derivative use used to influence the magnitude and direction for which the weight vector should be adjusted. The user-defined value α is also used to influence the step size of these adjustments. This processes is repeated until the gradient is sufficiently small, representing the peak (or valley) of the likelihood function, indicating an optimal weight vector has been reached.

Algorithm 1 Logistic Regression

```

1: INPUT: data  $X_{n \times (d-1)}$ , class labels  $C_{n \times 1}$ 
2: OUTPUT: weights  $W_{1 \times d}$ 
3: %% Define gradient descent parameters
4:  $\epsilon \leftarrow$  user threshold
5:  $\alpha \leftarrow$  user step size
6: %% Add ones column for offset
7:  $X \leftarrow [X_{n \times (d-1)} \quad \mathbf{1}_{n \times 1}]$ 
8: %% Initialize weights
9:  $W \leftarrow \mathbf{0}_{d \times 1}$ 
10: while  $|\nabla_W L(W)| > \epsilon$  do
11:   %% Class probability as function of weights
12:    $y_i(W) \leftarrow \frac{1}{1 + e^{-x_i W}} \forall x_i \in X$ 
13:   %% Likelihood as function of weights
14:    $L(W) \leftarrow -\sum_{i=1}^n c_i \log(y_i) + (1 - c_i) \log(1 - y_i)$ 
15:   %% Gradient of likelihood wrt weight vectors
16:    $\nabla_W L(W) \leftarrow \sum_{i=1}^n x_i (y_i - c_i)$ 
17:   %% Update weights
18:    $W \leftarrow W + \alpha \nabla_W L(W)$ 
19: end while
20: return  $W$ 
```

Random Forest Random Forests algorithm, creates multiple random decision trees using only important set of the features. It take samples from the training dataset with replacement, and trees are constructed by reducing the correlation between individual classifiers. Instead of choosing best split to construct trees, it uses random subset of features for each split. Advantage of using Random Forest Algorithm is that we don't have to prune trees to prevent over-fitting of the model on the training dataset.

We have used **randomForest** package for the Random Forest Algorithm. We have tuned following parameters of the

function to get better prediction.

1. `ntree` = Number of trees to grow. This should not be set to too small a number, to ensure that every input row gets predicted at least a few times.
2. `nodesize` = Minimum size of terminal nodes. Setting this number larger causes smaller trees to be grown.
3. `mtry` = Number of variables randomly sampled as candidates at each split.

XGBoost XGBoost(eXtreme Gradient Boosting) is an advanced implementation of gradient boosting algorithm. Boosting is sequential processing technique which works on the principle of ensemble. It gives improved prediction by combining multiple weak learners. At n^{th} instant, the model outcomes are weighed based on previous instant $n - 1^{th}$. The outcomes which are predicted correctly gets a lower weight whereas miss-classified predictions get higher weights.

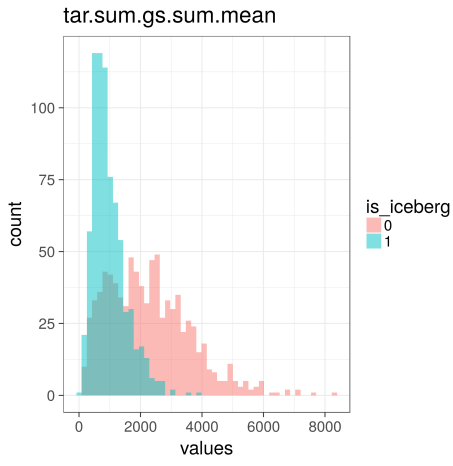
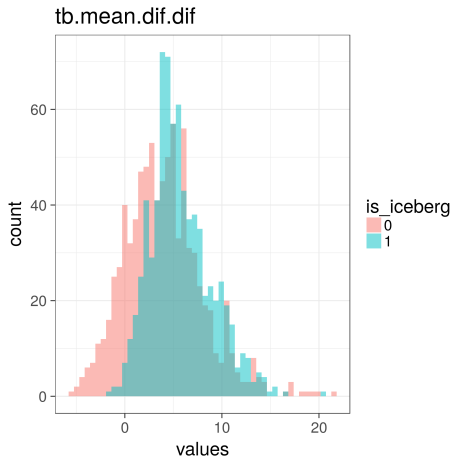
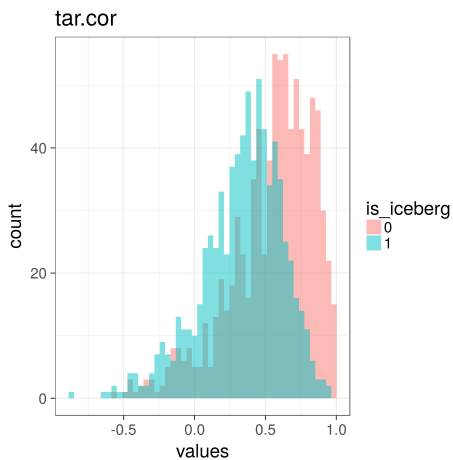
Highlights of XGBoost:

- 1 We have used **xgboost** package of R. Computational part of xgboost method is implemented in C++ and it can be multithreaded on single machine.
- 2 Customized Objective: XGBoost can be used to train model with custom objective function. Hence, user can define custom objective to improve the model.
- 3 Early Stopping: While building the decision trees, if the number of trees to be build is large then waiting time increases. Early stopping parameter when set, XGBoost will terminate the training process if performance is getting worst.
- 4 Handle Missing Values: XGBoost uses soft-coding method to handle missing values. When it encounters a missing value while splitting, it assigns direction instead of numerical value to the missing value.
- 5 Feature Importance: XGBoost tries to find best feature and splitting point to optimize the objective function. It calculates the information gain over the feature to decide feature importance.

Features The LgR modeling approach enabled experimentation of new potential features, which became the next focus. Table 1 summarizes, in some detail, a few of the more interesting features that were explored for LgR based on the previously mentioned visual analysis and domain research. Additional features are briefly described below.

- **band.1/2.tar.var** Variation of image pixels in the target for each band. The seed for this idea originated from Bentes [9] who recognized variance among

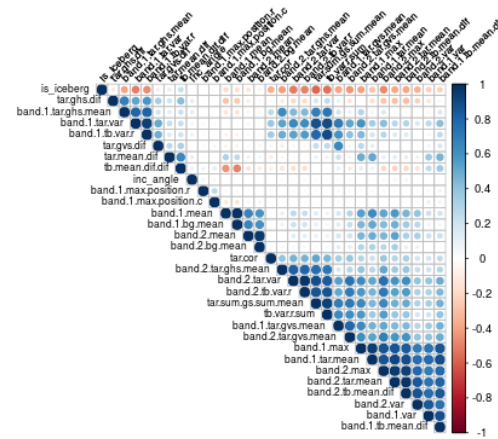
Table 1. Additional Features for LgR

Feature	Description	Histogram
<code>tar.sum.gs.sum.mean</code>	The target mini-matrix for each band is added together. The result is filtered by vertical gradient and horizontal gradient kernel in parallel. Those two results are then squared and added together, and the mean of the result is taken to be the new feature. The intent of this feature is to let both bands contribute to squared gradient result. Two theories are at work here: (a) The edges of ships are vertical and so one might expect a sharper ship-to-water transition, resulting in a higher gradient, and (b) since icebergs are more similar in texture and composition to water than are ships, that may also contribute to a sharper contrast between ship and water, again resulting in a higher gradient.	 A histogram titled 'tar.sum.gs.sum.mean' showing the distribution of values for two classes: 'is_iceberg' 0 (red) and 'is_iceberg' 1 (teal). The x-axis is labeled 'values' and ranges from 0 to 8000. The y-axis is labeled 'count' and ranges from 0 to 100. The teal distribution (iceberg) is highly concentrated at low values (below 1000), while the red distribution (ship) is broader and extends to higher values (up to 8000).
<code>tb.mean.dif.dif</code>	Visual analysis revealed most images to be rather centered on the OoI. A fixed-size border mask (think picture frame) was created to provide the subset of pixels presumed to reside in the background. For each band, the background mean was subtracted from the target mean. The result for band 2 was then subtracted from the result for band 1, and the final result was taken to be the feature. The intent was essentially to see if the difference in target-to-background contrast between bands may contribute to classification. Put another way, the depolarizing properties of ice may be similar to that of the surrounding water, creating a consistently lower contrast image in band 2 as compared with band 1 for icebergs.	 A histogram titled 'tb.mean.dif.dif' showing the distribution of values for two classes: 'is_iceberg' 0 (red) and 'is_iceberg' 1 (teal). The x-axis is labeled 'values' and ranges from 0 to 20. The y-axis is labeled 'count' and ranges from 0 to 60. Both distributions are centered around 5, but the teal distribution (iceberg) is slightly narrower and taller than the red distribution (ship).
<code>tar.cor</code>	The band 1 and band 2 images are first smoothed to reduce extreme values caused by noise or speckle and zero-adjusted. Next, all of the pixel values that are greater than 0.7 of the maximum value are kept, the others discarded. These pixels are then used as a mask to extract all of the pixels in the raw image which are presumed to belong to the OoI. Finally, a correlation is calculated between these pixels between band 1 and band 2, and the resulting value is taken to be the feature. The intent is to capture how well band 2 correlates to band 1 over the OoI. The theory is that depolarization effect would affect correlation and be different for icebergs than for ships.	 A histogram titled 'tar.cor' showing the distribution of correlation values for two classes: 'is_iceberg' 0 (red) and 'is_iceberg' 1 (teal). The x-axis is labeled 'values' and ranges from -0.5 to 1.0. The y-axis is labeled 'count' and ranges from 0 to 40. Both distributions are centered around 0.5, but the red distribution (ship) is slightly broader and taller than the teal distribution (iceberg).

other signal characteristics as harmful in a Neural Network approach and therefore something to be reduced. The intent is to see whether there is class information found within the variance of the return signal from the target.

- **band.1/2.tb.mean.dif** The difference of means of the target and border regions of the image pixels for each band. The intent is to see if there is class information derivable from return signal intensity from the target as compared with the background.

- We tried various combinations of these parameters to obtain best results in terms of log loss. But higher values of cells and orientations resulted into higher order



of HOG descriptors and lots of features in processed dataset which affected the performance. Hence, we tuned these parameters as 'cells' = 1 and 'orientation' = 9 to get 9-point descriptor.

Lgr - Given the manual nature of the feature development and experimentation approach chosen, it was important to automate updates to the analysis, training, and validation approach to the greatest extent possible. Automation of these aspects enables rapid identification and incorporation of improvements to the feature set. To this end, all database interaction, image processing, and feature calculation was centralized to a single code source. An analysis script was run after each change to the feature set to quickly visualize some characteristics of new features. In addition to automation, V-fold cross validation was utilized as the primary means of model performance prediction. It should be noted that test data used in cross-validation was present during all data analysis tasks, which may introduce bias as a result in the model selection process. [17] Finally, a simple greedy algorithm was chosen to select which features to use during the training process. Essentially, starting from an empty feature set, the feature which (when added to the set) results in the lowest Log Loss on the test data during V-fold cross validation is added. A subset of the (now ordered) feature list which results in the lowest total Log Loss on the test data is then selected for use. The author notes this algorithm is not guaranteed to find the optimal set of features on which to train. In fact it was observed that some of the later Kaggle submissions (which represented a superset of features as compared with past submissions) achieved worse performance! This is because the optimal subset of features was never encountered during the greedy selection process.

Algorithm 2 LgR Feature Selection

```

1: Def:  $vFoldXVal(\Delta[features]) \rightarrow LogLoss_{test}$ 
2: INPUT: feature list  $F$ , labeled data  $\Delta$ 
3: OUTPUT: selected feature list  $F_S$ 
4:  $F_S \leftarrow \emptyset$ 
5:  $F_S.last \leftarrow NA$ 
6:  $F_S.lossmin \leftarrow \infty$ 
7: while  $F \neq \emptyset$  do
8:    $f_{best} \leftarrow NA$ 
9:    $f_{best}.loss \leftarrow \infty$ 
10:  for  $f$  in  $F$  do
11:     $loss \leftarrow vFoldXVal(\Delta[\{F_S\} \cup \{f\}])$ 
12:    if  $loss < f_{best}.loss$  then
13:       $f_{best} \leftarrow f$ 
14:       $f_{best}.loss \leftarrow loss$ 
15:    if  $loss < F_S.lossmin$  then
16:       $F_S.last \leftarrow f$ 
17:       $F_S.lossmin \leftarrow loss$ 
18:    end if
19:  end if
20: end for
21:  $F_S \leftarrow \{F_S\} \cup \{f_{best}\}$ 
22:  $F \leftarrow F \setminus \{f_{best}\}$ 
23: end while
24:  $F_S \leftarrow F_S[1 : F_S.last]$ 
25: return  $F_S$ 

```

HOG with LgR - As we have observed in initial analysis, the training data has 133 NA values particularly in 'inc_angle' feature which is around 9% of the training data. Removing these NA values would have resulted in less training examples compared to test data. Hence, we tried to impute these values using **mice** package. The package is robust in terms of imputing missing values. We have generated 5-possible set of imputed data.

Next, we calculated HOG descriptors for both band_1 and band_2 images for both training and test data. We divided training data set into training and validation set to check the performance of LgR. Finally, we performed regression on the test dataset and processed the prediction in a way to make submission on Kaggle.

HOG with XGBoost - As mentioned earlier, we pre-processed the data in similar way by imputing missing values and then computing HOG descriptors for both band_1 and band_2 images. We divided training data set into training and validation set to check the performance of XGBoost. We have tuned parameters of XGBoost to following values to obtain best results:

- **nrounds:** 100
- **max.depth:** 2
- **eta:** 1
- **nthread:** 2

HOG with Random Forest - We divided training data set into training and validation set to check the performance of Random Forest Algorithm. We have tuned parameters of **randomForest** to following values to obtain best results:

- **ntree:** 50
- **nodesize:** 10
- **mtry:** 19

4.3 Results

Processing Performance Feature calculation, model training and feature selection, and classification were performed on a personal laptop, the specifications of which are found in Table 2 for LgR and 3 for HOG LgR. Processing times for Lgr are shown in Table 4.

Table 2. Machine Specifications for LgR

Processor:	Intel Core i5-5200 @ 2.2 GHz x 4
Memory:	8 GiB
Operating System:	Ubuntu 16.04 LTS
Applications:	R and MySQL
R Packages:	RMySQL, rjson, grid, ggplot2, corplot, pROC

Table 3. Machine Specifications for HOG with LgR, XGBoost and Random Forest

Processor:	Intel Core i7-8550 @ 1.9 GHz
Memory:	16 GiB
Operating System:	Windows 10
Applications:	R and MySQL
R Packages:	ROSE, OpenImageR, rjson, grid, RMySQL, mice

Table 4. Processing Time (mm:ss) for LgR

Feature Calculation:	02:22
Feature Selection:	01:26
Model Training:	00:00
Test Data Classification:	13:04
Total:	16:52

LgR Feature Selection Performance One might assume that the feature selection ordering may well correlate well with those chosen by visual examination of the histograms, but that is not the case. Since some of the features inform on some of the others, once a feature is selected and incorporated into the model, the addition of a feature that is highly correlated to one already in the set would result in little performance improvement. The performance of the LgR model as a function of features added is shown in Figure 7. Overall, the performance on test data tracks relatively well to performance on training data, suggesting that overfitting is not occurring to a great extent for the first dozen or so features.

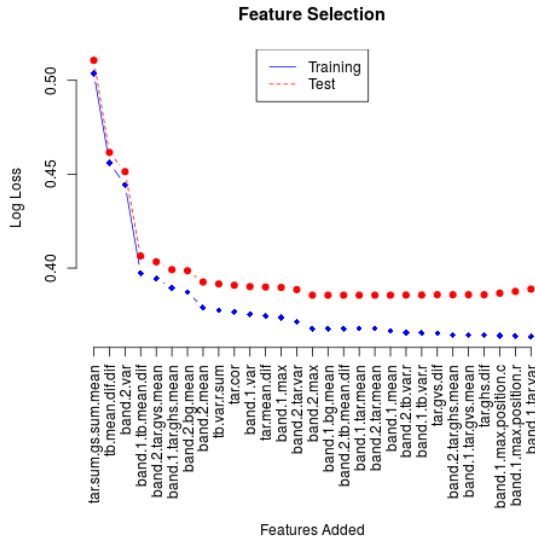


Figure 7. Performance During Feature Selection for LgR

Predictive Performance LgR - The best performance achieved as of the writing of this paper is **0.3682**, which is about the 18th percentile as of the submission of this paper.³ The Receiver Operator Characteristics (ROC) plot is shown in Figure 8.

HOG with LgR - When we used Histogram of Oriented Gradients with Logistic Regression, we are able to achieve **0.5505** score after submitting the predictions on test data to Kaggle. Figure 9 shows the ROC plot for the prediction made on validation data. Area Under Curve (AUC) for the model is **0.7133157**. Log-loss for the trained model is **0.6262356**.

HOG with XGBoost - When we used Histogram of Oriented Gradients with XGBoost, we are able to achieve **0.5986** score after submitting the predictions on test data to Kaggle. Figure 10 shows the ROC plot for the prediction made on validation data. Area Under Curve (AUC) for the model is **0.8170044**. Log-loss for the trained model is **0.6610249**.

HOG with Random Forest - When we used Histogram of Oriented Gradients with Random Forest, we are able to achieve **0.5345** score after submitting the predictions on test data to Kaggle. Figure 11 shows the ROC plot for the prediction made on validation data. Area Under Curve (AUC) for the model is **0.7778883**. Log-loss for the trained model is **0.5769474**.

4.4 Summary and Future Work

Summary The performance of the LgR classifier resulting from this effort was not exemplary. The methodology chosen

³Kaggle's scoring script uses Log Loss, which is consistent with the evaluation methods used herein

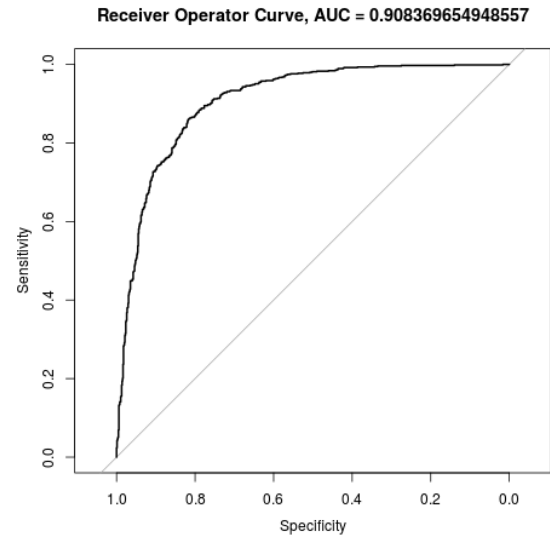


Figure 8. Logistic Regression ROC Results
Sensitivity represents icebergs correctly classified as icebergs, and specificity represents non-icebergs incorrectly classified as icebergs.

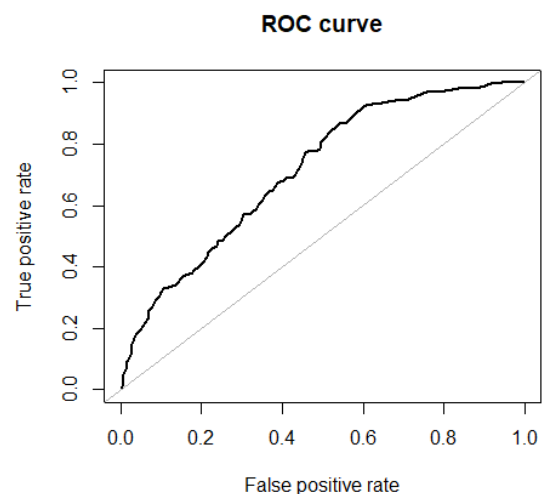


Figure 9. HOG with Logistic Regression ROC plot
True-Positive rate represents icebergs correctly classified as icebergs, and False-Positive rate represents non-icebergs incorrectly classified as icebergs.

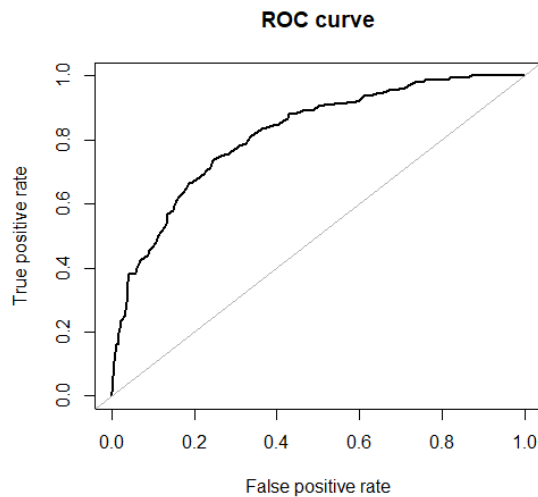


Figure 10. HOG with XGBoost ROC plot

True-Positive rate represents icebergs correctly classified as icebergs, and False-Positive rate represents non-icebergs incorrectly classified as icebergs.

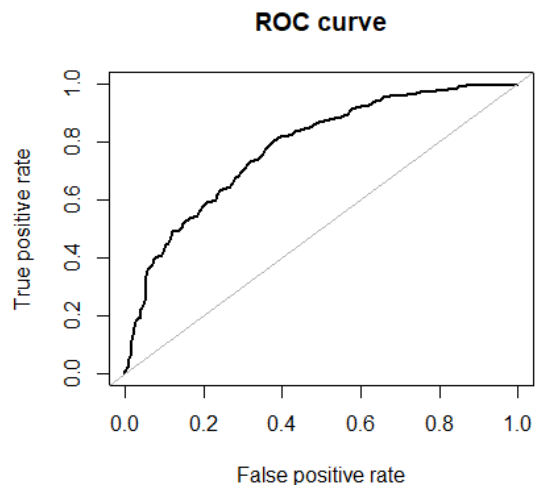


Figure 11. HOG with Random Forest ROC plot

True-Positive rate represents icebergs correctly classified as icebergs, and False-Positive rate represents non-icebergs incorrectly classified as icebergs.

for feature experimentation and calculation was manual and labor intensive, and frankly the problem is not within the author's domain of experience. The author hypothesizes that the high level of mutual information shared by many of the custom features ultimately limited learning performance of the model. This hypothesis is supported by both Figure 6 and Figure 7. The author considers the project a success, despite the average results achieved. The lessons learned about how to plan and structure a DM project as well as some of the practical aspects of implementing algorithms and dealing with real data will serve the author well.

Future Work Some potential future improvements to the LgR approach would be to adopt an entropy-based or correlation-based feature selection algorithm rather than the simple greedy algorithm described herein - that would enable the selection of features according to the information they contain about class, derived from their probability density functions or correlation to class label vs other existing features, respectively.

Further, one could revisit the Neural Network approach which would allow for nonlinearity and higher complexity. Neural Networks could incorporate both custom features (as described in the LgR approach) as well as image-oriented features such as the image processing filters discussed herein or the Histogram of Oriented Gradients. Additionally, some additional domain expertise applied to this problem would surely result in more relevant features which capture more of the known physical phenomena related to SAR imagery that could impact classification performance. The higher-mode behavior observed in the plots of background intensity as a function of incidence angle is just one example.

Acknowledgments

The authors would like to thank our professor, Dr. Mehmet Dalkilic, for his superior guidance and quick wit in the classroom; our parents for their continued recognition that we have not yet achieved our full potential; and Indiana weather, which provided ceaseless encouragement to stay indoors and pursue Datamining greatness.

References

- [1] Kaggle Inc. Spooky author identification.
- [2] Kaggle Inc. Statoil/c-core iceberg classifier challenge.
- [3] Lee B. Lusted. Signal detectability and medical decision-making. *Science*, 171(3977):1217–1219, 1971.
- [4] Xindong Wu, Vipin Kumar, J. Ross Quinlan, 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. *Knowl. Inf. Syst.*, 14(1):1–37, December 2007.
- [5] Statoil ASA. What we do.

- [6] Gianfranco de Grandi Jong-Sen Lee, Mitchell Grunes. Polarimetric sar speckle filtering and its implication for classification. In IEEE, editor, *IEEE Transactions on Geoscience and Remote Sensing*, volume 37.
- [7] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 886–893 vol. 1, June 2005.
- [8] Wolfgang Dierking Christine Wesche. Iceberg signatures and detection in sar images in two test regions of the weddell sea, antarctica. 58(208).
- [9] Carlos Bentes. Ship-iceberg discrimination with convolutional neural networks in high resolution sar images.
- [10] Carl Howell. *Iceberg and Ship Detection and Classification in Single, Dual and Quad Polarized Synthetic Aperture Radar*.
- [11] Juha Karvonen Marko Makynen. Incidence angle dependence of first-year sea ice backscattering coefficient in sentinel-1 sar imagery over the kara sea. 55(11).
- [12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.
- [13] Hierarchical neural networks for image interpretation.
- [14] Geoffrey Hinton Yann LeCun, Yoshua Bengio. Deep learning. 521:436–444.
- [15] J.A. Nelder P. McCullagh. *Generalized Linear Models*. 2nd edition.
- [16] A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms.
- [17] Nicola Talbot Gavin Cawley. On over-fitting in model selection and subsequent selection bias in performance evaluation. 11:2079–2107.
- [18] Using Maximum Entropy for Text Classification