Machine Learning Methods in

Predictive Arctic Cloud Classification

STAT 154: Project 2 Write-Up

Kulunu Dharmakeerthi *SID 3034365689*
Yigong Wang *SID26330853*
May 1, 2019

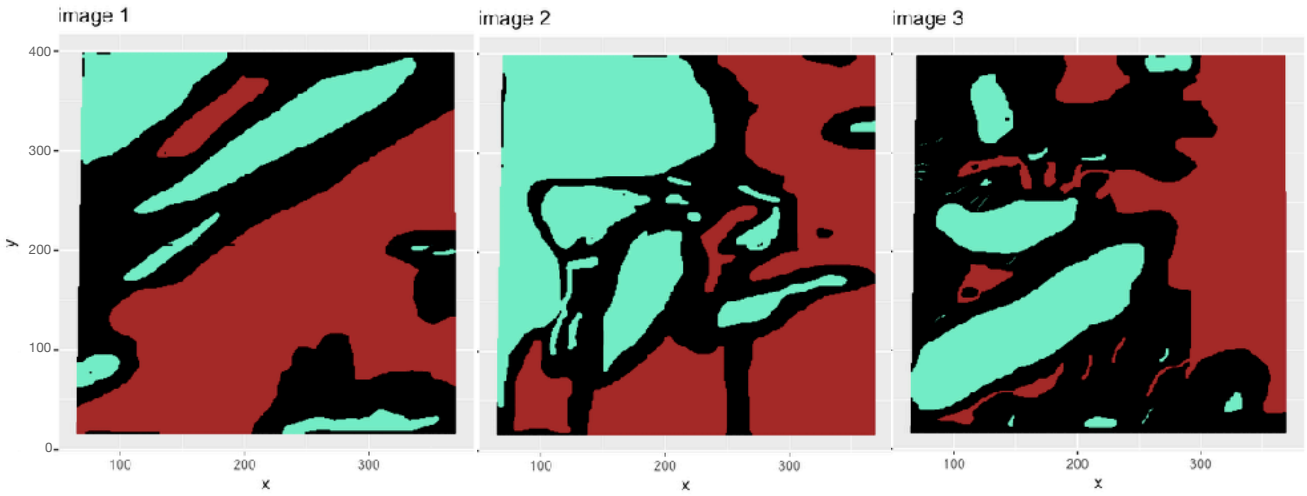
GitHub Submission

https://github.com/ryanwyg/Stat-154-SP19-Project-2

Data Collection and Exploration

(a) This research aims to target the classification problem of cloudiness over the arctic regions using existing MISR (Multiangel Imaging SpectroRadiometer) technologies. Using conventional methods (SDCM, ASCM and Offline SVM) runs the risk of not covering sufficient areas, not being able to discern clouds with terrain and ice, and not being able to run in a real time online environment. The new proposed method: Enhanced Linear Correlation Matching Algorithm, aims to detect clouds based on the detection of areas without clouds. This method first establishes three features based on EDA and domain knowledge: the linear correlation of radiation measurements from different MISR view directions (CORR), the standard deviation of MISR nadir red radiation measurements within a small region (SDAn), and a normalized difference angular index (NDAI). These would give enough information to separate clouds from ice and snow-covered surfaces. The ELCM algorithm then sets thresholds on each feature, and applying ELCM to each data unit would produce the "first cloud detection product". By training Fisher's Quadratic Discriminant Analysis (QDA) on the labels produced by ELCM, the "second cloud detection product" is formed in order to predict probability of cloudiness. These two products combined together, as the research team argues, produces more accurate measurement of cloudiness (even compared to offline advanced methods), offering up to over 90% accuracy and 100% coverage. The significance of this project is that this ELCM method offers a reliable way to efficiently analyse cloud information in the arctic regions, allowing further scientific research in this respect.

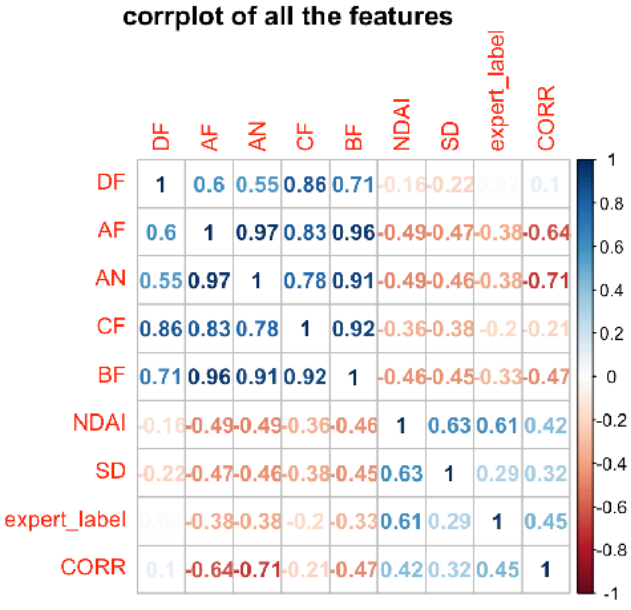
(b) The plots below are the plots generated based on x and y location information and the expert labelled data. The Light Blue represents high confidence cloudy; Red, high confidence clear; and Black, unlabeled pixels.



There are some noticeable trends: the smooth portions of the photo generally correspond to non-cloudy labeling, and the more rugged area responds to cloudiness in the expert marking.

i.i.d. does not make sense because the expert labeling relied on surrounding pixels to judge cloudiness, which makes each individual pixel dependent upon the surrounding ones.

"-1"(not cloud), "0"(unlabeled), and "1"(cloud) account for **39.43%, 31.41%, and 28.66%** of the expert labeling data.

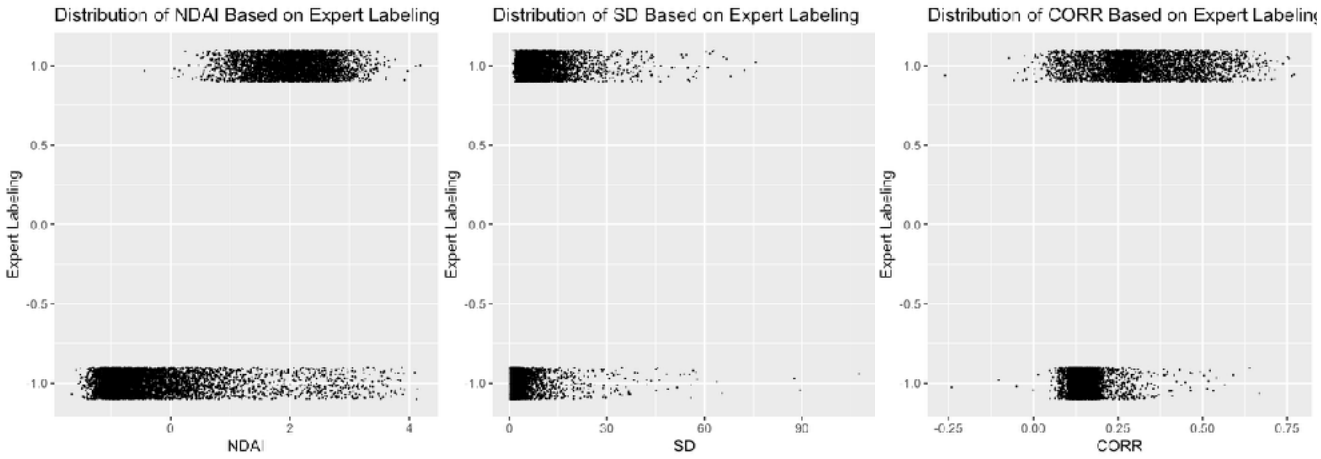


(c) Some quantitate and visual analysis can give us some introductory information.

Looking at the correlation pairwise plot of each individual features, we notice that **NDAI** has a strong positive association with expert labeling data. This makes sense as NDAI is related to surface radiation, which is directly associated with the discerning of low level clouds. The strong relation is expected as the North Pole area contains a lot of low level clouds (presumably in the three images provided as well.)

We notice the strong negative association between CORR and angel AN and AF, and since CORR is a direct reflection of cloudiness, one can expect such associations. However, the fact that other factors are not as strongly related is worth exploring.

Since the new ELCM method aims to predict cloudiness using the three features, we looked more closely at the relationship between the three factors and the expert labeling using the graphical jittering of data points, as the plots below would illustrate.



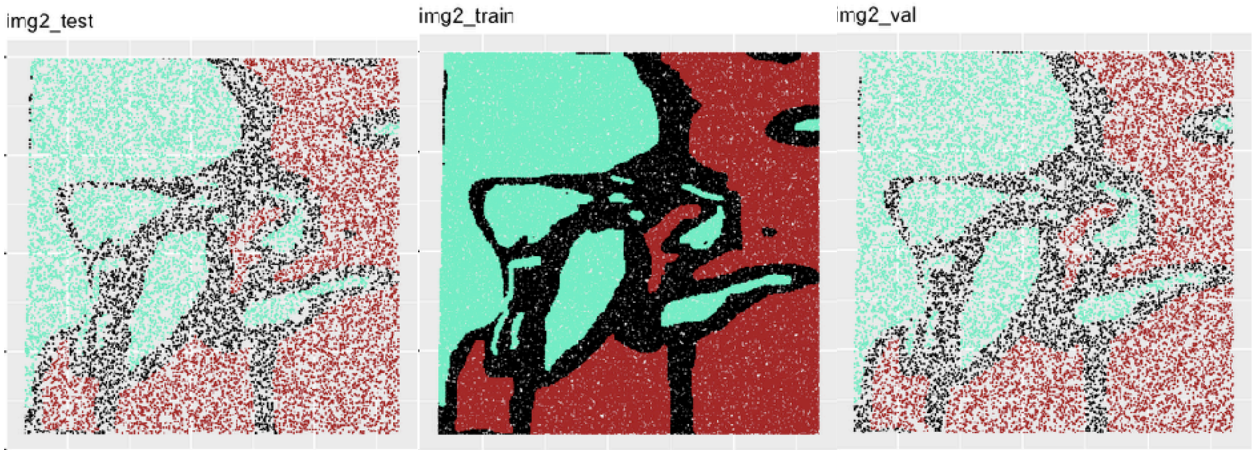
There is a strong positive association between NDAI and the expert labeling, where a high value of NDAI reflect more likeliness of being cloudy. More of the cloudy labeled points lie in the high NDAI ranges.

Next section will talk about the preparation of data to train the model.

Preparation

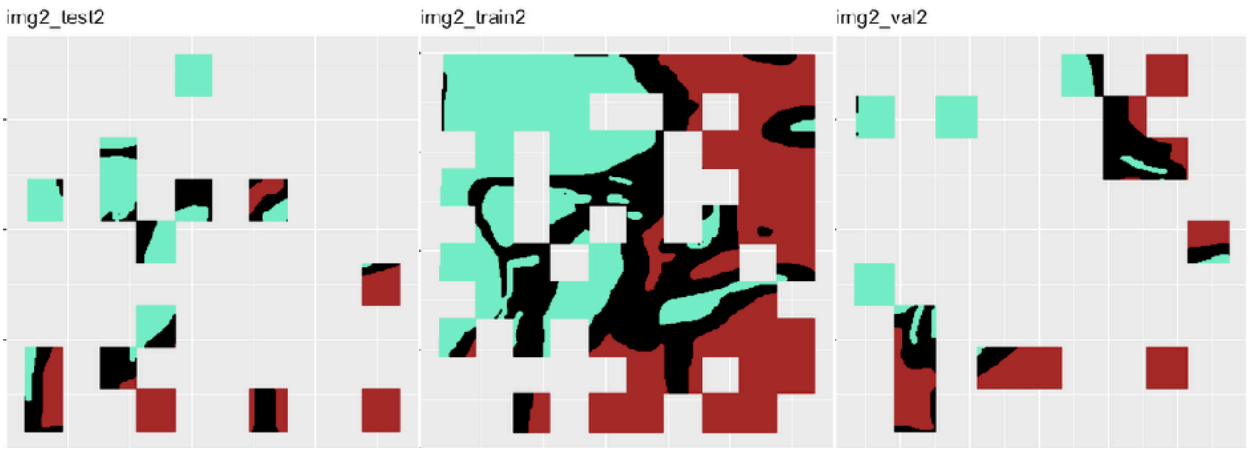
(a) The basic idea behind the splitting of data is to randomize each of the training, testing, and validation sets in the three images. Hence the fundamental idea in achieving this is to split the data into 10x10 grids and randomly assemble the three sets from these grids. Method 2 directly assembles the data set using the grids, and method 1 randomized further within the grids. Sample selections of image 2 is depicted for better illustration of our methods.

method 1



We randomly select points within each of the grids, and assemble the test, training, and validation sets accordingly.

method 2
The second way to split the data is to split each individual image into 10x10 grids. We then randomly select 30 grids as *test*, and the remaining 70 as *training*. We then select 15 grids from the whole set as validation set.



For the remainder of the project, we choose *method 2* as our primary splitting method as the accuracies are higher for this method. (We will test this in part 3 "Modeling".)

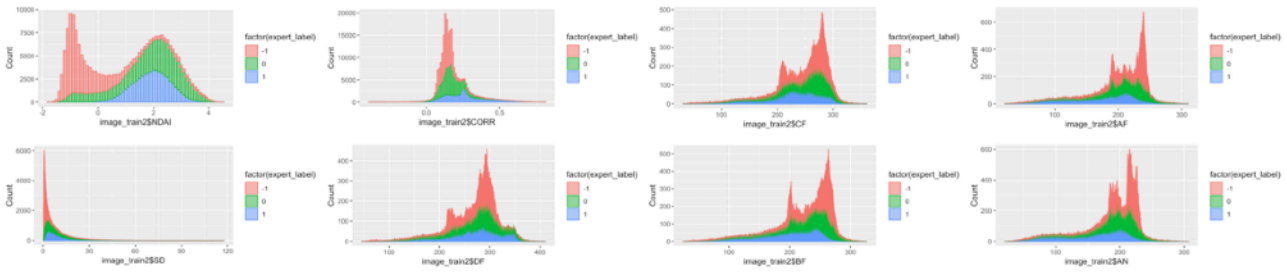
(b) In testing the accuracies of the trivial classifiers, we calculate the accuracies of setting values to -1 in both our methods, and the results are as followed (with the "0" expert label entries removed):

We can notice that the general accuracies of both methods are similarly low. Method two has a slight variation between the accuracies of the test and validation sets. This could be due to the grid separation provides large continuous areas, which generates inconsistencies between different grids.

	Method 1	Method 2
Validation	0.3690	0.2364
Test	0.3683	0.4723

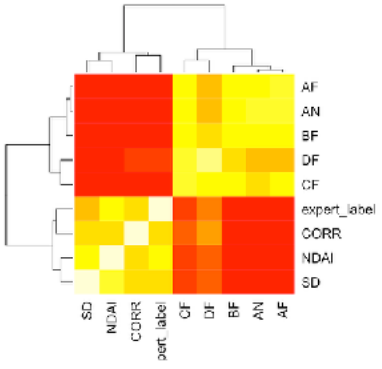
(c) We propose a 4 step methodology in selecting the "best" features:

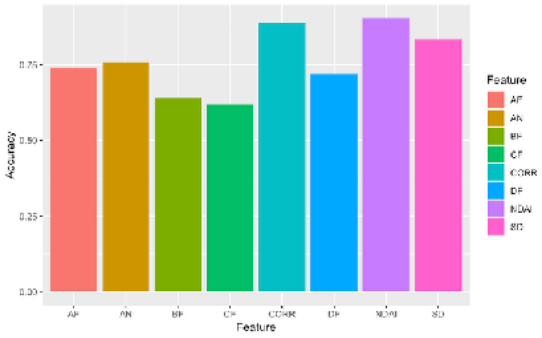
1. *Histogram with class separation to find any features which immediately appear to separate data:*



From this histogram, it is difficult to see if class separation will take place - perhaps NDAI;

2. *Look at correlation matrix, heat maps*:
Correlation plots/heat maps are enough if we are using a simple linear classifier. The heat map on the right and the correlation matrix in bottom of page 1 (**part 1b**) makes it easier to visualize





which features are good classifiers. Then plots are giving a direct visualisation of how model will use features. However, these are not enough as there might be some redundant features, which is the case in this scenario (the angel data are highly correlated with each other but provide little to no useful information for our classification of cloudiness.

3.Remove redundant features (highly correlated features): Removing features with correlation > 0.75 tells us to remove columns 7(AF), 8(AN), 6(BF), 5(CF), which leaves us with NDAI, SD, CORR.

4. *Run automatic feature selection for robustness*

   We train simple one feautre classification on each feature, to see which feature is best at predicting expert labels. This gives us NDAI, SD and CORR as the three most accurate features.

   As we see, this aligns with our correlation analysis. Not only are these features the least correlated, but also, if trivial classification is run (single feature prediction), NDAI, SD and CORR are the most accurate at predicting cloud class.

(d) Please refer to our *CVgeneric* function submission.

Modeling

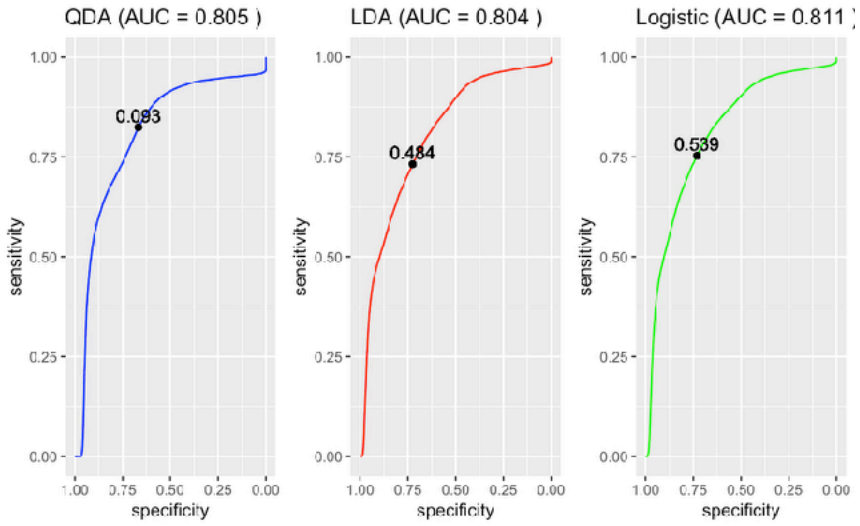
(a) For this part, we decide to use Logistic, LDA, QDA, SVM, and adaBoost to asses the fitted models. The accuracies of the test set is as followed:
To test the accuracies across folds for these methods, we developed methods where each classification method is tested for 10 folds (except for SVM and Logistic, which have 3 folds due to runtime issues). The adaBoost method has a tree depth of one. The accuracies and averages across folds are reported below.

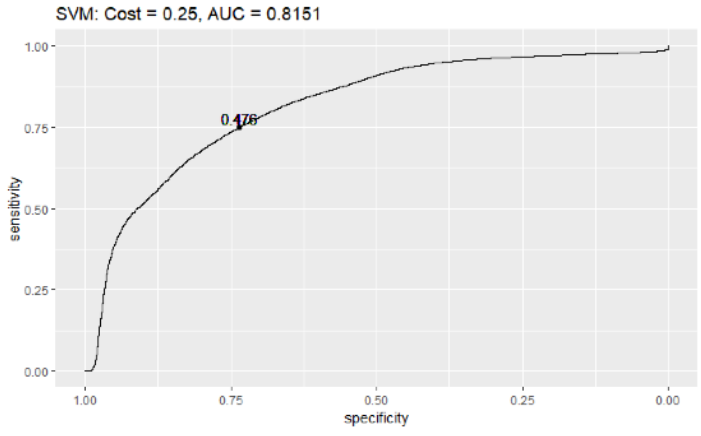
Accuracies	Splitting Method 1	Splitting Method 2
Logistic	89.49%	91.37%
LDA	89.92%	91.47%
QDA	89.76%	92.71%
SVM	90.15%	91.13%
adaBoost	90.82%	90.69%

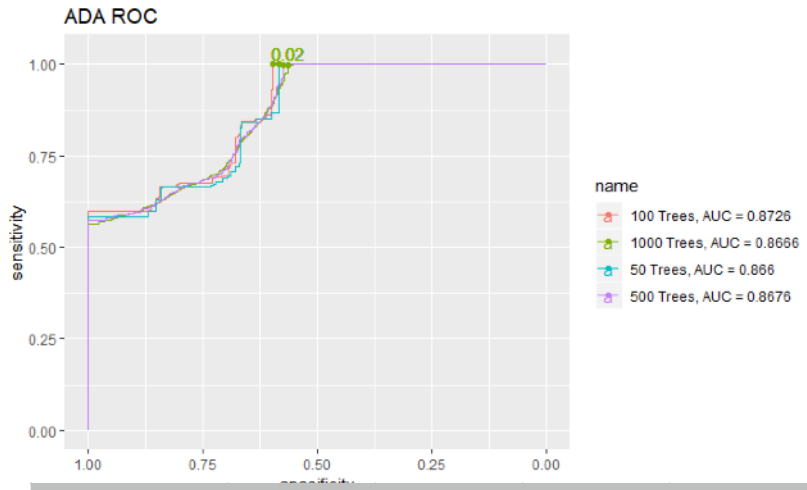
	1	2	3	4	5	6	7	8	9	10	AVG
LDA1	89.5%	88.5%	95.1%	89.7%	98.9%	87.5%	85.0%	95.4%	75.4%	93.0%	89.8%
LDA2	75.3%	87.2%	92.6%	90.9%	99.2%	92.5%	89.6%	92.4%	86.8%	89.3%	89.6%
QDA1	90.7%	82.5%	93.4%	95.6%	88.1%	87.2%	90.0%	94.1%	89.2%	83.3%	89.4%
QDA2	74.7%	97.6%	97.6%	89.8%	92.1%	83.9%	88.9%	85.2%	88.8%	87.1%	88.6%
ada1	81.8%	94.8%	88.0%	93.7%	96.5%	91.5%	93.2%	87.6%	84.8%	87.3%	89.6%
ada2	78.4%	89.0%	88.4%	93.3%	93.8%	92.0%	96.5%	96.4%	86.9%	85.6%	90.0%
Log1	89.9%	89.6%	87.5%								89.0%
Log2	88.7%	91.7%	85.6%								89.0%
***SVM1**	86.5%	91.5%	91.6%								89.9%
***SVM2**	89.5%	90.4%	88.6%								89.5%

From the table above, it is notable that all methods offer close to 90% accuracy across folds. The reason behind this is that our data set is very big (300,000+ entries), which makes the method less important. We do notice for some folds (LDA & adaBoost) the accuracies were poor (around 75%) for some of the folds, and this might be attributable to the gridding method (in method 2) and the random selection (in method 1). This is also why we select 10 fold for these two classification methods. Overall the methods offer very good accuracies in their classifications. SVM is hence not preferred in this scenario due to its inefficiency. (very long runtime - ranging up to 10 hours.)

(b) We are comparing three main methods here: QDA, LDA, and Logistic.







For this part, we plot the ROC curves for method 2 as that is our preferred method. The figures on the left are the plots for all the methods mentioned in part a. (see appendix and code for plots using method 1)

We select cutoff values using Yooden's index J. Youden's J is calculated by:

J = sensitivity + specificity - 1

expand the formula we get:

J = (true positives)/(true positives+false negatives)+(true negatives)/(true negatives+false positives)-1

The cutoff value is when this J index is at its maximum, or when the vertical distance between the ROC curve and the chance line (diagonal line) is the maximum. These points are noted on the plots.

The following table summarizes the cutoff values for the different methods.

Note that adaBoost has 4 different ROC curves because we want to test the effect of different number of tress on its performance and accuracy. We will expand on this tree concept (in terms of number of trees and tree depth) in the next part.

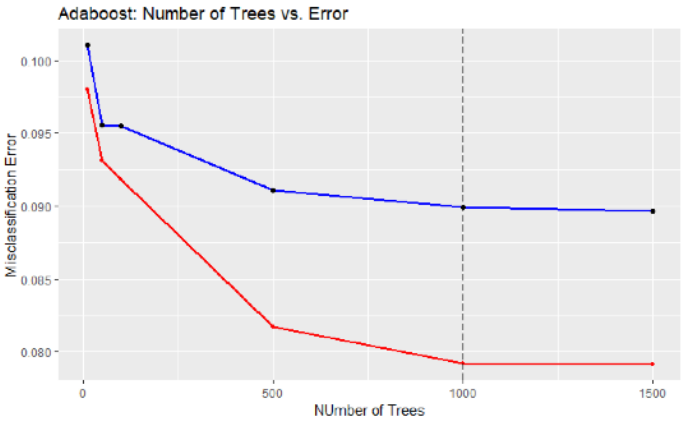
Method	QDA	LDA	Logistic	SVM	adaBoost
Value	0.093	0.484	0.539	0.476	0.020

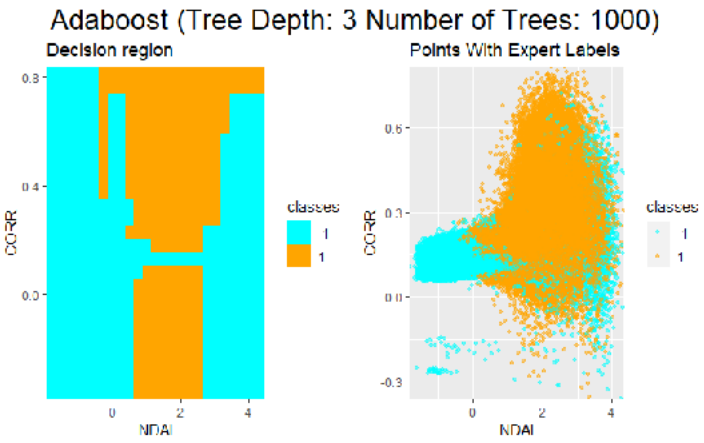
Diagnostics

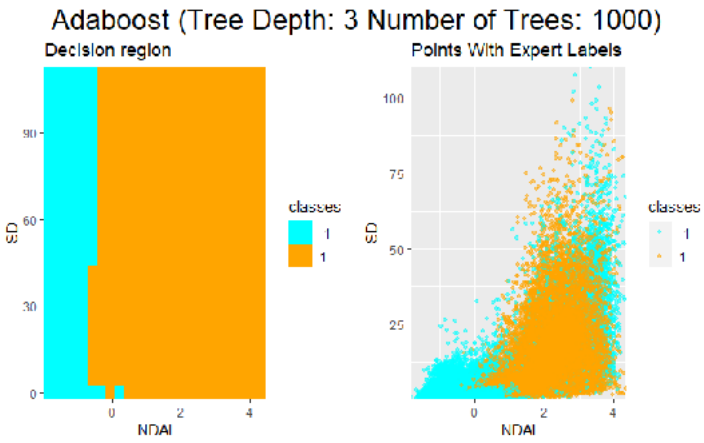
(a)

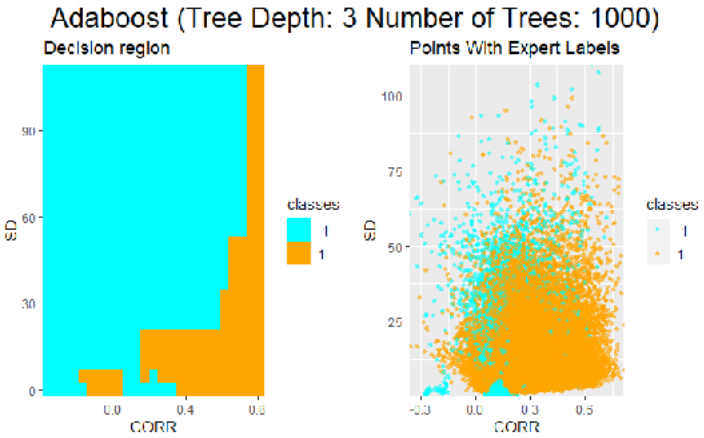
We choose the adaboost model, where specifically we decide to have **tree depth of 3 and 1000 trees**.

From the first two plots below (number of trees and tree depth vs. error) we can see that choosing such tree numbers and tree depth can produce the least errors and is the most efficient. While tree depth of 5 has the lowest test error, there is not much difference between level 3, and depth level of 5 takes considerably longer This model also produces good convergence as one can see from the following plots. Adaboost offers basic and continuous decision regions whereas expert labeling offers more scattered and complex regions. The general classification is good, however, and this method can predict areas where expert labeling would not be able to make a decision.



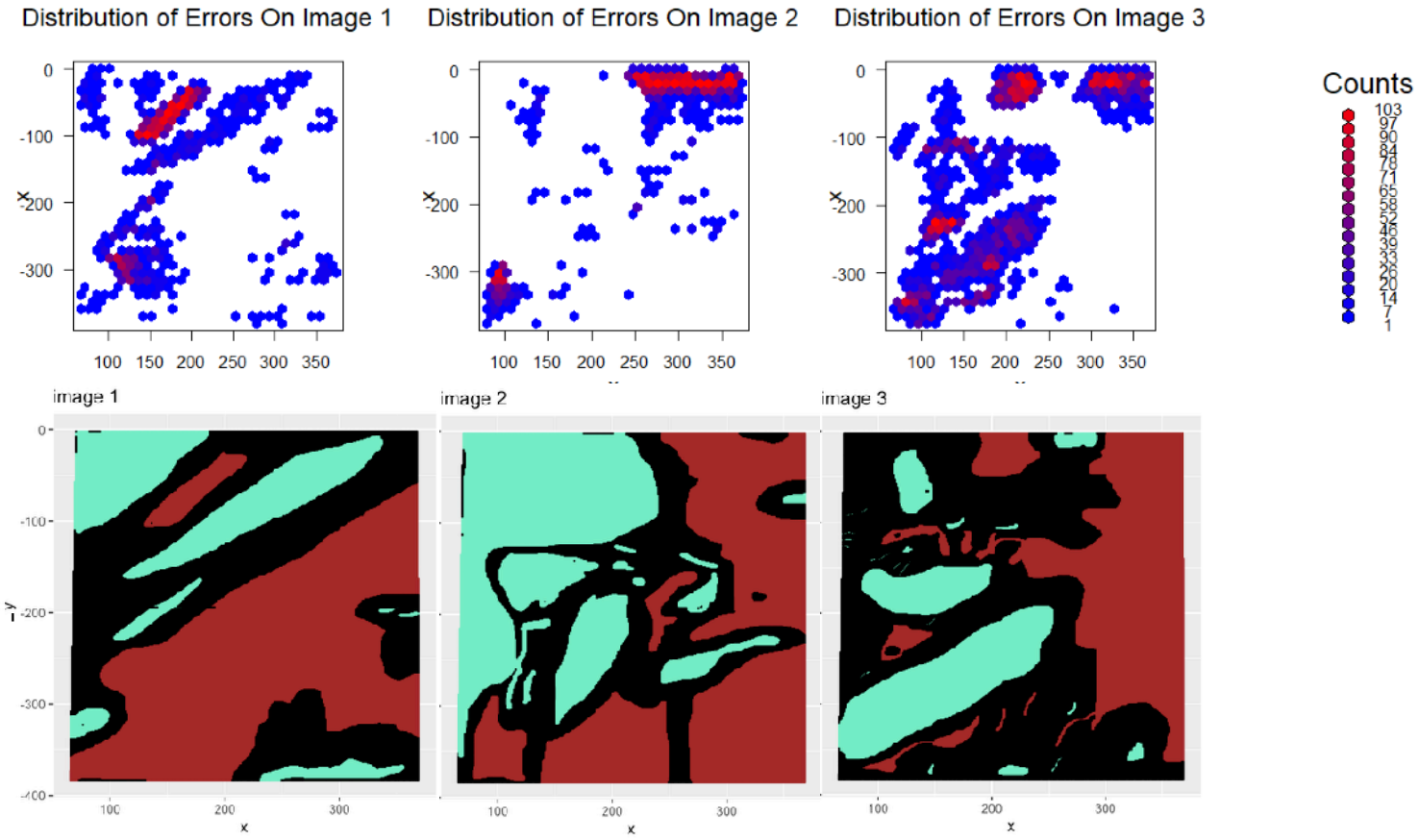




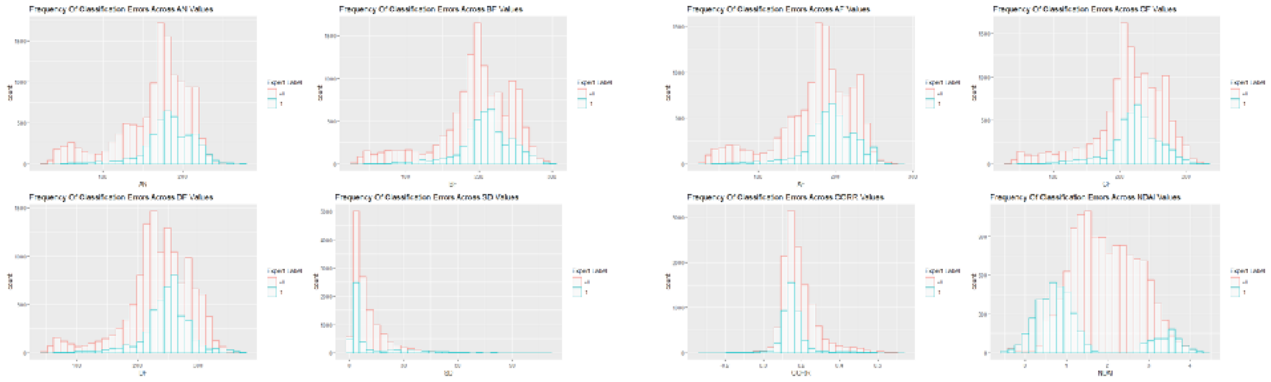


(b)

The first three plots are the misclassification errors generated by our model of choosing (adaboost: tree #1000, tree depth 3), and the red regions shows the areas with the most errors. Compared to the actual expert labeled plots we generated in the first part, we see that the error mostly resides in the cloudy portions. This might be due to continuous cloud regions offer uncertain feature attributes themselves, which requires improvements on existing technologies (new sensors) to solve.



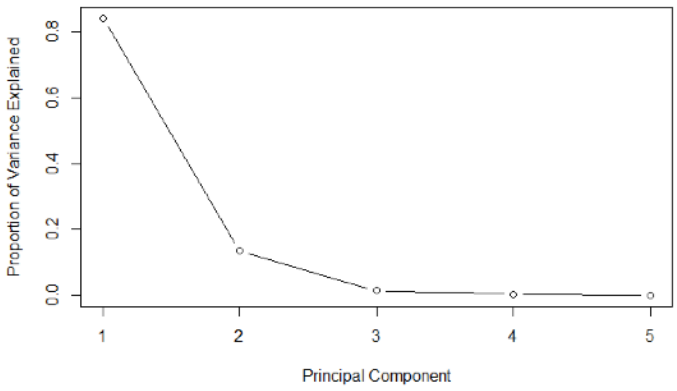
A detailed look at this frequency of classification error plots for each feature shows that NDAi has the most errors across the whole range of frequencies. However, this might be due to that being the feature we rely on and use the most. The other two important features, SD and CORR have very good error profiles.

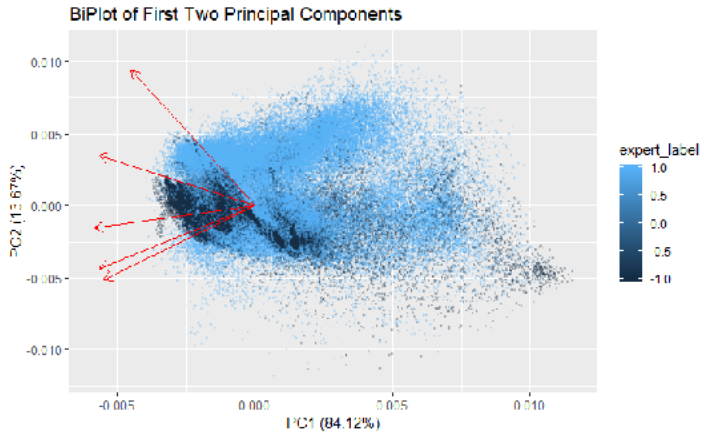


(c)
Here we outline three approaches we took in search of a better classification method.

PCA to enhance ADABoost

The fundamental problem with selecting from the 8 features given is that they are interrelated. Namely, NDAI, CORR and SD are derived from radiance angle measurements. This is one of the reasons we decided against having a combination of special features (NDAI, SD, CORR) and radiance features in the early stages of the project. However, an alternative is to construct our own 'special features' using the radiance features by using PCA. Selecting the three PCA components that most explained the variance in the data would perhaps form a better foundation for the adaboost model we have developed.

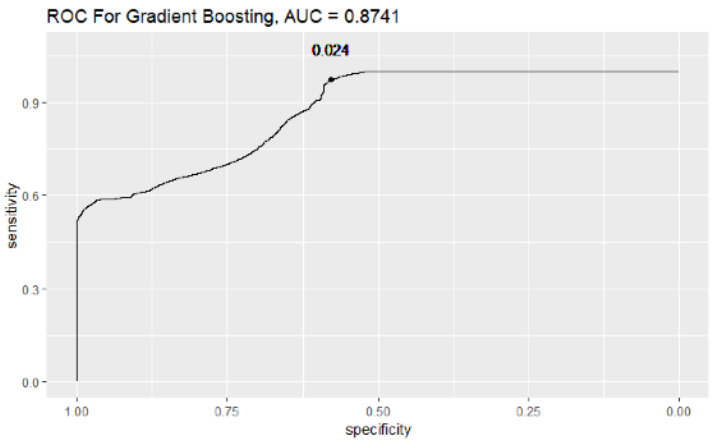


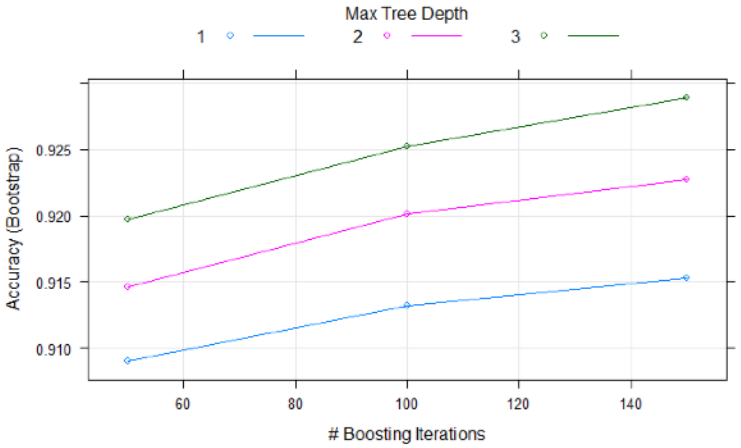


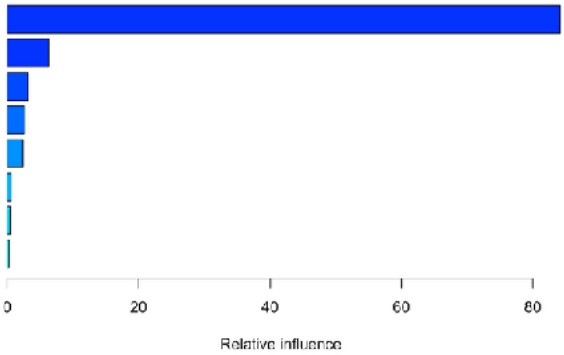
As we can see from the second diagram on the left, PCA has a lower test accuracy than adaboost, safe to say this method would do worse in predicting new image data

However, we find that PCA assisted adaboost performs worse than our original model. (Accuracy is 88.0%, which is slightly lower than adaboost itself.) Indeed, NDAI, SD and CORR are better suited features for the classification problem at hand. The possible reason is that the screeplot shown above shows what percentage of the radiance feature data is explained by our new components. They do not reflect how well our new components explain NDAI, SD or CORR, whose information would have factored into the expert labelling. We do not have sufficient access to (more than the equations given in the paper) the construction of the 'special' features to work around this.

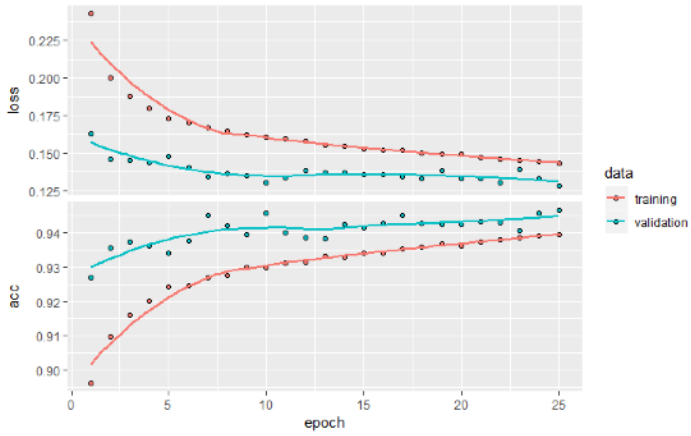
So we propose an alternative; to choose a better classification technique. **Gradient boosting** was selected as a candidate for replacing the adaboost model we developed previously. Both gradient boosting and adaboosting initialize a strong learner (decision tree) and iteratively build on weak learners (decision stumps, or small trees), thus, gradient boosting seemed a natural alternative to try. Note that it has the highest Area Under the Curve out of all methods.







Error table/Accuracy vs boosting iterations plot above: Clearly we find at iteration numbers at 150, gradient boosting performs better than adaboost. Test accuracy implies gradient boost is better than adaboost at predicting new image data.
Feature plot above: Our suspicions are confirmed: it is risky to disregard NDAI/CORR when moving forward with classification.



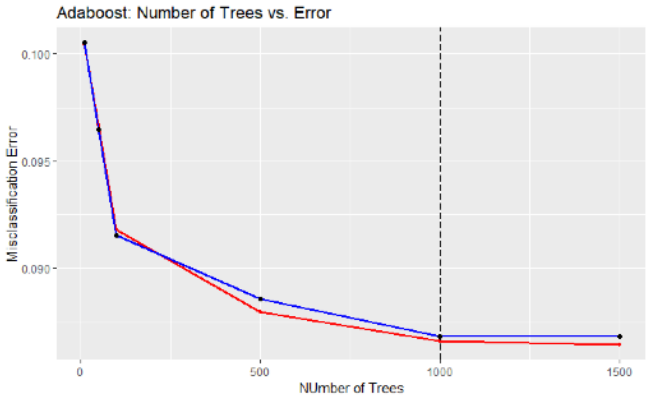
Neural Network
Here we show the results of a simple neural network based classifier. Neural network was initialized with a set of logistic layers. Dropout layers were added to help with overfitting we found in the initial runs of the program, and to prevent descent from stalling at local minima.
Plot/table
Although training and validation accuracy is better than adaboost at higher epochs, the test error is lower. Surely the neural network could be improved, but for now , we decide that gradient boosting is a viable alternative to the adaboost classifier we have developed.

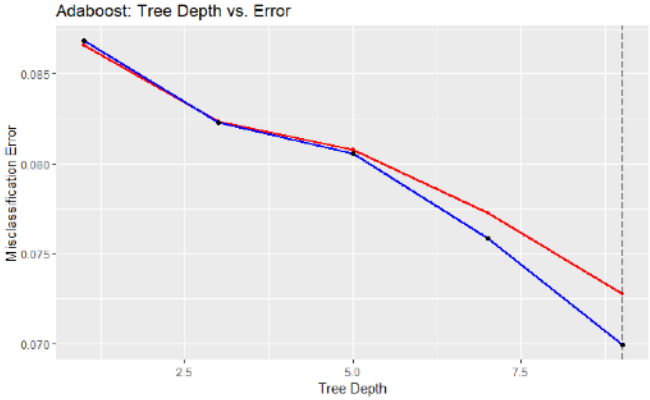
These three new propositions have their own drawbacks, but they all improve on the existing adaboost method. The models should work with similar accuracy which is very good as it allows for close real time prediction of cloud data with a very nice coverage. It is not 100% accurate as current technology (the data source) holds its limitation.

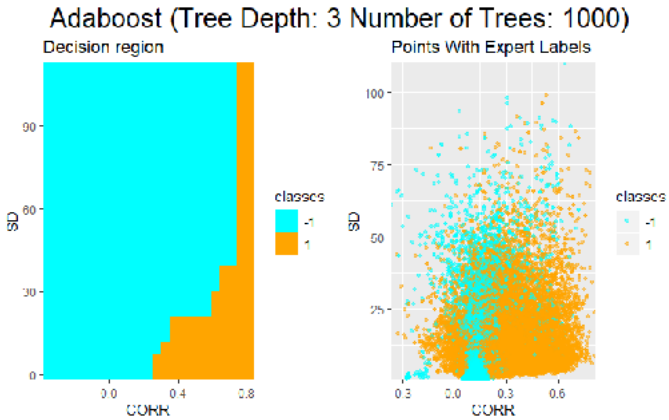
4d) - figures are on the next page
The most significant changes can be seen in the parameter convergence plots below (first two plots). Where previously our tree depth effectively converged at level 3, the plot shows that this is not the case if using the other split method. This is not ideal, as this means to optimize our model we will have to venture toward deeper trees, which will drastically increase the runtime of our method

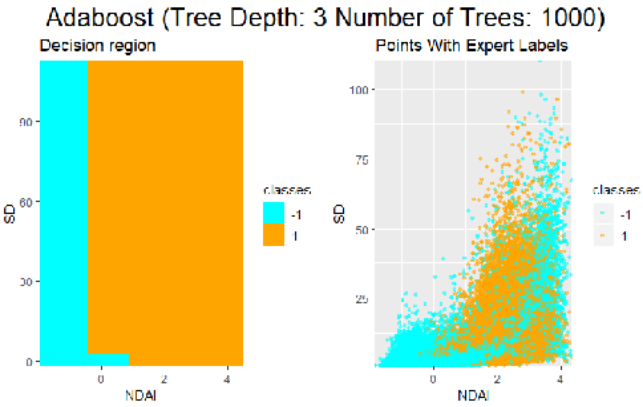
Feature Boundary plots are incredibly similar to previously.

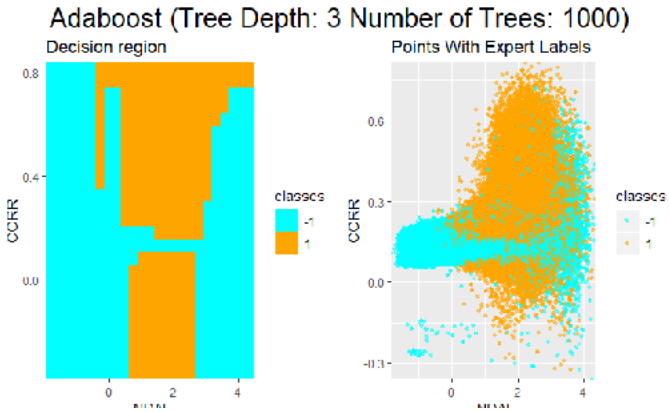
However, looking at our two dimensional map of errors, we see that there is a higher frequency of errors when we train the model on split method 2 data. Curiously, the maps are extremely similar in that the errors occur approximately in the same regions. This reaffirms the similar decision boundaries we observed in this section. Where they differ is that these regions are slightly larger for split method 2. Observe the cluster of points on the bottom of image 3 and the middle of image2. Compared to the map from our main splitting method, we see these clusters are larger.











Method 1 (4d)

Method 2 (previous)

Conclusion (4e)

The aim of this project was to develop an effective classifier for identifying cloud and no-cloud regions in images of polar regions obtained by the NASA satellite Terra. Prior to testing different classification techniques, an intuitive method for splitting the data was developed to preserve, as much as possible, the spatial dependence of observations in the data, and the inherently non-i.i.d nature of the dataset. Two methods were developed, the preferred method involving splitting image data into grids, and randomly assigning grids to training, validation and test sets. Identifying three key features with a combination of correlation analysis and simple classification methods allowed us to reduce the dimensionality of the problem and increase the efficiency of model testing. LDA, QDA, Logistic, SVM and Adaboost were candidate classification techniques, but by looking at k-fold cross-validation error, test error and validation error, Adaboost was ultimately selected as the tool of choice. Adaboost was tuned by looking at test errors, and 1000 trees with depth of 3 was identified as an optimal parametrization. However, in the final sections of the report we found that gradient boosting is a valid, and probably better method for classification. To that extent it would perform better in classifying new data without expert labels. Neural Networks are also suggested as a possible replacement, but their flexibility also means tuning them can be quite painstaking.

Contribution:
We appreciate the guidance of Prof. Yu's 2008 paper, the many great open source documentations of the R project, and the great mathematical knowledge from Wikipedia (Yooden's Index).
Kulunu: Coding for Part 2, and 4, Conclusion
Yigong: Writeup, Part 1, Part 3, Github Upload.