See from the Sky: Examining Cloud Detection Algorithms on the Arctic MISR Data

Yicheng Shen (yicheng.shen@duke.edu) & Yunhong Bao (yunhong.bao@duke.edu)

05 December, 2022

1 Data Collection and Exploration

1.1 Background & Motivation

With the global climates getting increasingly extreme, humans are making the best use of sciences and technologies to understand the environment, especially in the Arctic region. The detection of clouds in satellite images has become an important task, as cloud coverage is closely related to the surface air temperatures and atmospheric carbon dioxide levels. Yet it is a challenging problem since clouds are similar on snow- and ice-covered surfaces. In this study, we are going to examine various classification methods, build reliable models that distinguish the presence of cloud from Arctic satellite images using available features and evaluate our models' performance.

The data is obtained from a study by Yu et al. (2008). This team of researchers collected the data via the camera of Multiangle Imaging SpectroRadiometer (MISR) launched by the NASA. The data are in the forms of image pixels, with each MISR pixel covering a 275 m by 275 m region on the ground. Since standard classification frameworks of clouds were not readily applicable, their goal was also to build operational cloud detection algorithms that can efficiently process the massive MISR data set one data unit at a time without requiring human intervention or expert labeling.

Yu et al. (2008) proposed two algorithms, an enhanced linear correlation matching (ELCM) algorithm based on thresholding the three features with values either fixed or data-adaptive, and an ELCM algorithm with Fisher's quadratic discriminant analysis (ELCM-QDA).

Their results suggest that both proposed algorithms are computationally efficient for operational processing of the massive MISR data sets. The accuracy and coverage of ELCM are better and more informative compared with conventional MISR operational algorithms. This findings provides important implications and foundations for further analysis of MISR data.

1.2 Data Description

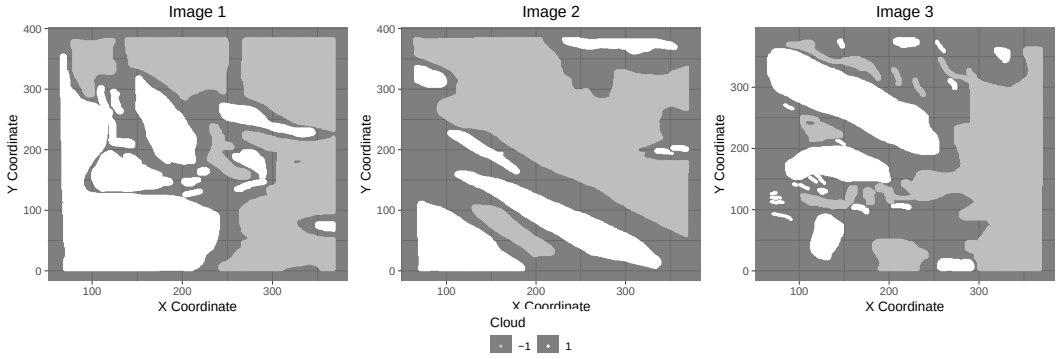

Figure 1: Maps of three images with expert labels. White represents high confidence of cloud; gray represents high confidence of clear; and dark represents unlabeled pixels.

In this study, we primarily focus on three of the MISR images. These three images contain 115110, 115229 and 115217 pixels respectively. However, not all pixels are labeled with confident experts' classification. As shown in in Figure 1, there are considerable portions of images not labeled. We notice the pattern that usually experts have difficulties distinguishing the areas around what they recognize as clouds. It is understandable that the borderlands between cloudy and clear surfaces are more challenging to determine.

We can also observe that labeled clouds are often clustered in chunks. Therefore, adjacent pixels' labels are not independent and instead seem to have very high positive correlations. For example, if a pixel's neighbors are all labeled as clouds, it is highly likely that it is also labeled as part of the cloud.

In Table 1, we present the percentages of expert labels in each image. Since we have no confident expert opinion on unlabeled pixels, they are viewed as missing / unknown values. After removing unlabeled pixels, we have 82148, 70917 and 54996 pixels in each image, with available information (labels and features).

Table 1: Proportions of Cloudy and Clear Surfaces by Expert Labeling

Cloud Labels	Clear Labels	Unknown	Image
0.3411172	0.3725306	0.2863522	1
0.1776549	0.4377891	0.3845560	2
0.1843825	0.2929429	0.5226746	3

In specific, each pixel has eight features. The first three are physically useful features for characterizing the scattering properties of ice and snow covered surfaces: the correlation (`CORR`) of MISR images of the same scene from different MISR viewing directions, the standard deviation (`SD`) of MISR nadir camera pixel values across a scene, and a normalized difference angular index (`NDAI`) that characterizes the changes in a scene with changes in the MISR view direction.

The latter five are five view zenith angles of the cameras. 70.5° (DF), 60.0° (CF), 45.6° (BF), and 26.1° (AF) in the forward direction and 0.0° (AN) in the nadir direction.

1.3 Explortary Data Analysis

Conducting a EDA on the available features gives a preliminary picture of the relationship within potential predictors as well as with the response. First of all, we notice in Figure 2 that there is positive correlation between the features. The correlation is particularly strong and positive among the five radiance angles. For example, `BF`, `AF` and `AN` are very strongly correlated. We also see positive correlations between `NADI`, `CORR` and `SD`.

However, the correlations between the first three features (`NADI`, `CORR` and `SD`) and five radiance angles (`Df`, `CF`, `BF`, `AF`, `AN`) are mostly negative.

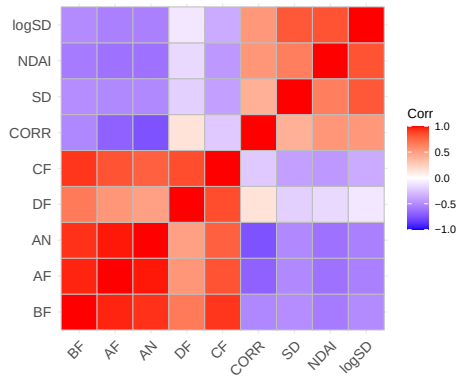

Figure 2: Pair-wise correlations between eight available features.

Figure 3 shows that for pixels labeled as cloud, their `NADI`, `CORR` and `SD` are likely to be higher than

those labeled as no cloud. This distinct pattern provides strong support to use these features as predictors of our classification models. It also seems reasonable to have a log transformation of `SD` so that it is in a similar scale as other features.

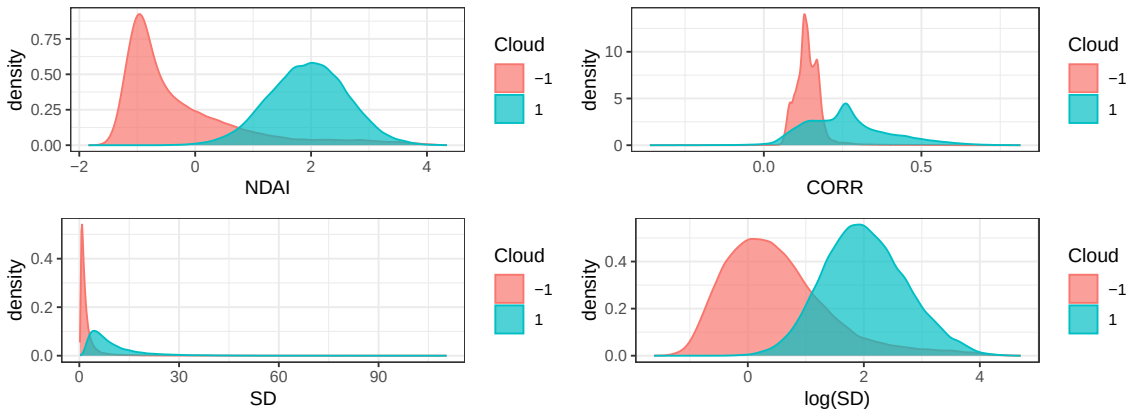

Figure 3: Density distributions of three features that describe cloud and clear pixels.

In terms of other features, we can see that the density distributions of radiance angles of cloud and cloud-free pixels are pretty consistent across angles in Figure 4. The radiance angles of cloud pixels are usually wider and right skewed, whereas the radiance angles of cloud-free pixels are usually higher and distributed in a bimodal shape. The distributions between cloud and cloud-free pixels are not as separable and distinct as the first three features.

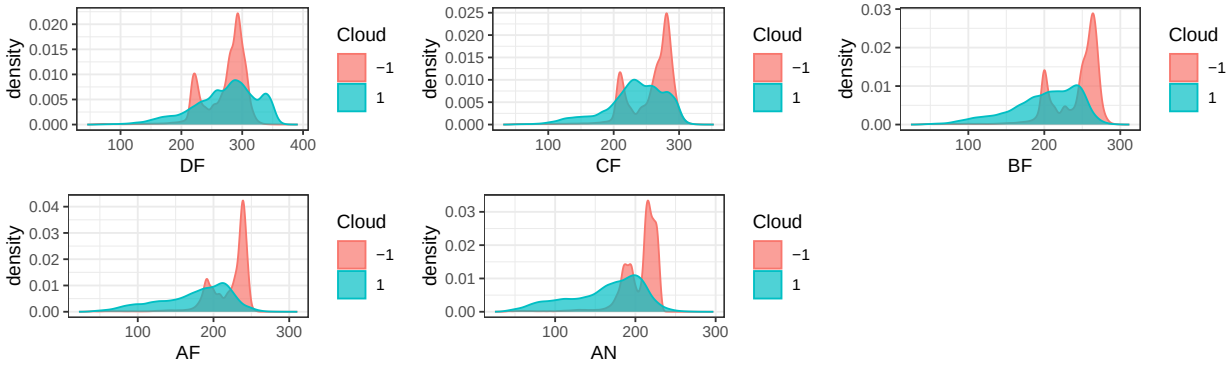

Figure 4: Density distributions of radiance angles from cloud and clear pixels.

2 Preparation

2.1 Data Splitting

Before building models, it is important to prepare the data as training, validation and testing sets so that we can make the best use of the data and evaluate our models. The key idea of our data splitting is to take into account the fact that this data set is not i.i.d. Therefore, we propose the two following ways of dividing data into blocks.

Method 1. Horizontal Cuts: The first method cuts each image horizontally in order to ensure that every resulting block has a reasonable portion of clouds and clear surfaces. Basically, each image is cut into five blocks by evenly dividing Y coordinates (shown in Figure 5), and three of them would be used as training data, the rest two blocks are used as validation and testing respectively. This methods splits the data into 58.59% training data, 19.03% validation data and 22.38% testing data (roughly 3:1:1).

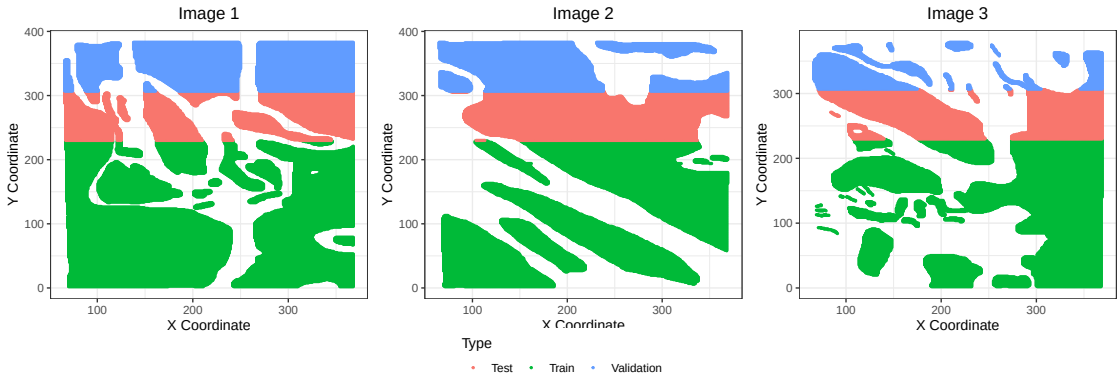

Figure 5: The first data splitting method is to divide each image horizontally.

Method 2. K-means Clusters: The second method of blocked data splitting is to use the K-means algorithm. By selecting a cluster size of five, we can divide each image's datapoints into five distinct groups (according to X-Y coordinates). Again, shown in Figure 6, three of these are used for training data, one is for validation and the last one is for testing. The K-means method splits all datapoints into 60.72% training data, 20.04% validation data and 19.24% testing data.

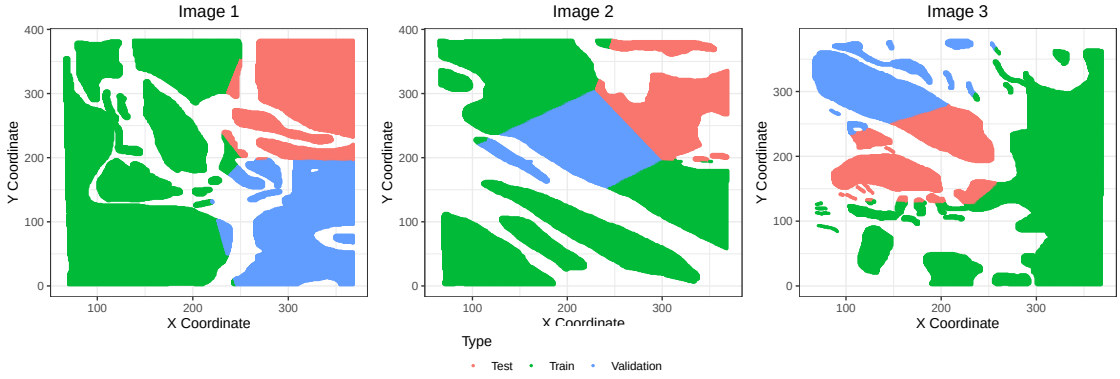

Figure 6: The second data splitting method is to divide based on the K-means algorithm.

The table below describes how many cloud pixels there are in each set after two ways of data splitting. We think they are both reasonable in terms of having both cloud and clear pixels in every subset of data.

Table 2: Proportions of cloud pixels in each set

Method	Type	Cloud Prop	Method.1	Type.1	Cloud Prop.1
Horizontal Cuts	Test	0.3384662	K-means	Test	0.3296281
Horizontal Cuts	Train	0.4778361	K-means	Train	0.4479412
Horizontal Cuts	Validation	0.1760129	K-means	Validation	0.2684817

2.2 Baseline Accuracy

We can examine the accuracy of a trivial classifier which sets all labels to -1 on the validation set and on the test set as shown in the table below. This classifier assumes that the image has no cloud pixels at all. Logically, the accuracy, or the success rate, of this trivial classifier depends entirely on the percentage of actual cloud free pixels labeled in the data. If the image is mostly cloud free, then labeling all points as -1 would easily achieve a high accuracy.

Table 3: Accuracy of a trivial classifier that sets all labels to -1

Method	Data Type	Accuracy
Horizontal Cuts	Validation	0.8239871
Horizontal Cuts	Test	0.6615338
K-means	Validation	0.7315183
K-means	Test	0.6703719

Here it is shown that labeling all points as -1 can only achieve an accuracy of 66.15% and 67.04% on the test data sets, which shows that to achieve high accuracy in this classification problem is not trivially easy. This also sets a baseline of reference for our subsequent classification methods.

2.3 First order importance

We consider `NADI`, `CORR` and `logSD` (notice that we apply a log transformation on `SD` as justified in section 1.3) as the three most important features when classifying pixels. Firstly, the EDA as shown in Figure 3 and 4 has shown that the density distributions of the first three variables are more distinct between cloud and non-cloud pixels compared with the angular radiance variables.

Then we also examine the pair-wise correlation between the response, `Cloud`, and each of the eight features. Looking at the two bottom rows in Figure 7, we notice that the correlation between `Cloud` and `NADI`, `CORR` and `logSD` are much stronger than the other five features.

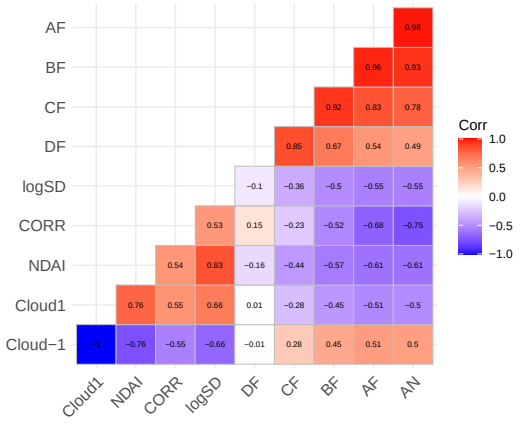

Figure 7: Pair-wise correlations between response (Cloud) and features (eight covaraites).

We also apply PCA on the eight features, which suggests that first three PCs could explain over 95% of the total variance. Among the first three PCs, `NADI`, `CORR` and `logSD` again consistently contribute a lot to the total variance. Thus, we tentatively suggest these three as the best features without model justification.

2.4 Cross Validation Method

After splitting training and test data, we use various classification methods to train models that can classify cloud and non-cloud pixels based on available features. To make the best use of the training data, we employ cross validation as a way to compare models.

As shown in our code appendix and github repository, our `CVmaster` function is able to take a generic classifier, training features, training labels, number of folds K and a loss function as inputs, then perform K-fold cross validation (CV) to assess the classification methods, and finally outputs the K-fold CV loss on the training data.

3 Modeling

In this section, different machine learning algorithms including logistic regression, LDA, QDA, Naive Bayes, and gradient boosting are used to construct the classification model. Hyperparameters are carefully selected base on cross validation accuracies. Performance is assessed through both CV error and prediction accuracy on the test data. A comparison across model fit is also conducted utilizing various performance metrics including the Area Under the Curve (AUC), precision, and F1 scores.

3.1 Model Fitting

Before fitting the chosen classification algorithms, we first assess different model assumptions and their fitness on the given data set. Logistic regression assumes linearity in the classification log odds and a linear decision boundary. Although such a strong assumption is not satisfied by the data, we can adjust the classification threshold to still achieve a desirable accuracy. LDA assumes predictors in different classes follow normal distributions with the same covariance matrix, Σ, but different means, μ_j, while QDA allows distinct class covariance matrices Σ_j. From Figure 3 and 4, we can observe that the distributions of covariants are hardly normal. Thus LDA and QDA's assumption of normality is unsatisfied here. Naive Bayes makes the assumption that conditional on the class labels, predictors follow i.i.d normal distributions. Such a assumption is also unsatisfied base on Figure 3 and 4. Tree-based models, for example the Boosting Trees, on the other hand, do not have any model assumptions, and should be able to provide good results.

After assessment of model assumption fitness, different classifications methods are applied to the data set. For Boosting Tree algorithm, hyperparameters including max-depth of each weak learner, number of weak learners, and the learning rate are tuned through K-fold validation. For algorithms that returns classification probabilities, we determine the model threshold by consulting the Youden statistics which maximized the sum of specificity and sensitivity. Using our `CVmaster` function and selecting hyperparameters with minimal cross validation error, we construct our final models on the entire training dataset and compute the respective test accuracies. Both CV and test accuracies are displayed in the following table:

Table 4: 10-fold CV Results and Test Accuracy based on Two ways of Data Splitting

Classifier	Data.Split	fold.1	fold.2	fold.3	fold.4	fold.5	fold.6	fold.7	fold.8	fold.9	fold.10	Cv.mean	Test
Logistic Reg	Horizontal Cut	0.6290	0.7970	0.7378	0.9947	0.8233	0.9879	0.9229	0.9571	0.9978	0.9448	0.8792	0.8847
Logistic Reg	K-means	0.9432	0.9973	0.7916	0.9963	0.8914	1.0000	0.8193	0.7226	0.6912	0.8900	0.8743	0.9386
LDA	Horizontal Cut	0.6830	0.8055	0.7082	0.9981	0.8046	0.9783	0.9012	0.9272	0.9983	0.9780	0.8783	0.8779
LDA	K-means	0.9619	0.9932	0.7916	0.9980	0.8274	0.9964	0.8431	0.6935	0.7418	0.8649	0.8712	0.9089
QDA	Horizontal Cut	0.6891	0.8274	0.6518	0.9989	0.8559	0.9926	0.8818	0.8983	0.9969	0.9863	0.8779	0.8957
QDA	K-means	0.7512	0.6402	0.8823	0.9978	0.8942	0.9990	0.8449	0.9705	0.8197	0.9966	0.8796	0.9031
Naive Bayes	Horizontal Cut	0.5574	0.8075	0.6151	0.9990	0.9580	0.9396	0.8071	0.8654	0.9877	0.9713	0.8508	0.8455
Naive Bayes	K-means	0.9340	0.9204	0.8083	0.9986	0.9618	0.9912	0.7678	0.6008	0.6447	0.8075	0.8435	0.8974
Boosting Trees	Horizontal Cut	0.7607	0.9934	0.6974	0.6578	0.9791	0.9245	0.9717	0.9878	0.8762	0.9985	0.8847	0.9538
Boosting Trees	K-means	0.9126	0.9320	0.9980	0.7327	0.9057	0.9994	0.9977	0.6807	0.7527	0.8988	0.8810	0.9579

As displayed above, Boosting Trees algorithm achieves the best CV and Test errors. This result aligns with our expectation as the Boosting Trees model has the weakest model assumption thus is able to fit data well. LDA, QDA, Naive Bayes, on the other hand, have a lower accuracy since their model assumptions are unsatisfied by the data. Logistic regression has a slightly better performance, an explainable result since its classification threshold is adjusted by Youden statistics.

3.2 Model comparision with ROC Curves

With final models being constructed, we comparison model fits with ROC curves and AUC values as in Figure 8. The decision thresholds (cut-off values used for predicted probabilities) for logistic regression and Boosting Trees are also plotted on the curve.

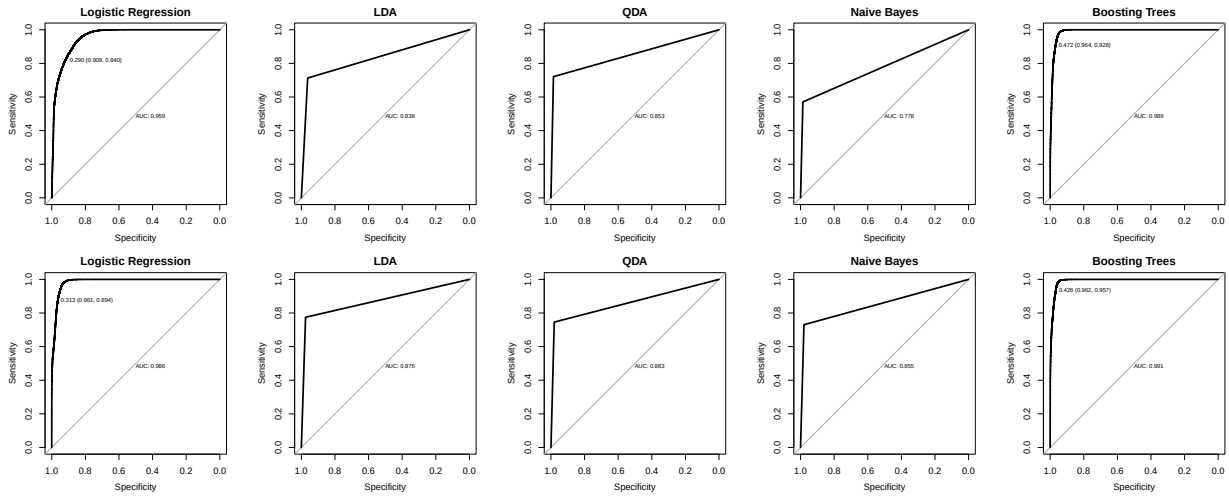

Figure 8: ROC curves on the test data (the first row is models trained on horizontal cut data, the second row is trained on K-means splitted data)

It can be observed from the above figure that Boosting Trees Algorithm has the best ROC curves and AUC values of 0.989 and 0.991. Its classification threshold is 0.472, slightly lower than 0.5. LDA and QDA have similar ROC curves with AUC values of 0.838 and 0.853, worse than Boosting as their normality assumptions are unsatisfied by data. Naive Bayes has the worst performance due to its strongest model assumption. A large change in AUC curve can be observed in logistic regression model, indicating its sensitivity to data splitting method might incur instability. Overall, Boosting has the best ROC value and is the most reliable model by this metric.

3.3 Comparision of Model Fit

Despite accuracy often being one of the most significant metric for classification models, other performance metrics provide valuable information concerning the overall model fit. In this section, we compare model performances based on precisions and F1 scores. Precision measure the percentage of true positive among all positive predictions and is calculated as

$$\text{Precision} = \frac{\text{True Positiveness}}{\text{True Positiveness} + \text{False Positiveness}} \tag{1}$$

A higher precision reflects a model's ability to avoid False Positiveness. F1 score, on the other hand, is

$$\text{F1} = \frac{\text{True Positiveness}}{\text{True Positiveness} + \frac{1}{2}(\text{False Positiveness}+\text{False Negativeness})} \tag{2}$$

A high F1 score indicates the model's ability to identify positive classses while minimizing error. The Precisions and F1 score of our final models are plotted in Figure 9 in descending order.

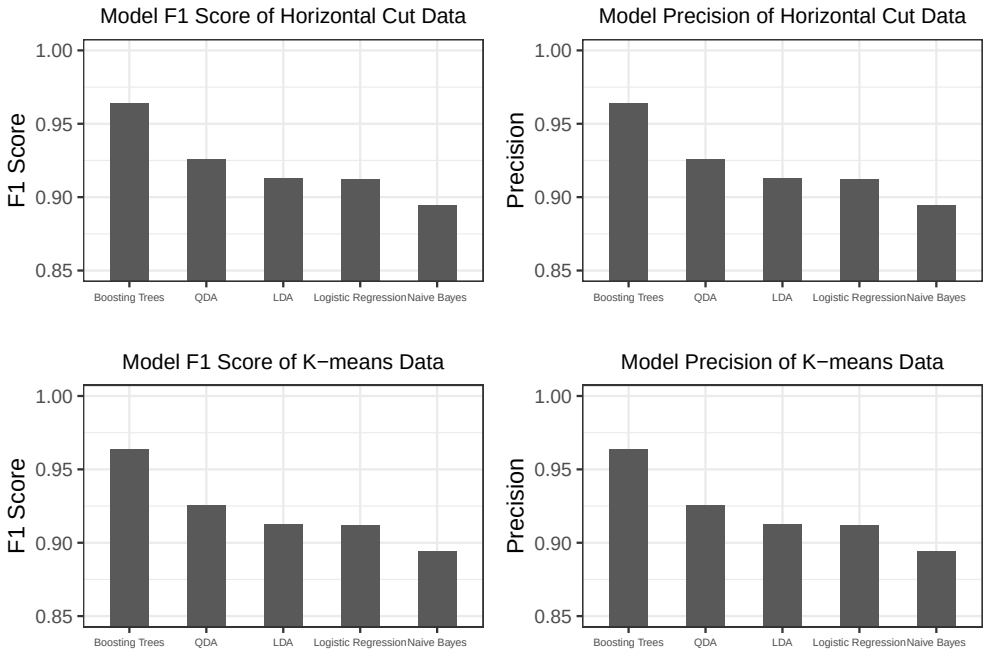

Figure 9: The F1 scores and precisions of five classification methods.

Boosting algorithm has the highest F1 score and precision under all circumstances, indicating it is the best model for this classification problem. Naive Bayes, on the other hand, has the worst performance. An interesting observation is that QDA and LDA has better model performance than logistic regression with respect to F1 score and precision. This results demonstrates that accuracy might be insufficient in determining model fit. Combining results from previous sections, we can conclude that Boosting is the best classification model for this particular classification problem.

4 Diagnostics

4.1 Examining Boosting Trees

In the previous model comparison section, it has been determined that boosting tree is the best model for this specific classification problem. We will further discuss the hyperparameter selection process and model convergence of boosting in this section.

There are three major hyperparameters in our Boosting Trees model: the depth of weak learners, learning rate, and number of weak learners. These hyperparameters are selected through the process of 10-Fold validation. Learning rate is adjusted in the range of 0.01 to 0.1, depth in the range of 2 to 4, and number of trees from 500 to 2500. CV result suggests that a learning rate of 0.05 with 2000 depth 4 trees produces the best model. With the best parameters in hand, we want to further assess model convergence by adjusting the number of trees and compare the respective test and train errors. In Figure 10, number of trees are adjusted from 1 to 10000 and misclassification errors are plotted.

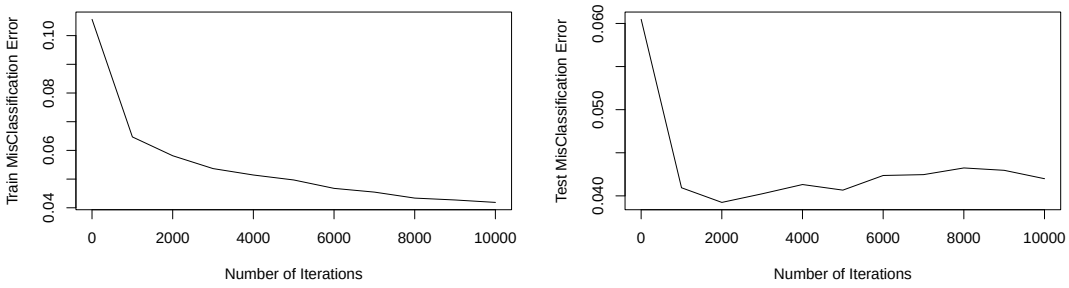

Figure 10: Missclassfication errros v.s. number of iterations

Distinct patterns can be observed in the two figure above. Training error maintains a decreasing trend as the number of weak learners increases. Since XGBoost put more weight on misclassfied points, each new weak learner aims to correct previous mistakes. Thus more weak learners would result in decreasing training errors. Test error, on the other hand, start to increase after number of weak learner exceeds 2000. This result aligns with our intuition as overfitting occurs occurs with growing model complexity, resulting in increased test error as shown in the test error plot. These two plots demonstrates that the chosen model parameter is the optimal that ensures model convergence while avoids overfitting.

4.2 Patterns of Missclassification

In this section, we conduct a detailed analysis of misclassifications. Figure 11 is a plot of model prediction on all the data points, with misclassified points plotted in yellow.

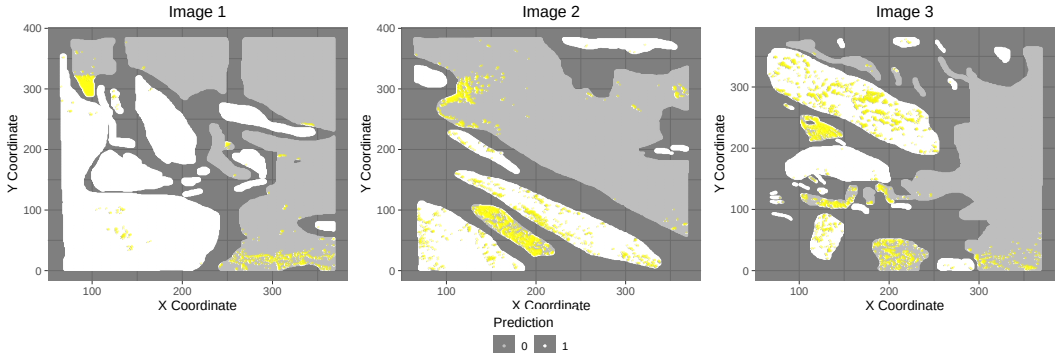

Figure 11: Prediction of cloud pixels in each image using a trained boosting trees model, with those missclassified points highlighted as yellow color.

Most classifications occur near the classification boundaries, where between class differences are less observable. Compare this result to Figure 14, 15, 16, it can be seen that areas with high misclassification error are indeed hard to differentiate base on covariate values. The confusion matrix of the proposed model is presented below.

Table 5: Confusion Matrix

	True Non-Cloud	True Cloud
0	122381	2713
1	4699	78268

A large proportion of the misclassifications (4699 out of 7412) occur as model misclassifies pixels of land into cloud (False Positive). To investigate the reason behind this phenomenon, we began by analyzing the most relevant covariates in the model.

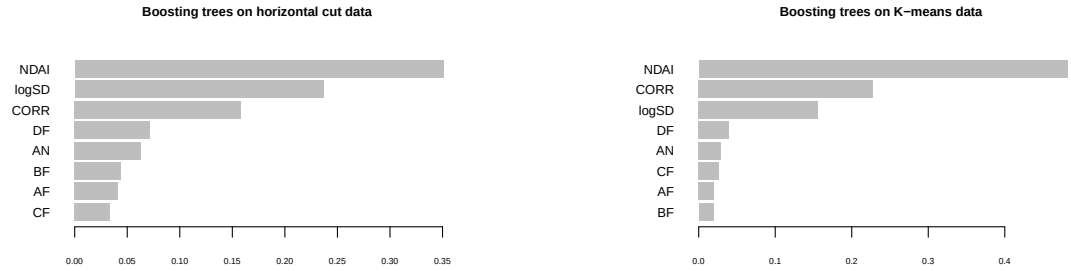

Figure 12: The feature importance plot in boosting trees.

Figure 12 is the importance plot of our boosting models. In both scenarios, `NDAI`, `logSD` and `CORR` are the most influential covariates. Figure 17 in the Appendix is a visualization of the first weak learner in the boosting model. The first split on `NDAI` is able to reduce total loss by more than 60 percent, indicating that `NDAI` is the most significant predictor. LogSD and `CORR` also further reduces the loss function by a observable amount. After identifying the most relevant covariates, we plot their density in the misclassified points.

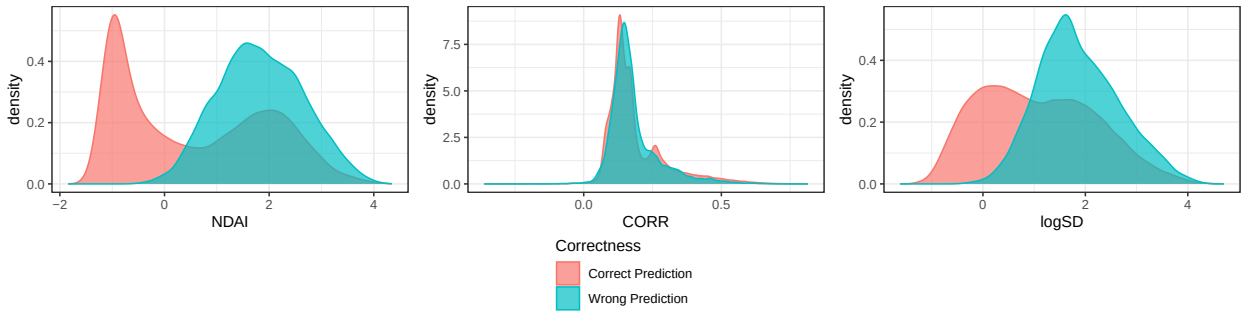

Figure 13: The feature density plot of correctly and wrongly predicted pixels.

Figure 13 is a covariate density plot, with red indicating correctly classfied points and blue being misclassified data. It can be seen that misclassification mainly occur in the upper range of `NDAI` and `logSD`, while equally distributes with respect to `CORR`. Such pattern suggests that models with even strong `NDAI` and `logSD` dominance might be preferable. Base on this result, we propose several ways to further improve our proposed classfier.

4.3 Model Improvements

Despite a high model accuracy, there are three potential improvements that can be made to further improve the proposed classifier. The first potential improvement is dimension reduction. As seen in the EDA plots, there are strong correlation between the angular covariates. A dimension reduction might be able to avoid repeatation of information and further strengthen the dominance of most relevant predictors. However, dimension reduction techniques such as PCA would complicate the interpretability of the model.

Another potential improvement is to add L_1 or L_2 regularization for model complexity reduction. A simpler model potentially reduces model variance and avoids over-fitting. Nevertheless, it would require more tuning of a proper regularization parameter and L_1 regularization performs variable selection arbitriarily.

Moreover, a combination of supervised and unsupervised learning methods might be beneficial. Observing that spatially close data points tend to share the same classification label we propose the following procedure for classification:

Step 1. Utilize pre-trained XGBoost Classifier on the desired data set to get the classification probability p_i for each data point i.

Step 2. Define an upper bound UB and lower bound LB, $UB>LB$, to be the classification thresholds. Classify data point i to 1 if $p_i \geq UB$, to 0 if $p_i \leq LB$.

Step 3. For the remaining data points j, where $LB < p_j < LB$, utilize KNN on geographical coordinates $(X, Y$ in this case) to classify point j to the class of its closet neighbors.

This methods would avoid false classification of data points within a cluster of same-class data and therefore increase the overall accuracy. However, it does not provide any performance improvements over datapoints on the cluster boundary.

Even without these improvements, the proposed XGboost model framework will have a robust performance in future classification of unlabeled data points. Since the employed block data splitting approach mimics providing a new image, the reported CV and test errors are reliable estimates of model accuracy.

4.4 Modefied way of data splitting

Similar to the discussion in the previous section, block splitting is a more reliable technique as test data sets can be viewed as new images to classify. If data is considered to be i.i.d, however, model behavior would be quite different. Suppose the test data is chosen randomly, test and training data would be geographically close to each other, thus have similar covariates values. This similarity between train and test data would result in a unreliable test error. For the same model convergence plot in Section 4.1, test error would keep decreasing with the training error there is no dissimilarity between testing and training data. Model evaluated under this metric would have weak performance due to overfitting.

5 Conclusion

Overall, there are various methods that we have tried to employ on this Arctic cloud classification problem. Among the methods discussed in this report, boosting trees yield the best performance on the training and test data. A well-trained boosting tree model could achieve over 95% accuracy when predicting cloud. Logistic regression is also a good method in this context, particularly for the less time it needs to train models. Other classification methods, for example LDA, QDA, and Naive Bayes, are less preferable, mostly because the structure of image data, with cloud and non-cloud pixels, does not satisfy the assumptions of these models. There are other methods that could potentially be examined, such as KNN, SVM, and random forest, which are suggested as future directions of our work.

6 Reproducibility

In order to ensure the reproducibility and applicability of our work on future MISR data, we have published our codes and model analysis on GitHub. Users may use the following link to visit our repository: https://github.com/sheny2/STA521-Project-2. The repository contains detailed codes and a README file, which illustrates our workflows as well as guides to reproduce our analysis.

7 Graphic Appendix

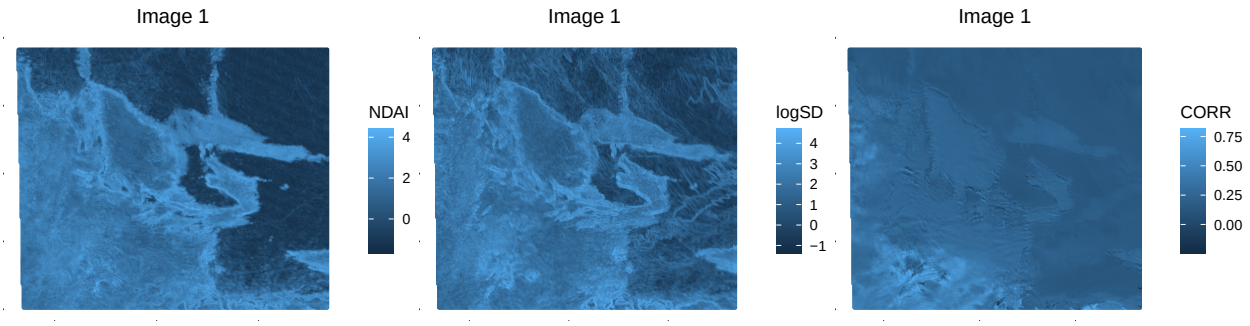

Figure 14: Important Features of Image 1.

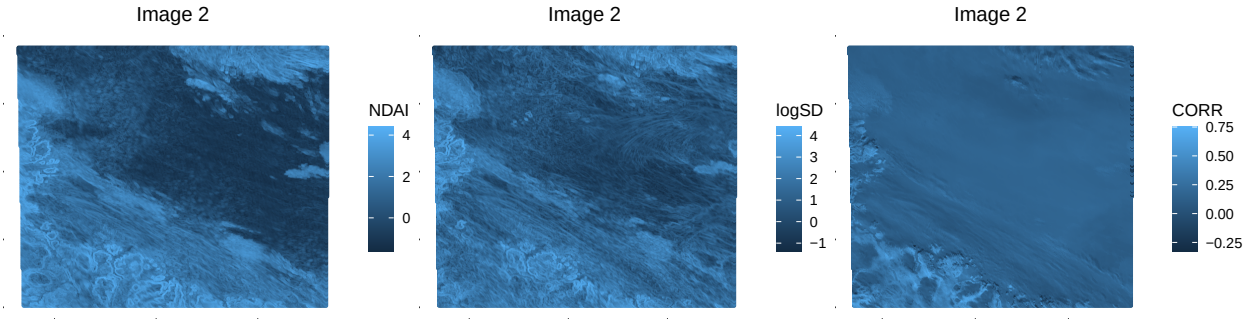

Figure 15: Important Features of Image 2.

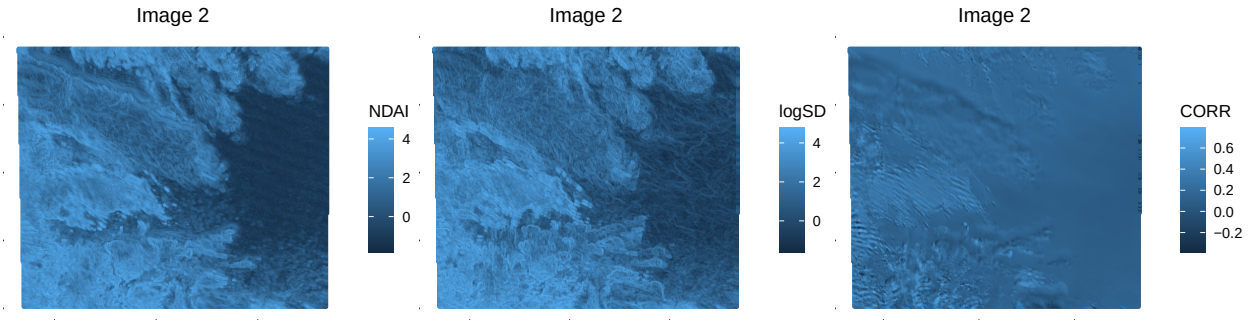

Figure 16: Important Features of Image 3.

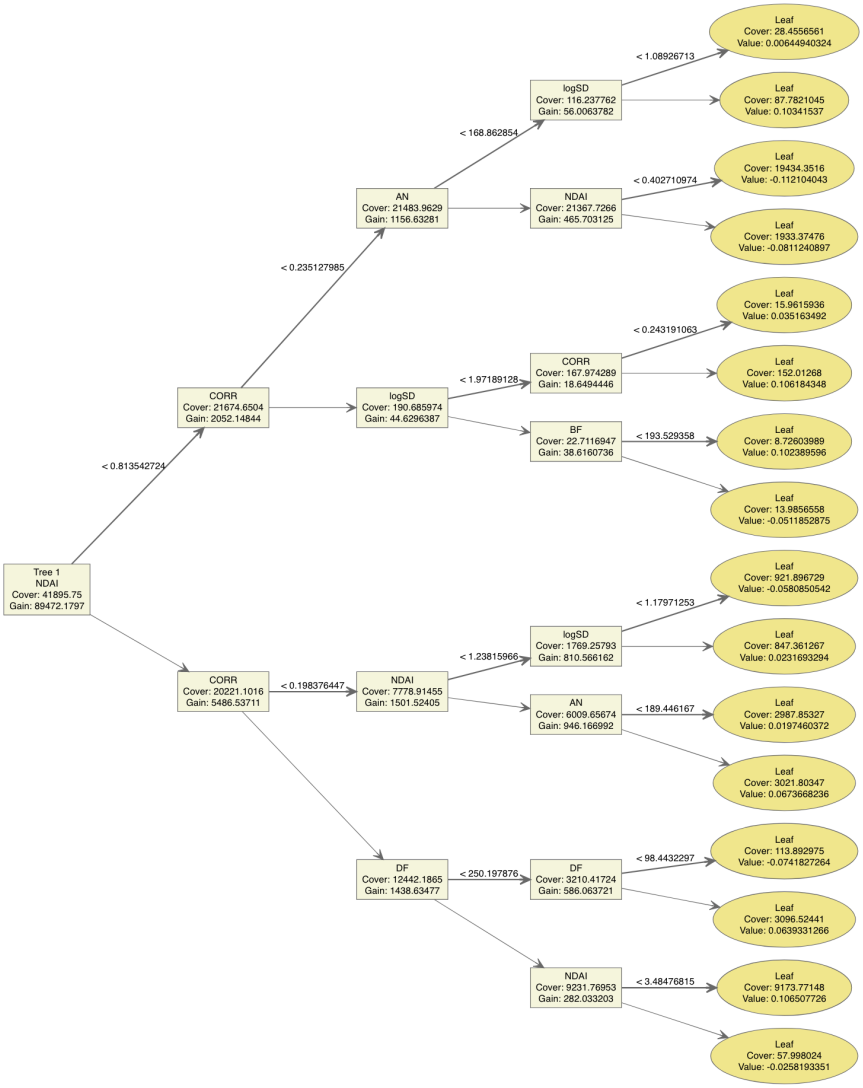

Figure 17: Visualization of Weak Learner

8 Reference

Yu, Bin, Tao Shi, Eugene E Clothiaux, and Amy J Braverman. 2008. "Daytime Arctic Cloud Detection Based on Multi-Angle Satellite Data with Case Studies." *Journal of the American Statistical Association* 103 (482): 584–93.