Project 2: Cloud Data

Matthew Dockman (matthew.dockman@duke.edu)
Richard Fremgen (richard.fremgen@duke.edu)

1. Data Collection and Exploration

In their 2008 paper "Daytime Arctic Cloud Detection Based on Multi-Angle Satellite Data With Case Studies," Yu et. al. tackle the problem of obtaining accurate Arctic-wide measurements of cloud coverage. This issue is of great scientific and public importance as it pertains to the threat of climate change and the sensitivity of the Arctic to increasing surface air temperatures. In order to gain traction in determining cloud coverage, which is often difficult because the amount of visible and infrared electromagnetic radiation emanating from snow- and ice- covered surfaces and clouds is similar, the authors used the Multiangle Imaging SpectroRadiometer (MISR) aboard the Terra satellite. This device yielded electromagnetic radiation measurements from nine different angles in four spectral bands and collected the data by covering an approximately 360 km wide area of Earth's surface that extends from the Arctic to Antartica in only 45 minutes. The device covers 233 distinct but overlapping paths that are each covered every 16 days. Each MISR pixel covers a 275m by 275m region on the ground, producing a huge amount of data. Due to the sheer volume, only the red radiances and all from the nadir camera are transmitted at full 275m resolution; the rest are transmitted at 1.1km resolution.

Due to the size of the data, the authors faced the challenge of combining classification and clustering, since expert labels are impossible to obtain operationally, in a computationally efficient manner. The novelty in their approach was to search for cloud-free pixels, instead of cloud pixels, to detect clouds. To aid in their algorithm, Yu et. al. transformed the feature space by introducing three new features: CORR, an average linear correlation at different view angles, which is expected to be small for clouds and high for no clouds, SD, the standard deviation of the red radiation measurements for the An angle, which identifies smooth surfaces, and NDAI, the normalized difference angular index for which higher values suggest the presence of clouds. Armed with these features, the authors employed an enhanced linear correlation matching algorithm to label the pixels. Pixels were labeled according to fixed thresholds of SD and CORR but a variable threshold of NDAI that was set according to the threshold learned from previous data units and a clustering algorithm. Once labeled, they trained a QDA model to provide an estimate of probability of cloudiness, finding an agreement with expert rate of around 92%. Yu et. al. conclude that NDAI, SD, and CORR contain sufficient information to separate clouds from surfaces and the ELCM algorithm provides for better accuracy and spatial coverage than the existing MISR operational algorithms. The impact of this study is that it shows statisticians have a role to play in analyzing Earth science data, demonstrates the power of statistical thinking and the ability of statistics to contribute solutions to modern scientific problems, and improves our understanding of the flow of visible and infrared radiation through the atmosphere in the Arctic.

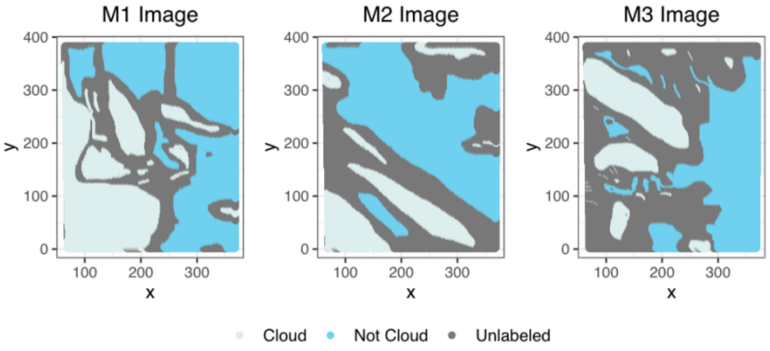

Figure 1: Image Data Colored by Expert Label

Using the data provided by Yu et. al., we will examine its features before fitting our own classification models. The data comprises three images depicting clouds over snow- and ice- covered surfaces. The first image contains an approximately equal number of cloud and no cloud pixels at 34.1% and 37.3% of all pixels, respectively. In the second, the number of no cloud pixels climbs to 43.8% with the cloud pixels dropping to only 17.8%. The third image compares to the second though almost 15% of the no cloud pixels are shifted into the unlabeled category. From this point, we will drop the unlabeled category in our analysis since it will not be used in our classification models.

	Cloud	Not Cloud	Unlabeled
M1 Image	34.1%	37.3%	28.6%
M2 Image	17.8%	43.8%	38.5%
M3 Image	18.4%	29.3%	52.3%
Total	23.4%	36.8%	39.8%

Table 1: Image Data by Expert Label %

Doing a simple summary across the whole data set and the different expert labels, we find that on average, the NDAI and SD is higher for cloud pixels, as expected, but on average, the CORR value is actually higher as well for cloud pixels despite CORR being designed so that it is higher for no cloud pixels. Examining the features themselves, we begin by conducting PCA to see if there is some low-dimensional representation in terms of the features. Interestingly though, we find that the components for the first principal loading all lie between 0.20 and 0.40, so all the variables receive similar weights in the projection to the first principal vector. Conducting K-means and GMM on the eight features sorts the pixels approximately according to the expert label which we would expect. Considering the pairwise relationships among the features by examining the empirical correlations and scatterplots, we see that RaAf, RaBf, RaAn, and RaCf, to a lesser extent, all are very highly positively correlated. In addition, RaCf and RaDf have a strong, positive linear relationship. CORR is somewhat negatively correlated with RaAn, but does not exhibit any other at least moderately strong relationships. Similarly, for NDAI and SD, these features pairwise have a positive, moderate relationship, but do not exhibit a strong relationship with any of the other features.

expert_label	count	ndai	sd	corr	ra_df	ra_cf	ra_bf	ra_af	ra_an
Cloud	80981	1.950	9.845	0.263	272.247	232.467	200.589	174.796	163.028
Not Cloud	127080	-0.263	2.979	0.140	271.295	256.866	240.506	219.465	204.783
Unlabeled	137495	1.821	11.714	0.183	270.906	244.869	223.038	201.159	187.929

Table 2: Variable Averages for Different Expert Labels

Adding in the expert label variable, we examine the densities of the features under the two different expert labels. We notice that for each Ra value, the distribution for no cloud pixels is centered slightly to the right of the cloud pixel distribution but has much less variance. For our three transformed features, NDAI, SD, and CORR, the distributions for each expert label are shifted farther from each other than with the Ra values. This indicates that maybe these variables could be the "best" in predicting the true labels. To make the visualizations more concrete by considering the correlations of the expert label with the features, we see that only NDAI maintains a positive, linear relationship with the expert labels. Going back to the notion of "best" features, intuitively, the best features, given our expert labels as the truth, are those that are the most "helpful" in accurately predicting cloud or no cloud. Put a different way, the best features will be the ones that have the largest difference in values when they are expertly labeled cloud vs no cloud. To make this rigorous, we will differentiate the best features from the rest by first scaling each covariate. This will place each feature on the same scale to permit comparison. We then split the pixels, based on the expert label, into cloud and no cloud groups and compute the means of each predictor in each group. The best features that will provide the most utility in predicting the true labels will have the greatest difference in

mean values between the cloud and no cloud groups. By this criterion, we can say that NDAI is the clear best feature with a difference of 1.555. Following NDAI is the CORR feature with a 1.130 difference. The third "best" feature is the RaAf feature with a difference of -1.041, edging out RaAn with a value of -1.035.

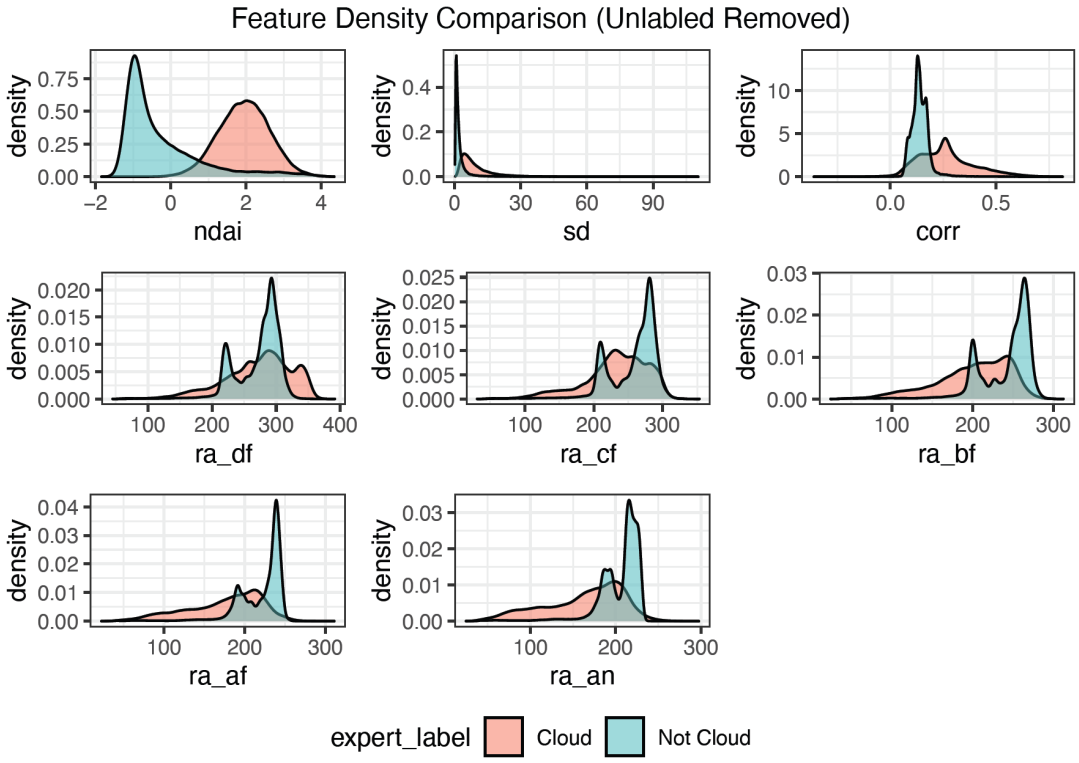

Figure 2: Variable Density Comparison

2. Preparation

These three also remain the best features when we consider the median to purge the effect of possible extreme values that we saw in the density plots on the mean. When we plot the pixels using the x and y coordinates and color the points according to their expert label, we notice immediately that the pixels expertly labeled as cloud tend to lie around pixels that are also labeled as cloud. The same can be said of the pixels that are expertly labeled as no cloud or unlabeled. In other words, given the expert label of a neighboring pixel, the probability of a pixel being that same expert label increases. Since nearby pixels impart information on a pixel, we can say the data is not independent and identically distributed. In particular, the data points are positively spatially autocorrelated. As a consequence of this fact, a simple aggregating of the pixels in the three images and conducting a simple random sample from there to split the data is inadequate. Such a procedure will allow an arbitrary classification model to take advantage of the spatial autocorrelation to better predict the test data because the training and the test data would be spatially autocorrelated. Thus, the resulting accuracy and other metrics would be upward biased. To account for the spatial autocorrelation, we propose two methods for splitting the data into training, validation, and test sets.

First, we propose combining the pixels into blocks based on their location within the images. By then randomly sampling among the blocks, we can ensure that the pixels in the training set are as spatially separate as possible from the test data, which would not allow spatial autocorrelation to upward bias our model performance. In doing this, we noticed that a simple aggregation by dividing the images at points along the x and y axis would not be sufficient; this aggregation resulted in blocks in which there either only pixels labeled as cloud or pixels labeled as no cloud and occurred no matter how fine the partition. If such

a block were sampled into the test set, then we would have a disproportionate number of pixels of one type so that our test set would not have the same distribution as our training set. To account for this, we split each image into 16 blocks and combined 4 sets of 4 blocks, not necessarily neighboring blocks, to obtain 4 total blocks for each image. Blocks were combined so as to preserve the relative proportion of cloud and no cloud pixels in that particular image. We then sampled two of these blocks to obtain the test set and then another one among the remaining ten blocks to serve as the validation set. Henceforth, this sampling method will be referred to as the block method.

Our second method of splitting the data is similar to our first in that we begin by aggregating the pixels into objects and then sample among those objects to obtain the relevant set. However, it differs in that we aggregate the pixels by performing K-means clustering on each image with K= 4. The drawback of this sampling method is that while we generally obtained an equal number of pixels across clusters, we were not able to maintain the relative proportion of cloud and no cloud as closely as with the block method. However, we believe the proportions between the training and test data are sufficiently close and the data size is sufficiently large that we have enough of each expertly labeled pixel to be able to generalize the training model to the test data. Henceforth, this sampling method will be referred to as the cluster method.

We also considered another technique, which is almost an "inversion" of our first aforementioned sampling procedure. Rather than combining the pixels into blocks and sampling from the blocks, we would begin by sampling five points from each of the three images. These points would serve as part of our test set. After conducting these samples, the surrounding 3,000 pixels for each of the randomly sampled pixels would be placed into the test set. This procedure allowed the test data and training data to be as spatially separate as possible to limit the effects of spatial autocorrelation. The same process was conducted on the remaining points to create the validation set with the leftover points from there serving as our training set. Ultimately, we chose not to employ this procedure due to its complexity. It should be noted that we also considered a fourth manner in splitting the data. This method is very similar to the second aforementioned method, but after randomly sampling pixels, it proceeds by removing all nearby pixels within a certain radius, which is determined by the amount of spatial autocorrelation. The result is that every pixel in the training set lies at least the length of the radius away from every pixel in the test set. This would severely limit the ability of spatial autocorrelation to upward bias our performance metrics, but we ultimately chose not to employ it to avoid throwing out useful data points.

Upon splitting the data in the first two manners discussed above, we find that for the block method, if we were to predict every pixel as no cloud, we would obtain a test accuracy of 66.2% and validation accuracy of 57.6%. For the cluster method, if we were to do the same, we would obtain a test accuracy of 59.8% and a validation accuracy of 89.8%. This baseline test ensures that we have a sufficient number of each kind of pixel in each set to make classification worthwhile; such a classifier would have high accuracy if the validation set or test set was populated mainly by no cloud pixels. In this case, we see that we have enough variation for the most part. An 89.8% accuracy is cause for concern though as this was the risk we took with sampling K-means clusters. Fortunately, we are using cross-validation, so we can merge this validation set with the training set when creating folds, and this validation set will serve as only one of ten folds.

Type	Cloud	Not Cloud	n
Training Data	40.1%	59.9%	148183
Validation Data	42.4%	57.6%	15302
Testing Data	33.8%	66.2%	44576
All Data	38.9%	61.1%	208061

Table 3: Block Data Split

Type	Cloud	Not Cloud	n
Training Data	42.1%	57.9%	155856
Validation Data	10.2%	89.8%	18983
Testing Data	40.2%	59.8%	33222
All Data	38.9%	61.1%	208061

Table 3: Cluster Data Split

To assist in model assessment and selection, a generic function called "CVmaster" was built to take in a series of model-specific inputs and output the K-fold cross-validation (CV) loss on the training set. CVmaster makes use of the built-in functionality of the tidymodels framework in R, to enable the user the ability the easily swap between various classification models. To use CVmaster, one must pass in a tidy version of their training data, a generic recipe or series of feature engineering steps to clean the data prior to modeling, along with specifying the type of classifier to be used. These so-called "classifier," and "m_recipe" arguments of the function should be created externally from CVmaster, since the function makes use of R's workflow feature, which makes multi-model analysis and comparison much easier. Additionally, CVmaster gives the user the ability to specify the number of folds for cross validation, and whether the training data should be split via the block or clustering method discussed earlier in this report. The output of this model is a data frame containing the accuracy for each of the k folds used in the cross-validation analysis. With this in mind, we are able to proceed.

3. Modeling

In order to classify the different pixels as cloud or no cloud, we begin first by fitting a simple logistic regression model, which will serve as our baseline classification model. We certainly do not expect that the logistic regression will have great performance with this particular data set primarily because logistic regression assumes a linear decision boundary between cloud and no cloud. Logistic regression models the log odds as a linear function of the covariates and assumes that the data points are iid draws from a Bernoulli distribution. In doing so, the probability of a point belonging to a particular class is modeled as a sigmoid function, where the decision boundary is a hyperplane. Obviously, a linear decision boundary will not suffice for clouds, which usually have ellipsoidal, complex shapes. Moreover, though identical Bernoulli distributions seems reasonable, we do not have independence of the different pixels due to spatial autocorrelation. This is a common assumption made in all the models here and below that is violated even though we adjusted for this as best we could with our different sampling procedures. Fitting the model for the block method yields a cross-validation accuracy of 0.868 and a test accuracy of 0.871 with the values for the specific folds listed below. Doing the same for the cluster method yields a similar result 0.843 and 0.865 for the CV and test, respectively.

Method	CV	Test	Fold Avg	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10
Log Reg	Block	0.871	0.868	0.988	0.875	0.97	0.792	0.755	0.845	0.863	0.738	0.866	0.99
Log Reg	Cluster	0.865	0.843	0.861	0.86	0.96	0.968	0.312	0.985	0.875	0.812	0.97	0.829
QDA	Block	0.859	0.878	0.992	0.911	0.967	0.865	0.623	0.889	0.823	0.793	0.923	0.996
QDA	Cluster	0.834	0.869	0.973	0.924	0.977	0.95	0.514	0.999	0.69	0.816	0.993	0.855
KNN	Block	0.859	0.884	0.984	0.875	0.762	0.971	0.881	0.909	0.904	0.995	0.815	0.743
KNN	Cluster	0.834	0.843	0.704	0.889	0.938	0.673	0.945	0.99	0.943	0.952	0.473	0.918
RF	Block	0.941	0.897	0.998	0.913	0.977	0.861	0.797	0.833	0.932	0.732	0.928	0.994
RF	Cluster	0.916	0.837	0.992	0.924	0.968	0.65	0.947	0.933	0.905	0.881	0.502	0.672

Table 4: Test and Fold Accuracy for each Model Type

Examining the ROC curve in Figure 3, we find an AUC (area under curve) of 0.964 and 0.93 for the block and cluster methods, respectively. The ROC curve graphs the true positive rate, the sensitivity, as a function of the false positive rate, 1 – specificity. A random classifier achieves an AUC of 0.5 and a perfect classifier achieves an AUC of 1, so the AUC values obtained here are definitely worth mentioning. Overall, model performance was decent but by no means optimal. This model is hindered by its strong assumption of a linear decision boundary and is not flexible enough to manage the complex decision boundaries implicit in this data set. We will not consider regularized logistic regression because this classification model is often used in the case when the dimension of the predictors is large, so one desires

to shrink some of the coefficients towards 0. Clearly, that is not the case in this problem where the total number of cloud and no cloud pixels is above 200,000 while we are considering only eight predictors.

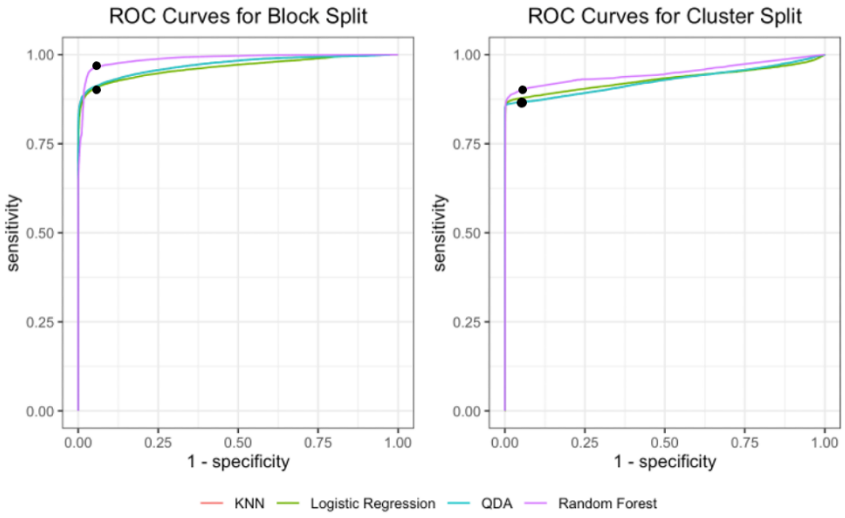

Figure 3: ROC Curves for Block and Cluster Splits

Next, we model the data using quadratic discriminant analysis as was used by Yu et. al. in their classification model. Quadratic discriminant analysis assumes that each class's data points are iid draws from some multivariate normal distribution with each class having its own mean vector and covariance matrix. Given the complex shapes sometimes seen by cloud and no cloud regions, this can be a tenuous assumption. Generating the mean vectors and covariance matrices by maximum likelihood estimation from the training data, assuming the proportion of each class is the marginal probability of a data point belonging to that class, and applying Bayes' theorem, for any test point, we can find the class for which the probability of belonging to that class is highest. Employing this method on our data, we find similar results as logistic regression. This model yielded CV accuracies of 0.878 and 0.869 for the block and cluster methods, respectively, and test accuracies of 0.859 and 0.834 for the block and cluster methods, respectively. The specific fold accuracies can be found in table on the previous page. As for the AUC, we find values of 0.971 and 0.927 for the block and cluster methods, respectively. We mention that for a similar reason as above, we will not attempt regularized QDA. Assessing the results, we notice that despite the fact that we relaxed our assumption of a linear decision boundary to a quadratic one, we still found little improvement. It seems that a single quadratic decision boundary, single because there are only two classes, is still too restrictive. A quadratic boundary cannot capture the complex shapes of cloud and no cloud regions.

To improve our model performance, we reason that we need to try a model-free approach such as k-nearest neighbors. k-nearest neighbors make no modeling assumptions on the underlying data. Instead, kNN finds the k closest points and takes a majority vote to determine the class of the point of interest. With minimal assumptions, running kNN yielded CV accuracies of 0.884 and 0.843 for the block and cluster methods, respectively, and test accuracies of 0.859 and 0.834 for the block and cluster methods, respectively. The AUC results were 0.971 and 0.927 for the block and cluster methods, respectively. Using cross-validation, we ran the model for k=1,2,3 only, determining that higher k was too computationally expensive. We found identical results for k=1,2,3 on cross-validation and decided on using k=1 to provide slightly faster computation time. Contrary to our expectations, kNN provided no improvement over QDA, or logistic regression for that matter. We reasoned that despite the fact that kNN provided for a non-parametric approach, we obtained a similar result because one or more portions of clouds were in our test set that lied adjacent to no cloud regions of our training. Thus, kNN is not great in determining a boundary between cloud and no cloud regions unless one is already given.

The final model we will employ is a random forest because random forests generally have one of the best model performances among classifiers. The random forest is an ensemble method that builds T decision trees of bootstrapped samples in an attempt to reduce variance. To decorrelate the trees for further variance reduction, at each node, only m of the features are used to determine the split and are chosen by random sample. The depth of the trees can also be tuned but generally does not have to very deep given a large number of trees. Random forests are nice in that they make no model assumptions, and thereby, give us the mode-free approach we so desire. Running cross-validation to determine mtry, the number of features to select, and min_n, the minimum number of data points in a node that are required for the node to be split further, we can see by the plots in Figure 4 that the CV error is minimized for mtry=3 and min_n=11. This result held for both methods of creating folds.

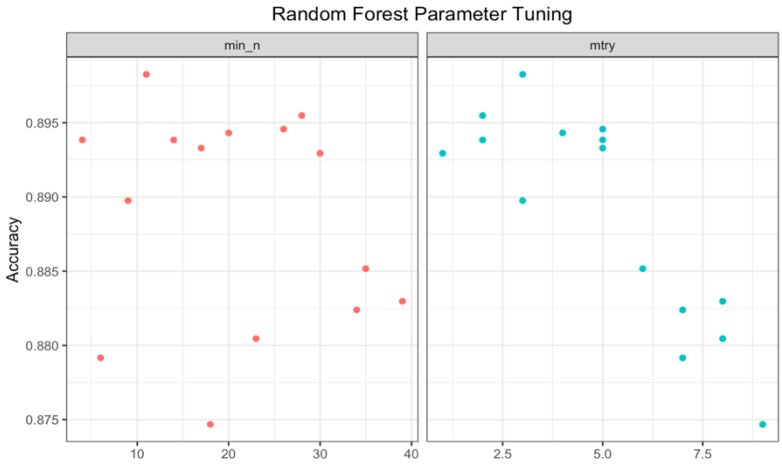

Figure 4: Random Forest Tuning

4. Diagnostics

Examining the convergence of the CV error in terms of the number of trees, we see that for both methods of creating folds, the CV error oscillates before reaching an even level, though it still oscillates a bit more for the block method. Ultimately, we found that 160 trees for the block method would provide the highest CV accuracy for the shortest computation time. Interestingly, only 40 trees were required for the cluster method to obtain the best CV accuracy with the shortest computation time. These hyperparameters are indicative of the bias-variance tradeoff for the random forest. With more nodes, we decrease bias, but increase variance. With a lower mtry value, we introduce bias but decorrelate the trees more and decrease variance as a result.

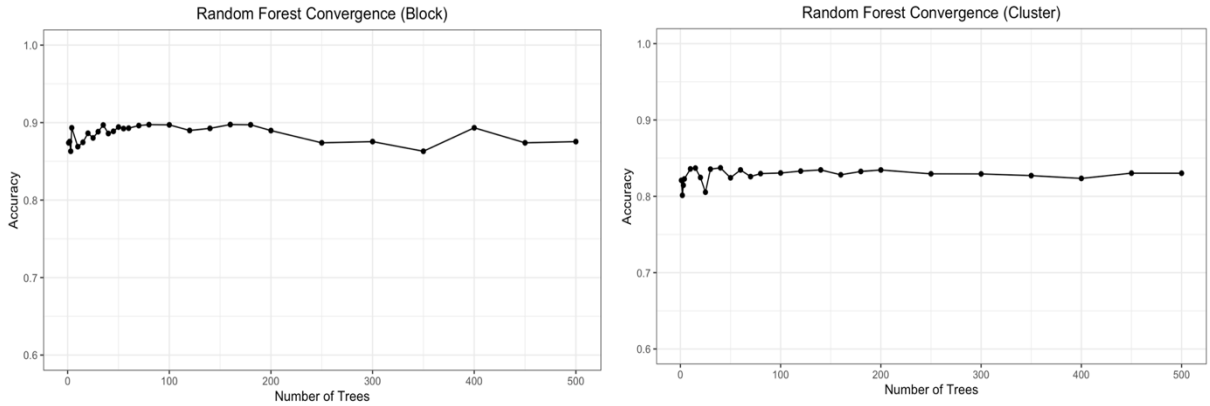

Figure 5: Random Forest Convergence

Similarly, as we increase the number of trees, we also decrease variance. Examining variable importance by using the permutation importance, which measures the decrease in accuracy for a particular feature when its values are permuted, we see that for the block method, the most important variables are NDAI, RaAf, and RaAn. For the cluster method, the most important variables are NDAI, RaCf, and RaDf. Not only are the variables different across folds, but they are also different from the variables that we chose as the "best" variables, which can serve as a reminder to let the data say what the most important variables are instead of choosing them a priori.

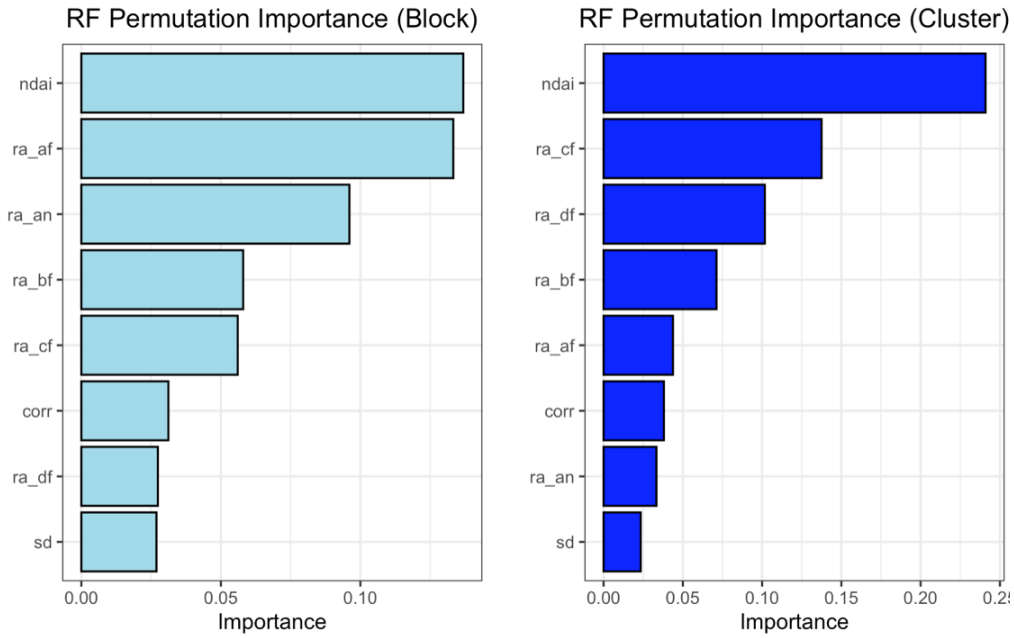

Figure 6: Random Forest Variable Importance

Finally, we will examine one of the trees in our random forest where we have edited the number of nodes so as to allow an easier visual interpretation. We can see in Figure 7 that at each node, a numerical value with a particular feature is given. The decision rule is if a pixel's value for that feature is below the given value, then we move left and if it is greater, we move right. Upon inspection of the plot, we notice that NDAI is the feature at many of the nodes. The feature and its value are chosen at each node so that there is the biggest drop in misclassification error. The fact that NDAI tends to be the selected feature means that splitting at some value of NDAI generally brings the largest drop in error, which helps to explain why it is seen as our most important variable. After tuning and fitting our model, we obtain CV accuracies of 0.897 and 0.837 for the block and cluster methods, respectively, and test accuracies of 0.941 and 0.916 for the block and cluster methods, respectively. Moreover, the AUC values are 0.988, a near perfect value, and 0.948.

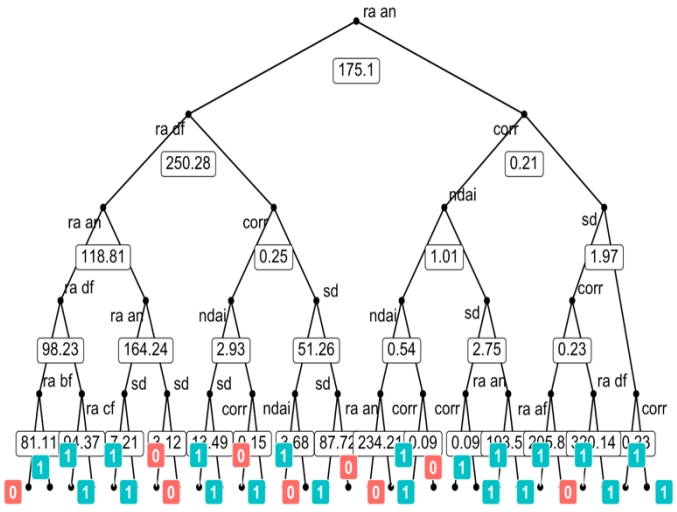

Figure 7: Tree 10 in Random Forest Model

In summary, the random forest performed strongly and the best out of all of our models. This can be further confirmed by the kappa metrics table. Cohen's kappa, which lies between 0 and 1 in this context, is determined for each model by the observed accuracy of our classifier and the expected accuracy by chance. With higher values indicating stronger performance, we observe that by this metric, the random forest is the best model, followed by logistic regression with kNN and QDA performing the worst. Comparing the ROC curves, seen in Figure 3, we see that for either method of creating folds, the ROC curve for random forest essentially encapsulates every other curve. Thus, for almost any given false positive rate, the random forest model has a higher true positive rate, which suggests the random forest is the best model. The same can be said for QDA and kNN with respect to logistic regression in the block method whereas this can be said for logistic regression in the cluster method with respect to kNN and QDA. Using the ROC curves, we can also choose a cutoff value for the specificity and sensitivity, which correspond to a particular threshold. For each curve, we have chosen the point on the curve for which the false negative rate, 1-specificity, is 0.05. We have chosen these cutoffs because we would prefer to falsely predict no cloud as opposed to falsely predicting cloud. This is because clouds modulate the sensitivity of the Arctic to increasing surface air temperatures. As a result, we would rather err on the side of caution and predict fewer clouds so that the threat of climate change in the Arctic seems more dire rather than less.

Method	CV	AUC	Kappa
Log Reg	Block	0.964	0.618
Log Reg	Cluster	0.93	0.717
QDA	Block	0.971	0.555
QDA	Cluster	0.927	0.65
KNN	Block	0.971	0.555
KNN	Cluster	0.927	0.65
RF	Block	0.988	0.836
RF	Cluster	0.948	0.791

Table 5: Model Comparison

Looking at the random forest even closer, we will inspect the misclassified points to see where we can improve our model. Coloring the test pixels according to the mislabeled points, we see that for the block method, the location of the mislabeled points varies. Some lie along the boundary of a region while others lie in the interior. There does not seem to be any pattern in that sense, but virtually all the mislabels come from one particular region. Counting the mislabeled by true label, we find that the mislabeled points disproportionately have true labels of cloud. To get a quantitative comparison, we computed the means of each feature grouped by whether that pixel was mislabeled and its true label. Upon doing so, it is clear why some were mislabeled. The no cloud pixels that were incorrectly labeled as cloud are, on average, very similar to the correctly labeled cloud pixels. On the other hand, the cloud pixels incorrectly labeled as not cloud, on average, lie between the averages of correctly labeled pixels of each type. For the cluster method of creating folds, we found that the mislabeled points again had no pattern in location type, but almost all mislabeled points were in one particular no cloud region.

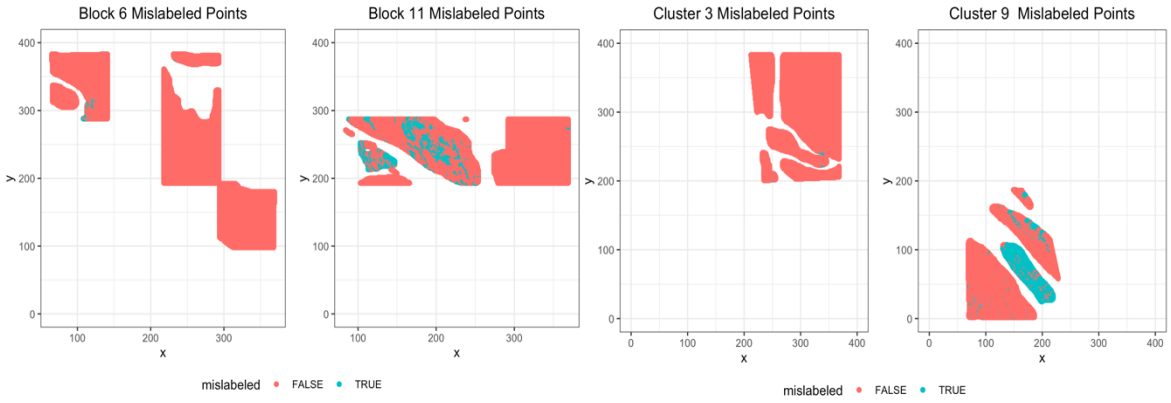

Figure 8: Mislabeled Points

Looking at the averages across the features for the same groups described above, we find a similar result as above, where, on average, the not cloud pixels incorrectly labeled as cloud lie between the averages for the correctly labeled pixels of each type. Moreover, for both folds, the problematic region was between two regions of a different expert label. To come up with a better classifier to remedy this problem, our solution was to introduce a new feature, along with the original eight, which is the square of the NDAI. Our thought process was to try to provide a clearer decision boundary along which to split NDAI at each node. Since the true no cloud pixels tend to have NDAI values close to 0, the squared term would shrink them further towards 0. Meanwhile, the true cloud pixels tend to have NDAI values above 1, so the squared term would move them farther away from 0. We believe this would allow each tree to better recognize the true labels. We also considered a better classifier could be rerunning a random forest, but eliminating the three least important variables. Doing this would allow the "better" variables to better purify the regions of the trees and produce less bias. To account for the increase in variance by the trees becoming more correlated, we could lower the mtry value and/or increase the number of trees.

mislabeled	label	ndai	sd	corr	ra_df	ra_cf	ra_bf	ra_af	ra_an
FALSE	Not Cloud	-0.695	1.49	0.134	275	261	246	226	210
FALSE	Cloud	1.7	8.75	0.188	256	226	207	188	176
TRUE	Not Cloud	2.08	9.47	0.168	254	227	211	194	180
TRUE	Cloud	0.973	5.02	0.172	248	221	205	188	177

Table 6: Mislabeled Points (Box Split)

Our final suggestion for a better classifier would be gradient boosted trees. If we are having a systemic problem in a particular region, gradient boosting could help solve this problem by placing more and more weight on that region until it is classified correctly. Though we would not expect a result in which the entire region is now classified correctly, it could definitely help in correcting some of the pixels and increasing our overall accuracy. On future data, we believe that the random forest will perform generally well but not quite as well as we saw in our test accuracy. This is because there is still some spatial autocorrelation assisting the model, which we would not have if given a new set of pixels. Moreover, our test accuracy benefited from the removal of unlabeled points, which lied on the boundary and our model would have had trouble consistently predicting. We would not have the benefit of unlabeled points in a new image.

In summary, to detect the presence of clouds given satellite image, we fit four models of increasing complexity: logistic regression, QDA, k-nearest neighbors, and random forest. We found that the first three performed very similarly whereas the random forest outperformed them all. We determined that the random forest performed the best by using CV accuracy, test accuracy, ROC AUC, and Cohen's kappa. Further examining the random forest to determine how it could be improved, we found that the mislabeled pixels tended to lie geographically between pixels of different expert labels and tended to have feature values between the normal distributions for cloud and no cloud pixels. To improve performance, we suggested expanding upon the random forest by considering a quadratic transformation of the NDAI feature, conducting variable selection, or employing gradient-boosted trees. While we do not expect the model to perform as well generally as in this test setting, we believe it can be a very effective tool in cloud detection.

5. Reproducibility and Acknowledgments

In terms of workload distribution, our team took a systematic approach of working together concurrently on the analysis of the image data set both from a programming perspective in R and constructing this report. The two areas we spent the most time on, were testing how to best split the data based on our initial EDA findings, and increasing our proficiency in using the tidymodels framework in R. In addition to relying on the coursework and notes from STA-521, we also referenced Dr. Rundel's STA-523 lectures on tidy models, along with the supplemental "Tidy Modeling with R" online book.

For readers of this paper interested in further exploring our results or the satellite image data set in general, we encourage those to examine the analysis and programming methodology we deployed. The next page of this paper is a copy of the README.md file that we developed to further assist one interested in reproducing and further expanding upon our results. This file is also found in the attached zip file that contains further supplemental material and the raw data set we used. Due to the length of the assignment and the amount of code used, we decided to split our Rmarkdown file into two main .Rmd files; one that could be used to recreate our EDA, and a second that could be used to recreate our process of model selection. Both documents contain the same data cleaning and preparation code chunks that we used to transform the data into a format more conducive for analysis. All of the figures and tables included in this report were rendered from one of those two files and can be recreated by running the associated .Rmd file with a connection to a working directory containing the image .txt files; see next page for more details on reproducibility.

README for STA-521 Project 2:

1 - Contents of this folder:

- README.md
- 1_EDA_Code.Rmd - run this Rmd to walk through the EDA performed
- 2_Model_Code.Rmd - run this Rmd to reproduce the models developed
- 521_Proj2_Paper.doc - final report of findings
- CVmaster.R - function that outputs the K-fold cross-validation loss for any tidymodel
- image_data folder - contains the three image data for analysis
- Cloud_Paper.pdf - "Daily Arctic Cloud Detection" paper
- project2.pdf - STA 521 Project 2 Instructions

2 - Getting Started:

The objective of this assignment is to build a classification model to distinguish the presence of clouds from the absence of clouds for three images provided from a MISR sensor on the NASA Terra satellite. The raw data provided for this assignment is located in the image_data folder and contains three .txt files with satellite image data.

To reproduce the analysis, first open the 1_EDA_Code.Rmd file and set your working directory to the location where the image_data folder is. Next, load the necessary packages in the Rmd and run code chunk 0 to load in the image data and perform some preliminary data cleaning on the variables. The next several code chunks in the Rmd under the header ## 1 - Data Collection and Exploration, contain several attempts to visualize and explore the data (EDA). The steps performed in this section ranged anywhere from creating basic summary tables of the variables of interest, to performing PCA on the predictors. Please note that running all of code chunks in ## 1 is optional and solely for EDA purposes.

Following, completion of EDA, run the series of code chunks under the ## 2 - Preparation header to modify the dataset in order to partition the data into twelve total blocks and clusters. These partitions will be used in the modeling phase of the analysis to split the data into a training, testing, and validation data set.

3 - Modeling

Once EDA is complete, open the 2_Model_Code.Rmd file to perform model assessment and selection in order to build a classification model to classify the satellite image data. Code chunks under ## 0 and ## 1 are duplicates from the 1_EDA_Code.Rmd file, as these code chunks are used to load, clean and transform the image data prior to splitting. Running this code is not required, if you have already done so in 1_EDA_Code.Rmd

Use Code Chunks 2A and 2B to split the data into training, testing and validation sets, before writing in the CVmaster function in Code Chunk 2C, which can be used to return the K-fold cross-validation (CV) error for a given model.

The rest of the Rmd is dedicated to fitting different models to the image data set, to determine which model best fit the data. In this Rmd, Logistic Regression, QDA, KNN, and Random Forests were all applied to both the block and cluster data splits. Each model's chunk will return the CV error for each fold from the CVmaster function and then fit the tuned model (if applicable) to the testing data set. The remainder of the Rmd includes code to further tune and explore our selected model (Random Forests), which includes extracting a tree, performing variable importance, and exploring mislabeled data points.

4 - Additional Warnings

For models that take more computational time (KNN and Random Forests), recommend including "doParallel::registerDoParallel()" in your code chunk to speed up the processing time. Using this function, decreased 10-fold CV with a KNN model by 60%. Recommend running a model's entire code chunk at once, since many of the variable names are the same for each fitted model.