Where Are The Clouds: A Life Story (Stat 154 Project 2: Cloud Data)

A Report by Team “15 More Days Until Graduation”

Bogeun Choi (26649533) and Katherine Sheng (26394880)

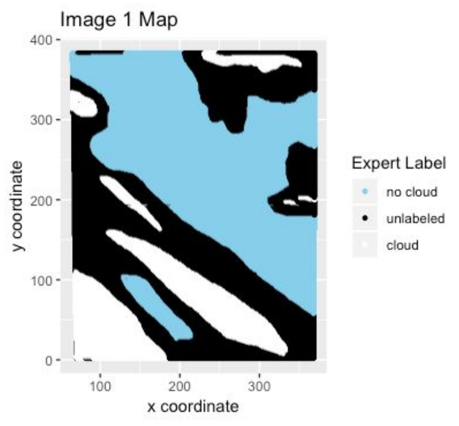
Introduction: Summary/Overview

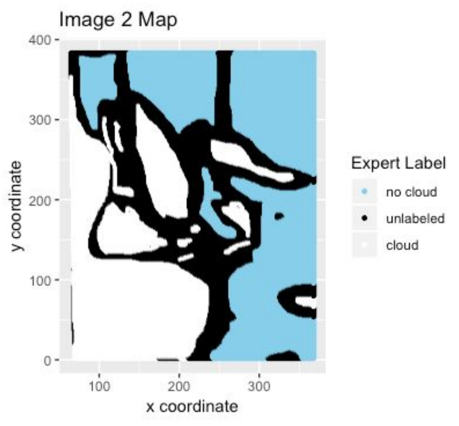
This paper explores how the algorithms ELCM and ELCM-QDA better detect Arctic clouds and identify cloud-free surface pixels. The data is comprised of the sensor’s images from its nine cameras, at different angles in four spectral bands. The sensor takes images in the 180 blocks within each of the 233 paths on a 16-day cycle. Threshold values CORR and SD are fixed for stability against the data collected in various times and locations. The non-nadir camera angles provide radiation-scattering patterns of the clouds. The L2TC algorithm retrieves the clouds’ heights and horizontal velocities. Within the study, the data is collected from 10 MISR orbits of path 26 over the Arctic, northern Greenland, and Baffin Bay spanning from April 28 to September 19, 2002. The data consisted of units with 7,114,248 1.1-km resolution pixels with 36 radiation measurements per pixel. The results actually show that the ELCM-QDA algorithm, even though does not improve the expert labels’ agreement rates in comparison to the ELCM algorithm, does provide probability labels rather than just the binary labels. One particular unit showed relatively low label agreement rates of approximately 70%, compared to 90%. That region is due to the poor terrain data from its sharp elevation changes. The physical features CORR, SD_{AN}, and NDAI show information sufficient to separate ice-covered from snow-covered surfaces. The ELCM algorithm, using these there features, provide better spatial coverage than the existing MISR algorithms. This has a large potential impact, as changes in Arctic clouds can potentially lead to more warming and strong sensitivity to increasing amounts of atmospheric carbon dioxide.

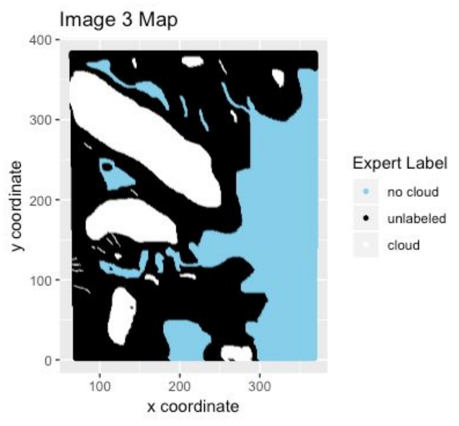
1. Data Collection and Exploration

Before using our data to create a classification model, we must know what the data contain through exploratory data analysis (EDA). We performed four main steps in our EDA: (1) plotted our data, (2) summarized expert label data by calculating % of pixels per given label, (3) explored relationships between expert labels and data features, (4) explored relationships between data features.

(1) PLOTS: We took each image dataset, plotted the points according to their (x, y) coordinates, and colored them based on their expert labels. The plots are shown below:







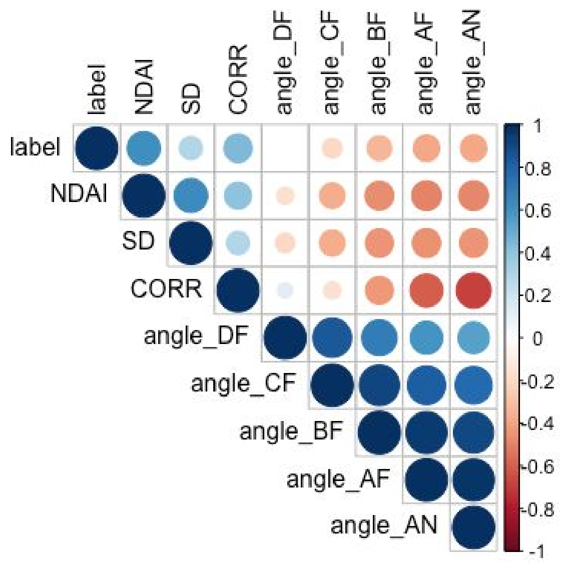
Based on the plots, we noticed a few patterns right away. Firstly, the unlabeled data points always lie between a region of “no cloud” points and a region of “cloud” points. This makes sense as unlabeled data points are ones which are ambiguous, those that could go either into the “no cloud” or “cloud” labeling. We also noticed that there were no single-point or single-line regions: in other words, regions with the same label usually encompassed many points.

Can we assume these samples follow an independent, identically distributed (i.i.d.) assumption? Unfortunately, no. The reason is that our data violates the identically distributed assumption: we cannot assume the distributions of previously collected data points are similar to future data points due to variability in countless factors. Some of these factors include solar illumination directions, terrain, and magnitude of solar radiation.

(2) PERCENT PER LABEL: We calculated the percentages of data points per given label for each image dataset as well as for the overall data comprised of all three image datasets combined.

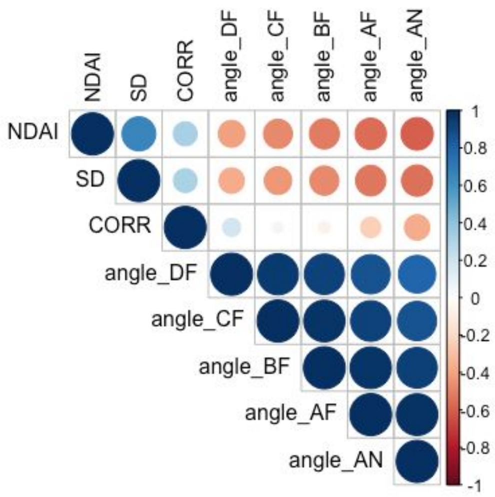
Overall, 23.4% of data points are labeled “cloud”, 36.8% of data points are labeled “no cloud”, and the remaining 39.8% of data points are unlabeled. An interesting observation we noticed when looking at each individual images’ proportions was the wild variance. For example, image 1 had only 17.8% “cloud” data points, while image 2 had 34.1% “cloud” data points, almost twice as much. Ultimately, though, as we planned to combine all three datasets for future analysis, the observation was non-consequential.

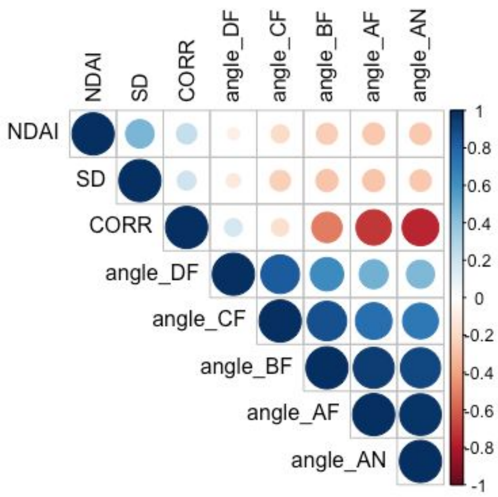
(3) LABELS v. FEATURES: Our main method of determining relationships between expert labels and features was looking at their correlations. The plot to the right is a correlation plot showing the relationships between features and the expert labels. From the first row, we can see there is a somewhat good correlation between expert labels and the features NDAI (0.617) and CORR (0.444). There is also decent correlation between expert labels and the angles BF, AF, and AN (all around 0.35).



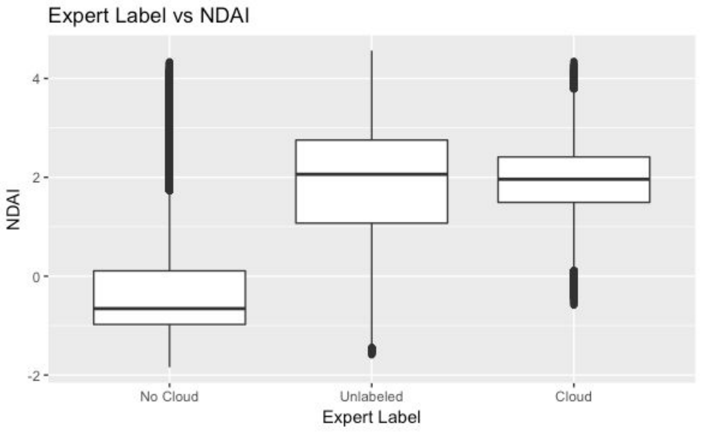
(4) FEATURES v. FEATURES: We can look at the same correlation matrix to explore relationships between features. Based on the plot, we can see a high correlation between the angle features (from 0.547 to 0.971), and a good correlation between CORR and angle AN (-0.682).

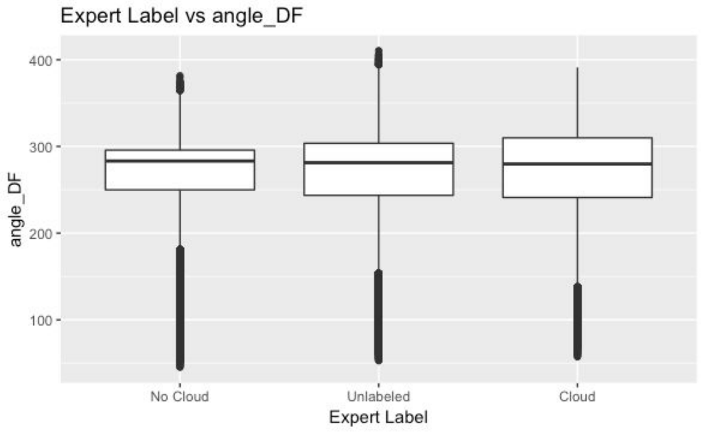
We were curious if there was any difference between the labels based on the features, so we made plots to find them. Below are the two correlation plots, the left with “cloud” data and the right with “no cloud” data.





There are a few noticeable differences such as a high correlation between CORR and angle AN for "no cloud" data that is much lower for "cloud" points. We could conclude that "cloud" and "no cloud" points have differences based on this, but we were not satisfied with just the correlations. Below are some box plots plotted based on label.





These plots further reinforce our belief that there is a noticeable difference between "no cloud" and "cloud" data points based on the features. Almost all of the features had big differences between "cloud" and "no cloud" boxplot. Even angle DF, which had the smallest differences, still has some between the two labels.

2. Preparation

With EDA done, we prepared our data for training. We knew that future data would come in as new images with features and no expert labels. Thus, to classify future data, we had to rely on said features (namely NDAI, CORR, SD, and the angles) and the idea that there were patterns within the images. Because our data is not i.i.d., we cannot randomly assign points to training/validation/test sets. The reason is that if we do this, the patterns in the current images would not be reflected in future predictions. The data is not identically distributed, so there are many different distributions in each plot.

So how do we split the data if these problems exist? To account for these issues, we came up with two different methods of splitting: regions and PCA.

Method 1 (Regions)
The basic idea with this splitting method is to divide the images into regions based on (x, y) coordinates and randomly pick certain proportions to be in the training set, validation set, and testing set. By doing so, we can keep the patterns within the image features while also keep some semblance of a random sampling for training our model to not overfit it. The different distributions can also be captured through this splitting method because they are kept within regions rather than split apart through random sampling.

The number of splits is an important factor to consider: too many means there is a risk that the split is too similar to random sampling, and too little means little patterns in the data are not reflected, making the regions too general/similar. We decided on 16 regions for each image, splitting the x-axis and y-axis into 4 lines. Out of the 48 total regions, we randomly chose 36 to be in the training set and 6 to be in the validation set and testing set each. In other words, ~75% of the data is training while the remaining ~25% is divided equally into the validation and testing sets (~12.5% for each).

Method 2 (PCA)
Similar to Method 1, Method 2 relies on splitting the data via patterns. However, instead of (x, y) coordinates, Method 2 uses a PCA component to divide the data. We ordered the first PCA component from smallest to largest, divided it into parts, and randomly sampled from the parts. Like Method 1, this keeps the patterns within the image features intact. However, with this method there are more features that are taken into account than just (x, y) coordinates when dividing up the data. We hope that including more features via PCA can help classify future data points better, especially since we demonstrated above that a lot of the features in the data are noticeably different between labels.

For this method, we decided to use the same splitting numbers used in Method 1 -- splitting the PCA component into 16, choosing 36 out of the total 48 "regions" to be in the training set, and leaving the other 12 to be divided equally into the validation and training sets.

BASELINE: To ensure that our classification problem is not trivial, we ran a trivial classifier on the validation and test sets. This classifier only sets labels to one category, so if the accuracy was high, it would mean that the data is primarily that label. In other words, the classification problem is trivial -- we can say any data point in the problem is label 1, and we would be right.

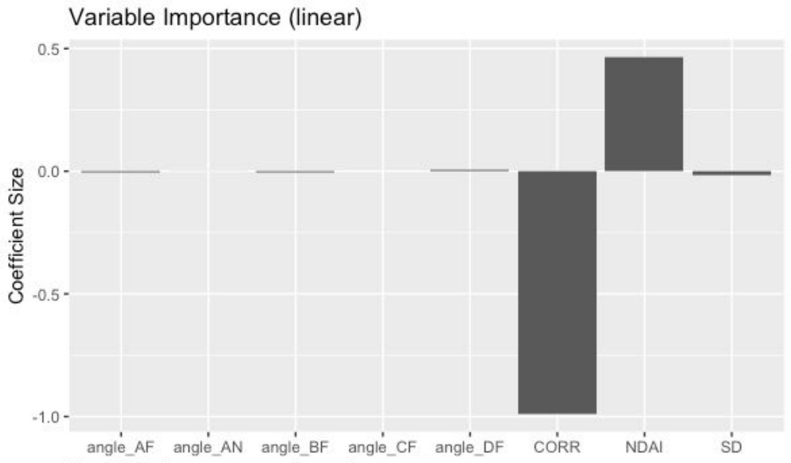
After splitting the data using both of our methods, here are the results from running the trivial classifier:

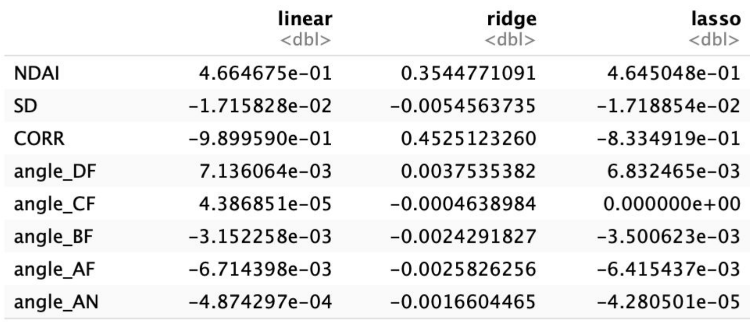
Accuracy of a Trivial Classifier	Split Method #1 (Regions)	Split Method #2 (PCA)
Validation Accuracy	0.7451914	0.7072735
Test Accuracy	0.7206112	0.5976263

Since the accuracies are not unreasonably high, we can be assured that the classification problem is not trivial.

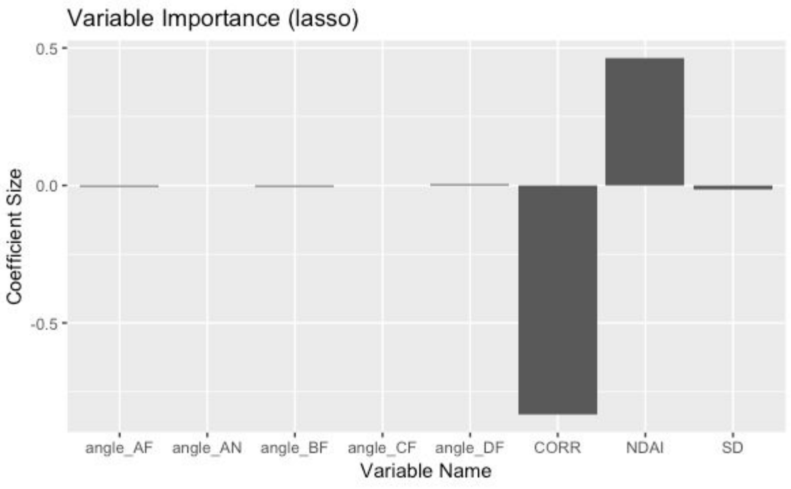
FEATURES: What are the three "best" features in our data? First, we needed to determine what "best" means. For us, since we were limited to not use fancy classification methods, we decided that the "best" features had high variable importance. In other words, their coefficients in a simple model were significant (not necessarily bigger because negative coefficients exist).

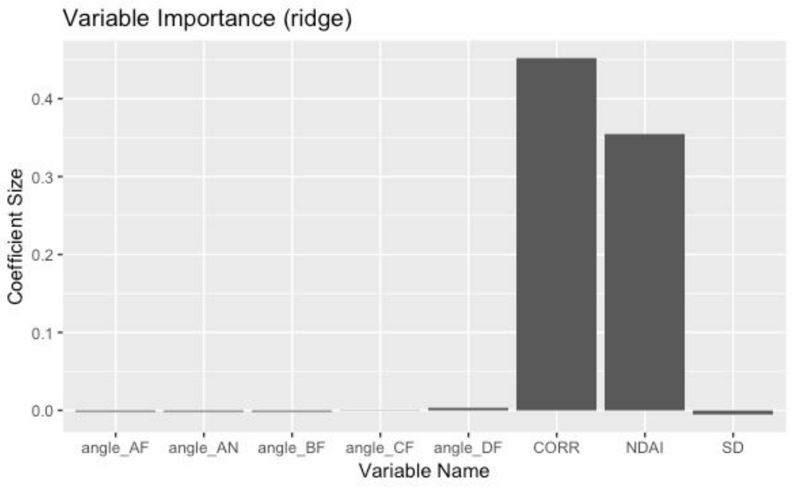
To do this, we ran three different models (linear, ridge, lasso) and graphed the coefficients. We ran this with both the entire dataset and the training set and found roughly the same results. Below are graphs are from running models on the entire dataset as well as the coefficients themselves.





	linear <dbl>	ridge <dbl>	lasso <dbl>
NDAI	4.664675e-01	0.3544771091	4.645048e-01
SD	-1.715828e-02	-0.0054563735	-1.718854e-02
CORR	-9.899590e-01	0.4525123260	-8.334919e-01
angle_DF	7.136064e-03	0.0037535382	6.832465e-03
angle_CF	4.386851e-05	-0.0004638984	0.000000e+00
angle_BF	-3.152258e-03	-0.0024291827	-3.500623e-03
angle_AF	-6.714398e-03	-0.0025826256	-6.415437e-03
angle_AN	-4.874297e-04	-0.0016604465	-4.280501e-05





Throughout all three models, the graphs show CORR and NDAI having high significance. As for the third feature, while much smaller compared to CORR and NDAI, SD has higher significance compared to the other angle features for all three models as seen in the coefficient dataframe in the top right. From this, we conclude that the three "best" features are NDAI, CORR, and SD.

3. Modeling

With the data split and ready to go, we move on to testing out different classification methods. To assess their fit, we used 5-fold cross validation, then found the test accuracy. We used four different methods: logistic regression, LDA, QDA, and SVM. The tables below display the model accuracies for each individual fold, total fold average, and test set.

Model Accuracies (Split Method 1)	Logistic Regression	LDA	QDA	SVM
Fold 1	0.9447151	0.947418	0.9646415	1
Fold 2	0.9007526	0.9014545	0.9019385	1
Fold 3	0.6847158	0.7127369	0.7576577	1
Fold 4	0.8154468	0.8585822	0.8739335	1
Fold 5	0.9396569	0.9455758	0.9647372	1
CV Average	0.8570574	0.8731535	0.8925817	1
Test	0.8536372	0.8213238	0.8951339	0.8755478

Model Accuracies (Split Method 2)	Logistic Regression	LDA	QDA	SVM
Fold 1	0.8196709	0.8296678	0.8506357	1
Fold 2	0.9267527	0.9252502	0.9198841	1
Fold 3	0.8658916	0.8724828	0.8788421	1
Fold 4	0.8805346	0.8890658	0.9403668	1
Fold 5	0.9192939	0.917783	0.9393634	1
CV Average	0.8824287	0.8868499	0.9058184	1
Test	0.8776817	0.8880153	0.8662249	0.7888352

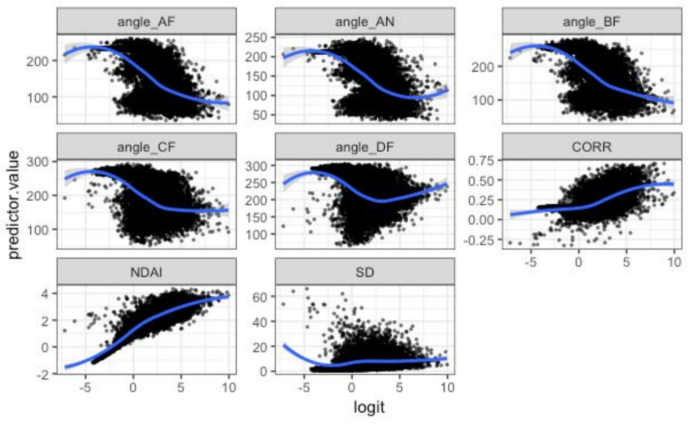
Logistic Regression: For logistic regression, there are a few assumptions to make:

- There is a large sample size -- with around 200,000 data points, we can safely assume this is true.
- The predicted variable should be binary. This is also true: “label” is a binary variable taking the value of either 0 or 1.
- There should be little to no multicollinearity among predictor variables. In other words, the predictor variables should not be correlated with each other. This might be a concern because our correlation plots above showed good correlation between variables. To further test this, we

computed the variance inflation factors (VIFs). A VIF value above 5 or 10 usually means there is a high amount of collinearity among variables. As we can see, there are high values in all of the angle variables as well as in the CORR variable, which shows high collinearity among the predictor variables.

```
        y         x      NDAI        SD      CORR  angle_DF  angle_CF 
 2.245149  1.443823  1.952450  1.751542  6.046993  7.776155 20.814422 
 angle_BF  angle_AF  angle_AN 
44.897422 77.662160 48.840871 
```

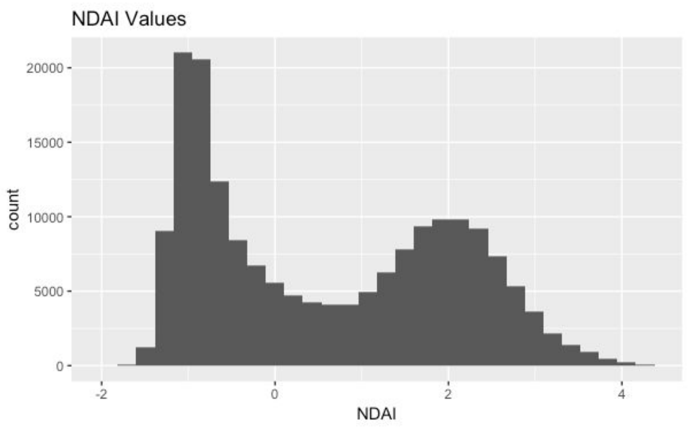
- Observations to be independent of each other. This is where things may get a bit tricky, because our data is not i.i.d.. However, this is fine because the data are not dependent on each other. Though there are patterns, a point labeled “cloud” does not depend on the points around it, but rather on its features, which is based on its distribution. Therefore this condition is satisfied.
- Finally, we want linearity between predictor variables and log odds (logit). To make sure this is true, we plotted the logit values of each variable. As we can see from the plots in the right, there are some variables that have non-linear relationships, namely the angle variables. However, that may be from the amount of data that we have -- around 200,000 points. We can make the case that the data can fit a linear relationship between the predictor variables and the log odds -- since the relationship isn’t too crazy, we should be fine with the data we have.



Out of the four models we tested, this one has the most expected behavior: its test accuracy is slightly worse than its cross validation accuracy for both splitting methods. Although the range of accuracies for each individual fold is concerningly large for split method 1, that seems to be the fault of the data split rather than the method itself since all of the other models share the same variance range. Also, the range of values for split method 2 is not as large, further proving this point. Overall, not a bad model. But we can do better.

LDA: The following are assumptions for the LDA model:

- The sample size is bigger than the number of predictor variables. With only 10 predictor variables and hundreds of thousands of sample data points, this is very clearly satisfied.
- The data is Gaussian -- in other words, predictor variables are shaped like a bell curve when plotted. To prove this, we did the obvious step in plotting the predictor variables which are shown below. Right away, there is a problem with this assumption. Take, for instance, the histogram for NDAI. Obviously not a bell curve. In fact, all of the variables’



histograms did not have a bell curve. This means that the data we have is not Gaussian.

- The predictor variables have roughly the same variance. We can test this by checking the covariance matrices between variances. However, we can see that the variances between variables is not the same, or even close to being the same.

```
                NDAI          SD        CORR    angle_DF    angle_CF
NDAI      2.10387833   7.4297156  0.09054692  -9.8957596  -27.278699
SD        7.42971558  61.4855599  0.36489964 -66.5184492 -136.827275
CORR      0.09054692   0.3648996  0.01283979   0.5513672   -1.310974
angle_DF  -9.89575959 -66.5184492  0.55136716 1779.6623583 1504.932090
angle_CF -27.27869920 -136.8272750 -1.31097372 1504.9320901 1792.729067
angle_BF -37.16440562 -172.4697095 -2.82287621 1242.3939520 1732.715475
angle_AF -39.66649747 -179.4577360 -3.61377619  990.6271698 1554.881986
angle_AN -37.16667821 -165.7631107 -3.68293959  844.0701880 1379.784723
           angle_BF    angle_AF   angle_AN
NDAI     -37.164406  -39.666497  -37.16668
SD      -172.469710 -179.457736 -165.76311
CORR      -2.822876   -3.613776   -3.68294
angle_DF 1242.393952  990.627170  844.07019
angle_CF 1732.715475 1554.881986 1379.78472
angle_BF 1988.457991 1909.137191 1728.30301
angle_AF 1909.137191 1977.911077 1827.51648
angle_AN 1728.303014 1827.516481 1750.32487
```

The method we used in R did not allow for hyperparameters, so we did not use them. Upon first look, the LDA model looks to be better than the logistic regression model based on the cross validation accuracies. However, when we run the model with the test set, we get a worse accuracy than the logistic regression test accuracy. Interestingly, this is only the case with split method 1: for split method 2, the test accuracy is actually better than the cross validation accuracy and the logistic regression test accuracy. Overall, depending on the splitting method, it is a toss-up as to whether the logistic regression model or the LDA model is better.

QDA: The assumptions for the QDA model are very similar to the LDA model. Only the first two points above are required -- that the sample size is bigger than the number of predictor variables, and that the data is Gaussian. We proved that neither condition is met, but we continued on. Similar to LDA, our method in R did not allow for hyperparameters so they were not used in the QDA model.

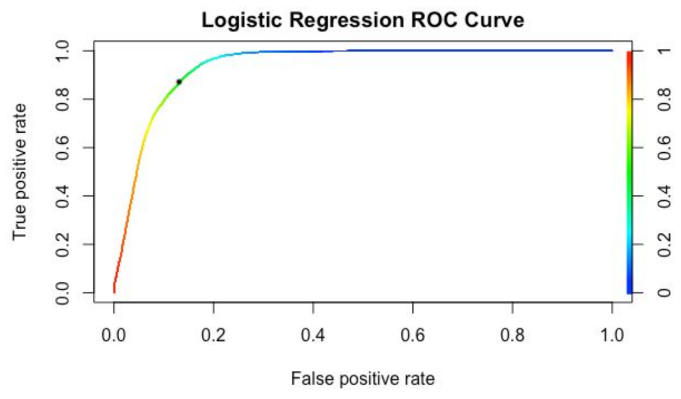
Similar to the LDA model, the QDA model either performs better or worse depending on which split method is used. When using split method 1, we found the test accuracy is better than the cross validation accuracy and the logistic regression test accuracy. However, if we use split method 2, the test accuracy is worse than both the cross validation accuracy and the logistic regression test accuracy. Again, a toss-up on whether QDA is better depending on the split method used.

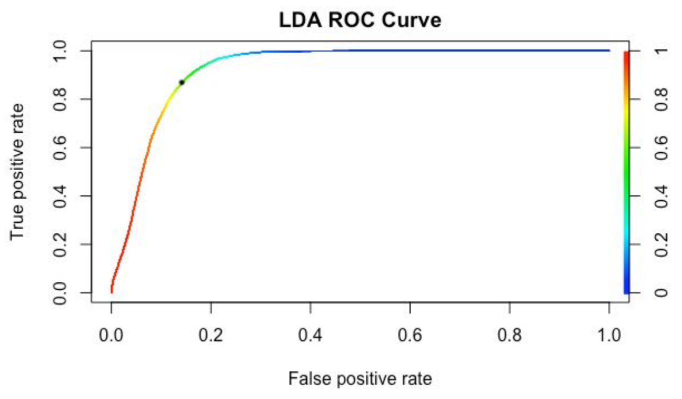
SVM: For the support vector machine model, only one assumption is needed: the data must be i.i.d.. Unfortunately, we know from previous explorations that is not the case at least for the data as a whole set. However, our methods of splitting should hopefully reflect the different types of distributions within the data and negate the fact that the data is not i.i.d. for SVM.

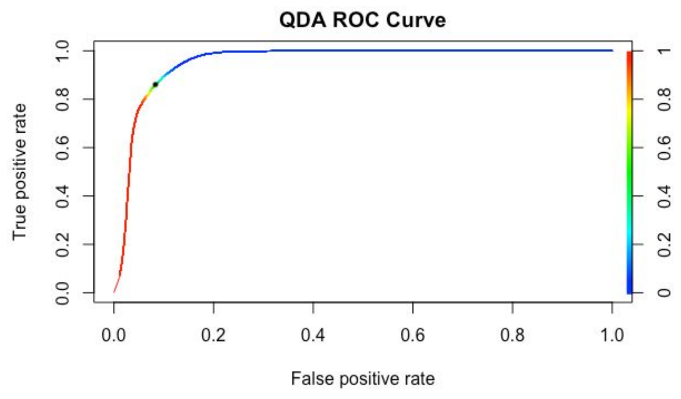
We tested a range of values for the hyperparameters. For our model, the hyperparameters being tuned was cost. We ran 5-fold CV and tested different values of cost ranging from 1 to 1e-9. The best cost parameter was used for CV and test modeling.

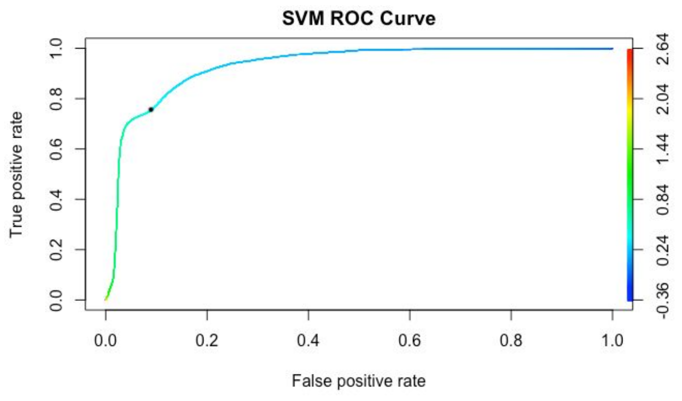
As we can see from the table above, SVM results some very high numbers, namely having 100% CV fold accuracy. We checked carefully to make sure nothing was wrong with our code, but nothing seemed off, so we accepted this weird outcome. Sure enough, when we ran the model on the test set, we got much lower accuracies. In the case of split method 2, a very low accuracy that is by far the worst out of the four models used. These results show that the model was most likely overfitting the training set.

ROC CURVES: Below are the ROC curves for each method. The area under curve (AUC) for each are the following: Logistic Regression (0.9366808), LDA (0.9236476), QDA (0.9549684), and SVM (0.9300037).







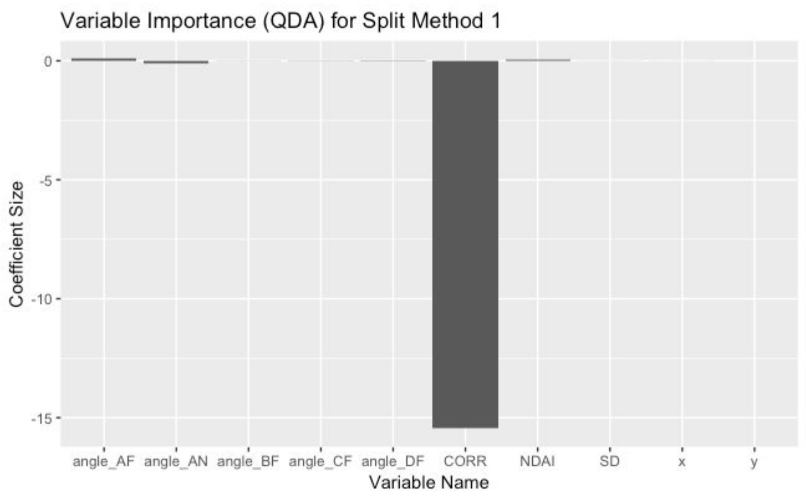


Our choice of cutoff value was estimated by finding the point with the minimum Euclidean distance to the point (0, 1). From there, we then estimated where it lied in the range of values, and picked a value from that. The plots above show the highlighted cutoff points that were chosen (Log Reg: ~0.5, LDA: ~0.75, QDA: ~0.6, SVM: ~0.5)

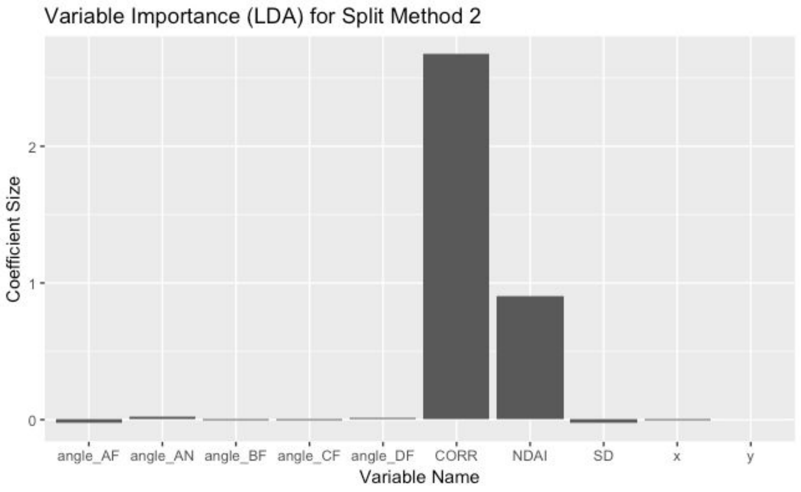
4. Diagnostics

Depending on the split method used, we get a different "best" model. For details' sake, we will explore the QDA model for split method 1 and the LDA model for split method 2.

First, we explored variable importance by plotting the coefficients of the QDA model for split method 1. As we can see, CORR has a very big coefficient in the model with -15 when other coefficients do not even reach an absolute value of 1. The other variables we determined as "best" in the above sections, NDAI and SD,



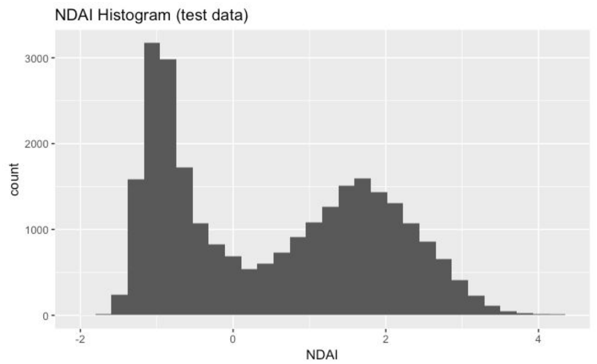
contribute little to the model with coefficients to the scale of 0.01 and 0.001 respectively. Though the small SD makes sense given it was not very significant during our testing, NDAI having a small coefficient is somewhat surprising. We were curious if this was the same for the other split method, so we plotted the coefficients for the LDA model with split method 2. Plotting the coefficients results in the following graph.

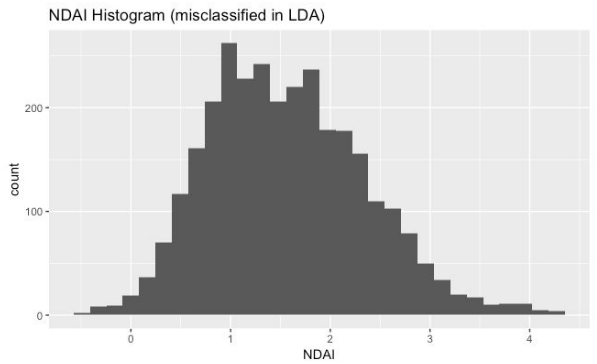


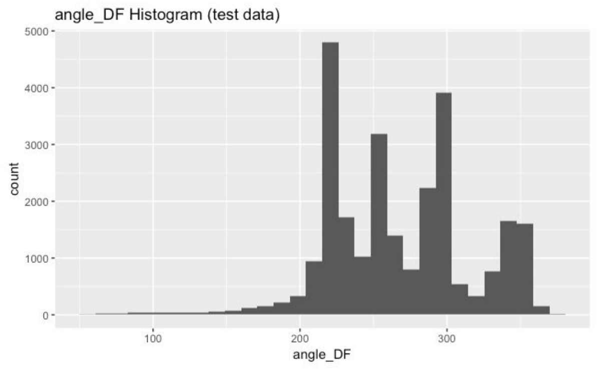
Similar to the QDA coefficient graph, CORR has the most significant coefficient, but this time NDAI follows suit with a pretty big contribution to the model. In fact, this plot looks very similar to the plots done when testing variable importance for linear/ridge/lasso models. This follows what we believe in terms of "best" features for our classification model.

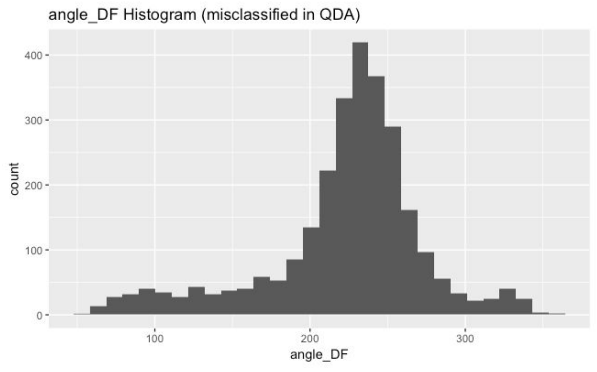
That made us wonder -- why does the QDA model for split method 1 have such a low contribution from the NDAI variable? One hypothesis we had was that because the data is split only according to (x, y) coordinates for split method 1, it renders the possible contributions of NDAI useless. This is further proven with the variable importance plot for split method 2.

The next step we did was look for patterns in misclassified points. For both split methods, we noticed that many of the misclassified points had a positive NDAI value. Also, for the angle variables, the misclassified points were centered around a certain angle when the plots were normally multi-modal.









So with all that in mind, how can we create a better classifier? The biggest idea we had was to change how we split the data in minor ways. For example, for split method 1, instead of splitting the data into 16 regions, try a different number. Or for split method 2, incorporating the second PC component and splitting the data into “regions” similar to how we use (x, y) coordinates.

We tried some of these proposed fixes for both splitting methods. These are the results we get from the improvements:

Model Accuracy Comparison (split method 1)	QDA (16 regions/image)	QDA (64 regions/image)
Fold 1	0.9646415	0.9401207
Fold 2	0.9019385	0.9239431
Fold 3	0.7576577	0.8925968
Fold 4	0.8739335	0.9672665
Fold 5	0.9647372	0.7916611
CV Average	0.8925817	0.9031176
Test	0.8951339	0.8951339

Model Accuracy Comparison (split method 2)	LDA (one PC component)	LDA (two PC components)
Fold 1	0.8296678	0.8888977
Fold 2	0.9252502	0.9403292
Fold 3	0.8724828	0.8865041
Fold 4	0.8890658	0.9245424
Fold 5	0.917783	0.8277285
CV Average	0.8868499	0.8936004
Test	0.8880153	0.8970759

Overall, we get better results for both CV and test set accuracy. The similarity between test accuracies for method 1 may be because the data was split the exact same.

In conclusion, there are many different classification models that can be used for this problem. We haven’t even touched some of the more advanced models out there such as KNN or random forest.

However, we believe that for our method of splitting into regions, LDA/QDA (depending on which type of splitting used) provides the best results while being a simple and easy-to-understand model. Future experiments may fine tune variables that we only touched on, but we believe this is a great starting point to create the best cloud classifier in the world.

Github repository: https://github.com/TheJPFDude/stat154_proj2

Acknowledgement Statement: Katherine did the summary/overview while Bogeun did everything else.

The way we worked on the project was step by step: going through the steps and fixing some methods along the way.