Stat 154 Project 2: Cloud Detection
Irene Ju (3031811225)
Josh Birck (3031853124)

PART 1: DATA COLLECTION / EDA

Summary of Paper

The paper starts off by giving the motivation behind why we should care about cloud detection, which is that cloud detection is important to study climate change over time. It then mentions that existing methods to detect clouds ran into problems when trying to do so over the arctic regions, because of the similar scattering properties of arctic terrain and clouds. The purpose of the study was to use the data from Multiangle Imaging Spectro Radiometer (MISR) to build a new cloud detection algorithm. The data was collected for this experiment from 10 MISR orbits on path 26 (since path 26 had rich surface features) leading to 7,114,248 1.1km resolution pixels with 36 radiation measurements per pixel. Radiation from each pixel was then measured from 9 different cameras, resulting in the AN, 4 forward, and 4 aft angles. In addition, the paper also introduced 3 new features, CORR, SD, and NDAI, that were generated from these raw features. These features were key for the methodology of the new algorithm: Enhanced Linear Correlation Matching, which set thresholds for CORR, SD, and NDAI, and classified a pixel as cloudy or not cloudy based on whether the threshold conditions were met. After the ELCM algorithm generated labels, the paper then talked about using the labels generated to train QDA to obtain probabilities that a pixel is classified as cloudy or not cloudy. After running ELCM-QDA, along with other classification methods such as SVM, it was found that ELCM had the highest agreement rate with the expert labels (91.8%). One of the main flaws of the ELCM algorithm involved the CORR features, which predicted a pixel to be clear when its value was high. The problem was that for many cases, its value was low even when there was no cloud. The conclusion was the ELCM performed poorly over regions with rough topography. There were two main impacts of this study: having statisticians directly involved with the data processing instead of doing analysis after the fact was novel and not common in the past, and showing how statistics can contribute to solving many scientific problems

Plots of Images

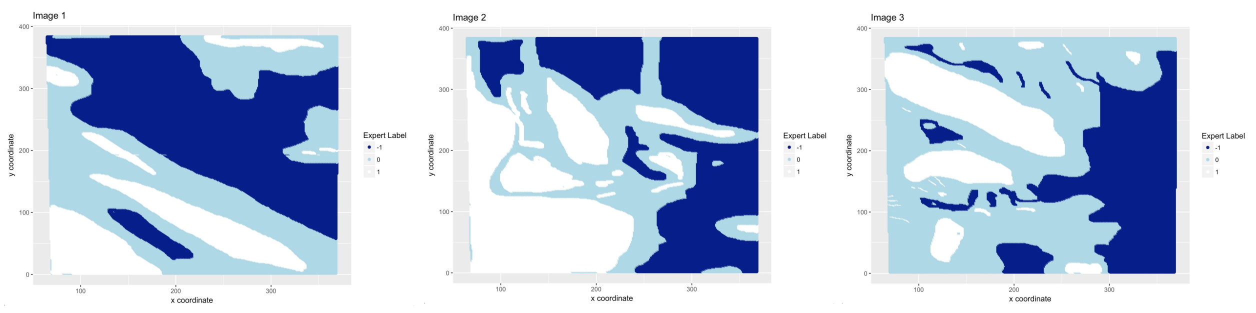


Figure 1

One trend that appears across all three images is that the pixels are likely to be grouped together by classes, which makes sense because clouds do not appear as individual scattered pixels, but as groups of pixels. Also, unlabeled pixels (denoted by the label 0) are always in between groups of pixels that are labeled as cloudy and not cloudy. This might be because on the edge of clouds, or areas where it is partly cloudy, it may be hard to confidently say whether there are clouds present. In addition, these MISR images were taken in different locations and at different times, so the distribution of class labels per image will be different. For these reasons, the data is not iid.

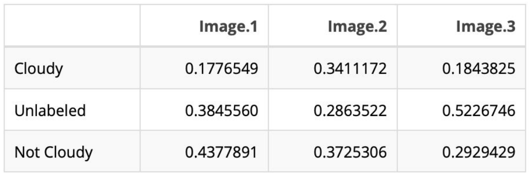


	Image.1	Image.2	Image.3
Cloudy	0.1776549	0.3411172	0.1843825
Unlabeled	0.3845560	0.2863522	0.5226746
Not Cloudy	0.4377891	0.3725306	0.2929429

Figure 2

After computing the proportion of pixels that belong to each class for each of the images, we can again see that the data is not iid, since the proportions vary depending on the image.

EDA

To first get an idea of what our features were, we plotted pairwise scatter plots for all 8 features, for each of the 3 images. These plots took a long time to run, and were not very informative because of the small plot size, and the sheer number of points on each plot. However, one thing that you can tell is that the camera angle radiance features (DF, CF, BF, AF, AN) are all positively correlated with each other, which makes sense because they are all measuring the same thing, but at slightly different angles. It is hard to determine clear relationships with the other features, so we decided to omit the plot.

Next, we looked at the distribution of the feature values, but grouped by class label. To do this, we plotted side-by-side boxplots for each of the 3 images.

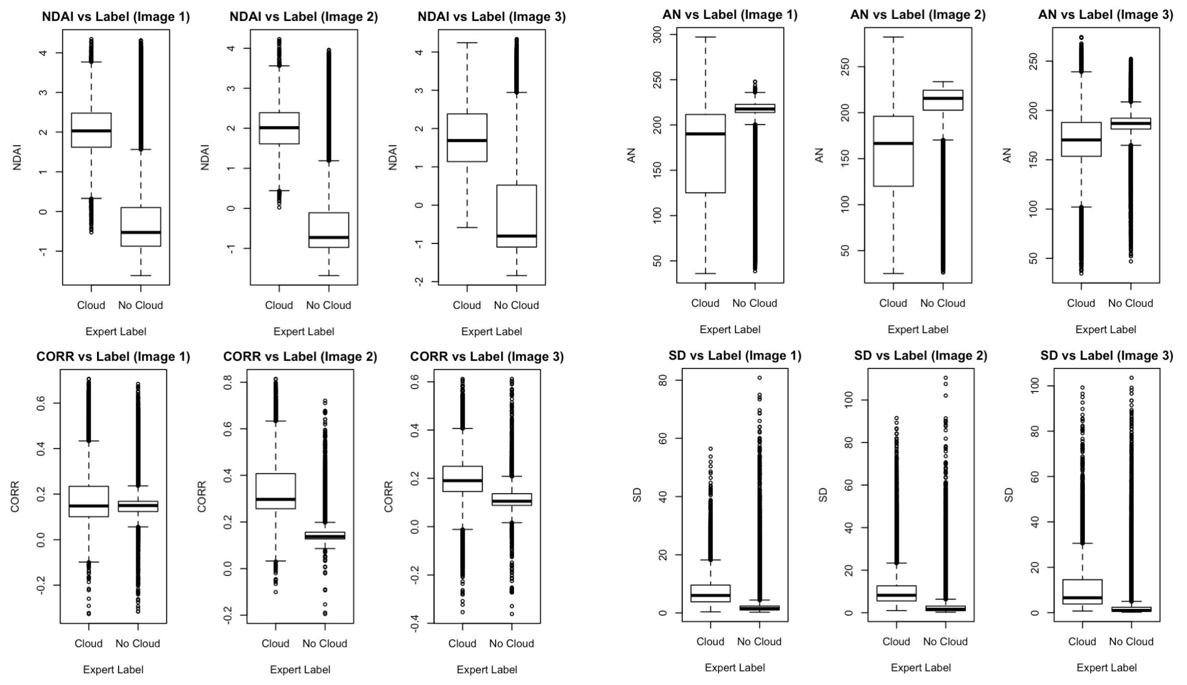


Figure 3

We decided to only include the boxplots for the features NDAI, SD, CORR, and AN to be concise, and because we found out from the pairwise scatter plots that the radiance angles were all positively correlated. Although there are a lot of outliers, and more data cleaning is necessary, by just looking at the median and 1st/3rd quartiles, **we could determine that pixels classified as not cloudy had lower values of NDAI, SD, and CORR, and higher values for AN.**

PART 2: PREPARATION

Splitting the Data

We split the data two different ways in order to compare how it would affect the accuracy of the prediction models. The first method is using a random split and the second is splitting the image by geographical blocking.

Method 1 Split: Random

We performed this method separately on the three images provided. For each image, we first deleted the unlabeled points from the data frame, then proceeded to set 80% of the remaining data as our training set and 20% as our test set. The reason for deleting the unlabeled points is because we want to train our data on points with definitive expert labels in order to guarantee a higher accuracy. Furthermore, some of the classifiers, such as logistic regression require a binary classification, so we set the label -1 (no cloud) to 0. Next, for the 80% training data set, we split it up into four folds. One of the four folds is used as a validation set, while the remaining three are used for the training set. The model's final accuracy rate is the average of the four validation accuracies from each trial. The biggest advantage of this method is that every data point is used for validation exactly once and for training 3 times and it computes a more accurate validation accuracy by averaging multiple accuracies together for each model. This method of splitting the data might be a more naive way, however, each fold contains a even distribution of clouds and no clouds shown in Figure 4. The even distribution guarantees less misclassification errors because a when training the model, the classifier will have more data points to determine a threshold for each expert label.

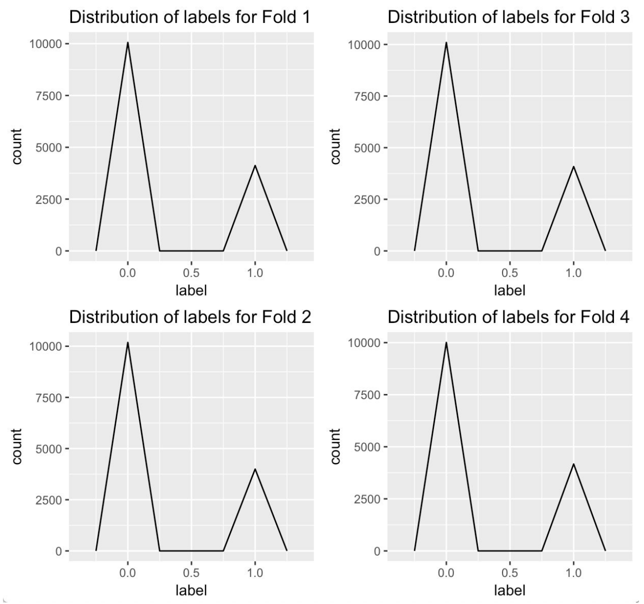


Figure 4

Below is the trivial classifier where it sets all the labels to no cloud for the first split method. In the table, we can see that the test set accuracies and validation set accuracies are very similar. Since the data is split randomly, the trivial classifier with around 70% accuracy is expected based on the frequency plots shown for the distribution of expert labels in Figure 4. We will expect the second method of splitting to have a higher average accuracy for some of the blocks and it will be explained more in detail below.

Images	test_set	validation_set
1	0.7115059	0.7112967
2	0.5239805	0.5215162
3	0.7826364	0.6137831

Figure 5

Method 2 Split: Geographical Blocking

Again, we performed this method separately on the three images provided. For each image, we used the cut() function in R to separate the image into nine blocks. After this, we deleted the unlabeled points in the data and set the label for "no cloud" as 0 instead of -1 to help with modelling later. After obtaining the 9 blocks, we left two blocks as test sets, and seven blocks as train sets. To select the testing blocks, we chose the two blocks that had the least and most number of observations so the testing data had variety. In the training set, one of the seven blocks is used as a validation set, while the remaining six are using for the training set. The model's final accuracy rate is the average of the 7 validation accuracies from each trial. The main reason we decided on using this method to split the data because it takes into consideration the non iid nature of the data. When new data comes in, it is in the form of images where the pixels are grouped together depending on whether they are cloudy or not cloudy. The random split does not take this into account because it aims to get an equal distribution of class labels per fold, but this would rarely the case for new data. By splitting the image into 9 blocks, pixels are grouped together based on geographical location, and having 9 blocks gives you a larger variety class distributions to train models on.

Below is the trivial classifier where it sets all the labels to no cloud for the second split method. Unlike the performance of the trivial classifier in the first split method, the test set and validation set had different accuracies because the distribution of class labels can vary greatly between folds. For a block that had more cloudy pixels than non cloudy pixels, it makes sense that the trivial classifier would perform poorly. However, there are some cases where it can perform really well. By splitting the data into finer blocks, there are likely to be folds that have a high frequency of only one class but not the other. In these cases, the classification accuracy could be nearly perfect with the trivial classifier.

Images	test_set	validation_set
1	0.6764248	0.6612651
2	0.6096818	0.4700778
3	0.6339425	0.5060876

Figure 6

Feature Selection

Throughout our analysis on the relationship between feature values and pixel class labels, we removed observations outside the 10th and 90th quantile, since after looking at the summary statistics, a lot of outliers were present. By doing this, our analysis was able to reflect the majority of the data without being altered by extreme values, and patterns in our plots were more interpretable.

To determine which of the eight features were the best, we plotted the conditional densities for each of the features (depending on class label) and for all 3 images. The idea is that if there is clear class separability based on the values that the feature takes on, it will be good for classification. (**SD**: Top Left, **NDAI**: Top Right, **CORR**: Bottom Left, **AN**: Bottom Right)

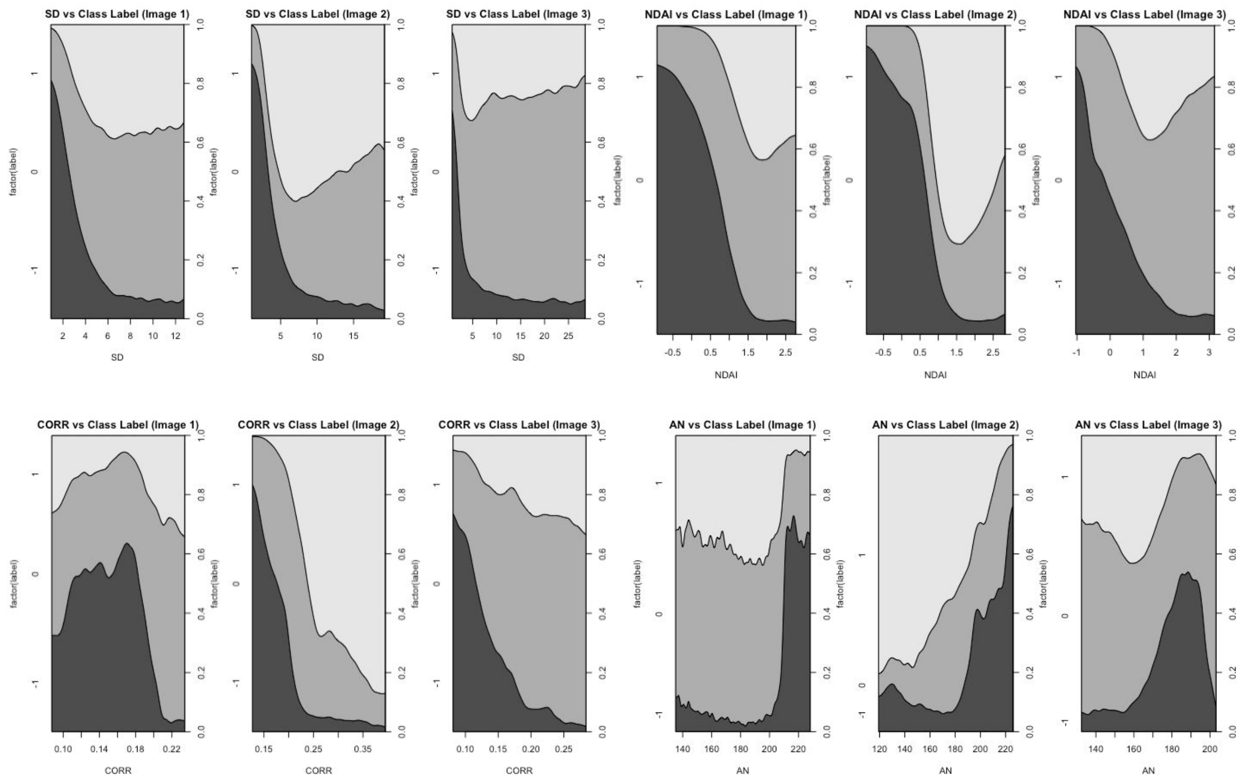


Figure 7

We see that for small SD values for all 3 images, for example less than 2, there is over an 80% chance that the pixel was classified as "no cloud", and virtually zero chance that it is classified as a cloud. While the chance that a pixel was classified as "cloud" goes up as SD goes up, so does the chance that the pixel was unlabeled. This means this feature really only gives good information when its value is very low, with the exception of image 2, where higher values of SD correspond to over a 50% chance of the pixel being classified as "cloud".

You can clearly see how the distribution of class labels changes as NDAI changes. In general a large proportion of the pixels (practically 90%) were labeled as clear when NDAI is small. For more neutral values of NDAI there is a larger proportion of unlabeled pixels, which makes sense because a neutral value is less likely to confidently point to the correct class label. As NDAI increases we see that a pixel has the highest likelihood of being classified as cloudy, but it is also more likely that it is an unlabeled pixel. To summarize, NDAI would be a good feature to classify as "no cloud" when its value is as small as possible.

We see that pixels are more likely to be classified as cloudy as CORR increases, although the chance varies depending on the image, so this is not a consistent argument. Also, we see that pixels are more likely to be classified as "no cloud" for smaller CORR values. Although there does seem to be some relationship between the conditional distribution of class labels as CORR varies, it is not consistent. This may be a reason that CORR was paired with NDAI while setting thresholds in the paper. CORR may not predict class labels alone, but will be helpful with the other features present. The threshold in the paper did not generalize very well to our images, since very few points had a value of CORR greater than 0.75. To conclude, the class separability for the CORR feature is very weak, so it may not be the best feature to use.

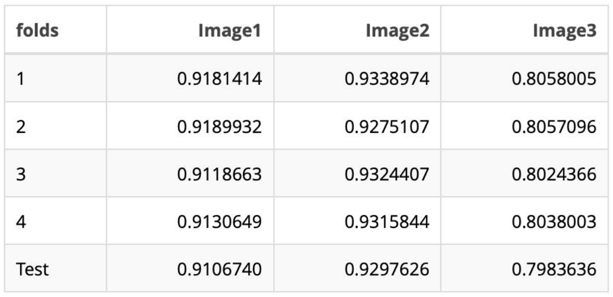
We found that radiance angles DF, CF, and BF had poor separability, so the plots were omitted to save space. The conditional density plots for AN are practically identical, but slightly better,compared to the ones for AF, which makes sense since AN is the closest to to AF. Since they practically classify in the same way, having both of them together is redundant. As AN increases, the likelihood that a pixel is classified as "cloudy"" is goes up considerably. For low values of AN, it is very unlikely that the pixel is classified as "not cloudy". AN is one of the best features since its values can clearly separate pixels by class.

<u>Conclusion:</u> Our three best features are NDAI, SD, and AN. Our definition for "best" are the 3 features that give the most separability in the conditional density plots. For example, if there is a clear relationship between the value of the feature and the conditional density, such as if the value of the feature increases, then the pixel is more likely to be classified as not cloudy, then it is a good feature.

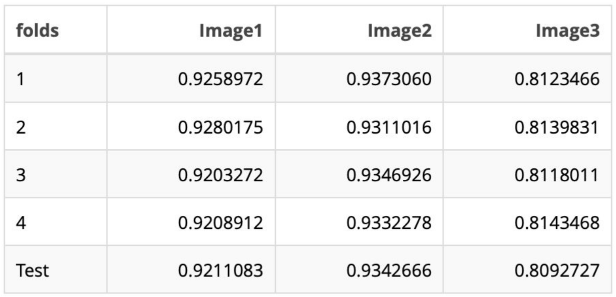
PART 3: MODELING

We now move on to our modelling process, where we used 4 different classification methods: Logistic Regression, Linear Discriminant Analysis (LDA), Quadratic Discriminant Analysis (QDA), and k-Nearest Neighbors (kNN). We fitted models for all 4 methods using both data splitting methods, which yielded different results which will be mentioned below.

Method 1 Split

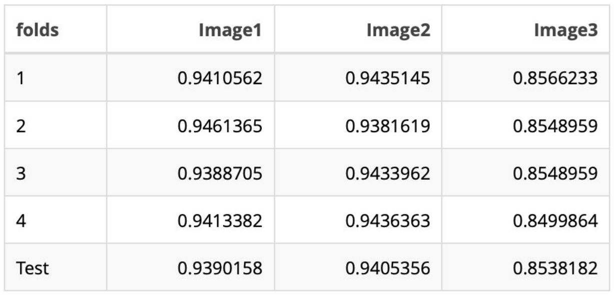


folds	Image1	Image2	Image3
1	0.9181414	0.9338974	0.8058005
2	0.9189932	0.9275107	0.8057096
3	0.9118663	0.9324407	0.8024366
4	0.9130649	0.9315844	0.8038003
Test	0.9106740	0.9297626	0.7983636

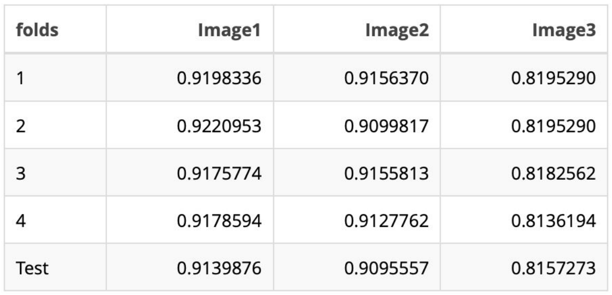


folds	Image1	Image2	Image3
1	0.9258972	0.9373060	0.8123466
2	0.9280175	0.9311016	0.8139831
3	0.9203272	0.9346926	0.8118011
4	0.9208912	0.9332278	0.8143468
Test	0.9211083	0.9342666	0.8092727

Logistic Regression (Left), LDA (Right)



folds	Image1	Image2	Image3
1	0.9410562	0.9435145	0.8566233
2	0.9461365	0.9381619	0.8548959
3	0.9388705	0.9433962	0.8548959
4	0.9413382	0.9436363	0.8499864
Test	0.9390158	0.9405356	0.8538182

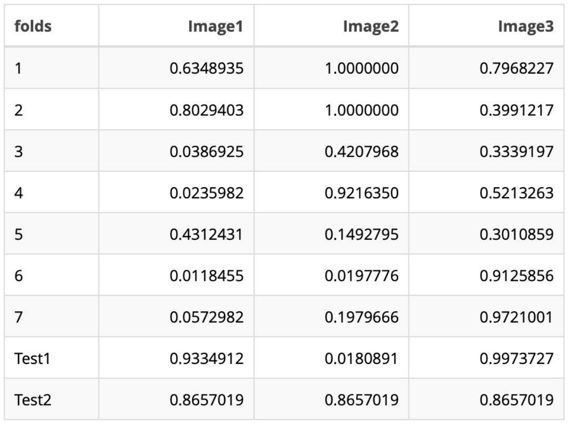


folds	Image1	Image2	Image3
1	0.9198336	0.9156370	0.8195290
2	0.9220953	0.9099817	0.8195290
3	0.9175774	0.9155813	0.8182562
4	0.9178594	0.9127762	0.8136194
Test	0.9139876	0.9095557	0.8157273

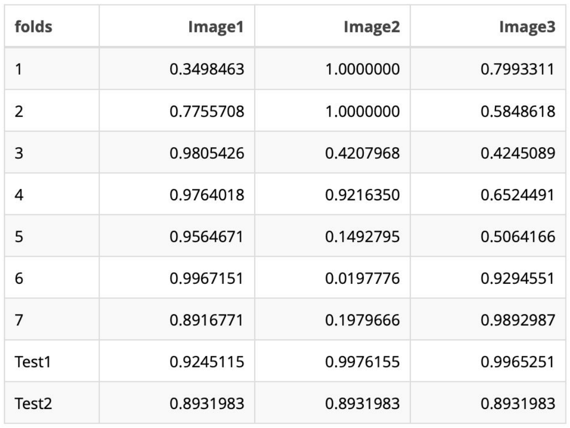
kNN (Left), QDA (Right)
Figure 8

Figure 8 shows the accuracies across folds and the test accuracy for logistic regression, LDA, k-Nearest Neighbors, and QDA. The four different classifiers seem to have very similar accuracies. However, k-Nearest Neighbors seems to have more accurate results compared to the rest, so we will look at this classifier in more detail below.

<u>Method 2 Split</u>

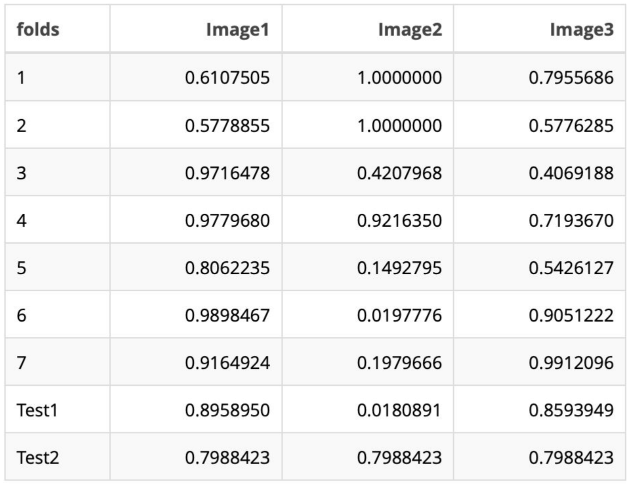


folds	Image1	Image2	Image3
1	0.6348935	1.0000000	0.7968227
2	0.8029403	1.0000000	0.3991217
3	0.0386925	0.4207968	0.3339197
4	0.0235982	0.9216350	0.5213263
5	0.4312431	0.1492795	0.3010859
6	0.0118455	0.0197776	0.9125856
7	0.0572982	0.1979666	0.9721001
Test1	0.9334912	0.0180891	0.9973727
Test2	0.8657019	0.8657019	0.8657019



folds	Image1	Image2	Image3
1	0.3498463	1.0000000	0.7993311
2	0.7755708	1.0000000	0.5848618
3	0.9805426	0.4207968	0.4245089
4	0.9764018	0.9216350	0.6524491
5	0.9564671	0.1492795	0.5064166
6	0.9967151	0.0197776	0.9294551
7	0.8916771	0.1979666	0.9892987
Test1	0.9245115	0.9976155	0.9965251
Test2	0.8931983	0.8931983	0.8931983

Logistic Regression (Left), QDA (Right)



folds	Image1	Image2	Image3
1	0.6107505	1.0000000	0.7955686
2	0.5778855	1.0000000	0.5776285
3	0.9716478	0.4207968	0.4069188
4	0.9779680	0.9216350	0.7193670
5	0.8062235	0.1492795	0.5426127
6	0.9898467	0.0197776	0.9051222
7	0.9164924	0.1979666	0.9912096
Test1	0.8958950	0.0180891	0.8593949
Test2	0.7988423	0.7988423	0.7988423

kNN

Figure 9

Figure 9 shows the accuracies across folds and the test accuracy for logistic regression, QDA, and k-Nearest Neighbors. We tried doing LDA using the method 2 split, but the accuracy was so inconsistent that it didn't give any insight. This is most likely because it is not a compatible model, since the distribution of features is not the same, so the classes cannot be modeled with the same covariance matrix. This explains why QDA works for this split method, and even has very high accuracy, because of the nature of the different class label distributions between blocks. Unlike the random split, the accuracies between models and images varied a lot, which is expected because the random split has folds with equal distribution of class labels, but the geographical blocking does not.

For split method 1, we plotted ROC curves to compare the different classification methods. We only are displaying plots for figure 1 since the results were the most consistent, and the goal is to compare classification methods, not splitting methods. Overall, kNN had the best true positive rate without sacrificing an increase in false positives. On each of the plots, you can see the optimal cutoff value indicated by the dot on the bend on the curve. Each cutoff value was chosen to obtain the highest false positive rate without sacrificing increases in false negatives. This cutoff is obtained by choosing the pair of TP/FP values that minimize the distance to the point (1,0). There are cases where this point may not be optimal, like if you care more about a higher true positive rate, but don't mind an increase in false positives. Or, there may be cases, such as the diagnosis of diseases where we want to make the false positive rate as small as possible, since a FP misclassification can have extreme consequences. However, for our purposes, since the consequences of misclassifying a pixel is not severe, we choose the cutoff value as explained before.

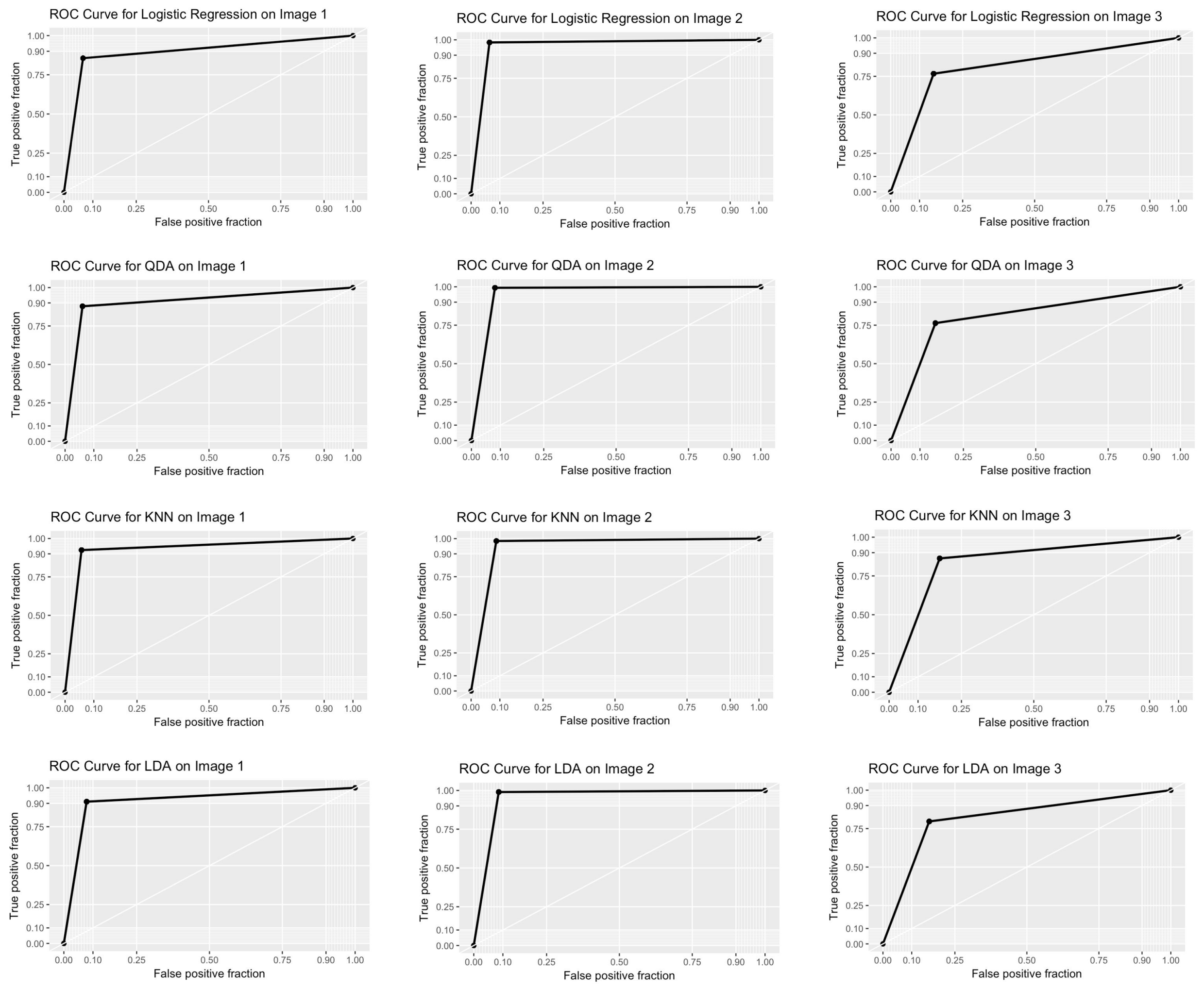


Figure 10

PART 4: DIAGNOSTICS

With the kNN classifier, it takes the attributes of its k nearest neighbors, so the k is the key tuning parameter of the algorithm. The goal of these plots is to show which value of k is optimal for the kNN.

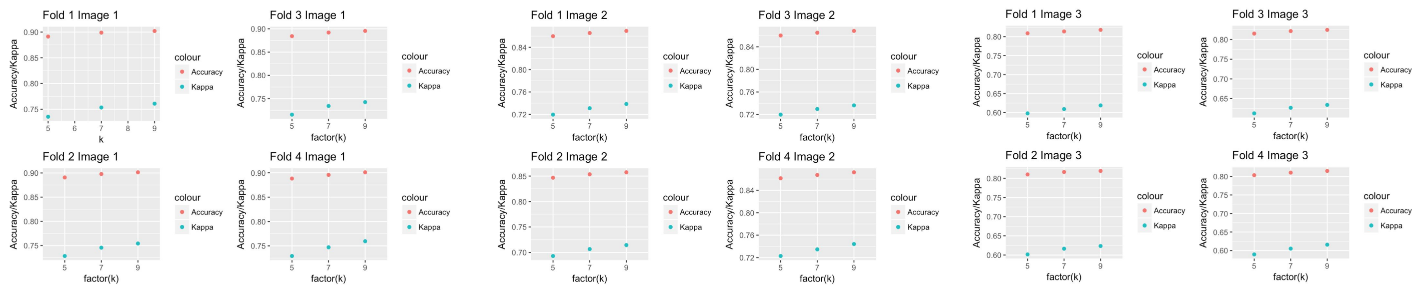


Figure 11

We the method 1 split method to determine information about parameter estimation because the results were more consistent than the results of method 2 split. Figure 11 shows the results of fitting kNN to each of the four folds for each image and calling "results" from the model output. Across the board, as the number of neighbors increases, the prediction accuracy increases. The results also had a kappa statistic, which is a measure of agreement between true labels and predicted labels, but is a more robust measure than simple percent-agreement calculation.

Continuing with diagnostics, we looked for patterns in misclassification using the method 2 split. To do this, we plotted images of all 9 blocks, then obtained each block's validation accuracy. We decided to use image 2, because it had the lowest average accuracy between folds for method 2 split. The accuracies we obtained for the 7 training folds are: 99.9%, 99.7%, 60.68%, 95.10%, 99.57%, 35.75%, and 63.97%. Now, we take a closer look at the blocks that had lower accuracies, which are folds 3, 6 and 7.

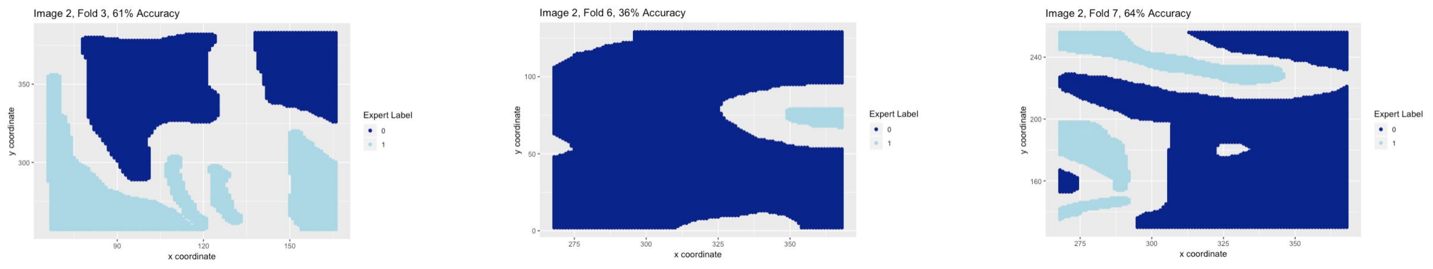


Figure 12

From initial inspection, it makes sense that these blocks had poor accuracy since none of them are linearly separable. Since we are using kNN, which takes the feature values of the 9 nearest neighbors for each data points, then classifies by majority vote, pixels are likely to be misclassified when they are surrounded by data points of the opposite class. You can see this in each of the plots, where one class has a much smaller grouping and is surrounded by a much larger grouping of another class. To get a better idea of what the misclassifications were, we obtain predicted labels and compared them to expert labels. Then, we looked for rows where the predicted label doesn't equal the true expert label, and see which misclassification is most common and try to explain why based on images. While we did this for each of the 7 folds, only folds 3, 6 and 7 had significant misclassifications. To justify why we omitted the other plots, we display the image of a fold with 98% accuracy, and it is obvious that the data points are linearly separable.

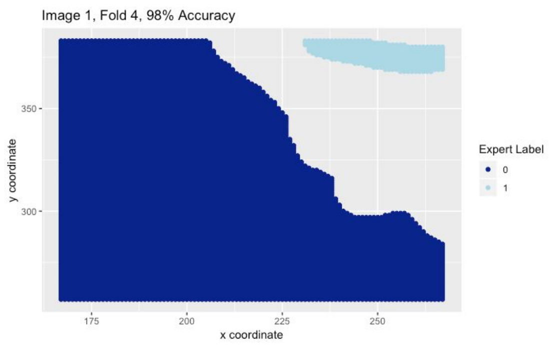


Figure 13

For fold 3, we got 2910 "non cloud" pixels misclassified as "cloud", but only 2 "cloud" pixels misclassified as "non cloud". For fold 6, there were 7050 "non cloud" pixels misclassified as "cloud", and no misclassifications for "cloud" pixels. For fold 7, there were 3136 "non cloud" pixels misclassified as "cloud", and only 18 "cloud" pixels misclassified as "non cloud".

We see that the most common misclassification with splitting the data using method two is non cloudy pixels being misclassified as cloudy pixels. By looking at the graphs, this makes sense, because there is a much higher frequency of non cloudy pixels, and those non cloudy pixels that are on the border along with unlabeled pixels next to cloudy pixels are likely to have similar feature values with cloudy pixels, hence the misclassifications.

From parts a and b, we can conclude that although k-Nearest Neighbors seems to be the most accurate classifier out of the four that we tested, there are still very low accuracy rates for our Method 2 of splitting the data. We noticed certain patterns with the kNN classifier when running it on the second split method. It was not as accurate when the blocks were not linearly separable. The other classifiers also did not seem to work as well on the second split method, especially with logistic regression. Logistic regression is a type of binary classification method and during our split, some of the images seem to only have one categorical variable after removing the unlabeled points, so the accuracy was very low. We decided to look into other methods and found that random forest might be a good classifier to use. The random forest classifier creates a set of decision trees from the training set. In the paper, the researchers mentioned certain thresholds for their selected features. These decision trees are more of a rule based system which is great for setting boundaries on the features as the same set of rules will be used to perform the prediction on the test dataset. Random forest also will handle missing values, which is good in our case due to the sheer number of unlabeled points, and is good for classifying categorical variables, which is key to our data set images. Thus, It will be helpful in predicting future data when the expert labels are not given.

We now discuss how our diagnostic results change based on which split method was used. It turns out that our results do change drastically based on the methods of splitting our data. For method 1 split, the prediction error on the validation set for each of the 4 folds was pretty uniform, since by using a random split, the distribution of class labels per fold is practically the same. With method 2, however, the distribution of class labels for folds vary greatly. So some folds had a very high prediction accuracy (nearly 100%), while some of the folds had accuracies as low as 35%. The value of k, the number of neighbors in kNN, depends on the fold for method 2 of splitting the data, because the distribution of class labels is different between folds. With method 1, the random split, the distribution of class labels between all 4 folds is very similar, so k = 9 neighbors is uniformly the best across all folds. In the method 2 case, the highest accuracy was achieved by different k values (5, 7, or 9)

Conclusion:

The purpose of this project was to examine data obtained from NASA's MISR and evaluate the performance of different classification methods for detecting clouds over polar regions. We first started off by conducting EDA to visualize the data we were given. After seeing the images, it was clear that our classification method had to use features that would provide good enough separability to differentiate between cloudy pixels and non cloudy pixels. To start, we created two methods for splitting the data in order to evaluate our models, which were random split and geographical blocking. As expected, model performance varied depending on the split. After conducting more rigorous EDA on the features, we came to the conclusion that NDAI, SD, and AN were the best features in terms of class separability. We then used these features, along with the two different split methods, to evaluate the performance of Logistic Regression, Linear Discriminant Analysis (LDA), Quadratic Discriminant Analysis (QDA), and k-Nearest Neighbors (kNN). In short, the random split gave more consistent results over the 4 methods, since each fold had an approximately equal distribution of class labels in them, while geographical blocking lead to a high variance of prediction accuracies, since in each of the blocks, the distribution of class labels could be very extreme. After computing the performance metrics for each of the models, we decided that kNN was our best classifier. In the diagnostics section, we fit the kNN model to different folds with different values of the hyperparameter k, the number of neighbors. To follow the trend from before, method 1 split had more consistent accuracy and agreement rates than method 2 split. Method 2 split had accuracy rates of practically 100% for some blocks, but single digit accuracies for others. By inspecting problematic blocks in terms of kNN model accuracy, we again confirmed that poor accuracy was due to some blocks using the method 2 split not being linearly separable. In conclusion, both split methods for kNN had their pros and cons. The random split was more consistent, but may not generalize well for new data that doesn't come in as random pixels. Method 2 split had more variance, but resulted in very high prediction accuracies and agreement rates when blocks were linearly separable. With this in consideration, aspects of both methods can be applied to deliver the best cloud detection results for future data.

GITHUB LINK: https://github.com/irenej98/Cloud-Detection