Cloudy with a Chance of Non-Clouds

William Tirone (ID: 2774025) and Natalie Smith (ID: 2819547)

March 2, 2024

1 Data Collection and Exploration

The purpose of the cloud imagery study was to be able to detect the difference between cloud and non-cloud pixels, a classification problem that proven difficult because clouds and ice have similar appearance. The distinction between these two substances is important when considering the warming of the Arctic, as "changes in the properties and distribution of ice- and snow-covered surfaces, atmospheric water vapor, and clouds can potentially lead to further warming and thus strong sensitivity to increasing amounts of atmospheric carbon dioxide.[1]" Accurate classification can help us figure out the impact of the clouds on Arctic warming.

The images used for classification were recorded automatically by satellite through the MISR satellite. The goal is to build a classification model based on these images to distinguish clouds from non-clouds in images using domain-knowledge-constructed features based on the radiance angles recorded. One of the three features looks at the average correlation of the nine radiation measurements, where high values (threshold 0.75) suggest clear or low-altitude clouds, and low values suggest cloudy. The second feature is SD, which helps identify smooth surfaces (threshold 2.0), where smaller values indicate smoother surfaces, and is constructed on five of the measurements. NDAI is a data-adaptive algorithm calculated with help of the other features. Clear surfaces are either smooth (low SD) or have high correlations. Clouds rarely exhibit both of these features simultaneously. The key area here behind the classification model was identifying cloud-free pixels rather than cloudy ones through the use of these features.

Some of the classification methods tried in this paper include SVM with Gaussian kernel, logistic regression (with S = 1, 2), and logistic regression with L1 penalty, while noting how clustering doesn't work. Nonetheless, the three features discussed above were able to separate the data well in terms of predicting the probability of cloudiness for partly cloudy data by training QDA on the ELCM labels. Model training either occurred on data units without expert labels (the unknown area), or on previously identified data. 10-fold CV was used with SVM.

With reliable classification methods, we can obtain a better look at global climate regarding the impact of clouds in the Arctic as atmospheric carbon dioxide increases globally.

1.1 Data Summary

Looking at the images with expert labels, we can see that all regions identified as "cloud" have surrounding spaces of "uncertain" classification before reaching "no cloud," indicating that there isn't a clear distinction between cloud and no cloud here (hence the need for the uncertain region). The percent of uncertainty seems to vary across the three images, but around 25-50 percent of the pixels cannot be distinguished between cloud vs non-cloud. Also for the three images, there are a differing proportion of pixels classified as non-cloud than as cloud with the difference ranging from around 3 - 25 percent.

Figure 1 below shows a map of the three images using the labels as color and the x,y pairs as the respective axes. The proportion of labelled points shows that there is not much relation between the three images and there is no obvious trend or pattern among them. Image 1 and 2 are roughly the most similar

with somewhat similar proportions of the three labels, and image 3 appears the most different with a high proportion of uncertain labels at about 51%.

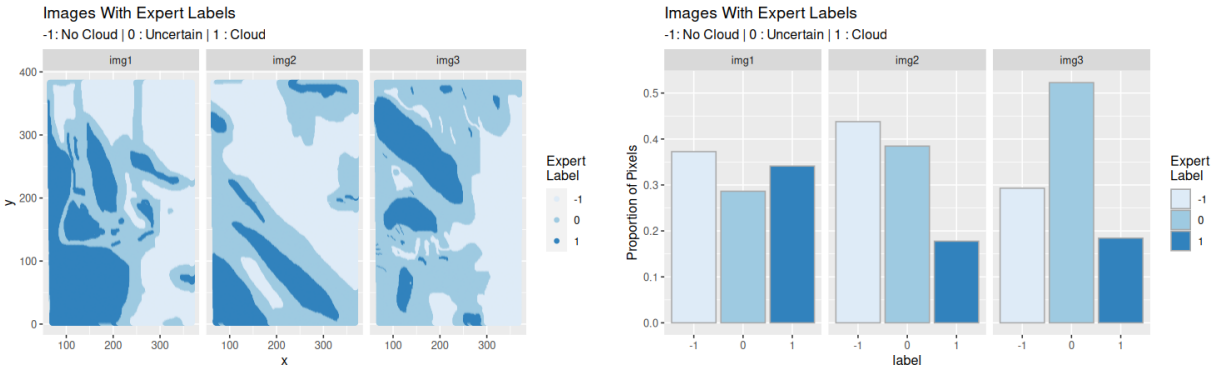


Figure 1: Map of X and Y Labels and Proportions of Labels

We cannot assume the data points are i.i.d. (independent and identically distributed), since the image data is spatially correlated. Intuitively, if a given pixel is labeled "cloud", there is a high probability that the adjacent pixels are also clouds. This means we cannot randomly sample to split the data and have to be more careful. This is addressed later in our datat split methodology.

1.2 EDA

1.2.1 i)

To perform EDA, we combined all three images and took a random sample of size 10,000. Since the combined image data was more than 1 million pixels, this allowed for quicker plotting and comparisons through use of a representative sample. While this removes the spatial correlation, it is justified since we are just comparing the features against each other.

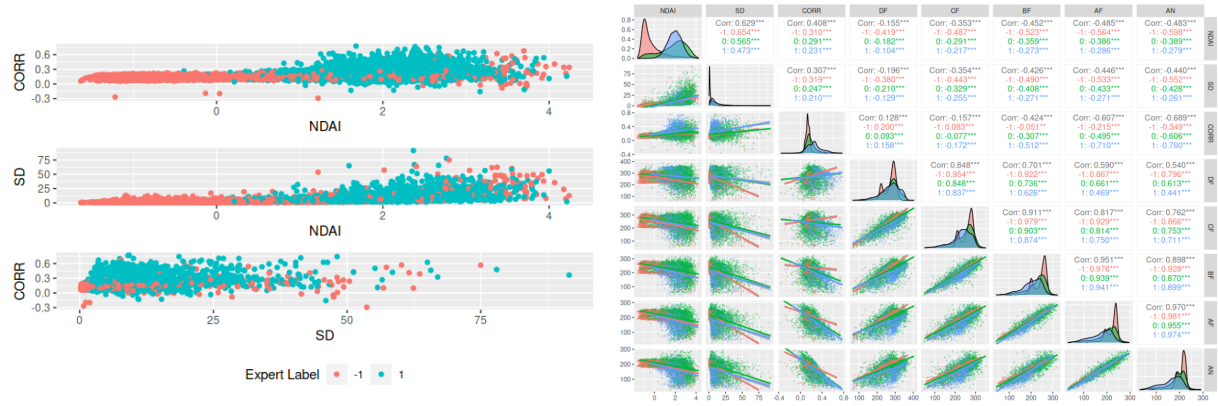


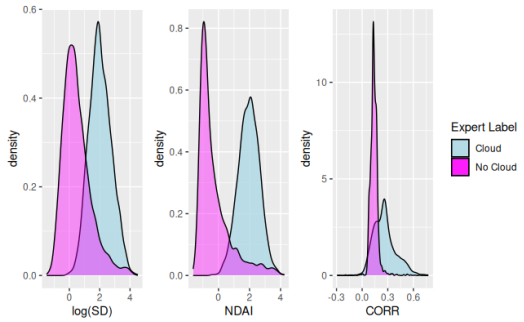
Figure 2: Pairwise Plotting of All and Select Features

Immediately, we chose to use NDAI from the pair-wise plot above (Figure 2) since the densities among the three labels look quite distinct with different means and different peak heights. Ideally, we want three distinct features that "contribute" new information without much overlap from previous features. Looking at scatterplots, densities, and correlations of DF, CF, BF, AF, and AN, there is a high amount of similarity and collinearity, so we thought that there may be some merit in the author's choice to use NDAI, SD, and CORR to yield better classification results. We plotted these in the left plot of Figure 2, and while there is some overlap, these features seem to separate the data quite well and will be useful in classification. Note

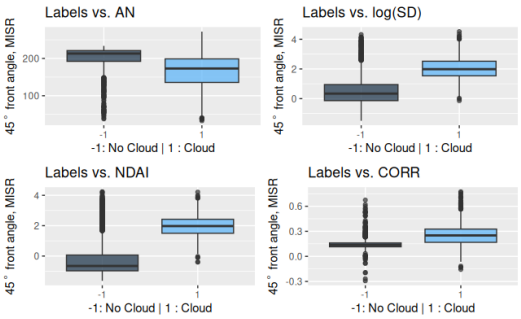
that we removed the unlabelled points in this plot to contrast with the ggpairs plot. By doing so, the data looks more separable, which shows promise for fitting our classification methods.

1.2.2 ii)

Examining the domain-expert-constructed features, we can see a relationship between the label of the pixel (cloud or non-cloud) and its value of CORR, NDAI, and SD. Looking at pairwise scatterplots (Figure two, left hand side), we observe that points classified as clouds generally have higher SD, NDAI, and CORR values than points seen as non-clouds. When considering theses density distributions on all of the radiance features, we noticed that DF values aren't well separated as the values tend to overlap between classes. There is greater separability between the cloud vs non-cloud classification for CF, BF, AF, and AN (non-cloud exhibits the higher values). Particularly, in the AF and AN densities, the non-cloud values have a much smaller standard deviation than the cloud values. We plotted the densities and boxplots for the most distinguished features across categories in Figure 3. There is a very apparent separation in NDAI between expert labels, and still notable separations in CORR, SD, and AN (Figure 3). (Note: we converted SD to a log scale for visualization purposes.)



(a) Box Plots of Cloud Labels vs. Select Features



(b) Feature Densities

Figure 3: Variable Importance Plots

In Figure 4, we performed PCA on all the features with the three categories of labels. This further confirms our use of CORR, NDAI, and SD, since these PCs seem to capture much of the variance and have a different direction compared to the radiance measurements by themselves.

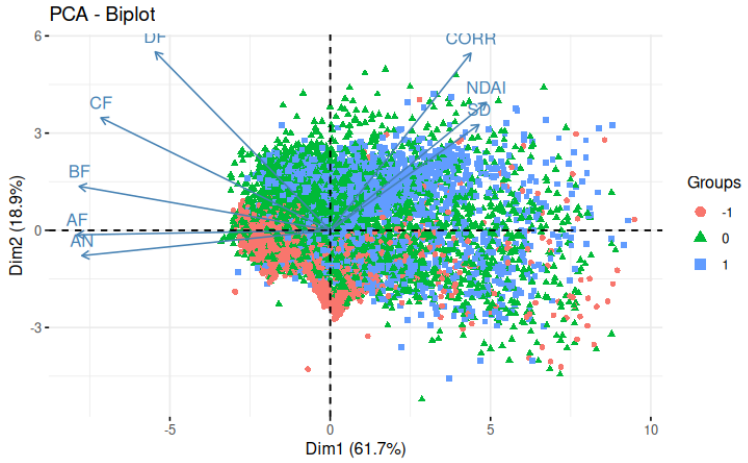


Figure 4: PCA of All Features

2 Preparation

2.1 Data Split

The first, and most naive approach, to separate the data would be to simply treat the three images as training, validation, and test data sets respectively. We know the data is not i.i.d, so it would be inappropriate to randomly sample points for our different sets. While this is naive, we suspect it may not be a bad method since it will give us three distinct sets to work with and has no computational cost, which could be large if we were working with enormous images or hundreds of images.

Second, since the data is correlated, we can hand-pick slices from each of the three images that result in the desired split proportionally, but in a way that retains the ordering. If we split each image into three chunks, we can then choose the first, middle, and last chunk for each of the desired sets and combine them. If there is some kind of feature or interesting detail present in the "position" of a slice of the data, hopefully this will allow us to capture it without overstating accuracy in training. To do this, we calculated the indices by hand. Visually, this is displayed in Figure 5.

A third alternate method would have been to use our second approach, but split the images vertically and horizontally into different blocks, and choose random blocks for training, validation, and testing. We did not actually implement this, but we suspect it may achieve similar or slightly better results compared to our hand-slicing method.

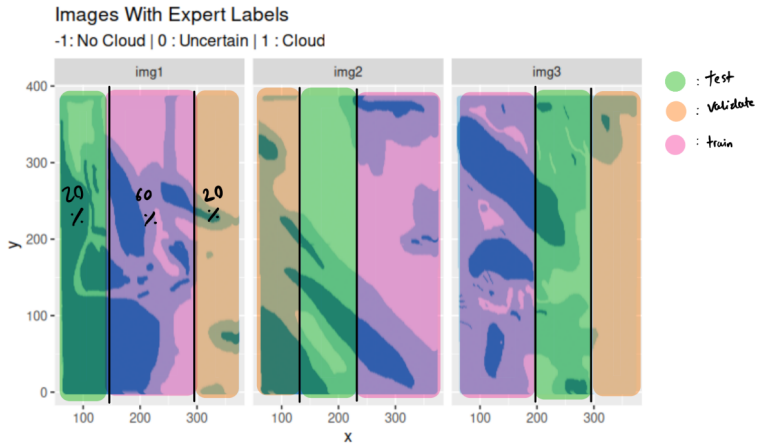


Figure 5: Hand-drawn Data Split, Block Split Approach

2.2 Baseline

We checked a trivial classifier (setting all points to -1) on both our validation methods outlined in the section above. The naive split test, naive split validate, block split test, and block split validation methods had classification errors of 28.9%, 38.6%, 38.2%, and 21% respectively. We checked the classification accuracy by removing the 0 labels which the experts considered too uncertain to label since we eventually fit our classifiers without these points.

This would have a high accuracy if the image was more cloud-free than not. Considering the right side of Figure 1, this is true in all the images. The trivial classifier is also better than a coin flip or randomly guessing in every case, so it sets a fairly high bar.

2.3 First Order Importance

Initially, we suspected that the author's choices for using CORR, NDAI, and SD would most likely produce the best classification results based on our EDA in 1.2 along with Figures 2 - 4. Intuitively, we want features that overlap as little as possible so that we can explain more of the data with them.

The paper, though, describes features that are not present in our images - we only have access to the "forward" directions of the MISR camera (i.e. Df in our data indicates the forward direction) along with An (n for "nadir" direction). The MISR camera also has Aa, Ba, Ca, and Da with "a" corresponding to the "aft" direction, though we do not have access to these to attempt to construct alternative features.[1]

The best features would be ones that separate the two classes well. In our EDA (1.2.2, Figure 3), we discuss that CORR, SD, and NDAI do the best job here. This was confirmed to be a good choice through our modeling seen in Section 3.2.

2.4 CV Function

The *CVmaster* function in CVmaster.R takes a generic classifier, training features and labels, number of folds, loss function, and fit type and outputs a dataframe with the losses calculated at each fold. We wanted to add as few additional arguments as possible to make the function easier to interpret, but the challenge was that some models returned probabilities, which then needed to be classified with a threshold, and some returned the actual class membership, like QDA. For a more in depth look, please view the function itself.

3 Modeling and Model Comparison

Initially, we started by merging the validation and training sets for both our naive method and block splitting method to perform cross validation as the instructions stated. Then, we fit the models below on both split methods and performed cross validation on each. We have included R output of the misclassification error for both split methods. "Val Method 1" corresponds to using each image as train, validation, and test (the "naive split") and "Val Method 2" corresponds to the block split from Figure 5. The error is reported at each fold and a percentage difference column is calculated for each as well.

For every model that follows other than Logistic With L1 Penalty, we used NDAI, CORR, and SD as covariates; for Logistic with L1 Penalty, we used every feature to see if the penalty provided better accuracy than our hand selection of covariates. The output below is the result of the **CVmaster()** function from CVmaster.R.

3.1 Model 1 - Logistic Regression

Logistic regression performed quite well with both split methods on train and test data. Both splits had training error rates near 10%.

This classifier achieved a test error rate of 12.87% on the naive split method and 9.57% on the block split method, both of which were better than the naive classifier. We would hope that the second method has a higher accuracy since it does a little more to train the model on different cloud vs. non-cloud slices, and that seems to be the case here.

Though we set a prediction threshold at 0.5, a nice added benefit of logistic regression is that it outputs a probability of class membership. It would be interesting to modify this threshold to determine if better classification metrics could be obtained.

3.1.1 Logistic Regression Assumptions

This model assumes a linear relationship of the covariates. While this is most likely not the case, our goal was to choose covariates, like NDAI, CORR, and SD, that had a "linear enough" relationship that classification could be effective. Logistic regression also assumes that the observations are independent, which is obviously not true in this case since the data is spatially correlated, but we hope that our chosen features are "independent enough" to be effective. Considering the right side of Figure 2, the camera angles by themselves would have been too collinear to use by themselves.

k <dbl>	Val Method 1 <dbl>	Val Method 2 <dbl>	percent_diff <dbl>
1	0.1087131	0.1023637	-5.840460
2	0.1108356	0.1088754	-1.768575
3	0.1093773	0.1071173	-2.066246
4	0.1082835	0.1093964	1.027729
5	0.1106817	0.1038547	-6.168196
6	0.1098797	0.1065963	-2.988135
7	0.1148378	0.1089405	-5.135262
8	0.1077002	0.1064079	-1.199851
9	0.1058043	0.1094686	3.463304
10	0.1055855	0.1112920	5.404595

Figure 6: Logistic Regression CV

3.2 Model 2 - Logistic Regression with L1 Penalty

We chose Logistic Regression with an L1 Penalty since the authors mentioned that they used it, and we hadn't implemented it before in class or in homework. We used every feature available to us to observe the effect of both classifying and variable selection with the L1 penalty. While this performed fairly well on the naive split method, it did the worst out of all classifiers on the block split method. The betas on the right side of Figure 7 indicate, though, that our choice of NDAI, CORR, and SD for the other classifiers was a good one since their values were the farthest from 0 out of all covariates.

To choose a λ for the penalty term, we cross-validated with **cv.glmnet()** which resulted in lambdas of 0.0001504401 for the naive split method and 0.0001370769 for the block split. While these are quite small, the l1 logistic regression accuracy had a slight improvement over the standard logistic regression in the block split method (but was slightly worse on the first).

This classifier achieved a test error rate of 13.5% on the naive split method and 9.18% on the block split method, both of which were better than the naive classifier.

3.2.1 Logistic Regression with L1 Penalty Assumptions

This model makes the same assumptions as standard logistic regression but has an added penalty to help us shrink unimportant variables.

k <dbl>	Val Method 1 <dbl>	Val Method 2 <dbl>	percent_diff <dbl>
1	0.10157503	0.3535074	248.0258
2	0.10324462	0.3578095	246.5648
3	0.10360919	0.3585126	246.0239
4	0.10303340	0.3569814	246.4716
5	0.10309880	0.3593408	248.5403
6	0.09756453	0.3588304	267.7877
7	0.10193962	0.3558877	249.1162
8	0.10178637	0.3554032	249.1659
9	0.10529386	0.3572731	239.3104
10	0.09785621	0.3523407	260.0596

```
8 x 1 sparse Matrix of class "dgCMatrix"
                s0
NDAI  1.54695589
SD   -0.06568933
CORR 20.39674983
DF    0.02451470
CF   -0.03210193
BF   -0.02277778
AF    0.00315880
AN    0.04621354
8 x 1 sparse Matrix of class "dgCMatrix"
                s0
NDAI  1.97733756
SD   -0.08716890
CORR 10.25771676
DF    0.01690722
CF   -0.01205675
BF   -0.02548931
AF   -0.01099908
AN    0.04217529
```

Figure 7: Logistic Regression with L1 Penalty CV and Coefficients

3.3 Model 3 - QDA

QDA performed well on both split methods on the training and validation data with a misclassifcation error around 10% at each fold (Figure 8). It achieved test errors of 8.10% and 7.80% on naive and block split methods respectively.

3.3.1 QDA Assumptions

QDA assumes that the underlying data generative mechanism is Gaussian and that each class has its own covariance matrix.[2] We think that the covariance matrices are likely different considering the density plots on the left side of Figure 3, though the distributions do not appear approximately Normal for our three chosen covariates. Figure 3 also shows that "No Cloud" seems to have wider tails than the "Cloud" labels and is more right-skewed except in the case of CORR.

k <dbl>	Val Method 1 <dbl>	Val Method 2 <dbl>	percent_diff <dbl>
1	0.1082756	0.1017192	-6.055300
2	0.1102443	0.1048382	-4.903723
3	0.1068251	0.1030216	-3.560512
4	0.1068174	0.1079638	1.073273
5	0.1074814	0.1043172	-2.943911
6	0.1130232	0.1035291	-8.400119
7	0.1113461	0.1055545	-5.201440
8	0.1098797	0.1052940	-4.173374
9	0.1106898	0.1062708	-3.992283
10	0.1081377	0.1062777	-1.720023

Figure 8: QDA CV

3.4 Model 4 - Boosting

We implemented XGBoost since we've heard that it typically performs very well in data science competitions and it was recently discussed in class. Though we didn't tune the parameters using cross validation (we used default parameters supplied in an article by the authors of XGBoost and its R implementation[3]), it performed well on the train and validation sets and performed the best of any classifier on the block split method with an average of 8.7%.

Interestingly, this achieved test misclassification errors of 7.08% and 9.48% on the two split methods, with the naive split method being the lowest misclassification of any method we tried.

3.4.1 Boosting Assumptions

Boosting is non-parametric and makes no underlying distribution assumptions, so there are no obvious violations of assumptions by fitting this model here.

k <dbl>	Val Method 1 <dbl>	Val Method 2 <dbl>	percent_diff <dbl>
1	0.10237713	0.08536824	-16.613961
2	0.10018959	0.09182079	-8.352964
3	0.09865111	0.08562870	-13.200468
4	0.10433832	0.08758791	-16.053932
5	0.10178637	0.08960083	-11.971674
6	0.10178637	0.08770673	-13.832533
7	0.09924165	0.08986715	-9.446133
8	0.09916873	0.08777185	-11.492420
9	0.10595012	0.08627987	-18.565577
10	0.09924165	0.08582964	-13.514495

Figure 9: XGBoost CV

3.5 Model Comparison - ROC Curves

We've continued our analysis through selecting the naive and block methods for QDA and XGBoost for further exploration and comparison. Comparing the four ROC curves selected, we noticed both of our alternative approaches for QDA and XGBoost fared better than the naive approaches in terms of the AUC for our ROC curves. A higher area under the curve provides evidence for a stronger model,[4] but all of the QDA and XGBoost have very high AUC values close to the ideal value of 1. Our Naive QDA gave an ROC value of 0.959, and our block split QDA has a slightly better value of 0.963. Our Naive XGboost has an ROC value of 0.953, and our block split XGboost has a slightly better value of 0.967. So, our alternative XGBoost appears to model the data best when considering accuracy - however, this AUC is so close to the XGboost that both do an equally good job.

Naive QDA: In an ROC curve (Figure 10, left side), sensitivity (true positive rate) is displayed on the y-axis and selectivity (1 - false positive rate) is on the x-axis. Considering our first ROC curve for Naive QDA, we are looking for a cutoff point that is close to (1, 1) as it helps us achieve more accuracy in our classifications (the upper left corner). In this model, the sensitivity of 0.98 is the probability that the model correctly predicts a cloud. The corresponding selectivity is around 0.91. Using the geometric mean as a measure of optimality, we have our optimal threshold of 0.39. That means we should classify anything above 0.39 as a cloud and below that as non-cloud. The red line indicates the ideal sensitivity for this threshold and where the red line crosses the black line reflects the ideal selectivity.

Block Split QDA: In this ROC curve (Figure 10, right side), our ideal threshold (again, using the geometric mean) gives a sensitivity of 0.98 and a selectivity of 0.86, indicating we will correctly classify 98% of clouds to be clouds and 45% of clouds as non-clouds. Here, our optimal threshold is 0.52, indicating that anything with a probability 0.52 or above should be classified as a cloud, and anything below as non-cloud.

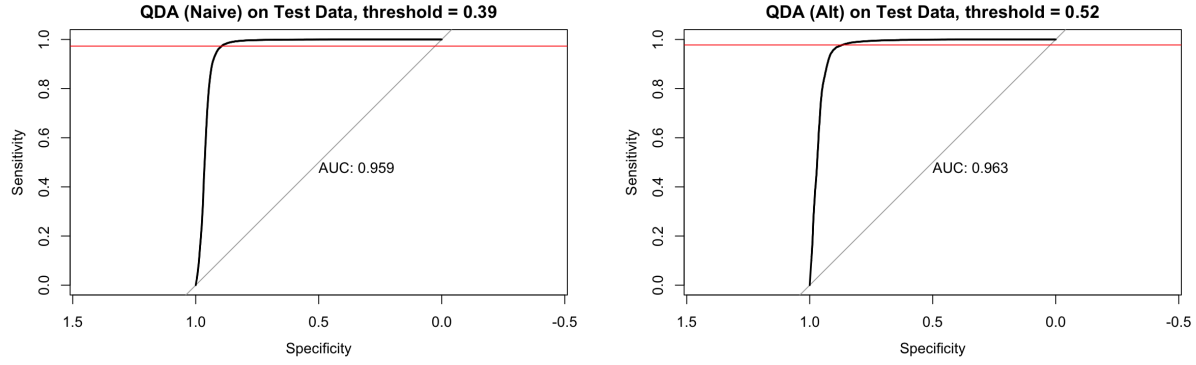


Figure 10: QDA ROC Curves

Naive XGBoost: In this ROC curve (Figure 11, left side), our ideal threshold appears to be one that gives a sensitivity of 0.97 and a selectivity of 0.91, indicating we will correctly classify 97% of clouds to be clouds and 55% of clouds as non-clouds. Here, our optimal threshold is 0.39. Anything above 0.39 should be classified as a cloud, and anything below as non-cloud.

Block Split XGBoost: In this ROC curve (Figure 11, right side), our ideal threshold appears to be one that gives a sensitivity of 0.98 and a selectivity of 0.86, indicating we will correctly classify 98% of clouds to be clouds. Here, our optimal threshold is 0.52. Anything above should be classified as a cloud, and anything below as non-cloud.

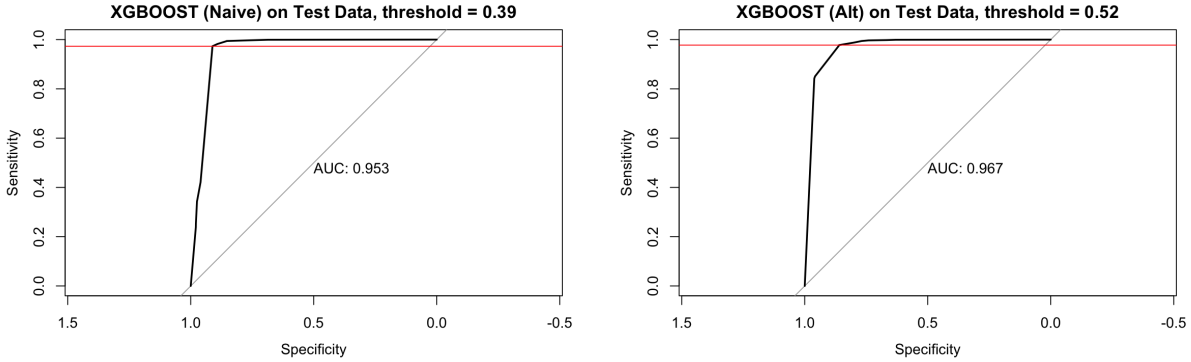


Figure 11: XGBoost ROC Curves

4 Diagnostics

4.1 QDA Diagnostics

The ROC discussion above cover the sensitivity and specificity interpretation, and the misclassification discussion in section 4.4 covers the positive and negative predicted values. Still, the confusion matrices and statistics in Figure 12 give us a more in depth look. The "No Information Rate" metric, the rate we need to exceed to prove we have a useful classifier, is much lower for the block split than the naive split. Nonetheless, both models have statistical significance with small p-values such that its accuracy is greater than its no information rate. Considering the rest of the accuracy measures, it seems QDA on both data split methods are almost indistinguishable and could be used for similar levels of overall effectiveness, but we would still recommend block splitting to try to find more distinct sections of images on which to train the model.

4.2 XGBoost Diagnostics, Naive Split:

We chose two passes of the data with the parameter nrounds=2 which produced two iteratively computed trees. In the second round, Tree 1, the first split was on SD which was different from Tree 0. The "gain" metric, which measures the information gain metric of the split (the importance of the node[3]) was similar for the first node of both trees, though for the first tree, CORR had a huge amount of importance over NDAI with a gain of 9161 vs. 1533. In the second tree, this was only 2122 compared to 810 for the same metrics. See Figure 13 for results.

4.3 XGBoost Diagnostics, Block Split:

We used the same parameters for the block split and observed roughly similar trees. The biggest difference is the split value for CORR which was -0.043 for the naive split and 0.236 for the block split.

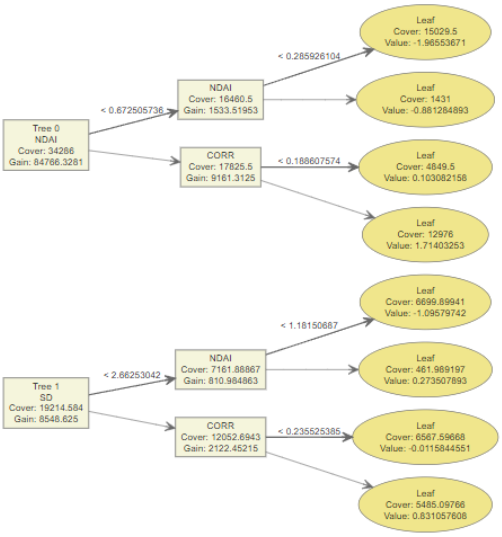
```
Confusion Matrix and Statistics

          Reference
Prediction    -1     1
        -1 47321  3125
        1   2622 17849

               Accuracy : 0.919
                 95% CI : (0.9169, 0.921)
    No Information Rate : 0.7042
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.8041

 Mcnemar's Test P-Value : 3.546e-11

            Sensitivity : 0.9475
            Specificity : 0.8510
         Pos Pred Value : 0.9381
         Neg Pred Value : 0.8719
             Prevalence : 0.7042
         Detection Rate : 0.6673
   Detection Prevalence : 0.7113
      Balanced Accuracy : 0.8993

       'Positive' Class : -1
```

(a) QDA Test Fit, Naive Split Method

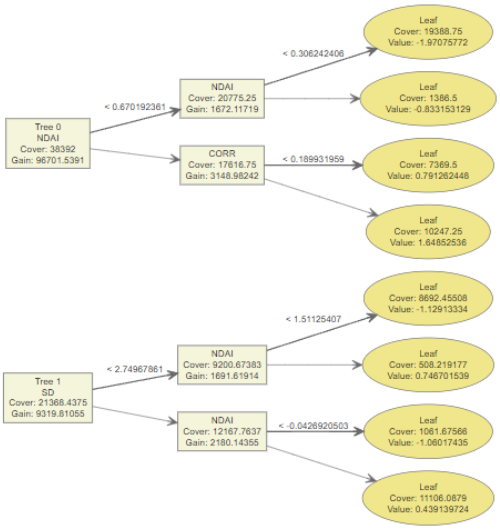
```
Confusion Matrix and Statistics

          Reference
Prediction    -1     1
        -1 30154  3500
        1    751 20088

               Accuracy : 0.922
                 95% CI : (0.9197, 0.9242)
    No Information Rate : 0.5671
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.8389

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.9757
            Specificity : 0.8516
         Pos Pred Value : 0.8960
         Neg Pred Value : 0.9640
             Prevalence : 0.5671
         Detection Rate : 0.5534
   Detection Prevalence : 0.6176
      Balanced Accuracy : 0.9137

       'Positive' Class : -1
```

(b) QDA Test Fit, Block Split Method

Figure 12: QDA Confusion Matrix and Diagnostics



(a) XGBoost, Naive Split Method



(b) XGBoost, Block Split Method

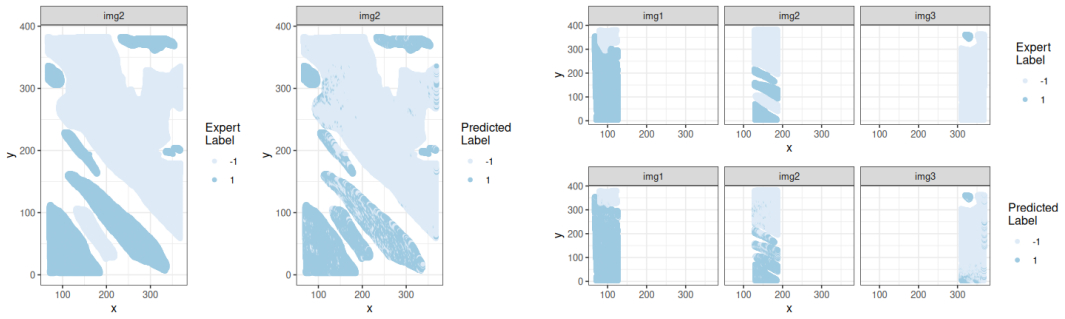
Figure 13: XGBoost Trees

4.4 Misclassification Patterns

In both QDA and XGBoost, we noticed similar misclassification patterns. Considering the Naive approach for QDA and XGBoost, most of the misclassification comes from clouds being predicted as non-clouds. Further, most of the classification errors appear to be near each other (not random). This pattern is seen in all four images in Figure 15.

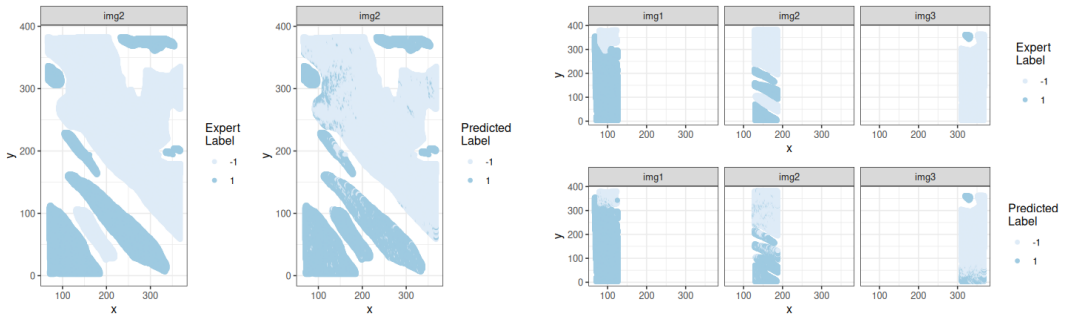
When analysing the block-split method, there is difficulty in classifying the 2nd image block – there are a lot of non-cloud pixels classified as clouds in the lower half of the image. We believe this occurred because, when examining the actual photo, the non-cloud region is surrounded by clouds. It is likely the

values of NDAI, SD, and CORR were all similar and the surrounding similarity likely influenced this gap of non-clouds and was missed when classifying.



(a) Naive Split Method (b) Block Split Method

Figure 14: QDA Comparison to Expert Labels



(a) Naive Split Method (b) Block Split Method

Figure 15: XGBoost Comparison to Expert Labels

To support our findings here, we can look at this same idea numerically. When looking at everything the experts labeled as a cloud (dark blue), our classification models correctly predicted the pixel as a cloud 87% of the time in Naive QDA, 96% of the time in Block QDA, 97% of the time in Naive XGBoost, and 98% of the time in Alternative XGboost.

Similarly, when looking only at the true non-cloud data pixels (light blue), our classification models correctly predicted the pixel as not a cloud 94% of the time in Naive QDA, 90% of the time in Block QDA, 91% of the time in Naive XGBoost, and 86% of the time in Alternative XGboost.

It appears the block methods perform better than the naive methods when it comes to correctly predicting data that are truly clouds, and does worse than the naive methods when it comes to dealing with non-cloud data. It also appears that all four of our classifiers are better at correctly predicting clouds than correctly predicting non-clouds.

	Naive QDA	Block QDA	Naive XGBoost	Block XGBoost
Truly Cloud	0.8719164	0.9639618	0.9723511	0.9776381
Truly No Cloud	0.9380526	0.8960005	0.9116679	0.8603435

Figure 16: Correct Classifications

4.5 Comments and Future Work

A better classifier would likely be one trained on many more images, since our current classifier has only been trained on three. Since pixel values are spatially dependent on each other, despite having many pixels trained on, it would be useful to have more images overall.

Further, our model was only trained on expert labels of 0 and 1, indicating the expert was confident in these pixels. It is likely that the confidence of the expert corresponds to feature values of NDAI, SD, and CORR that were more separable between clouds and non-clouds. In our exploratory data analyses, features classified as "uncertain" by experts would exhibit values of the features between non-cloud values and cloud values. Thus, for intermediate values where our expert can't determine if the pixel is a cloud or non-cloud, we are uncertain if the classifier would be able to correctly classify this point.

So, to improve our classifier, it would be very beneficial to have better experts to label the "uncertain" values as either a cloud or not a cloud, so that our model would be able to handle difficult pixel values.

4.6 Conclusion

Overall, our two best classifiers, XGBoost and QDA, were able to very effectively predict whether or not the data point contained a cloud. Notably impressive, our Naive classifier, which was tested on a never-before-seen image, was still able to distinguish between cloud and non-cloud quite well. Still, our block split method that extracted training, validation, and test data across all three images performed better than our naive split. This makes sense, as we were able to train our data on three different images with different spacial patterns. XGBoost and QDA performed very similarly, with QDA edging out XGboost in many metrics. However, since boosting works by aggregating many weak learners rather than relying on a single learner, we believe the XGboost approach would be more robust to future images.

5 Acknowledgements

We both contributed equally and did much of the work collaboratively. Will wrote the CV master function, fit the models and tested, made about half the EDA plots and suggested the alternative data split methodology. Natalie wrote the summary, performed EDA and made plots, wrote about model comparison, made the ROC curves and wrote the discussion, discussed misclassification patterns and made those plots, and wrote the conclusion. To complete the project, we read the paper and did research on classification methods through ISL / ESL and the lecture notes. Then, we fit models that we were more familiar with like logistic regression and QDA, then branched out to logistic with l1 penalty and XGBoost. We researched basic diagnostics and tried to understand how well our models fit. Thank you to Annie and Santa, our cats.

References

[1] Eugene E. CLOTHIAUX Tao SHI, Bin YU and Amy J. BRAVERMAN. Daytime arctic cloud detection based on multi-angle satellite data with case studies. *Journal of the American Statistical Association*, 103(482), 2008.

[2] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An Introduction to Statistical Learning: with Applications in R.* Springer, 2013.

[3] Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 785–794, New York, NY, USA, 2016. ACM.

[4] https://www.geeksforgeeks.org/plotting-roc-curve-in-r-programming/. Online; accessed 12/1/2022.