STA 521 Project 2: Cloud Recognition

Xiaozhu Zhang (xiaozhu.zhang@duke.edu) and Xuyang Tian (xuyang.tian@duke.edu)

1. Data Collection and Exploration

(a) Summary of Yu (2008)

In this paper, the authors propose the first detection algorithm to accurately characterize the properties of clouds over the daylight Arctic, efficiently processing the massive MISR dataset without requiring human intervention (expert labeling). The authors use the results of an extensive exploratory data analysis and interactions with the MISR science team to construct a labeling scheme, or detector. For each data unit, they combine clustering information on the current data unit with the detectors learned from previous data units. Thus, the algorithm is sequential, and it adaptively combines classification and clustering methods. The methodological novelty in their approach is to search for cloud-free instead of cloudy surface covered by ice and snow.

The data used in this study were collected from 10 MISR orbits of path 26 over the Arctic, northern Greenland, and Baffin Bay. The repeat time between two consecutive orbits over the same path was 16 days, so the 10 orbits span approximately 144 days from April 28 through September 19,2002 (a daylight season in the Arctic). The authors chose path 26 for their study because of the richness of its surface features. Therefore, the data investigated contained 57 data units with 7,114,248 1.1-km resolution pixels with 36 radiation measurements for each pixel. They use the 275-m red radiation measurements to build some of their features, so the actual size of the data set is even larger.

The result is astonishing: the ELCM algorithm agreement rate of 91.80% over the 5 million testing pixels is 8.57% higher than the MISR ASCM (83.23%) algorithm and 11.90% higher than the SDCM (80.00%) algorithm. This represents a significant improvement from both scientific and statistical standpoints. The authors expect that more reliable polar cloud properties also will translate into more accurate global climate model simulations through improved model cloud physics. These studies will eventually enable the scientific community to study how changing cloud properties may enhance or ameliorate any initial changes in the Arctic brought about by increasing concentrations of atmospheric carbon dioxide.

(b) Maps and percentages

The first step of our project is to replicate the dataset in the form of three graphs (see Figure 1). We utilize blue color to denote the clouds (i.e. data points labeled as 1) and orange color to represent the non-cloud data points labeled as -1, to simulate the color of rock. For the data points without expert label, i.e. labeled as 0, we choose grey color to show that there is no decision yet.

From the graphs, it is clear that both clouds and non-clouds appear in the form of several large blocks. And we believe that the expert labels are quite conservative, leaving all uncertain points as unlabeled. Thus, clouds and non-clouds are separated by a relatively large margin and do not overlap. It would fair to say that observations in this dataset (representing pixels) are not independent and identically distributed, since pixels in the same blocks must reserve certain spatial structures and similar patterns.

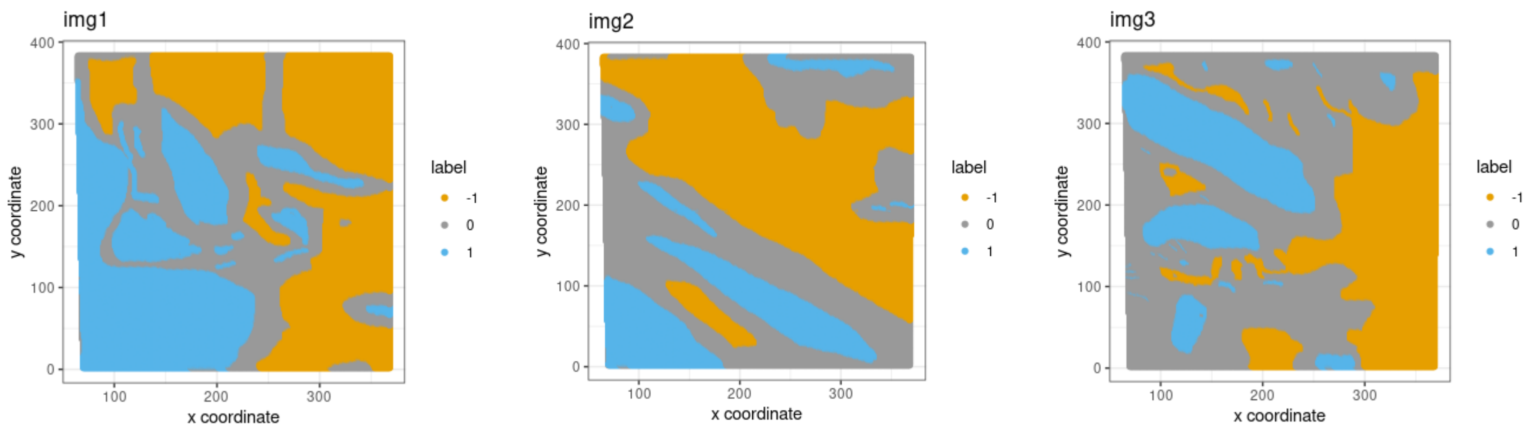

Figure 1: *Three maps of with labels. Orange area indicates "non-cloud", grey area indicates "unlabeled", while blue area indicates "cloud".*

Table 1: *Counts and percentages (shown in parentheses) of each label in the three image datasets.*

Label	Img1	Img2	Img3	Overall
-1	42882 (0.3725306)	50446 (0.4377891)	33752 (0.2929429)	127080 (0.3677552)
0	32962 (0.2863522)	44312 (0.3845560)	60221 (0.5226746)	137495 (0.3978950)
1	39266 (0.3411172)	20471 (0.1776549)	21244 (0.1843825)	80981 (0.2343499)

In addition, we also investigate the proportions of each label in these three datasets. The overall ratio is around 4:4:2 (for non-cloud, unlabeled, cloud). That is to say, we see no evidence of data imbalance and do not need to worry about this issue at least for now.

(c) EDA

We utilize the correlation plot (see Figure 2) to show the pairwise relationship between features. It is noticeable that here we decide to create one-for-all plot showing the correlation in the dataset of three images combined rather than creating a single plot for each image.

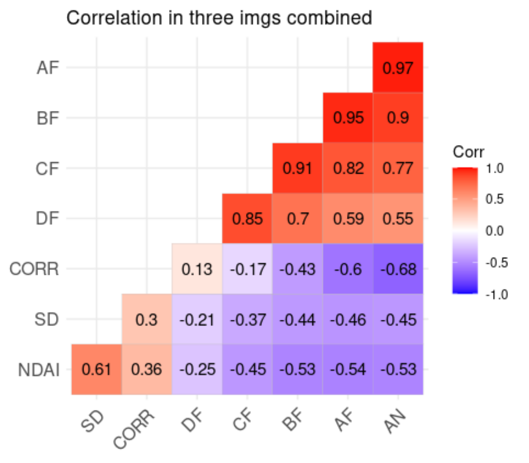

Figure 2: *Correlation plot of eight features in the dataset (three images combined).*

To better investigate and understand the relationships between these variables, we can imagine that the parameters are divided into two categories: radiance angles {AN, AF, BF, CF, DF} and subject statistics {CORR, SD, NDAI} since all radiance angles are highly similar.

It is obvious that both of these two groups are highly correlated internally, especially for the radiance angle group where all correlations are bigger than 0.5 and most are over 0.8. However, the correlation between these two categories is relatively lower. For example, CORR is not very related to DF and CF, with a correlation only around 0.15.

To perform an exhaustive and comprehensive EDA, we also draw graphs to show the difference between feature densities of clouds and non-clouds (see Figure 3).

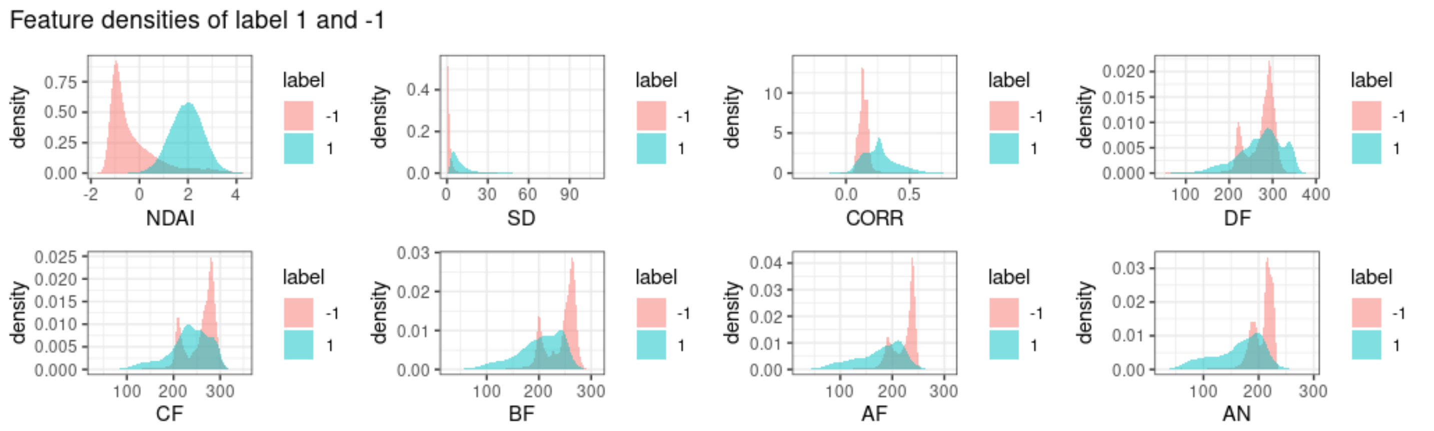

Figure 3: *The density plots for each of the eight features with label -1 (non-cloud) and 1 (cloud) respectively.*

For the graph featuring NDAI, it is clear that clouds have a larger NDAI on average. NDAI stands for the normalized difference angular index characterizing the changes in a scene with changes in the MISR view direction. This means clouds will cause larger direction changes. In addition, the NDAI variable has the most significant difference and thus we are confident that it will be an important feature in the following classification models as well.

For the graph featuring SD and CORR, clouds clearly have a large long tail while non-clouds' distribution is very concentrated in a peak. In other words, clouds are rather correlated to each other and have a larger variance. On the other hand, we can infer that the distribution of non-clouds ranges in a very small interval.

As for the seven graphs showing radiance angles, they have a lot in common. The dominant configuration is that non-clouds' distribution is still concentrated (and have two modals) whereas clouds have a long tail on the left-side this time. If the radiance angle data sit on the left long tail, we can directly spot the difference between clouds and non-clouds; otherwise, differentiating through eyeballing would be much harder since there is a decent amount of overlapping between the densities of clouds and non-clouds.

2. Preparation

(a) Data Splitting

After we complete our awesome pre-modeling analysis above, it is time to preprocess data to prepare for the subsequent modeling steps. Here, the data is quite clean and no further preprocessing is required for us. We can directly go to the split step to get data ready for the cross validation.

Due to the fact that quite a large proportion of data are unlabeled and their true labels are very ambiguous, we decide to drop all data points without expert label to improve the efficiency of our classifiers. In other words, from this point, only data points labeled with 1 or -1 will appear in either of our training, validation, and testing sets.

One notable thing about these three datasets is that they are all heavily encoded with location information and we want our splitting method to reflect this important information as well. Thus, we adopt two distinct types of data split. The first type is the random shuffling one which takes label balancing into consideration, while the second method preserves and highly emphasizes location information. The details of these two methods are as following:

<u>Type-1 splitting:</u> We adopt the classic random sampling and set the ratio of three datasets (training vs. validation vs. test) as 40%-30%-30%. However, we choose not to shuffle all data but for each class separately. That is to say, we first randomly sample 40% data points from observations with label -1 (no cloud), and then the same proportion of observations with label 1 (cloud). We combine these two parts together to get the overall training set. This is to guarantee that the proportion of cloud to non-cloud in each dataset match the overall ratio. With similar steps, we can get our validation and test set as well.

<u>Type-2 splitting:</u> We split the dataset into several blocks and manually pick blocks to constitute the train, validation, and test sets. For this method distinct from Type 1 method, our goal is to retain the spatial correlation or spatial structures of the data, while at the same time to balance the data splitting procedure so each set has adequate number of observations with different labels. As a result, our strategy is to first adopt K-means clustering, using the features "x coordinate" and "y coordinate", to divide the graph into several blocks. The resulting blocks preserve as much spatial correlation or structures as possible, and at the same time each block has similar number of data points. We should notice that due to the differences in numbers of labeled data points across these three images, the numbers of blocks in each graph are also different, ranging from 6 (the third graph) to 9 (the first graph). It is also noticeable that due to the concentrated distribution of clouds and non-clouds, as well as the nature of K-means clustering, some blocks consist of clouds only while others are non-clouds only. To address this issue and make our datasets balanced, we manually select blocks to form the three sets and help the proportion of cloud to non-cloud in each dataset match the overall ratio. In this way, we make sure each of our train, validation, and test sets all have plenty of cloud and non-cloud data points. Moreover, we deliberately make our blocks in each set as far away as possible. This is to alleviate the potential bias caused by adjacency and to make our model more robust.

The final splitting results and assignment are as following:

• Training:	img1 (1, 2, 4),	img2 (2, 3, 4, 8),	img3 (4, 5)
• Validating:	img1 (5, 7, 8),	img2 (6, 7),	img3 (2, 6)
• Test:	img1 (3, 6, 9),	img2 (1, 5),	img3 (1, 3)

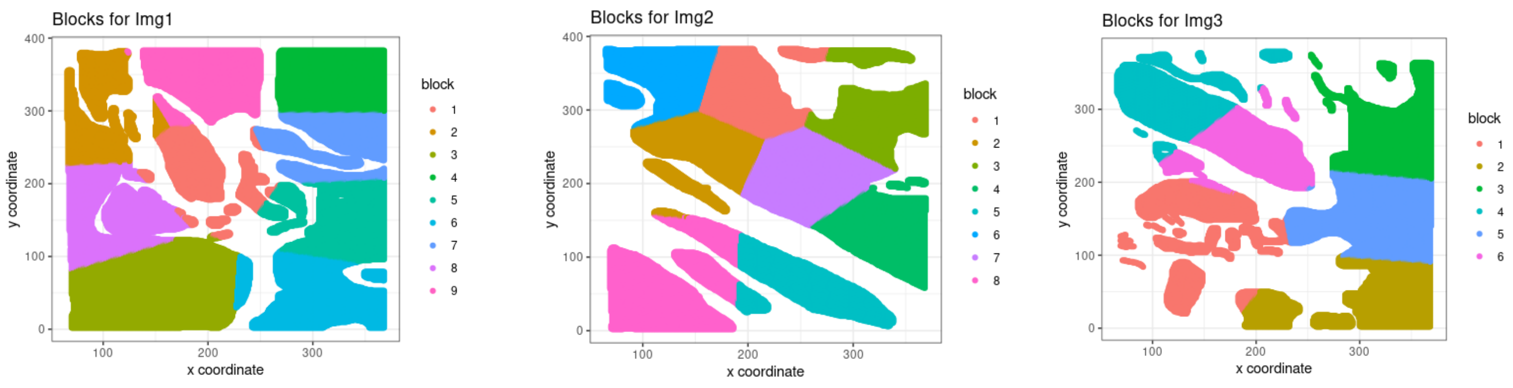

Figure 4: *Blocks obtained through K-means clustering for three images.*

To make supplementary, we believe that Type-2 splitting have several advantages over Type-1 splitting:

(1) We can take the two-dimension spatial information into full consideration. The resulting blocks reflect the inner spatial structures of cloud and non-cloud areas, while other splitting methods, such as using grids, are too rigid and unreasonable.

(2) K-means is flexible. We can choose any k value, i.e. any number of blocks in each image. Thus, we can modify k in each image, so that each block will have relatively equal data points. For example, image 1 has more labeled data and thus more blocks.

(3) For Type-1 splitting, since we select the data from random sampling, the pixels in train, validation, and testing sets are close to each other and heavily intertwined. In this case, we have to test on pixels that are extremely close to those that we build models on. This makes the prediction problem much easier and is far away from reality, where we have to make predictions on brand-new blocks of pixels. In this sense, Type-2 splitting uses tests points that are far from training sets and should lead to more stable results.

(b) Baseline

We adopt the suggested method to sets all labels to -1 on the validation set and on the test set to get the baseline accuracy:

For Type-1 splitting: baseline accuracy for validation set is 0.6107756, and for test set is 0.6107854.

For Type-2 splitting: baseline accuracy for validation set is 0.6682315, and for test set is 0.5309495.

Actually, this baseline accuracy for trivial classifier is just the proportion of data points labeled as -1 in these datasets. These proportions agree with our statement that there are no data imbalance phenomenon in our sets. If we have more non-cloud points, then we will have a higher standard for our classification models.

This baseline accuracy provides us a very decent criterion for our classifier. If our classifier's performance is poorer than that, all our efforts is meaningless. Particularly, if we have a classifier with accuracy lower than 0.5, even simply flipping all predictions will give us a better result. These situations are definitely something we want to avoid.

(c) First order importance

We choose F-statistic as our quantitative justification metric:

$$F = \frac{\text{variance between groups}}{\text{variance within groups}}.$$

As the definition of F-statistic shows, it characterizes the ratio of between-group variance to within-group variance. A higher F-statistic means higher between-group variance and lower within-groups variance. In other words, features which are very important and have major differences across groups, will have high F-statistic.

The variables with highest F-statistic are NDAI, CORR, and AF. In addition, these three features are not highly correlated, as justified in the previous section (see Figure 3). Thus, they are our best features.

Table 2: *F- statistic for all eight features*

Feature	NDAI	SD	CORR	DF	CF	BF	AF	AN
F- statistic	281706	48842	90707	24.21	18080	52146	72108	71073

(d) Cross-Validation method

For Type-1 splitting, we want to make sure that each fold has the same proportion of cloud and non-cloud data points. Thus, through random shuffling, we divided the datasets labeled with 1 and those labeled with -1 into K equal-sized folds respectively again. Then we combine each cloud fold with one non-cloud fold to get a complete fold, and repeat this process for K times.

For Type-2 splitting, to reserve the spatial information, we choose not to shuffle or treating the data separately. We directly break each block into K subblocks, pick one subblock from each of all blocks in training and validation sets, and combine these subblocks to form a fold. Then we repeat this process for K times to get K folds. Codes are placed into the folder and submitted to Gradescope.

3. Modeling

(a) Cross-validation classification results

After establishing a comprehensive and solid cross-validation strategy, the next step is to perform the classification task with diverse classifiers and to figure out which classification algorithm is the best fit for this problem.

We choose five methods in total: Logistic Regression, K-Nearest Neighbor, LDA, QDA, and Naïve Bayes. These algorithms have different characteristics and assumptions. We will illustrate them in detail.

Logistic Regression assumes 1) dependent variable are binary 2) observations are independent of each other 3) there are little or no multicollinearity. We believe that the zero-multicollinearity assumption is somewhat violated because the radiance angles features are highly correlated. In addition, the independent assumption is also violated since each data point is heavily influenced by its neighbors in each block.

K-NN is a rather "model-free" algorithm and has no assumption on the data. However, it is noticeable that during the training process, we always choose odd numbers for the value of K (the number of neighbors). Otherwise, if we have an even number as K, we will probably face the tie problem: half of the neighbors are from first class and another half are from the second class. The tiebreaker to fix this issue is kind of tricky: either choosing one class at random or re-evaluation based on the distances and weights of neighbors. Either method is complicated and unnecessary. Thus, we should always set the value of K as an odd number.

LDA and QDA all make Gaussian assumptions. LDA makes a strong assumption that the density follows multivariate normal distribution with class-specific mean and shared covariance. On the other hand, QDA makes a slightly weaker assumption that the density follows multivariate normal distribution with class-specific mean and class-specific covariance. Unfortunately, here the assumptions of LDA and QDA are seldom satisfied. Finally, Naïve Bayes, which is another variant of LDA, makes a different type of strong assumption that within each class all predictors are independent. Nevertheless, this is quite against the reality of this problem as well.

It is clear that none of the assumptions described above are satisfied per se. However, violations in assumptions does not necessarily lead to bad or worse performances in terms of prediction in practice, and empirically people apply these models regardless of these assumptions and "get their hands dirty": Indeed, "in handling messy situations, it is impossible to come away clean". With that being said, we still include these methods as our options.

Table 3: *Accuracies of five methods across folds, their averages and accuracies on the test set.*

	LR		KNN		LDA		QDA		NB	
Splitting type	1	2	1	2	1	2	1	2	1	2
Fold 1	0.8907	0.8666	0.9292	0.8879	0.8926	0.8670	0.8879	0.8767	0.8506	0.8705
Fold 2	0.8895	0.8923	0.9305	0.9419	0.8912	0.8932	0.8899	0.9142	0.8480	0.8857
Fold 3	0.8939	0.8982	0.9331	0.9385	0.8953	0.8996	0.8921	0.9154	0.8552	0.8799
Fold 4	0.8923	0.8960	0.9289	0.9446	0.8951	0.8954	0.8906	0.9031	0.8507	0.8737
Fold 5	0.8918	0.9049	0.9295	0.9175	0.8949	0.9031	0.8897	0.9047	0.8511	0.8715
Average	0.8916	0.8916	0.9302	0.9261	0.8938	0.8917	0.8900	0.9028	0.8511	0.8763
Test	0.8945	0.8998	0.9325	0.7874	0.8965	0.9030	0.8914	0.8879	0.8508	0.8549

In Table 3, we only report the performances when the threshold for probability is 0.5. The number of neighbors for KNN is chosen to be 5 through cross validation.

In general, all these classification methods do a relatively pretty good job on the validation fold, for both data types of splitting, with an average accuracy around 0.88. Due to the strong assumption on data which is not fully satisfied, Naïve Bayes has a slightly lower accuracy.

However, when we come to the test set, the story is quite different. The most eye-catching difference is the sharp drop of accuracy for KNN in Type-2 splitting. At the same time, KNN's accuracy for Type-1 splitting even improves a little bit. At the first glance, this drop seems very counter-intuitive; nevertheless, after a thorough examination, the reason behind this drop is simple and insightful. Due to the nature of KNN algorithm, the test data tends to be more accurately classified if it is "close" to the training data in terms of features distance, while Type-1 splitting exactly result in such a scenario. On the other hand, as we mentioned earlier, Type-2 splitting breaks the images into blocks and the blocks for test set are far away from those for training and validation sets, so KNN tends to behaves worse in this case.

As for other algorithms, as we expect, Naïve Bayes' accuracy further decreases. The remaining three models: Logistic Regression, LDA, QDA are strong candidates for our best classification algorithm.

(b) ROC curve

In this section, we present the ROC curves for each classifier in both two types of splitting, and the cutoff points are marked as bold points in the graph (see Figure 5). The ten cutoff points have one thing in common: they are all at location where the slope is around 1. When we have steep line with slope larger than 1, the true positive rate quickly increases; and when we have a smaller slope close to zero, only false positive rate increases while true positive rate "freezes". So, this is a choice about tradeoff. We take both traits into consideration and select a position where both two metrics have meaningful increase. Thus, points get fair classification along our cutoff standard.

For Type-1 splitting, it seems that KNN has the largest AUC, approaching a high true positive very quickly. But we also notice that its slope after the threshold point is kind of larger, meaning that it converges to 1 more slowly than other methods. LDA, QDA, and Logistic Regression's ROCs all have similar shape and area under curves, while Naïve Bayes is the slowest to reach the threshold point and have the smallest AUC.

And for Type-2 splitting, KNN becomes the method with worst performance and lowest AUC this time, which agrees with our observation and justification in the last section. For the other four classifiers, there is no major change in AUC. However, we do notice that the curves for QDA and Naïve Bayes are pretty wiggly and their cutoff probabilities are suspiciously low as well, which may result in unstable estimations and predictions.

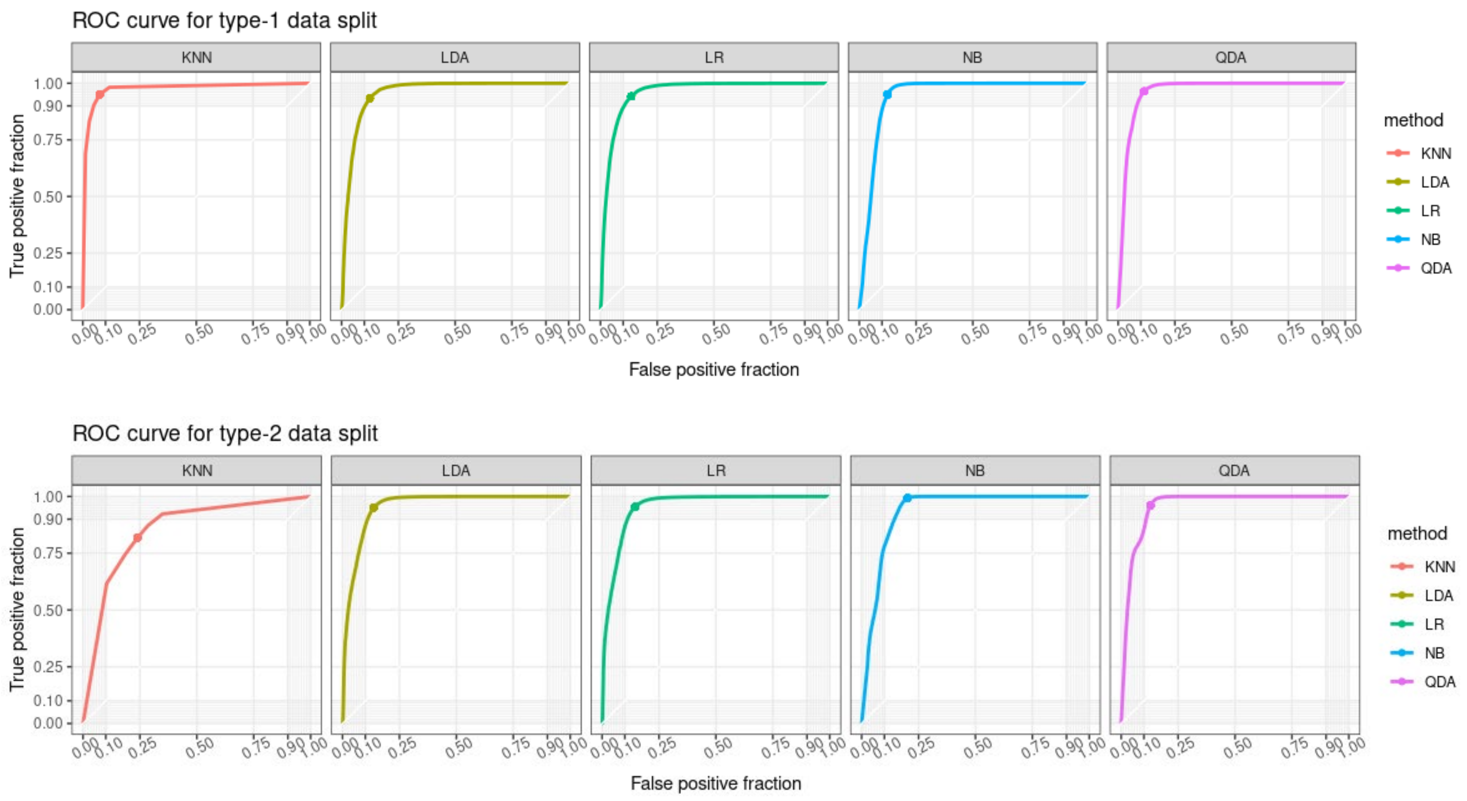

Figure 5: *ROC curves for both types of data splitting and five classification methods.*

Table 4: *Cutoff choices and resulting metrics.*

Method	Cutoff probability		False positive rate		True positive rate	
Data split type	1	2	1	2	1	2
LR	0.3	0.4	0.1357	0.1454	0.9421	0.9542
KNN	0.35	0.4	0.0742	0.2395	0.9504	0.8178
LDA	0.30	0.45	0.1241	0.1369	0.9331	0.9550
QDA	0.02	0.1	0.1142	0.1285	0.9639	0.9601
NB	0.015	0.005	0.1244	0.2012	0.9506	0.9932

(c) Bonus: other relevant metrics

Here we choose three classical metrics to measure the classification problem (or any problem regarding the performance of discrete quantities such as beta recovery in Hastie et al., 2020): recall, precision and F-score:

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}}, \qquad \text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}},$$

$$\text{Fscore} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}.$$

These statistics (see Table 5) further consolidate our discussion in previous two sections. Since Type-2 splitting is regarded to be more stable and closer to reality, we focus on the metrics of Type-2 splitting here. For Recall metric, LR, LDA, QDA and NB are all larger than 0.95. For Precision metric, LDA and QDA are the top 2. For F-score, similarly, LDA and QDA are the highest and greater than 0.90.

Generally speaking, LDA and QDA have slightly better statistics, so we choose to eliminate Logistic Regression from the candidate list for best classifier. In the following section, we will further investigate the question "Which is better, LDA or QDA?".

Table 5: *Recall, precision and F-score for five methods and two types of splitting.*

Method	Recall		Precision		F-score	
Splitting type	1	2	1	2	1	2
LR	0.9421	0.9543	0.8156	0.8529	0.8743	0.9007
KNN	0.9504	0.81784	0.8909	0.7510	0.9196	0.7830
LDA	0.9332	0.9502	0.8272	0.8598	0.8770	0.9027
QDA	0.9639	0.9601	0.8432	0.8685	0.8995	0.9120
NB	0.9506	0.9931	0.8297	0.8134	0.8860	0.8944

4. Diagnostics

(a) In-depth analysis of LDA and QDA

In this part, we choose to conduct the diagnostics from two directions. The first is to explore the convergence property of LDA and QDA in terms of accuracy when we vary the proportion of samples that go in to the training set. Here we use the whole test set to report accuracy, but randomly choose pixels of given proportions from the training set and validation set combined to fit the LDA and QDA models. The second diagnostics direction is to examine the robustness of LDA and QDA by adding perturbations to original dataset, adopting the procedure as follows: 1) calculate the variances of each feature; 2) generate a noise matrix whose rows are i.i.d. samples from the multivariate Gaussian distribution with mean 0 and covariance matrix as

$$r \cdot \text{diag}([\text{var}_1, \cdots, \text{var}_8]^\top),$$

where r is the given ratio, and var_i indicates the variance of variable i; 3) add the noise matrix to the original design matrix; 4) estimate parameters by LDA and QDA on the training and validation set; 5) report accuracies on the test set. For both directions, we report accuracy on two methods (LDA and QDA) and two types of splitting, resulting in four possible lines in Figure 6.

For the graph on the left of Figure 6, it is clear that even if we keep only a small proportion of data as our training set, we can still achieve a very impressive accuracy. This means that our model already converges well in terms of accuracy v.s. training sample size. In addition, LDA overall performs better than QDA, and particularly, LDA achieves higher accuracy under Type-2 splitting while QDA achieves higher accuracy under Type-1 splitting.

For the graph on the right of Figure 6, interestingly, the difference between these two types gradually increases as we add more perturbation. That is to say, due to the significant role of spatial information and block breaking, Type-2 splitting is more robust in case of large noise. In addition, even though the gap shrinks a little bit when the induced variance is as six times as the original variance, this scenario seldom come true and thus does not weaken our statement.

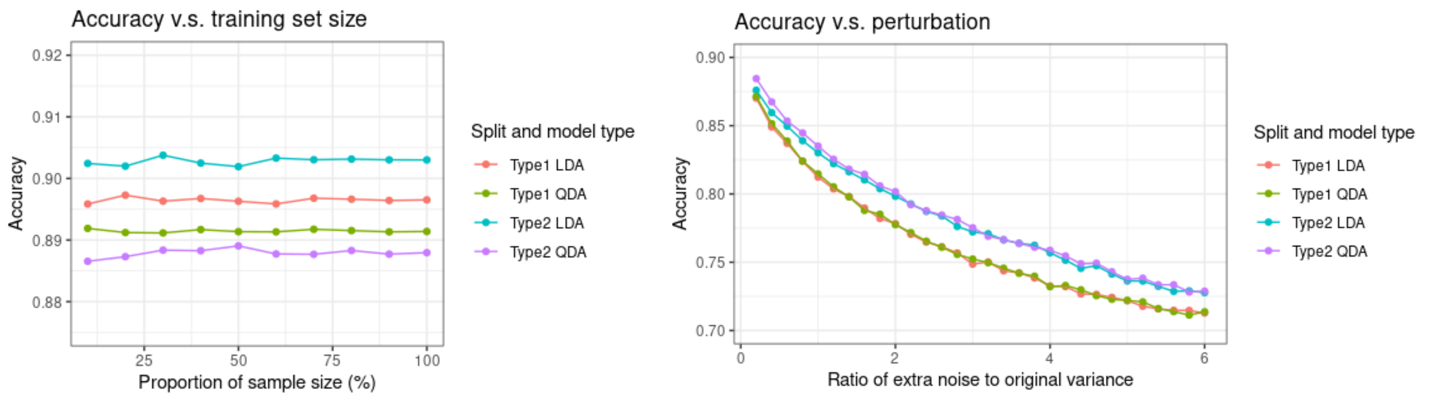

Figure 6: *Two directions of diagnostics. (Left) Accuracy v.s. proportion of sample size (%) for LDA and QDA with two types of splitting. (Right) Accuracy v.s. ratio of extra noise to original variance for LDA and QDA with two types of splitting.*

Based on these two analyses, and our preferences over Type-2 splitting, we conclude that LDA is our best classifier among these five models.

(b) Misclassification pattern

For our best classifier, LDA, the overall test accuracy for Type-1 splitting is 0.895856 and the overall test accuracy for Type-2 splitting is 0.892861, if we train our model only on the training set, and report the accuracy on both validation and test sets (so not surprisingly, the accuracy is slightly lower than that in section 3). The reason we decide to do so is that we want to observe more blocks on the images when we use Type-2 splitting, so that we can observe more potential misclassification patterns.

Figure 7 shows the correctly-labeled point in blue and misclassified points in red. Since for Type-1 splitting, pixels in training, validation, and test sets are generally close to each other, the final pattern looks like the whole dataset while actually the training set is not in this graph. For Type-2 splitting, it is clear that the graphs only contain the validation and test sets.

We can clearly see that all misclassified points appear in large chunks rather than random uniform distribution. Between the two types of splitting, Type-2 has a more concentrated misclassification pattern.

There are large blocks of misclassification on both the image1's right-bottom corner and image 2's middle-bottom island. This fact leads us to suspect that the expert labels of these regions are not valid and needs further investigation. Our misclassification might be caused by some internal properties of these regions.

To better understand what the major differences behind correct classifications and misclassification are, we implement another feature density plots just like what we did in section 1 (see Figure 8). Interestingly, we see a major difference in NDAI again and confirm that it is not only the most important variable, but also the variable that algorithms need to pay further attention to. For the rest of features, SD and CORR does not show too much difference, while all radiance angles show much difference. As a result, in future further analysis, we should focus more on the regions where misclassification accounts for high density.

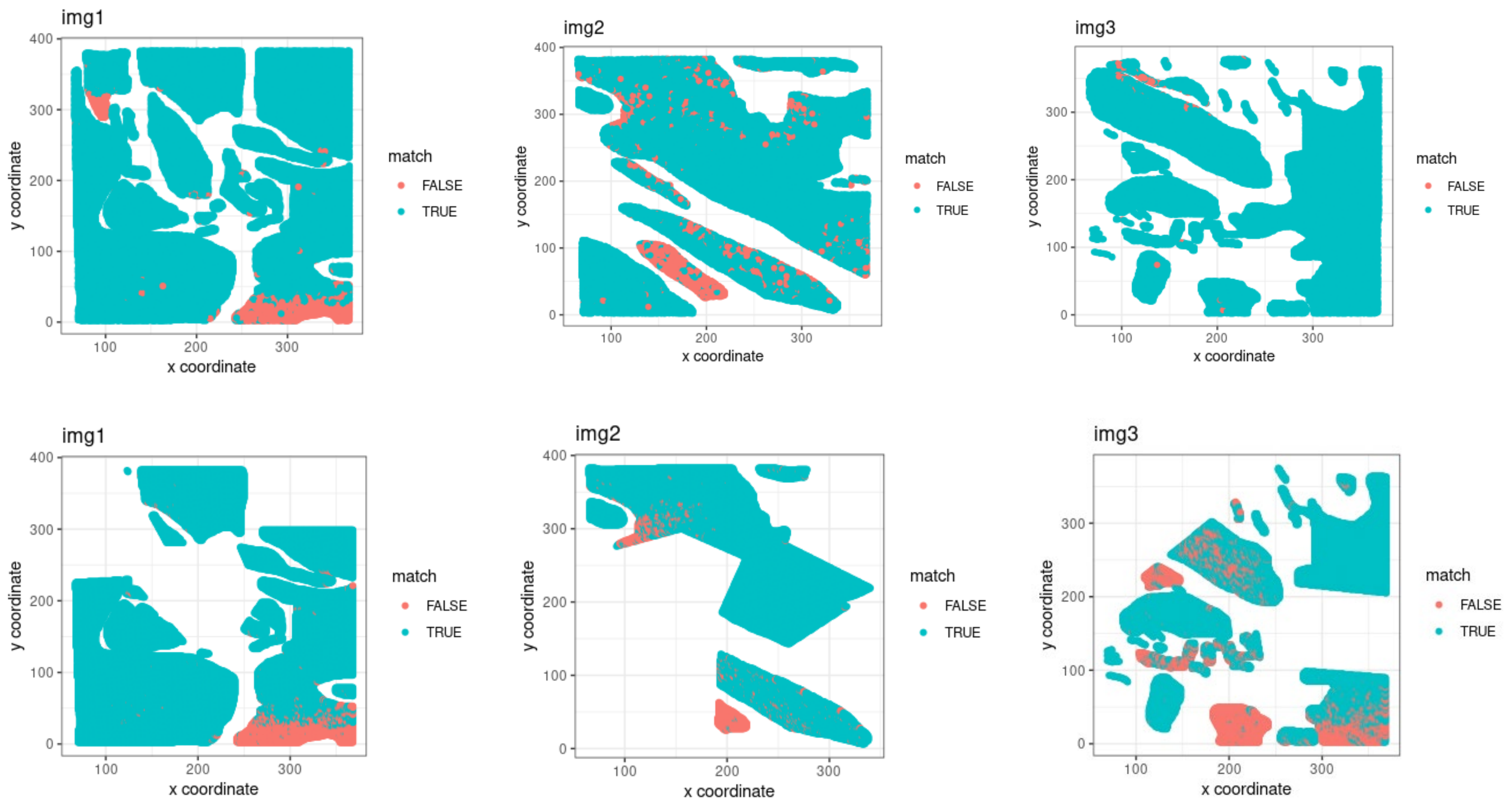

Figure 7: *Misclassification patterns in three images. Correctly-labeled points are in blue and misclassified points are in red. (Up) Misclassification patterns for Type-1 splitting. (Bottom) Misclassification patterns for Type-2 splitting.*

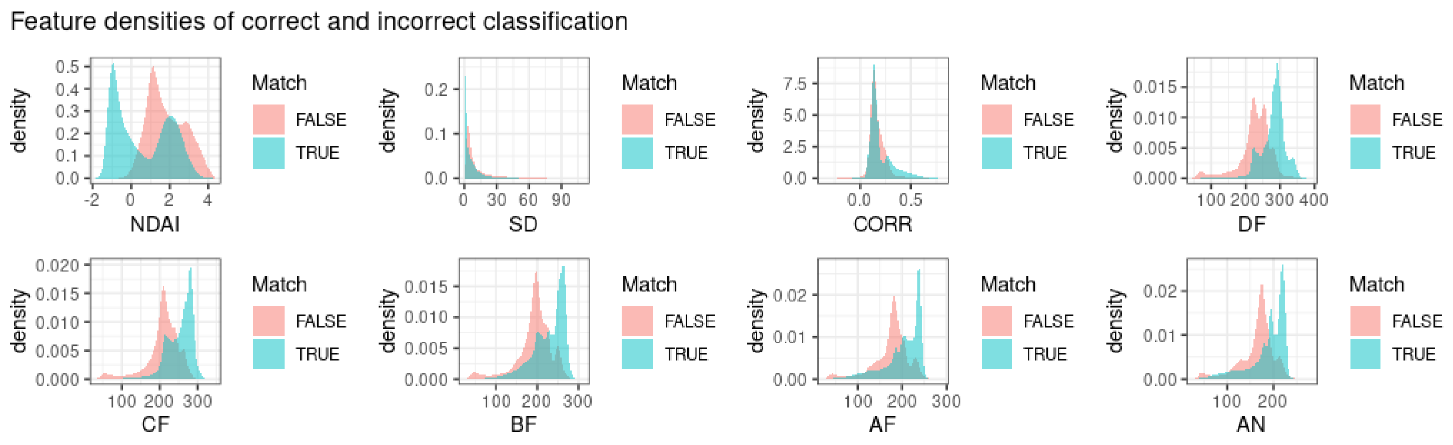

Figure 8: *Feature densities of correct and incorrect classification of Type-2 splitting.*

(c) A better classifier

Under the assumptions that expert labels are highly reliable, in this section, we decide to use the popular and powerful XGBoost as our new classifier. This is due to the desire to further decrease misclassified pixels, as well as the fact that boosting works in a way of emphasizing misclassified points in each iteration. We run cross validation across different numbers of rounds and different cutoff probability thresholds (see Figure 9). The feature importance ranking basically agrees with our previous observation and discussion about Figure 8; in other words, features with top importance also have discrepant densities between correctly-labeled pixels and misclassified pixels. For the misclassification pattern of image 2 after we run XGBoost, the misclassification issue in the middle-bottom island region is greatly alleviated. We propose that XGBoost is an outstanding classifier for this problem, and will work on future data without expert labels.

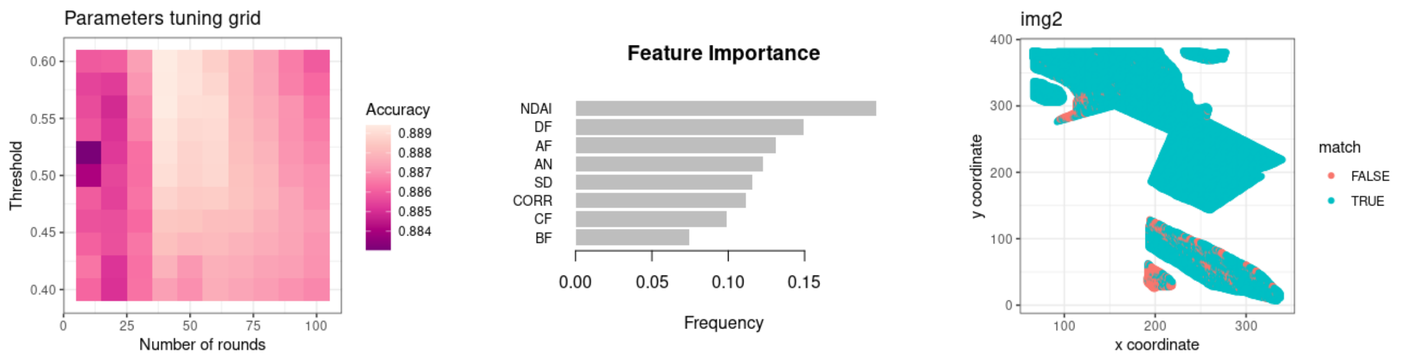

Figure 9: *Plots regarding XGboost. (Left) tuning parameter results shown in grid, where x axis is the number of rounds, and y axis is the cutoff threshold of predictive probability. The highest accuracy occurs when #(rounds) = 40 and threshold = 0.56. (Middle) Feature importance reported by XGboost using frequency as metric. (Right) The misclassification pattern of image 2.*

(d) Discrepancy caused by splitting types

In Figure 6 in section 4(a), we can clearly spot the difference between lines representing Type-1 and Type-2 splitting. The difference between misclassification patterns of these two types of splitting (see Figure 7) are also apparent, especially for image 3 where Type-1 has a much less misclassification proportion. In a nutshell, the results significantly change as we modify the way of splitting the data.

(e) Conclusion

Based on our extensive and comprehensive analysis, we find LDA and XGBoost provide us the most powerful tools to classify cloud detection. In addition, Type-2 splitting in general yields more robust and slightly better accuracy results than Type-1 splitting, and Type-2 splitting is closer to the way of training and testing in the real world as well. In this project, we understand the importance of models' assumptions and limitations of different algorithms under certain scenarios, and we foresee that other robust classification models like random forest will also generate promising performance. However, in this case study, we only focus on the expert labeled data, where cloud and non-clouds are far away from each other. Our conclusion might become quite different if we are required to predict the unlabeled region close to the edges and boundaries. We hope that more expert meteorology knowledge and better measurements will contribute to the research in this challenging case.

Acknowledgment

This project is equally contributed by Xiaozhu Zhang (building models and writing codes) and Xuyang Tian (building models and writing papers). We acknowledge the references of Yu (2008) which provides a clear overview and inspiration of this whole project. The coding part of this project is based on the R environment and commonly used R packages, and all codes with descriptions are submitted to Gradescope for the sake of reproducibility.