Project 2: Cloud Detection

Tianhao Li(tl301@duke.edu), Haoming Yang(hy190@duke.edu)

January 13, 2023

1 Data Collection and Exploration

1.1 Paper Summary

Evidently from general environment study, Global climate and global warming are highly dependent on the carbon dioxide levels in the Arctic. The purpose of the paper by Tao et.al. [1] is to develop operational cloud detection algorithms, which adaptively combine classification and clustering methods, to ascertain the properties and identify the presence of clouds in the Arctic in satellite photos. The ultimate aim is to understand the flow of radiation through the atmosphere which enables us to identify the relationship between clouds and changes in the arctic climate.

The data comes from Multiangle Imaging SpectroRadiometer (MISR) on the Terra satellite, which comprises 9 cameras that view the Earth from different angles in four spectral bands. The data used in this paper comes from 10 MISR orbits of path no.26, a path with rich surface features, over the Arctic, northern Greenland, and the Baffin Bay. Six data units from each orbit are included in the study, and 3 of all 60 data units are excluded since the surfaces are open water which does not need a new algorithm. So the data contains 7114248 1.1-km resolution pixels with 36 radiation measurements for each pixel, while the authors also construct some other features.

The main conclusion of Tao et.al. [1] is that by using three constructed features – CORR, SD, and NDAI, the ELCM algorithm provides better accuracy and better spatial coverage than existing MISR algorithms for cloud detection in the Arctic, and the results can be used to train QDA to provide probability labels for partly cloudy scenes.

There are two main impact of the previously mentioned work. First, the paper shows that statistics can go beyond the implementation of statistical methods and be directly involved in the core of data processing. Second, it demonstrates the power of statistical thinking and the fact that it can create effective and innovative solutions to complex scientific problems.

1.2 Data Summary

	Img1 proportion	**Img2 proportion**	**Img3 proportion**	**Summary**	**Summary proportion**
Not Cloud	0.3411	0.1776	0.1843	80981	0.2343
Unlabeled	0.3725	0.4377	0.2929	127080	0.3677
Cloud	0.2863	0.3845	0.5226	137495	0.3978

Table 1: Percentages of Pixels for Different Classes

Here we draw a table of percentages of different classes in 3 images and in total. To draw a well-labeled beautiful plot (see Figure 1), we first replace the numeric labels in the original dataset with text labels, then fill the plot with colors corresponding to each label. We can see from the plots that pixels labeled as "Cloud" or "Not Cloud" group together to form clusters, and pixels labeled as "unlabeled" fill in the cracks between these two groups. This observation suggests that there exists some spatial dependence, so the i.i.d. assumption for the samples does not hold for this particular type of data.

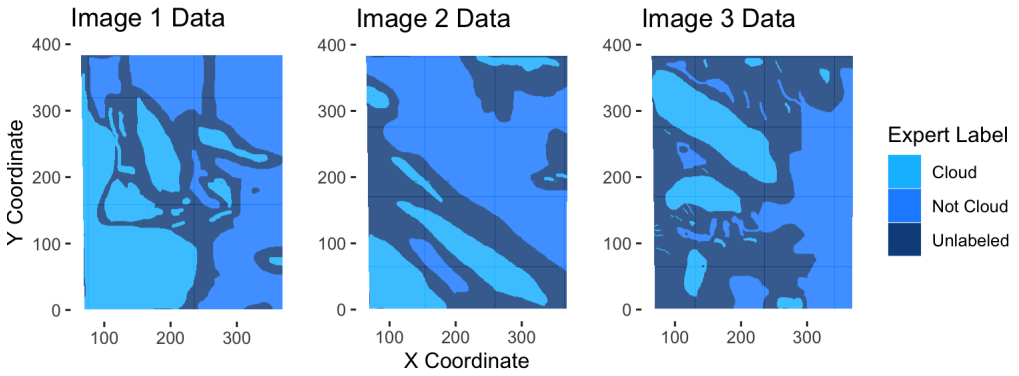


Figure 1: Plots of Three Images by Expert Labels

1.3 Exploratory Data Analysis

To inspect the relationship between the features themselves from a quantitative perspective, we calculate the correlation matrix for these features. Then from a visualization perspective, we draw a pairwise scatter plot using *ggpairs( )*. Note that to reduce the computational complexity, we randomly select 10000 data to draw the scatter plot. From the correlation matrix and pairwise scatter plot(see Figure 2), we see a high correlation in AN-AF, AN-BF, AF-BF, BF-CF — they all have correlation coefficients larger than 0.9.

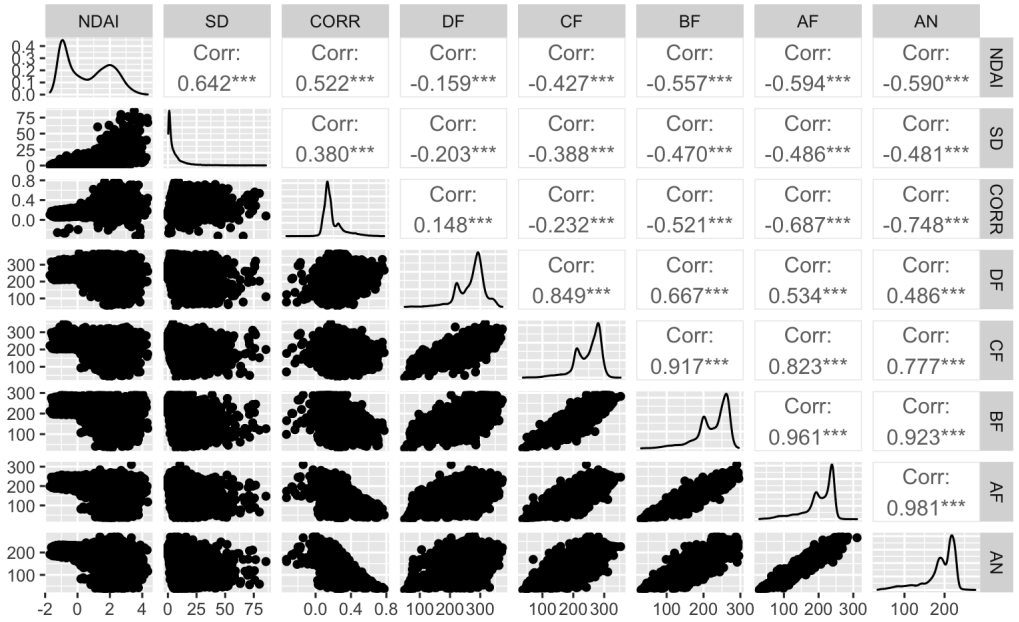


Figure 2: Pairplot Between Features (Images Combined then Sampled)

Then to inspect the relationship between the expert labels with the individual features, we decide to use density plots to best visualize the distribution of different features on 2 labeled classes. We used *summary( )* to inspect different features on 2 classes, but for better visual effect the density plot is a more evident way to show the differences between the 2 classes. Since the distribution of SD is highly right-skewed, we do a log transformation to make the difference between classes more obvious (the added 1 is to deal with 0s in our data). We see that there are clear differences in the constructed features NDAI, SD, and CORR between "Cloud" and "Not Cloud" pixels. Pixels with the "Cloud" label have higher NDAI, SD, and CORR values. As for radiance features, apart from DF, "Not Cloud" pixels generally have higher values on other features.

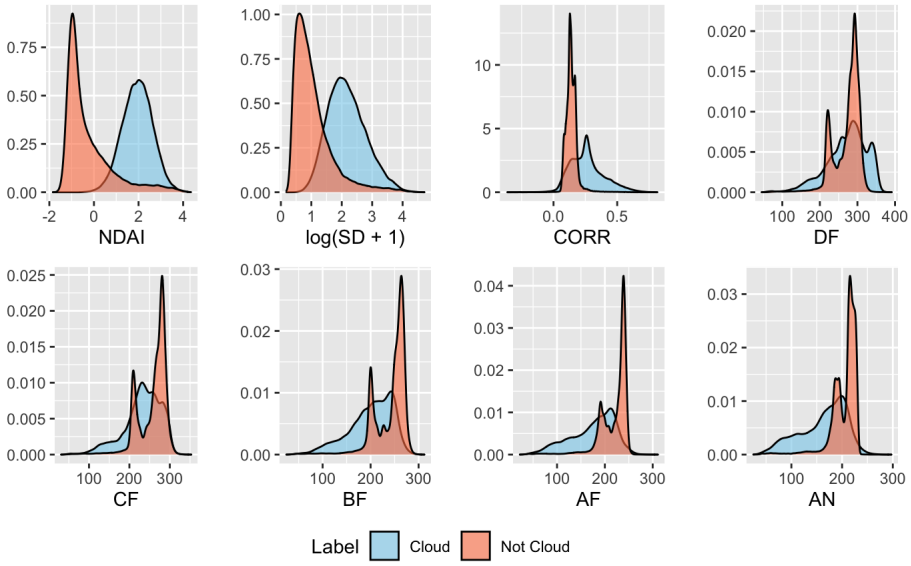
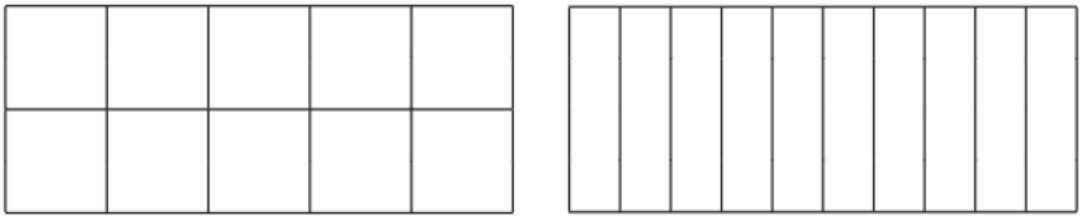


Figure 3: Density Plots of all Features by Different Labels

2 Preparation

2.1 Data split

In the previous section, we concluded that the data have spatial dependencies: if a pixel is labeled as "Cloud", it is highly likely for the surrounding pixel to also be labeled as "Cloud" and vice versa. Therefore, the traditional data splitting method that treat data as i.i.d is inappropriate. To prepare for our later training and tuning of statistical learning models, we have developed block-wise and vertical-wise data splitting methods that take spacial dependencies into account.



Left:Block-wise split, **Right**:Vertical-wise split

Figure 4: Demonstration of Image Split

For the block-wise method, we allow intake of images as a list, the number split on x and y axis. Then, given a image, we will find the split point on x and y axis, and split the data into blocks. For the vertical-wise method, the idea is similar, but this time we only need to find splitting points on the x axis. After finding the splitting point, we will split the data according to their axis information. To improved the usability of the data splitting functions, both splitting function allow users to specify the train-test split proportions, and validation proportions if needed. Since our images contains unlabeled data, we added another control for deleting unlabeled data via splitting. The output of the two functions are similar: a list that contains sublists called train, validation, and test, each contains the corresponding block or vertical strips of data.

For the next two subsections of section 2, we will split the data into train, validation, and test dataset to conduct some baseline exploratory modeling on our image dataset.

2.2 Baseline

We fitted a baseline classification algorithm that classifies all data into "Not Cloud". This was done on both validation and test dataset to ensure that the current classification problem is not trivial. The baseline classification method generated the classification accuracy showed in table 2. We observe that while some accuracy for both splitting methods are high, the validation accuracy of vertical split remained below random guessing. The accuracy of the baseline model also shed light on the spacial dependencies of our dataset: the pixels around each other are more likely to be split into the same set, and they might all contain the same label. The baseline model will return a high accuracy when the label proportion is unbalanced: if there is a high amount of negative data, the accuracy will be high.

Dataset	**Accuracy (%)**
Block Split Validation Accuracy	83.08
Block Split Test Accuracy	62.13
Vertical Split Validation Accuracy	38.34
Vertical Split Test Accuracy	67.75

Table 2: Accuracy of Baseline Classification on Validation and Test Set

2.3 First order Feature Importance

We also explored the first order feature importance of the dataset. Before we fit any model, we first made an observation on the distribution of the features. In general, if we observed obvious distribution difference of some feature between the positive and negative labels, we can easily separate the labels base on these features. From boxplots shown in figure 3, we concluded that there are limited difference between the distribution of features DF, CF, and BF. Also, we can see from figure 2, there is high-correlation between DF, CF, BF and AF, AN features; therefore the features mentioned above might not be highly important during modeling, as their contribution to the model can be well covered by AF and AN. Also note that from the density plot (see Figure 3), the two classes exhibited large difference in their Log(SD + 1) distribution ; therefore, we have tested this log transformed feature in the following first order importance experiment.

After we ruled out DF, CF, and BF, we fitted a one feature logistic regression for each remaining feature. Each single feature logistic regression model is fitted on the entire training set's feature, and tested on validation set's label. The prediction accuracy is presented in table 3, we can observe that both block and vertical splitting methods have shown similar result. In the original features, NADI, SD, and CORR showed significant high predictive power when compared to AF and AN. Also note that the log-transformed SD does not evidently out-perform the original SD feature, therefore we will remain in using SD, in the later sections.

To conclude, we have chosen NDAI, SD, and CORR to be our best first feature because their single feature classifier have produced the best validation accuracies.

Split Method	**NDAI**	**SD**	**Log(SD + 1)**	**CORR**	**AF**	**AN**
Block Split	85.38	81.54	81.62	84.21	42.67	42.04
Vertical Split	86.34	81.19	81.58	84.71	69.23	79.15

Table 3: Accuracy For Single Feature Logistic Regression Classifiers in percentage

2.4 Generic CV function

To train with our special splitting algorithms, we implemented a Cross Validation function called *cvMaster( )*. The function can in take a list of image and a splitting method, or pre-split train set as training data. It can also in take a specific classifier to train and a classifier-specific tuning parameter. This generic cross validation will rotate the blocks in the training split: first, we will calculate the number of blocks k_n needed for each fold, then we will shuffle the blocks, and take the first k_n blocks as validation set for the first iteration, then take the next k_n blocks as the validation set for the

second iteration and so on. We also allow inputs of different metrics, and the training will be done on optimizing the designated metrics by user's choice. Within each iteration, we will loop through the tuning parameters (if there is any), and use the pROC package to calculate the best threshold for that particular tuning parameter. The output of *cvMaster( )* is a list containing two sublists: the tuning matrix that contains the validation accuracy of each tuning parameter at each fold, and the threshold matrix that contains the best threshold of each tuning parameter at each fold.

3 Modeling

3.1 Classification Methods Tryouts

In this section, we apply the previously mentioned ways of splitting the data and then try 5 classification methods and assess their fit using our *CVmaster* function. Although we have selected 3 "best" features, for best prediction ability we still use all the features. For later convenience of merging the misclassified points back to the original image, we add an indicator showing which image the pixel belongs to when splitting the data. Before carrying out further training and testing, we also set the seed to be 520 which returns evenly distributed training and testing sets.

Here we choose 5 basic classification methods, namely LDA, QDA, Logistic Regression, Decision Tree, and Random Forest. The reasons for us to choose the first 4 methods are their simplicity; they are easy to train, easy to penalize, and easy to tune their parameters given the large amount of pixels and our limited computing resources. Since Random Forest is one of the best off-the-shelf methods, we also take it into consideration.

The assumptions of the applied methods are as follows:

- LDA: Models the data within each class as normal distributed with common covariance matrix.
- QDA: Models the data within each class as normal distributed with different covariance matrices.
- LR: Response variable is binary, e.g. comes from a Bernoulli distribution. Observations are independent and there's no strong multicollinearity between explanatory variables. A logit function links the response variable and explanatory variables. The sample size is sufficiently large.
- Decision Tree: A non-parametric algorithm, no specific strong assumptions.
- Random Forest: A non-parametric algorithm, no specific strong assumptions.

From the density plots(see Figure 3), we know the data within "Cloud" or "Not Cloud" classes does not follow normal distributions, the assumptions of LDA and QDA are invalid. As for Logistic regression, the assumption concerning multicollinearity does not hold because we have seen some strong multicollinearity between explanatory variables. Although some of the assumptions of the methods are not met, we can still use these methods here for predictive purpose.

When training the model, we used a 5-fold CV to see the generalization ability and stability(for LDA and QDA) and to tune the hyper parameters in regularized Logistic Regression and Decision Tree with complexity parameter. As for the tuning parameters, we choose c(0,0.0005,seq(0.001,0.01,0.001)) for the λ [2] in regularized Logistic Regression. As for complexity parameter of Decision Tree, we try a wide range of parameters of c(0,seq(0.001,0.03,0.001)). These range are decided by a series of ad-hoc testing that lead us to believe the best parameter is in the middle of the selected range. For the *mtry* in Random Forest, which stands for the number of randomly selected predictors of each split, we selected integer sequence 1 through 8 since the dataset only contains 8 features. We select the hyper parameters that brought the highest across-folds accuracies. The selected hyper parameters are shown in Table 4.

The results of Logistic Regression, Decision Tree and Random Forest we use here are the results correspond to the best hyper parameter.Then we present the test set accuracies at the last column. One thing to mention here is that when doing prediction on the test set, we need a threshold to determine the prediction result. We obtain the threshold of 5 classification methods via the ROC plots of 5 methods in the next part, which are drawn based on the whole training set (see next section).

From the results table we see that, as for the Block-wise splitting method, these methods yield close accuracies across folds and on the test set. In the Block-wise splitting setting, among these

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Average	Test Set
QDA Block	0.9879	0.7884	0.7232	0.8219	0.9321	0.8507	**0.9025**
LDA Block	0.8822	0.9455	0.5689	0.8030	0.9618	0.8323	0.8863
LR Block (λ = **0.007**)	0.9768	0.8520	0.6702	0.8245	0.8957	0.8438	0.8880
DT Block (**cp = 0.018**)	0.9775	0.9647	0.8933	0.8200	0.8594	**0.9030**	0.8628
RF Block (**mtry = 1**)	0.9712	0.9634	0.8590	0.8868	0.8057	0.8972	0.8690
QDA Vert	0.9637	0.8153	0.8566	0.7598	0.9324	0.8656	0.8113
LDA Vert	0.9668	0.7651	0.8867	0.7045	0.7920	0.8230	0.7890
LR Vert (λ = **0.0005**)	0.9433	0.7825	0.8443	0.7743	0.9184	0.8526	0.7814
DT Vert (**cp= 0.02**)	0.9434	0.8997	0.8790	0.8645	0.9197	0.9013	0.8826
RF Vert (**mtry= 1**)	0.9762	0.8768	0.9852	0.9041	0.9716	**0.9428**	**0.9323**

Table 4: Accuracies across Folds

methods Decision Tree obtains the highest across-folds accuracy and QDA obtains the highest test set accuracy. As for the Vertical-wise splitting setting, Random Forest outperforms all the other methods and performs extremely well on the Test set. If we look at the performance in general, we could find out that Random Forest performs generally well.

3.2 ROC curves

	DT	LDA	LR	QDA	RF
Block	0.5642	0.6693	0.6049	0.9567	0.5010
Vert	0.5734	0.7863	0.7152	0.9791	0.5790

Table 5: Cutoff(threshold) Values

In the ROC plots(see Figure 5) the LDA curve and Logistic Regression curve overlapped so they looks like one single line, and some of the cutoff points also overlaps. We then display the exact value we obtain from the ROC curve in Table 5. The AUC values are shown in section 3.3 in Table 6 of other alternative metrics. As we can see, Random Forest has a high AUC value in both the block-wise splitting setting and the vertical-wise splitting setting, which suggests that it is the best model here in terms of the overall ability to distinguish between the positive and negative classes.

To choose the cutoff values, we select the point on the ROC curves which are the closest to the top-left corner because it has the largest true positive rate and lowest false positive rate. We obtain the information via the *coords* function in the pROC package.

3.3 Alternative Metrics

As for other alternative metrics, we selected precision, recall, F1 and train AUC(See Table 6). Precision talks about how accurate our model is out of those predicted positives.Recall calculates how many of the Actual Positives our model captures through labeling it as Positive. If we want to seek a balance between Precision and Recall, then F1 is another important metric. From F1 perspective, QDA, LDA and Logistic Regression perform better in the block-wise splitting setting and the other 2 tree-based algorithm perform better in the vertical-wise splitting setting. Random Forest has the best overall performance across 2 splitting settings among all the classification methods.

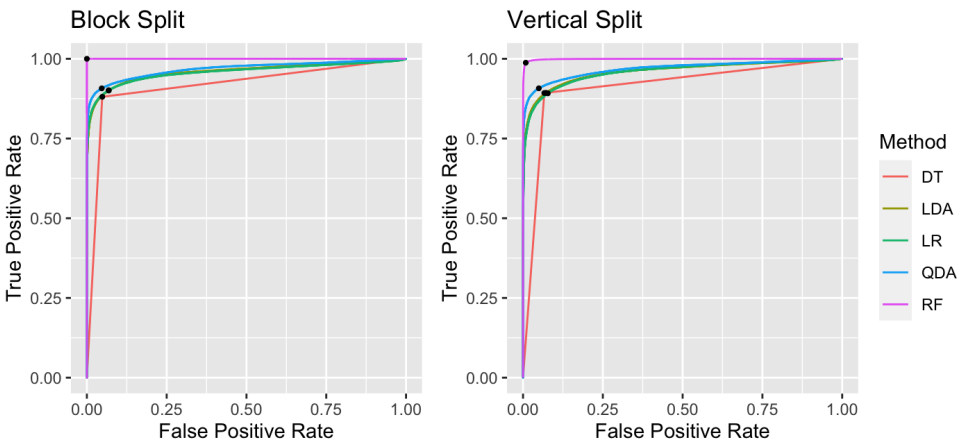


Figure 5: ROC curves

Classifier	Precision	Recall	F1	Train AUC
QDA Block	0.8251	0.9380	0.8779	0.967
QDA Vertical	0.9365	0.6218	0.7473	0.963
LDA Block	0.7793	0.9495	0.8560	0.959
LDA Vertical	0.8600	0.7138	0.7801	0.959
LR Block	0.7848	0.9576	0.8626	0.957
LR Vertical	0.8615	0.6694	0.7534	0.959
DT Block	0.7323	0.9799	0.8382	0.971
DT Vertical	0.8560	0.8922	0.8738	0.952
RF Block	0.7540	0.9730	0.8496	1.000
RF Vertical	0.9107	0.9425	0.9263	1.000

Table 6: Alternative Metrics including Precision, Recall, F1 score, and AUC of the ROC curves

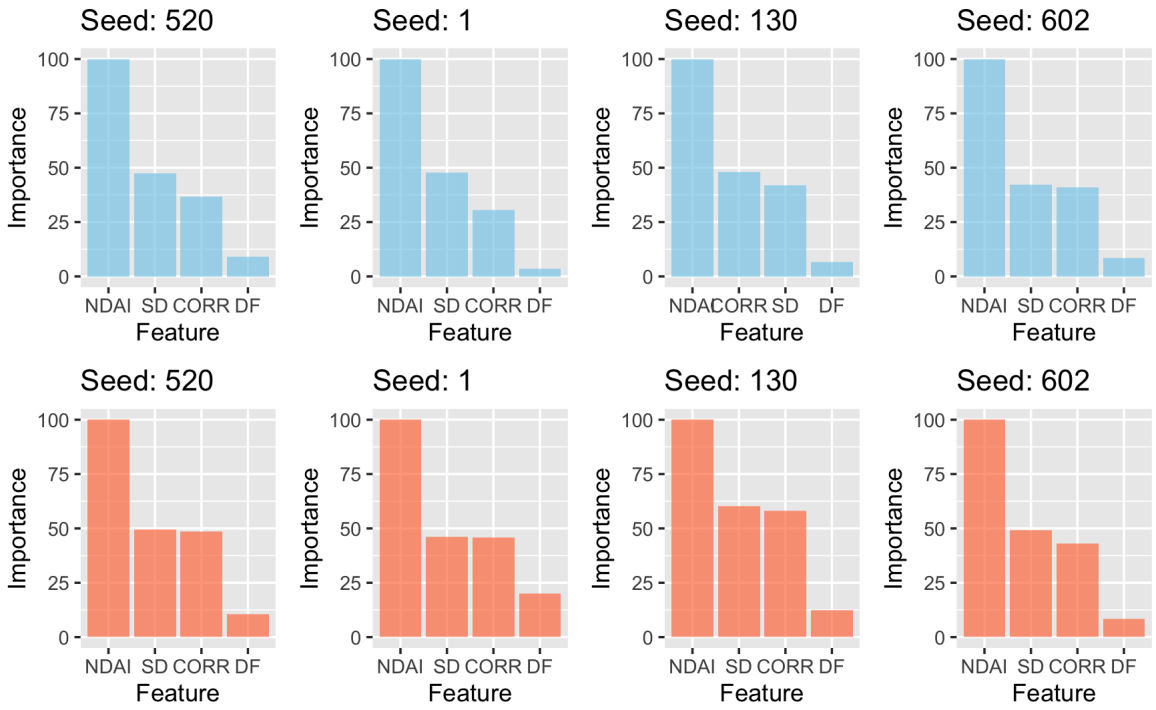
4 Diagnostic and Conclusion

4.1 Model Convergence, Stability, and Robustness

From the previous cross validation result and test accuracy as well as the alternative metrics, we have concluded that Random Forest is the best model. Although there are parametric methods that does comparably well, the normality assumption and the independent assumption had made the parametric methods less appealing.

As Random-Forest is a non-parametric method, we cannot directly assess the parameter convergence of the model. However, splitting the data at different seed and examine the variable importance of Random Forest trained with different data can offer us insights on our Random-Forest model's predictive process.

We re-split the data 8 times, 4 block-wise, and 4-vertical wise split with different seed, and fitted the Random-Forest model with the same parameter for all 8 different split. The top 4 most important feature of the eight models by different splitting method and seed is shown in Figure 6. We can see that the general ranking of feature importance is the same across seed and different splitting method. This suggest that the difference in split does not truly affect model's predictive power. All feature importance essentially converged into similar distribution. Note that for seed 130's block split model, the ranking of SD and CORR is different from all other models, but this small difference is not enough evidence to suggest against our model convergence.



Skyblue:Block-Split **Coral**:Vertical-Split

Figure 6: Feature Importance of Random-Forest Model in both Splitting Method.

We also decided to assess these 8 different models' predicting result to see if their predictions converge as well. In Table 7, we showed the confusion matrix of the models, both block and vertical splitting methods. We can see that for block splitting method, seed 520, 130, and 602 showed similar result: all 3 had exhibited high False Negative rate; however seed 1 has led to a low False Negative rate, which led us to believe that the block splitting method is not as stable.

The 4 models of the vertical split method resulted in stable confusion matrix. Across the 4 different seeding, the element in the confusion matrix only fluctuated a little.

To explore the robustness of our model, we will attempt to answer the following question: how will our model perform when the training data is wrong? To perform this analysis, we perturbed the training label and analyzed the test result of the models trained by these "wrong/fake" data. We also fitted a baseline model, logistic regression, through a similar process to compare the robustness of our Random-Forest model. From Figure 7, we can see that as the perturbed proportion increase, the test accuracies of the models, for both data splitting methods, has decreased. However, by comparing the model, we can see a clear difference between Logistic Regression and Random Forest. Logistic regression trained with block split data will decrease drastically in accuracy when we move from 0.3 to 0.45 perturb proportion, while block split Random Forest's accuracy decreased slower. The difference between vertical split Random Forest and Logistic Regression is even more obvious. The blue dashed line of Logistic Regression started to descend from the first perturbation, while the blue solid line of Random Forest stayed level till 0.3 perturbed proportion. From this analysis of perturbed data and test accuracy, we have concluded that our Random Forest model is robust, as it performs well even with 30% of "wrong" data.

Generalizing from the discussion above, we believe that our model has converged and is stable. Although Random-Forest's performance under block-split data is not ideal, it can reproduce similar variable importance and predictive power most of the time. Random Forest model is also robust when compared to other method such as logistic regression, as it test accuracy stayed leveled when proportion of perturbed training data increased to up to 30% while logistic regression failed to do so.

A interesting finding is the relationship between average cross validation error and value of parameter *mtry* in Random Forest(see Figure 7). *mtry* represents the number of variables available for splitting at each tree node, the larger *mtry* is the more similar the trees in the random forest model

	T Cloud	T Not Cloud
P Cloud	80437	544
P Not Cloud	5561	121519

(a) Block Confusion Matrix, Seed = 520

	T Cloud	T Not Cloud
P Cloud	80018	963
P Not Cloud	1511	125569

(b) Vert Confusion, Seed = 520

	T Cloud	T Not Cloud
P Cloud	79436	1545
P Not Cloud	904	126176

(c) Block Confusion, Seed = 1

	T Cloud	T Not Cloud
P Cloud	80057	924
P Not Cloud	1143	125937

(d) Vert Confusion, Seed = 1

	T Cloud	T Not Cloud
P Cloud	79754	1227
P Not Cloud	6406	120674

(e) Block Confusion, Seed = 130

	T Cloud	T Not Cloud
P Cloud	80524	457
P Not Cloud	993	126087

(f) Vert Confusion, Seed = 130

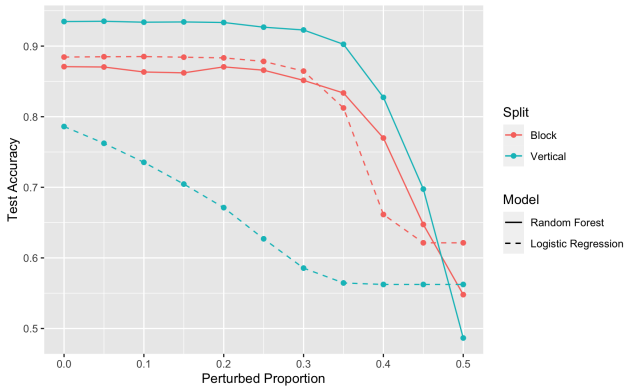
	T Cloud	T Not Cloud
P Cloud	80288	693
P Not Cloud	5843	121237

(g) Block Confusion, Seed = 602

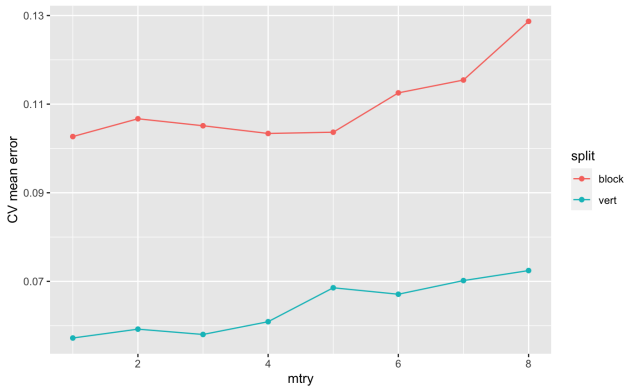
	T Cloud	T Not Cloud
P Cloud	80737	244
P Not Cloud	1824	125256

(h) Vert Confusion, Seed = 602

Table 7: Confusion Matrix of Different Starting seed



(a) Test Accuracy vs Proportion of Perturbed Training Label



(b) CV mean error vs mtry

Figure 7: Random Forest model stability

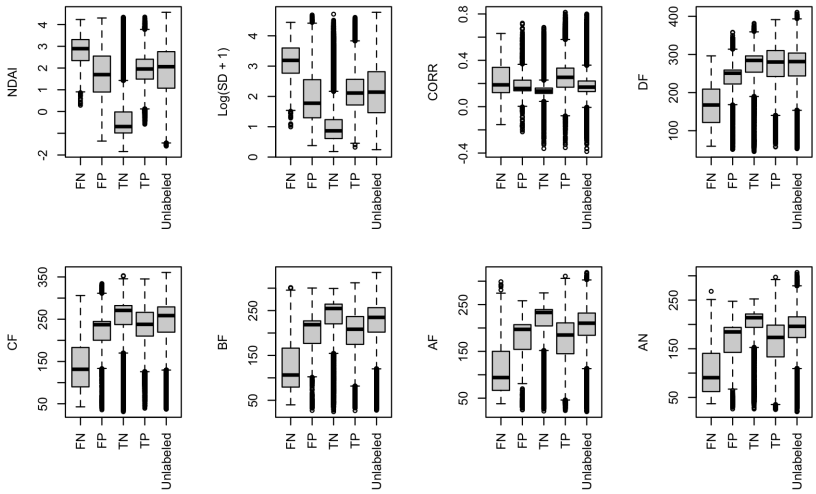
are. We see that although when *mtry* equals to 1 the error reaches its lowest value, the error is also low when *mtry* is around 3, which is closest to the typically chosen value $\sqrt{p}$. Then as the trees become more similar, the Random Forest is less capable of making robust decision. With *mtry* equals to 1, it means that the tress in our random forest are built completely at random.

4.2 Miss-classification Patterns

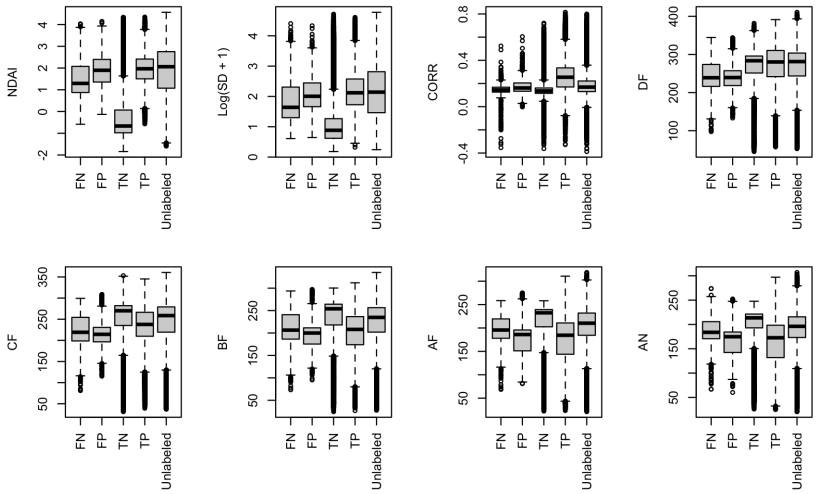
We are also interested in the patterns of the miss-classified data. Analyzing the classification error could give us insight on the distribution of the incorrectly classified data, and potential leads to how we might be able to improve the classification error.

We plotted the distribution of each feature of the correctly classified, miss-classified, and unlabeled dataset for both splitting method's best model in the Figure 8. From the boxplot of both block-wise and vertical-wise split, we can see that the False Positive (FP), True Positive(TP) data's feature NDAI and Log(SD+1) are quite similar, which could have been the reason for the miss-classification of the FP points. However, the distributions of False Negative (FN) and True Negative (TN) are quite different, and it is different across splitting method as well. We can easily observe that the distribution of FN points' NDAI and Log(SD+1) is in a different range comparing to TN points for either splitting method. From Figure 8 (a), the FN point's of the block split method result exhibited

below average Radiance across all Radiance related features. It is possible that these data are not well represented during the training set, which led to the result of miss-classification of these data. From the boxplot of the Figure 8 (b), the FN and TN data's radiance related features remained in similar range. However, from DF and CF features, we can see that the mean and the general distribution of incorrectly classified points (both FN and FP) are lower than the correctly classified points. This findings have led us to our suggestions of using transformations in the later section.



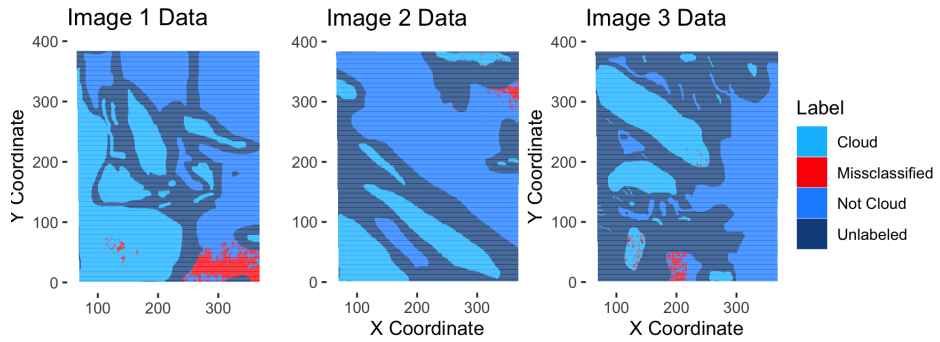
(a) Block-Wise Split Method, Random Forest Classification Algorithm



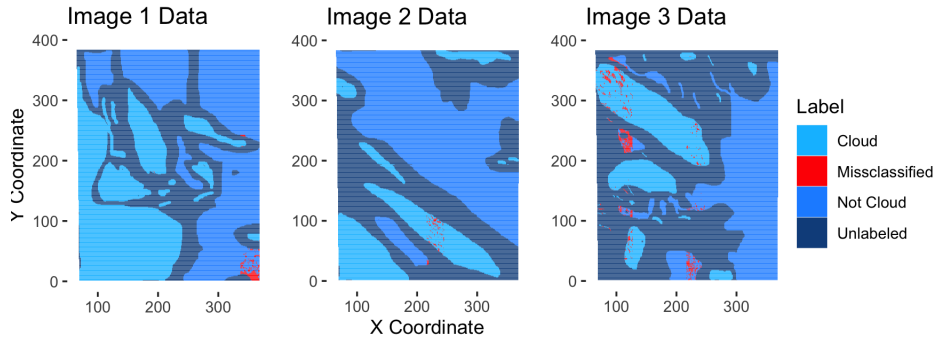
(b) Vertical-Wise Split Method, Random Forest Classification Algorithm

Figure 8: Boxplots of False Negative, False Positive, True Negative,True Positive, and Unlabeled Data

We can also highlight the miss-classified points on the original image to find the patterns in positions of the miss-classified points. As presented in Figure 9, we plotted the miss-classified data on top of the original image, and we can see that most of the miss-classified data reside close to each other. The miss-classified patterns for the two different splitting method has some similarity: for image 1, there are chunks of miss-classified negative points ("Not Cloud") on the bottom right corner; for image 2, there are little miss-classification; in image 3, both block-wise and vertical-wise split Random-Forest miss classified parts of the non-Cloud island at the bottom, but the model trained on vertical-split also miss-classified small parts of the "Cloud" on the top left corner.



(a) Block-Wise Split Method, Random Forest Classification Algorithm



(b) Vertical-Wise Split Method, Random Forest Classification Algorithm

Figure 9: Images of Correctly Classified, Miss-Classified, and Unlabeled Data

4.3 Future Improvement

Due to the dependence nature of the dataset and the non-normal distribution of features between "Cloud" and "Not Cloud", we believe that it is only reasonable to fit the data using non-parametric methods such as decision tree, Random Forest, KNN and so on. Our final model on vertical splitting methods has achieved an optimistic classification result with more than 93% test accuracy, and while the test accuracy for block splitting methods is only 86% it is still a resonabale performance. The reason for under-performance of Random-Forest with block-wise split data is unknown, but given the strong performance of Random Forest on vertical splitting methods and a reasonable performance on block splitting methods, we believe our choice of classification can perform well without expert label.

From the boxplot of the features plotted in Figure 8, a few features has attracted our attention. The FN data in NDAI and Log(SD+1) features has a generally different distribution than the TN, which means that it is possible to use this distribution difference to help the classifier. In order to achieve such detection, some transformation on the feature might be needed. From the similar patterns of the miss-classified points on the 3 images, we can see that there are some data that is prone to miss-classification. One can potentially apply boosting and re-weight the miss-classified data heavier to achieve a better performance for those miss-classification prone data. Also note that the Radiance related features, especially DF and CF showed different mode of distribution for the falsely classified data. This discovery have led us to another possible improvement: kernel method. Kernel method is essentially a data transformation technique, but it allows more flexible and power transformations done directly on the inner product. By using kernel, it is possible to map our data to a higher dimension space, and transform the originally non-separable data to be separable. Although the distribution difference are small for the Radiance features shown in the boxplot, a good kernel that can well separate the difference can improve classification performance. After we kernalized the data, we can apply some simple logistic regression classification or lda and qda again to test the performance.

4.4 Split Stability

From the distribution of the miss-classified points, we can see that there exists a difference between data-splitting method. This is expected because of the spatial dependence of the data. Different splitting method might lead to significantly different feature distributions in train and test set. We can also infer the split different from our model performance: in general, the vertical split Random Forest performs better than the block-wise split Random Forest.

4.5 Conclusion

By developing two different image splitting methods and 5 different classification algorithm with rigorous tuning, we have reached a final model–Random-Forest on Vertical Splitting. The Random-Forest on Vertical Splitting had consistently classified with high accuracy (see confusion matrix and tables in previous sections). By diagnostic and error analysis, we have examined the error patterns of our model and have provided some possible improvement ideas. Due to the observed difference distribution of NDAI, and Log(SD+1) in FN and TN points, we believe that some transformation that separate these two features can help the Random-Forest method to reach a even better performance.

5 Acknowledgement

In this project, Haoming did part 2 and part 4, wrote the *datasplit* function and *cvMaster* function, inspected first order importance, checked model convergence, stability, and robustness, found out miss-classification patterns and thought of future improvement; Tianhao did part 1 and part 3 and a small fraction of part 4, summarized the paper and data, did the EDA, trained and tested 5 classification methods, reported ROC curve and other alternative Metrics. The data used in this project comes from the research of Tao et.al [1]. When training the models, we read the Max Kuhn's documentation of *caret* package: https://topepo.github.io/caret/

References

[1] Tao Shi, Bin Yu, Eugene E Clothiaux, 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):584–593, 2008.

[2] Max Kuhn. *caret: Classification and Regression Training*, 2021. R package version 6.0-90.