Project 2 – Taiga Forest: Artic Cloud Prediction using Random Forest

1. Data Collection and Exploration

1.1 Paper Summary

Systematic investigation of the dependences of surface air temperatures on increasing atmospheric carbon dioxide levels requires accurate cloud coverage data because clouds play an important role in modulating the sensitivity of the Arctic to increasing surface air temperatures. However, clouds are hard to distinguish in the Arctic region because they exhibit similar visible and infrared electromagnetic radiation as snow- and ice-covered surfaces. As a result, the existing MISR algorithm (L2TC) does not perform well in these snow-covered regions. In "Daytime Arctic Cloud Detection Based on Multi-Angle Satellite Data With Case Studies", Shi et al. built an operational cloud detection algorithm that can efficiently process the massive MISR data set without human intervention.

The data used in the study were collected from 10 consecutive MISR orbits of path 26 over the Arctic, northern Greenland, and Baffin Bay. As the repeat time between two consecutive orbits over the same path was 16 days, the 10 orbits span approximately 144 days from April 28 through September 19, 2002. Each path is divided into 180 blocks, and each MISR pixel covers a 275m x 275m region on the ground. The MISR data comprises Earth scenes at nine zenith angles in four spectral bands (blue, green, red, and near-infrared): the nine angles are 70.5° (Df), 60.0° (Cf), 45.6° (Bf), and 26.1° (Af) in the forward direction; 0.0° (An) in the nadir direction and 26.1° (Aa), 45.6° (Ba), 60.0° (Ca), and 70.5° (Da) in the aft direction. Three data units were excluded from the study because they cover mostly open water and the existing MISR operational algorithm can detect clouds over water very well. As a result, the data investigated contained 57 data units with 7114248 1.1-km resolution pixels with 36 radiation measurements for each pixel.

In their research, the key idea is to identify cloud-free surface pixels in the imagery instead of cloudy pixels as in the existing MISR operational algorithms. Through analysis, three physical features – the linear correlation of radiation measurements from different MISR view directions (CORR), the standard deviation of MISR nadir red radiation measurements within a small region (SD_{An}), and a normalized difference angular index (NDAI) – are constructed from red radiation measurements. The CORR and SD cutoff values are fixed but the NDAI threshold can be either fixed or updated on new data. The first algorithm, enhanced linear correlation matching (ELCM), classifies pixels as cloudy if $SD_{An} < threshold_{SD}$ or $(CORR > threshold_{CORR}$, and $NDAI < threshold_{NDAI})$. The paper concluded that ELCM is more accurate and provides better spatial coverage than the existing algorithms for cloud detection in the Arctic. Moreover, using Fisher's QDA trained on ELCM labels, the second algorithm, ELCM-QDA, is more informative than the first algorithm as it provides confidence of cloudiness. However, ELCM-QDA does not improve prediction accuracy.

In conclusion, this study shows the important role of statistics in solving modern scientific problems through collaborations between experts from different areas. It also shows how carefully designed features and existing statistical methods can solve difficult scientific problems.

1.2 Data Summary

The MISR data is spatial data that contains a spatial structure with certain patterns. An obvious pattern is that nearby pixels are highly similar as cloudy pixels are contiguous which forms cloud pieces. Moreover, unlabeled pixels are mostly located at the boundaries of clouds and cloud-free areas, as they usually have ambiguous radiance values that make it hard for experts to determine their labels. Therefore, an i.i.d. assumption for the samples is not justified for this dataset.

To visualize the data, maps for image 1 to 3 is shown below as Figure (1) - (3). Cloudy pixels are colored white, cloud-free pixels are colored grey, and unlabeled pixels are colored black.

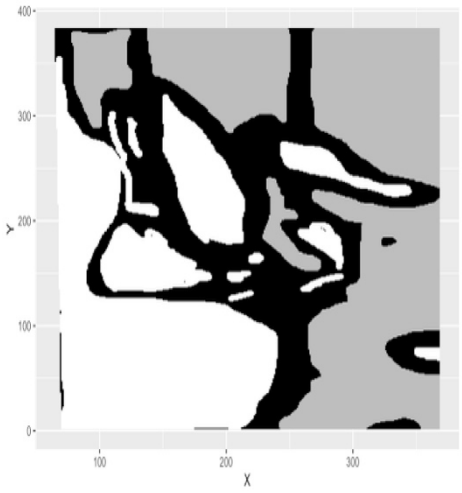

Figure (1): Image 1

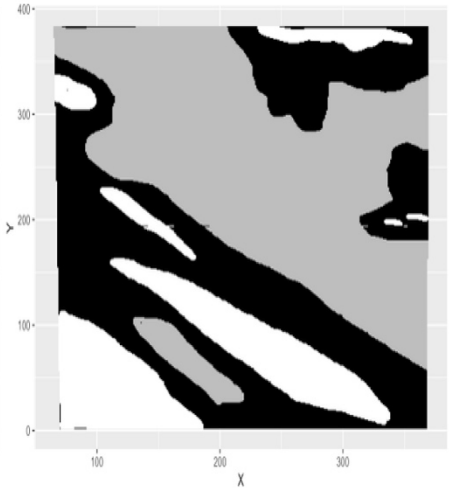

Figure (2): Image 2

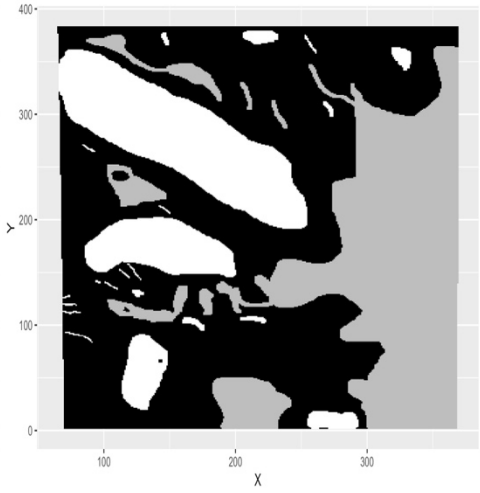

Figure (3): Image 3

The percentage of pixels by classes for each image is shown below in Table 1. As shown, Image 1 has a much more balanced cloudy to cloud-free pixels ratio (34.1-37.3%) than Images 2 (17.8-43.8%) and 3 (18.4-29.3%). In addition, the cloud pieces in image 1 are larger and closer with each other, compared with the more discrete cloud pieces in images 2 & 3, as shown in the maps.

Label	Image1	Image2	Image3
Cloud-free (-1)	37.3	43.8	29.3
Cloudy (1)	34.1	17.8	18.4
Unlabeled (0)	28.6	38.5	52.3

Table (1): Pixels by Label %

	value
NDAI	0.76
CORR	0.55
SD	0.44
Rad_Df	0.01
Rad_Cf	-0.28
Rad_Bf	-0.45
Rad_An	-0.50
Rad_Af	-0.51

Table (2): Predictor correlation with Label

1.3 Exploratory Data Analysis

To examine the pairwise relationship between predictors and the expert label, we first treat expert labels as numeric values (-1 for cloud-free pixels, 1 for cloud pixels, unlabeled pixels are removed), and computed their correlation with the predictors. If a feature is a good predictor of cloud, it should have a high absolute correlation with the numerical label values. The result is shown in Table (2) above. NDAI has the highest absolute correlation with Label, at 0.76, suggesting that it could be very useful for label prediction. CORR, SD, Rad_An & Rad_Af also has a decent absolute correlation with labels, at above or equal to 0.5.

We then visualize this “discriminative power” of predictors by plotting their density with label classes in Figure (5) below. Since predictor SD is skewed and heavy-tailed, we perform a log transform on it. As shown in the figure, cloudy & cloud-free pixels have drastically different NDAI densities. For CORR, most cloudy pixels have >0.2 value, which is quite rare in cloud-free pixels. The density of log(SD) for cloudy and cloud-free pixels have different centers (0 and 2 respectively). Moreover, cloudy pixels have a thick left tail for their Rad_Af values.

We summarized the differences in densities quantitatively through KS (Kolmogorov–Smirnov) statistics, which measures the maximum distance between the CDF of cloudy & cloud-free pixels. The greater the KS statistics, the greater the difference between predictor densities for two classes. As shown in Table (3) below, K-S statistics for NDAI, SD & CORR are very large (0.82, 0.7, 0.63), meaning that their density are very different for the two label classes.

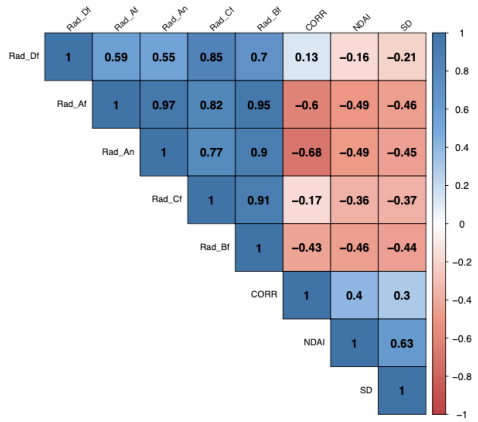

Figure (4): Feature Correlations.

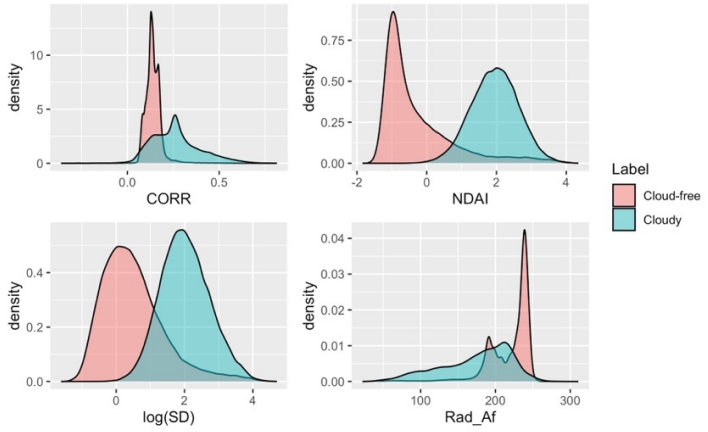

Figure (5): Feature Density by Label

We did not plot other radiance values in Figure (5) as they are mutually correlated, and they have similar density shape for both label classes. Their correlation is shown in Figure (4) above. From the correlation plot, we observe that the author-defined predictors (NDAI, SD, and CORR) are mutually positively correlated, and they are negatively correlated with the radiance values (except CORR with Rad_Df). This suggests that the predictors share mutual information to some extent, and that radiance values share very similar information.

	Statistics
NDAI	0.82
SD	0.70
CORR	0.63
Rad_Af	0.48
Rad_An	0.47
Rad_Bf	0.46
Rad_Cf	0.32
Rad_Df	0.20

Table (3): KS Test Statistics: cloudy vs cloud-free density.

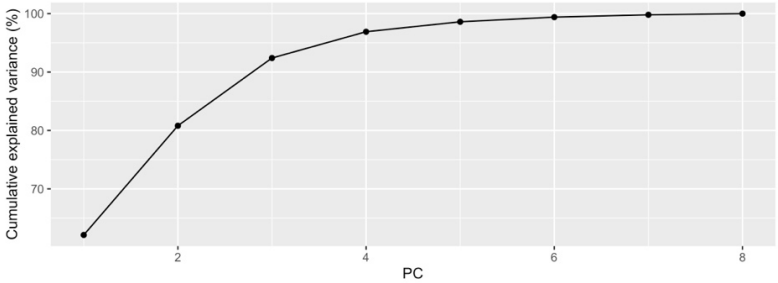

Figure (6): PCA Cumulative explained variance (%)

In addition, the high correlations between radiance values suggest that we can reduce dimension of the predictors through PCA. From Figure (6) above, the first 2 PCs explained 81% of the total variance within predictors. From the principal direction in Table (4) below, the first PC (61% of the total variance) comprises of positive values of radiance values and negative values of author-defined predictors. This confirms our previous hypothesis that the predictors, especially radiance values, share very similar information, and we can easily dimension reduce the predictors. Although the PCs explain within-predictors variation very well, they are not necessarily superior in predicting cloud labels. As shown in Table (5), their correlation with the numeric value of Label is weaker than the original predictors, and some PCs are even uncorrelated with it.

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8
NDAI	-0.27	-0.39	-0.48	0.74	0.05	0.02	-0.01	0.00
SD	-0.26	-0.31	-0.62	-0.67	0.04	-0.02	0.01	0.00
CORR	-0.24	-0.55	0.50	-0.07	0.38	-0.42	0.25	-0.06
Rad_Df	0.30	-0.56	0.14	-0.04	-0.72	-0.07	-0.23	0.02
Rad_Cf	0.39	-0.34	0.05	-0.04	0.27	0.71	0.33	0.16
Rad_Bf	0.43	-0.13	-0.09	0.00	0.45	-0.08	-0.61	-0.45
Rad_Af	0.43	0.02	-0.20	0.03	0.19	-0.45	-0.01	0.73
Rad_An	0.43	0.08	-0.26	0.05	-0.12	-0.31	0.63	-0.48

Table (4): PCA directions of features

	value
PC4	0.47
PC7	0.06
PC3	-0.01
PC6	-0.04
PC8	-0.06
PC5	-0.13
PC2	-0.51
PC1	-0.52

Table (5): Correlation of PCs with Label

2. Preparation

2.1 Data Split

The objective of data splitting is to reserve part of the data to project outside the structure of the training data. The goal is to evaluate model error through the evaluation set and hence approximate model performance on future unseen data. One key assumption of this approach is that the training and evaluation data are independent. If not, models may use predictors to overfit the dependent structure present in both the training and test data, and this leads to unrealistically optimistic error estimates. In addition, violation of this assumption will prompt us to adopt more complex models, as they overfit the dependent structure more easily.

For our spatial data, there is a spatial structure in which nearby very similar: clouds tend to form in pieces. Therefore, we cannot randomly split data into three sets, as nearby pixels might be grouped into both the training and evaluation sets, violating the independence principle. To better split the data, we adopted two methods that split data into blocks. Data within a block is still dependent, but the dependency between training and evaluation blocks is minimized, leading to a more accurate estimate of model error.

The first method is block splitting. It cuts each image into 3 pieces by the x-coordinates, yielding 9 blocks in total. Then, one block is randomly selected as the test set, and one is randomly chosen as the validation set. After splitting, unlabeled pixels are removed as they cannot be trained or tested. The intended train-test-validation ratio is 77.8-11.11-11.11%. Since the number of unlabeled pixels in each block is random, the actual train-test-validation ratio is 78.6-10.4-10.9%. One advantage of this method is that it is straightforward and intuitive: Pixels between the training and evaluation blocks, except those on the boundaries of the two blocks, are only weakly dependent. This is because the spatial dependency decays with distance: the further away two pixels are, the less dependent they are. Therefore, this method reduces the dependency between training and evaluation sets. An illustration of this method is shown in Figure (7) below.

The second method is K-means splitting. The method takes (X, Y) coordinates as input, and runs K-means clustering to yield 3 polygonal blocks for each image. Then, one block is randomly selected as the test set, and one is selected as the validation set. Like the first method, the intended train-test-validation ratio is 77.8-11.11-11.11%, but the actual ratio becomes 73.4-9.3-17.3% after the removal of unlabeled pixels. Compared with the first method, this creates blocks with different dependencies, as boundaries between blocks are different. This method serves as a good comparison tool to evaluate the effect of data split on our modeling. An illustration of this method is shown in Figure (8) below.

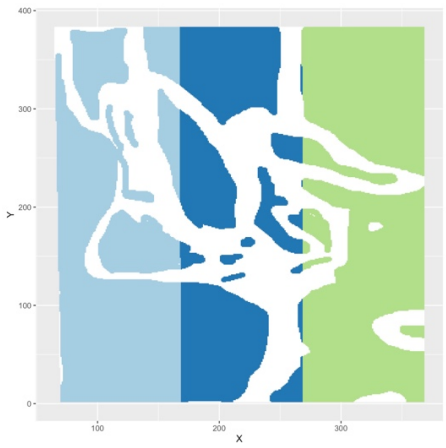

Figure (7): Block split: image 1

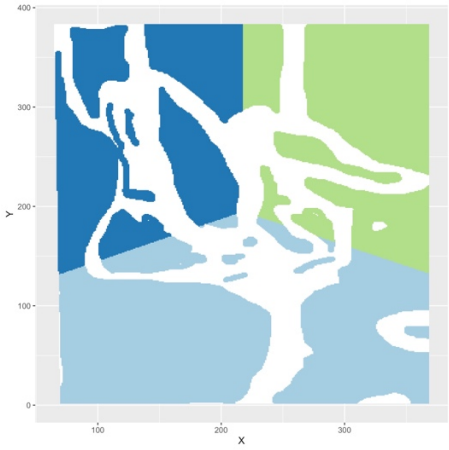

Figure (8): K-means split: image 1

2.2 Baseline Accuracy

The accuracy of a trivial classifier that predicts all pixels to be non-cloudy (Label=-1) is shown in Table (6) below. If an evaluation set comprises mainly of cloud-free pixels (imbalanced data), a trivial classifier would perform very well. The baseline accuracy serves as a great tool for model evaluation: performance of models should be evaluated based on their relative accuracy to the baseline, since any classifiers with good predictive power should do better than simply guessing negatives.

As shown in Table (6) below, test and validation sets created by block splitting are balanced, with baseline accuracies of both sets close to half (41%). But the baseline accuracy for the K-means split is quite high, especially for the validation set (93%).

	Accuracy
K-means Test	0.71
K-means Validation	0.93
Block Test	0.41
Block Validation	0.41

Table (6): Baseline accuracy by split method

	K-means split	Block split	Avg
NDAI	2.32	2.57	2.44
SD	1.42	1.71	1.57
CORR	2.25	1.71	1.98
Rad_Df	0.31	0.08	0.19
Rad_Cf	-0.33	-0.55	-0.44
Rad_Bf	-0.84	-0.98	-0.91
Rad_Af	-1.15	-1.21	-1.18
Rad_An	-1.23	-1.25	-1.24

Table (7): Logistic regression coefficient estimate

2.3 First-Order Importance

To select the “best” features, we must first define a clear criterion. We define a good feature as the one that has the strongest discriminative power for label classes (cloud or cloud-free). We quantitatively define this criterion as the feature coefficient in a logistic regression model, since the larger the coefficient, the greater the “propensity score” of a feature. In other words, the value of that feature is most associated with the occurrence of being ‘cloudy’.

To estimate the coefficients, we trained a one-predictor logistic regression model for each feature. The coefficient estimates are shown in Table (7) above. For both data splitting methods, author-defined features (NDAI, CORR, SD) has the largest average absolute coefficient estimates, at 2.44, 1.98, and 1.57 respectively. This means that the greater the feature values, the larger the probability that the corresponding pixel is cloudy.

To visualize the first-order importance, we plot a 95% confidence interval of the coefficient estimates for both ways of data splitting. As shown in Figure (9) below, the lower bounds of NDAI coefficient are higher than the upper bounds of CORR coefficient for both split methods. In addition, the lower bounds of CORR coefficient are higher or close to the upper bounds of SD coefficient for both split methods. This suggests that the “true” coefficients of the three predictors likely have the order: $\beta_{NDAI} > \beta_{CORR} > \beta_{SD}$. By this criterion, NDAI is the “best” feature, CORR second, and SD the third.

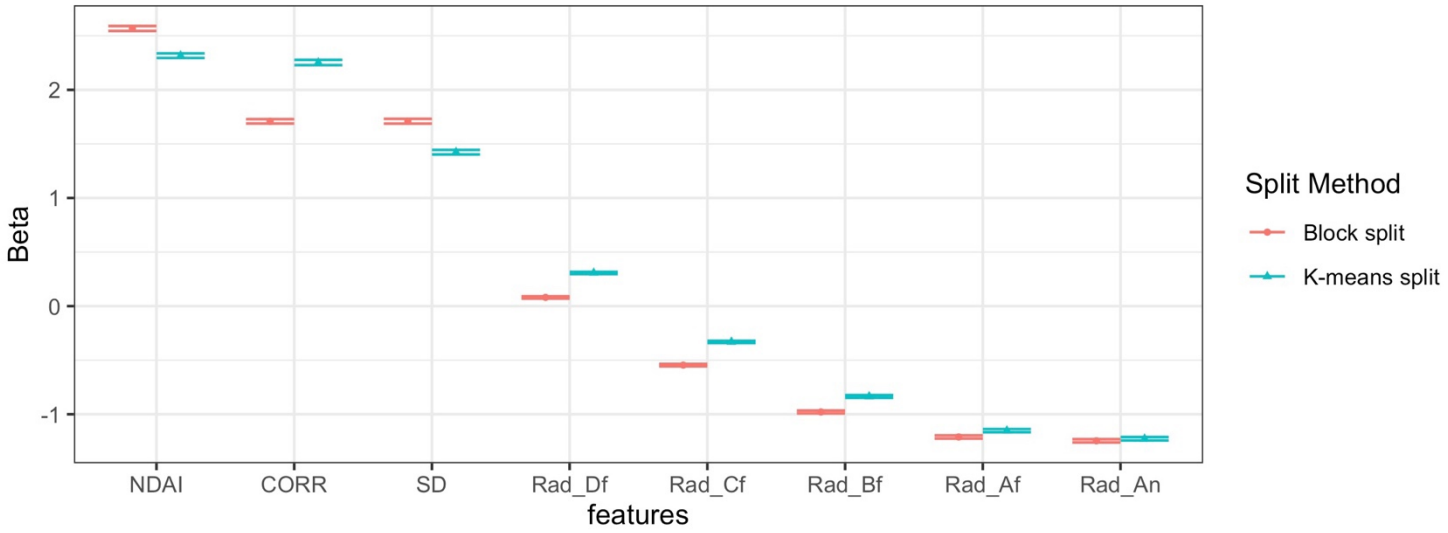

Figure (9): 95% Confidence Interval of logistic regression coefficient estimate

2.4 Cross Validation

To implement a generic Cross Validation function that helps us easily train and cross-validate models using the 2 split methods, we adopted the "tidymodels" framework in R. The framework integrates numerous statistical learning packages and unifies the modeling workflow, making it extremely easy to switch between algorithms. Moreover, it provides built-in cross-validation and parameter-tuning functions for model training and comparison. To integrate the framework into our CVmaster function, we split the training data by the splitting method specified, turn it into a format that is usable by 'tidymodels', and provide it to the 'fit_resamples' function. If there are parameters to tune for an algorithm, the user can provide a data frame containing tuning values, and it will be fed to 'tune_grid' in 'tidymodels'. At the end of the cross-validation, the loss of the best model (by the criterion specified by users) will be returned.

To use the CVmaster function, a user simply needs to provide a 'model' object in 'tidymodels' framework, a training dataset, the number of fold K, and a 'metric_set' object in 'tidymodels' framework which specifies the loss functions.

3. Modeling

We tried 6 classification models using both ways of data splitting and assessed the fitted model using prediction accuracy and Area Under Curve (AUC) in a 5-fold Cross-Validation. We combine the training and validation set into a single training set to train the models, since we will re-split the data in the cross-validation process. We first list the models and described their assumptions:

Logistic Regression: It assumes response $Y_i \mid X_i$ to be independent, and $Y_i \mid X_i \sim Bernoulli(logistic(X_i^T \beta))$. Since our spatial data has spatial dependence, this independence assumption is clearly violated. The distributional assumption seems reasonable here, as our response is a 2-class label which is binary.
Linear Discriminant Analysis (LDA): It assumes covariates to follow normal distribution, and that the covariance matrix of the covariates is the same across different classes. In addition, it assumes independence of observations. As shown in Figure (5) above, covariate densities are heavy-tailed and skewed, which do not look like normal distribution. In addition, we can see from the same figure that the covariance between predictors seem to be different for different classes. Moreover, since our data has spatial dependence, the independent observation assumption is also violated.
Quadratic Discriminant Analysis (QDA): It assumes covariates to follow normal distribution, and that the covariance matrix of the covariates is different across different classes. In addition, it assumes independence of observations. As mentioned above, covariates do not look normal in our data which violates our assumption. And the spatial dependence of our data also violates the independent observation assumption.
Naive Bayes: It assumes that the predictors are independent given the class. In addition, it assumes independence of observations. As shown in the correlation figure and due to the nature of our data, both assumptions are violated.
Random Forest: It is an algorithmic model that does not make specific assumptions
XGBoost: It is an algorithmic model that does not make specific assumptions.

To assess fit of the models, 5-fold CV accuracy using both ways of splitting is shown below in Table (8) & (9). For Random Forest and XGBoost, parameter combination with the best 5-fold CV average accuracy out of 30 combinations are selected:

Fold	Logistic regression	Naive bayes	LDA	QDA	Random Forest	XGBoost
1	0.981	0.835	0.859	0.855	0.811	0.889
2	0.821	0.700	0.846	0.861	0.944	0.988
3	0.849	0.977	0.983	0.991	0.986	0.989
4	0.838	0.980	0.821	0.934	0.924	0.785
5	0.833	0.890	0.828	0.851	0.736	0.774
Avg	**0.864**	**0.876**	**0.867**	**0.898**	**0.880**	**0.885**

Table (8): 5-Fold CV accuracy using K-means split

Fold	Logistic regression	Naive Bbayes	LDA	QDA	Random Forest	XGBoost
1	0.864	0.932	0.880	0.893	0.912	0.910
2	0.815	0.885	0.830	0.848	0.894	0.891
3	0.804	0.802	0.812	0.755	0.896	0.900
4	0.935	0.907	0.936	0.933	0.966	0.970
5	0.939	0.915	0.937	0.966	0.945	0.957
Avg	**0.871**	**0.888**	**0.879**	**0.879**	**0.923**	**0.926**

Table (9): 5-Fold CV accuracy using block split

Model performance on the test set in terms of accuracy, AUC, F measure and Kappa are summarized below in Table (10) & (11):

Model	Accuracy	AUC	F measure	Kappa
Random Forest	0.932	0.992	0.954	0.826
LDA	0.926	0.998	0.950	0.807
XGBoost	0.920	0.993	0.946	0.792
Logistic	0.918	0.997	0.945	0.782
Naive Bayes	0.859	0.989	0.909	0.601
QDA	0.794	0.998	0.873	0.370

Table (10): Test performance using K-means split.

Model	Accuracy	AUC	F measure	Kappa
Random Forest	0.989	0.999	0.987	0.977
XGBoost	0.987	0.999	0.984	0.973
QDA	0.975	0.976	0.969	0.948
Naive Bayes	0.971	0.962	0.964	0.940
Logistic	0.968	0.982	0.960	0.933
LDA	0.967	0.979	0.960	0.932

Table (11): Test performance using block split

Based on the CV results and test performance, Random Forest, XGBoost and QDA perform the best. From the CV result, although QDA performs better than tree-based classifiers in the K-means split, its performance is worse than the two in the block split. In contrast, tree-based classifiers Random Forest and XGBoost perform consistently well in terms of CV accuracy in both ways of splitting the data.

One interesting observation is that the test AUC of the two tree-based classifiers pales in comparison to QDA when K-means split is used, even though QDA has a much lower test accuracy (only marginally higher than the trivial classifier, which is at 71%). Regardless, the tree-based classifiers still obtain very satisfactory test AUC at 0.992 and 0.993. When we look at the test performance using the block split, Random Forest and XGBoost outperform all the other methods both in terms of test accuracy and test AUC.

We consider Random Forest to be the best classifier given its stable performance using both ways of splitting the data.

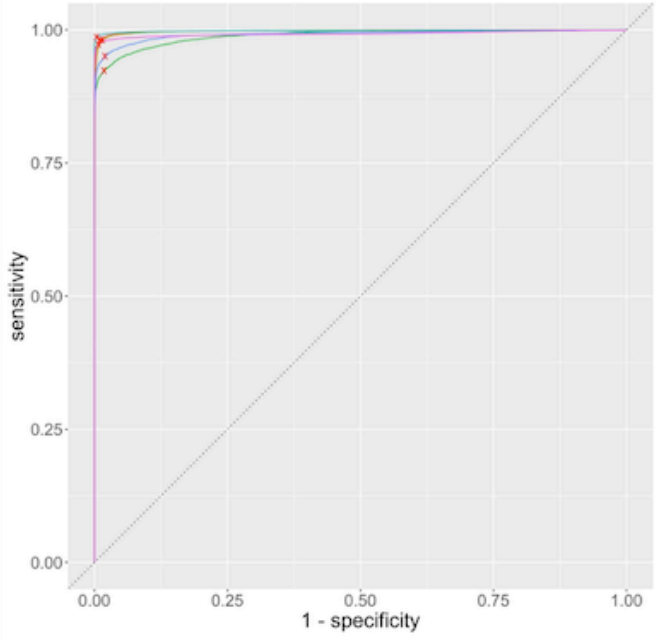

Figure (10): Test ROC using K-means split

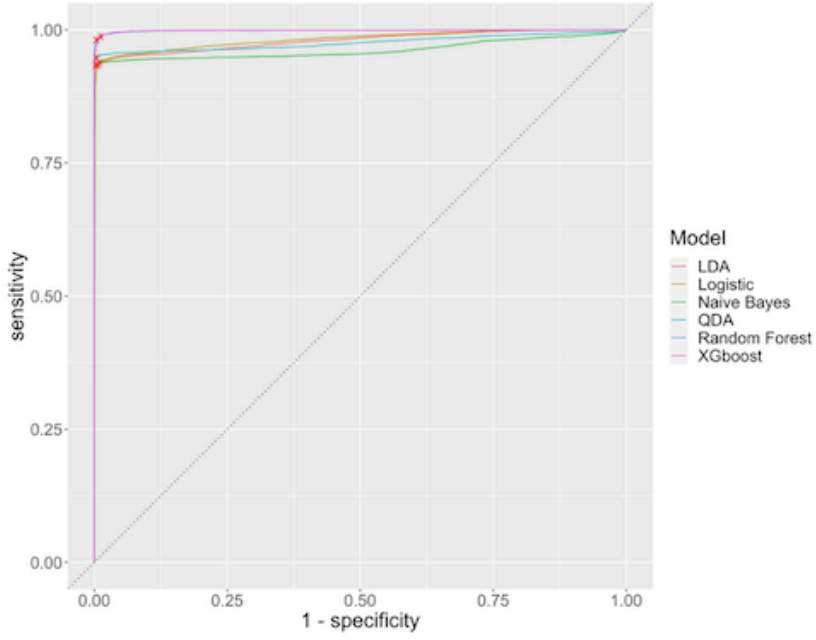

Figure (11): Test ROC using block split

From the ROC curves in Figure (10) & (11) above, all methods perform very well on the test set in both ways of splitting, with AUCs being higher than 0.96. Using the K-means split, Random Forest and Naive Bayes perform slightly worse than the others but are still considered good (AUC are 0.992, 0.989 respectively). Using the Block split, XGboost and Random Forest outperforms the other methods, and they are almost the perfect classifiers (both achieve 0.999 AUC).

An ideal cutoff value for the ROC curves depends on the goal of the classification problem at hand. If the cost of a False Negative prediction is high, one might want to minimize FNR (pick the highest "sensitivity"). If the cost of a False Positive prediction is extremely high, one might want to minimize FPR (pick the lowest "1-specificity"). In our classification problem, we want to accurately predict both cloudy and cloud-free pixels, as they exhibit very different dependencies of surface air temperature on atmospheric CO2 level. Misclassifying a cloudy pixel as cloud-free and vice versa will equally affect the analysis. So, we think the cost of FN and FP are similar in our settings. Therefore, our goal is to maximize both TPR (sensitivity) & TNR (specificity) at the same time. Mathematically, we pick the cutoff points that maximize $(\text{sensitivity}^2 + \text{specificity}^2)$. We maximize the squared sum instead of the sum since it penalizes the imbalance of the two values more. The cutoff values for each ROC curves are highlighted in red cross in Figure (10) & (11).

Bonus part

To better assess the fit of the models, we included F measure & Kappa in Table (10) & (11) above. F measure, which is defined as $F = 2 * \frac{precision * recall}{precision + recall}$, is the harmonic mean of precision and recall. It is another measure of test's accuracy, but it balances precision and recall on the positive class (cloud label), whereas accuracy looks at both true positives and true negatives. In general, F measure is more relevant when the cost of false negatives is high and when the data is imbalanced. If the cost of misclassifying a cloud pixel as cloud-free is high, or when an image has very imbalanced classes, F measure is a more informative measure of performance. Kappa is like accuracy, but it is more useful when one class has a large frequency. It tells us how much better our classifier is compared with simply guessing at random by the class frequency. From the two figures, Random Forest and XGBoost wins over most of the other methods in terms of test set F measure and Kappa, suggesting that they are a good fit for the data.

Moreover, in order to evaluate fit of our models, we look at ROC curves on the validation sets for different folds. If a model consistently obtains a "nice" ROC curve for different folds, we can expect its performance on future unseen data to be stable. This is more informative than simply looking at the fold-wise AUC, since AUC does not provide information on the trade-off between FPR and TPR.

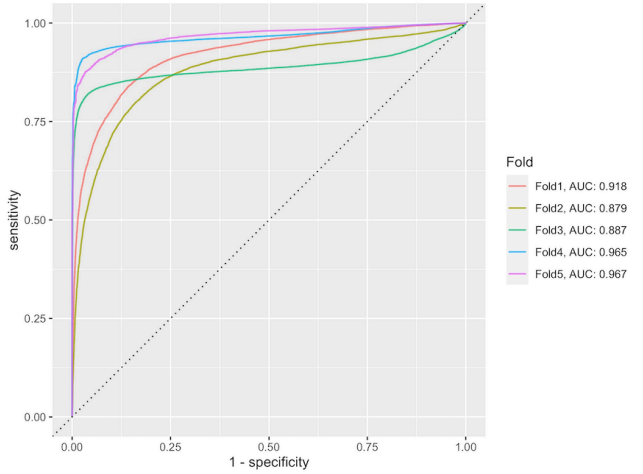

Figure (12): Logistic regression ROC of CV folds using block split

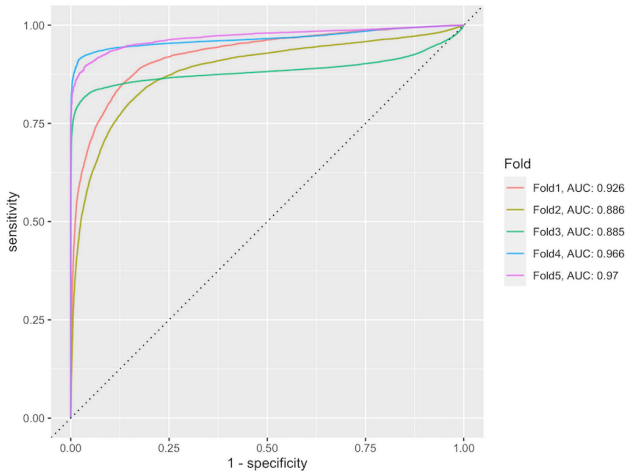

Figure (13): LDA ROC of CV folds using block split

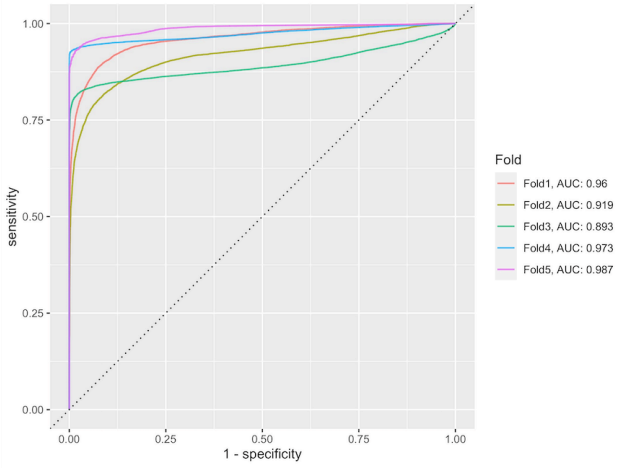

Figure (14): QDA ROC of CV folds using block split

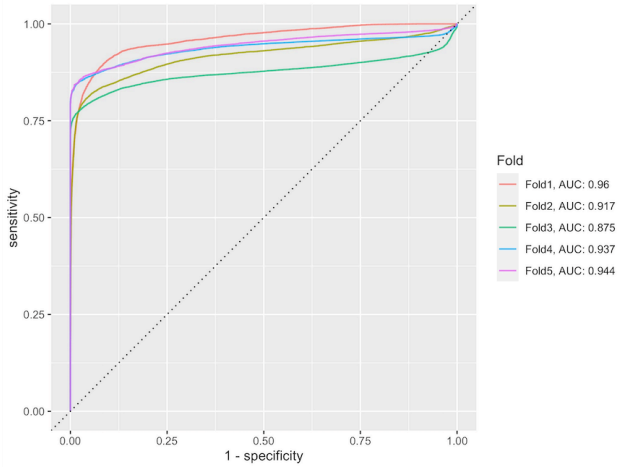

Figure (15): Naïve Bayes ROC of CV folds using block split

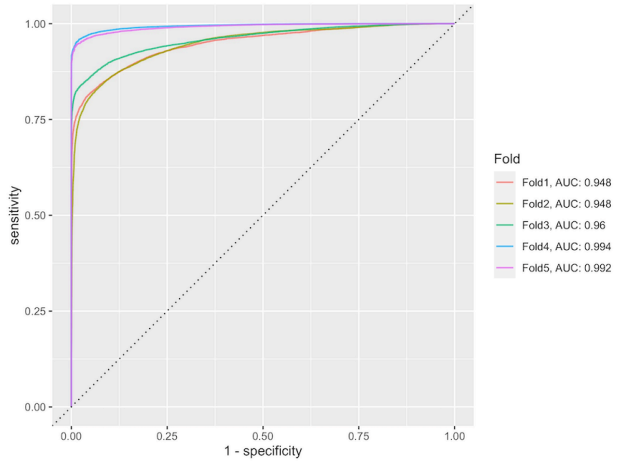

Figure (16): XGBoost ROC of CV folds using block split

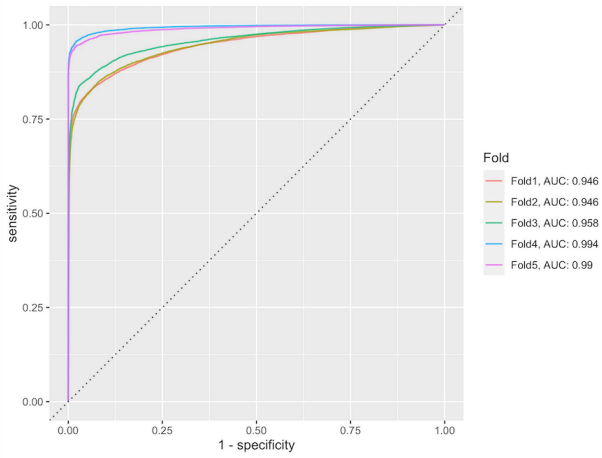

Figure (17): Random Forest ROC of CV folds using block split

As shown above in Figure (12) – (17), XGBoost and Random Forest have more consistent shape of ROC curves across folds than other models. It means that they provide a similar trade-off between TPR and FPR on different folds. This can be a desirable property as users might want to maintain a specific target FPR on future prediction (that has no expert labels). If a model provides unstable trade-off that varies a lot when a different training set is used, it would be hard for users to determine a cut-off value, or estimate their FPR on the future predictions without expert labels.

4. Diagnostics

We pick Random Forest as the best classifier due to its stable performance. For the analysis below, we will use the block-splitting method to split the data and perform cross-validation.

For Random Forest, there are 3 main parameters to tune: mtry (number of randomly selected predictors for each split), trees (number of trees) and min_n (minimal node size to split). Previously in our 5-fold cross-validation, we tested 30 combinations of (mtry, min_n) while fixing the number of trees to be 300, and we obtained a pair of parameters that achieve the best 5-fold CV accuracy. The parameters are: (mtry, min_n, trees) = (2, 6, 300). We will first analyze the optimal parameters in terms of 5-fold CV accuracy.

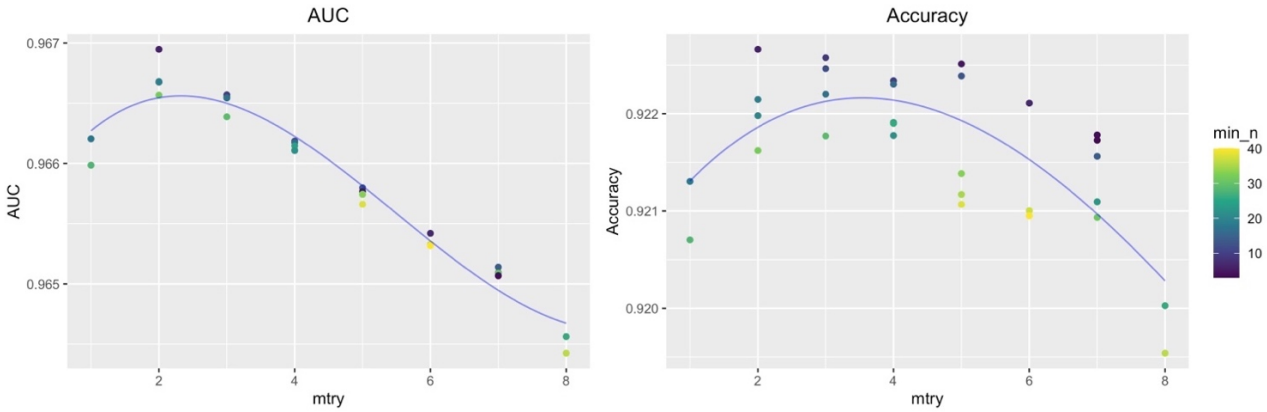

Figure (18): 5-fold CV performance of random forest with varying parameters

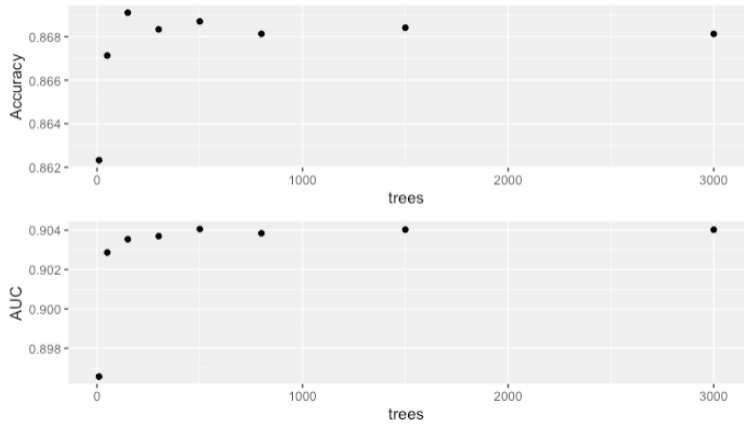

Figure (19): 5-fold CV performance of random forest with varying trees, fixed mtry=2, min_n =6

As shown above in Figure (18), both the 5-fold CV accuracy and AUC achieves maximum at around mtry=2 or 3 and decrease afterward. The lower the mtry, the fewer predictors are selected at each split, and hence the greater the bias. But the variance will decrease as trees are less correlated. This suggests that more uncorrelated trees are favorable in this problem, and the optimal mtry is likely to be either 2 or 3.

When we look at the performance of different min_n at mtry=2 or 3, we see that lower min_n yields better accuracy and AUC, and the best min_n is likely to be below 10, as performance increases when min_n decreases. Combining the two observations, we conclude that the optimal pair of (mtry, min_n) is close to (2, 6) when we fixed trees=300.

Next, we look at the 5-fold CV accuracy & AUC with varying number of trees and fixed (mtry, min_n) = (2, 6). As shown above in Figure (19), tress=300 is not the optimal parameter, and the performance continues to improve until at ~500 trees. Beyond that, there is no more improvements. Hence, we determine that the optimal parameters for Random Forest are around (mtry, min_n, trees) = (2, 6, 500).

To assess whether the model overfits the data, we look at the ROC curves of different validation folds. As shown below in Figure (20), the AUC of folds varies from 0.946 to 0.99. The model performs extremely well on 2 folds and relatively poor on the remaining 3 folds. Still, it is an excellent classifier, achieving an AUC of above 0.945 for all 5 folds. We can expect the model to perform relatively well on future unseen data.

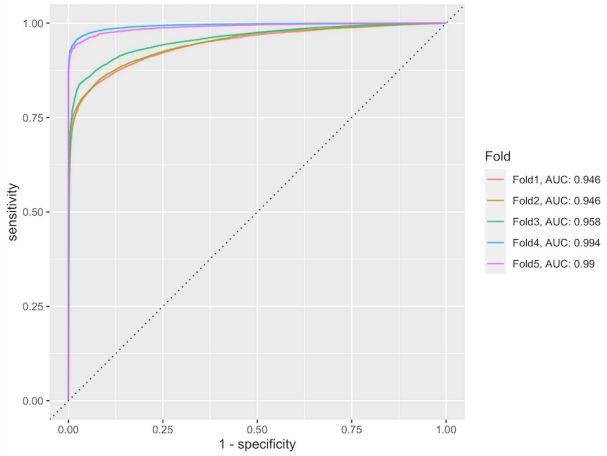

Figure (20): 5-fold CV fold-wise ROC.

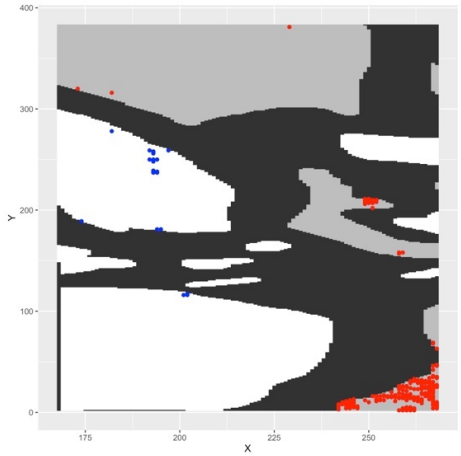

Figure (21): FP (red) &FN (blud) in test set (image 1)

Based on Figure (21) above, misclassified points are mostly located at the boundaries with unlabeled pixels (colored in black). Moreover, they tend to be close with each other, forming clusters. This is reasonable as unlabeled pixels usually have predictor values that are common in both cloud & cloud-free pixels, making them difficult for the experts and our model to classify.

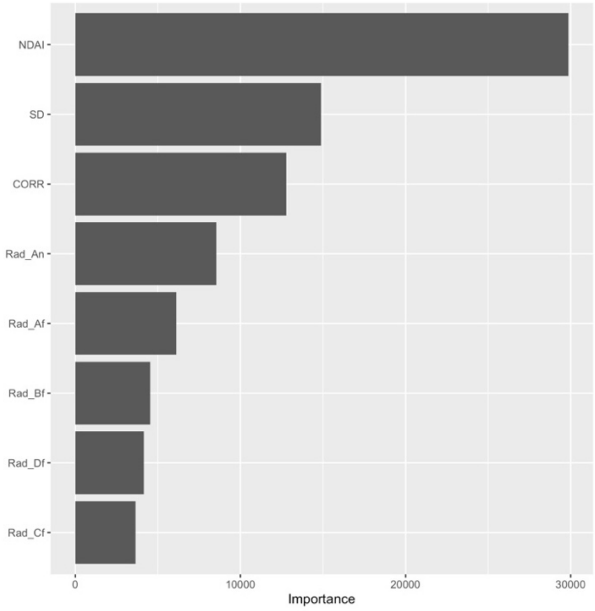

Figure (22): Impurity-based feature importance

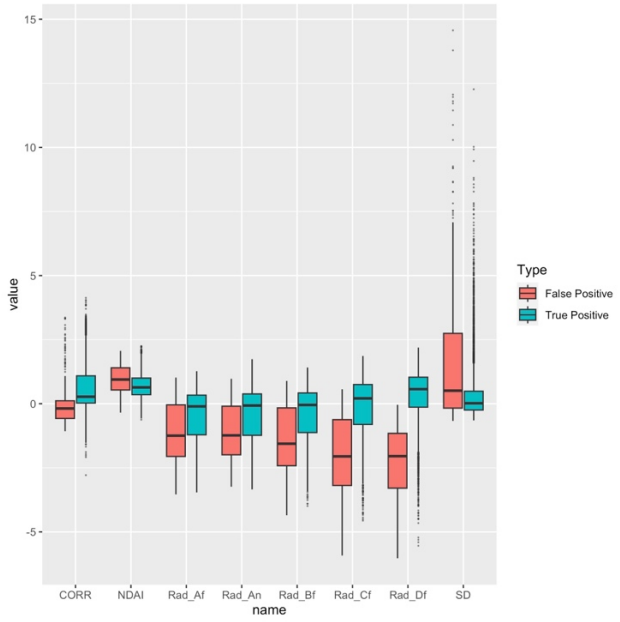

Figure (23): Standardized feature values of FP points

To gain intuitions of how the classifier works, we look at the impurity-based feature importance ranking in Figure (22) above. NDAI, CORR, and SD are the most important features as splitting by these predictors reduces impurity in tree nodes the most. This agrees with our EDA, which shows that they have the strongest predictive power for cloud labels.

Moreover, since radiance values are less important features, we can fix them at their means, and create an approximate 3-dimensional decision boundary, as shown in Figures (24a) & (24b) below. We observe that the model classifies pixels as cloud-free if they are in the three rectangular spaces. Very interestingly, this classification rule is very similar to the one used in the ELCM algorithm in the paper, which is discussed in the paper summary section, except that it has 2 more small rectangular blocks that are classified as cloud-free. This verifies that our classifier produces reasonable decision rules.

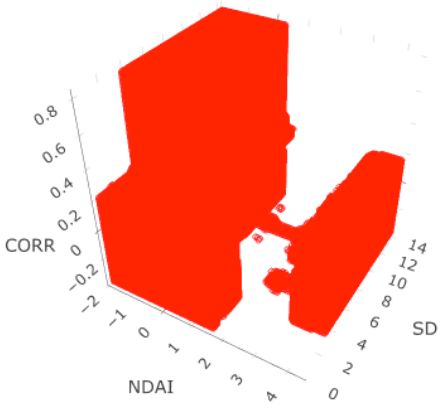

Figure (24a): Angle 1

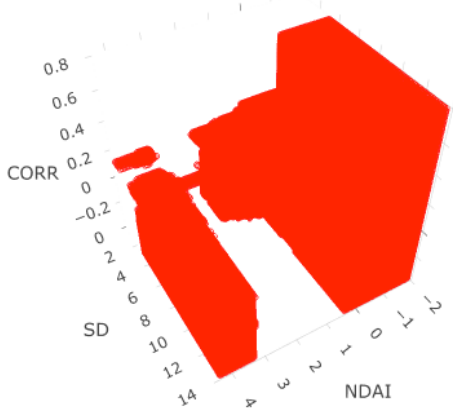

Figure (24b): Angle 2

Figure (24): Approximate 3D decision boundary with regions of 'cloud-free' (Colored in Red)

In order to create a better classifier, we look at predictor values of misclassified points to see if they exhibit certain patterns. Since there are many more false positives than false negatives as shown in Figure (21), we will focus on the former. As shown in Figure (23) above, FP pixels have very similar NDAI values as TP, this is likely why they are misclassified. Moreover, we noticed that true positive pixels have increasing radiance values from Rad_Af to Rad_Df, whereas false positive pixels have decreasing radiance values. This "linear slope" of radiance values, which is related to the 'CORR' variable, seems to be more parabolic for the false positive pixels. Therefore, we might be able to reduce false positives by incorporating a "non-linear slope", a new variable which is the degree 2 coefficient of degree 2 polynomial regression across the radiance values from Rad_Af to Rad_Df. As shown in Figure (25) below, this new variable has similar density for TP & FN pixels, and for TN & FP pixels, suggesting that it could improve our classifier.

By increasing the number of trees to 500, and incorporating the new feature mentioned above, we yield a new classifier that achieves higher test set accuracy of 99.17% (previously 98.9%). Moreover, as shown in Figure (26) below, the new variable is the 4th most important feature, exceeding all radiance values.

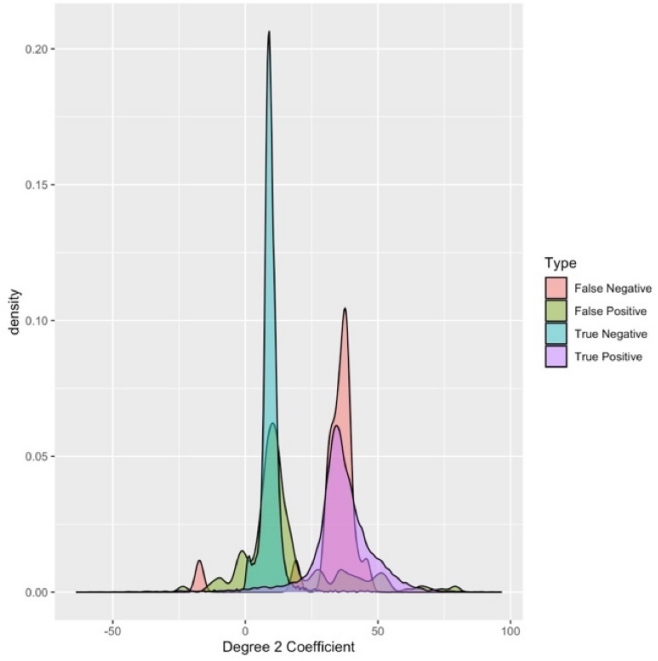

Figure (25): Density of degree 2 coefficient

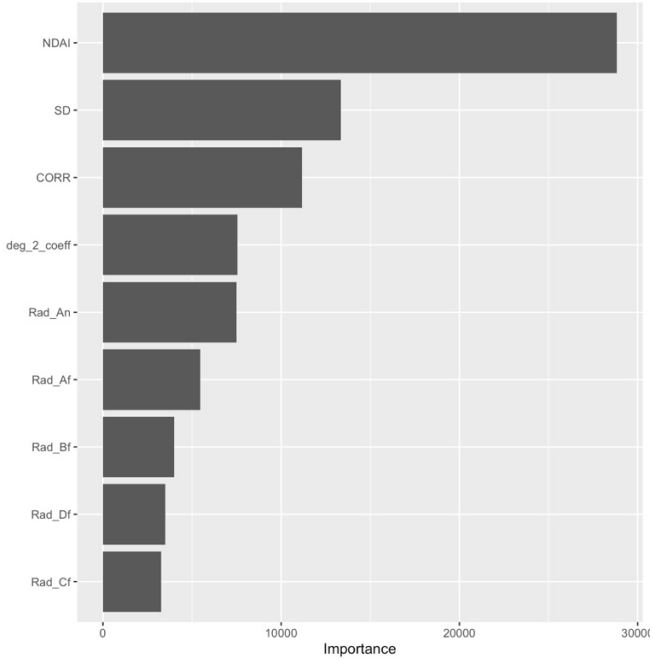

Figure (26): Impurity-based feature importance of new classifier

Moreover, we repeated the analysis using the K-means split method and obtained a similar but slightly different result. From Figure (27) & (28) below, we see that the optimal (mtry, min_n, trees) in terms of AUC is still around (2, 6, 500), but the optimal 'mtry' seems to be 1 in terms of CV accuracy. From Figure (29) below, the top 3 most important features are still NDAI, CORR & SD. In addition, based on Figures (31a) & (31b), the approximate 3-d decision boundary is

almost the same as the previous one, except that it is less noisy, and the two blocks are not connected. Again, this classification rule is similar to the previous one, and also similar to the one used in the ELCM algorithm in the paper. This is a good sign that our model does not overfit the data as decision rules remain similar when a different splitting method is used.

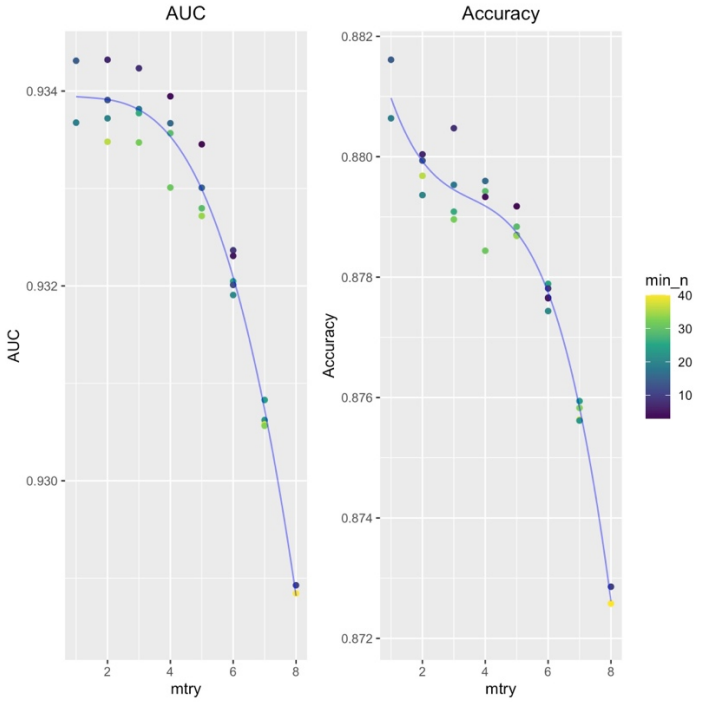

Figure (27): K-means split 5-fold CV with varying parameters

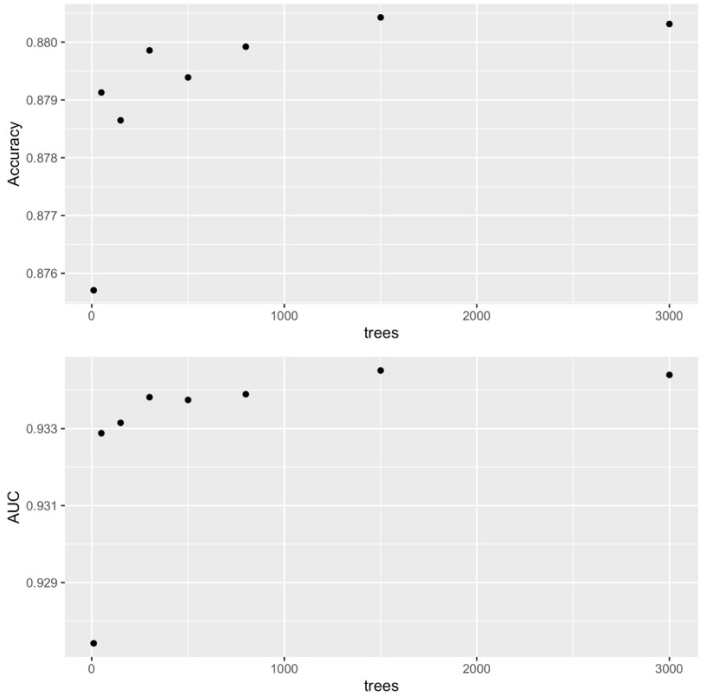

Figure (28): K-means split CV with varying trees

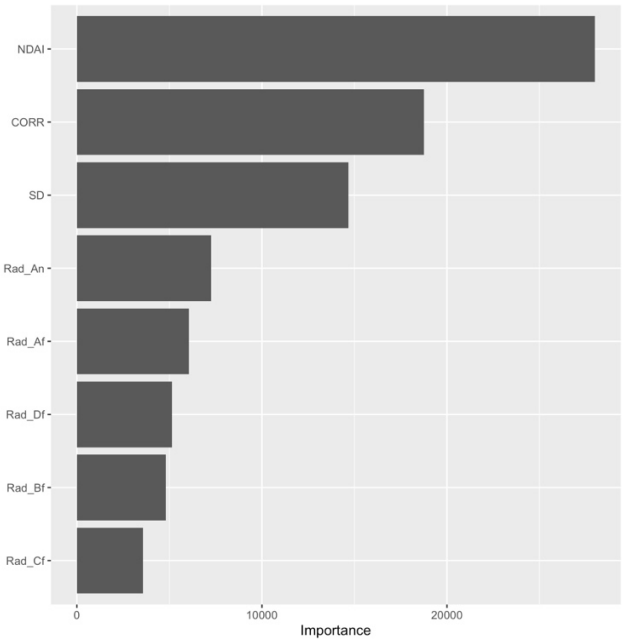

Figure (29): K-means split feature importance

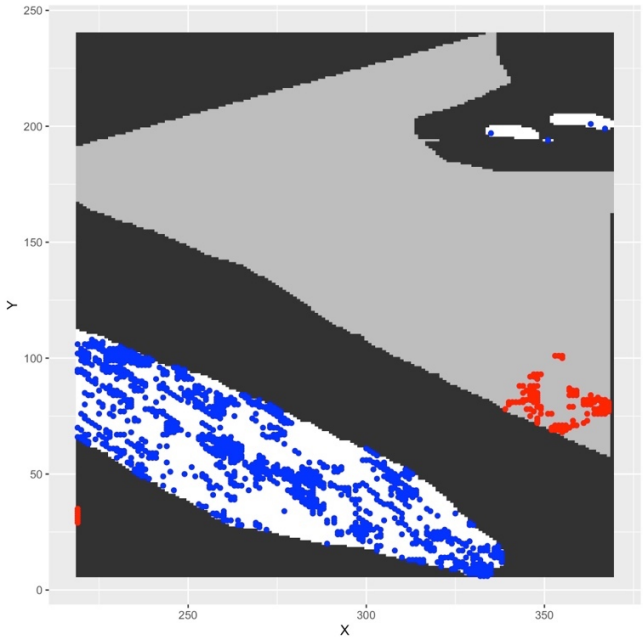

Figure (30): K-means split test set FP (red) & FN (blue) points

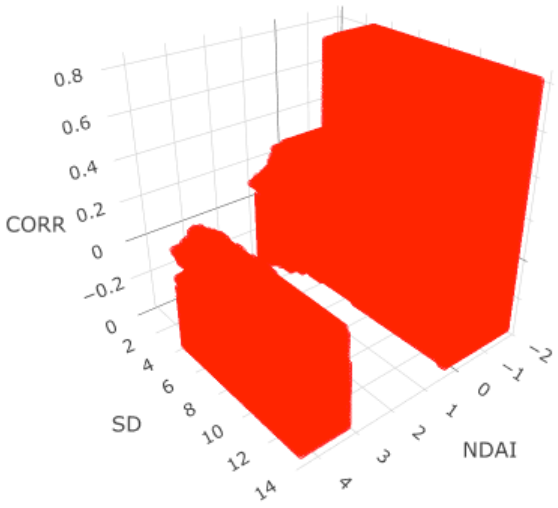

Figure (31a): Angle 1

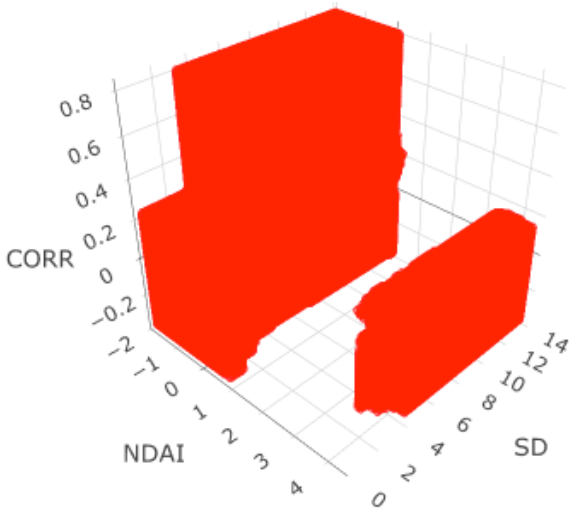

Figure (31b): Angle 2

Figure (31): Approximate 3D decision boundary with regions of 'cloud-free' classification marked in red using K-means split

Conclusion

Training a model to accurately identify cloud pixels in the arctic is difficult because of the similarity between clouds and snow- and ice-covered surface. In this project, we have demonstrated that state-of-the-art tree-based methods like Random Forest and XGBoost achieves the goal very well.

From the EDA & diagnostic section, we discovered that feature engineering is a key part of predictive modeling: well-designed features (three physical features proposed by Shi's team) can help create powerful classifiers. As shown throughout the project, expert-designed features (NDAI, SD, CORR) consistently win over raw features (radiances) as the most important variables.

In addition, spitting data that has dependency structure is not trivial. We must carefully consider the data splitting methods to ensure an unbiased evaluation of models. In the project, we discovered that block spitting and K-means spitting are good spitting methods for spatial data as they minimize the dependency between training and evaluation sets, and hence ensure an accurate estimate of model error. Moreover, it would be helpful to analyze model performance using different ways of data spitting, as models could exhibit different behaviors.

Finally, diagnosing and interpretating Random Forest is not an easy task, as it is an ensemble of many trees. But there are still helpful tools, such as feature importance and decision boundaries, to help us understand them. In addition, to train a good Random Forest model, we learnt that it is best to fine tune the parameters to find good bias-variance trade-off.

5. Reproducibility

All tables and figures contained in this project can be automatically regenerated & saved by running the R scripts inside the folder in specific order. Randomization seed is set at the beginning of the scripts to ensure reproducibility of results. A README file is included to provide descriptions and guidance to future users.

6. Acknowledgements

The project is contributed equally by Kwong Yu Chong and Jiahua Wang. Majority of sources comes from STA 521 Lecture notes. We have also read Information on the Internet regarding spitting methods of spatial data before conducting the project.