STAT 154: Project 2

Shichun Wang *SID: 26310944*
Yue Chen *SID: 26028535*
May 3rd, 2019

Data Collection and Exploration

(a) This paper presents a statistical case study of the clouds detection algorithms in the Arctic. The purpose of the study is to differentiate between cloudy pixels and cloud-free surface pixels in order to assist further scientific research. In addition, researchers worked on the algorithms so they could effectively handle a massive MISR dataset in an efficient way. The data was collected from 10 MISR orbits of path 26 in a study area over the Arctic: northern Greenland, and Baffin Bay. The recording lasted 144 days during a daylight season in the Arctic. Path 26 was chosen because it detected the area with various surface features, including permanent sea ice, snow-covered and snow-free coastal mountains, permanent glacial snow and ice, and ice-melted sea. Each orbit measured six data units including 7,114,248 1.1-km resolution pixels with 36 radiation measurements of each pixel. However, three out of sixty data units were excluded due to the ice liquefying. In this study, researchers proposed two new clouds detection algorithms for the Arctic, ELCM and ELCM-QDA, both of which offer more effective and automatic computations. Both algorithms are based on three physical features, namely CORR, SD, and NDAI, to identify the proper clouds coverage regarding the massive MISR data. This method significantly outperforms the previous model, making two strong impacts in the field. Frist, statisticians collaborated with science colleagues throughout the analysis, collectively devising the best methodologies. Second, this study demonstrates the power of the statistical learning. The combination of statistical learning and scientific domain knowledge together lead to effective and innovative solutions to challenging scientific issues.

(b) To begin, we take a brief look at three datasets. We notice that the ranges of the x and y coordinates are similar among all three datasets. The values of the x coordinate vary from 2 to 283; whereas the y coordinate' values are spread from 65 to 360. Next, we check the percentage of pixels in different classes for all three datasets. In dataset image1, the percentage of clouds, not clouds, and unlabeled is 44% versus 18% and 38%, respectively. Comparatively, 37% versus 41% and 29% in dataset image2, and 29% versus 18% and 52% in dataset image3. Additionally, other features of the three datasets have a similar range. For example, the ranges of the radiance for all angles are between 20 and 410. After plotting the region based on the expert labels (Figure 1), we discover that the unlabeled areas mostly lie between cloudy and clear area. This is not surprising because the area between cloudy and clear surface is usually harder to determine. In addition, the samples are obviously not i.i.d, otherwise the points would be more evenly distributed, and big clusters are not likely to appear.

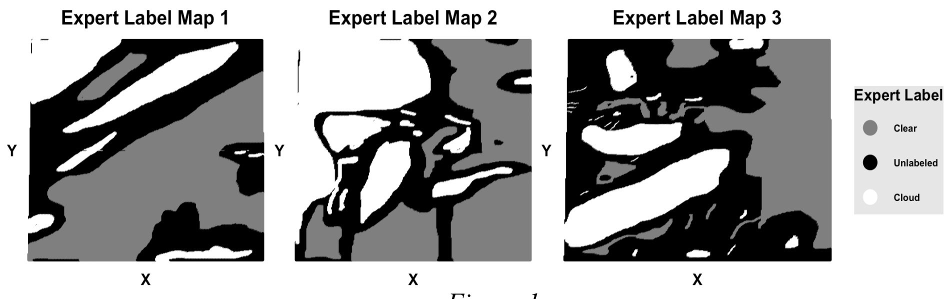


Figure 1

(c) Taking a closer look at the data set, we focus on Image1. By plotting the NDAI and SD, we can see a clear positive relationship (Figure 2). This is to be expected because both NDAI and SD measure the spread of radiances. Next, we plotted the Radiance angle AN versus CORR, respectively. We can observe that there is a negative relationship between the Radiance angle AN versus CORR with more points clustered at the upper left region (Figure 3). As such, the higher measure of radiance with angle AN leads to smaller correlations among different angles and wavelengths.

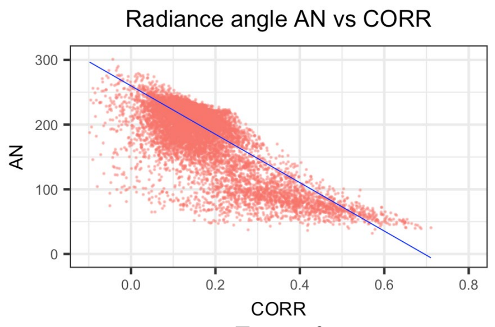


Figure 2

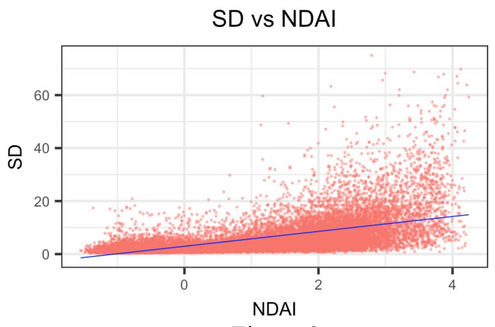


Figure 3

We merge the 3 datasets to find the relationship between classes based on radiance and other features. As we can see from the correlogram (Figure 4), many features have a strong correlation with the classes. NDAI has the strongest positive correlation, while classes and Radiance angle AN have the strongest negative correlation with classes, as indicated by the circle color. Therefore, we draw the plots of NDAI versus classes and Radiance angle AN versus classes. As shown in Figure 5, high values of NDAI suggest the presence of cloud pixels and negative values of NDAI indicate clear conditions. This is intuitive, because the visible wavelength is more isotropic for ice and snow-covered surfaces than clouds, which indicates a clearer area has smaller NDAI. In contrast, a lower value of the AN associates to a larger probability of cloud pixels (Figure 6).

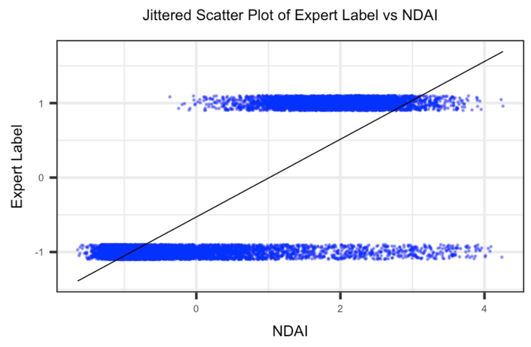


Figure 5

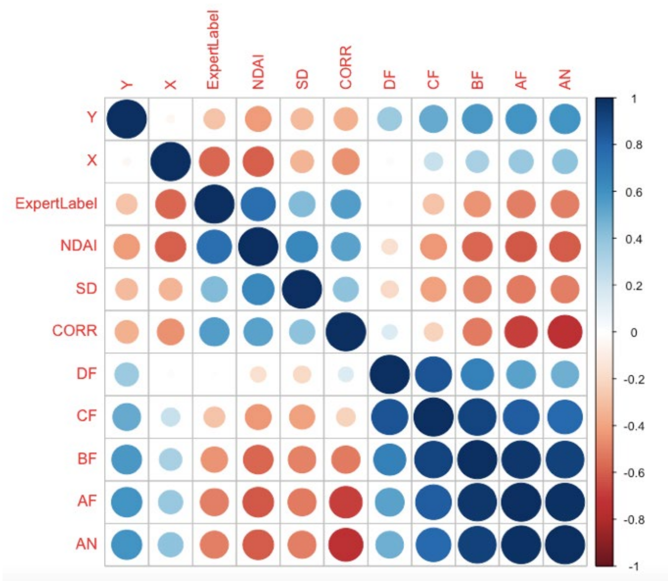


Figure 4

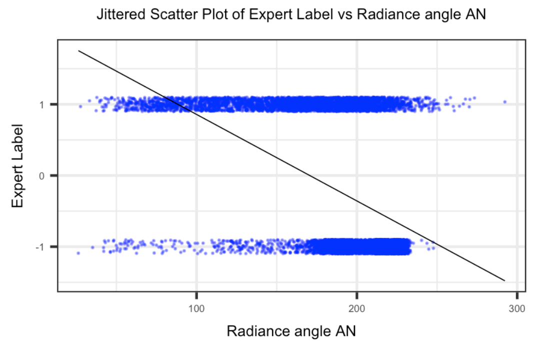
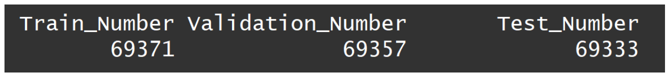


Figure 6

Preparation

(a) For the remainder of this report, we do not include data that is indecisive concerning clouds (i.e. the unlabeled points). As mentioned earlier, the data are not i.i.d distributed since the clouds and clear areas appear in clusters. Therefore, we cannot simply split the data randomly for cross validation or other purposes. We need to take into account of the dependence structure among the data as geographical structures and distances are crucial information. As such, we want to split the data in a way that preserves locality, but still randomly assigns points to different subsets. Thus, we come up with two ways to balance these two objectives: randomly assign points *within* small grids or assign the whole grids randomly to three sets.

Method A: First, we break the data set into small grids based on coordinates. Then, within each grid we randomly assign 1/3 of the points to one of three sets: test, train, and validation set. In other words, we are doing a stratified random sampling based on geographical location. This ensures the randomness when splitting the data. In detail, we choose to have grids that are 10*10 for each data set. Consequently, we can simply combine all the test sets, validation sets, and train sets across the three data sets and see that the split is fairly uniform (Figure 7).

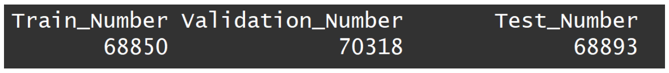


```
Train_Number Validation_Number       Test_Number
       69371             69357             69333
```

Figure 7

One potential criticism for this method is that it might not preserve the closest locality structure, meaning that the points right next to a specific point could be in any set. To address this problem, we have a second way of splitting the data.

Method B: We break the image into same number of grids as previous method and randomly assign each grid to a set. Nevertheless, the split for the second method is not as uniform as the first one, as some grids have very few labeled points (Figure 8). The difference is created by assign these grids to one of the sets.



```
Train_Number Validation_Number       Test_Number
       68850             70318             68893
```

Figure 8

(b) In this section, we set all labels to -1 (cloud-free) on the validation set and on the test set to find out accuracy. We use the 0-1 classification error described in ISL Chapter 2.2.4. The misclassification rate is defined as the proportion of mistakes that are made if we apply estimators to observations. The trivial classification (all cloud-free) misses about 39% of the points in both the validation set and the test set. This is unsurprising because the data has fewer clouded areas in the first place. This classification will have a high accuracy only if the proportion of -1's (cloud-free) is very high. In other words, if the actual labels are sparse, in which most of labels are -1's (cloud-free), then guessing -1's (cloud-free) for every point will have a relatively high accuracy. However, this is not the case here, where we have about 39% 1's (cloud) areas.

(c) For this section, we just choose Method A to explore the data. The reason why we apply this procedure to data set is to preserve locality. To understand the dataset holistically, we attempt to use boxplot and PCA analysis to reduce the dimension of the data. We try to find three "best" features, meaning those are most closely related to the Expert Label variable. In other words, these features best differentiate the cloud vs cloud-free areas. We first construct the boxplot with different features versus Expert Label. As shown in Figure 9, NDAI, CORR, and AN features appear to have the greatest difference between the two classes. To further explore the relationship, we apply a simple, naïve LDA algorithm, each using only one feature. Figure 10 shows the classification error rates among all the features, where we can see the minimum three are NDAI, CORR and AN. Additionally, by running the PCA analysis, we are able to confirm that NDAI, CORR and AN are indeed aligning closely to the Expert Label loading direction. The top two PCs explain more than 80% of the variability of the data. As shown in Figure11, the three loading vectors projected on the subspace that is closest to the direction of the Expert Label (or its opposite direction) are also NDAI, CORR, and AN.

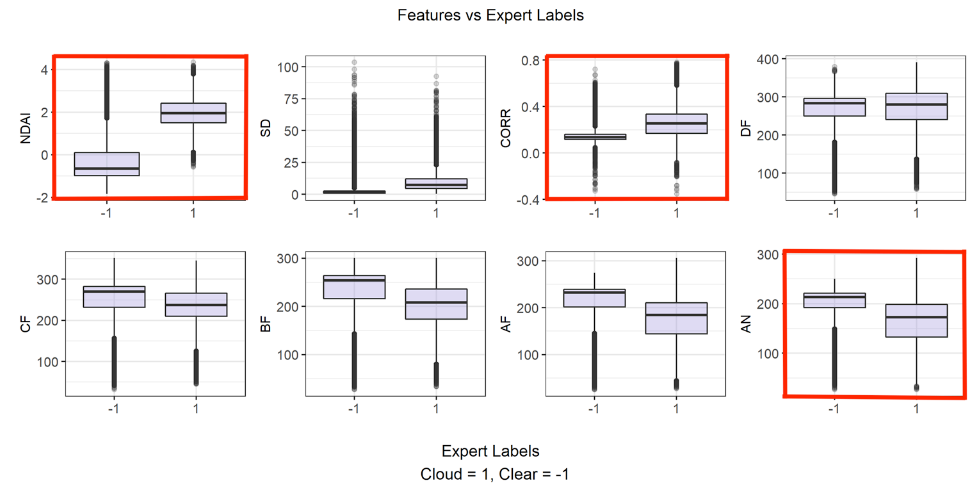
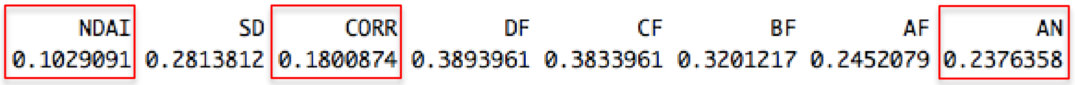


Figure 9



```
    NDAI        SD      CORR        DF        CF        BF        AF        AN
0.1029091 0.2813812 0.1800874 0.3893961 0.3833961 0.3201217 0.2452079 0.2376358
```

Figure 10

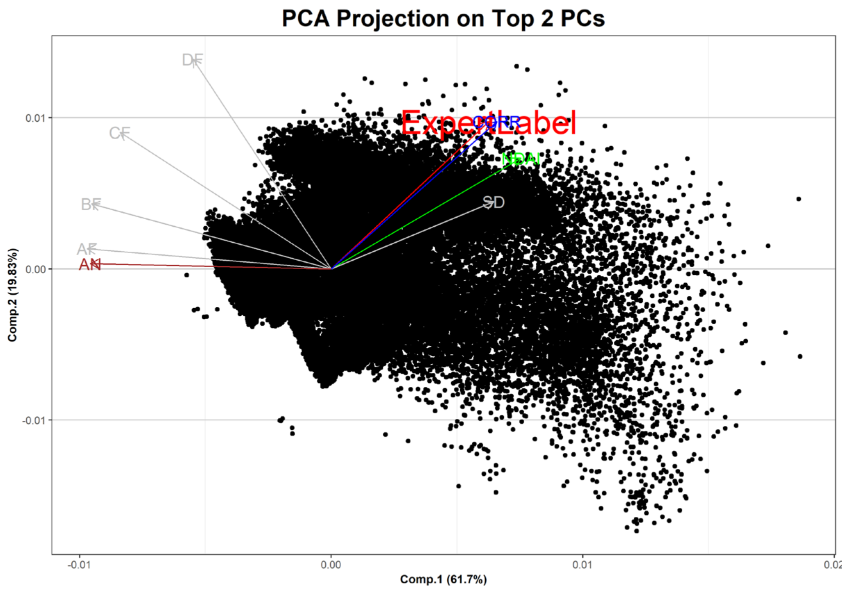


Figure 11

(d) For details, see the CVgeneric.R file in the repository. The main inputs of our function are:

1. classifier: classification method; a string. Supported models are: "logistic", "lda", "qda", "knn", "svm", "randomforest".
2. train: training features with X, Y data, excluding labels; a data frame
3. labels: response variable; a binary vector (of any type)
4. split_method: A character string ("A" or "B") specifying the data splitting method
5. loss: loss function. The default is our 01 classification error.
6. …: any hyperparameters can be passed in as well if needed.

We allow both ways of splitting data mentioned in 2a to create the folds. Next, we try out multiple classification methods, including Logistic, LDA, QDA, SVM, random Forest, and KNN, to find out each method's classification accuracy. We only look at the SVM method when the fold number is less or equal to three due to the time consideration, similarly for KNN. In addition to these methods, we are also allowing all of them to be performed on top of a PCA transformation of the dataset (if we set PCA = T).

Modeling

(a) As mentioned in 2d, our classification models include: LDA, QDA, Logistic, SVM, and random Forest. We need to check if the assumptions of each method we are going to use are satisfied in this case. First of all, i.i.d is the fundamental assumption of almost all statistical learning methods, and in our case, the whole purpose of creating folds in ways devised in 2a is to get quasi-i.i.d data. Moreover, the assumption of LDA and QDA models is that the data will follow the Gaussian distribution in each class under each feature. Figure 12 shows the features distribution based on cloud or clear. We can see that there is no bio-modal indication. However, some of them do show the skewness. This indicates that the assumption of LDA/QDA roughly holds. Second, Logistic Regression requires there to be little to no multicollinearity among the independent variables. In our data, the majority covariance among variables is less than 0.5, which means they are not highly correlated with each other and thus fulfill the assumption. Third, we are allowed to use SVM method since it is quite tolerant of input data, especially with the soft-margin version. Fourth, the only assumption of random Forest relies on representative sampling, which is satisfied in our case, and all of our explanatory variables are continuous (RF does not like categorical data).

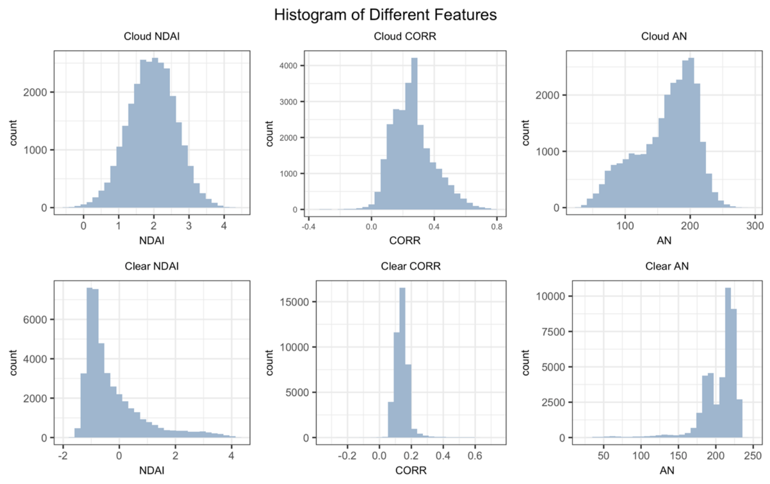


Figure 12

After checking all assumption hold, we assess models' fit using cross-validation. Note that our data to test ratio is 2:1, so in order to get the best approximation we will do a 3-fold CV to get the same ratio. Meanwhile, a small number of folds can also help us to speed up the process. To recap from 2a, we assess the accuracy for two splitting methods, with 10*10 grids.

Our initial thoughts of what methods to use are LDA, QDA, Logistic, SVM, and random Forest. We abandon KNN method because it is simply too time consuming. However, before we check out SVM and random Forest methods, we need to decide which splitting method to use, since

SVM and random Forest will also take quite long relatively. Also, we want to check whether PCA will make a difference or not. Our first-round method selections are LDA, QDA, and Logistic. After running 3-fold cross validation, we observe that PCA does not make a significant difference. This is because we only have 3 features in this case. Maybe we can get a better result by applying PCA methods when there are more features; however, this would need to be verified by further exploration. Together, we select the best average accuracy within each method family and also the methods with the highest single result. We can see from Figure 13 that QDA is uniformly better than LDA. Thus, our first round winners are GLM-A, GLM-B, QDA-A, QDA-B (A, B indicate the two ways of splitting methods). Next, we pick the best methods to calculate the test accuracies. As shown in Figure 14, the splitting method B provides better results than the splitting method A. The reason for this is that the splitting method B learns faster. In other words, the second splitting method takes advantage when we have more data to train. We will dive deeper into this issue in part 4. In conclusion, we will use non-PCA for computational speed, and splitting method B to run SVM and random Forest methods. After an hour, we get the results of SVM and random Forest and compare it with the first round winners. Figure 15 shows that random Forest and SVM have higher average accuracy and test accuracy than other methods. We will take a deeper exploration of ROC curve and AUC value in 3b in order to determine the best classification method.

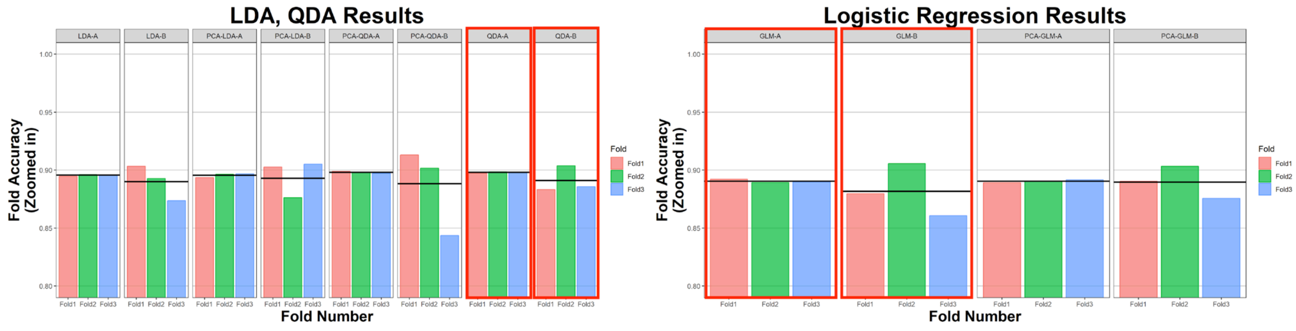
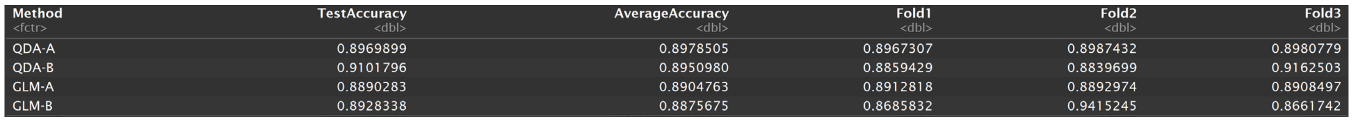
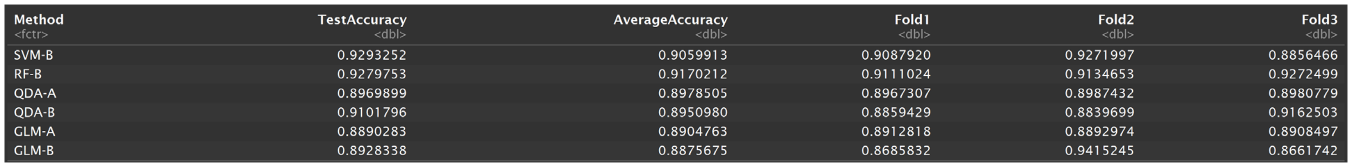


Figure 13



Method	TestAccuracy	AverageAccuracy	Fold1	Fold2	Fold3
<fctr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
QDA-A	0.8969899	0.8978505	0.8967307	0.8987432	0.8980779
QDA-B	0.9101796	0.8950980	0.8859429	0.8839699	0.9162503
GLM-A	0.8890283	0.8904763	0.8912818	0.8892974	0.8908497
GLM-B	0.8928338	0.8875675	0.8685832	0.9415245	0.8661742

Figure 14



Method	TestAccuracy	AverageAccuracy	Fold1	Fold2	Fold3
<fctr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
SVM-B	0.9293252	0.9059913	0.9087920	0.9271997	0.8856466
RF-B	0.9279753	0.9170212	0.9111024	0.9134653	0.9272499
QDA-A	0.8969899	0.8978505	0.8967307	0.8987432	0.8980779
QDA-B	0.9101796	0.8950980	0.8859429	0.8839699	0.9162503
GLM-A	0.8890283	0.8904763	0.8912818	0.8892974	0.8908497
GLM-B	0.8928338	0.8875675	0.8685832	0.9415245	0.8661742

Figure 15

(b) ROC plots the true positive rate against the false positive rate at each cutoff value, which provides tools to select optimal model within each family. Since we do not have a preference between true positive rate and false positive rate, we want the overall highest prediction accuracy of clouds and clear surfaces. We will select the curve and cutoff value based on the distance to point (0,1). From the eyeball testing of Figure 16, random Forest and SVM with Radial kernel seem to have the most accurate classification. In the next step, we adjust the parameters of random Forest and SVM with Radial kernel methods to check the model stability and further assess the model-fit based on AUC value. We can conclude from Figure 17-A & Figure 17-B that random Forest makes slightly better performance than SVM, and different *mtry* does not create significant difference results (since we only have 3 features to start with). In addition, since SVM takes longer time to train than random Forest, the final candidate is random Forest method with *mtry* equals to three. Finally, we decide the cutoff value to be 0.306 because this value provides the total smallest classification error (Figure 18-A & Figure 18-B).

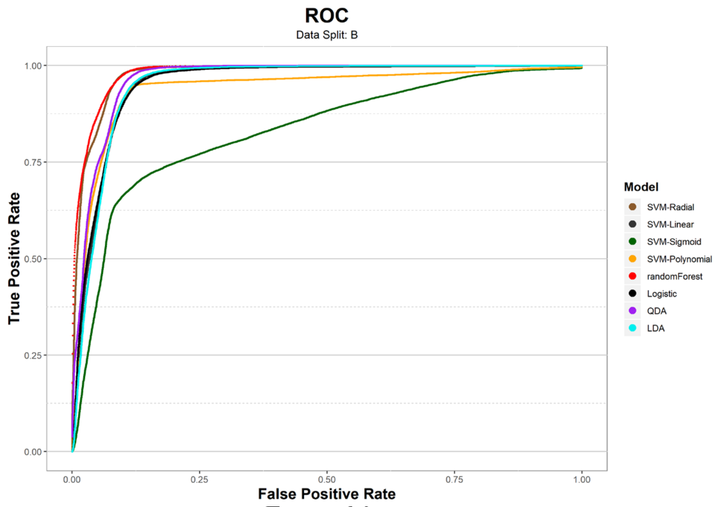
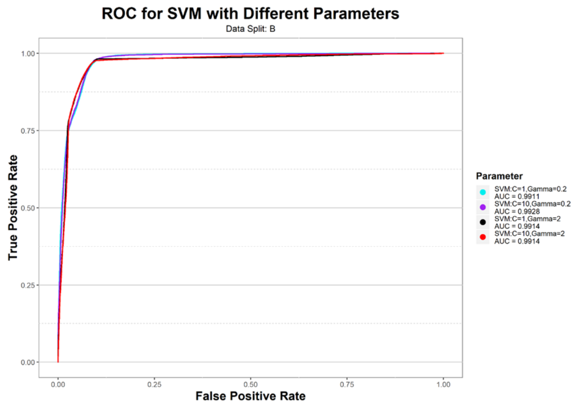


Figure 16



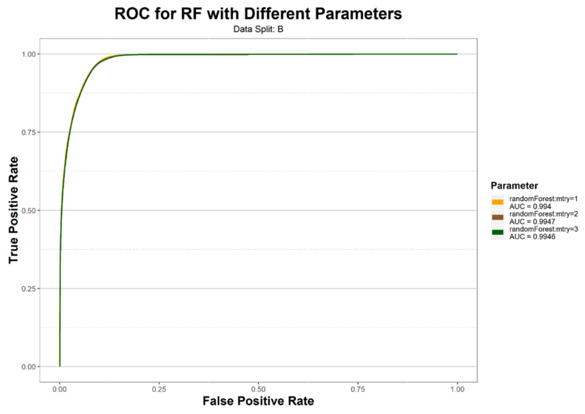
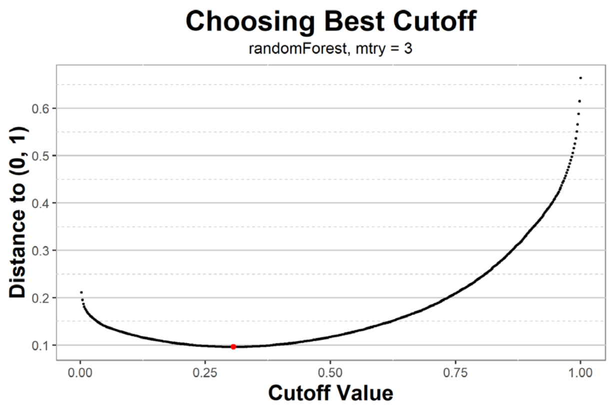


Figure 17-A *Figure 17-B*



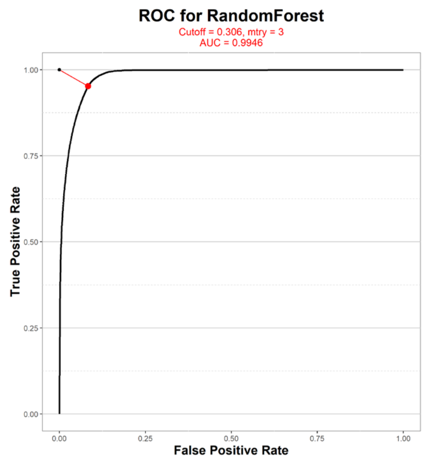


Figure 18-A *Figure 18-B*

Diagnostics

(a) As mentioned in part 3, our final candidate is the random Forest model. In this section, we will do an in-depth analysis of this model. First, we want to see how fast this model converges as more data come in. To measure this, we add in more and more data in the test and validation set to see how fast the accuracy increases. Specifically, we increase sample size by increments of 20,000, beginning with 20,000 and going up to 140,000 to check out the model converges (Note that we are not merely adding more data in, but each sample size is a fresh random sample created from data_split functions). After plotting the test accuracy graph that corresponds to train sample size (Figure 19), we conclude that when the train sample size larger than 60,000, the test accuracy starts to be more consistent (around 0.926). Furthermore, we want to check the time cost of training and prediction when the sample size increases. As shown in Figure 20, the fitting time complexity is approximately linear with respect to the train sample size. This means that it is probably a good idea to pursue the accuracy by increasing the train sample size.

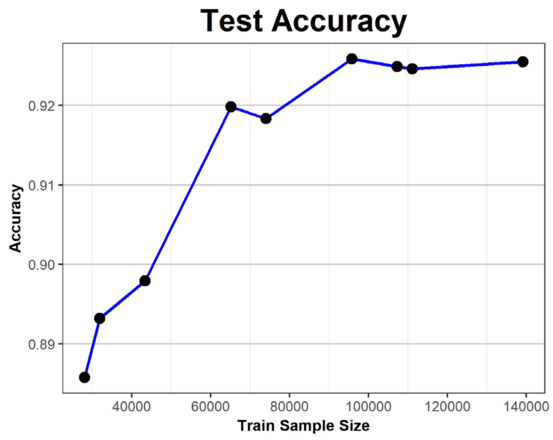


Figure 19

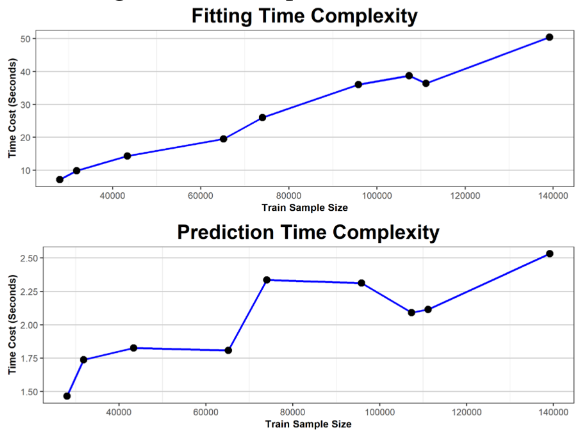


Figure 20

Secondly, we want to explore the relationship between the parameters and the test accuracy. We have already looked at the *mtry* and found that it does not make a significant difference since the dimension of feature space is quite low. Therefore, we will dive deeper into how the *nTree* (total number of trees) will affect the model fitting. Since we have seen that half of the current sample size can maintain stability, we will take half of the sample size to train for computation convenience. Figure 21 demonstrates the change of test accuracy based on the variation of the *nTree.* We observe the test accuracy is pretty stable fluctuates around 0.925 and that there is a peak of the test accuracy when the *nTree* equals 400. In addition, we need to check the time complexity for the *nTree*. If the time complexity for the *nTree* is costly, we do not want to use high *nTree* values even if it has better results because it may be too time consuming. Nevertheless, in our case see that the time complexity is also roughly linear in *nTree* (Figure 22), so it is not a big deal to choose a higher value if necessary (in our case, again, 400 is best)

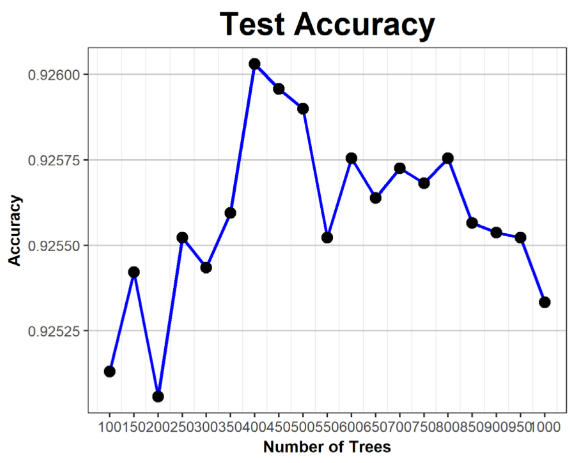


Figure 21

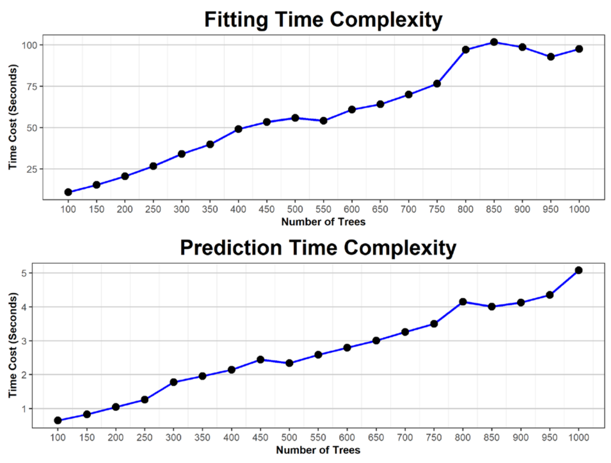


Figure 22

(b) In this section, we want to discover whether the misclassification errors appear in any patterns or not. We first map the classified points that are incorrect on the test set. As shown in Figure 23, we see that the misclassifications do not happen randomly (as there are clusters of red points). This indicates that potential improvements can be made for our classification method, which we will explore further in 4c. Figure 23 displays that there is an obvious cluster of misclassifications in the top middle area, we our method misclassified the clear surfaces as clouds. We further analyze this misclassification pattern by plotting the features with correct and incorrect classifications that overlie on the histograms of the correct classifications (Expert Label). As shown in Figure 24, we notice that NDAI is the major reason of misclassifications. There is an interesting bi-modal pattern in NDAI where almost all the misclassification points applying a cluster with higher values centered at 2, whereas the correct cluster centering is -1. This made NDAI a very important feature in classifying the cloud areas since they are centered nicely around 2. However, the cloud-free points have a thin long right tail distribution that overlaps this range. This may help to explain the reason why misclassifications occurred, since the method basically “sacrificed” these points and classified the vast majority of the large NDAI points as clouds.

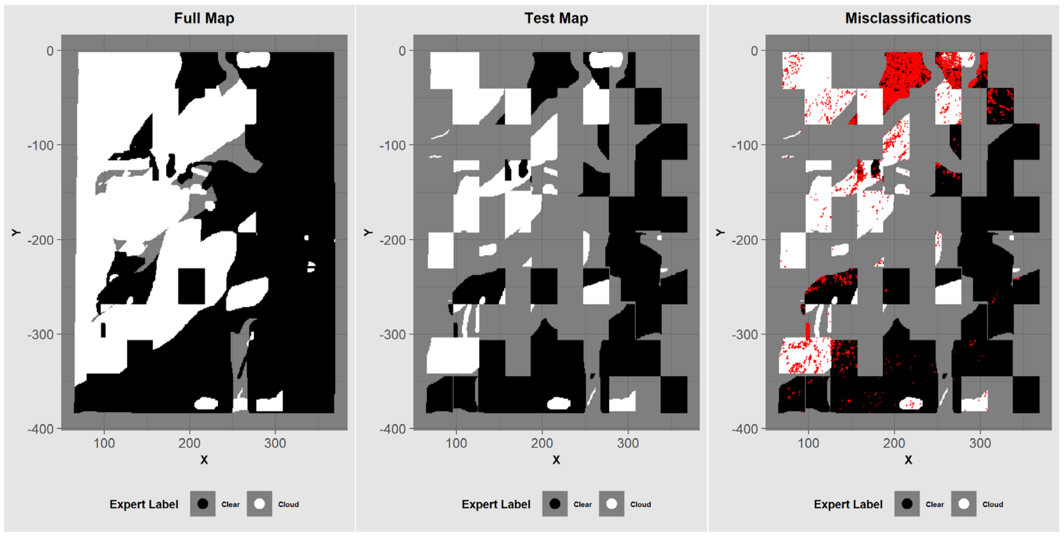


Figure 23

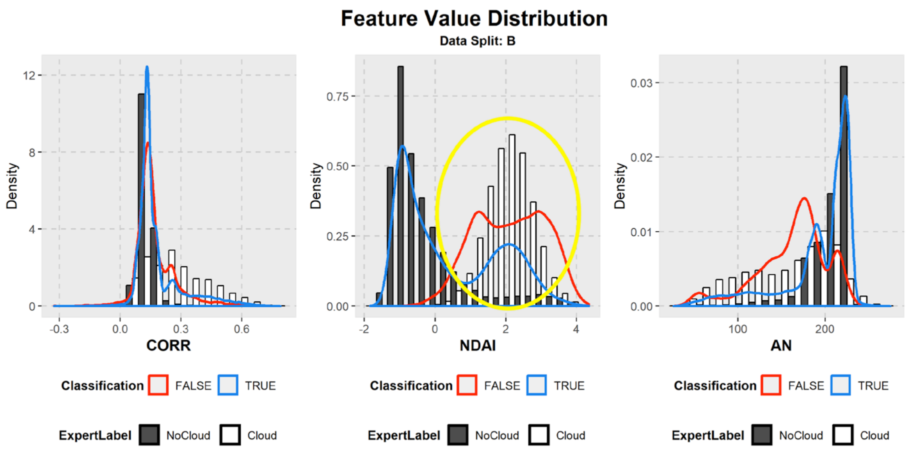


Figure 24

(c) Based on 4a and 4b, we discover that NDAI caused the major problem for classification. Our model incorrectly classified the points in the top middle area because the algorithm does not have enough information to distinguish between the two classes (cloud & clear) in the range of NDAI equals 0 to NDAI equals 4 (Figure 25). In other words, the number of cloud surfaces dominates the number of clear surfaces in the range of NDAI = 0 to 4, and we do not have strong differentiating features, causing the false positives for those cloud-free points. To address this issue, we want to find some features that can differentiate between the two classes in this specific range. After we check all other features, we select AF and SD to add in our model since these two features have better classification performance in the range of NDAI equals 0 to NDAI equals 4 (Figure 26). As a result, shown in Figure 27, our test error decreases from 0.072 to 0.061.

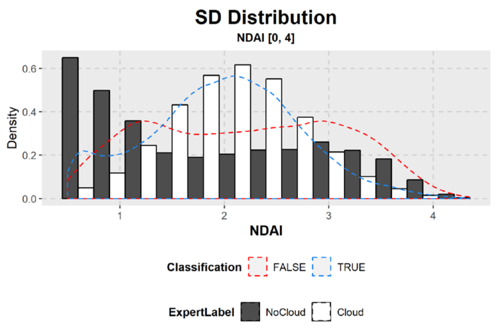


Figure 25

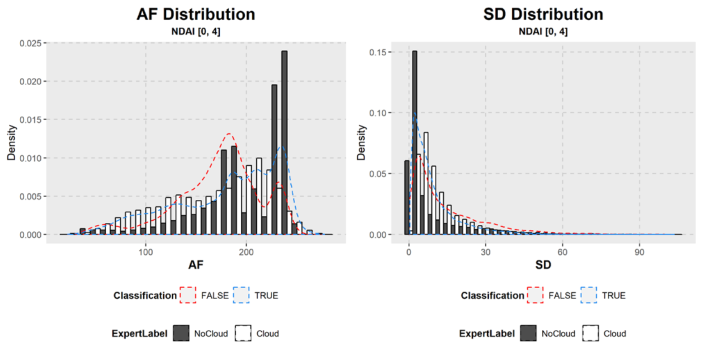
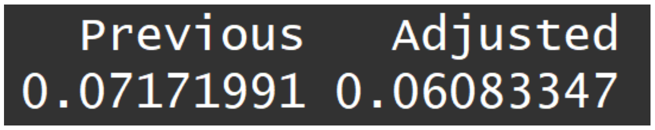


Figure 26



```
   Previous    Adjusted
0.07171991 0.06083347
```

Figure 27

Since we do not use X and Y coordinates to make predictions, we expect our model to work fairly well with incoming data. However, because of the data splitting method we choose, we assume that the training data is similar to the test data. Consequently, in future predictions, our prediction accuracy will fluctuate due to geographic dependence, even if we are not using the exact location in our training set. When we have the information around the prediction points, the model will give us a higher prediction accuracy; whereas if we do not have data information near the unknown points, we will get a lower prediction accuracy.

(d) In this section, we also take a look at data splitting method A to test the model convergence, parameter estimation, and misclassification pattern. Figure 28 demonstrates the test accuracy graph that corresponds to the train sample size. We can see that after we change the data splitting method, the test accuracy increases in general and it converges when the sample size is larger than 80,000. However, the variance of test accuracy also increases. This means that the test accuracy of splitting method A is more volatile than method B. As shown in Figure 29, the time complexity of the sample size has a similar graph with the time complexity graph of splitting method B. This is expected because the fitting time depends on the algorithm, not split method.

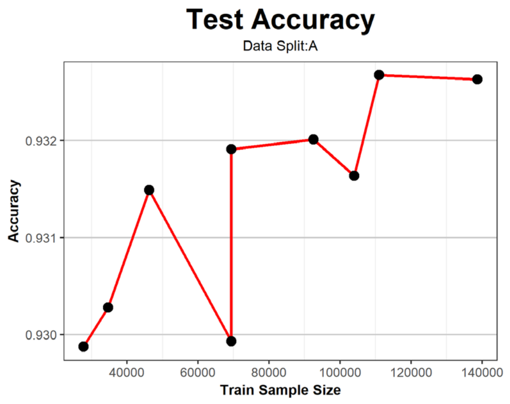


Figure 28

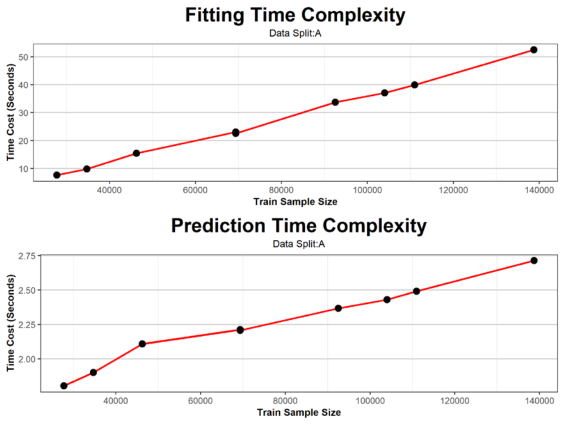


Figure 29

As for the test accuracy based on parameters, as shown in Figure 30, we attain the highest test accuracy when *nTree* equals 850, which is higher than the *ntree* of splitting method B with the highest accuracy. Similar to the convergence, the change of test accuracy based on *ntree* is also less stable than data splitting method B. These two results indict that splitting method A is highly disadvantaged concerning stability. In addition, there is not a significant difference between the two splitting methods' fitting time regarding the variation of parameters (Figure 31). Again, this is expected because we used the same model.

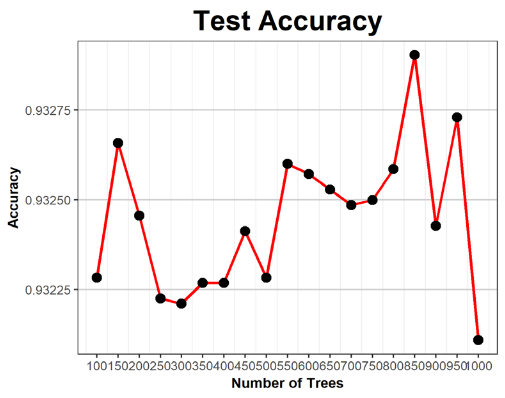


Figure 30

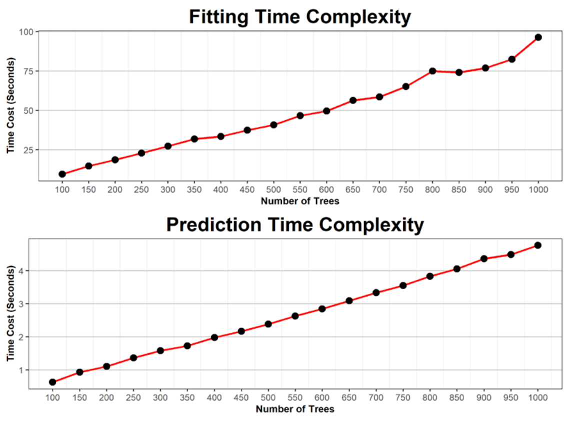


Figure 31

As for the misclassifications, as shown in Figure 32, there is an unclear pattern. These incorrectly classified points are spread out across the whole test map. This is unsurprising because we break the data locality in data splitting method A. But considering that the training set has points very close to the test set, this misclassification rate is not very ideal, and it definitely will perform less nicely when we add future data in.

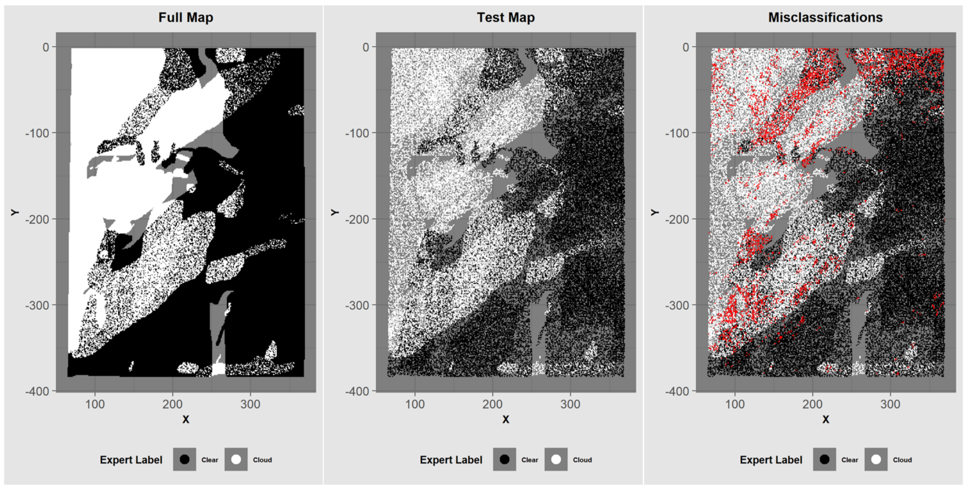


Figure 32

(e) In conclusion, both splitting methods have similar time complexities regarding the sample size and parameter, because we use the same algorithm. Even though splitting method A provides slightly higher test accuracy than data splitting method B, we still prefer data splitting method B because the accuracy for its stability. Furthermore, data splitting method B shows a more apparent misclassification pattern than data splitting method A. This is because the purpose of data splitting method B is to keep the locality. For example, if we assign a clear grid to the test set, then the algorithm may wrongly classify all the points inside that grid; whereas, in data splitting method A, the random assignment inside each grid helps to avoid this problem. Nevertheless, in real life, data splitting method A is unrealistic because we won't have the label information for points that are very close to the unknown points.
From part b, we see how important domain knowledge is. If one blindly tries to increase test accuracy without understanding what is causing the misclassification, then he or she might miss a huge chunk of the points. It would be a lot more helpful if we have expert who understands why cloud-free points have a thin, long right tail in NDAI, and even potentially explain what makes the top-middle section of the map special. This will significantly increase the accuracy of the classification algorithm, and many data analysts tend to overlook its impact.

Reproducibility

GitHub link: https://github.com/peterwangshichun/Stat154Project2.git

Contribution

Yue Chen: Data Collection and Exploration. Preparation.
Shichun Wang: Modeling. Diagnostics. Reproducibility.