Project 2 Cloud Data Classification

Junfang Jiang, Kaiqian Zhu
SID 26159093, 26590233

1. Data Collection and Exploration

1.1 Introduction

Arctic warming is always a crucial problem regarding global warming. The systematic study of it requires accurate Arctic-wide measurements of the cloud coverage which is especially challenging since the image of the cloud is similar to the snow-covered surface. The goal of this study is to build operational cloud detection algorithms that efficiently process the gigantic amount of MISR image data without expert labeling.

The cloud image data is collected by MISR, a satellite consisting of nine cameras which view the Arctic cloud scenes at different angles in four spectral bands. In this study, the paper authors use the images collected from the four forward facing camera and one camera in the nadir direction. To measure the performance of the proposed algorithm, one of the authors hand-label each pixel with 1 and –1 indicating cloud and no-cloud respectively.

The authors further design three physical features: CORR, the correlation of MISR images of the same scene from different MISR viewing direction; SD, the standard deviation of MISR camera (An) pixel values; NDAI, a normalized difference angular index that characterizes the changes in the scene with changes in MISR viewing direction.

The study shows that CORR, SD_An, and NDAI contains sufficient information to detect cloud and with ELCM algorithm, the authors are able to process and label the massive MISR data stream online. The results of ELCM are then used to train QDA for probability labels for partly cloudy scenes. The study demonstrates the potentials of statistician working directly in the nuts and bolts of data processing. Moreover, this study shows the power of statistical thinking and how statistics can make a huge contribution to solve modern scientific problems.

1. 2. Data Collection Summary

The data contains three images from MISR and each of them contains multiple rows of observations encoded as 11 features, including the x, y geo-coordinates and expert labels (+1 for cloudy, -1 for cloud-free and 0 for unlabeled). In order to discover any pattern or discrepancy within the data contained in three images, the following analysis is conducted.

First, we are going to analyze the distribution of the three classes (cloudy, cloud-free and unlabeled). Table 1 shows the computed percentage of pixels that each of the three classes takes in the three images.

Table 1: Percentage of pixels for the different classes

	Total Pixels	**Cloud (label = 1)**	**Cloud Free (label = -1)**	**Unlabeled (label = 0)**
Image 1	115229	17.76%	43.78%	38.46%
Image 2	115110	34.11%	37.25%	28.64%
Image 3	115217	18.44%	29.29%	52.27%
Total	345556	23.44%	36.78%	39.79%

From Table 1 above, we can see the observations of three classes are unevenly distributed among images. For example, Image 1 and 2 has a relatively low percentage of cloudy labels and Image 3 contains a relatively high percentage of unlabeled observations. We may further discover this pattern in the diagram below.

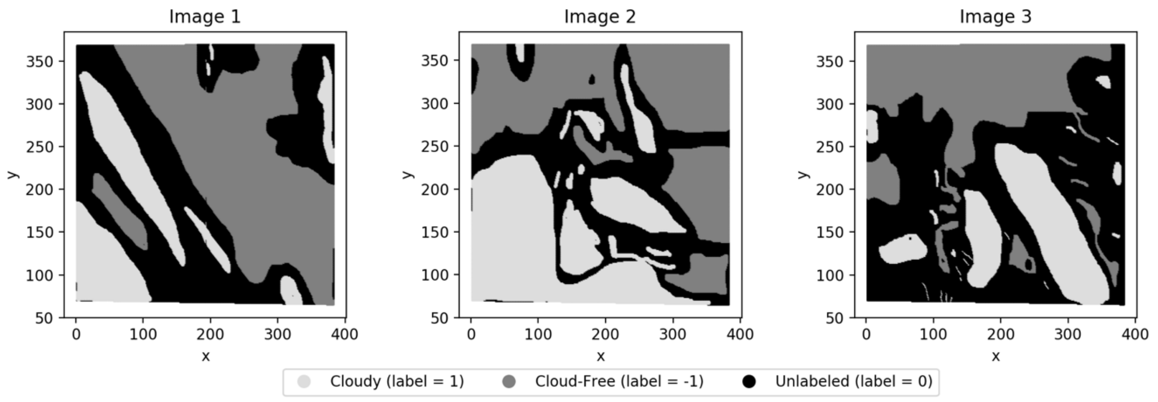

Figure 1: Graphical analysis on distribution of the three type of labels

In Figure 1 above, all observations are plotted in scatter plots based on the corresponding x and y coordinates. The three classes are also plotted in different colors (light-grey for cloudy, dark-grey for cloud-free and black for unlabeled) so that we can evaluate spatial patterns of the given data. We may first confirm the discovery we made in Table 1 that the percentages of each class are significantly different in each diagram. In addition, we can also see that cloud-free pixels concentrate on the top-right of Image 1 and 2 and top-left of Image 3 while cloudy pixels concentrate on the opposite side of cloud-free pixels. Pixels of the same class are generally clustered together, which means we should not make the assumption of independent and identical distribution for observations (at least geographically) since the probability of seeing pixels of a certain class is significantly higher if we know there are other pixels of the same class nearby.

Then, we are going to evaluate the distribution of all other features.

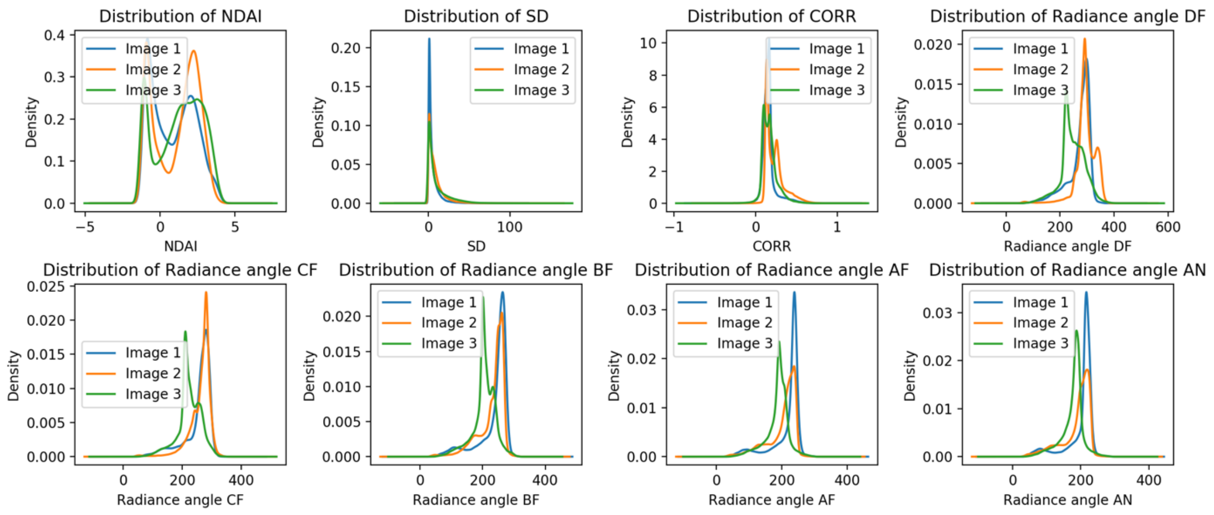

Figure 2: Distribution of different features in three images

We have plotted the overlay distribution of all features (except x, y coordinates, and expert labels) for all three images. From Figure 2, we can see that all features have roughly the same distribution in all three images. In addition, all of them mostly follow the symmetric normal distribution except for NDAI, which follows a bimodal distribution as shown above.

1.3 Exploratory data analysis

In Figure 3, we generated pairwise scatter plots of all features. Notice that the scatter plots of the first three features, (NDAI, SD, CORR) show low to no correlation between the pairs of features in the plots (the data points distributed randomly). However, the scatter plots between the radiance angle features show relatively high correlation as the data points scattered around a positive slope straight line. For the linear model, the high correlation between two pair of features means that we can use a linear combination of one feature to represent the other.

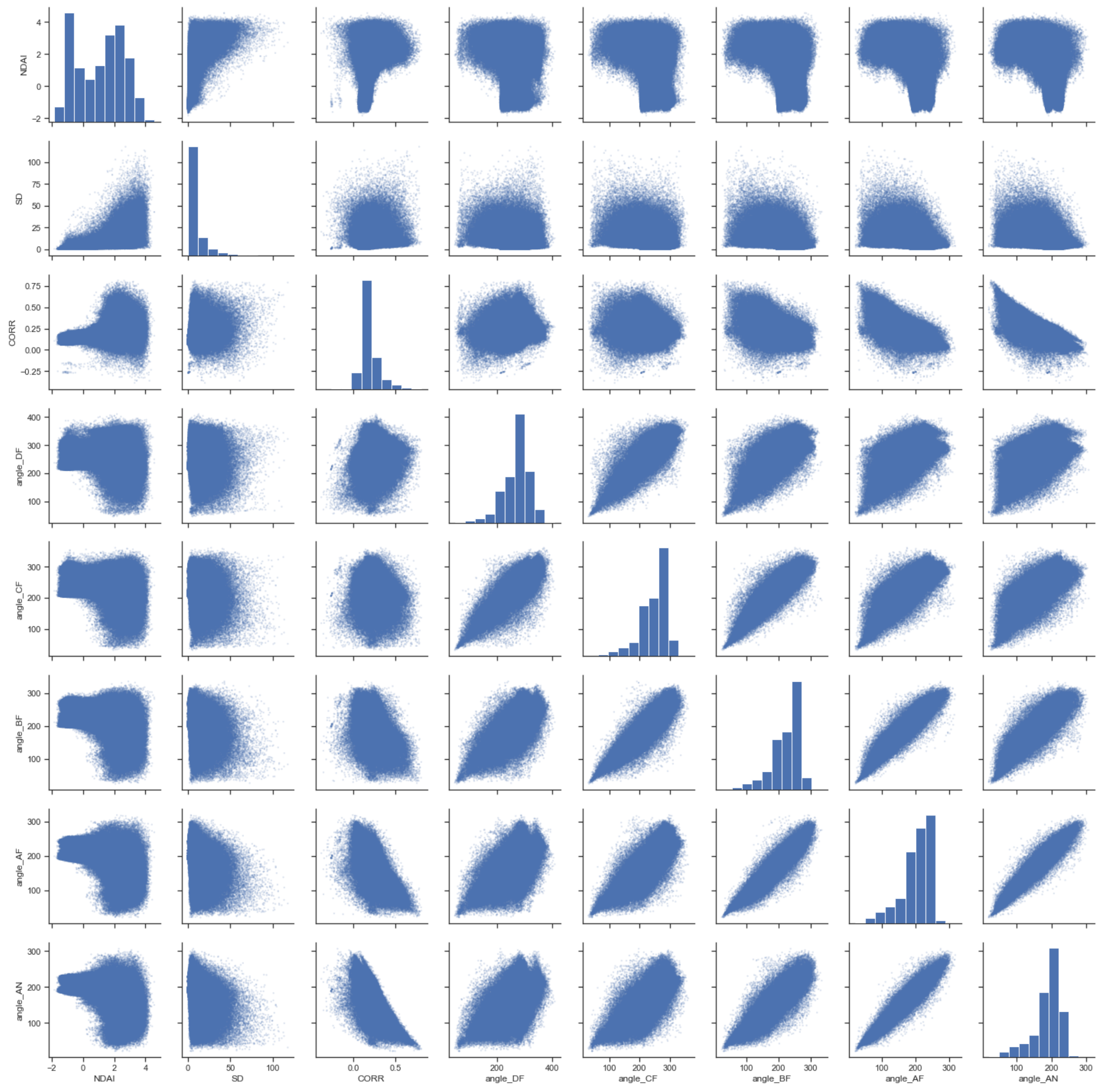

Figure 3: Pairwise scatter splots between features

Then, for each of the eight features, we generate side-by-side box plots of no-cloud and cloud data as shown in Figure 4. Notice that the first three features, NDAI, SD, CORR, the side by side box plots show different means and distribution between the two labels. However, the side-by-side box plots of the radiance angle features indicate a small difference in mean and distribution between the two labels. Thus, we can conclude that the first three features may be more useful in models that we will create later.

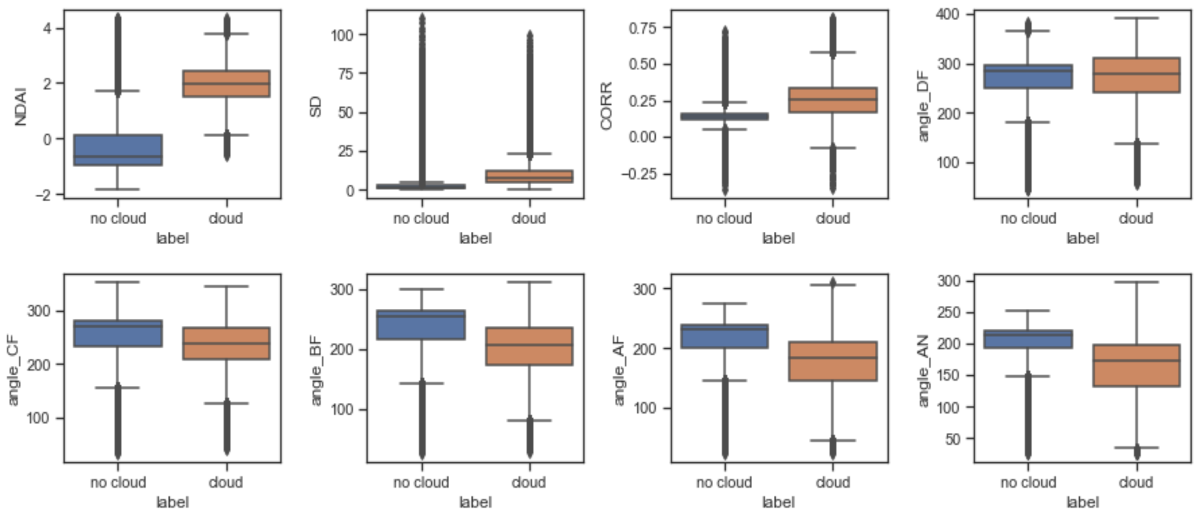

Figure 4: Side-by-side boxplots between the two classes for all features

2. Data Preparation

2.1 Split Data

In order to build our classification model, we will need to split the given data into training, validation and test sets so that we can compare and evaluate different classification models. Due to the fact that data is not independent and identically distributed, the native random split will cause the problem of different distributions among different data sets (i.e., the data used for training may be different from the test/validation dataset which represent the future real-world data for prediction.) and therefore test accuracy and validation accuracy cannot correctly reflect the performance of the model in real-world and goodness of the models respectively.

In this section, we will introduce two non-trivial methods to split the data while preserving the same distribution within each data set. In both approaches described below, we are going to split under the ratio of 6:2:2 (train v.s. validation v.s. test).

2.1.1 Split each class separately

In the first approach, we will split the data separately within each of the three classes, and then combine the three obtained training, validation and test sets (in total 9 sets) from each class into the final training, validation and test sets. By this way, we guarantee all sets have the same distribution because each class is split separately under the same ratio. In addition, when future data comes in, we can again split the new images by individual classes and preserve the same distribution within each set.

2.1.2 Split by down-sampling images

In the second approach, we will use the fact that down-sampling an image won't change the pattern or distribution on that image. In another word, for example, if we want to split 25% of data from the original image, we can simply choose 1 pixel out of every 2x2 block (block width and height equal to the square root of inverse of split ratio) to obtain a lower resolution image which has the same pattern/distribution as the original image. All images will be processed in sequential order and all pixels in the lower resolution images will be used as validation and test data sets.

2.2 Model Baseline

In this section, we will use a trivial classifier that sets all labels to -1 (cloud-free) the validation set and the test set to construct a baseline accuracy for all our models. All future models should at least perform better in accuracy comparing to the baseline to indicate the models are better than random guesses. We should expect our baseline accuracy to be equal to the proportion of cloud-free observations in validation and test sets. In other words, in scenarios with a high proportion of cloud-free observations, the average accuracy of the trivial classifier should also be higher.

For splitting algorithm described in section 2.1.1, the trivial classifier above achieves an accuracy rate of 36.78% on both validation set and test set if preserving all unlabeled data, or 61.08% if only using labeled data. The result matches our expectation since it equals the percentage of cloud-free observation in the data set.

For splitting algorithm described in section 2.1.2, the trivial classifier above achieves an accuracy rate of 36.69% on the validation set and 36.52% on test set if preserving all unlabeled data, or 61.07% on the validation set and 60.68% on test set if only using labeled data. Again, we see a similar accuracy because each set has roughly the same distribution of each class.

2.3 First order importance

In order to find the first order importance of the features, we decide to first apply principal component analysis to find the first three principal components and analyze the contribution of the original features to the three principal components.

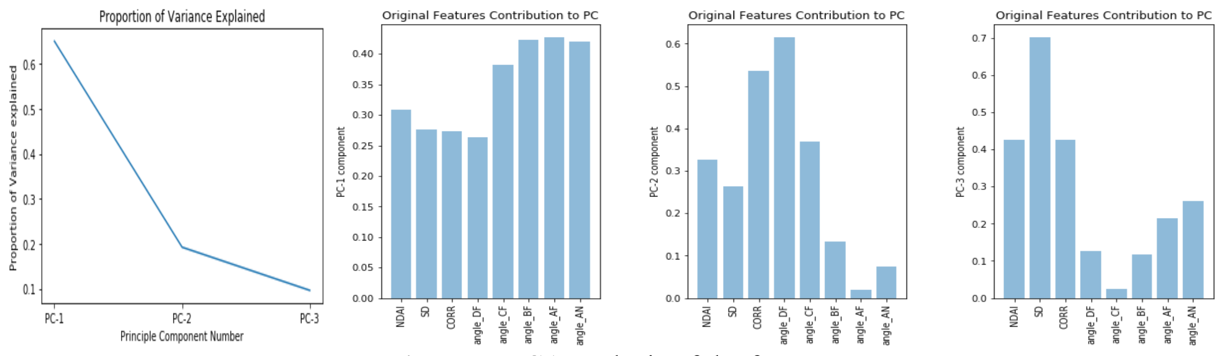

Figure 5: PCA analysis of the features

The PCA shows that the first three principal components capture over 93% variance of the data and each PC explained a proportion of variance indicated by Figure 5. The bar plots in Figure 5, describes the contribution of each principle components which indicates the importance of them in capturing variance. We see that all original features contribute equally to the first principle component. The CORR and angle_DF contribute significantly to the second principal component and the NDAI, SD and CORR contribute noticeably more than the other features to the third principal component. Thus, we rank NDAI, SD, CORR and angle_DF higher than other features.

Then, we further investigate the correlation between each feature and the label. From Figure 6, we notice that even though angle_DF contribute significantly to the variance of the dataset, it has really low correlation with the label. Also, we see that the features NDAI, SD and CORR all have relatively high correlations with the label. Since NDAI has the highest correlation with the label, we plot the correlation between other features with NDAI (Figure 6) and find that the radiance angle features we exclude earlier have a relatively high correlation with NDAI, and thus we can safely exclude them. Therefore, we conclude that the features NDAI, SD and CORR are more important than the other radiance angle features.

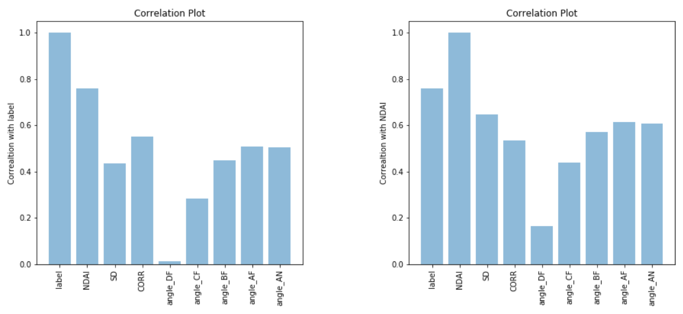

Figure 6: Correlation between features

2.4 Generic Cross Validation Function

Cross-validation is an important technique in machine learning for evaluating the goodness of models without explicitly splitting a validation set. Therefore, we will implement generic cross-validation (CV) function that takes a generic classifier, training features, training labels, number of folds and a loss function as inputs and outputs the K-fold CV loss on the training set. In the cross-validation function, we use the same data splitting techniques as described in section 2.1 to create folds. This function will be re-used in model evaluations in later sections. The source code of the function is included in section 5.

3. Modeling

3.1 Classification Models

In this section, we are going to try four different classification models and assess their performance using 5-fold cross-validation implemented in section 2.4. In addition, we have removed all unlabeled data during training since predicting an observation to be "unlabeled" is not useful in the classification problem we face, and data splitting technique described in 2.1.1 is used for creating training set and cross-validation folds.

3.1.1 Logistic Regression

We first attempt logistic regression as our classification model. We take the four fundamental assumptions of logistic regression into consideration. First, the label should be binary and thus we removed all the records that labeled 0 (unlabeled). Second, the observations should be independent of each other. Notice that each record represents a single pixel in the original image, and we know the cloud pixels maybe close to each other. Therefore, we removed the x and y coordination features which are not very helpful in our model to decrease the spatial dependence among observations. Third, the correlation among features should be low or no multicollinearity. In the EDA section, we discovered that the correlations between radiance angle features are relatively high, so we try to exclude them in model 1. Finally, large sample size is needed for logistic regression and we know we have a massive amount of MISR data.

In Table 2, we report the accuracies folds and test accuracies. Notice that the first model which only include the NDAI, SD and CORR features, achieves an overall higher accuracy than the model 2 which include additional five radiance angle features. This result might be caused by the multicollinearity of the five radiance angle features which violates the assumption of logistic regression. Also, we notice that there is no noticeable difference between the two methods.

Table 2: Cross-validation fold accuracies for logistic regression models

	Split Method	Test Accuracy	Mean CV Accuracy	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Model 1	1	89.26%	89.27%	89.50%	89.39%	89.28%	89.88%	89.32%
	2	89.16%	89.28%	89.26%	89.29%	89.23%	89.40%	89.22%
Model 2	1	89.05%	89.16%	89.22%	89.16%	89.21%	89.28%	88.91%
	2	89.21%	89.21%	89.25%	89.17%	89.19%	89.31%	89.13%

(Model 1 uses 3 features: NDAI, SD, CORR; Model 2 uses 8 features: NDAI, SD, CORR, plus 5 radiance angles; For splitting methods, see section 2.1)

3.1.2 Random Forest Classifier (with AdaBoost)

Then, we try the random forest classifier with AdaBoost. Since it does not use any probabilistic model, the only assumption that we need to take care of is the sample in bootstrap aggregation should be representative. We rely on the implementation of AdaBoostClassifier from sklearn, which by searching through its documentation, we have confirmed the algorithm does satisfy the assumptions.

In addition, there are two hyperparameters in the AdaBoost algorithm: the number of trees in the forest and the learning rate. By using the GridSearchCV from sklearn to iteratively search through a grid of all potential hyperparameter values with cross-validation, the optimal value of hyperparameters are obtained where the number of trees is set to be 125 and the learning rate is set to be 1.25.

Table 3: Cross-validation fold accuracies for AdaBoost models

	Split Method	Test Accuracy	Mean CV Accuracy	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Model 1	1	91.41%	91.38%	91.54%	91.59%	91.43%	91.04%	91.33%
	2	91.35%	91.41%	91.25%	91.56%	91.38%	91.59%	91.26%
Model 2	1	92.13%	91.99%	91.81%	92.24%	92.04%	91.79%	92.08%
	2	91.91%	91.95%	91.84%	92.17%	91.87%	92.13%	91.73%

(Model 1 uses 3 features: NDAI, SD, CORR; Model 2 uses 8 features: NDAI, SD, CORR, plus 5 radiance angles; For splitting methods, see section 2.1)

The accuracies we get for random forest classifier in Table 3 show similar results to that of logistic regression: the simpler model 2 with only NDAI, SD, CORR features achieves a slightly higher accuracy of all kind than the model 1 with all features. Also, there is no noticeable difference in accuracy rates between two splitting methods for random forest classifier too.

3.1.3 K-nearest neighbors

K-nearest neighbors (KNN) algorithm is also used as a classification model to predict our data. We again remove all unlabeled data and convert the problem into a binary classification problem. Standard KNN algorithm is used, except our model will also output probability estimates of the observations so that we may use the ROC curve to choose the optimal threshold in the next section. However, for now, the threshold is set to be 0.5 to compute the accuracy of the model.

Since KNN is a non-parametric model, it does not contain any probabilistic model and thus has no assumption on the underlying data distribution. However, since KNN relies on L2 distance, we normalize all features before pipelining into our KNN model. In addition, the algorithm contains one hyperparameter K, and it is also tuned with GridSearchCV from sklearn by testing potential K values ranging from 10 to 30. The resultant optimal value of K is found to be 35.

Table 4: Cross-validation fold accuracies for KNN models

	Split Method	Test Accuracy	Mean CV Accuracy	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Model 1	1	92.07%	92.04%	92.14%	92.10%	92.19%	91.88%	91.91%
	2	92.07%	92.09%	91.92%	92.20%	92.08%	92.28%	91.99%
Model 2	1	95.29%	95.26%	95.38%	95.26%	95.20%	95.22%	95.23%
	2	95.40%	95.32%	95.31%	95.45%	95.46%	95.40%	94.97%

(Model 1 uses 3 features: NDAI, SD, CORR; Model 2 uses 8 features: NDAI, SD, CORR, plus 5 radiance angles; For splitting methods, see section 2.1)

Unlike previous two classification models, KNN works better with more features in our case as shown in Table 4. We see that the model 2 with all eight features achieves noticeably higher accuracies than model 1 which only includes the NDAI, SD, and CORR features. Similar to previous models, there is no noticeable difference in accuracy rates between two splitting methods for KNN.

3.1.4 QDA

Lastly, we are going to attempt Quadratic Discriminant Analysis (QDA) as the classification model. The model is quite straightforward without any hyper-parameter. QDA model assumes normal distribution in data, which we can see it is satisfied by using Figure 2. QDA has no assumption regarding data variances.

Table 5: Cross-validation fold accuracies for QDA models

	Split Method	Test Accuracy	Mean CV Accuracy	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Model 1	1	89.69%	89.67%	89.86%	89.59%	89.78%	89.29%	89.80%
	2	89.63%	89.69%	89.62%	89.82%	89.63%	89.74%	89.63%
Model 2	1	89.10%	89.02%	89.00%	89.16%	88.96%	88.93%	89.06%
	2	88.98%	89.11%	88.93%	89.22%	89.07%	89.21%	89.10%

(Model 1 uses 3 features: NDAI, SD, CORR; Model 2 uses 8 features: NDAI, SD, CORR, plus 5 radiance angles; For splitting methods, see section 2.1)

Similar to method 1 and 2, more features actually result a worse accuracy rate due to potential random noise and overfitting. In addition, split method 1 is better than splitting method 2 in general as shown in Table 5.

3.2 ROC Curve Analysis

Figure 7 on the right is the ROC curve for all four classification methods attempted above (only the model with the highest test accuracy is selected). All models are now outputting probability estimates instead of binary prediction labels. The optimal cut off value (labeled in red) is chosen to be the point where the slope is 1 on the ROC curve. This is because beyond this point, per unit increase in true positive rate has a larger increase in false positive rate, which is undesirable. The cut-off value chosen here should serve as an improvement based on the models trained by accuracy score in the previous section. The optimal cut-off values corresponding to the optimal are listed in Table 6 and can be used as improvements on all previous models (predicting cloudy if the probability is larger than the new cut-off value instead of 0.5).

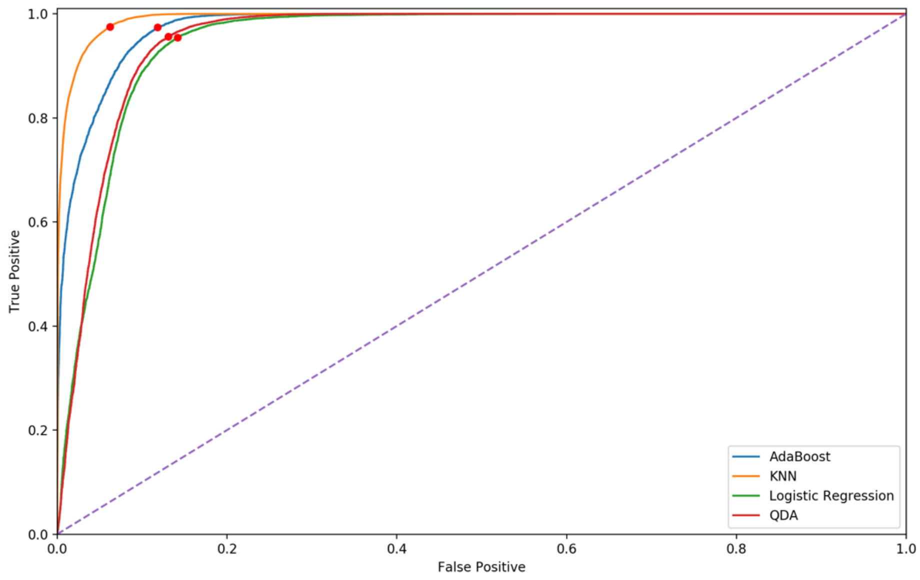

Figure 7: ROC curve for classification models

Table 6: Optimal cut-off values for classification models

	Logistic Regression	AdaBoost	KNN	QDA
Cut-off Value	0.2940	0.3714	0.4984	0.1513

(Models should predict cloudy when the probability of cloudy is larger than the cut-off value)

We may further compare the models using the area under the curve (AUC) on the ROC curve. A larger value of AUC means the observations are more separable under the specific model, which from the diagram above, we can see the KNN model has the largest AUC and therefore outperform the other models. This is consistent with our previous result where we use accuracy rate as the criteria to compare among models.

3.3 Further Assessment of Model Performance

In this section, we will first apply the new optimal cut-off thresholds from 3.2 and then further investigate the performance of the classification models by using multiple metrics other than accuracy rate and separately assessing in all classes.

Table 7(a): Classification report of Logistic Regression and KNN

Logistic Regression					KNN				
	precision	**recall**	**f1-score**	**support**		**precision**	**recall**	**f1-score**	**support**
Cloudless	0.97	0.86	0.91	25416	**Cloudless**	0.98	0.94	0.96	25416
Cloudy	0.81	0.95	0.88	16197	**Cloudy**	0.91	0.98	0.94	16197
micro avg	0.9	0.9	0.9	41613	**micro avg**	0.95	0.95	0.95	41613
macro avg	0.89	0.91	0.89	41613	**macro avg**	0.95	0.96	0.95	41613
weighted avg	0.91	0.9	0.9	41613	**weighted avg**	0.95	0.95	0.95	41613

Table 7(b): Classification report of AdaBoost and QDA

Random Forest (with AdaBoost)					QDA				
	precision	**recall**	**f1-score**	**support**		**precision**	**recall**	**f1-score**	**support**
Cloudless	0.98	0.88	0.93	25416	**Cloudless**	0.97	0.87	0.92	25416
Cloudy	0.84	0.97	0.9	16197	**Cloudy**	0.82	0.96	0.88	16197
micro avg	0.92	0.92	0.92	41613	**micro avg**	0.90	0.90	0.90	41613
macro avg	0.91	0.93	0.92	41613	**macro avg**	0.90	0.91	0.90	41613
weighted avg	0.93	0.92	0.92	41613	**weighted avg**	0.91	0.90	0.90	41613

In Table 7, all four models are assessed by precision (true positive divided by actual results), recall (true positive divided by predicted results), and f1-score (a measurement combines precision and recall) in both classes. We may first notice that all models have observable improvements after applying the new cut-off thresholds. In addition, we can see all models have a significantly higher precision in predicting cloudless observations (around 97 to 98%) than cloudy observations (around 80% to 90%). Among them, KNN works the best in all type of labels in terms of both precision and recall.

4. Diagnostics

In the following section, we are going to conduct a further assessment on the performance of the best classification model, the KNN model, obtained from the previous analysis.

4.1 In-depth analysis on the performance of KNN

In previous sections, we have evaluated the model performance in terms of accuracy and precision. In this section, we are going to shift our focus, and assess the performance on the issues might arise in a real production environment after deployment. Unlike other supervised models we tried above, KNN is a non-parametric classification model and its prediction time increases as the training dataset size increases. Therefore, considering the massive amount of MISR data, we want to analyze how the prediction time and the number of training data points are related and also how accuracy rate of the model converges with an increasing number of data points.

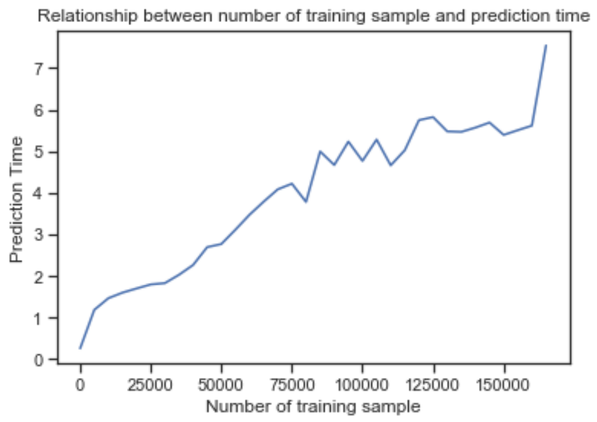

Figure 8(a): Prediction time versus training set size

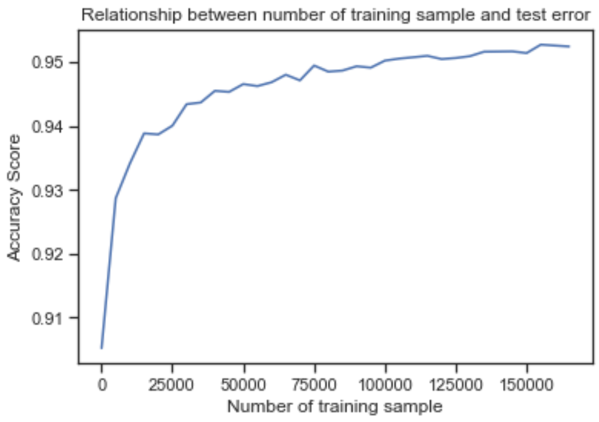

Figure 8(b): Prediction time versus training set size

From Figure 8(a) above, we see that the empirical prediction time grows linearly as we increase the number of training data points. Therefore, due to the tremendous size of MISR data (345,556 observations for only

3 images) and new images being constantly flowing in, our KNN model will suffer from excessively long prediction time despite extremely high accuracy rate if we use all previously labeled data as the training set. Fortunately, from Figure 8(b) above, we observe that the accuracy converges to around 95% with only around 100,000 data points, which is not so large comparing to the size of the entire MISR dataset.

4.2 Patterns in Misclassification

Although the KNN model achieves over 95% of accuracy rate, we still want to analyze the problem that makes our KNN model misclassify in order to further improve the model. In Figure 9, we plot the misclassified data points on the position (based on x, y coordinates) of the original images.

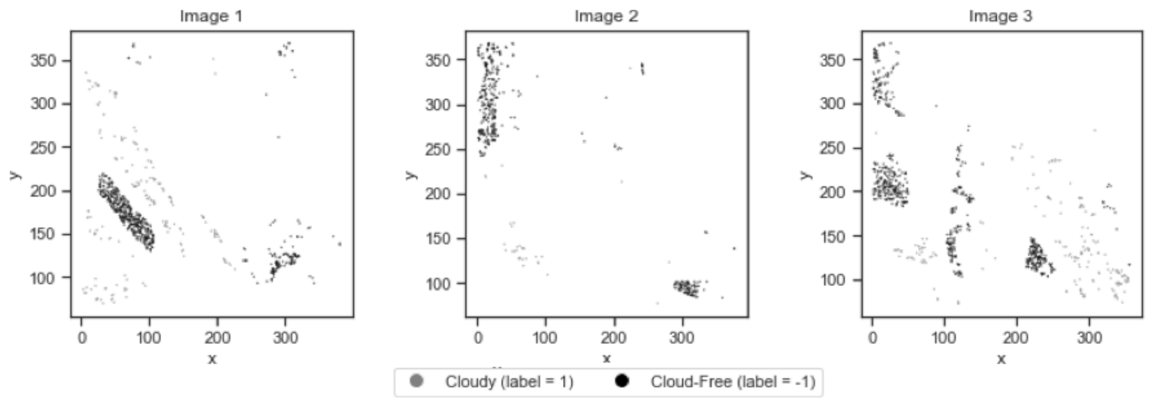

Figure 9: Regions for misclassified observations

By comparing both Figure 1 and Figure 9, we notice that if a data point is in a region that is surrounded by data points that are in the other class, our model will have a relatively higher misclassification rate, for example, the cloud-free region in the bottom-left of Image 1.

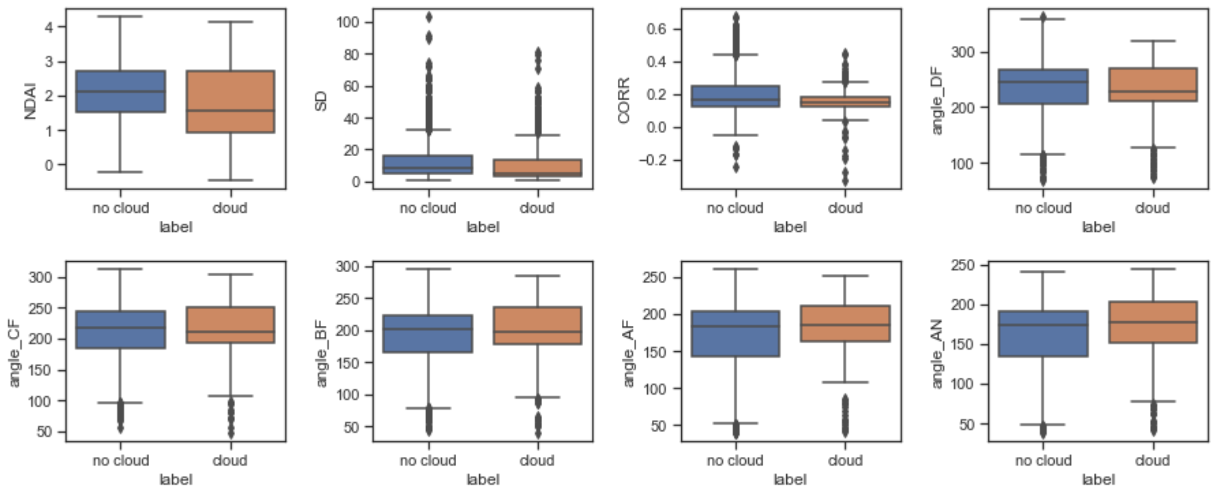

Figure 10: Boxplots of features from misclassified data

Moreover, we generate side-by-side box plots of all misclassified cloud-free and cloudy observations from our KNN model, as shown in Figure 10. We can see that the distribution between the two labels are quite similar for all eight features, meaning our model cannot distinguish the labels easily for these data points given only the existing features.

4.3 Future Improvement

Recall the findings in section 4.1, in which we discover that the test accuracy score converges to 95% with only a hundred thousand training data points. For future labeled training data, we can simply replace the old data points with the new one under a certain probabilistic model while keeping a fixed size training dataset. One can argue that the model achieves a zero training-time and a constant prediction-time. Moreover, as we show in section 4.2, the current set of features won't help our model correctly label the misclassified data points. In order to further increase the accuracy rate, we need to design new features or augment the existing features with interaction terms and higher degree terms. Doing so allows the KNN model to correctly label these misclassified data points in new dimensions.

As discussed in section 4.3, our KNN model can predict future data without expert-label in constant run time with high accuracy rate compared to other models we have tested.

4.4 Effect of data splitting methods

There is no significant difference in the number of training data points required for the test accuracy to converge between the two splitting methods. Moreover, the region and feature distributions of misclassified data points would not change noticeably when we switch between the data splitting methods.

4.5 Conclusion

In this section, we choose the KNN model to do an in-depth analysis regarding the prediction runtime and the misclassified data points. In section 4.1, we demonstrated that the empirical prediction runtime for KNN increase linearly as the size training dataset increase but we also discover that the test accuracy will converge to over 95% with only one hundred thousand training data points. This discovery allows us to explore the possibility of maintaining a fixed training dataset even with new labeled training data points and can indicate our prediction time to be constant (measuring distance to a fixed number of data points). In our second analysis, we explore the region property and feature distribution of the misclassified data points. We find that the misclassified data points are surrounded by data points of the other label and they have similar distribution among all eight features. Our suggested improvement for these issues is to design new features or augment the existing features. In addition, we find no significant effect of data splitting methods on our analysis results.

5. Reproducibility

All resources that are required for reproduction are publicly available at https://github.com/junf97/stat154-project2. You may find the Word format of this report and all Python codes associated with each section to generate the diagrams, tables, etc. that are included in this report. For more information, please refer to the README of the repository.

6. Acknowledgment

This project including all associated parts (e.g., code, report, etc.) is completed and published by Junfang Jiang (junfang@berkeley.edu) and Kaiqian Zhu (tim3212008@berkeley.edu) at UC Berkeley. Both members contributed equally to all works. The Python scikit-learn library is heavily used in this project, which provides implementations of various classification models and machine learning functionalities. We would like to also express our deepest appreciation to all instructors and GSIs, who have provided us great help throughout the entire project.