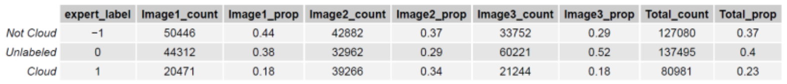
# MISR Cloud Classification

Catherine Wang 3031990027
Yijin Wang 25544087

Github: https://github.com/catherinewang1/Stat154-Project2

## 1. Data Collection and Exploration

**(a)** The purpose of the paper "Daytime Arctic Cloud Detection Based on Multi-Angle Satellite Data with Case Studies" by Tao Shi, Bin Yu, et al, is to detect clouds in the Arctic region using MISR imagery by identifying cloud-free surface pixels, different from the previous MISR algorithm which identifies cloudy pixels. This is important in studying the effect of global climate change and atmospheric CO2 levels in the Arctic region. The data itself is collected from the MISR onboard the NASA Terra satellite in 1999. There are a total of 9 cameras each from different angles and each detecting 4 spectral bands (blue, green, red, and near-infrared). There are 233 "paths," a strip of the Earth in the North to South direction, each collected every 16 days. Each path is divided into 180 "blocks," and 3 consecutive blocks is defined as a data unit. In addition, each orbit of the Earth is recorded. Each pixel represents a 275x275m region, which means that there is a massive amount of raw data to deal with, which is an issue when trying to compute efficiently. Because the data is so large, only the red radiances from the nadir cameral were transmitted fully at the 275m resolution, and the other colors and cameras were aggregated onboard to 1.1km x 1.1km and then transmitted. From the raw data, 3 computed features used are CORR, SD, and NDAI. CORR captures the surface correlation of images of the same scene from different viewing directions. SD is the standard deviation of the MISR nadir camera pixel values from a scene, and NDAI is the normalized difference angular index that captures the changes in scene due to different viewing directions. In addition, there are the x-y coordinates to locate where the pixel actually is, and `Expert Label` which are labeled by an expert as +1 cloud, -1 not cloud, and 0 unlabeled. The final two algorithms developed in this paper are ELCM (Enhanced linear Correlation Matching) and ELCM-QDA. ELCM labels a 1.1x1.1m pixel clear when (SD < thresholdSD = 2.0) or (CORR > thresholdCORR = .75 and NDAI < thresholdNDAI) where thresholdNDAI is learned adaptively. This result is binary and classified as cloudy or clear; however, ELCM-QDA results in a probability in [0,1] by performing QDA on the result of ELCM (only if the entire data unit doesn't result in one label (either cloudy or clear) being more than 98%). The results of this new algorithm give better agreement rates than other methods such as SDCM, ASCM, and SVM. When compared to each other, ELCM and ELCM-QDA give similar results in agreement rates, but ELCM-QDA was able to give probabilities, and gave probabilities near .5 near cloud-clear borders, a good sign. ELCM-QDA is also more desirable because some pixels may be ambiguous or have varying degrees of cloudiness, so probability is more informative than a binary label. This paper concludes that the 3 features are sufficient to separate clouds to surfaces, the new algorithms are a better classifier than the existing MISR algorithm, the new algorithm is suitable for real-time processing, and the result of ELCM can be used for QDA. This work shows a significant application of statistics to the massive amount of Earth science data, that less sophisticated methods (QDA with just 3 features) give performance comparable to more sophisticated ones, that results are better when statisticians are directly involved from the beginning of the study, and the significance of statistical thinking for modern scientific problems.

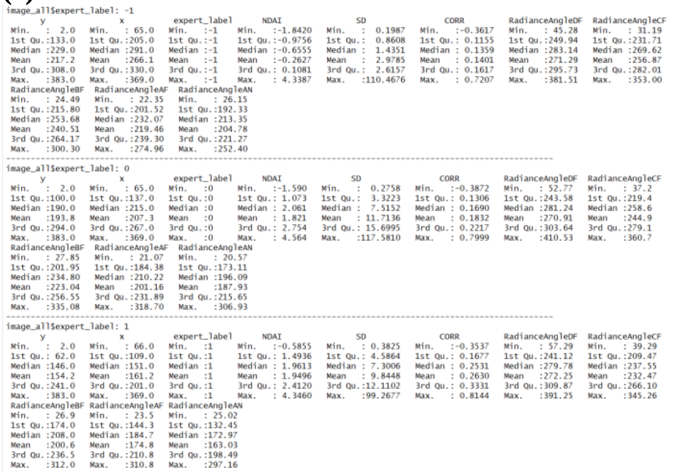


| | expert_label | Image1_count | Image1_prop | Image2_count | Image2_prop | Image3_count | Image3_prop | Total_count | Total_prop |
|---|---|---|---|---|---|---|---|---|---|
| *Not Cloud* | −1 | 50446 | 0.44 | 42882 | 0.37 | 33752 | 0.29 | 127080 | 0.37 |
| *Unlabeled* | 0 | 44312 | 0.38 | 32962 | 0.29 | 60221 | 0.52 | 137495 | 0.4 |
| *Cloud* | 1 | 20471 | 0.18 | 39266 | 0.34 | 21244 | 0.18 | 80981 | 0.23 |

*Figure 1.b.1*

**(b)** Table 1.b.1 summarizes the number of pixels and proportion in each class in each image and in total. We see that in total, there is around 37% not cloud pixels, 40% unlabeled pixels, and 23% cloud pixels. Figure 1.b.2 shows a map of the pixels and their classification on the x y coordinate plane. From the map of the pixels and their labels, we see a trend that pixels in the same region tend to be the same classification. This strongly suggests that the i.i.d. assumption is violated for this dataset because two pixels close together are more likely to have the same classification (ie pixels close together on the x-y plane are highly correlated with each other).

**(c)**



```
image_all$expert_label: -1
       y               x            expert_label      NDAI               SD                 CORR            RadianceAngleDF  RadianceAngleCF
 Min.   :  2.0   Min.   : 65.0   Min.   :-1     Min.   :-1.8420   Min.   :  0.1987   Min.   :-0.3617   Min.   : 45.28   Min.   : 31.19
 1st Qu.:133.0   1st Qu.:205.0   1st Qu.:-1     1st Qu.:-0.9756   1st Qu.:  0.8608   1st Qu.: 0.1155   1st Qu.:249.94   1st Qu.:231.71
 Median :229.0   Median :291.0   Median :-1     Median :-0.6555   Median :  1.4351   Median : 0.1359   Median :283.14   Median :269.62
 Mean   :217.2   Mean   :266.1   Mean   :-1     Mean   :-0.2627   Mean   :  2.9785   Mean   : 0.1401   Mean   :271.29   Mean   :256.87
 3rd Qu.:308.0   3rd Qu.:330.0   3rd Qu.:-1     3rd Qu.: 0.1081   3rd Qu.:  2.6157   3rd Qu.: 0.1617   3rd Qu.:295.73   3rd Qu.:282.01
 Max.   :383.0   Max.   :369.0   Max.   :-1     Max.   : 4.3387   Max.   :110.4676   Max.   : 0.7207   Max.   :381.51   Max.   :353.00
 RadianceAngleBF  RadianceAngleAF  RadianceAngleAN
 Min.   : 24.49   Min.   : 22.35   Min.   : 26.15
 1st Qu.:215.80   1st Qu.:201.52   1st Qu.:192.33
 Median :253.68   Median :232.07   Median :213.35
 Mean   :240.51   Mean   :219.46   Mean   :204.78
 3rd Qu.:264.17   3rd Qu.:239.30   3rd Qu.:221.27
 Max.   :300.30   Max.   :274.96   Max.   :252.40
-----------------------------------------------------------------------------------------------------------
image_all$expert_label: 0
       y               x            expert_label      NDAI             SD                CORR            RadianceAngleDF  RadianceAngleCF
 Min.   :  2.0   Min.   : 65.0   Min.   :0      Min.   :-1.590   Min.   :  0.2758   Min.   :-0.3872   Min.   : 52.77   Min.   : 37.2
 1st Qu.:100.0   1st Qu.:137.0   1st Qu.:0      1st Qu.: 1.073   1st Qu.:  3.3223   1st Qu.: 0.1306   1st Qu.:243.58   1st Qu.:219.4
 Median :190.0   Median :215.0   Median :0      Median : 2.061   Median :  7.5152   Median : 0.1690   Median :281.24   Median :258.6
 Mean   :193.8   Mean   :207.3   Mean   :0      Mean   : 1.821   Mean   : 11.7136   Mean   : 0.1832   Mean   :270.91   Mean   :244.9
 3rd Qu.:294.0   3rd Qu.:267.0   3rd Qu.:0      3rd Qu.: 2.754   3rd Qu.: 15.6995   3rd Qu.: 0.2217   3rd Qu.:303.64   3rd Qu.:279.1
 Max.   :383.0   Max.   :369.0   Max.   :0      Max.   : 4.564   Max.   :117.5810   Max.   : 0.7999   Max.   :410.53   Max.   :360.7
 RadianceAngleBF  RadianceAngleAF  RadianceAngleAN
 Min.   : 27.85   Min.   : 21.07   Min.   : 20.57
 1st Qu.:201.95   1st Qu.:184.38   1st Qu.:173.11
 Median :234.80   Median :210.22   Median :196.09
 Mean   :223.04   Mean   :201.16   Mean   :187.93
 3rd Qu.:256.55   3rd Qu.:231.89   3rd Qu.:215.65
 Max.   :335.08   Max.   :318.70   Max.   :306.93
-----------------------------------------------------------------------------------------------------------
image_all$expert_label: 1
       y               x            expert_label      NDAI              SD                CORR            RadianceAngleDF  RadianceAngleCF
 Min.   :  2.0   Min.   : 66.0   Min.   :1      Min.   :-0.5855   Min.   : 0.3825   Min.   :-0.3537   Min.   : 57.29   Min.   : 39.29
 1st Qu.: 62.0   1st Qu.:109.0   1st Qu.:1      1st Qu.: 1.4936   1st Qu.: 4.5864   1st Qu.: 0.1677   1st Qu.:241.12   1st Qu.:209.47
 Median :146.0   Median :151.0   Median :1      Median : 1.9613   Median : 7.3006   Median : 0.2531   Median :279.78   Median :237.55
 Mean   :154.2   Mean   :161.2   Mean   :1      Mean   : 1.9496   Mean   : 9.8448   Mean   : 0.2630   Mean   :272.25   Mean   :232.47
 3rd Qu.:241.0   3rd Qu.:201.0   3rd Qu.:1      3rd Qu.: 2.4120   3rd Qu.:12.1102   3rd Qu.: 0.3331   3rd Qu.:309.87   3rd Qu.:266.10
 Max.   :383.0   Max.   :369.0   Max.   :1      Max.   : 4.3460   Max.   :99.2677   Max.   : 0.8144   Max.   :391.25   Max.   :345.26
 RadianceAngleBF RadianceAngleAF RadianceAngleAN
 Min.   : 26.9   Min.   : 23.5   Min.   : 25.02
 1st Qu.:174.0   1st Qu.:144.3   1st Qu.:132.45
 Median :208.0   Median :184.7   Median :172.97
 Mean   :200.6   Mean   :174.8   Mean   :163.03
 3rd Qu.:236.5   3rd Qu.:210.8   3rd Qu.:198.49
 Max.   :312.0   Max.   :310.8   Max.   :297.16
```

*Figure 1.c.2*

Figure 1.c.1 shows the pairwise plots between all the 11 variables of 100 randomly chosen points (there are too many in total to reasonably plot all), colored by their expert labels, the fitted histograms, and their correlations. Not surprisingly, we see that there are very strong correlations between the Radiance Angle features, especially if their corresponding cameras are close to each other. (For example, Radiance Angle AF and Radiance Angle AN have a high correlation of .955 and the scatterplot between them is almost exactly a linear line). Although not shown, the same plot applied to image1, image2, and image3 are very similar.

To see the differences between expert_label and other features, we can focus on the third column (the scatterplots between expert_label and all the other features). From this, we see that the distribution of each label has different means and spreads for the features NDAI, SD, CORR, and the 5 radiances. For example, let's focus on the scatterplot between expert_label and NDAI (4 down, 3 right). We see that not cloudy points have a lower mean and medium spread, unlabeled pixels have a middle mean and a much larger spread, and the cloudy points have a higher mean and a similar spread to the not cloudy pixels. This suggests that cloudy points have higher NDAI values, not cloudy points have lower NDAI values, and the unlabeled points are an unknown combination of cloudy and not cloudy points which contributes to a mean in between and a higher spread. This pattern of different cloudy and not cloudy distributions, with unlabeled in between those distributions, is seen also in SD and all the Radiance Angles. We can also focus on any of the scatterplots and look at the distribution of the colors of the dots on the scatterplots because the color denotes the label, and these also show that clouds and non clouds have different distributions in features spaces. For example, look at the scatterplot between NDAI and Radiance Angle CF. We see that the non-cloud points seem to be clustered in the top left corner while the cloud points have both a spread centered around the scatterplot center and a much wider spread of NDAI and CF values. The same results are seen in Figure 3.a.4 which shows a larger pair plot between the features colored by their classification and showing just the cloud and non cloud data. The result is the same, but it easier to see the separation of points and their distributions according to their cloud label.

Figure 1.c.3 shows boxplots of each feature variable, grouped by the classification (cloud, unlabeled, non-cloudy), and similar to the previous two plots', we see that the range and distributions across each feature is different for each classification group. For example, focus on the left plot (boxplots for NDAI). We see that non-cloudy points are distributed way lower than the cloudy points.

Figure 1.c.2 shows the summary statistics for the entire dataset grouped by the classification. We see the same conclusions numerically that we before saw visually. That is, the feature's distribution of values is different for different classification groups. For example, the mean NDAI value for the cloudy group is 1.95 and the mean for the non-cloudy group is -.26

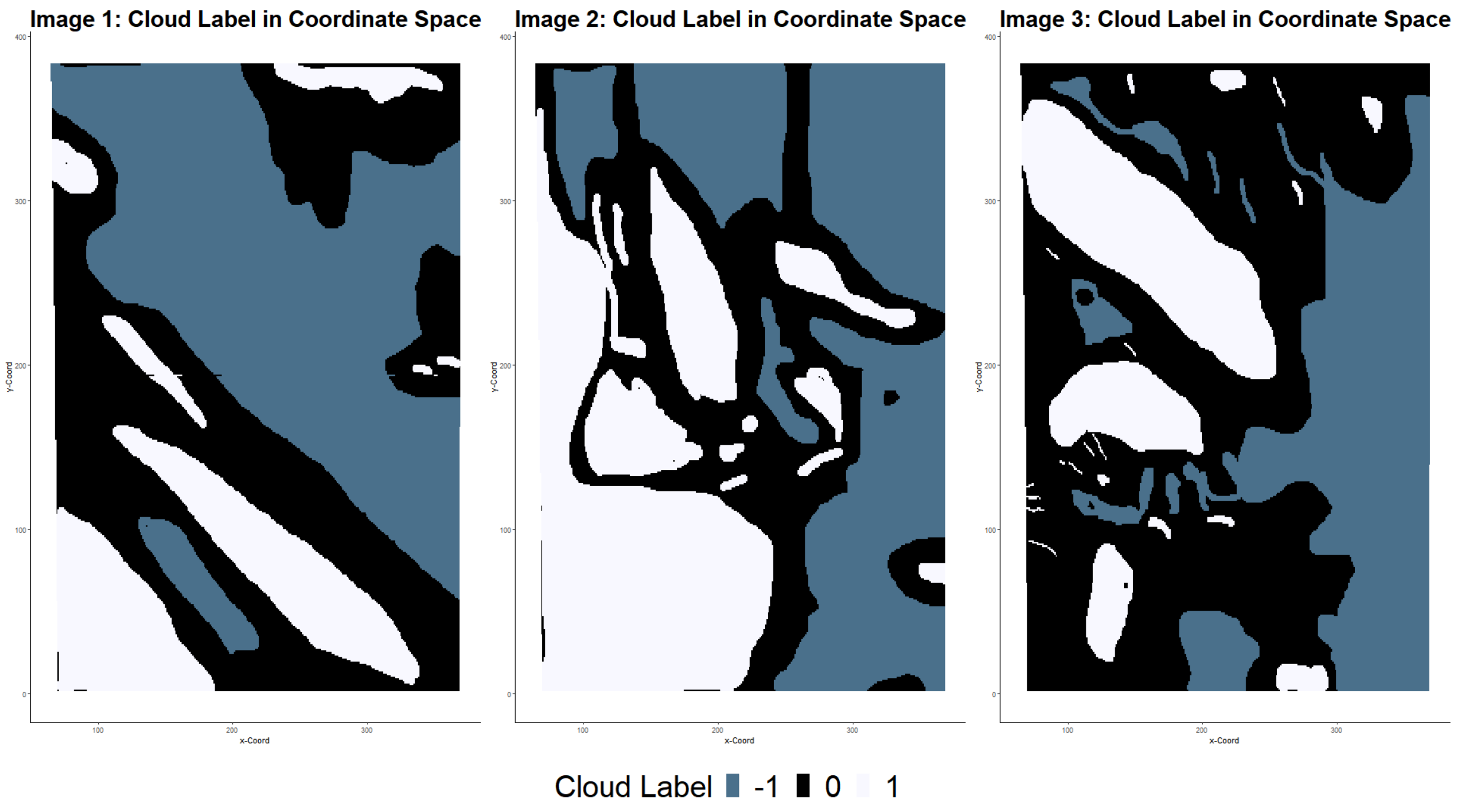


Image 1: Cloud Label in Coordinate Space
Image 2: Cloud Label in Coordinate Space
Image 3: Cloud Label in Coordinate Space
y-Coord
x-Coord
0
100
200
300
400
Cloud Label
-1
0
1

*Figure 1.b.2*

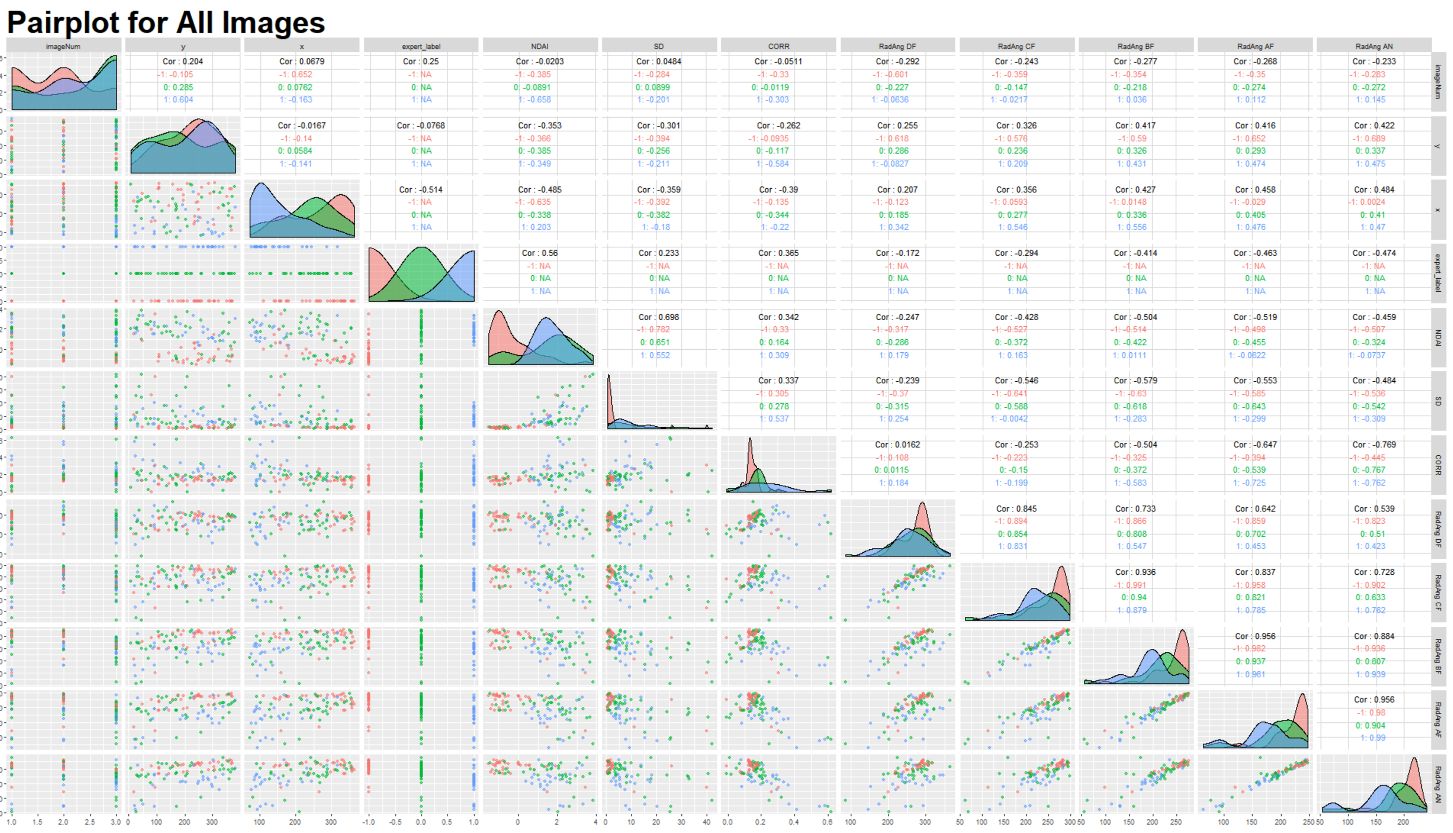


Figure 1.c.1

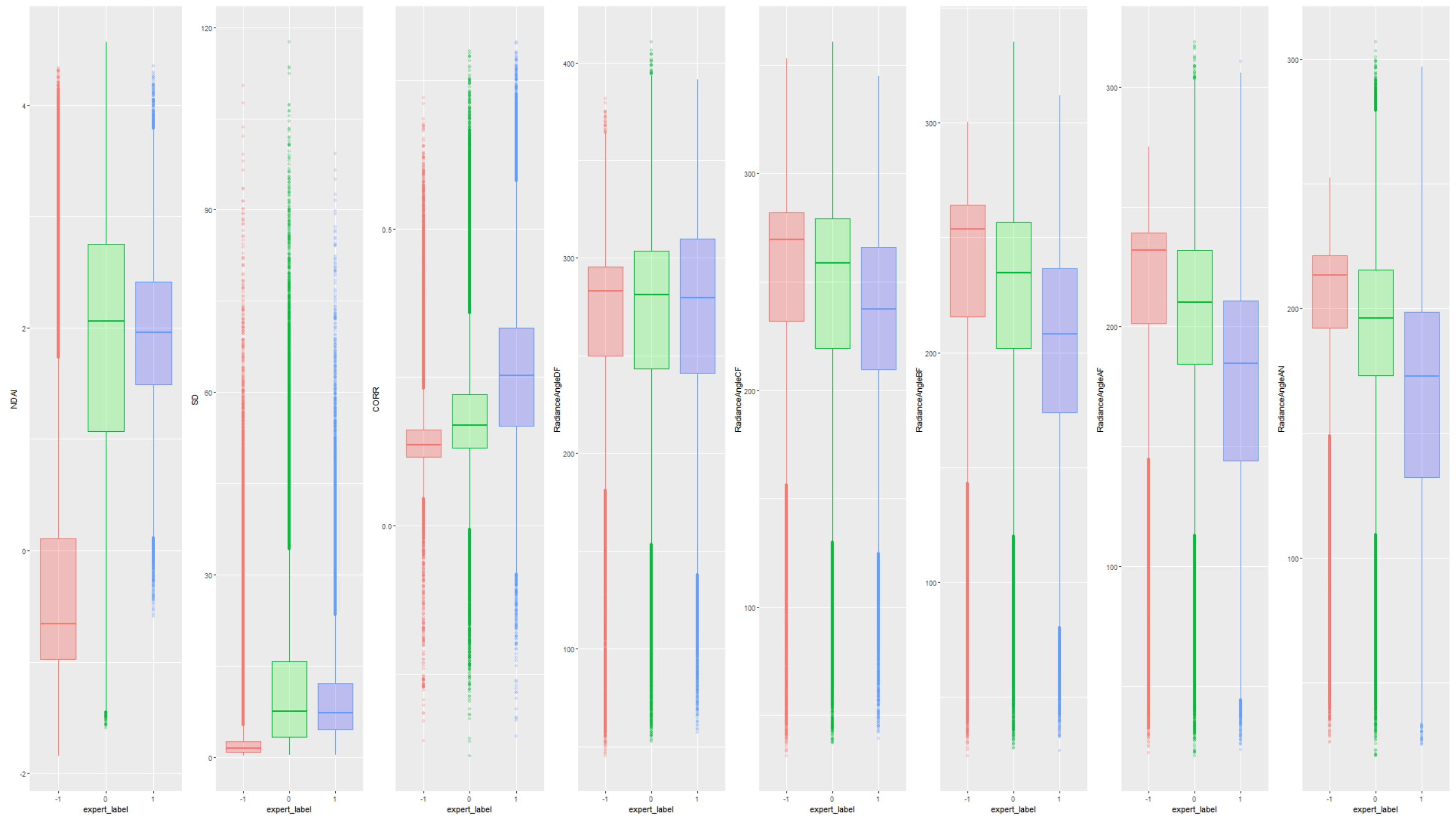


Figure 1.c.3

## 2. Preparation

**(a)** We make the percent of training data be 70%, the percent of validation be 15%, and the percent of testing data be 15%. The trivial way to split the data into training and test would be to randomly select .7 of the pixels to be training and .15 of the remaining to be validation, and the remaining to be test. (In choosing the dataset, we only consider rows expertly labeled as 1 and -1, because those labeled 0 are unknown so we therefore can't use those to try to classify)

**Method 1:** The non-trivial method that we chose to divide into testing and training datasets is by dividing each of the images into squares and then randomly selecting squares to be either in the training or testing dataset. This takes into account that the data is not iid because those pixels next to each other will be grouped together. We chose 40x40 pixel sized squares because we noticed from the maps that squares of those size tend to all be one classification and but they are large enough that they are distinct and not as iid as individual pixels. We justify merging all the images together by assuming that the images themselves are similarly distributed, which means training on image12 should be good enough for predicting on 3.

An extra modification we do to the previous non-trivial way is to divide the images into squares as before and also block on image. Because we want the conclusions to be applicable to all images, we might get a bad randomization from the trivial method (for example, the training data might have much more points from image 1 than from image 2 and 3). To fix this, we block on image by selecting equal proportions of training, validation, and test from each image. This ensures that an equal proportion of each image in each of the training and test datasets, which gives each image equal weight in its effect on the classifier. Further modifications on this method might be to change the squares to have different dimensions (i.e. instead of 40 x 40 pixels, perhaps larger 100x100 squares, larger 100x50 rectangles, or smaller 10x10 squares)

**Method 2:** A second non-trivial way to divide the data is to just choose 2 of the images for training and validation datasets and use the third as the testing dataset (while also implicitly dividing each image into squares). This might be desirable because we want an accurate measurement of how well the classifier does on future data, which are images that the classifier has never seen before. It makes sense that squares within the same image might be highly correlated with each other, so if within an image, we use some squares to train and some to test, the measurement of accuracy on the test dataset would not be representative of the accuracy on a completely new image.

We split the data using both methods. As a sanity check, we map the training dataset below in Figure 2.a.1, which meets what we expect. Method 2's plot is similar to Figure 2.a.1 but with image1 (randomly chosen) as an entire test set and image2 and image 3 split by squares to be in training and validation.

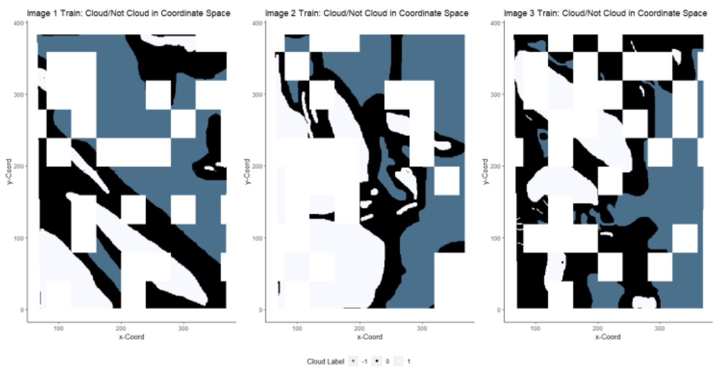


Image 1 Train: Cloud/Not Cloud in Coordinate Space
Image 2 Train: Cloud/Not Cloud in Coordinate Space
Image 3 Train: Cloud/Not Cloud in Coordinate Space
y-Coord
x-Coord
0
100
200
300
400
Cloud Label
-1
0
1

*Figure 2.a.1*

| dataset | Prop Correct {−1,1} | Prop Correct {−1, 0, 1} |
|---|---|---|
| Validation | 0.3483522 | 0.5533115 |
| Test | 0.3405853 | 0.5943577 |
| Val&Test Combined | 0.3445424 | 0.5724814 |

*Figure 2.b.1*

**(b)** Table 2.b.1 shows a table of the accuracy across the different datasets of the trivial classifier in 2 situations: considering labels {-1, 1} and considering {-1, 0, 1}. Because we are testing the classifier in a -1, 1 (binary) classification context, we are mainly focused on the points labeled -1 or 1, and we remove the points labeled 0 in the middle column of Table 2.b.1. The right column considers the classification case of -1, 0, 1.

If we let the classifier be trivial and set all the labels to -1, then the accuracy on the validation and test datasets are shown in Table 2.b.1 and they are around 36% accurate. This is because around 36% of all the data points are labeled as -1 (cloudy), and *this trivial classifier would perform well when the proportion of points labeled -1 is large.* For example, if 99% of the points in some dataset is cloudy, and we classify all points as cloudy, then that classifier would get a 99% accuracy.

**(c)** Using quantitative and visual justification, we chose the three features NDAI, RadianceAngleBF, and SD by looking at the scatterplots and correlations from Figure 1.c.1. We chose the three most important features by focusing on the diagonal and the distribution of each label in each feature space. We choose the features that have the most difference in distribution between each label, but especially focusing on just not cloudy (-1) and cloudy (1). In particular, we look at the fitted distributions on the diagonal, and the most differently distributed labels appear on the NDAI feature space, where the peak and distribution for cloudy is much higher than that for not cloudy points. We then look for the next most different distribution, which appears to be in one of the Radiance Angle measurements, in particular, in Radiance Angle BF, where the peaks are different, which is also evident in the corresponding scatterplots where the cloudy points tend to be separated from the not cloudy points. Next, we chose SD as the final third feature because looking at the scatterplots, we can see a clear division between non cloudy and cloudy points. These visual observations are also supported by the Figure 1.c.2 where we can see that the summary statistics grouped by the classification, and we see that the differences in the distributions are paralleled by the differences in mean in the summary statistics.

**(d)** (The code for CVgeneric is in the project2.R file on github) To create the folds for Cross Validation, we have to split in the same way we split the data in part (a), where we chose 40x40 pixel squares at a time. Because of this, we need information about the x,y coordinate, so we added another input `train_whole` which contains the same points as feature and label but also includes the corresponding x,y coordinate. In addition, in the R code, we include an example of how to use CVgeneric for logistic regression. The function works by first grouping the train_whole dataframe into 40x40 pixel squares, and then creating a mapping from the square to the corresponding indices in the training features, training labels, and train_whole data. Then the folds are created by partitioning the squares into K partitions. Then the generic model is trained and validated from the corresponding squares in each fold.

## 3. Modeling

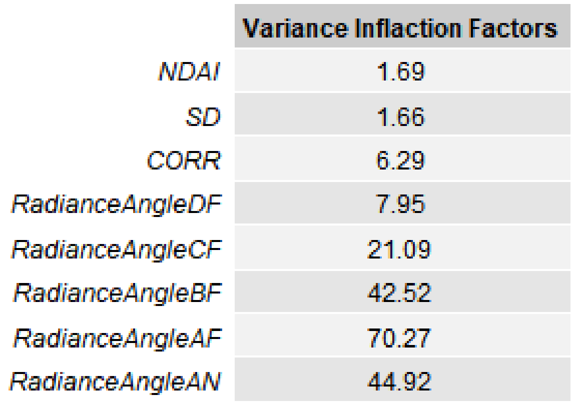
**(a)** Classification methods we try are Logistic Regression, LDA (Linear Discriminant Analysis), QDA (Quadratic Discriminant Analysis), SVMs (Support Vector Machines), K-NN (K- nearest neighbors).

**Checking Assumptions:**

One assumption that all of the models make are that the data is independent and identically distributed by the expert label classification. Unfortunately, by looking at Figure 1.b.2, we see that pixels that are close together in the x,y coordinate space are likely to have the same label. We try to account for the non-iid property of the data by splitting up the data into 40x40 pixel squares during the CV process. In addition, all the methods assume that the response is a binary classification, which is satisfied, because we only consider the cloudy and non-cloudy responses {-1, 1} and we filtered out the unlabeled {0} responses in all the data.

Logistic regression also assumes that the features are linearly related to the logit response, and assumes that the features have low collinearity. To check the linear relationship between the features and logit response, Figure 3.a.2 plots the predictor values against the logit response. We do see some linear relationships such as NDAI where the curve appears very linear. However, we see that many of the features do not result in linear relationships such as Radiance Angle DF where the curve goes up, down up, and down again. This implies that the linearity assumption is violated. To check the assumption of low collinearity, we can look at Figure 1.c.1 again, and focus on the correlations on the top right graphs. We immediately see that the Radiance Angles are very highly correlated with each other.

For example, the linear correlation between RadianceAngleAN and RadianceAngleAF have a is a very high .955. Similar correlations occur with other pairs of radiance angles. This intuitively makes sense because Radiance Angles measure the same radiances just at slightly different angles, but because of this high correlation between features, the resulting classification method will have high variance. We can also check for correlation numerically through variance inflation factors (Figure 3.a.1), where we see the maximum VIF is over 70, which means that the low collinearity assumption is very violated. This could be reduced by features selection (selecting a subset of features or using PCA to use the top most important Principal Components).



| | Variance Inflaction Factors |
|---|---|
| *NDAI* | 1.69 |
| *SD* | 1.66 |
| *CORR* | 6.29 |
| *RadianceAngleDF* | 7.95 |
| *RadianceAngleCF* | 21.09 |
| *RadianceAngleBF* | 42.52 |
| *RadianceAngleAF* | 70.27 |
| *RadianceAngleAN* | 44.92 |

*Figure 3.a.1*

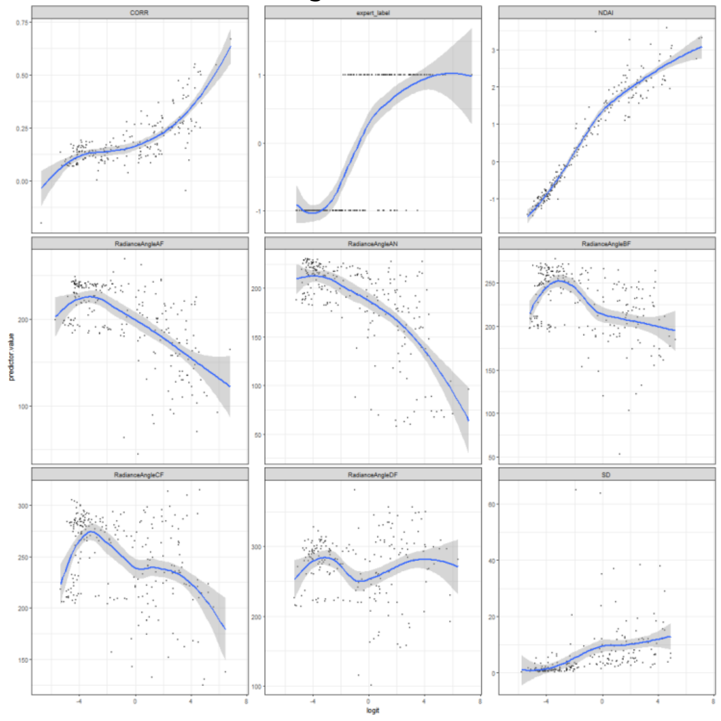


*Figure 3.a.2*

One of the assumptions made by both LDA and QDA is that the data is Multivariate Gaussian. While LDA assumes constant variance for each class, QDA does not assume constant variance across the classes. To check the normality, we plot qqplots of each of the features against a normal. To check the constant variance or non-constant variance, we plot side by side boxplots for each feature and see if the spreads are similar across the labels. The qqplots (plotted with a random sample of 200 from all image data) show that a lot of the features are not distributed normally because the path from points do not follow the 45 degree line marked by the red line. For example, looking at the Cloudy plot on SD (row 1, column 2), This violates both the LDA and QDA assumptions. Just from looking at the boxplots in Figure 1.c.3, we see that the constant variance assumption is violated from visually seeing that the length of the boxes from expert label -1 and 1 are very different. For example, look at the third to left boxplot, with CORR on the y axis. The interquartile range of CORR for non cloudy points is much smaller than the interquartile range for the cloudy points. This suggests that LDA's assumptions are not met. SVM does not really have assumptions except that it wants the data from 2 classes to be as separable as possible, as in the margin between the two classes should be large. In addition, the idea of unequal covariance matrices is supported by the actual correlation matrix in Figure 1.c.1 (the top right corner) where the values are different for each group (indicated by the different colors)

SVM assumes that the there is a hyperplane that mostly separates the two classes or transformations of the data (kernel) within a margin. To check this, we take a closer look at a pairwise scatterplot of the features in Figure 3.a.4. We see that for some features such as NDAI, SD, and CORR, the two classes do seem to mostly be separated by a hyperplane, but for the Radiance Angle measurements, both the cloudy and non-cloudy features occur in the same area with no clear hyperplane dividing them.

KNN assumes that points of the same class are close together in some distance metric (such as Euclidean distance) and farther from points of other classes. To check this, we also look at the pairwise scatterplots, and this assumption does mostly seem to be met, as we see that points of the same class are clustered near each other and away from the other class.

Nevertheless, even if the plots show that many of the model assumptions are violated (ie constant variance, independence, normality), we can still use them as algorithms for classification.

After performing Logistic Regression, LDA, QDA, SVM (with cost hyperparameter = .01, which was chosen in part 4 to have the best validation error), and KNN (with K hyperparameter = 9 for method 1 and 5 for method 2 which was also gotten in part 4) on each of the separate CV folds of the data, we get the results in Figure 3.a.5. We also see the test accuracy gotten from training the classification method on the training data and predicting on the testing dataset for each of the methods. From this table, we see that for split method 1, SVM worked best with a test accuracy of 92.2% and an average CV accuracy of 89.0%. For split method 2, logistic regression actually worked best with a test accuracy of 89.4% and an average CV accuracy of 88.3%. One interesting note about Figure 3.a.5, is that the accuracies from data split method 2 is consistently lower than those of method 1. This might have occurred because method 1 resulted in more training data than method 2 or because we are right to try to split through method 2 because this accuracy estimate is a better estimate for predicting on an entirely new image. Method 1 trains on parts of the same image and tests on the same image; however, in reality, we want to predict on future images where we won't be able to train on parts of the image and just predict other parts of the image. When want to predict on an entirely new image, the Method 2 accuracy estimate is a better estimate for future image accuracy.

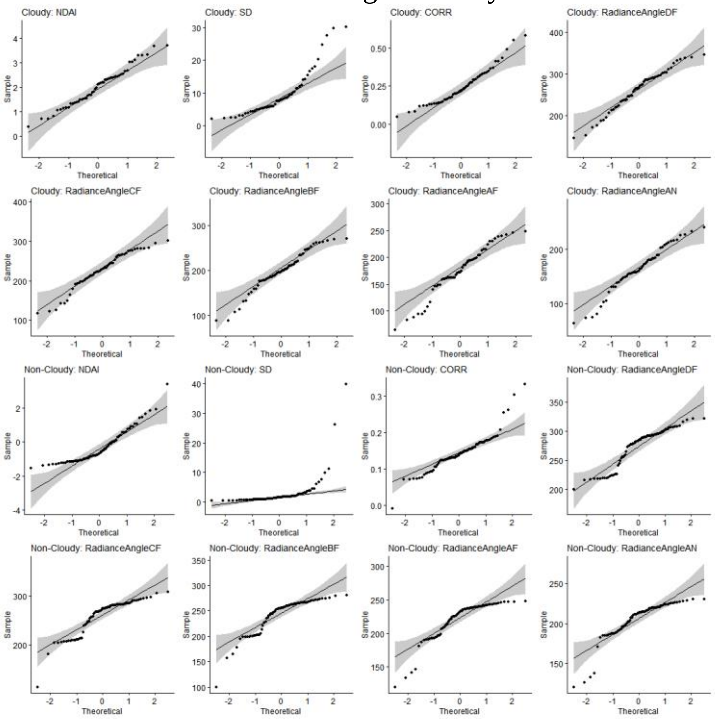


*Figure 3.a.3*

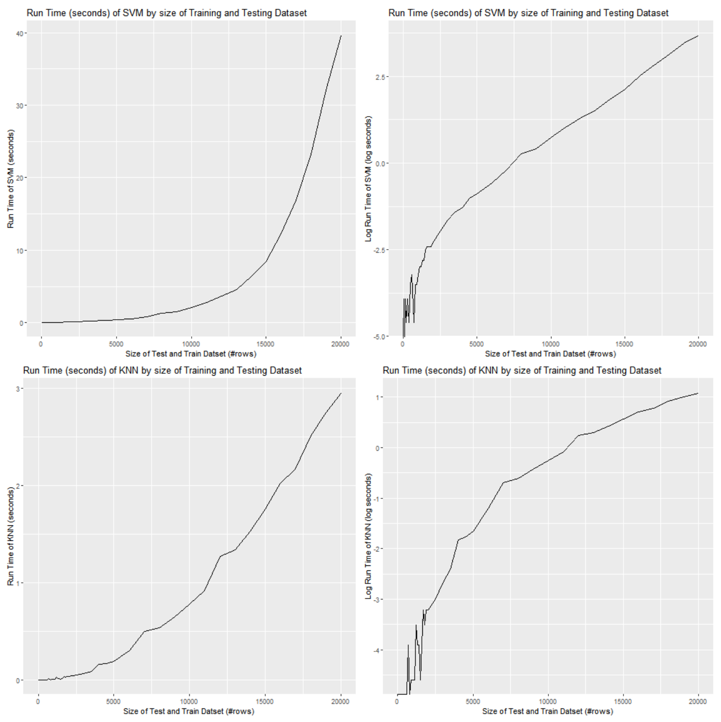

Run Time (seconds) of SVM by size of Training and Testing Dataset
Run Time of SVM (seconds)
Size of Test and Train Datset (#rows)
Run Time (seconds) of SVM by size of Training and Testing Dataset
Log Run Time of SVM (log seconds)
Size of Test and Train Datset (#rows)
Run Time (seconds) of KNN by size of Training and Testing Dataset
Run Time of KNN (seconds)
Size of Test and Train Datset (#rows)
Run Time (seconds) of KNN by size of Training and Testing Dataset
Log Run Time of KNN (log seconds)
Size of Test and Train Datset (#rows)

*Figure 3.a.6*

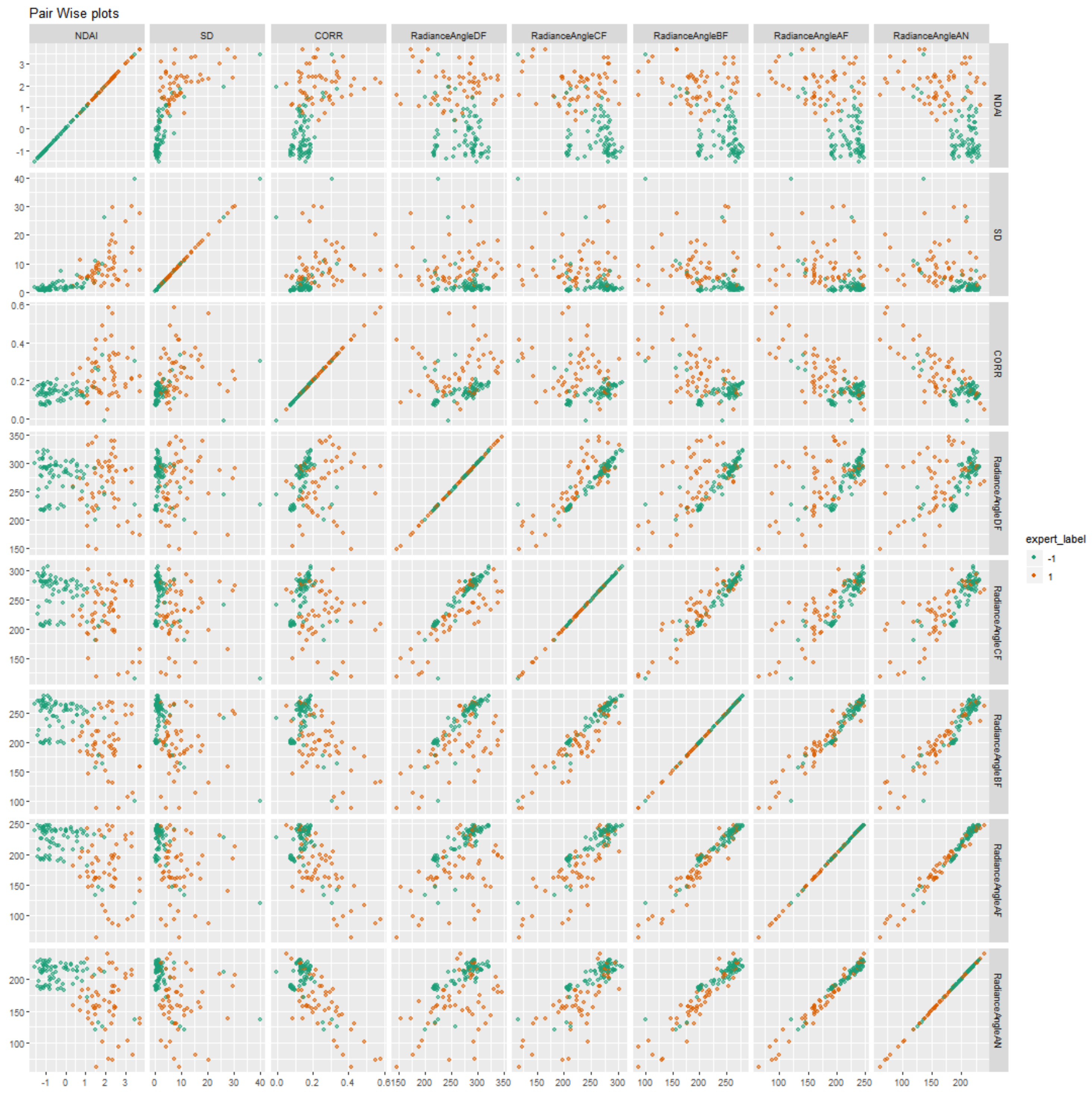


*Figure 3.a.4*

| | Data/CV Fold | Logistic (Cutoff .5) Method 1 | Logistic (Cutoff .5) Method 2 | LDA Method 1 | LDA Method 2 | QDA Method 1 | QDA Method 2 | SVM Method 1 | SVM Method 2 | knn.Method.1 | knn.Method.2 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Test | 0.9096908 | 0.8939323 | 0.9127829 | 0.8018529 | 0.8995537 | 0.8318880 | 0.9219955 | 0.8600618 | 0.8927957 | 0.8135708 |
| 2 | Average Folds | 0.8787398 | 0.8834036 | 0.8815770 | 0.8762022 | 0.8818264 | 0.8977618 | 0.8895948 | 0.8534605 | 0.9002263 | 0.8151256 |
| 3 | 1 | 0.9479812 | 0.9127365 | 0.9529849 | 0.9128093 | 0.9621501 | 0.9300582 | 0.9651163 | 0.9130435 | 0.9001815 | 0.8200692 |
| 4 | 2 | 0.9340610 | 0.8995972 | 0.9356487 | 0.8922612 | 0.9337137 | 0.9123993 | 0.9588477 | 0.8545455 | 0.9614792 | 0.7797203 |
| 5 | 3 | 0.9052494 | 0.9577668 | 0.9057536 | 0.9578568 | 0.9109475 | 0.9610086 | 0.8852459 | 0.9219858 | 0.9081197 | 0.8571429 |
| 6 | 4 | 0.7975584 | 0.9383673 | 0.8098241 | 0.9391767 | 0.8064106 | 0.9647317 | 0.8235294 | 0.9104478 | 0.8380567 | 0.8571429 |
| 7 | 5 | 0.9087554 | 0.8899921 | 0.9019471 | 0.8810087 | 0.9373886 | 0.8963751 | 0.9047619 | 0.8742515 | 0.8805310 | 0.8283186 |
| 8 | 6 | 0.8748323 | 0.8717378 | 0.8730265 | 0.8550351 | 0.8818491 | 0.9041687 | 0.8943089 | 0.8695652 | 0.8926829 | 0.8124006 |
| 9 | 7 | 0.8846606 | 0.9429498 | 0.8889542 | 0.9446313 | 0.8666627 | 0.9603651 | 0.9009009 | 0.9398496 | 0.7747368 | 0.8666667 |
| '0 | 8 | 0.8593678 | 0.7028327 | 0.8676897 | 0.6816319 | 0.8650939 | 0.7406913 | 0.8955224 | 0.6449275 | 0.9489194 | 0.7625755 |
| '1 | 9 | 0.8096901 | 0.8222399 | 0.8086368 | 0.8080909 | 0.8195576 | 0.8278198 | 0.8070175 | 0.7602740 | 0.9842767 | 0.8151261 |
| '2 | 10 | 0.8652416 | 0.8958157 | 0.8713045 | 0.8895202 | 0.8344897 | 0.8800000 | 0.8606965 | 0.8457143 | 0.9132791 | 0.7520938 |

*Figure 3.a.5*

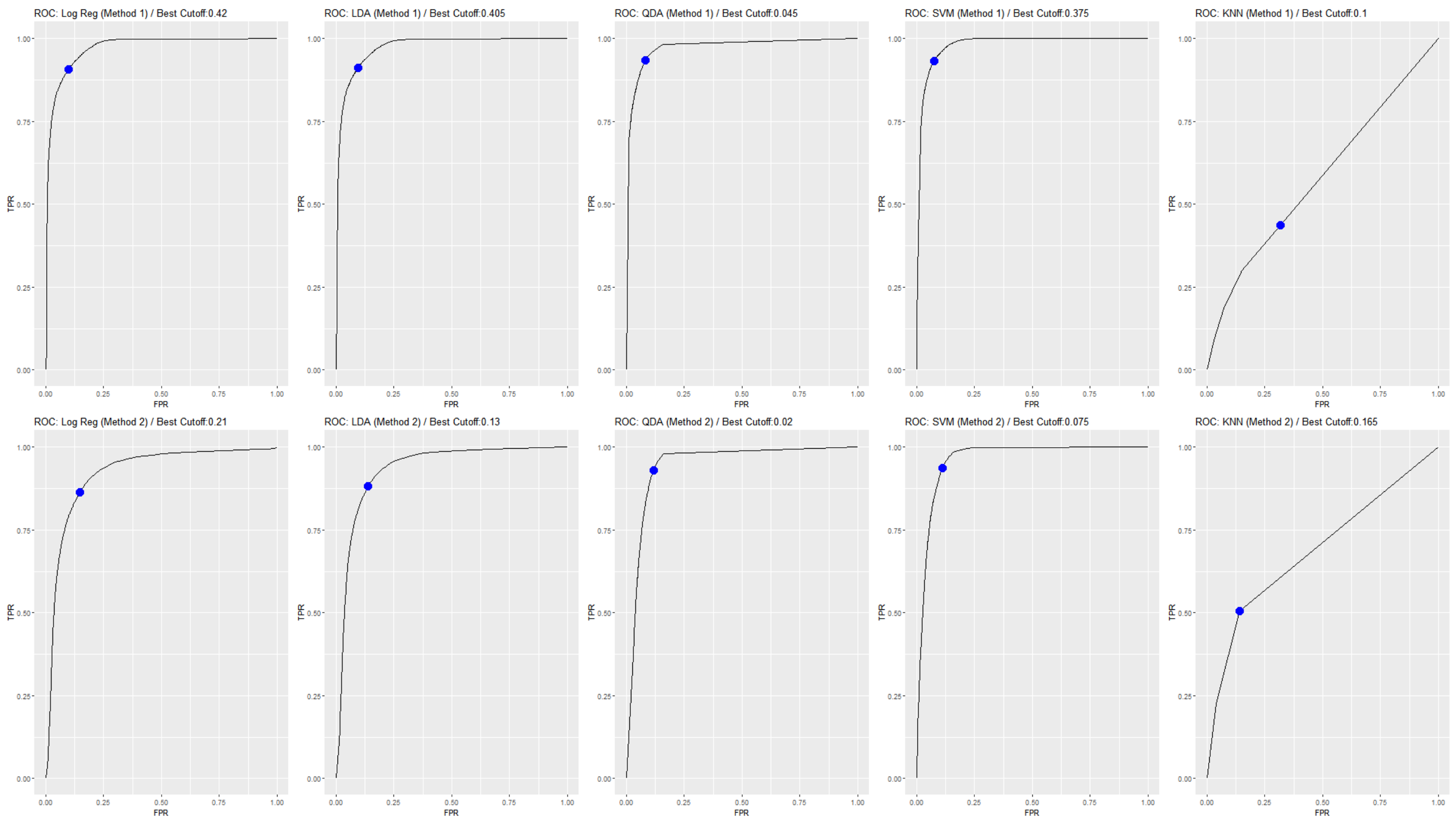


*Figure 3.b.1*

One note is that SVM and KNN was not able to finish running with the original training dataset. To ensure that SVM and KNN finished running in a reasonable amount of time, we merged 10x10 pixel squares into a large pixel for SVM and 5x5 pixel squares for KNN. For the values of the features, we averaged over them, which gives a more robust measurement of the feature (ie the variance of the measured feature of the square has less variability than the measured feature of the individual pixel if we assume that the square has one set expert label), and the Expert Label was set to be the mode of the pixel labels in the 10x10 square. For an analysis of the run time, we ran svm and knn with differently sized inputs and timed, which resulted in Figure 3.a.6. In the graphs in the right column, the size is plotted against log(seconds), and we see that after around 2700 rows, the see that the curve is linear, so we conclude that the run times for svm and knn are exponential to the number of rows (size).

**(b)** Figure 3.b.1, shows the ROC curves for Logistic Regression, LDA, QDA, SVM, and KNN. We chose the cutoff point that results in the closets ROC curve value closest to the point (0,1) (the upper left corner) and marked the cutoff point in the title. We chose to pick the cutoff point in this way because it provides the best balance of high True Positive Rate and low False Positive Rate in terms of Euclidean distance.

We also note that KNN has a very odd ROC curve, and this is probably because KNN finds the K (9 for method 1 and 5 for method 2) nearest neighbors and predicts the probability by the proportion of neighbors that are of the class. Therefore, there is only a limited amount of possible values that the predicted probability could be (ie only 0, .2, .4, .6, .8, 1 are possible predicted probabilities for method 2), which means that choosing cutoff points from 0, 1 will make the resulting ROC curve piecewise linear with long pieces.

In addition, we found the Area Under the Curve of the ROC curve which is displayed in Figure 3.b.2. A good classifier will have AUC's close to 1, and we see some that some get close (the maximum is .85), but others are quite low (the minimum is .63). Another note is that the AUC's tend to do much better on the method 1 data split than method 2's by a significant amount (an average difference of .134 across the 5 classifiers)

| | method | AUC_method1 | AUC_method2 | difference |
|---|---|---|---|---|
| 1 | Logistic Regression | 0.8055670 | 0.6171936 | 0.18837340 |
| 2 | LDA | 0.8528896 | 0.6495948 | 0.20329479 |
| 3 | QDA | 0.8151084 | 0.6376591 | 0.17744932 |
| 4 | SVM | 0.8513823 | 0.7036408 | 0.14774156 |
| 5 | KNN | 0.1162440 | 0.1513399 | -0.03509594 |

*Figure 3.b.2*

## 4. Diagnostics

**(a)** From comparing the classification errors and AUC of the different classifiers in part 3, we find that SVM with a linear kernel provides a relatively accurate fit for the data. We choose, by performing cross-validation on the combined train and validation data, the hyperparameter *Cost* which yields the lowest mean CV error. Figure 4.a.1 shows that the Cost value which results in the lowest mean CV error is 0.01. The CV error is on the whole very low, at around 0.07 for all values of Cost.

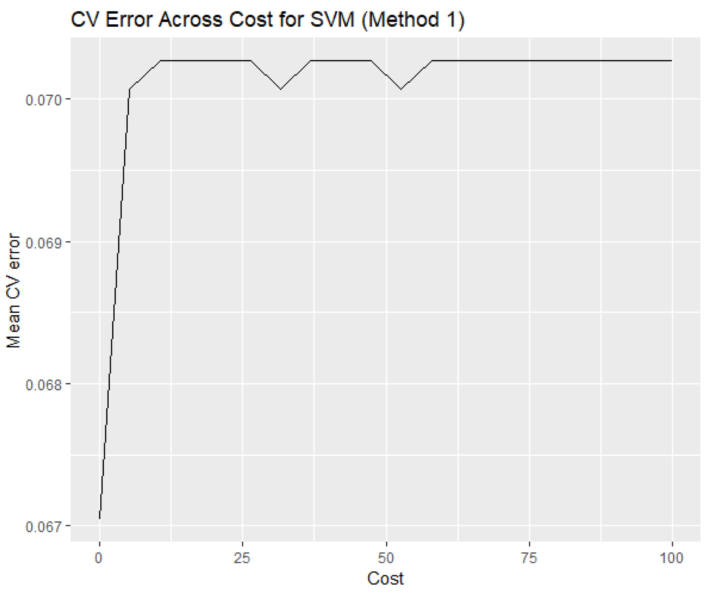


*Figure 4.a.1*

After performing cross-validation, an SVM model with linear kernel and *Cost* set to 0.01 (as described above) was fit on the training data. The output of *Summary* on the model tells us that the model's support vectors comprise a large proportion of the training data; this is because our *Cost* value is so low, meaning that there is a smaller penalty placed on vectors appearing on the wrong side of the margin. The model was then fit on the test data; the test accuracy is 93.0%, indicating that the model is relatively accurate at classifying the data. Figures 4.a.2, 4.a.3, and 4.a.4 each depict the estimated separating hyperplane superimposed on the plot of two chosen features from the test data, namely *RadianceAngleAF* vs. *NDAI*, *RadianceAngleDF* vs. *CORR*, and *NDAI* vs. *CORR*. It appears from these plots that the boundary between the two classes may be somewhat non-linear.

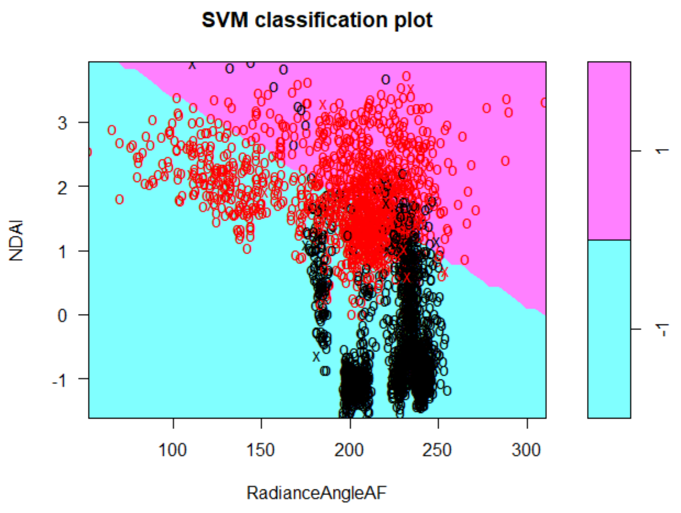


*Figure 4.a.2*

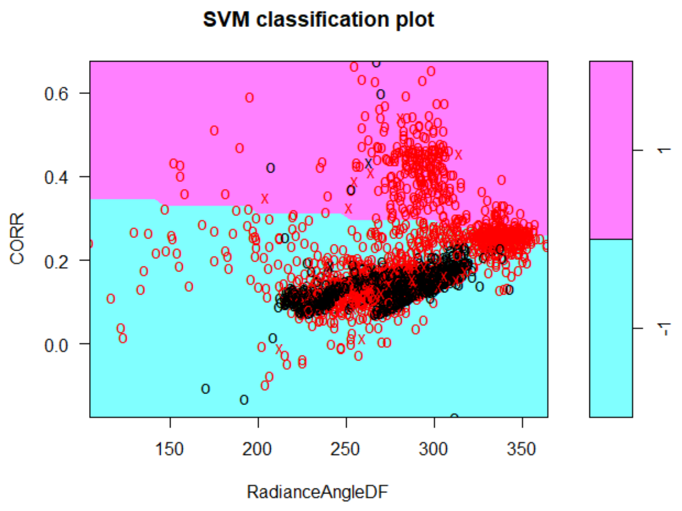


*Figure 4.a.3*

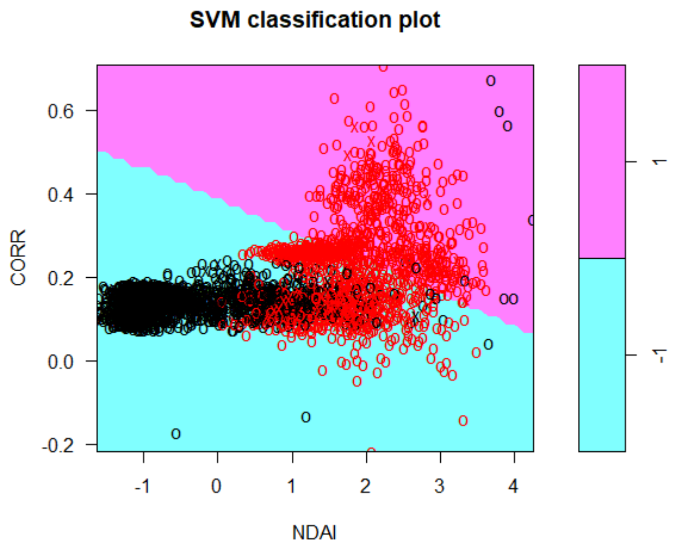


*Figure 4.a.4*

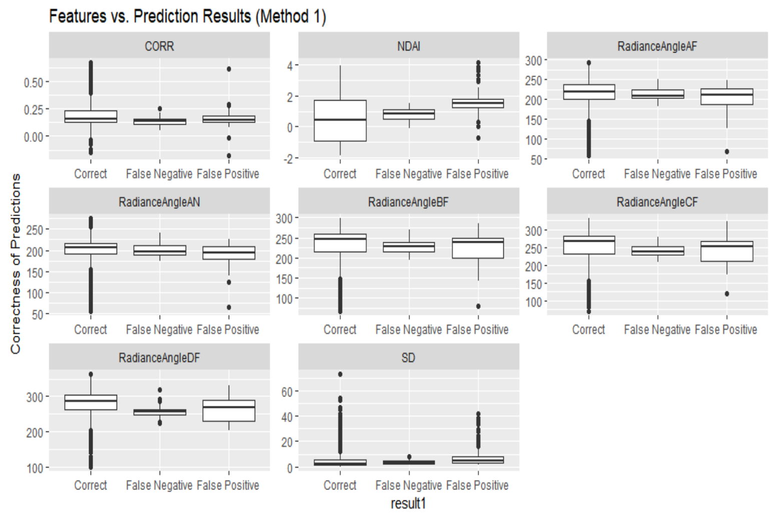
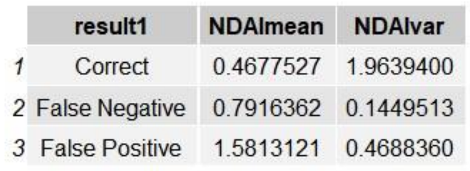


*Figure 4.b.1*

**(b)** Figure 4.b.1 shows the two types of misclassification (False Positive and False Negative), as well as the instances in which expert_label was correctly classified, vs. the ranges of each feature used in prediction on the test data. From these plots, it appears that the range of values for which false classification occurs is limited, and that the false positive and false negative predictions largely overlap in terms of the range of values for each feature. Thus, because it is not the case for any of the features that in a certain range, the classifier systematically makes largely false-positive errors or vice-versa, largely makes false-negative errors, we cannot conclude that there are any patterns in the misclassification errors possibly due to bias.



|  | result1 | NDAImean | NDAIvar |
|---|---|---|---|
| 1 | Correct | 0.4677527 | 1.9639400 |
| 2 | False Negative | 0.7916362 | 0.1449513 |
| 3 | False Positive | 1.5813121 | 0.4688360 |

*Figure 4.b.2*

To quantitatively see the difference between the different types of misclassified and correctly classified pixels, we focus on NDAI and group by the type of result (Correct, False Negative, or False Positive), which is shown in Figure 4.b.2. From this result, we see that the correctly classified points have a low average NDAI value (.47) with high variance (1.96) while the incorrectly classified points have higher NDAI averages (.79 for FN and 1.58 for FP) with much lower variances of .13 and .47. The largest difference is seen in the difference of means of NDAI between Correctly classified points and False Positives. Similar differences in mean values and variances are seen in other features and most easily seen in the boxplots of Figure 4.b.1.

**(c)** Parts a and b suggest there may be some nonlinearity of the boundary between the two classes; thus, we explore the performance of the SVM model with polynomial and radial kernels. For the model using the polynomial kernel, we perform a sweep of CV error with 10 folds over values of *degree* (2 to 5; 1 is not used, as this corresponds to the linear kernel, which we have already explored in part a). Figure 4.c.1 shows that the CV error is greater than that of the linear kernel for all values of the hyperparameters *Degree* and *Cost*. Furthermore, the test accuracy of the SVM model with polynomial kernel with degree 3 is 88%, which is lower than that when using the linear kernel. Thus, we conclude that the classification boundary between the two classes may tend more towards a hyperplane than one of a polynomial nature, and that usage of the linear kernel may be preferred over the polynomial kernel.

Likewise, for the model using the radial kernel, we perform a sweep of CV error over selected choices of gamma. Figure 4.c.2 shows that the lowest CV error for the radial kernel is comparable to those for the linear kernel. The SVM model with radial kernel trained with gamma = 2 (the value corresponding to the lowest CV error) is 93.7%, indicating that the classification boundary is perhaps more nonlinear as characterized by the radial function than it is linear; thus, the SVM model with radial kernel may perform slightly better than that using the linear kernel.

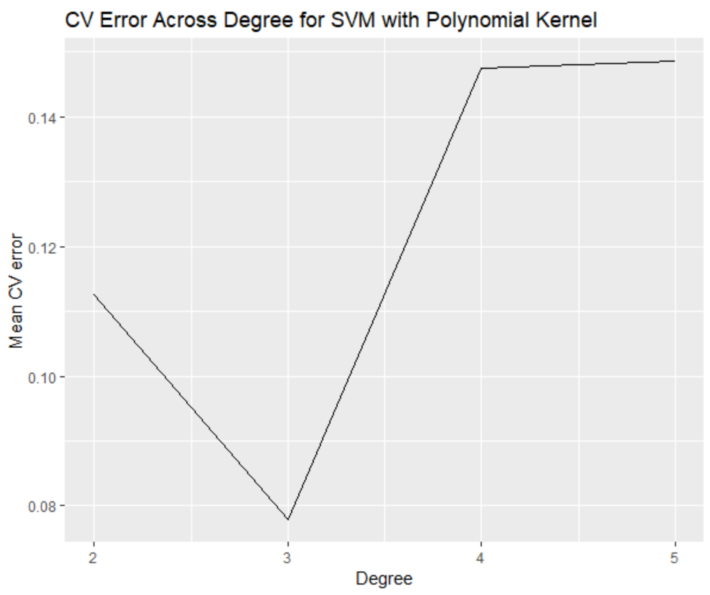


*Figure 4.c.1*

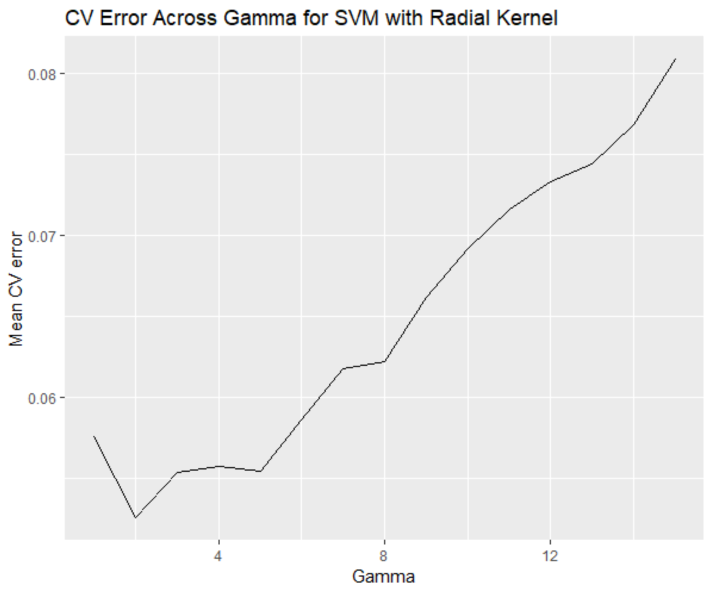


*Figure 4.c.2*

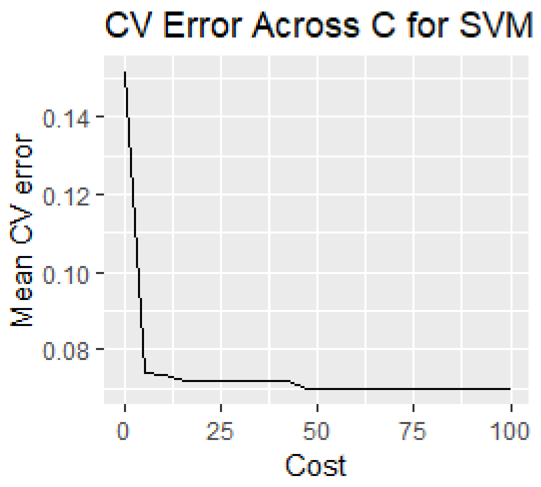


*Figure 4.d.1*

**(d)** The results in the first two parts do change as we replicate the same tasks in those parts, except now we use data split using Method 2 described in the Preparation

Section. In the Figure 4.d.1, we see that the mean CV error is greater on average than those using Method 1.

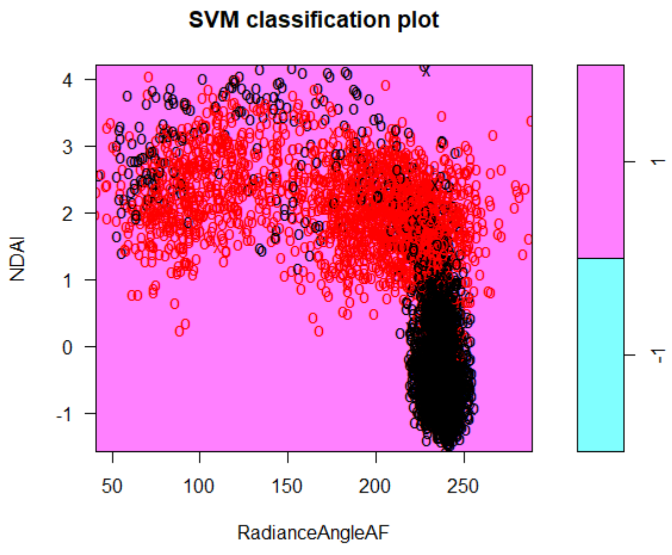


*Figure 4.d.2*

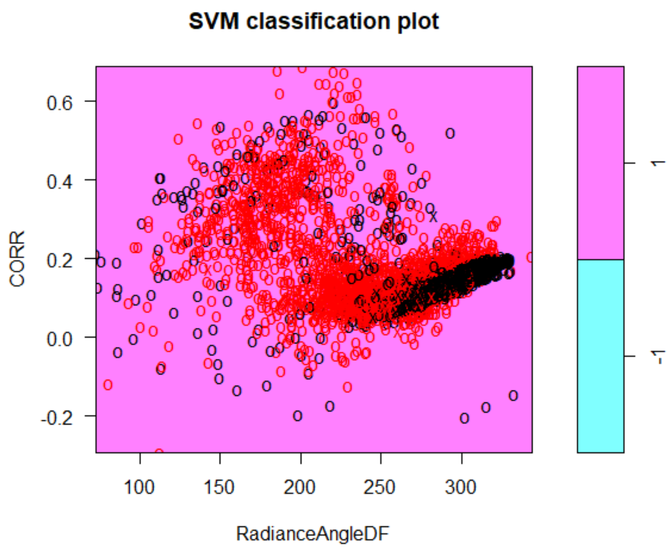


*Figure 4.d.3*

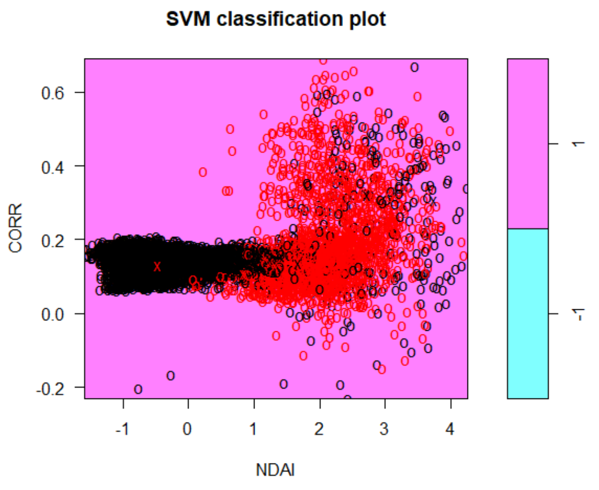


*Figure 4.d.4*

Furthermore, the test accuracy has decreased (93% vs 85% corresponding to Methods 1 and 2, respectively). The left three figures show the plot of classified points and support vectors, and the estimated separating hyperplane intersecting each pair of features. The position of the hyperplane has not appeared to improve the classification of the data.

Finally, Figure 4.d.5 shows the misclassification types and true classifications corresponding to the range of values for each feature. Now, the False Positive and False Negative misclassifications extend across the range of values for most of the features, instead of for just a portion of the range, as we see in part b. This agrees with our findings earlier that the classification accuracy has dropped by using Method 2.

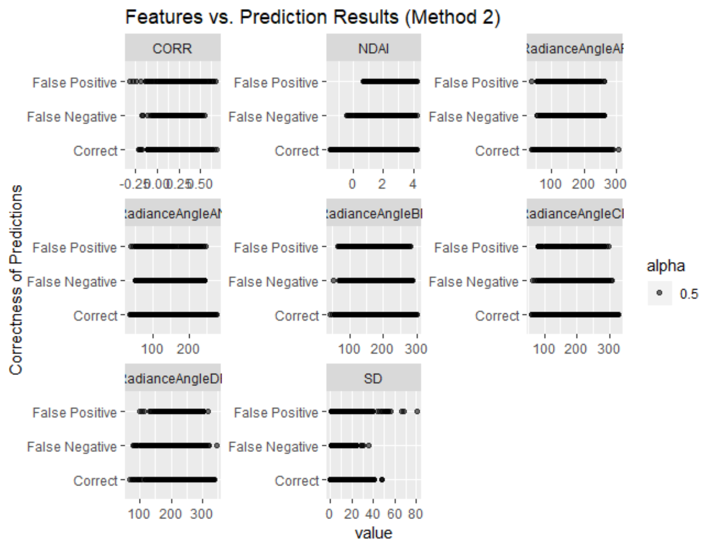


*Figure 4.d.5*

Method 1 trains on subsets of squares from all the images and tests on the rest, while method 2 trains on two of the images and tests on the third image. We believe that the second method will give a more accurate measurement of real prediction error because when actually predicting, we won't be training on parts of the image while testing on the rest of the image; instead, we would be training on previous whole images, and then testing on completely new images.

**(e)** asdf

## 5. Reproducibility

Github: https://github.com/catherinewang1/Stat154-Project2

**Acknowledgements:**

Contributions:

**Catherine Wang:**
1abc, 2abcd, 3a (log reg, lda, qda, svm) 3b, 4b (small quantitative part) d (small paragraph), 5, formatting and creating word doc/pdf rendering, Readme.md

**Yijin Wang**:
3a (knn), 4, 5 (project2_4.R code)

Resources Used:

ISL Textbook:

- http://www-bcf.usc.edu/~gareth/ISL/

For explanation of Logistic Regression:

- https://www.statisticssolutions.com/assumptions-of-logistic-regression/
- http://www.sthda.com/english/articles/36-classification-methods-essentials/148-logistic-regression-assumptions-and-diagnostics-in-r/

For explanation of LDA (and QDA) and tutorial:

- http://thatdatatho.com/2018/02/19/assumption-checking-lda-vs-qda-r-tutorial-2/

For plotting help in extracting legends from ggplots:

- https://stackoverflow.com/questions/13649473/add-a-common-legend-for-combined-ggplots
- https://github.com/hadley/ggplot2/wiki/Share-a-legend-between-two-ggplot2-graphs

For a function getting the mode of a vector:

- https://www.tutorialspoint.com/r/r_mean_median_mode.htm