Project 2 Cloud Data

By: Minjeong Kim(3031914770) and Timlan Wong (26222491)

Question 1a) Summary of Literature

In order to detect the presence of clouds, the research aims to use cloud-free surface pixels rather than the cloud-detecting surface detection algorithms like in MISR. ELCM and ELCM-QDA area used to distinguish surface pixels from cloudy pixels. The goal of the research is to correctly characterize the clouds in the Arctic.

MISR covers the exam same path every 16 days. Each path is subdivided into 180 blocks, with block numbers increasing from the North Pole to the South pole. Each complete trip is given its own orbit number. A huge amount of data is collected (275 m x 275 m). Red radiances are transmitted at full 275 m resolution while the blue, green, and near-infrared radiances from the non-nadir cameras are aggregated to 1.1 km x 1.1 km resolution before transmission. Cloud detection in the arctic has been difficult. Gaussian kernel support vector machines and clustering has been used, but they have not been able to tell apart data units to be cloud-covered or totally cloud-free. There is a challenge to combine classification and clustering schemes in a computationally efficient manner. The goal of the work is to build operational cloud detection algorithms that can efficiently process the massive MISR data set one data unit at a time without human intervention. They look for cloud-free instead of ice and snow covered surface image pixels because surfaces do not change much between views whereas clouds are always different from one view to the next.

The L2TC algorithm produces two cloud masks: the SDCM and the ASCM. The SDCM compares retrieved object heights with known terrain heights, and objects more than 650 m above the terrain height are classified as clouds. The ASCM is based on BDAS, which characterizes the relationship between the difference of two solar spectral reflectances with view angle. The longer the solar ray paths through Earth's atmosphere before reaching MISR, the larger the BDAS. Since radiation reflected from clouds has much shorter paths than radiation reflected from Earth's surface, small BDAS indicates clouds. The SDCM cannot detect low clouds because the distances between the low clouds and the surface are often smaller than MISR height retrieval accuracy. The ASCM is good at detecting high and thin clouds but has difficulty detecting low clouds over terrain.

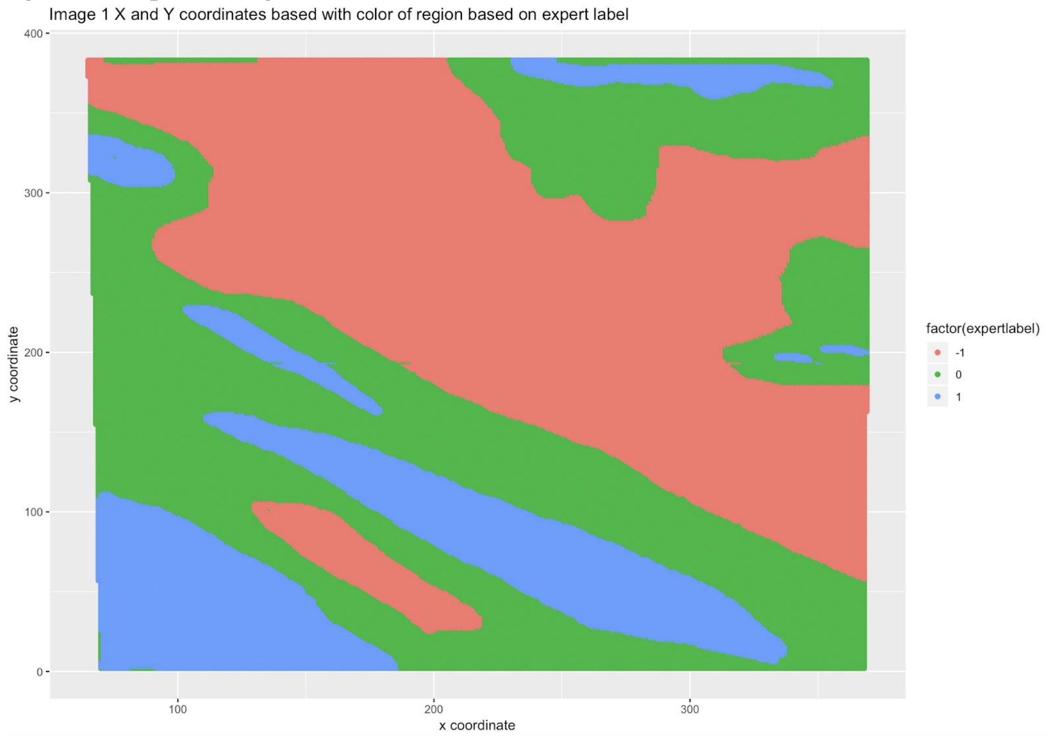
The data used in this study was collected from 10 MISR orbits (April 28 - September 19, 144 days) of path 26 over the Arctic, northern Greenland, and Baffin Bay. Six data units (MISR blocks 11-13, 14-16, 17-19, 20-22, 23-25, and 26-28) from each orbit are included in this study. **Expert** labeling of clear and cloudy scenes was at one point their best method for producing validation data for assessing automated polar cloud detection algorithms (<u>used only on the first day</u>). Then JPL labels the pixels in MISR nadir camera images as clear or cloudy. White meant cloudy, gray meant clear, and black is the ambiguous regions left unlabeled.

The performances of the ELCM and ELCM-QDA algorithms was compared with the expert labels. They discuss results from the data unit with the lowest agreement rate between the ELCM algorithm and expert labels to provide insight into the reasons why ELCM algorithm fails at certain geo-locations.

This work is significant for statistics in two ways that go beyond technical development and implementation of statistical methods. The past decade has experienced an explosion of Earth science data that support weather and climate studies. A second significant aspect of this research is that it demonstrates the power of statistical thinking, and also the ability of statistics to contribute solutions to modern scientific problems.

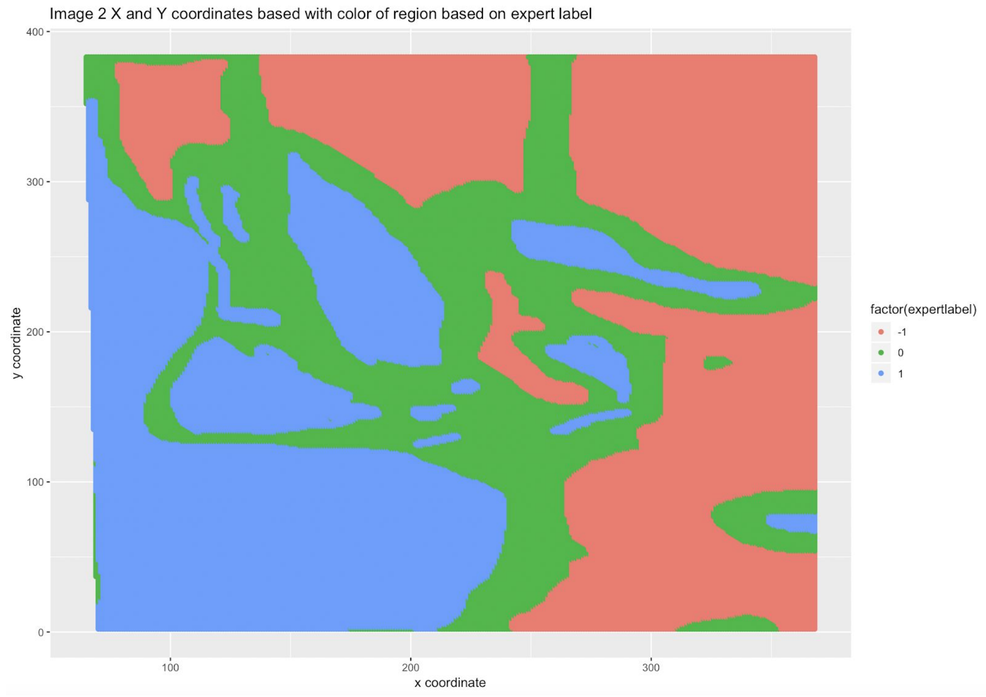
Question 1b) Maps

Figure 1: Maps of Image 2 with XY coordinates



From this map of just image 1 we can clearly see that the x-coordinate and y-coordinate has a strong predictive power of the different outcome labels: -1, 0, 1. Moreover, we see from this clearly that our data points aren't i.i.d. If we are given a data point in some region of this xy space, we probably know what the outcome of the points' outcome is around it because each region is correlated according to its geospatial region. This is also physically reasonable because cloud data's result shouldn't be completely independent given their geographical coordinates. Therefore this is the first clue that we don't have an i.i.d sample given these covariates. In order to have a more robust analysis we will also consider one other image below.

Figure 2: Map of Image 2 with XY coordinates

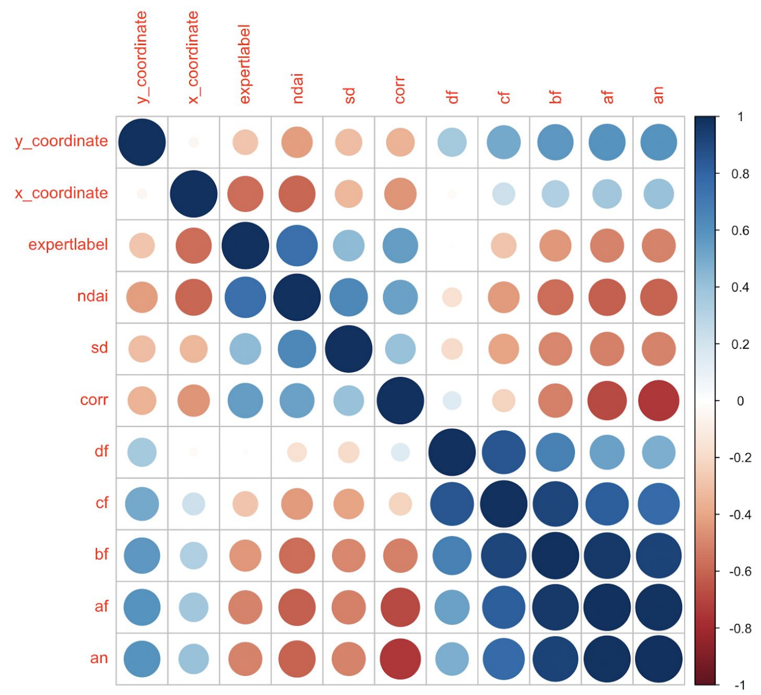


Although Image 2 shows somewhat a different trend, the story it tells about the i.i.d and the predictive power x and y coordinates have is similar. We see that actually in image 2 at least vertical strips (i.e. the y-coordinate) probably seems to predict the result more. We also see from both image 1 and image 2 that the more right-side of the graph seems to be a bit more not cloudy (-1) while the left hand side seems to be more cloudy in both pictures. However, from these simple EDA it is clear that we have good predictive power from the coordinates and that also these are not i.i.d, i.e. given some of these covariates the distribution of our main dependent variable, expert label, is not the same.

Question 1c) EDA

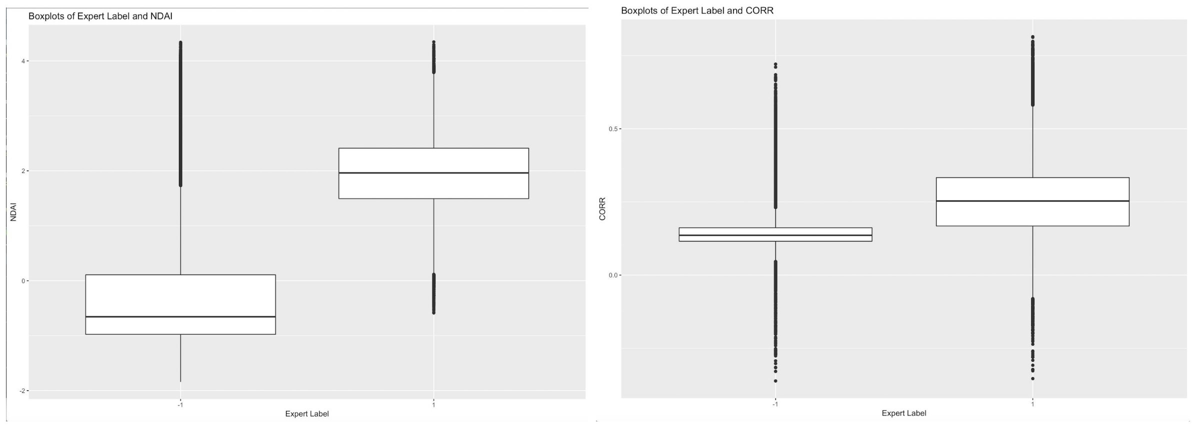
From Figures 1 and 2 it is clear that we see the x and y coordinates have strong predictive power but we are curious whether or not other variables also have this power and also whether or not they are perhaps more strongly correlated. One visually powerful way to visualize this is through a correlation matrix/plot. The following figure shows this. For the following EDA because we only care about classifying 1 and -1 (cloudy and not cloudy) data points we will remove all the unlabelled ones.

Figure 3: Correlation Plot:



We want to pay most attention to the third row with expert label. We see that it is quite strongly correlated with the coordinates and ndai and corr. This is expected. From this plot it gives us motivation to consider all the three added variables from the paper (ndai, sd, and corr) and the coordinates as we saw from the map. The radiance angles are very highly correlated with each other and also the y_coordinate, which makes sense given the geographical interpretation of these variables. Therefore it is probably not a good idea for multicollinearity reasons to include these radiance angles along with the coordinates. This correlation plot is a good EDA to tell us what variables to consider for our models. We will now further explore specifically the boxplots of NDAI and CORR since they are quite strongly correlated with expert label.

Figure 4: Boxplot of Expert Label and NDAI and Boxplot of Expert Label and CORR

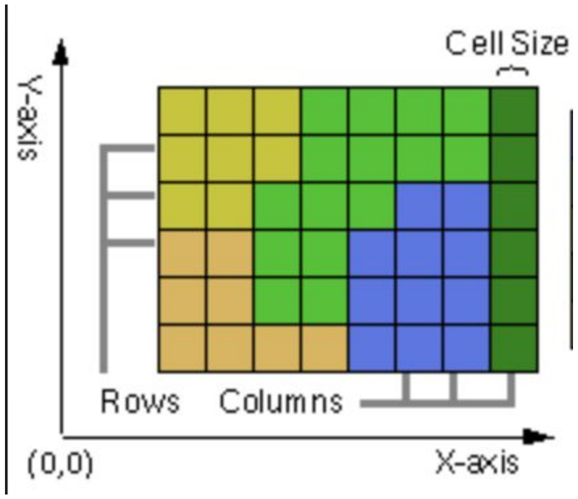


From figure 4 we can see that NDAI can split the expert label very well while CORR does not as well. This is probably a very good indication that NDAI could be a very significant and powerful predictor when we run our models. What we can also see is that the distribution of the CORR in level 1 expert label is “fatter” than the other. While in NDAI there are more outliers in the right tail. These are things that can drive certain coefficients in models like logistic regression to be less robust. These are things to watch out for!

Question 2a) Splitting First and Main Method:

In our above EDA and plots we clearly see that the data is not i.i.d given the covariates. Therefore the best way to split our data is to randomly select blocks that we think are i.i.d data inside each block. Now the problem becomes how to define these blocks. One way to define the blocks is to literally geospatially create blocks according to the x and y coordinates. The following figure represents a visual representation on what we did:

Figure 5: Splitting based on xy coordinates



Although our dataset will be much bigger than figure 5 and it will contain more splits (we split the grid into 10,000 blocks: the x_coordinate split into 100 equal parts and the y_coordinate split into also 100 equal parts). Then once we got the individual splits we grouped them up to create one “block factor” to identify the block group. From here we then sampled for example 70% randomly without replacement from the blocks rather than the data points to make it i.i.d. Note this doesn’t actually equal to sampling 70% of our original train data because the blocks that we artificially created will not be necessarily always balanced. This works because once we have blocked them within small geospatial blocks that are relatively fine grained (10,000 blocks is quite a big number) there is good reason to believe inside these blocks all the points are relatively next to each other and hence can be considered i.i.d without worrying the same problems we had above. We then sample also 10% from the remaining for our validation set and the remaining data for our test set. All this can be further explained in our Rmd file and README. Also this is the main method of splitting we used to further carry out the analysis in later sections.

Second Method:

Our second method of splitting it into the train, validation, and test was very similar to our first method in that we blocked by a certain covariate that had strong predictive power of expert label. The logic of why blocking with these covariates is exactly the same as blocking for x and y coordinates. The variable we chose is NDAI simply because the above EDA suggested strong predictive qualities. Therefore, we split NDAI into 1000 equal blocks using the same method and used these blocks to create our “i.i.d blocks”.

Then we got our training, validation, testing sets by sampling from these blocks that were split according to NDAI.

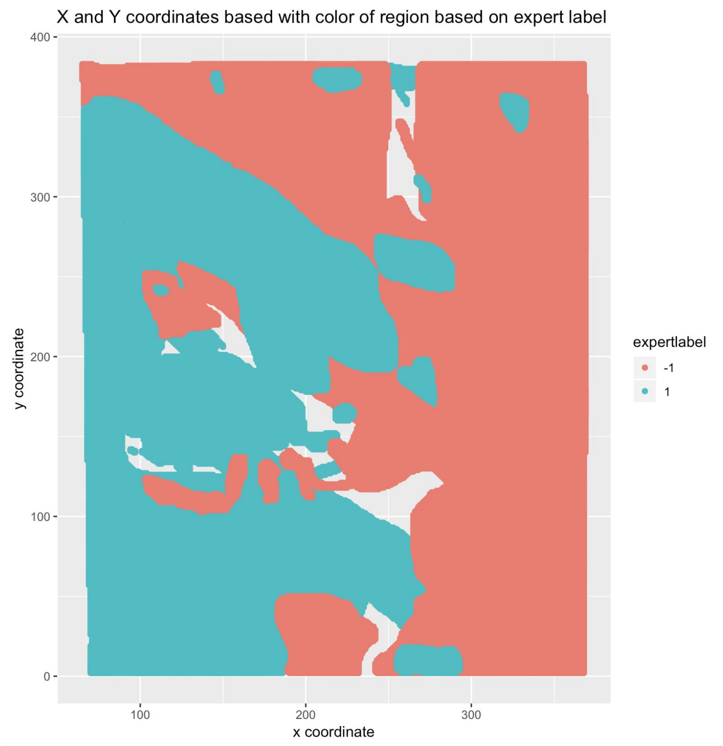
Question 2b) Trivial Classifier

We see that our validation accuracy from our split is about 61.85% while the test accuracy from our split is 61.1%. This is a good baseline indication that any model we build should probably be higher than this or we might as well just guess -1 every time. Also this probably tells us there are more -1 (not cloudy) data points than cloudy data points or else the accuracy would be less than 50%. In a scenario where it's more extreme, i.e. has maybe 90% -1 data points while only 10% 1 data points, this classifier might reach more than 85% accuracy rate. It all depends on the proportion of each class the data set has.

Question 2c) Variable Importance

What we first wanted to see was a map with just the two binary labels -1 and 1 on the xy space to see if the coordinates are still good to separate out the response variable. Here we concatenated all image 1, 2, and 3 into one dataframe and plotted it.

Figure 6: All combined image plotted in XY space



We see Figure 6 that our initial guesses of having more cloudy data points on the left and less cloudy data points on the right is exactly corrected in the combined data frame. Therefore it is obvious that our x and y coordinates should prove to be a very important feature. From this figure we would definitely like to include these two.

One quantitative method to check variable importance is to actually use the step() function where the methodology is relative straight forward. It first tries to put one variable at a time and see and adds a variable step by step based on how much AIC is increased (AIC is a very good proxy for predictive power). It does this until it exhausts all variables or if there is a decrease or no significant improvement in AIC. This is a good way to see which variables are selected and in which order they are selected to prove which one is the most important. From this result we found that the variables were ranked in the following way:

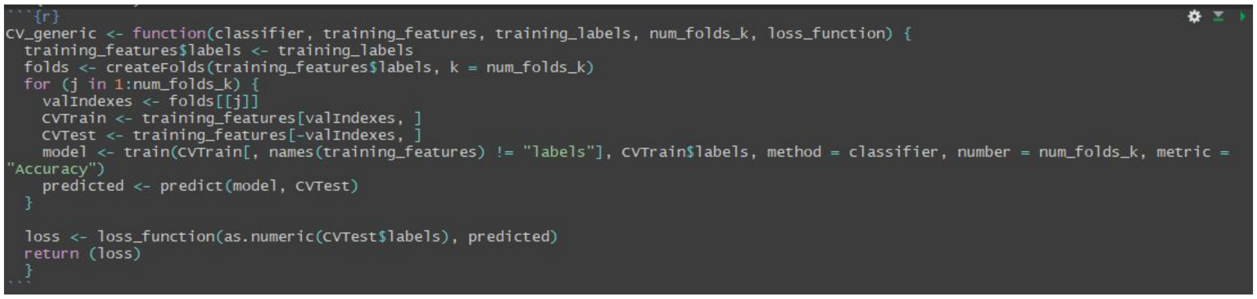
Note: Due to the EDA and multicollinearity we only decided to look at x, y coordinates, NDAI, SD, and CORR

Table 1: Quantitative variable selection using step method

	Ranking	**AIC**
y_coordinate	5	108358
sd	4	112215
x_coordinate	3	113144
corr	2	115000
NDAI	1	152553

The AIC column represents the AIC of the model if it didn't have the variable. Therefore, we see actually that for example if we didn't include NDAI, AIC goes up to 152553 (lower AIC is better) thus making it the most important. From here we see, perhaps surprisingly, that corr is more important than x_coordinate but actually that x_coordinate is much more important than y_coordinate. This is expected as we saw going left and right is much more important than going up and down.

<u>**Question 2d) CVGeneric Function**</u>



~~~
```{r}
CV_generic <- function(classifier, training_features, training_labels, num_folds_k, loss_function) {
  training_features$labels <- training_labels
  folds <- createFolds(training_features$labels, k = num_folds_k)
  for (j in 1:num_folds_k) {
    valIndexes <- folds[[j]]
    CVTrain <- training_features[valIndexes, ]
    CVTest <- training_features[-valIndexes, ]
    model <- train(CVTrain[, names(training_features) != "labels"], CVTrain$labels, method = classifier, number = num_folds_k, metric =
"Accuracy")
    predicted <- predict(model, CVTest)
  }

  loss <- loss_function(as.numeric(CVTest$labels), predicted)
  return (loss)
  }
```
~~~

<u>**Question 3a) Modelling**</u>

For our models, because this is a binary classification problem, we will use KNN, logistic regression, SVM, and classification trees. These are all relatively free of assumptions except the logistic regression. Each model was trained using a 5-fold cross validation on the combination of our train and validation data set. The caret package train() was extremely helpful in this process and we report each training accuracy using the cross validation method and finally the test accuracy using the test set in table 2 below.

Table 2: Final Result of accuracies

	CV_Train_Accuracy	**Test_Accuracy**
KNN	0.976896	0.9703114
Logistic	0.8938436	0.8903133

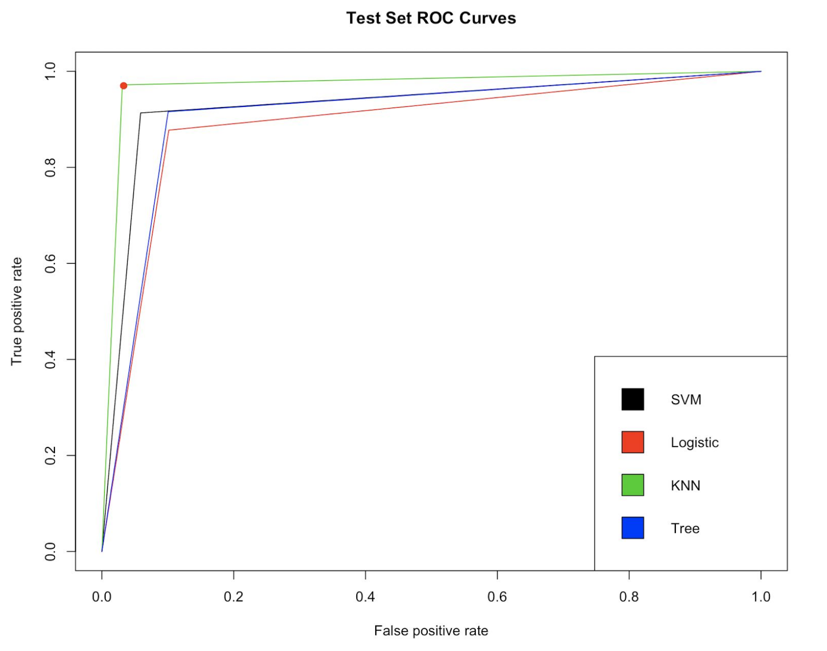
SVM	0.9680566	0.9700251
Classification Tree	0.9129354	0.9058102

We see that actually the KNN, a relatively simple but extremely powerful algorithm had the highest test_accuracy using only k = 3 as the hyperparameter which was obtained by the train() function. It is perhaps not so surprising that logistic didn't perform as well due to the outliers that might pull the coefficients to become unstable and also perhaps the large number of assumptions needed to satisfy this model. The SVM, which we used the radial kernel, also did extremely well to almost have the same test accuracy as the KNN. At this point it is hard to seperate the two methods. Lastly, the classification tree did not perform as well as we expected but that's perhaps it is hard to partition our space into discrete sections to fully distinguish the results.

<u>Question 3b) ROC Curves</u>

Prediction accuracy on the test set is a very powerful methodological evaluation of our models, but it does have its limitations. ROC curves are an alternative method to see which model actually best describes the tradeoff between specificity and sensitivity (respectively the type 1 and type 2 errors). Therefore we show the ROC curves below for all our different models.

<u>Figure 7: ROC Curves for different models</u>



We see here in figure 7 our smoothed ROC curves that the KNN again has the most optimal value (highlighted with a red dot for the cutoff) nearing 0.033 and 0.97 respectively for FPR and TPR. These rates are extremely good and is also expected given the above high classification success in table 2. Perhaps surprisingly the SVM is not nearly as good as our KNN despite having the SVM's prediction accuracy being so close. This is possible because SVM is quite sensitive to the cost parameter that was selected and the choice of kernel. KNN is a relatively robust methodology, which will of course be further explored in the diagnostics section below.

Question 4 Diagnostics:
First the best model that we chose is KNN. Luckily KNN is a model that is general and does not require a lot of assumptions like the logistic regression. Therefore, we have reason to believe that this performs quite well under robustness checks. The first thing is that the only parameter of interest in KNN is the “K” in the KNN algorithm which determines how many neighbors we want to consider. The correct convergence and optimal values of this is theoretically not well known and is similar to a black box. We will still explore the robustness of this parameter is several ways. First we will see how the cross validation results varied across K.

Figure 8: Cross Validated optimal K values for KNN

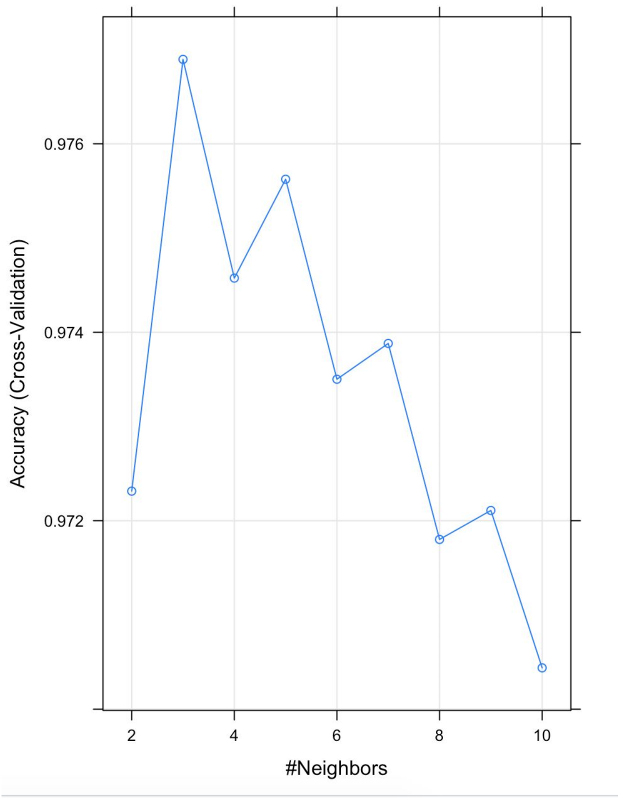


Figure 8 actually shows a very interesting pattern where it keeps going up and down alternating with a period of 2 but has a general pattern of going down. Ideally we would have wanted to test more values K, i.e. maybe K = 2-25, but because this algorithm was so computationally expensive it wasn’t necessary. Moreover, it is highly unlikely that somehow the optimal K will jump at around K = 25. K = 3 might also have a good interpretation. When one looks at figure 6, we can see there are places on the image where on the boundaries things like SVM might make a lots of mistakes. KNN, however, with low K (like K = 3) might be able to correctly classify by really just taking only a few neighbors instead of a lot.

However, to truly check the robustness of this K value we would like to see how this model performs on different training and testing sets. It is very possible that we just got lucky and perhaps under a different training and testing set our prediction accuracy and K values will be very different. There is actually lots of literature on this methodology of doing robustness checks on different training sets to see if the results are reproducible based on the same procedure. We will do exactly this by resplitting our training and testing data by setting a different seed every time 5 times and construct 5 different cross validated KNN models just like how we constructed our KNN model above. The following will show the result:

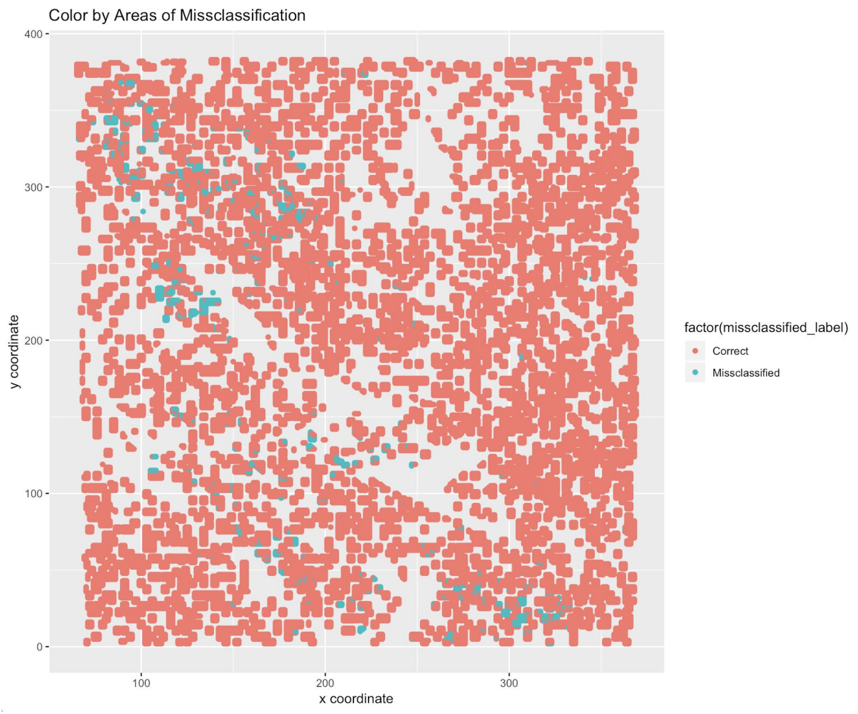
Table 3: Robustness check of KNN

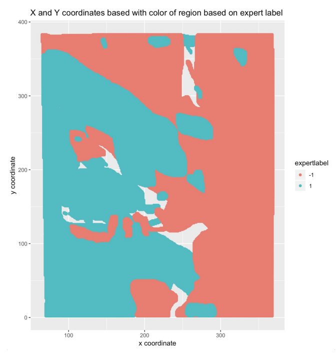
	Test Accuracy	Optimal K Values
Run 1	0.976573	3
Run 2	0.977032	3
Run 3	0.978523	4
Run 4	0.975632	3
Run 5	0.976153	3

We see in this table that the K value seems to be very robust and probably is the “right” optimal given any random subset of the proper i.i.d training set and testing set. There is actually only one value that went to K = 4. We also see very close test accuracy that converge around our original test accuracy of 0.976896. All of this serves to validate the robustness of our model given that we checked it for different random seeds of our splitting and thus we did not just get “lucky”.

Lastly, for our diagnostics we will try to figure where KNN is mostly misclassifying. It is possible KNN is always mistaken in a certain area of the xy plane. What if that area was specifically very important to certain groups of people, or maybe that area is actually the most inhabited place. We would like to run such diagnostics for these kind of motivations. The following figure shows such a figure.

<u>Figure 9: Misclassified Plot of KNN</u>





A very brief repetition of figure 6 is also reproduced on the right for easier comparison.

Figure 6 (on the right) represents the labels of cloudy and not cloudy data for all our combined data. We see in Figure 9 that as expected our model isn't doing as well on the boundaries and the random "island" like structure on the left side of the map. This is very expected as those points are literally in the middle of all the cloudy data and hence these areas are much harder to classify correctly. This is exactly what one would expect from an algorithm like KNN and this diagnostics just shows our model is really doing the best it can given such a "hard" dataset on these boundary cases. We would assume algorithms like SVM or logistic regression to do even worse on these boundary cases since our K = 3 was a relatively small value and it still performed poorly in these "islands". Luckily we do manage to classify many of the inner points inside these "islands" and hence can be considered quite successful in this regard.

Lastly, we do believe that given new data points in the future without the expert label we might be able to conclude with quite a big confidence that we can correctly classify it with our model. Assuming the expert labels we were given are correct and if we have the additional assumptions that all the points are uniformly randomly distributed in this grid, then we can actually say we expect about 97% of the time to guess correctly, which is quite a high rate. The above tests shows robustness of the parameters and test accuracies despite rerunning it with different sets of training and testing and in general the KNN is also a model with good properties that doesn't rely on heavy assumptions.

The only thing we would be very worried about is extrapolation of this model on datasets that we haven't seen. But this is a problem any model would face. For example if we were to see a data point in the future that wasn't in the range of our given x and y coordinates, then we are probably lost on what to do. For a model especially like KNN which requires on "nearest neighbors" it would perform much more badly than a more extrapolation based method like logistic. However, even regressions theoretically do not justify such high extrapolation.

Conclusion

We see from this report that the data we received were not i.i.d and more sophisticated methods for splitting were required such as blocking. Once we had the right splits we could perform modelling and evaluate these models based on certain criterias like ROC and prediction accuracies on the test sets. We see from these evaluations that KNN had by far the best performance and through the diagnostics we found the parameter of interest, K, and algorithm to be robust and stable after some sensitivity analysis. Moreover, we found misclassification rate to be quite low (around 3%) and thus can conclude this is a relatively strong model.

Acknowledgements

Minjeong did the summary, summarization of the data, visual and quantitative EDA of the dataset, splitting the data, ROC Curves
Timlan did the feature importance, reporting the accuracy of the trivial classifier, CVgeneric function, trying several classification methods and accessing their fit using CV
Both worked on the diagnostics and reproducibility together

Resources

https://www.rdocumentation.org/packages/stats/versions/3.6.0/topics/step (step function)
https://www.rdocumentation.org/packages/caret/versions/4.47/topics/train (caret package)
https://drive.google.com/file/d/1xqv1wKzPFiq-slukgIUmUSFa0hlIHIpJ/view (HW5 solutions, specifically reviewing the CV procedure)
http://topepo.github.io/caret/index.html (caret package uses and examples)
https://medium.com/@Mandysidana/machine-learning-types-of-classification-9497bd4f2e14 (information on different classification methods)
https://rviews.rstudio.com/2019/03/01/some-r-packages-for-roc-curves/ (ROC curve information and examples)

Link to github:

https://github.com/realllllminj/Stat154Proj2