STAT 154: Project 2 Clouds Exploration In The Arctic

Yang Xiang 3034504578
Zhihua Zhang 3034504526

1 Data Collection and Exploration

(a) Summary of the paper:

As am important part of environmental protection, Earth's climate, takes up more and more attention. From the conclusion of Global climate models, it is very worthwhile to study the climate in the Arctic. Since the accuracy Arctic-wide measurements greatly is essential, the paper aims at proposing two new operational Arctic cloud detection algorithms.

To precisely ascertain the cloud pattern, the paper uses the data produced by MISR. Using nine cameras and four spectral bands, MISR collects useful and gigantic data from all 233 paths on a cycle of 16 days and each path is divided into 180 blocks.

At last, three physical features $CORR, SD_{An}$ and $NDAI$ indeed contain sufficient information. And the ELCM algorithm is more accurate, has better spatial coverage, and generates probability labels for partly clouds by training QDA.

Moreover, the paper not only provides a pratical method to detect cloud in the Arctic, but also set a good example for future statisticians use their power of statistical thinking to influence the world.

(b) Summary of data, map plots and pattern observation:

(i) : Summary of data:

For Image 1: clear, uncertain and cloudy regions are percents $43.78\%, 38.46\%, 17.76\%$.

For Image 2: clear, uncertain and cloudy regions are percents $37.25\%, 28.64\%, 34.11\%$.

For Image 3: clear, uncertain and cloudy regions are percents $29.29\%, 52.27\%, 18.44\%$.

(ii) : Maps Plots:

(iii) : Pattern observation:

For Image 1: cloud-free regions are mostly in the northeast, cloudy surface locate mainly in the southwest and the uncertain parts, which transits from one to another by intuition, indeed lies between them.

For Image 2: cloud-free regions are mostly in the north, while cloudy surface locate mainly in the south. And uncertain parts are also in the middle.

For Image 3: Similarly to former two images except cloud-free regions are northwest and locate southeast.

Thus, there seems to have intrinsic geographic causes, which implies the $i.i.d$ assumption fails here. Otherwise, we should see a disorderly overlap between cloudy and cloud-free areas.

(c) Visual and quantitative EDA:

(i) : Pairwise relationship among features themselves:

For themselves: 'SD' and 'CORR' are unimodal, but "NDAI" is bimodal, which is a good signal about separability of clouds.

For relationship between each other:(1)There is a significant positive correlation between "NDAI" and "SD"; (2)The five radiance angles are positively correlated to each other; (3)The negative correlation between 'SD', 'CORR' and "NDAI" with radiance gets stronger as the angles gradually come back to nadir direction.

(ii) : Relationship between features the expert labels with the individual features:

For "NDAI" and "SD": The separation standard remains same for three images, which is low level indicates clear weather and high value means cloudy areas.

For "CORR": The decision rule seems to be unstable, which is similar to "NDAI" and "SD" in Image 2 and Image 3. In Image 1, clear and cloudy regions have highly overlapped "CORR" distribution.

For five radiances: "AF" and "AN" have steady decision boundary in three images, but others do not. And the criteria is contrary to the previous one.

(iii) : Differences notification:

By EDA about these three images, we observe that on the whole, low levels of "NDAI", "SD" and "CORR" imply the clear areas, whereas high levels of five radiation signify the clear regions. However, it is not immutable, we can observe that there is variation among different images.

2 Preparation

Now that we have done EDA with the data, we now prepare to train our model.

(a) Split the entire data in non-trivial ways:

(i) Method 1:

First we decide the proportion of data assigned to three sets: training 60%, validation 20%, and test 20%. The reason is that we truly want to have a lot of test and validation data to ensure robustness, but we still need enough data to train which makes our model grow healthily.

Then, considering the existence of spatial dependency, our data is not i.i.d here. We will lose or even get wrong information if we do ramdom spliting imprudently, because we may destroy the dependent structure. In this way, to remain this dependency, we divide our data to block/cluster, then do shuffling and sampling. The gist is that we find if we find weather is cloudy at some position, its neighborhood

inclines to be cloudy as well, and so does the clear area. Thus, block-sampling may work well in a way, and for reproductivity, we set random seed as '2019' here.

Furthermore, letting blocks have same size is unapt because the number of pixels contained in each block is different. Thus, we determine the block size according to quantiles of coordinate x and y. For example, $block_{ij}$ has length $q_{x,\frac{i}{n}} - q_{x,\frac{i-1}{n}}$ and width $q_{y,\frac{j}{n}} - q_{y,\frac{j-1}{n}}$, where n^2 is the total blocks. Here, we choose $n^2 = 225$.

(ii) Method 2:

We still follow the rule that spliting data as training, validation and test sets with size 60%, 20% and 20% respectively. Moreover, we believe that the block-sampling trick captures the dependency correctly.

Thus, we only need to consider the hyperparameter, total number of blocks, which is crucial towards training a good. In method 1, we choose $n^2 = 225$, so each block has about 700 pixels, which may be too large. To distinguish the influence caused by n^2, we here let $n^2 = 625$, which gives significantly small block.

(b) Accuracy of a trivial classifier:

(i) Method 1:

For Image 1: Validation accuracy and test accuracy are 66.95% and 72.42%; For Image 2: Validation accuracy and test accuracy are 52.16% and 46.43%; For Image 3: Validation accuracy and test accuracy are 59.87% and 64.47%.

Comparing the results of trivial classifier in different images and the images' own conditions. We conclude that in the scenario where one class dominates the other, trivial clasifier would behave well, otherwise, it may not work well. Thus, by the performance of image 2 and image 3, we know the classification problems are not trivial here.

(ii) Method 2:

For Image 1: Validation accuracy and test accuracy are 67.56% and 73.41%; For Image 2: Validation accuracy and test accuracy are 51.31% and 52.65%; For Image 3: Validation accuracy and test accuracy are 64.13% and 57.19%.

We observe that the results from three images are still not plausible, so we again confirm the classification problems are not trivial here.

(c) Suggest three of the "best" features:

(i) Method 1:

First, we use all the features to run a classification model, then we check the feature importance, which is the t-value of each feature. We choose first three important features and just include them to run the same classification model again to observe the improvement. We have indeed tried "lda", "qda", "logistic" and "SVM" here, since they give bascially the same information, we just show result about "lda" to be succinct. And following the convention not to use test set, we train model using training set and evaluate in validation set.

For image 1: we find "NDAI", "SD" and "Ra_DF" are three most important features, which are slightly different from the papers with replacing "CORR" by "Ra_BF". And we visualize the classification result and observe it is great; For image 2: we find "CORR", "NDAI" and "SD" are three most important features. This time the result coincides with paper's. And again the classification result is pretty good; For image 3: we find "NDAI", "SD" and "CORR" are three most important features, which also accord with the paper. And again the classification result is convincing.

(ii) Method 2:

We do similar process to select the first three important features and rerun the same classification model using only them to compare. The difference here is that our split data changes, each block is smaller.

The results are basically same, but change a little. For image 1: we find "NDAI", "SD" and "Ra_CF" are three most important features, which are slightly different from before with replacing "Ra_BF" by "Ra_CF". The classification visualization is great; For image 2: we still find "CORR", "NDAI" and "SD" are three most important features. Again the classification result is good; For image 3: we find "NDAI", "SD" and "CORR" are three most important features, which also keeps same. And the classification result is not bad.

(d) $CV_{generic}$ R function:

3 Modeling

In this part, we try five different models: LDA, QDA, Logistic Regression, Linear SVM and kernel SVM with Gaussian radial basis function. From part 2 and paper, we know that NDAI, SD and CORR are the most informative features, so we train and predict using these three features in this part.

(a) Model assumptions and assess model by cross validation and test error:

(i) Model assumptions:

LDA and QDA both have generative model assumptions, which is that data from each class are multivariate normal. Additionally, for LDA we assume that data from different classes have the same covariance matrix, while for QDA we assume that data from different classes have different covariance matrixes. To check whether our data is multivariate normal, we perform Mardia's MVN test on our data. Note that here we only perform test on a subset of our data by taking a sample without replacement of our data with size 5000, since function mvn in R package MVN requires that sample size between 3 and 5000. The p-values of Mardia Skewness and Mardia Kurtosis are less than 0.001, which means that our data is not approximately multivariate normal. We also take a look at Chi-square Q-Q plot and univariate histograms in Figure 1. From Chi-Square Q-Q Plot we can find that many deviations occur due to outliners, and from univariate histograms we can find that NDAI looks bimodal, and SD and CORR are right-skewed. Therefore,

our data may not be multivariate normal. Then we also take a look at the covariance matrixes of two classes with the whole training set in Table 1, and they appear similar but not the same. Considering that data in practice are usually not perfect, we still do LDA and QDA and see how well they perform.

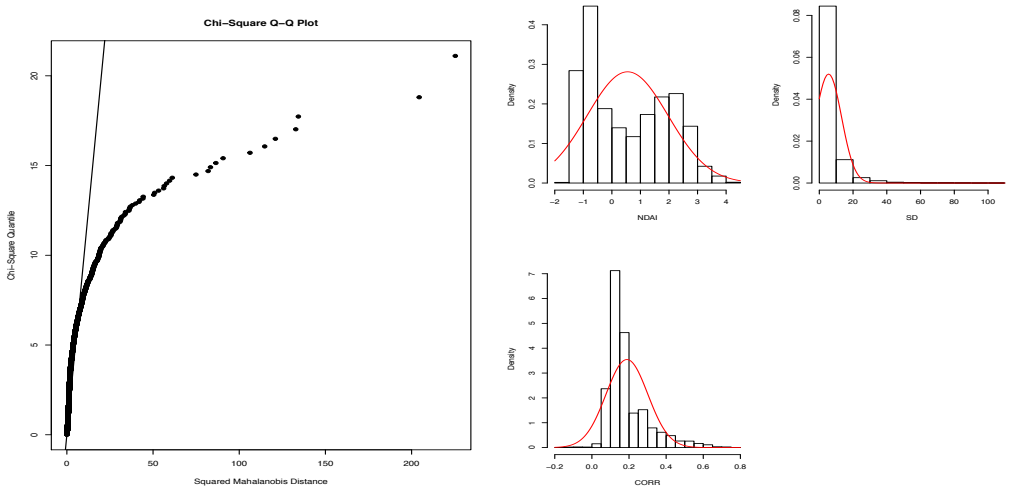

Figure 1: Left: Chi-Square QQ plot. Right: univariate histograms.

	NDAI	SD	CORR
NDAI	0.44	2.33	0.02
SD	2.33	66.84	0.24
CORR	0.02	0.24	0.02

	NDAI	SD	CORR
NDAI	1.12	4.14	0.02
SD	4.14	35.22	0.09
CORR	0.02	0.09	0.01

Table 1: Covariance matrixes. Left for label 1 (cloud), right for label -1 (not cloud).

For logistic regression for two classes, we assume that $P(y=1|x) = \frac{e^{w^\top x+b}}{1+e^{w^\top x+b}}$ and $P(y=-1|x) = \frac{1}{1+e^{w^\top x+b}}$, which can be seen as approximately satisfied considering that we are doing 2-class classification.

For SVM, there are no probabilistic assumptions, but we have to choose hyper-parameters here. Since training SVM is very time-consuming, we only try several values here and pick the best one based on cross validation. For Linear SVM, we need to choose hyper-parameter C. We try three values $\frac{1}{3}, 1, 3$ for it and find that $C = 1$ generates the largest CV accuracy, so we choose $C = 3$ for Linear SVM. For Kernel SVM with Gaussian radial function, we need to choose two hyper-parameters C and γ. We try every combination of $C, \gamma \in \{\frac{1}{3}, 1, 3\}$ and find that $C = 1, \gamma = \frac{1}{3}$ generates the largest CV accuracy, so we choose $C = 1, \gamma = \frac{1}{3}$ for Kernel SVM.

(ii) Cross validation and test accuracy

By results of cross validation in Table 2,3 and test accuracy in Table 4, we can find that CV accuracy and test accuracy of all models are around 0.9, and it is difficult

to further increase the CV/test accuracy. Besides, Kernel SVM performs best with largest CV accuracy and test accuracy.

	Accuracy Across Folds										CV Accuracy
LDA	0.876	0.968	0.898	0.891	0.897	0.877	0.894	0.880	0.896	0.923	0.901
QDA	0.903	0.960	0.889	0.858	0.879	0.889	0.892	0.884	0.875	0.892	0.893
Logistic Regression	0.865	0.959	0.875	0.877	0.897	0.878	0.891	0.879	0.899	0.899	0.892
Linear SVM	0.872	0.968	0.893	0.894	0.898	0.875	0.896	0.880	0.896	0.925	0.900
Kernel SVM	0.938	0.982	0.928	0.961	0.953	0.951	0.949	0.951	0.932	0.980	0.952

Table 2: Cross validation with split method 1

	Accuracy Across Folds										CV Accuracy
LDA	0.894	0.884	0.926	0.924	0.902	0.882	0.911	0.881	0.909	0.897	0.901
QDA	0.880	0.861	0.912	0.907	0.901	0.873	0.908	0.873	0.923	0.901	0.894
Logistic Regression	0.887	0.878	0.918	0.917	0.891	0.875	0.906	0.870	0.895	0.894	0.893
Linear SVM	0.896	0.885	0.925	0.926	0.901	0.882	0.910	0.879	0.907	0.897	0.901
Kernel SVM	0.953	0.955	0.976	0.968	0.959	0.943	0.947	0.931	0.972	0.955	0.956

Table 3: Cross validation with split method 2

	LDA	QDA	Logistic Regression	Linear SVM	Kernel SVM
Test Accuracy	0.8826	0.8920	0.8739	0.8842	0.9122

Table 4: Test Accuracy

(b) ROC curves

By ROC curves in Figure 2 we can find that all models perform well since all of them have AUC around 0.95, while Kernel SVM appears to perform best with a largest AUC=0.9608.

Here, we choose the cutoff value to be 0.5. For one thing, it is reasonable and natural to classify one observation to be cloud when the probability of being cloud is larger or equal to 0.5. For another, from plots we can see that by choosing the cutoff value 0.5 we can maintain a relatively high True Positive Fraction while keeping False Positive Fraction relatively low.

(c) Assess models by Confusion Matrix and F1 score

By confusion matrixes of all methods in Table 5 we can find that Kernel SVM has a significant small False Negative value compared with other methods.

F1 score, defined by the harmonic average of the precision and recall, is a measure of accuracy. With results in Table 6 We find that Kernel SVM has the largest F1 score.

Figure 2: ROC curves

LDA		Actual Label	
		Cloud	Not Cloud
Predicted Label	Cloud	16089	2777
	Not Cloud	2412	22914

QDA		Actual Label	
		Cloud	Not Cloud
Predicted Label	Cloud	15969	2240
	Not Cloud	2532	23451

Logistic Regression		Actual Label	
		Cloud	Not Cloud
Predicted Label	Cloud	15537	2608
	Not Cloud	2964	23083

Linear SVM		Actual Label	
		Cloud	Not Cloud
Predicted Label	Cloud	16402	3017
	Not Cloud	2099	22674

Kernel SVM		Actual Label	
		Cloud	Not Cloud
Predicted Label	Cloud	17534	2912
	Not Cloud	967	22779

Table 5: Confusion Matrixes

	LDA	QDA	Logistic Regression	Linear SVM	Kernel SVM
F1 Score	0.8611	0.8700	0.8480	0.8651	0.9004

Table 6: F1 scores

4 Diagnostics

In previous part, we find that Kernel SVM with Gaussian radial function appears to perform best among all the models, so we do diagnostics for Kernel SVM in this part.

(a) Diagnostic plot

From the plot of coefficient convergence in Figure 3, though we can not determine

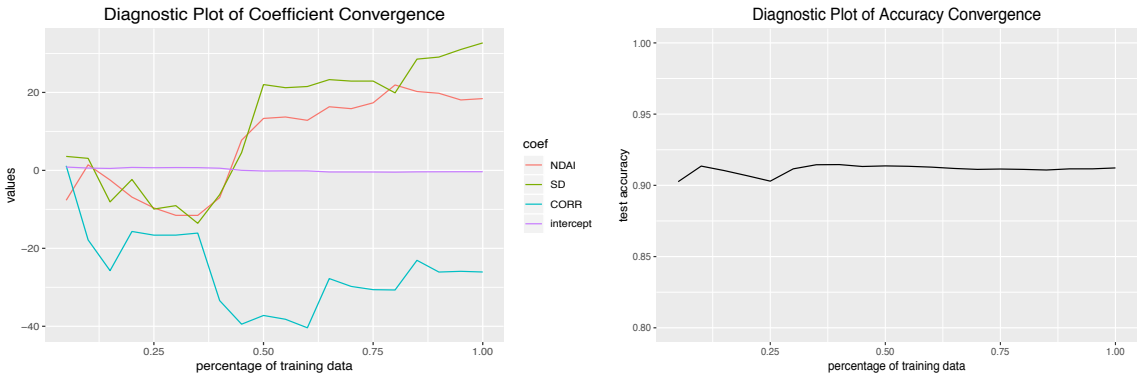

Figure 3: Left: Plot of coefficient convergence. Right: Plot of accuracy convergence.

whether the coefficients are going to converge, we can find that they appear to be less volatile (less variance) as we increase traing data.

From the plot of accuracy convergence in Figure 3, we can find that the test accuracy gradually converge to around 0.91.

(b) Analysis of misclassified data

First, we plot correct and wrong classified data with respect to x and y coordinates in Figure 4 (including both training and test set). Some blank area occur since we exclude unlabeled data before training and testing models, so they do not show up. The misclassified data are mainly in lower triangular part of the plot, but we do not find a specific pattern here.

Then we plot the histograms of every feature, and find it interesting for NDAI in Figure 5. Here we plot histograms of NDAI with respect to right/wrong classified data for training set and test set separately. We can find that the misclassified data have NDAI mainly between 0 and 4, and for data with NDAI less than 0, we almost correctly classify all of them.

(c) A better classifier, mixture of models

In previous part, we can see that a specific model has its limitation. So we design a mixture of models and see how it performs. Basically, we let each model vote for

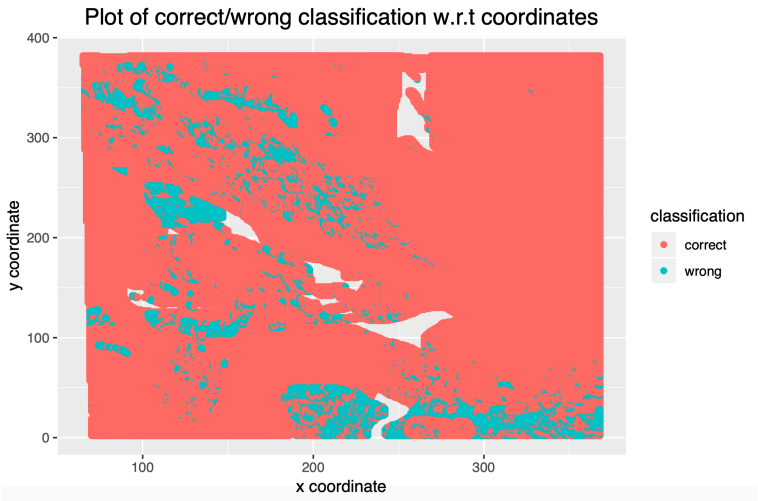

Figure 4: Plot of correct/Wrong classified data with coordinates

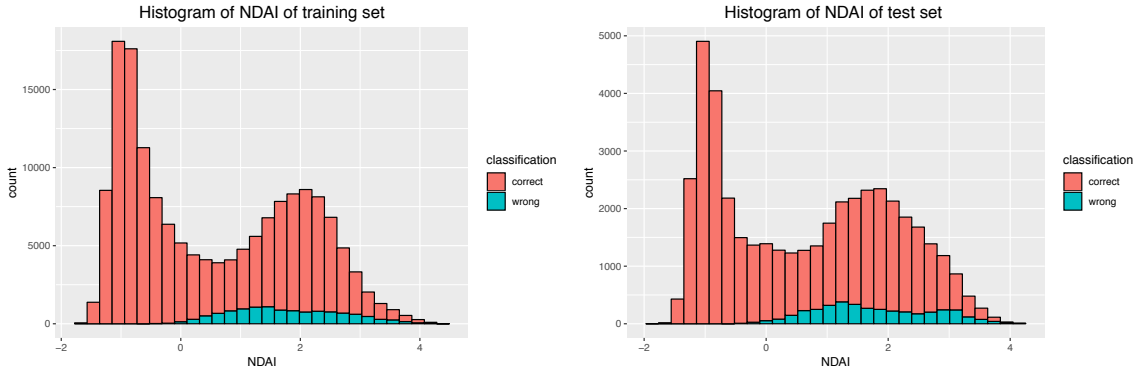

Figure 5: Left: Histogram of NDAI of training set. Right: Histogram of NDAI of test set.

classification based on its results and each model has a weight for its vote. Based on their performance in part 3, we choose a mixture of QDA and kernel SVM and set the weight to be $0.5, 0.5$ respectively, which sum up to one. We classify an observation to be cloud $(+1)$ if it gets score larger or equal to 0.5. And note that we should not assign a weight larger than 0.5 for one model, otherwise that model will dominate and the mixture will be no different from a single model. We also do cross validation in Table 7 and find test accuracy is 0.9128 for our mixture model, which outperforms every single model. Therefore, we expect our model to outperform every single model in part 3 on future data.

(d) The results remain almost same as we change the way of splitting data as shown in figure 6.

(e) Conclusion
In this part, we first do diagnostic analysis on our best model Kernel SVM and find that

	Accuracy Across Folds										CV Accuracy
Mixture Model	0.941	0.980	0.933	0.965	0.951	0.955	0.950	0.952	0.934	0.976	0.955

Table 7: Cross validation of mixture model

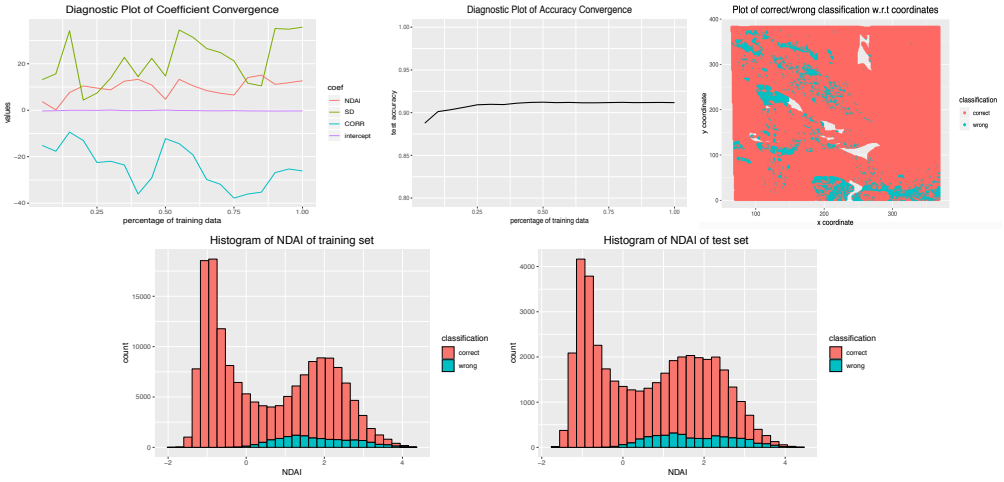

Figure 6: Plots when changing the way of splitting data

parameters seem to get stable and less volatile and test accuracy appears to converge as we increase training data. Then we find some certain patterns for our misclassified data. In (c), we come up with a new model, a mixture of all our models in part 3, and find it performs better.

5 Reproducibility

In addition to a writeup of the above results, please provide a one-line link to a public GitHub repository containing everything necessary to reproduce your writeup. Specifically, imagine that at some point an error is discovered in the three image files, and a future researcher wants to check whether your results hold up with the new, corrected image files. This researcher should be able to easily re-run all your code and produce all your figures and tables. This repository should contain:

(i) The pdf of the report,

(ii) the raw Latex, Rnw or Word used to generate your report,

(iii) your R code (with CVgeneric function in a separate R file),

(iv) a README file describing, in detail, how to reproduce your paper from scratch (assume researcher has access to the images).

You might want to take a look at the GitHub's tutorials

Acknowledge section

In the process of completing this project, we discuss, interchange our thoughts and to be more efficient, we divided to focus on different parts. The specific allocation about five sections is listed as following:

Section 1: Zhihua Zhang

Section 2: Zhihua Zhang

Section 3: Yang Xiang

Section 4: Yang Xiang

Section 5: Zhihua Zhang & Yang Xiang

During finishing this project, we refer to different resources like R document *train*[1] , *trainControl*[2] and *trainModelList* [3] for help.

To proceed our project, first, to have a domain background sketch, we read the reference paper carefully, exchange our views and try to reach accordance. Because the data is cleaned here, we then begin to visualize and do EDA, whereby we verify that there exists spatial dependency here. Considering the existence of non i.i.d data, instead of splitting original data randomly, we do block-sampling to divide data. To be specific, block-sampling includes partitioning data into blocks to remain dependency and random sampling. After that, we check the performance of trivial classifier and find out the problem at hand is non trivial. To obtain higher accuracy, we need good features, so we then select three best features through quantitative and visual justification, which are $NDAI$, SD and $CORR$. We all know CV is a terrific way to evaluate our model, and a customized generic could do us a lot of favor, so we design our own CV function before starting to train our models.

[1] Rdocument-train. `https://www.rdocumentation.org/packages/caret/versions/4.47/topics/train`.

[2] Rdocument-traincontrol. `https://www.rdocumentation.org/packages/caret/versions/6.0-82/topics/trainControl`.

[3] Rdocument-trainmodellist. `http://topepo.github.io/caret/train-models-by-tag.html#Support_Vector_Machines`.