Introduction to machine learning final report

Yinong Li - Group 109

1. Introduction

The formation of new particles, is the process by which small clusters of atoms or molecules come together to form new particles, such as aerosols or droplets. Several factors can affect the rate of nucleation, including temperature, humidity, and the presence of certain chemicals in the atmosphere. Several factors can influence the occurrence of this phenomenon, including temperature, humidity and the presence of certain chemicals in the atmosphere [1].

The goal of this project is to use the data from the monitoring stations to train a classification model that can predict whether the event will occur or not. We need to do two tasks: one is a binary classification task (identifying "event" and "nonevent") and the other is a multiclass task (subdividing events into "Ia", "Ib" and “II”).

2. Preliminaries

Before training a specific classification model, we need to analyze and process the data.

2.1 Data Cleaning

Before feature selection, a simple cleaning of the data is performed. The following steps are:

- Remove the “ID”, “date” and “partlybad” columns.
- Add a "class2" column to the data, which is set as “nonevent” where “class4” is “nonevent”, or “event” where “class4” is “Ia”,“Ib” or “II”.

2.2 Normalisation

Based on the observation of the dataset, we found that the scale of the features have relatively large differences, so in order to improve the accuracy of the model as well as to avoid numerical problems, we normalize the training samples. So for some models, we used the standard scaler to transform the data, which fits data to have a mean of 0 and standard deviation of 1. It is worth noting that we also need to perform the same

normalization process on the test data when predicting (mean and standard deviation from the training set).

2.3 Feature Correlation based Feature Selection

First we start by visualizing the correlation between each feature and the label to see which features barely affect the value of the label (see Appendix Figure A.1). The results showed that there were many variables with correlation coefficients less than 0.1 with the labels, so we then set a threshold value of 0.1 to filter these features (see Appendix Figure A.2).

2.4 Multicollinearity based Feature Selection

Multicollinearity is a situation in which two or more features are correlated with each other. Although correlations between independent and dependent features are desired, in some cases multicollinearity of independent features is less desired. In fact, they can be omitted because they are not necessarily more informative than the features with which they are correlated [2]. Based on this knowledge, we perform a multicollinearity analysis by visualizing the correlation matrix of the training data (see Appendix Figure A.3). Through it, many redundant features can be found in the features. Subsequently, we set up the threshold (0.98) to filter them (see Appendix Figure A.4).

3. Binary Classifiers

For the binary classification task (labeled 'class2') we chose four classification models, which are decision tree, Naive Bayes, KNN and logistic regression. We then briefly describe the algorithms and our experimental results.

3.1 Decision tree

Description: A decision tree is a tree structure in which each internal node represents a judgment on an attribute, each branch represents the output of a judgment result, and finally each leaf node represents a classification result. The decision tree generation algorithms are ID3, C4.5 and C5.0 etc.

Pros and Cons: Decision trees have two advantages: first, the resulting model is easy to visualize and understand by non-experts, and second, the algorithm is completely independent of data scaling. Since each feature is treated separately and the division of the data is not dependent on scaling, the decision tree algorithm does not require feature pre-processing, such as

normalization and normalization. And its biggest drawback is that it is easy to over-fit. It leads to the actual prediction is not good.

Hyperparameter Tuning: The parameter that controls the complexity of the decision tree model is the pre-pruning parameter, which stops the tree construction before the tree is fully expanded. In general, choosing a pre-pruning strategy (setting *max_depth*, *max_leaf_nodes* or *min_samples_leaf*) is sufficient to prevent overfitting. So here we use 10-fold cross-validation to adjust the hyperparameter *max_depth*. Figure 3.1 shows the depth parameter tuning, and the change in the 10-fold CV 0-1 accuracy as the value of the depth varied. The *max_depth* which maximize the 10-fold CV 0-1 accuracy is 3 (maximum accuracy is 83.0%). Thus we build up a decision tree model with *max_depth* = 3 to do binary classification.

Results: We then predicted the model on the real data ***npf-test*** dataset, and we obtained an accuracy of 85%. Figure 3.2 and Figure 3.3 show ROC curve and confusion matrix based on test data, and we also visualizes the decision tree we built in this case (see Appendix Figure A.5).

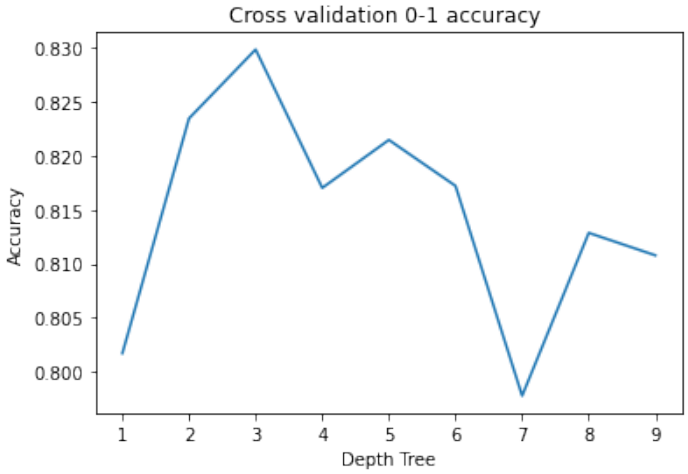

Figure 3.1: CV binary accuracy

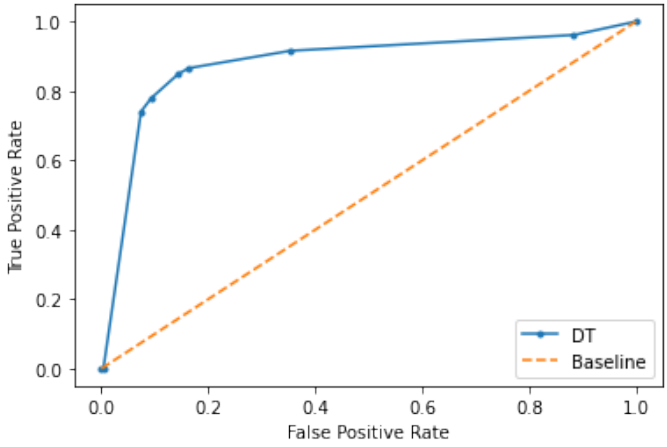

Figure 3.2: ROC curve

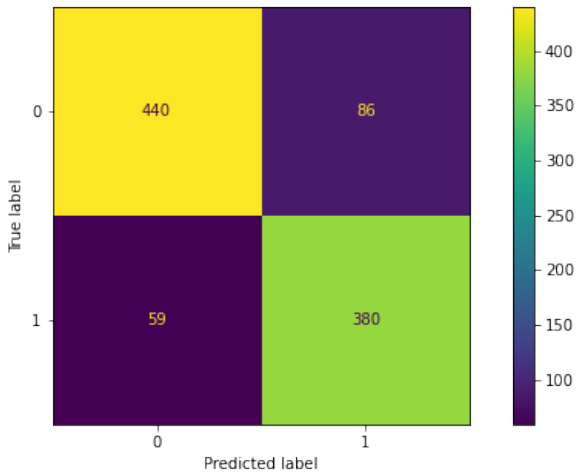

Figure 3.3: Decision tree confusion matrix

3.2 Naive bayes

Description: The classification strategy of the Naive Bayesian classifier has two main points: 1. Bayesian rule. 2. Conditional independence assumption. All parameters of the model are derived from the data (likelihood and prior). When we get all the required parameters we substitute them into the Bayesian formula to get the result.

Pros and Cons: The main advantage of the model is that the algorithm logic is simple and easy to implement. In theory, Naive Bayes model has the smallest error rate compared to other classification methods. But this is not always the case in practice. This is because the Naive Bayes model assumes that the features are independent of each other. This assumption is often not true in practical applications. When the number of features is large or the correlation between attributes is large, The classification effect is not good.

Hyperparameter Tuning: The only hyperparameter that need to be tuned in the Naive Bayesian classifier is the coefficient of Laplace smoothing, and the default value is widely considered to be the best choice, so we only use the default value in our experiments.

Results: We normalized the data before training the classifier, and we conducted experiments on different types of classifiers. The results of the experiments are shown below.

	Training-accuracy	Testing-accuracy	10-CV-accuracy
GaussianNB	0.799569	0.796891	0.776364
BernoulliNB	0.775862	0.789637	0.773728

Figure 3.4: Naive Bayesian classifier accuracy

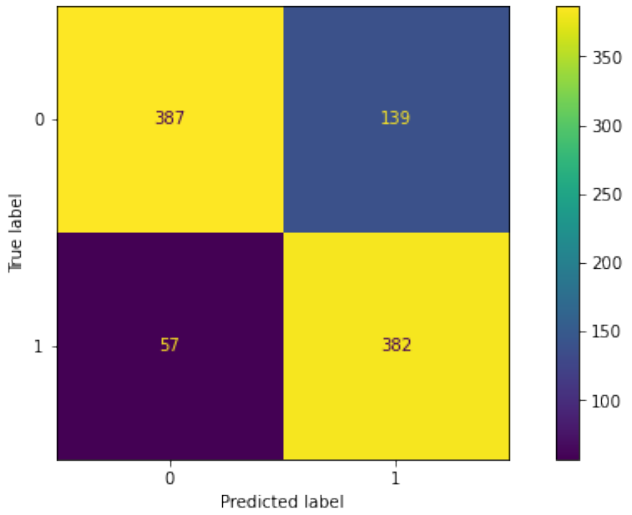

Figure 3.5: Prediction of GaussianNB

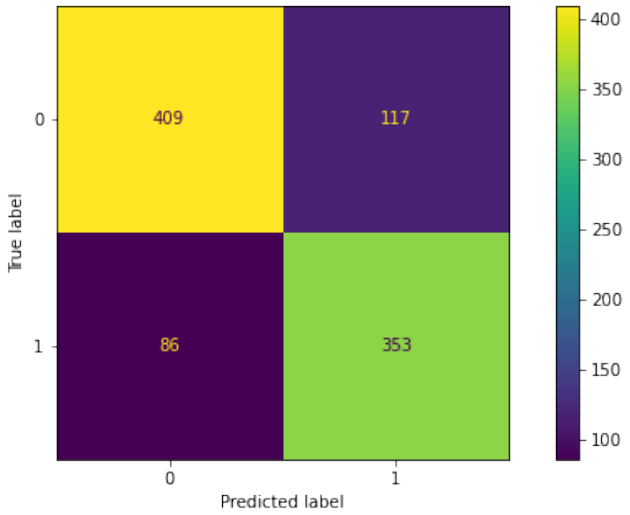

Figure 3.6: Prediction of BernoulliNB

3.3 K-nearest neighbors

Description: The obvious difference between KNN and other classifiers is that it has no training data, each new input calculates the distance to all data points, and the mode of the labels in the k closest data points is selected as the classification result.

Pros and Cons: Its main advantages are: 1. no assumptions on data, high accuracy and insensitivity to outliers compared with algorithms like Naive Bayes. 2. Mature theory, simple idea, can be used for both classification and regression. The disadvantage is that it is too computationally intensive, especially when the number of features is very large, and the prediction accuracy for rare categories is low when the samples are not balanced.

Hyperparameter Tuning: We select the optimal model in our experiments by tuning the size of k (the number of data points for reference). We also apply cross-validation to select the best hyperparameters, and the experimental results are shown in Figure 3.7.

Results: We normalized the data before training the classifier. Experiments show that the classifier achieves 85% accuracy on the test data. The results of the experiments are shown below.

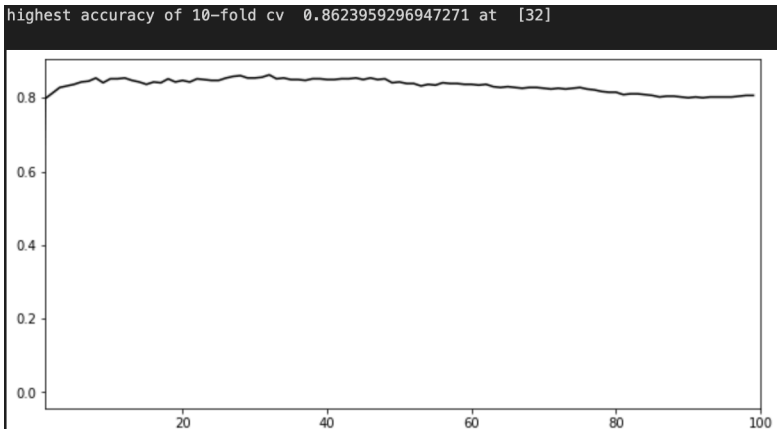

Figure 3.7: CV binary accuracy

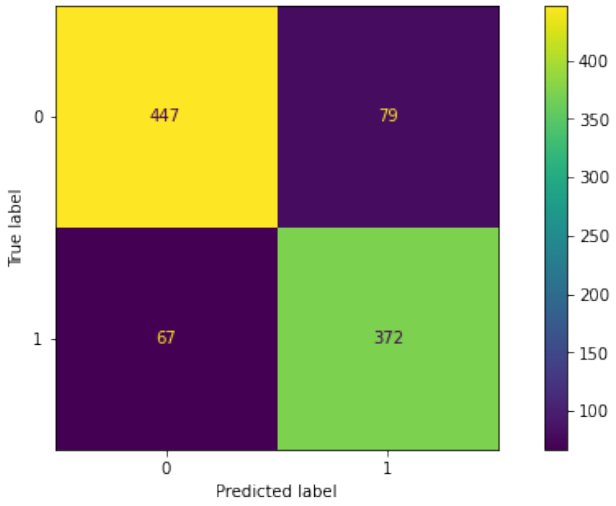

Figure 3.8: Prediction of KNN

3.4 Logistic Regression

Description: Logistic regression is a statistical method used to predict binary outcomes based on one or more independent variables. It estimates the probability of a certain event occurring and is useful when there are many independent variables and few observations.

Pros and Cons: Pros: widely used, easy to implement, efficient, works well with a small number of observations, handles non-linear relationships. Cons: does not perform well when the classes are well separated, does not provide good probability estimates when the classes are imbalanced.

Hyperparameter Tuning: In our experiments we tuning the values of the penalty terms of Lasso and Ridge, and then select the best model based on the accuracy of cross-validation. As we see in Figures 3.9 and 3.10, the Lasso model with C=0.8 is the optimal model.

Results: The experiments show that the optimal logistic regression model achieves an accuracy of 86% on the test set, and the following prediction results are presented.

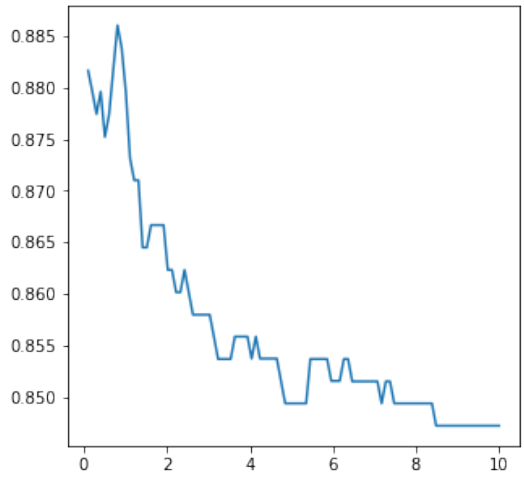

Figure 3.9: CV accuracy of Lasso

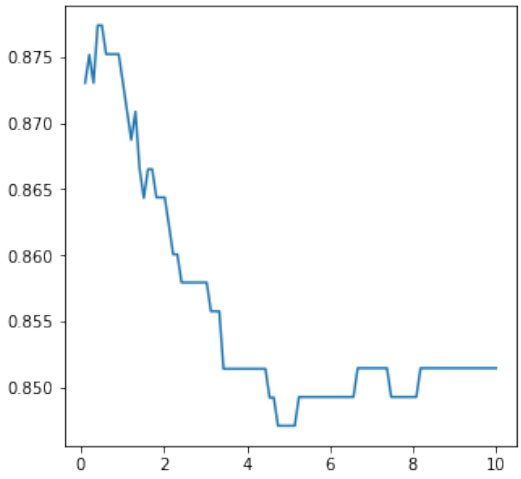

Figure 3.10: CV accuracy of Ridge

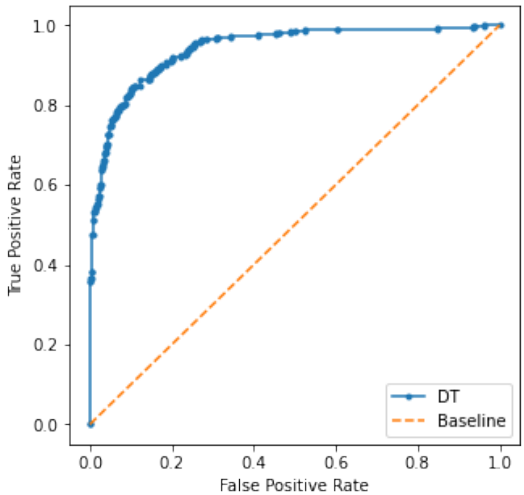

Figure 3.11: ROC curve

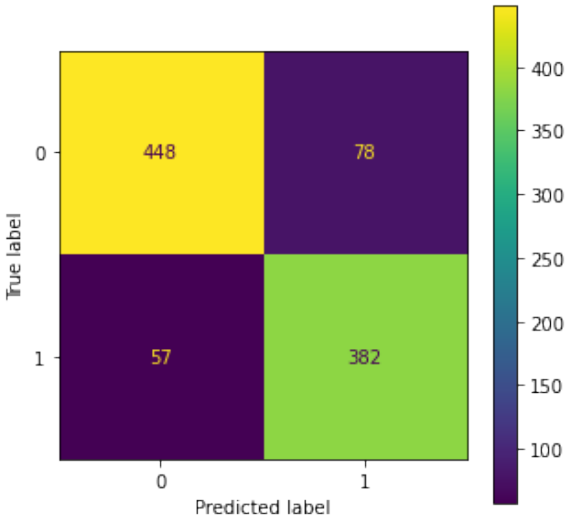

Figure 3.12: Prediction of LR

4. Milticlass Classifiers

In addition to binary classification, we also built some classifiers for multiclass tasks (predicting 'class4'). And except classifiers we mentioned before, we added a Multi-layer perceptron classifier to do that.

4.1 Decision Tree

The decision tree model is constructed in a similar way as in the case of binary classification, where the best hyperparameter *max_depth* is first selected based on the cross-validation accuracy, and then predictions are made and accuracy is calculated on the test set.

In summary, the results show that the decision tree is the best model when max_depth is 2, the accuracy on training data is 66% and the accuracy on testing data is 67%. The Figures below show the experimental procedure of hyperparameter tuning and the experimental results on the test set.

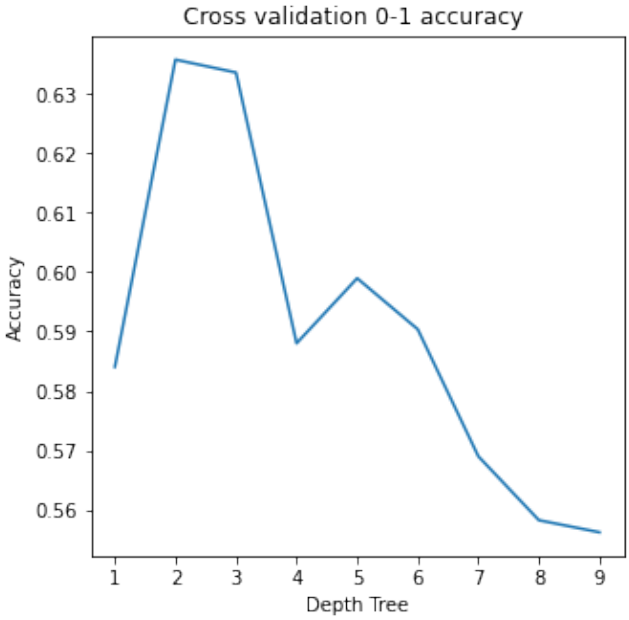

Figure 4.1: CV accuracy of decision tree

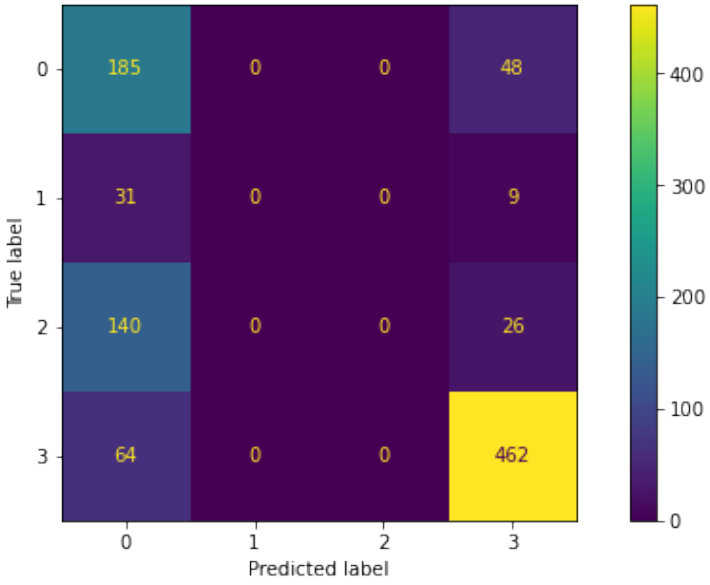

Figure 4.2: Prediction of decision tree

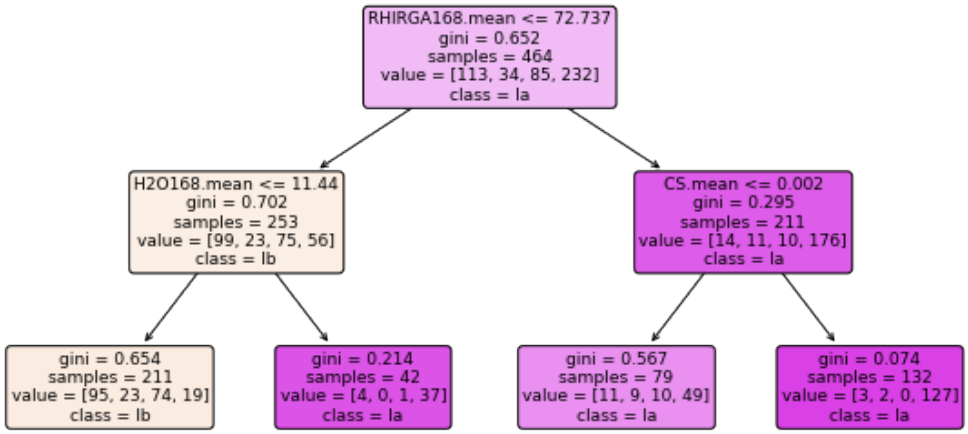

Figure 4.3: Decision tree for multiclass task

4.2 Naive bayes

Similarly, we experimented on two different types of Naive Bayesian classifiers and then selected the one with better accuracy to make predictions on the test set.

In summary, as we can see in Figure 4.4, the BernoulliNB is a better choice than GaussianNB on multiclass task. Figure 4.5 shows prediction results of the classifier.

	Training-accuracy	Testing-accuracy	10-CV-accuracy
GaussianNB	0.564655	0.565803	0.483117
BernoulliNB	0.584052	0.597927	0.553700

Figure 4.4: Accuracy of NB classifier

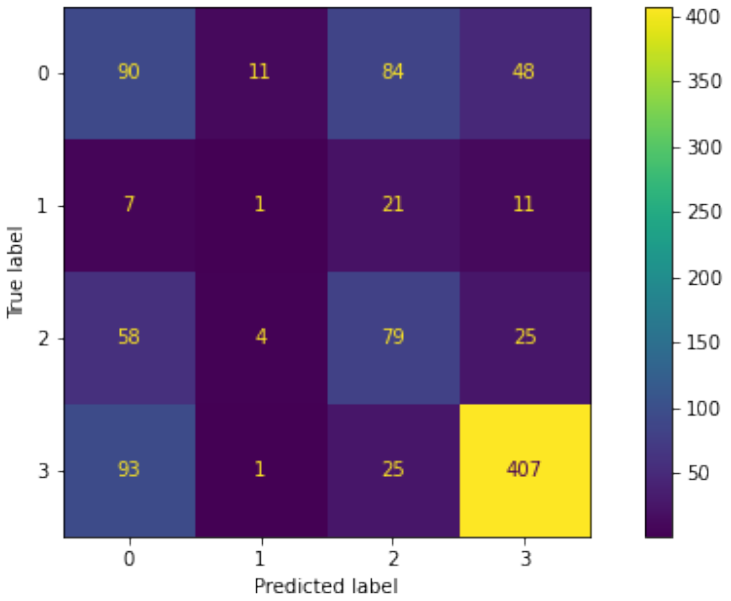

Figure 4.5: Prediction of BernoulliNB

4.3 K-nearest neighbors

First, we first find the optimal model by adjusting the size of the k-value and comparing the cross-validation accuracy. Then we let it make predictions on the test set and calculate the accuracy.

In summary, we can take the optimal KNN model at k of 40, which can achieve 66% and 67% accuracy on the cross-validation and test set. We can see the results clearly in Figure 4.6 and 4.7.

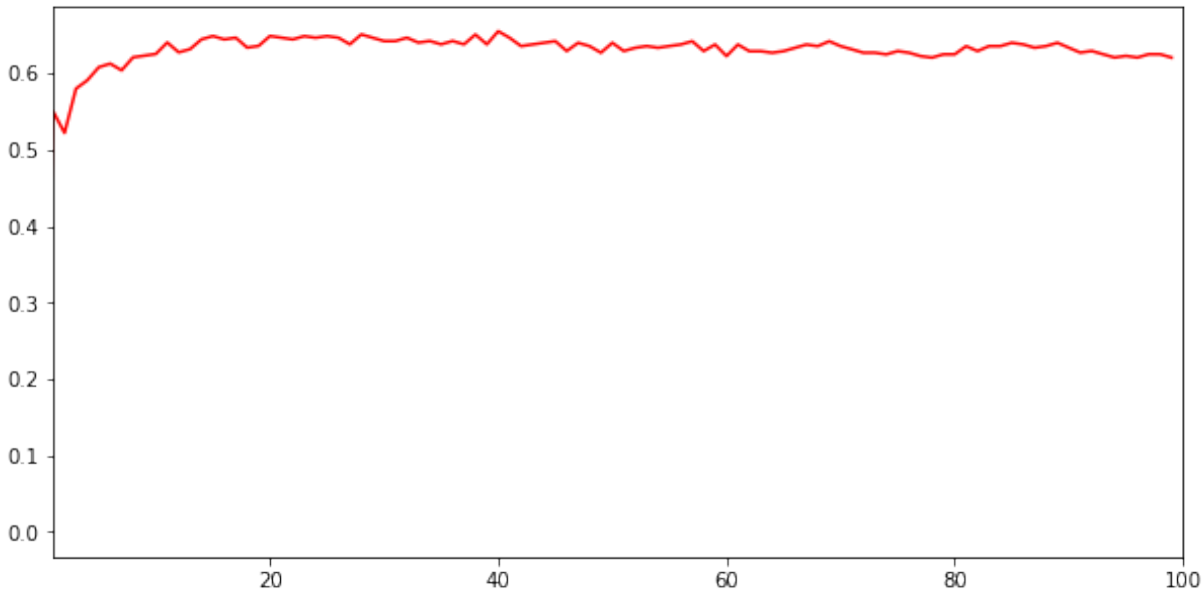

Figure 4.6: CV accuracy of KNN

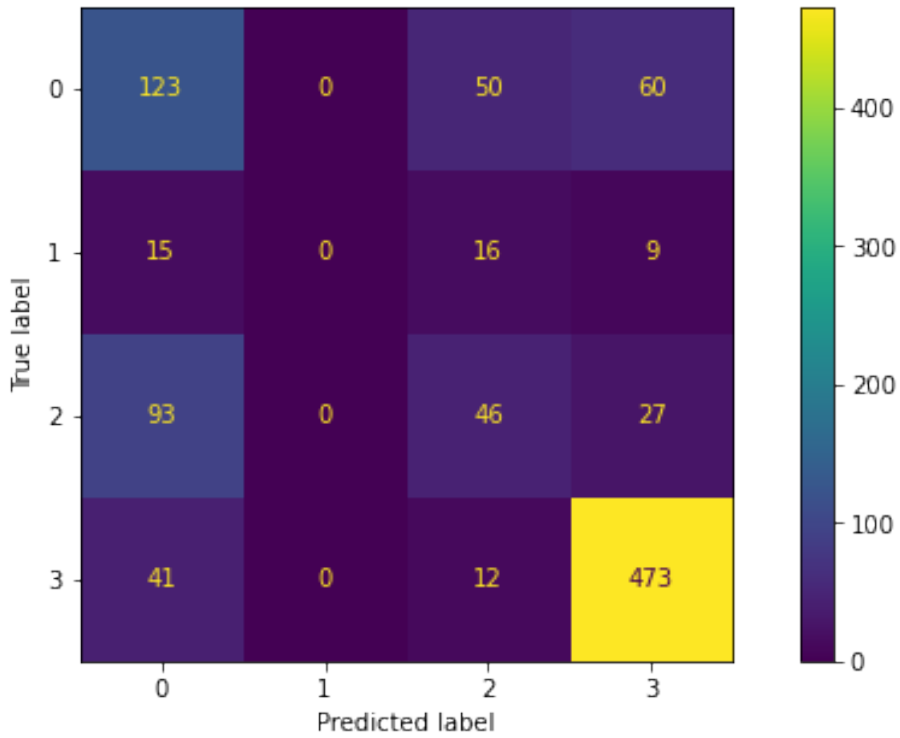

Figure 4.7: Prediction of KNN

4.4 Logistic Regression

The logistic regression algorithm in the sklearn library automatically adapts to the multiclassification case, so our steps are almost identical to the previous one. First, the best penalty term is selected by cross-validation (Lasso and Ridge experiment separately). After finding the best model, it is then used to make predictions.

As we can see in Figure 4.8 and 4.9, Lasso model with C of 0.2 is the best choice for multiclass classification (accuracy of 66% on cross-validation). And it get accuracy of 69% on test set. Figure 4.10 shows the result of prediction on test set.

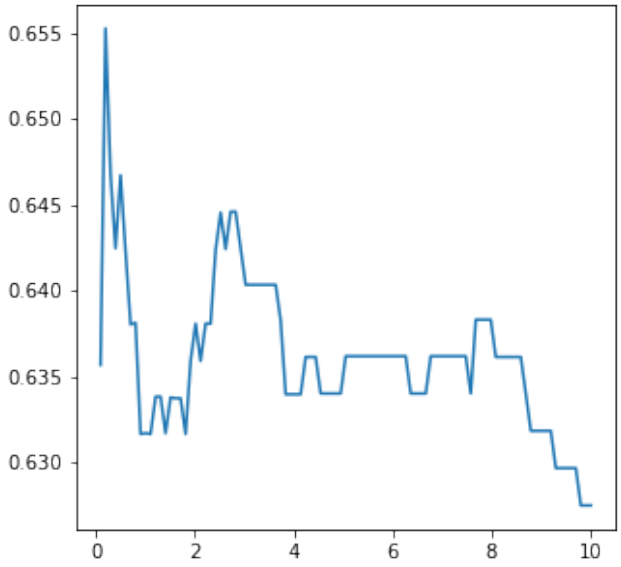

Figure 4.8: CV accuracy of Lasso

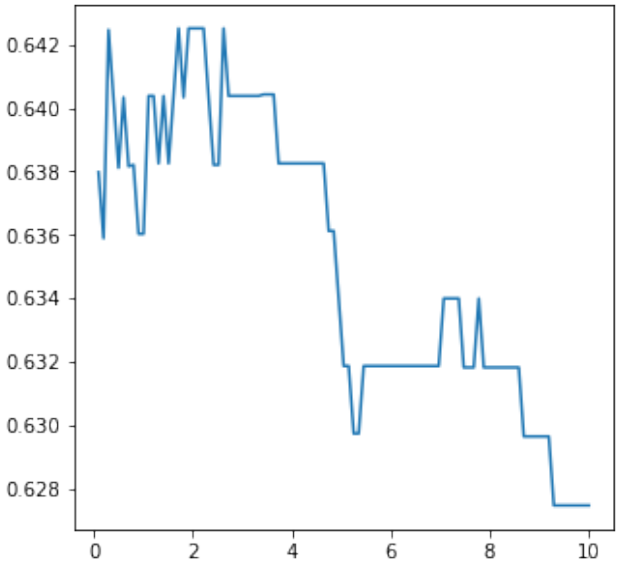

Figure 4.9: CV accuracy of Ridge

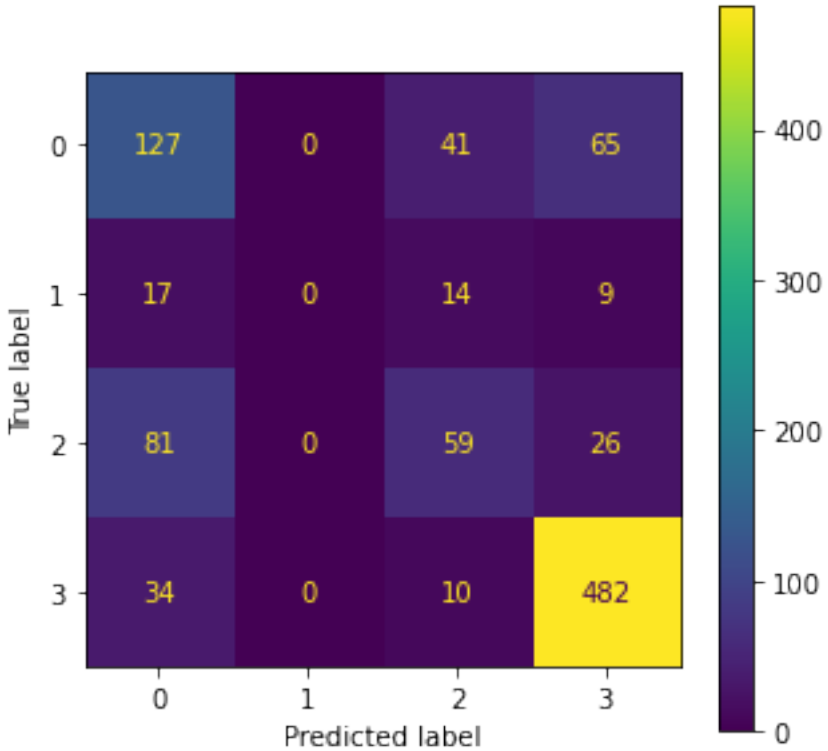

Figure 4.10: Prediction of LR

4.5 Multi-layer perceptron

Multi-layer perceptron is a classical model in the field of deep learning [3]. Since we have been studying deep learning theory recently, we would like to observe the performance of multi-layer perceptron in this project. The hyperparameters of the multi-layer perceptron model are mainly the shape of the hidden layer, and the final model parameters are selected by our intuition because the program takes too long to run due to too many collocation methods.

In summary, we chose the hidden layer of 16x16 shape and the model achieved 67% accuracy on cross-validation and 68% on test set.

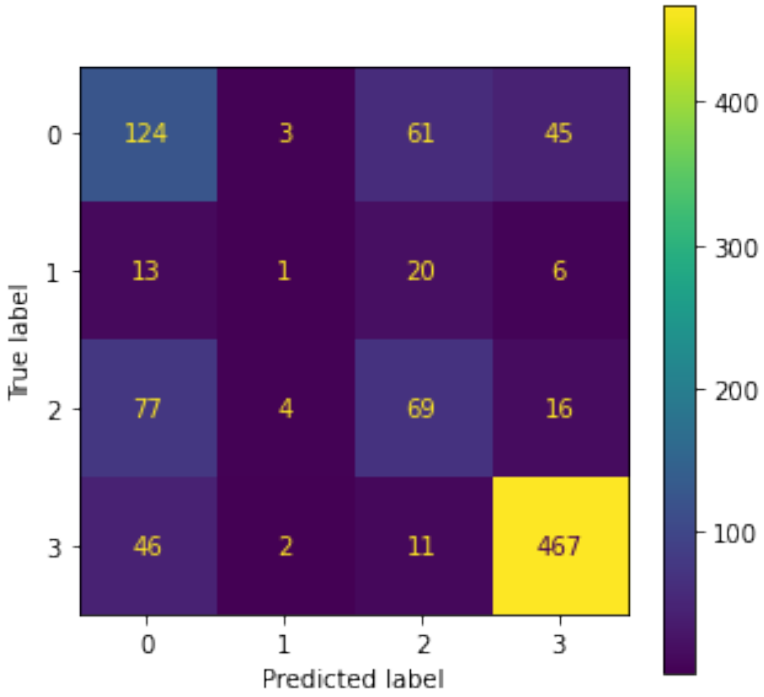

Figure 4.11: Prediction of MLP

5. Conclusion

Based on the accuracy performance of all models in both binary and multiclassification tasks, we finally chose Lasso logistic regression as the final model in the challenge. The results showed that the accuracy was 86% for the binary classification task and 69% for the multiclassification task.

6. Something can be improved

Our model did not get good results in challenge, I think the main reason is that not enough work has been done on feature selection, we did not compare different feature selection methods to select the best one. Secondly, I think there are other normalization methods that can be tried

and compared. However, due to the large workload (the project was done by only one person), we would like to improve these at a later date.

7. References

[1] Kulmala, Markku, et al. "Measurement of the nucleation of atmospheric aerosol particles." *Nature protocols* 7.9 (2012): 1651-1667.

[2] Chandrashekar, Girish, and Ferat Sahin. "A survey on feature selection methods." *Computers & Electrical Engineering* 40.1 (2014): 16-28.

[3] Sklearn website: https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html#sklearn.neural_network.MLPClassifier

A. Appendix

Figure A.1: Raw features

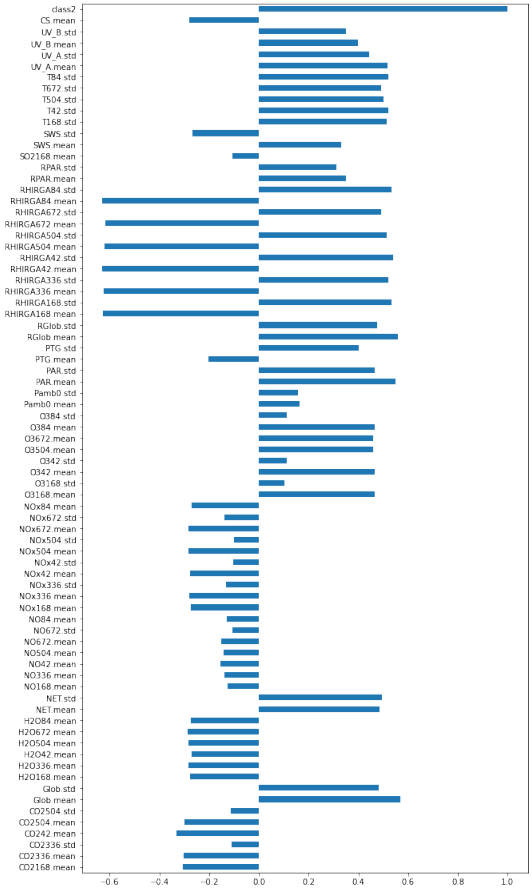

Figure A.2: Filtered Features

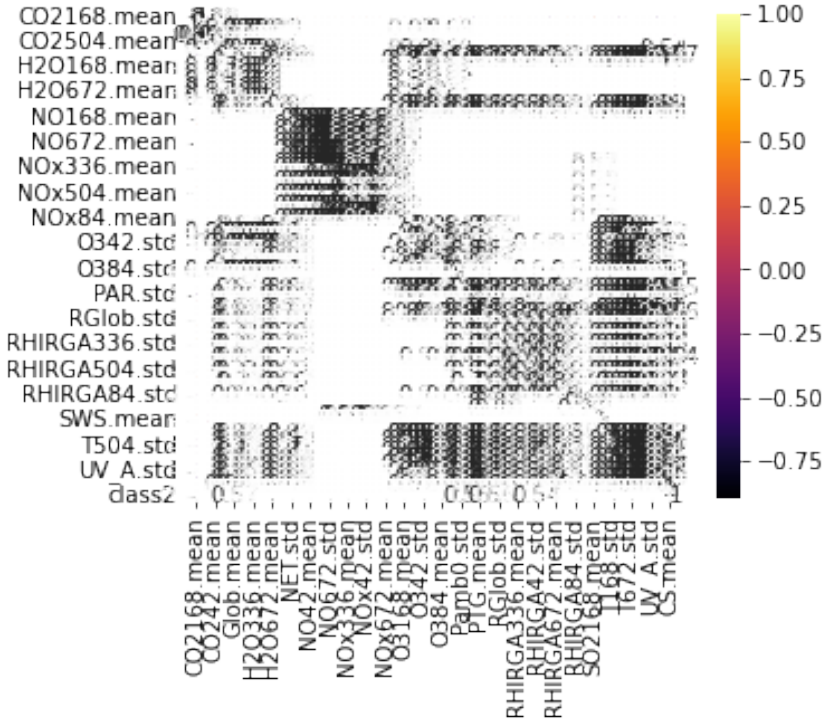

Figure A.3: Correlation matrix

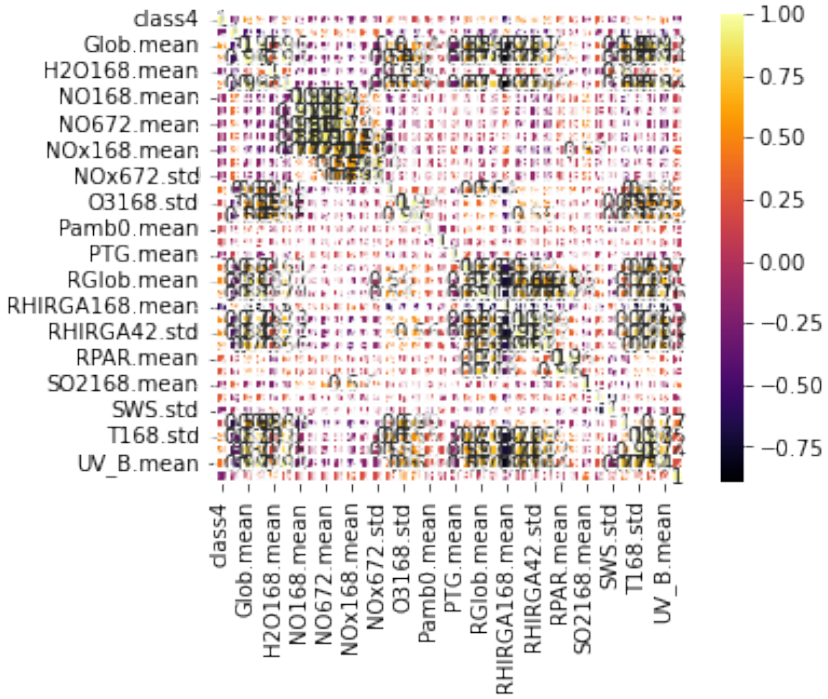

Figure A.4: Correlation matrix

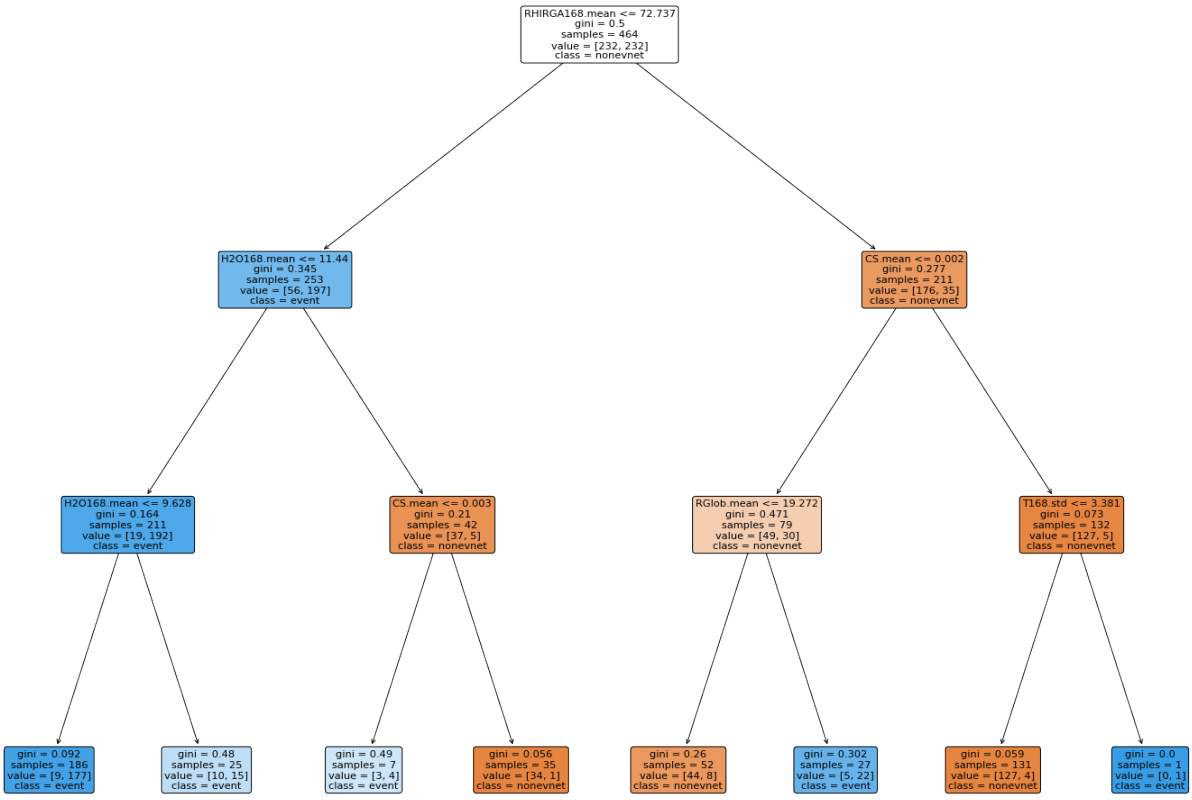

Figure A.5: Decision tree for binary classification

B. Grading section

First, I would give my work this semester **5** out of **5**. I will give the reasons below.

1. Final report: First, we not only show the process and results of the experiments in our final report, but also present all the algorithms we used including an analysis of its advantages and disadvantages in this project. Secondly, the figures in the report are clear and easy to understand and we have fully explained each of them. Finally, the references to the literature are cited in the report.

2. Video presentation: The video is short but explains the main points of the project and shares the interesting things found during the experiment. For example, a simple model does not necessarily perform worse than a complex model and the different effects of different normalization methods.

3. Challenge submission: In the challenge task, we gained a lot of practical tips to improve the accuracy of the model. We have two tips here: 1. the model is trained using all training at submission, which helps to improve the accuracy of the test set. 2. cross-validation tuning. It was a lot of fun to complete the task, even though we ended up submitting the wrong prediction due to an oversight.