COMP 9417
Machine Learning and Data Mining

Project Report

Otto Group Product Classification Challenge

Group 90

Hongyi LUO, z5241868

Zihan LIN, z5271413

Introduction

Some groups operate in a wide range of businesses, like Otto, which is one of the largest e-commercial companies. They provide their dataset but without real information about the features and labels. This can be found in Otto Group Production Classification Challenge via Kaggle. The challenging aim is to build a series of models to find an optimal one that can predict the class of the products based on the predicted probabilities of classes of a certain item. The log loss of the optimal model is officially required as the model evaluation metric.

As a wide field of products, the scale features 93 and over 200,000 items while there are only 9 classes. Toughly browse the dataset, it can be found that for a product, only around 10% values of features are non-zero. In other words, one of the 9 classes may include several subclasses, However, since any values in the dataset are processed, and the meaning of the features are unknown, the challenge may not be spitted to subproblems. The final work is to train learners directly and evaluate the models.

In this project, several supervised learning models are constructed to find the best model to classify the items of 9 classes in terms of 93 features. It is an uncommon machine learning task since standardization and normalization will not affect to a large extent. The data are only shuffled and directly feed into models. SVM, random forest, decision tree, naïve Bayes model, KNN, and XGBoost model is evaluated in terms of training accuracy, test accuracy, and log loss. The Grid search method is applied if the model is not extremely time-consuming. Feature distributions are displayed in the APPENDIX.

Implementation

Support Vector Machine (SVM)

Support vector machine is a two-classification model [1]. This model is based on the linearizer with the largest spatial separation on the features. Here the strategy of SVM is to find the largest interval. Also, the SVM can solve the non-linear classification problem[2]. This is because the kernel trick of SVM can help to find the relationship between the input and different features' spaces[3]. In this problem, Otto data is a non-linear dataset that has high-dimension feature spaces. Therefore, SVM is one way to help us find the solution to this question.

K-Nearest Neighbours (KNN)

K-Nearest neighbours is another method that can solve the multi-classification problem. The training dataset of KNN is high-dimension feature spaces. In the training period, this algorithm only stores the label of training samples and feature vectors. In the classifying period, mostly using Euclidean distance to check the relationship between classes and training sample[4]. K in this algorithm is a constant set by the user. For the Otto dataset, using KNN implement clustering of Otto dataset is feasible[5].

Naive Bayesian algorithm

As far as traditional machine learning is concerned, the naive Bayesian algorithm is well-known as one of the famous classifiers[6]. This algorithm is based on the probability theory. Features in the dataset are independent. Furthermore, when data show different features, the performance of this classifier has no big changes[7]. Therefore, the Naive Bayesian algorithm is stable and has strong robustness. Due to the characteristic of this algorithm, using this algorithm to do the classification is also a good choice to compare with other algorithms.

Decision Tree

Decision tree is a tool in operations research and machine learning. The node in the tree structure represents the object. The branch represents the value of the attribute. Furthermore, the decision tree is a white box model, the user can easily observe the structure of the model[8]. Also, the structure of the decision tree is convenient for users to set up, understand and use. When the dataset is large, such as Otto dataset, the performance of the decision tree is still good. This means the user can get a good result in a short period.

Random Forest Algorithm

Random forest is a typical ensemble learning. It is a cluster of decision trees[9]. Therefore, it contains all advantages of decision trees. This model can solve a large quantity of data. Also, it can give a good performance in a short period. Consider the data distribution in the Otto dataset, one of the advantages of random forest is that it can balance the error brought by the dataset to give a good performance[10].

$XGBoost$ Algorithm

Extreme gradient boosting is one of the famous algorithms in recent years. The $XGBoost$ algorithm uses a stepwise forward additive model. The difference between gradient boosting is that $XGBoost$ is no need to calculate the parameters. This method can reduce the time of the training. Also, it can avoid the happening of overfitting[11]. Here is the target function of $XGBoost$.

$$\mathcal{L}^{(t)} = \sum_{j=1}^{T}\left[G_j\omega_j + \frac{1}{2}\left(H_j + \lambda\right)\omega_j^2\right] + \gamma T$$

Here is the target function of each leaf node j in $XGBoost$.

$$f\left(\omega_j\right) = G_j\omega_j + \frac{1}{2}(H_j + \lambda){\omega_j}^2$$

Based on the equations, we can get the best situation of $XGBoost$.

$$When\ \left(H_j + \lambda\right) > 0\ and\ \omega_j = -\frac{G_j}{H_j + \lambda},$$

$$the\ f\left(\omega_j\right)\ can\ get\ the\ minimum\ value, which\ is\ \ f\left(\omega_j\right)_{min} = -\frac{1}{2}\frac{G_j^2}{H_j + \lambda}$$

Then using the $\omega_j = -\frac{G_j}{H_j+\lambda}$, we can also simplify the target equation of $XGBoost$ and get the following result. This means we can get the current best result when the $\omega_j = -\frac{G_j}{H_j+\lambda}$.

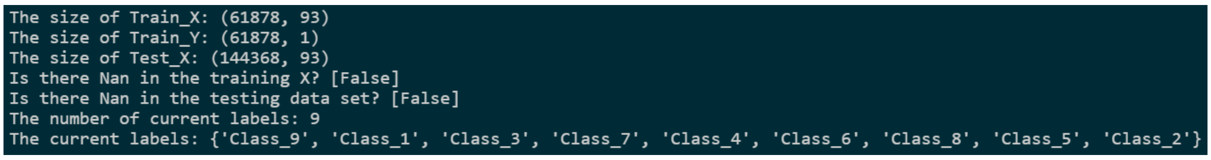
$$\mathcal{L}_{best} = -\frac{1}{2}\sum_{j=1}^{T}\frac{G_j^2}{H_j + \lambda} + \gamma T$$

It should be mentioned that the function of each leaf node in the $XGBoost$ is independent. This means if all leaf nodes in the $XGBoost$ get the best value, the whole target function can get the best result. The most important point is that when $\mathcal{L}_{best}$ goes to the minimum, the user can get the best tree for the current dataset which is also the optimal solution of the current question[12].

Experimentation

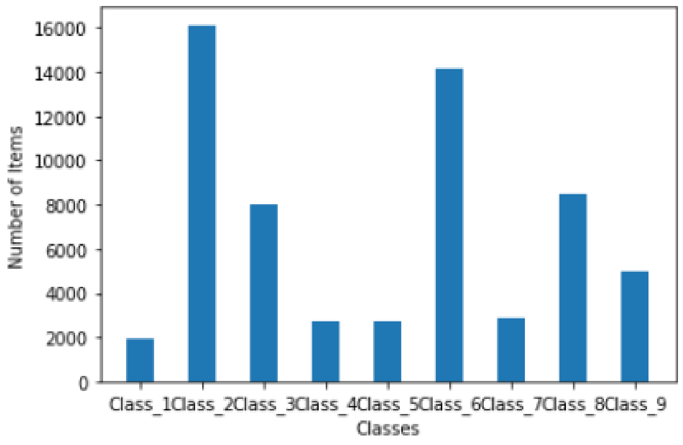
1. Data Analysis

By using Pandas and NumPy reading and processing the data, there are 61878 samples in the train.csv file. Also, there are 93 columns of features and 1 column of the target. After getting the size of the training and testing data, then we should focus on the check of the data, which means checking the none value or nan value in the dataset. In the following picture, there is no nan in the current dataset which means there is no need to do the data replacing. Furthermore, notice the value of a target(label), there are 9 classes in the current label set which are from Class_1 to Class_9. By the way, there is also no nan value in the label set. The results can also be found in the picture given below.



```
The size of Train_X: (61878, 93)
The size of Train_Y: (61878, 1)
The size of Test_X: (144368, 93)
Is there Nan in the training X? [False]
Is there Nan in the testing data set? [False]
The number of current labels: 9
The current labels: {'Class_9', 'Class_1', 'Class_3', 'Class_7', 'Class_4', 'Class_6', 'Class_8', 'Class_5', 'Class_2'}
```

Then, the data distribution of the Otto dataset is a key point as well in data analysis. There are 9 classes in this dataset. Counting the features in every 9 class, we can get the picture as showing below. From this picture, we can find that the sum of features in class 2 is the largest. Furthermore, the sum of features in class 1 is the lowest.



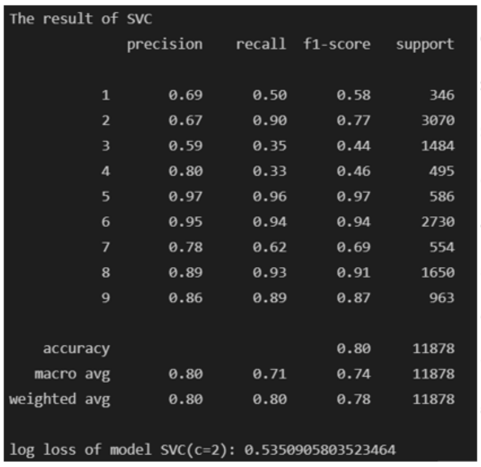
Then we do the statistic of the current features. The resulting image can be found in **APPENDIX I**. By counting the total number of different features, the feature67 and feature24 are larger than other features. The sum of feature 6 is the smallest in all 93 features. In this project, using the previous 50,000 training data to train the model, as well as using the extra 11,878 training data to do the testing.

To find more relationships between classes and features, for each class, we got the feature distribution. the picture result can be found in **APPENDIX II**. From the picture given in **APPENDIX II**, if the bars are gathering, the feature represented by the current bar plays a decisive role in this class. For example, consider the class5 in **APPENDIX II**, feature 34 is significantly more than other features. Therefore, we can understand that class 1 is determined by feature 24 and feature 67.

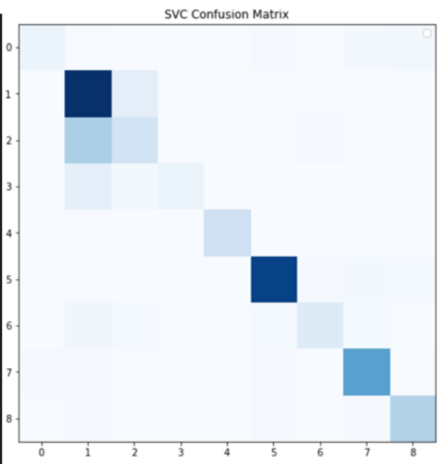
2. Training Models

(a) Support Vector Machine

The SVM model is a time-consuming model when facing a large quantity of data and such that the grid search approach is not applied to avoid the bad impact on the computer and the kernel dead. Therefore, a grid of C, [0.1, 2, 3] is tested and finally find the best model with C = 2. After that, using C = 2, the following result can be gotten.



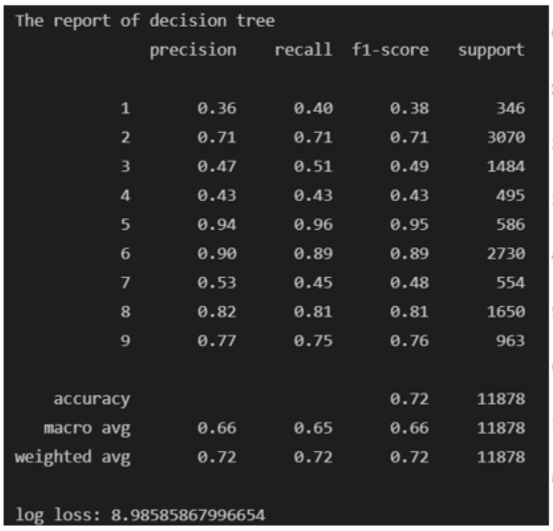
```
The result of SVC
              precision    recall  f1-score   support

           1       0.69      0.50      0.58       346
           2       0.67      0.90      0.77      3070
           3       0.59      0.35      0.44      1484
           4       0.80      0.33      0.46       495
           5       0.97      0.96      0.97       586
           6       0.95      0.94      0.94      2730
           7       0.78      0.62      0.69       554
           8       0.89      0.93      0.91      1650
           9       0.86      0.89      0.87       963

    accuracy                           0.80     11878
   macro avg       0.80      0.71      0.74     11878
weighted avg       0.80      0.80      0.78     11878

log loss of model SVC(c=2): 0.5350905803523464
```



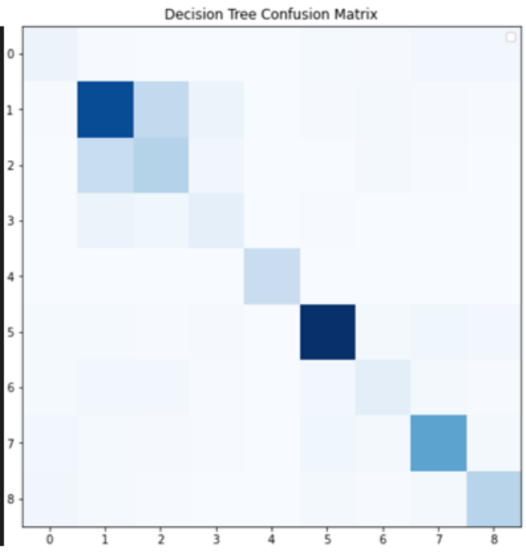
From the picture given above, the prediction accuracy of class5, class6 and class8 is good, which is more than 0.90. Also, for class3 the performance is the worst which an accuracy is 0.44. Finally, the average accuracy is 0.80. The log loss of the current model is 0.535. The average recall of this model is 0.71. When doing the training, the time-consuming is one disadvantage of SVM, due to the dataset is too large.

(b) Decision Tree

As few hyperparameters can be altered in the decision tree model, the different values of max_depth are applied but the optimized one is that until all leaves are pure, such that the training accuracy should be 1. In this model, using $GridSearchCV$ can get the best parameters for the current training dataset[15]. From the picture given below, we can find that the decision tree model performs well in class5 and class6 which the accuracy is near 0.90. This model performs worst in class1 whose accuracy is 0.38. Furthermore, the final accuracy is 0.72 and the average of recall is 0.65. However, the log loss is 8.98 which is too large than the SVM model.

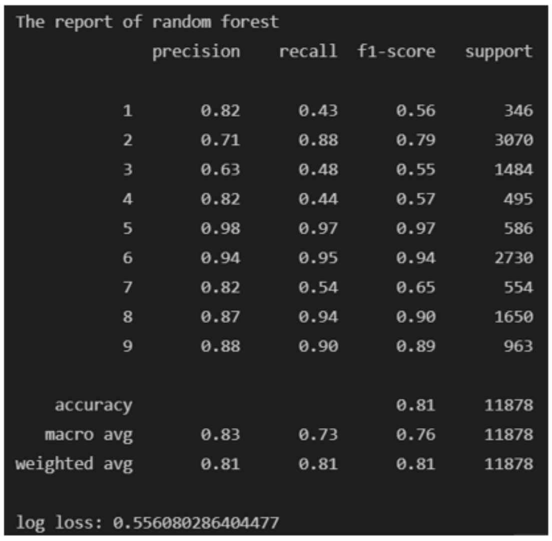


```
The report of decision tree
              precision    recall  f1-score   support

           1       0.36      0.40      0.38       346
           2       0.71      0.71      0.71      3070
           3       0.47      0.51      0.49      1484
           4       0.43      0.43      0.43       495
           5       0.94      0.96      0.95       586
           6       0.90      0.89      0.89      2730
           7       0.53      0.45      0.48       554
           8       0.82      0.81      0.81      1650
           9       0.77      0.75      0.76       963

    accuracy                           0.72     11878
   macro avg       0.66      0.65      0.66     11878
weighted avg       0.72      0.72      0.72     11878

log loss: 8.98585867996654
```

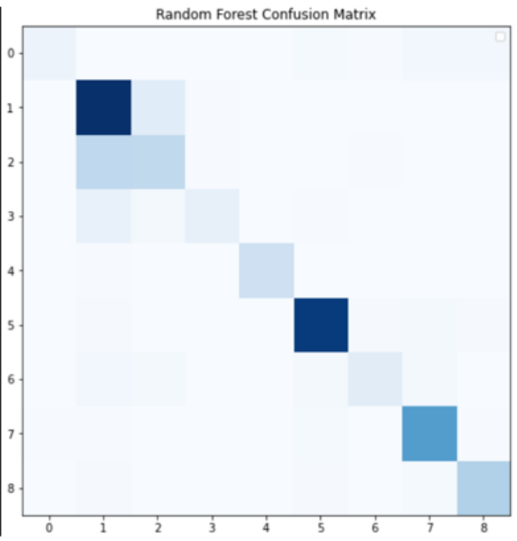


(c) Random Forest

Random Forest is a kind of ensemble learning which is based on the decision tree. As a common parameter for the random forest, a grid of $n\_estimators$ of [300, 400, 500, 600, 700] is tested by using the $GridSearchCV$ method and the best estimator is when $n\_estimator$ is 600[16]. However, the accuracy will not vary too much. For random forest, the performance is better than the decision tree. The average accuracy is 0.81 and the recall is 0.73. The log loss is 0.556 which is smaller than the decision tree. This means as for a kind of ensemble learning, random forest performs well than decision tree in multi-classes classification.

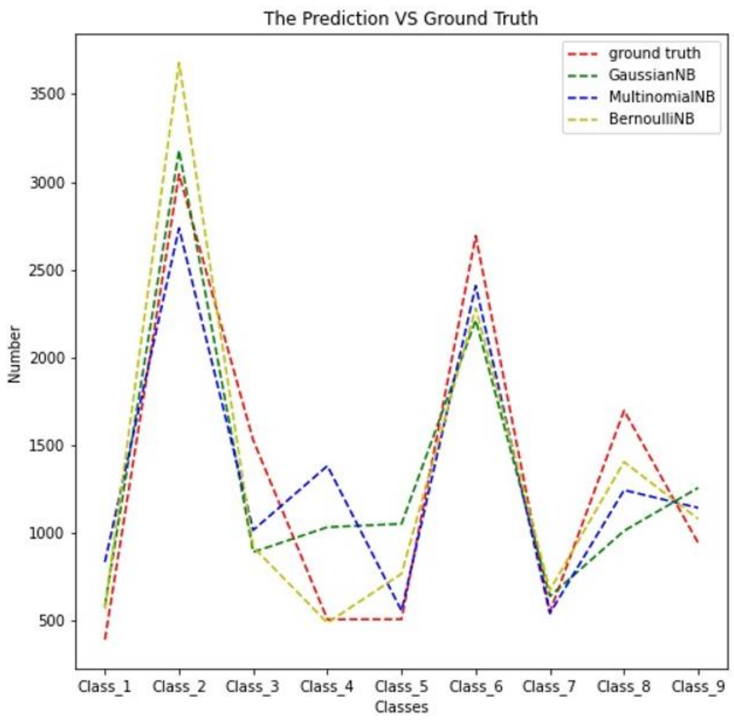


```
The report of random forest
              precision    recall  f1-score   support

           1       0.82      0.43      0.56       346
           2       0.71      0.88      0.79      3070
           3       0.63      0.48      0.55      1484
           4       0.82      0.44      0.57       495
           5       0.98      0.97      0.97       586
           6       0.94      0.95      0.94      2730
           7       0.82      0.54      0.65       554
           8       0.87      0.94      0.90      1650
           9       0.88      0.90      0.89       963

    accuracy                           0.81     11878
   macro avg       0.83      0.73      0.76     11878
weighted avg       0.81      0.81      0.81     11878

log loss: 0.556080286404477
```

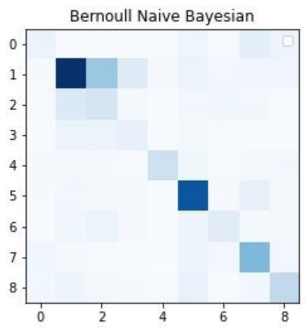


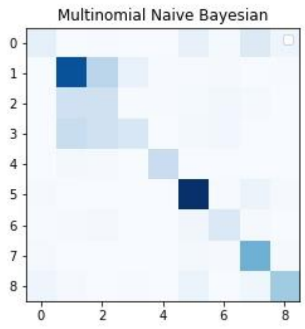
(d) Naive Bayesian algorithm

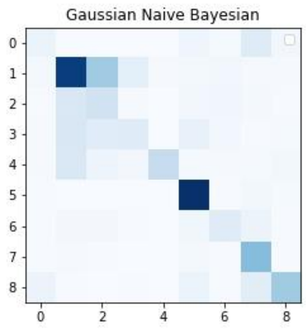
The parameters of the naive Bayesian algorithm using in predicting the Otto dataset in this project are default values. In this part, use 3 different methods provided by scikit-learn to do the comparison. Here is the result:

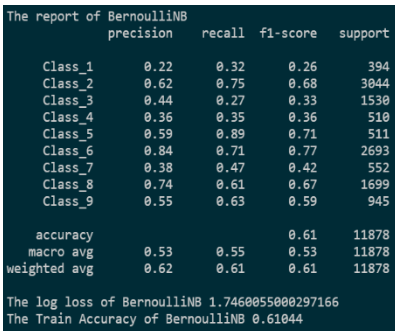


In this picture, the red line is the test label y which is the ground truth label. From this picture, the result in class3, class4 and class5 is different. Also, the result of the multinomial naïve Bayesian algorithm is more accurate than other naïve Bayes methods. Furthermore, by using the metrics function provided by scikit-learn, it is obvious that the accuracy of the multinomial Naive Bayesian algorithm is 0.67 which is larger than Gaussian Naive Bayesian algorithm and Bernoulli Naive Bayesian algorithm. In addition, using the confusion matrix is also a good way to check the result. Here are the picture results of the confusion matrix.

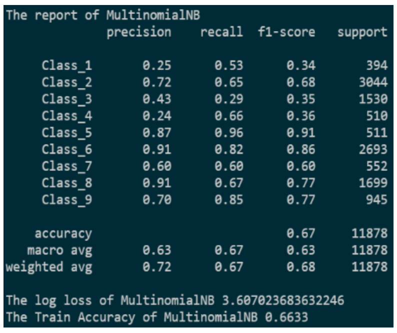




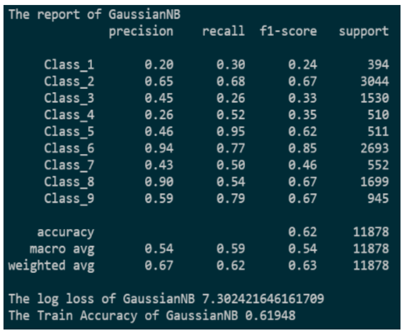




```
The report of BernoulliNB
              precision    recall  f1-score   support

     Class_1       0.22      0.32      0.26       394
     Class_2       0.62      0.75      0.68      3044
     Class_3       0.44      0.27      0.33      1530
     Class_4       0.36      0.35      0.36       510
     Class_5       0.59      0.89      0.71       511
     Class_6       0.84      0.71      0.77      2693
     Class_7       0.38      0.47      0.42       552
     Class_8       0.74      0.61      0.67      1699
     Class_9       0.55      0.63      0.59       945

    accuracy                           0.61     11878
   macro avg       0.53      0.55      0.53     11878
weighted avg       0.62      0.61      0.61     11878

The log loss of BernoulliNB 1.7460055000297166
The Train Accuracy of BernoulliNB 0.61044
```



```
The report of MultinomialNB
              precision    recall  f1-score   support

     Class_1       0.25      0.53      0.34       394
     Class_2       0.72      0.65      0.68      3044
     Class_3       0.43      0.29      0.35      1530
     Class_4       0.24      0.66      0.36       510
     Class_5       0.87      0.96      0.91       511
     Class_6       0.91      0.82      0.86      2693
     Class_7       0.60      0.60      0.60       552
     Class_8       0.91      0.67      0.77      1699
     Class_9       0.70      0.85      0.77       945

    accuracy                           0.67     11878
   macro avg       0.63      0.67      0.63     11878
weighted avg       0.72      0.67      0.68     11878

The log loss of MultinomialNB 3.607023683632246
The Train Accuracy of MultinomialNB 0.6633
```

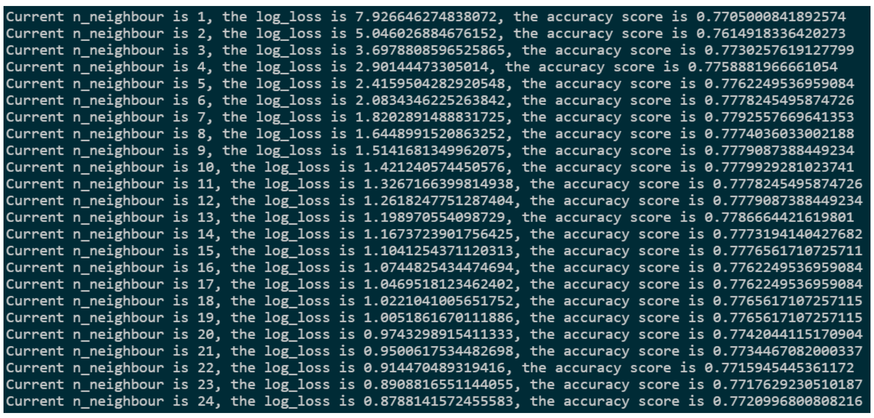


```
The report of GaussianNB
              precision    recall  f1-score   support

     Class_1       0.20      0.30      0.24       394
     Class_2       0.65      0.68      0.67      3044
     Class_3       0.45      0.26      0.33      1530
     Class_4       0.26      0.52      0.35       510
     Class_5       0.46      0.95      0.62       511
     Class_6       0.94      0.77      0.85      2693
     Class_7       0.43      0.50      0.46       552
     Class_8       0.90      0.54      0.67      1699
     Class_9       0.59      0.79      0.67       945

    accuracy                           0.62     11878
   macro avg       0.54      0.59      0.54     11878
weighted avg       0.67      0.62      0.63     11878

The log loss of GaussianNB 7.302421646161709
The Train Accuracy of GaussianNB 0.61948
```

From the metrics report give above, the accuracy of the Multinomial Naïve Bayesian algorithm is higher than others. The reason for this situation is that the current dataset is categorical which contains several classes[17]. Also, if the dataset obeys normal distribution, then the Gaussian Naïve Bayesian algorithm can perform well. Otherwise, if the label of the dataset is binary, then the Bernoulli Naive Bayesian algorithm will give a good performance. Therefore, in this situation, the multinomial Naïve Bayesian algorithm is suitable for the Otto dataset.

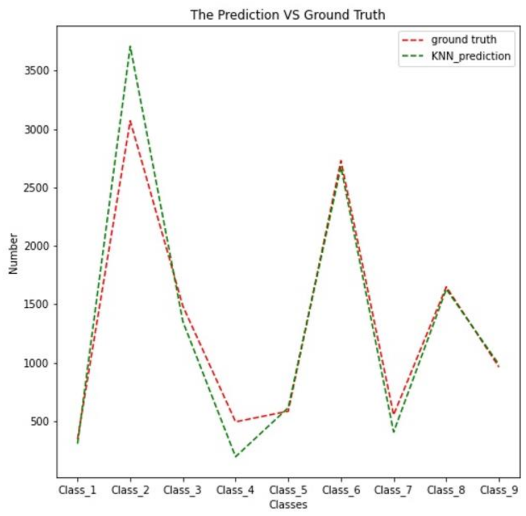
(e) K-Nearest Neighbour algorithm

To check the best neighbour value in the KNN algorithm, storing the n neighbour value and accuracy value from 1 to 24 to find the best result. Here is the result of finding the best parameters.

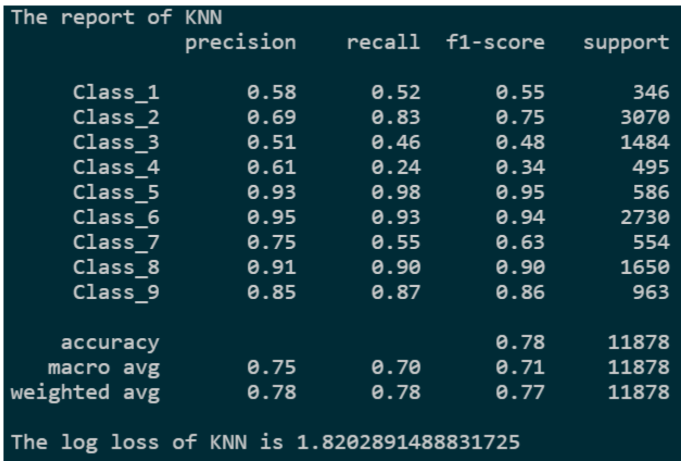


```
Current n_neighbour is 1, the log_loss is 7.926646274838072, the accuracy score is 0.7705000841892574
Current n_neighbour is 2, the log_loss is 5.046026884676152, the accuracy score is 0.7614918336420273
Current n_neighbour is 3, the log_loss is 3.6978808596525865, the accuracy score is 0.7730257619127799
Current n_neighbour is 4, the log_loss is 2.90144473305014, the accuracy score is 0.775881966661054
Current n_neighbour is 5, the log_loss is 2.4159504282920548, the accuracy score is 0.7762249536959084
Current n_neighbour is 6, the log_loss is 2.0834346225263842, the accuracy score is 0.7778245495874726
Current n_neighbour is 7, the log_loss is 1.8202891488831725, the accuracy score is 0.7792557669641353
Current n_neighbour is 8, the log_loss is 1.6448991520863252, the accuracy score is 0.7774036033002188
Current n_neighbour is 9, the log_loss is 1.5141681349962075, the accuracy score is 0.7779087388449234
Current n_neighbour is 10, the log_loss is 1.421240574450576, the accuracy score is 0.7779929281023741
Current n_neighbour is 11, the log_loss is 1.3267166399814938, the accuracy score is 0.7778245495874726
Current n_neighbour is 12, the log_loss is 1.2618247751287404, the accuracy score is 0.7779087388449234
Current n_neighbour is 13, the log_loss is 1.198970554098729, the accuracy score is 0.7786664421619801
Current n_neighbour is 14, the log_loss is 1.1673723901756425, the accuracy score is 0.7773194140427682
Current n_neighbour is 15, the log_loss is 1.1041254371120313, the accuracy score is 0.7776561710725711
Current n_neighbour is 16, the log_loss is 1.0744825434474694, the accuracy score is 0.7762249536959084
Current n_neighbour is 17, the log_loss is 1.0469518123462402, the accuracy score is 0.7762249536959084
Current n_neighbour is 18, the log_loss is 1.0221041005651752, the accuracy score is 0.7765617107257115
Current n_neighbour is 19, the log_loss is 1.0051861670111886, the accuracy score is 0.7765617107257115
Current n_neighbour is 20, the log_loss is 0.9743298915411333, the accuracy score is 0.7742044115170904
Current n_neighbour is 21, the log_loss is 0.9500617534482698, the accuracy score is 0.7734467082000337
Current n_neighbour is 22, the log_loss is 0.914470489319416, the accuracy score is 0.7715945445361172
Current n_neighbour is 23, the log_loss is 0.8908816551144055, the accuracy score is 0.7717629230510187
Current n_neighbour is 24, the log_loss is 0.8788141572455583, the accuracy score is 0.7720996800808216
```

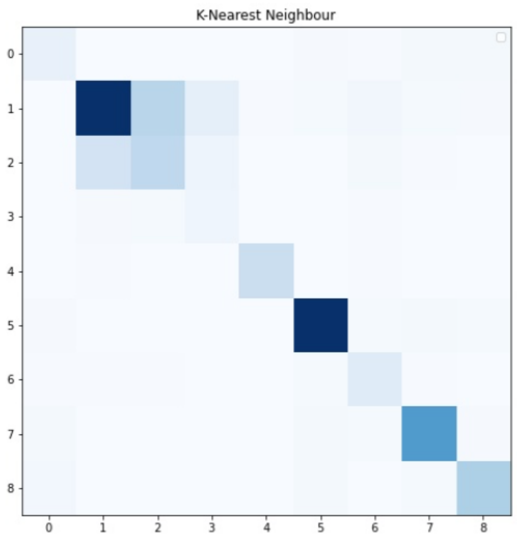
From this capture given above, when the neighbour value equals 7, the accuracy score goes to maximize, which is around 0.779. In the following part of KNN, using the best performance neighbour value to do the following training and testing. Using the prediction result to compare with the ground truth label, the following result of the K-Nearest Neighbour algorithm can be found in the picture.



From this picture, the results of prediction in class2, clsss3 and class4 are different from the test target. Furthermore, the tendency of the two curves is almost the same. This means the result provided by KNN is good. This also proves that for the clustering problem, such as Otto classification, KNN gives a good performance. Here are the confusion matrix and the metrics report of KNN. From the picture, we can find that the accuracy of this model is 0.78 and the log loss of KNN is around 1.82. Furthermore, the recall of KNN is 0.70.

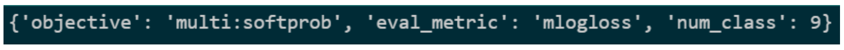


```
The report of KNN
              precision    recall  f1-score   support

     Class_1       0.58      0.52      0.55       346
     Class_2       0.69      0.83      0.75      3070
     Class_3       0.51      0.46      0.48      1484
     Class_4       0.61      0.24      0.34       495
     Class_5       0.93      0.98      0.95       586
     Class_6       0.95      0.93      0.94      2730
     Class_7       0.75      0.55      0.63       554
     Class_8       0.91      0.90      0.90      1650
     Class_9       0.85      0.87      0.86       963

    accuracy                           0.78     11878
   macro avg       0.75      0.70      0.71     11878
weighted avg       0.78      0.78      0.77     11878

The log loss of KNN is 1.8202891488831725
```



(f) $XGBoost$ Algorithm

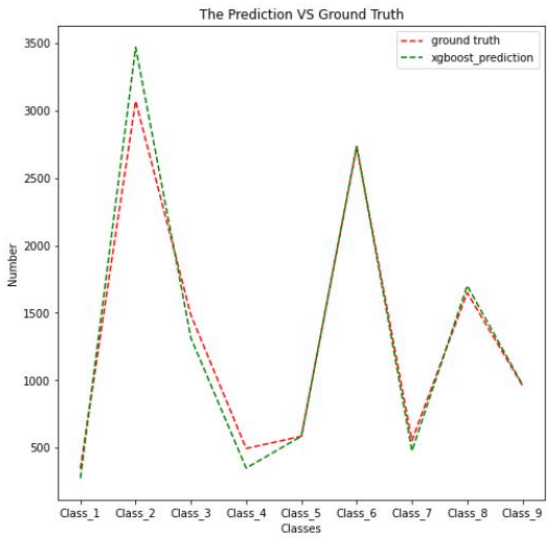
For $XGBoost$ in the Otto dataset, setting the parameter objective as $multi: softprob$. This is to make sure the size of output as $n\_dimensional\, data * n\_dimensional\, classes$. Also, based on the question, setting the metric as $mlogloss$. This means using the log-loss method to evaluate the result of the model. The last parameter in $XGBoost$ is the number of classes. In this project, the total number of classes is 9. Therefore, the "num_class": 9.[14]



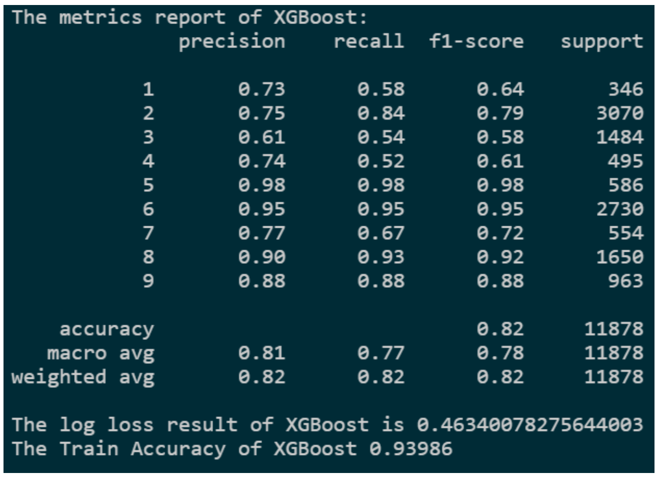
```
{'objective': 'multi:softprob', 'eval_metric': 'mlogloss', 'num_class': 9}
```

One point should be mentioned here, using the log-loss method to evaluate the result, the output is probability. To get the metrics report, mapping the largest probability to the classes can help us obtain the report.

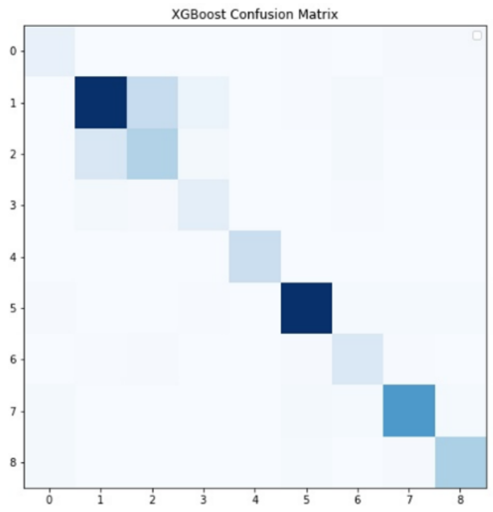
By doing the model training and testing, the following result can be found in the picture given below.



From the picture given above, we can find that not only the tendency of the line but also the highest accuracy provided by $XGBoost$. Using the same way in the previous part, the confusion matrix and metrics report can be gotten as showing below.



```
The metrics report of XGBoost:
              precision    recall  f1-score   support

           1       0.73      0.58      0.64       346
           2       0.75      0.84      0.79      3070
           3       0.61      0.54      0.58      1484
           4       0.74      0.52      0.61       495
           5       0.98      0.98      0.98       586
           6       0.95      0.95      0.95      2730
           7       0.77      0.67      0.72       554
           8       0.90      0.93      0.92      1650
           9       0.88      0.88      0.88       963

    accuracy                           0.82     11878
   macro avg       0.81      0.77      0.78     11878
weighted avg       0.82      0.82      0.82     11878

The log loss result of XGBoost is 0.46340078275644003
The Train Accuracy of XGBoost 0.93986
```



Compared with other models implemented in this project, the $XGBoost$ performs best. This is because $XGBoost$ is similar to Random Forest. This method can get the split direction of the tree model by doing the training and studying. This means it is more adaptive than Random Forest[11]. For $XGBoost$, the accuracy of class1 is higher than other models, also the average accuracy of $XGBoost$ is 0.82 which is also the largest. Furthermore, the recall of $XGBoost$ is 0.77.

Methods		Parameters	Train Acc	Test Acc	Log Loss	Parameter Range
SVM		C=2	0.81	0.80	0.535	[0.1, 2, 3]
Random Forest		n_estimators=600	1.0	0.81	0.556	[300, 400, 500, 600, 700]
Decision Tree		None	1.0	0.72	8.986	[None, 100, 200]
KNN		n_neighbour=7	0.82934	0.78	1.820	None
Naïve Bayesian	GaussianNB	None	0.61948	0.61	7.406	None
	MultinomialNB		0.6633	0.66	3.654	None
	BernoulliNB		0.61044	0.60	1.799	None
XGBoost		"multi:softprob" "mlogloss", "num_class": 9	0.93986	0.82	0.463	None

Conclusion.

To find a feasible algorithm to train and test the Otto dataset, several models have been used in the project, such as SVM, decision tree, random forest, KNN, naïve Bayesian algorithm and $XGBoost$. Here is the conclusion.

The random forest is a good choice in training the Otto dataset. This is because of the feature of ensemble learning. This method is a kind of enhanced version of the decision tree. By setting up the weights, it can finally find a good situation to fit the current model.

The $XGBoost$ is the best model in the Otto dataset. Also, the training accuracy is 0.93 which is different from Random Forest and Decision Tree which training accuracy is 1. This is because, in the previous part, one feature of $XGBoost$ is to avoid overfitting. Also, this model is more adaptive for fitting different datasets in different competitions.

Consider other training models, KNN's performance is stable, however, this method is not adaptive. For some datasets with special distribution, the performance of KNN may not be good. For SVM, when facing the large-scale dataset, the efficiency is poor. This is also the reason why not using $GridSearchCV$ to do the deep optimization. As for the decision tree, the disadvantage of this model is obvious. This model is easily overfitting as well as sometimes the decision tree model will ignore the relationship of different attributes in the dataset. Finally, for Multinomial Naïve Bayesian, although this method can do classification, the performance is not good. Therefore, to improve the performance, using the boosting method, such as $Adaboost$, may help to enhance the model.

Reference

[1] Kecman, V., 2005. Support vector machines–an introduction. In Support vector machines: theory and applications (pp. 1-47). Springer, Berlin, Heidelberg.

[2] Nie, F., Zhu, W. and Li, X., 2020. Decision tree SVM: an extension of linear SVM for non-linear classification. Neurocomputing, 401, pp.153-159.

[3] Tang, T., Chen, S., Zhao, M., Huang, W. and Luo, J., 2019. Very large-scale data classification based on K-means clustering and multi-kernel SVM. Soft Computing, 23(11), pp.3793-3801.

[4] Zhang, Z., 2016. Introduction to machine learning: k-nearest neighbors. *Annals of translational medicine*, *4*(11).

[5] Zhang, M.L. and Zhou, Z.H., 2007. ML-KNN: A lazy learning approach to multi-label learning. *Pattern recognition*, *40*(7), pp.2038-2048.

[6] Huang, Y. and Li, L., 2011, September. Naive Bayes classification algorithm based on small sample set. In *2011 IEEE International conference on cloud computing and intelligence systems* (pp. 34-39). IEEE.

[7] Farid, D.M., Zhang, L., Rahman, C.M., Hossain, M.A. and Strachan, R., 2014. Hybrid decision tree and naïve Bayes classifiers for multi-class classification tasks. *Expert systems with applications*, *41*(4), pp.1937-1946.

[8] Abpeykar, S. and Ghatee, M., 2019. An ensemble of RBF neural networks in decision tree structure with knowledge transferring to accelerate multi-classification. *Neural Computing and Applications*, *31*(11), pp.7131-7151.

[9] Ali, J., Khan, R., Ahmad, N. and Maqsood, I., 2012. Random forests and decision trees. *International Journal of Computer Science Issues (IJCSI)*, *9*(5), p.272.

[10] Chaudhary, A., Kolhe, S. and Kamal, R., 2016. An improved random forest classifier for multi-class classification. *Information Processing in Agriculture*, *3*(4), pp.215-222.

[11] Chen, T., He, T., Benesty, M., Khotilovich, V., Tang, Y. and Cho, H., 2015. Xgboost: extreme gradient boosting. *R package version 0.4-2*, *1*(4), pp.1-4.

[12] Ramraj, S., Uzir, N., Sunil, R. and Banerjee, S., 2016. Experimenting XGBoost algorithm for prediction and classification of different datasets. *International Journal of Control Theory and Applications*, *9*, pp.651-662.

[13] Different training models performance in Otto dataset.
Link: https://github.com/amaltarghi/Otto-Group-Product-Classification-Challenge

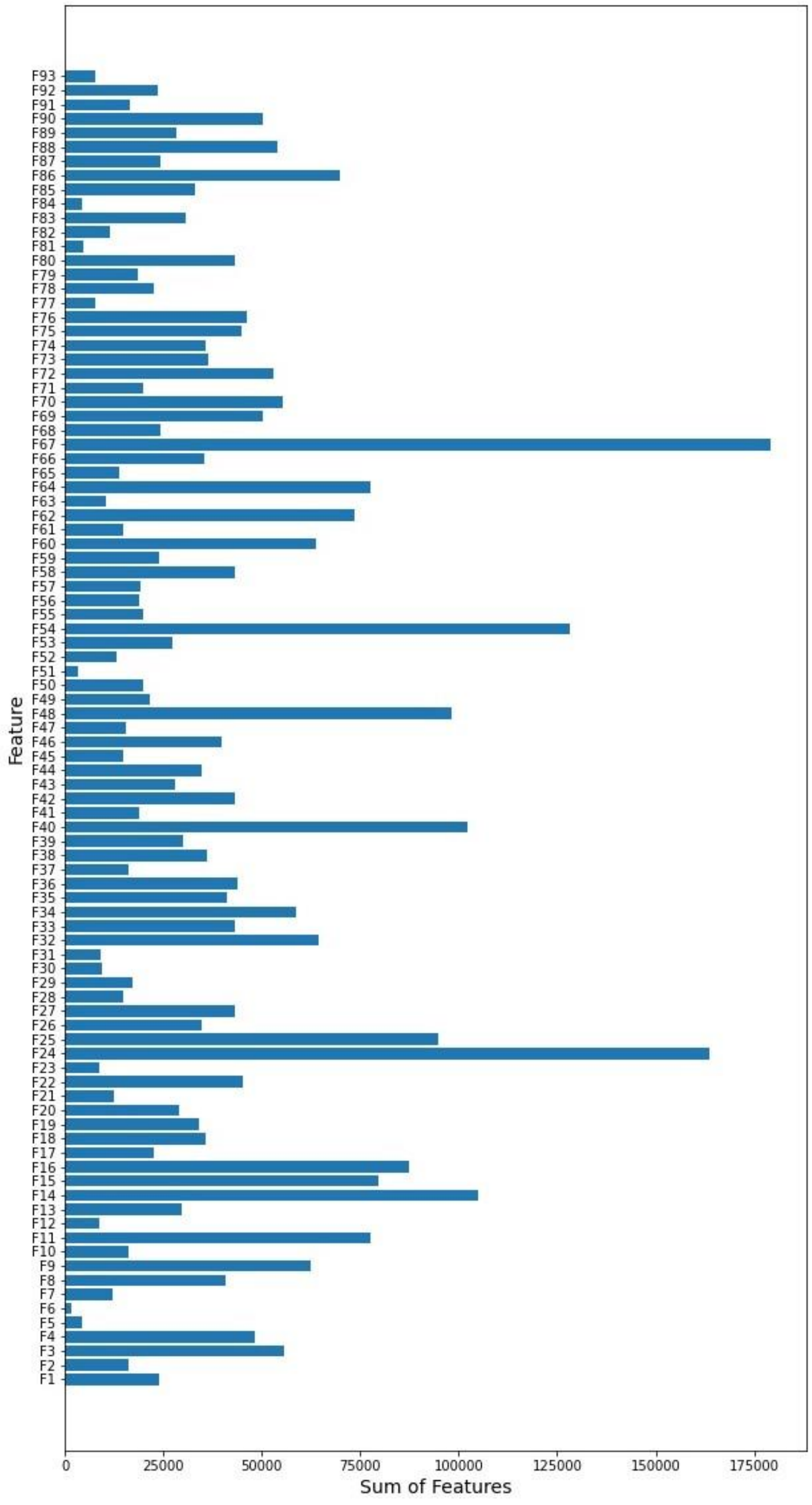
[14] How to understand the parameter in XGBoost.
Link: https://www.cnblogs.com/TimVerion/p/11436001.html
Link: https://blog.csdn.net/u010657489/article/details/51952785
Link: https://xgboost.readthedocs.io/en/latest/parameter.html

[15] How to use GridsearchCV to optimize the hyperparameters in the decision tree.
Link: https://stackoverflow.com/questions/38709690/scikit-learn-using-gridsearchcv-on-decisiontreeclassifier
Link: https://blog.csdn.net/evolution23/article/details/85243980

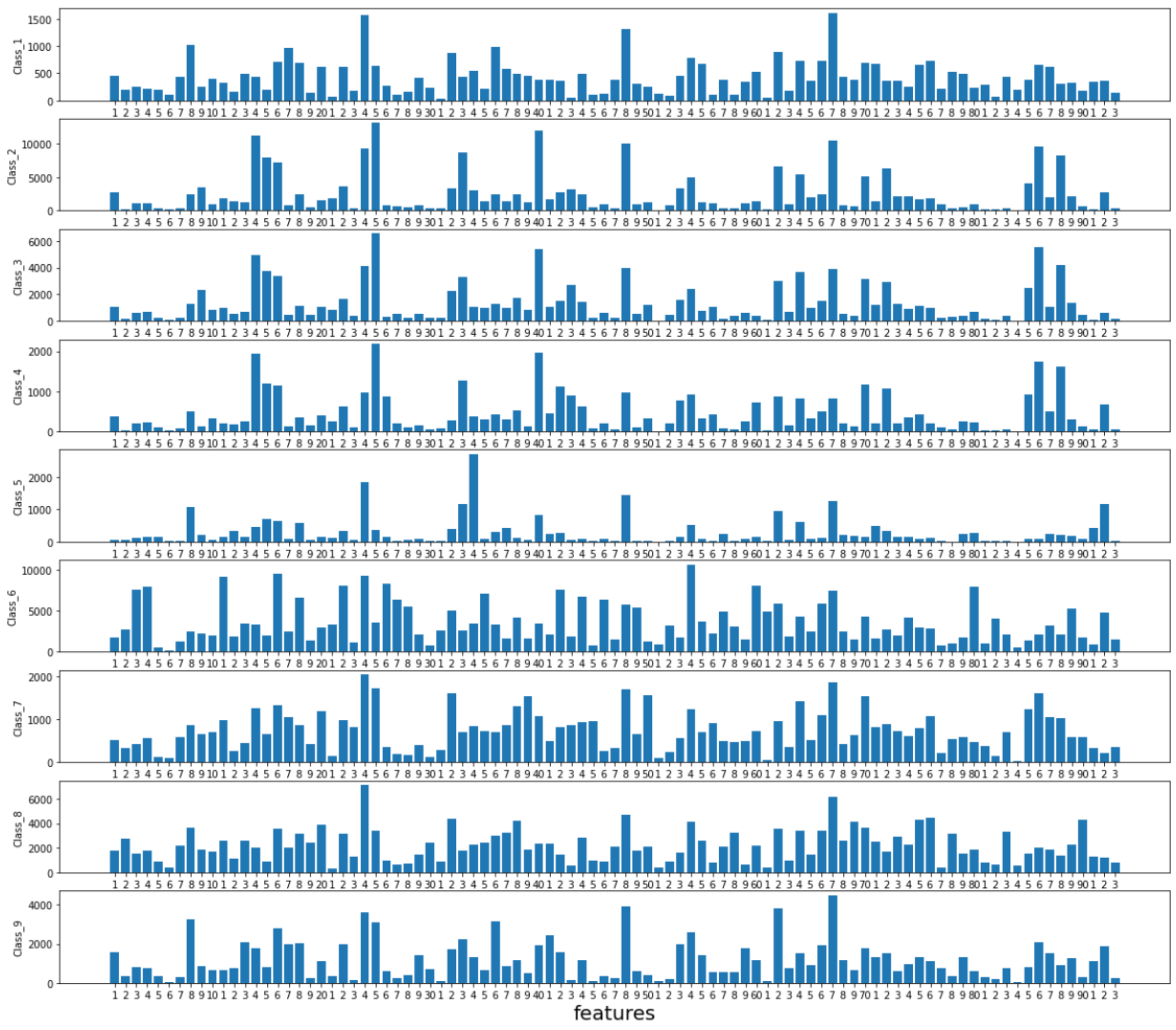
[16] How to use GridsearchCV to optimize the hyperparameters in the random forest.
Link: https://stackoverflow.com/questions/30102973/how-to-get-best-estimator-on-gridsearchcv-random-forest-classifier-scikit
Link: https://zhuanlan.zhihu.com/p/113014922

[17] How to understand different Naive Bayes classification algorithm.
Link: https://zhuanlan.zhihu.com/p/39780650

Appendix I. Plot that shows the amount of appearance of each feature



Appendix II. Plot that shows the amount of appearance of each feature non-zero value for each class


Class_1
Class_2
Class_3
Class_4
Class_5
Class_6
Class_7
Class_8
Class_9
features