Machine Learning in the Unknown Project Report

Team Members

Name: Zhenxiang Lin	Student ID: z5240946
Name: Haojin Guo	Student ID: z5216214
Name: Xinrui Liu	Student ID: z5235006
Name: Qianhui Guo	Student ID: z5291528
Name: Rongrong Zheng	Student ID: z5346267

1. Introduction

Nowadays, the problem of information security has been paid more and more attention. This project not only ensures the security of user information, but also processes and analyzes the data, so as to build a best multi-classification model for data prediction. This project is about the construction of models for multi-classification problems, with data prediction. After analyzing and processing the data by means of feature correlation and outlier point processing, we compared the effects and precision of various models, and selected the ones with the best effects for voting, so as to generate the final prediction file. To optimize the model, it is necessary to find the optimal parameters, including data dimensions, k-fold stability, and selection of internal parameters of the model according to Grid Search along with cross validation. Finally, model screening was carried out according to the F1 score, and several groups of models with the best effect were selected. X_test was predicted respectively, and the final classification was selected by voting. Finally, our model achieves an ideal accuracy effect in the verification set.

The dataset is composed of a class column, representing the target variable and which takes one of 6 possible values, and the remaining columns are 128 real valued features. In this data set, the data type is float. Due to privacy protection, the information of each feature is not indicated. Therefore, correlation analysis is also an important component of this project.

2. Exploratory Data Analysis

2.1. Data class distribution

The class distribution of training data set is shown as Fig. 1. It can be seen that the distribution of different classes is relatively average. There is no class which accounts for an extremely large or small percentage.

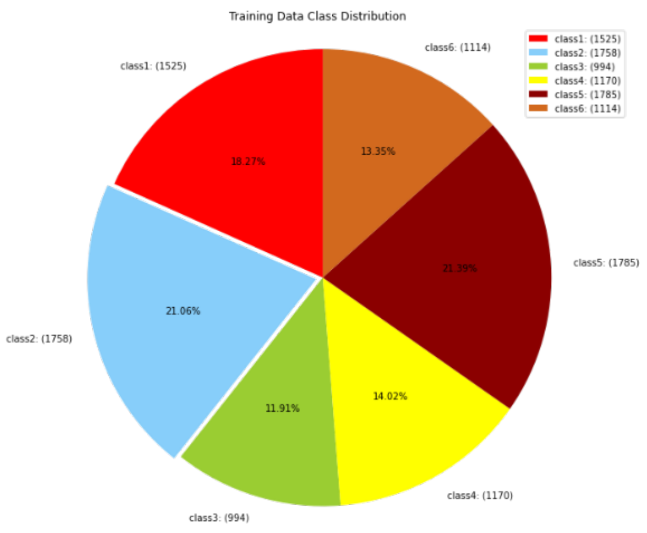

Fig. 1. Training Data Class Distribution

2.2. Multiple Feature Analysis

There are 128 features in the training data. Each feature is an anonymous feature, it can be seen that data in all features are continuous numerical data.

TABLE I shows the statistic information of feature 0 to feature 5.

TABLE I. STATISTIC INFORMATION

Feature	0	1	2	3	4	5
Count	8346	8346	8346	8346	8346	8346
mean	49902.61	6.686965	12.82396	18.5636	26.48538	-9.07464
std	68758.94	16.29977	17.43445	24.64926	36.98463	12.54488
min	-16757.6	0.088287	0.0001	0.0001	0.0001	-127.407
max	668677.1	1339.879	164.2579	221.5585	664.2081	-0.02167

By the analysis of skewness and kurtosis, it can be seen that the skewness and kurtosis values of some features are small. This shows that these features obey gaussian distribution. However, the features that both values are high do not obey gaussian distribution.

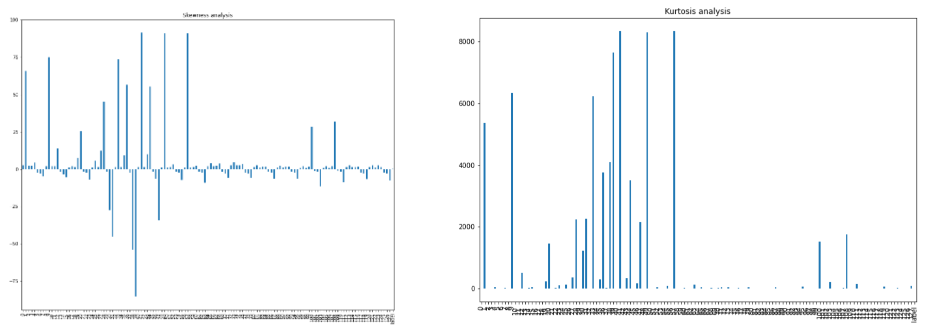

Fig. 2. Skewness(left) and Kurtosis(right) for every features

2.2.1. Feature-Feature Analysis

2.2.1.1. Correlation coefficient

Since the specific meaning of the label by this project is unknown, the correlation analysis of features is necessary to some extent. Here, based on Pearson correlation coefficient, which is used to measure the degree of correlation between two variables and its values is between -1 and 1, we carry out overall or partial correlation analysis of features and draw heatmaps to help the analysis.

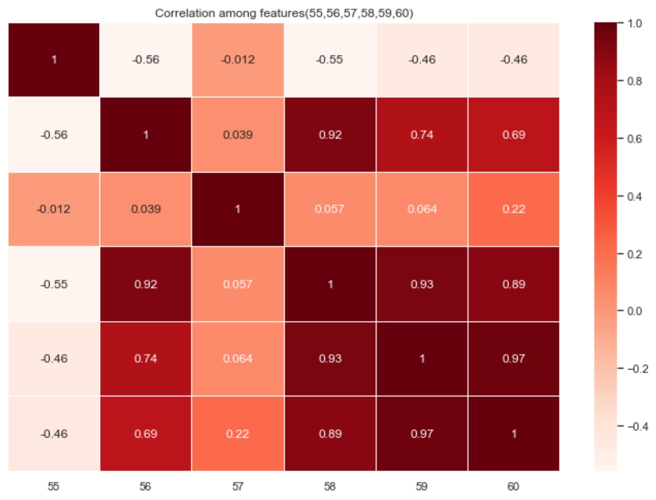

Fig. 3. Correlation Analysis among Features

Fig. 3 shows the degree of strength associated with different features. And, setting a threshold (e.g. corr > 0.96) and we consider that these two feature variables contribute the same to the final model prediction results, hence, one of them can be chosen.

2.2.2. Single Feature Analysis

2.2.2.1. chi-square test

The importance of all features by chi-square test. The importance score is shown as Fig. X. It can be seen that there are some features whose importance score is nearly 0. This shows that these features are nearly useless. The importance score graph is shown as Fig.4.

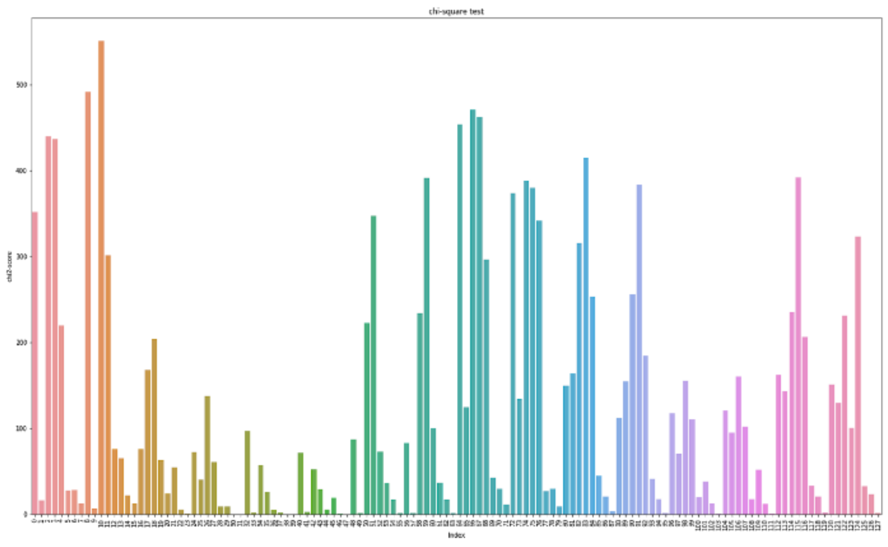

Fig. 4. Importance Score of all features calculated by chi-square test

2.2.2.2. Correlation Between Feature and Class

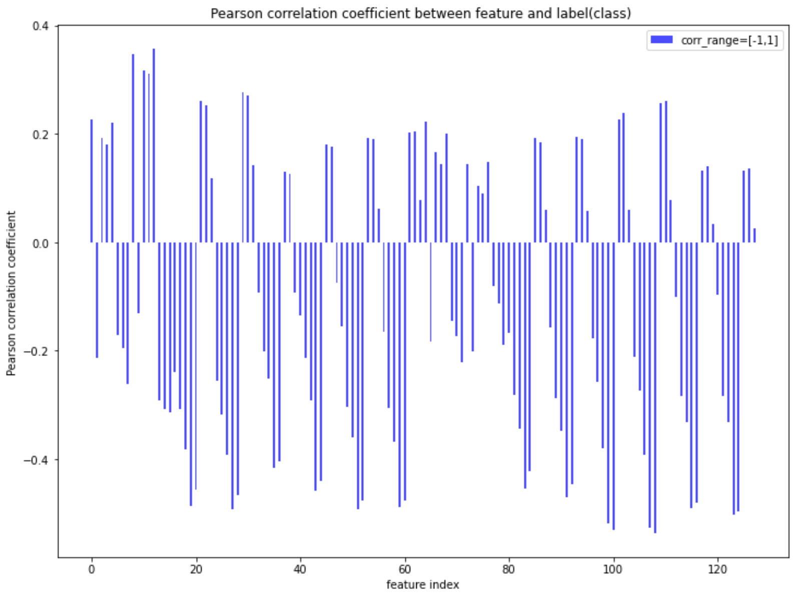

Fig. 5. Pearson correlation coefficient between feature and label

For the relationship between feature and class(label), we also conducted correlation coefficient analysis. From Fig. 5, the coefficient greater than 0 indicates a positive correlation, while the coefficient less than 0 shows a negative correlation. Then, in the subsequent process of feature selection, the correlation coefficient of features and class greater than 0.15 and less than -0.15 would be selected for model training and prediction. And Meanwhile, this method will be compared with other feature analysis and screening methods, such as chi-square test, Gini index and PCA, and finally the method with the best score will be chosen for prediction of the test set(X_test).

2.2.2.3. Gini index analysis

Gini index is another way to analysis the relevance between the feature and class. It is originally used in Decision Tree Classifier to pick features for splitting branches, but we can use it as an intuitive way to show the importance of each feature. It is calculated as total reduction of the Gini purity brought by that feature. If the features importance is lower than threshold, it would be treated as irrelevant features.

$Gini\ inpurity = \sum_{i=1}^{j} P_i(1 - P_i)$, ($P_i$is the probability of an item with i class being chosen)

$Gini\ gian = Gini(Parent\ node) - Weight\ sum\ of\ Gini(Child\ branch)$

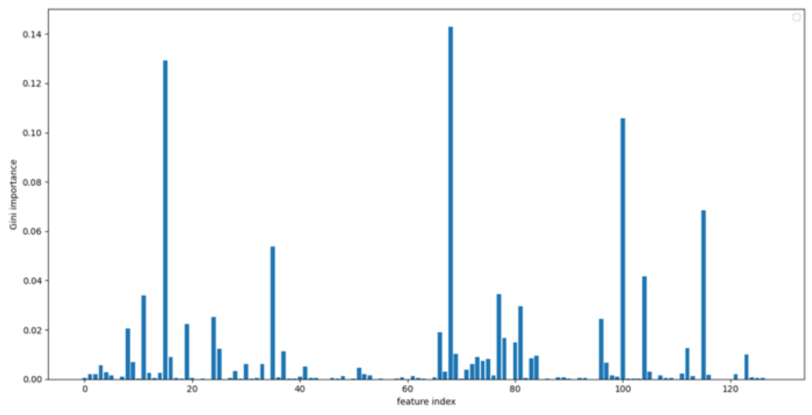

Fig. 6. Gini importance of all features

2.3. Data Clean

2.3.1. Retaining valid values

Retaining valid values requires removing redundant data and dealing with null and infinite values in the data. Null values can be handled by means of mean filling or direct deletion. It is checked that there are no duplicate rows, null values and infinite values in the given raw data.

2.3.2. Outlier detection

Outliers are those points that show complete inconsistency with the overall sample in these properties. If the algorithm is sensitive to outliers, then the generated model cannot have a good expression for the whole sample, so the prediction will be inaccurate. In this project, we first divided all the data into six classes according to the labels. For each class of data, the following outlier detection methods were adopted:

In addition, multiple approaches have been attempted for outlier detection in this project.

1. **Isolation Forest**: It is an unsupervised learning approach used in continuous numerical data to detect outliers. It splits the data recursively until all data is isolated. The outliers are easy to split to be isolated, while the dense data should be split many times.
2. **3-σ Rule**: Data approximately conform to normal distribution. Under the 3σ principle, we consider points that are 3 times the standard deviation away from the mean as outliers.
3. **K-means**: Treat this kind of data as a cluster and calculate the distance from each point to the center of the cluster. If it exceeds a certain range, the point will be regarded as an exception point.
4. **Box plot**: A box plot is a graphical representation of data through its quartiles. The upper and lower whiskers are used as the boundary of the data distribution. Any data point above the upper whisker or below the lower whisker can be considered an outlier. The box plot is more robust than the 3σ algorithm.

However, compared with the results of the later training, we found that the effect after the above method used to delete the abnormal points was not ideal, or even inferior to the original data. The reason may be that the original data itself is relatively clean, but these outlier cleaning algorithms eliminate small batches of local data, which may lead to changes in the distribution of data, thus having a negative impact on the training.

3. Methodology

3.1. Pre-processing

3.1.1. Normalization and Standardization

According to different models, we have used different pre-processing methods of data, namely normalization and standardization. In the multi-standard evaluation system, due to the different nature of each evaluation, it usually has different dimensions and orders of magnitude. When there is a big difference in the level of magnitude of various features, if the original value is directly used for analysis, the role of indicators with higher values in the comprehensive analysis will be highlighted, and the role of indicators with lower values will be relatively weakened. Therefore, in order to ensure the reliability of the results, it is necessary to standardize and normalize the original data. The benefits of such treatment include improving the convergence speed of the model, improving the accuracy of the model, and preventing the gradient explosion of the model in deep learning.

- **Min-max normalization：**

In general, normalization is a linear transformation of the original data, mapping the data values between [0, 1]. Where the minimum becomes 0 and the maximum becomes 1. The specific conversion formula is as follows:

$$x^* = \frac{x - x_{min}}{x_{max} - x_{min}}$$

- **Z-score standardization:**

When the distribution of the feature value can be approximate to the normal distribution, standardization can be used. The processed data conform to the standard normal distribution, that is, the mean value μ is 0 and the standard deviation σ is 1. The specific conversion formula is as follows:

$$x^* = \frac{x - \mu}{\sigma}$$

3.2. Feature engineering (feature extraction and selection)

3.2.1. General Approach

3.2.1.1. Pearson correlation coefficient

The method of the correlation coefficients is mainly used for the analysis of features and features, as well as between features and classes. To be more specific, the first is to calculate the correlation coefficient between any two features and set the threshold value of 0.96, and the second one is to calculate between features and class (and set the threshold $-0.15 <$ "unvalid features" < 0.15). Finally, 32 valid dimensions can be obtained.

The formula principle of Pearson Correlation Coefficient is as follows, $\boldsymbol{r} = \frac{\sum_{i=1}^{n} \sum X_i - \bar{X}(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^{n} (X_i - \bar{X})^2} \sqrt{\sum_{i=1}^{n} (Y_i - \bar{Y})^2}}$, where $\boldsymbol{X}$ and $\boldsymbol{Y}$ represent two different features sample respectively.

3.2.2. Principal Component Analysis (PCA)

PCA is an unsupervised linear dimension reduction method that faithfully reproduces all the information of the original data as far as possible. Specifically, z_i (input after dimensionality reduction) is the orthographic projection of the original training input sample $\boldsymbol{x_i}$ (equal to X_train in this project), the projection matrix $\boldsymbol{T}$ make z_i and x_i as similar as possible. First, we transform the m-dimensional $\boldsymbol{z_i}$ into the d-dimensional space by $\boldsymbol{T}^T$, and then calculate the distance between z_i and $\boldsymbol{x_i}$. And then the sum of distance between the training sample $\boldsymbol{T}^T \boldsymbol{z_i} (= \boldsymbol{T}^T \boldsymbol{T} \boldsymbol{x_i})$ and $\boldsymbol{x_i}$ squared can be expressed by the formula, $\sum_{i=1}^{n} \left\| \boldsymbol{T}^T \boldsymbol{T}_{x_i} - \boldsymbol{x_i} \right\|^2 = -tr\left(\boldsymbol{TCT}^T\right) + tr(\boldsymbol{C})$, where C is the covariance matrix of the training sample.

The learning process of PCA can be expressed by the formula, $\max_{T \in \mathbb{R} m \times d} tr(\boldsymbol{TCT}^T) \quad if \quad \boldsymbol{TT}^T = I_m$, where I_m is a $m \times m$ unit matrix.

In this experiment, multiple models applied this method to achieve feature extraction, including Random Forest, Light GBM, Adaboost and Xgboost. And for different classifier models, by PCA, we manually adjusted the dimension reduction degree of training samples to achieve the best training effect, and the finial model evaluation will be showed in the part of result below.

3.2.3. Neural Network Approach

As a kind of model in NN, Auto-encoder uses unsupervised learning method to extract and represent features of high-dimensional data efficiently. The Auto-encoder framework contains two major modules: the encoding process and the decoding process. Batch normalization means that the distribution of input values of arbitrary neurons in each layer of the neural network is forced back to the standard normal distribution with a mean of 0 and a variance of 1. Dropout is to significantly reduce overfitting by omitting half of the hidden layer nodes in each training batch.

In the process of Encode, during the Batch Normalization period, the feature space of 128 dimensions is randomly input Gaussian noise to prevent overfitting, and then it is mapped to the 64-dimension feature space. After dropout, it is then mapped to the 32-dimensional feature space. That is the encoded features we need. In the process of Decode, the 32-dimensional encoded features were mapped to the 64-dimensional feature space after dropout, and then to the 128-dimensional feature space after another dropout, which is the original input dimension. Then through the mapping of 64 dimensions feature space, decoded features finally get a 6-dimension label. The decoded features and label outputs were combined as Auto-encoder for training, and the encoded features were the 32-dimensional feature representation extracted by us. At this stage, the feature representation that started with 128 dimensions has been reduced to 32 dimensions. The structure of auto-encoder network is shown as Fig. 7.

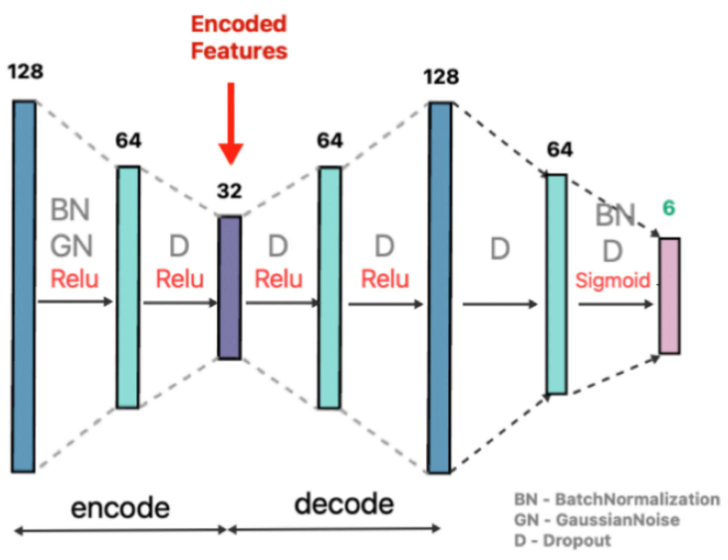

Fig. 7. Structure of Auto-encoder Network

3.3. Model Selection

Adaboost: The working mechanism of Boosting algorithm is:

- First of all, from the training set with initial weights training out of a weak learning 1, according to the weak learning of error rate performance to update the weights of the training sample, makes the weak learning 1 study the training of the high error rate before the weight of sample points higher, make the error rate of high point in the back of the weak learning 2 received more attention.
- Then, based on the training set after adjusting the weight, the weak learner is trained 2. This process is repeated until the number of weak learners reaches the pre-specified number T.
- Finally, the T weak learners are integrated through the set strategy to obtain the final strong learner.
- The final strong classifier is: $f(x) = sign \sum_{k=1}^{K} (\alpha_k G_k(x))$

The classification accuracy of Adaboost is very high. Theoretically any learner can be used in Adaboost. But generally speaking, the most widely used Adaboost weak learners are decision trees and neural networks. In this project, the decision tree is finally selected as the weak learner. Adaboost is very sensitive to the samples of outliers, so it may obtain a higher weight in the generation, which will affect the prediction accuracy of the final strong learner. However, there are fewer outliers in the data this time, so Adaboost also gets a good effect in the training.

Random forest: R Random forest is a classifier that build a number of different decision tree classifier on subsets of original feature set and aggregates all results to use prediction which appears the most of times.

Extremely Tree: Compared with decision tree, extreme forests have not only random samples, but also random splitting conditions, although this is not the best splitting condition (the splitting condition of condition of decision trees is the maximum information gain.) Furthermore, as in a random forest, a random subset of candidate feature is used, but instead of looking for the most distinct thresholds, thresholds are drawn randomly for each candidate feature and the best of these randomly generated thresholds is selected as the partition rule.

Multilayer Perceptron: Generally, perceptions have only one input layer and one output layer, which leads to the limited learning ability and can only solve the linear separable problem. Multilayer Perceptron is an extension of feedforward artificial neural network and perceptron. In addition to input layer and output layer, there are hidden layers between them. In this experiment, the original data and the 32-dimensional encoded feature extracted by the Auto-encoder are concatenated together as the input layer, and the neural network selects the features autonomously. The output layer is a sparse representation of the label. The specific structure of hidden layers and between layers can be referred to the following Fig. 8.

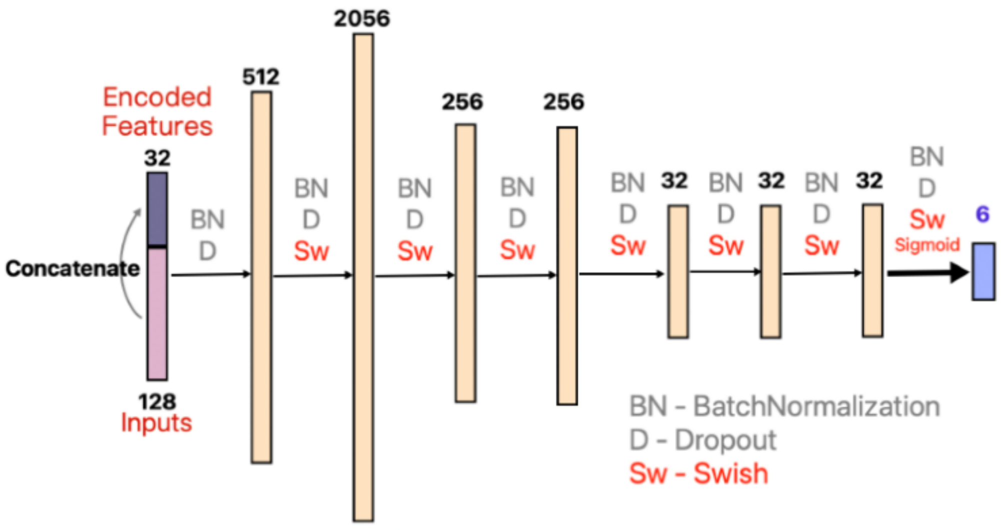

Fig. 8. Structure of MLP

Ensemble (Voting): This method integrates various method and calculates the number of predictions for each label. Finally, choose the label which appears most times as the final prediction. The purpose is to balance the performance of a number of equally good learners to eliminate their respective flaws.

Additionally, the classifiers of SVM, KNN, GradientBoost, Logistic Regression, Bagging, Light GBM and XGBoost are also used in this experiment, and the performances are reflected in the results.

3.4. Hyper-parameter Tuning

Two kinds of automatic parameters selection have been tried, grid search and randomized search, along with cross validation to get the best parameter combination.

3.4.1. Grid Search

A method of tuning parameters, using exhaustive Search, that is, among all the candidate parameter selection, through the loop to try every possibility, the best performance of the parameter is the result. It works like finding the maximum value in an array. This method is more accurate when exploring relatively few parameter combinations.

3.4.2. K-fold Cross Validation

In this experiment, the training set is divided into k (k=5) subsamples, one single subsample is retained as the data of the validation model, and the other k-1 samples are used for training. Cross validation is repeated k times, once for each subsample, averaging k results or using some other combination, resulting in a single estimate. K-CV can effectively avoid the occurrence of over-learning and under-learning states, and the final results are more persuasive.

3.5. Evaluation Metrics

In this project, F1-score is used to evaluate the performance of each model. This evaluation metric considers both precision and recall at the same time. The calculation formula is as follow:

$$\mathrm{f1} = (\frac{1}{n}\sum_{k=1}^{n}\frac{2 \times \mathrm{precision}_k \times \mathrm{recall}_k}{\mathrm{precision}_k + \mathrm{recall}_k})^2$$

where n is the number of classes.

4. Result

4.1. Analysis of Feature Importance

It can be seen from Fig. 9, when using feature selection and feature extraction, the performance also increases with the increase of the number of features, but it will converge. More features will not improve the improvement.

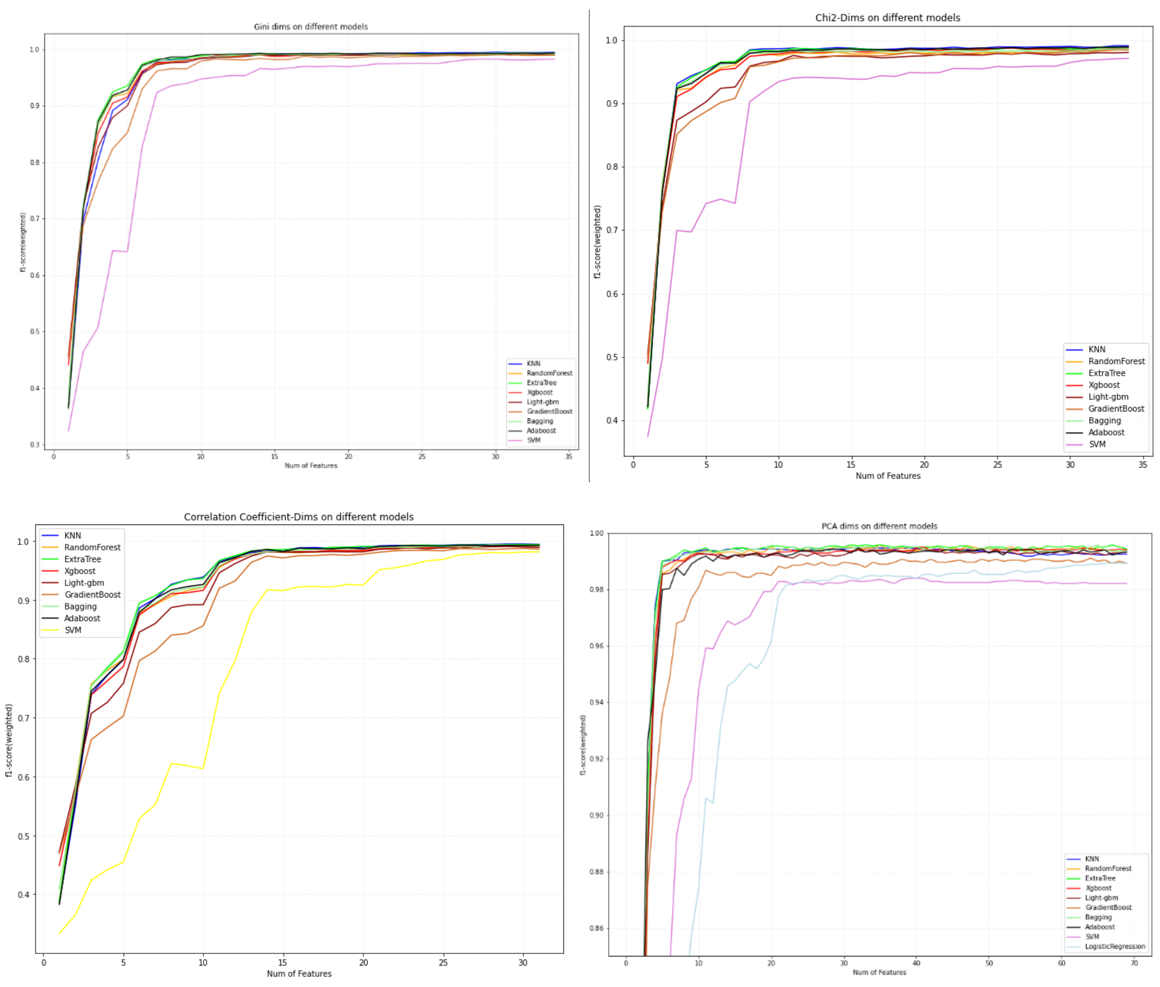

Fig. 9. Performance of different feature choice (Gini Index (left upper), chi-square test (right upper), correlation coefficient (left lower), PCA dimensional deduction (right lower))

4.2. Performance of Different Hyper-parameters

The process of hypermeters tuning mainly depends on Grid Search and k-fold cross validation, which k is set 5. In TABLE.II shows the details of the selection for two parameters, learning rate and number of estimators, in Adaboost Classifier. By comparing different parameters' performance, f1-score, the best learning rate (=0.5) and estimators (=100) within the given range can be obtained, so as to optimize the training effect of the model.

TABLE II. Performance of AdaBoost (Part)

Learning Rate	**# estimator**	**f1 score**
0.01	100	0.9718
0.01	150	0.9740
0.1	100	0.9863
0.1	150	0.9858
0.5	100	0.9866

0.5	150	0.9860

TABLE III shows a part of process for the tuning of random forests. Finally, the number of estimators, min samples leaf and min samples split are set as 160, 1 and 2.

TABLE III. Performance of Random Forests (Part)

# estimator	**# min samples leaf**	**# min samples split**	**f1 score**
100	1	2	0.9946
150	1	2	0.9950
170	1	2	0.9950
160	1	2	0.9951
160	2	2	0.0094
160	1	3	0.9944

TABLE IV shows a part of process for the tuning of neural network. Finally, learning rate and Laplace smoothing value are set as 0.001 and 0.05.

TABLE IV. Performance of Multi-Layer Perceptron (Part)

Learning Rate	**Laplace Smoothing**	**f1 score**
0.0001	0.05	0.9876
0.0005	0.05	0.9907
0.001	0.05	0.9942
0.0015	0.05	0.9928
0.01	0.05	0.9869
0.001	0.01	0.9931
0.001	0.1	0.9930

Similarly, Grid search along with 5-fold cross validation are also used to tune parameters for other models. The results are shown as TABLE V:

TABLE V. Other models' hyperparameter tuning results

Other models	Bagging	Gradient-Boost	Light-GBM	Extra-Tree	Xgboost	KNN

Selected
Parameters
	# estimator= 170
	Max samples= 6400
	Max features= 34
		# estimator=150	# leaves= 31
			Max depth= -1
			Learning rate=0.1
				Criterion = 'gini'
				Max feature= 'auto'
					Max depth=7
					Learning rate=0.05
					# estimator =1000
					Objective = 'multiclass
						# neighbors =2
						Weight ='uniform'
						Leaf size =5
						P=1

4.3. Evaluation of Models

Fig. 10 shows the f1 score of different models. It can be seen that the scores of most models are over 0.992. After multiple experiments, voting algorithm is relatively steady. Its score can be over 0.996 in all experiments. However, although the f1 scores of other algorithms are nearly 0.99, they will be affected by different random seed. Therefore, voting algorithm is the best.

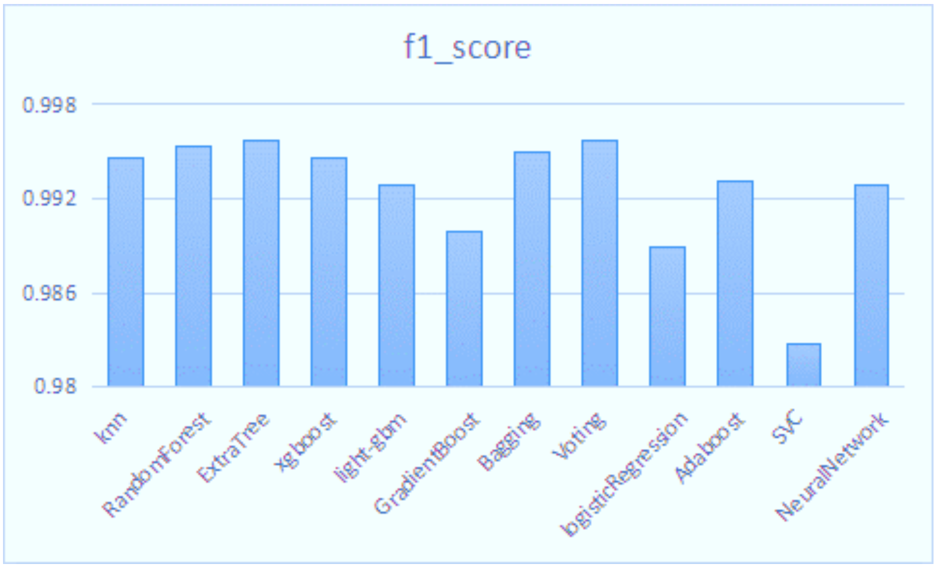

Fig. 10. f1 score of different models

5. Discussion

5.1. Comparison of different models

According to the experiments, Random Forest, ExtraTree and XGBoost are relatively better, but their performance is affected by different random state. However, after integrating 11 models and using voting algorithm, the result keeps steady.

5.2. Metrics selection

The dataset is balanced on classification, and each class is treated equally important. In that case, we want to find an optimal blend of precision and recall, so f1_score is chosen as evaluation metric which combines precision and recall.

5.3. Future improvements

In data preprocessing step, various approaches of detecting outliers have been tried. However, the number of deletions is much larger than expected. In the future, we need to gather more information on data distribution patterns as well as find a better approach to check outlier, and meanwhile, avoid normal data points being deleted accidently.

Additionally, another improvement could be on utilizing validation set more fully. We simply rely on k fold cross validations on training data to prevent overfitting and evaluating performance on validation set. Maybe we can discover a more complicated and randomized approach to choose training data and testing data, but it would potentially cost more computing resource.

6. Conclusion

The purpose of this project is to build a multi-label classifier to predict the data samples containing 128 features into 6 possible classes. After preprocessing and analyzing the original dataset, our group chose Neural Network, KNN, Random Forest, Extra Tree, Logistic Regression, Adaboost, Xgboost, Lightgbm, GradientBoost, Bagging, SVC as base classifiers. Most of them have a good accuracy score which is higher than 0.99. After training and tunning hyper parameters for each classifier separately, we ensemble those results and use the majority vote as new prediction. It is proven that ensemble method can make a more stable and accurate prediction. The final f1 score on validation dataset is 0.996. It is shown that the final model is good enough to predict classes on unknown dataset.

Reference

[1] Wikipedia contributors. "Pearson correlation coefficient." *Wikipedia, The Free Encyclopedia*. Wikipedia, The Free Encyclopedia, 24 Apr. 2021. Web. 28 Apr. 2021.

[2] Wikipedia contributors. "68–95–99.7 rule." *Wikipedia, The Free Encyclopedia*. Wikipedia, The Free Encyclopedia, 12 Apr. 2021. Web. 28 Apr. 2021.

[3] Parviainen, E., 2010, September. Deep bottleneck classifiers in supervised dimension reduction. In *International Conference on Artificial Neural Networks* (pp. 1-10). Springer, Berlin, Heidelberg.

[4] Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q. and Liu, T.Y., 2017. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, *30*, pp.3146-3154.

[5] Chen, T. and Guestrin, C., 2016, August. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining* (pp. 785-794).