COMP9417 MLinUnKnown Report

Yunqi Wang
School of Computer Science and Engineering (CSE)
The University of New South Wales (UNSW Sydney)
Sydney, Australia
z5324194@ad.unsw.edu.au

I. Introduction

The 21st century is a century of information technology. Under this background, the artificial intelligence is becoming a major topic discussion, and it also will continue to be the direction of future social development. As an important research field of artificial intelligence, machine learning is developing rapidly. This report discusses how to use the knowledge of machine learning to train and optimize a better model to efficiently predict the given data.

In this task, we used what we learned in COMP9417 to filter and normalize the given data, select the optimal parameters, and final established a good model to predict the labels of unknown data.

The challenges of this task are: 1. Too large number of features that the given dataset contains. 2. The selection of training model and the selection of its parameters.

To achieve this, we have made the following attempts:

1) Data re-processing: remove nan values and noises.

2) The selection of the features that contribute most to the model prediction from a large number of features: by using selection criteria: a. the correlation between features and labels. b. the distribution of different labels. c. the dependency between features.

3) The normalization of the data： Three methods of normalization were chosen in this report: Standard normalization, Robust normalization and Normalizer to normalize the data.

4) Parameter selection of classifiers: SVM, RandomForest, DecisionTree, Xgboost and Lightgbm were selected and optimized in this task. And the function we selected to compute the parameter selection is GridSearchCV(sklearn).

5) The selection of the classifiers: by testing the performance of the above classifiers on the given dataset and use evaluation metrics to analyze.

After the experimental process above, we found that RandomForest classifier can train to get the best model, which finally chose the f1_score of 0.995320 on the given validation set.

II. Exploratory Data Analysis

A. Outlier Detection

When we are training machine learning algorithms or applying statistical techniques, outliers can be a serious problem. They do not have the characteristics of describing the dataset. So, they usually result in measurement errors.

There are some methods for outlier detection such as Numeric Outlier, Z-Score and 3σ. According to the observation, the dataset does not conform to the normal distribution, so Z-Score and 3σ are not suitable. Therefore, we choose Numeric Outlier methods and implement it through box-plot.

B. Feature exploration

In order to make the training model get more effective data, we usually need to screen out the less valid features, and retain the features that contribute lots to the prediction. The standard is to judge the relevance between features and labels, and check the distribution of labels, as well as the correlation between features. Details are showed down below:

1) The relationship between features and label

When training data, the model training is affected by the features that are almost unrelated to labels. As a result, we need to remove these features to avoid the model learning the independent features. In order to achieve this, we computed ANOVA F values for every feature to get the relevancy between features and labels. The plot below shows the result:

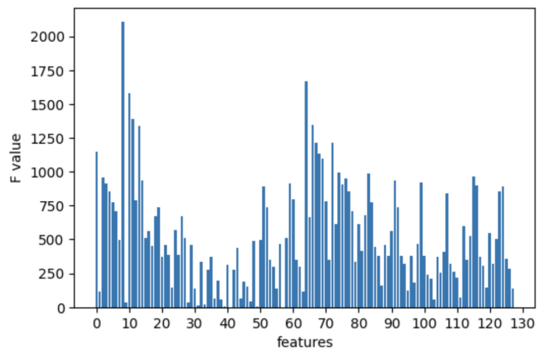

Fig. 1 Relevancy between features and labels

2) The distribution of label

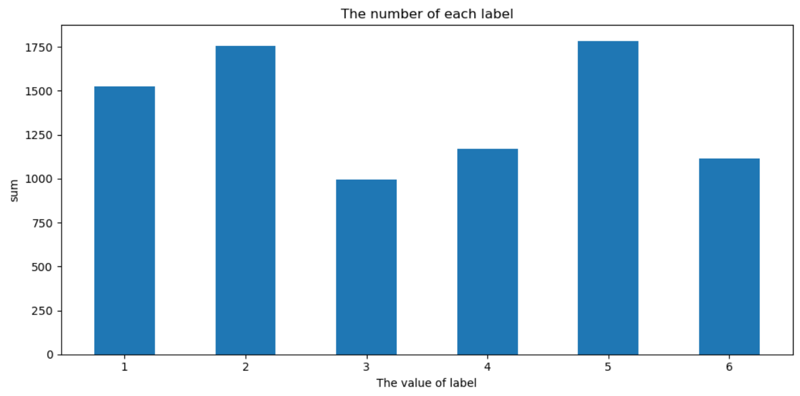

Fig.2 Distribution of labels

This figure is the distribution of labels. The x-axis represents different labels and the y-axis represents the number of instances belonging to each label. Normally, when we analyze the distribution of labels, we should focus on the labels which have few instances. If we ignore them and train model directly, the performance of model could be terrible, because the model might not learn these labels correctly due to insufficient data of certain labels. However, for this task, data size of each label is similar. In addition, even if there were labels which have insufficient data, we could not be sure that this condition is unnormal, because we were not told what the data represent, which means that it possible for some labels to actually only have a little distribution. Therefore, we did not do data processing for labels.

3) The relationship between features

After checking the features which do not influence label, we also should check the correlation of rest of features. In many cases of machine learning which have data of high dimensionality, researchers commonly apply feature selection methods mainly focus on calculating the correlation between two features to reduce the number of features (Jiang and Wang, 2016)[1]. It means that if a pair of features have a strong relation, we could delete one of them to reduce the dimensionality of data for more efficient learning.

According to Akoglu[2], the Pearson's correlation coefficient and Spearman's rank correlation coefficient is common measures of measuring the correlation. Pearson can be used when features are normally distributed continuous variables, while Spearman is for non-normal distributions. Hence, we should identify the distributions of features to decide which method should be applied.

After selecting the correlation method, we calculate the correlation between 128 features to obtain a 128*128 symmetric matrix. Determine the absolute value of the correlation between each column, if it is greater than the threshold, record it, and finally get a dictionary, the key is the value of 128 columns, the corresponding value is a list of the column that is judged to be correlated to and greater than the key. In the results obtained, there are cases where column A is related to column B, column A is related to column C, but B and C are not. Our decision is to delete column A to simplify the feature and ensure the integrity at the same time.

Regarding the selection of the threshold, it can be divided into the following situations according to the value of the correlation coefficient r: When $|r| \geq 0.8$, it can be regarded as a high correlation between two features; when $0.5 \leq |r| < 0.8$, it can be regarded as a moderate correlation; When $\leq |r| < 0.5$, it is regarded as low correlation; when $|r| < 0.3$, it indicates that the correlation between the two features is extremely weak and can be regarded as irrelevant.

III. Methodology

A. Dataset Description

Table I. The dataset description

Dataset					
Trainingset			Validationset		
Size	Number of features	Labels	Size	Number of features	Labels
8346	128	1,2,3,4,5,6	2782	128	1,2,3,4,5,6

From the table I above, we can see that the number of features is 128, which is too large for model training. As a result, we have carried out the following attempts to filter out some irrelevant features.

B. *Outlier detection*

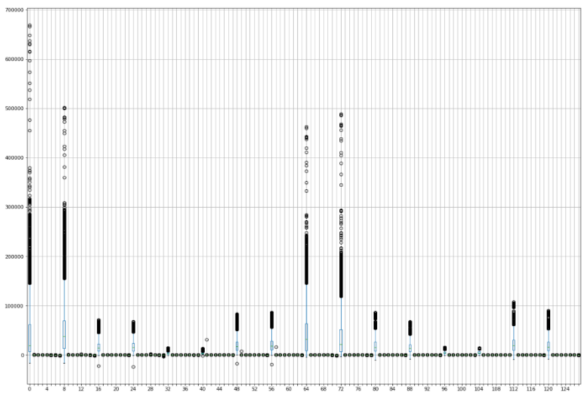
Fig.3 Box-plot based on original dataset.

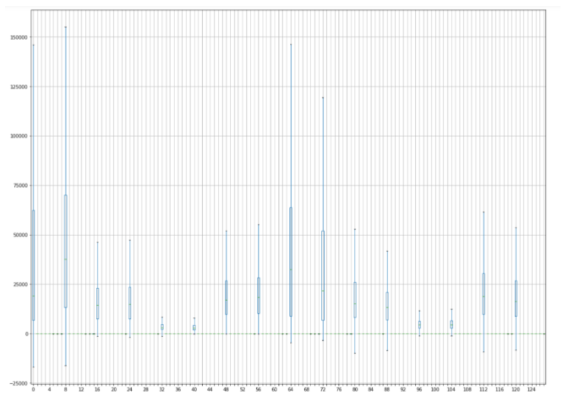
Fig.4 Box-plot based on dataset that outliers are replaced.

Figure3 is the box-plot based on original dataset. According to this, we can observe the outliers, black points. Considering the numerous number of outliers, we don't remove those values. We replace the outliers that exceed the upper bound with the maximum value among normal. Likely, we replace the outliers below the lower bound with the smallest value among normal.

Figure4 is the box-plot based on dataset that outliers are replaced. We use the new dataset to fit our model and test the model through validation dataset, however, the accuracy and f1-score drop. Then we try to use mode to replace the outliers. This method also doesn't perform well. Given that we don't know what the data is and the bad performance of the above methods, we decide not to handle the outliers to avoid replacing those possible special true data.

C. *Feature Filter*

1) By feature-to-label relationship

In order to remove these features to avoid the model learning the independent features, we have made the following attempts: 1) computing ANOVA F values for every feature to get the relevancy between features and label. 2) Train and test the data filtered by different thresholds on the classifier (by default parameters except random_state=0), and plot the line chart of their accuracy to find a best threshold value. 3) filter out the features which has less relation to labels.

Firstly, the ANOVA F values for every feature were computed by using 'f_classif' function from 'sklearn.feature_selection'. Then, normalized the F values to get a suitable range for finding threshold.

Secondly, test the features filtered by threshold values in 0.05 increments from 0.2 to 0.8 (when threshold equals to 0.8, the number of selected features is 1, so we stop to 0.8) on Randomforest classifier (by default parameters except random_state=0). The accuracy and f1_score lines plot in figure down below (the accuracy line and f1_score lines are almost identical in the graph):

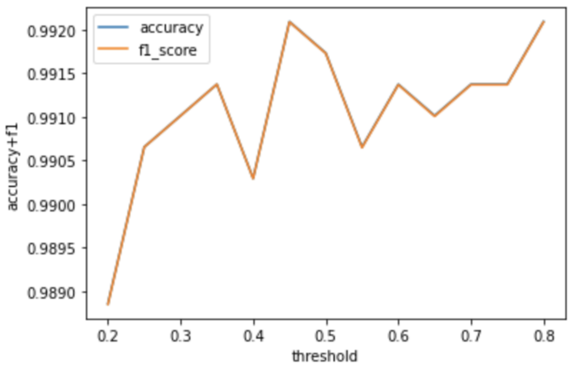

Fig.5 The accuracy and F1 score when threshold in range 0.2-0.8(0.05++)

From figure5, we can see that when threshold in range from 0.4 to 0.55, and from 0.75 to 0.8. The average accuracy and f1_score are highest.

Thirdly, to narrow the threshold value, we chose the two range above and test the accuracy by 0.01 increment of threshold. The results showed in figure 6(the accuracy line and f1_score lines are almost identical in the graph):

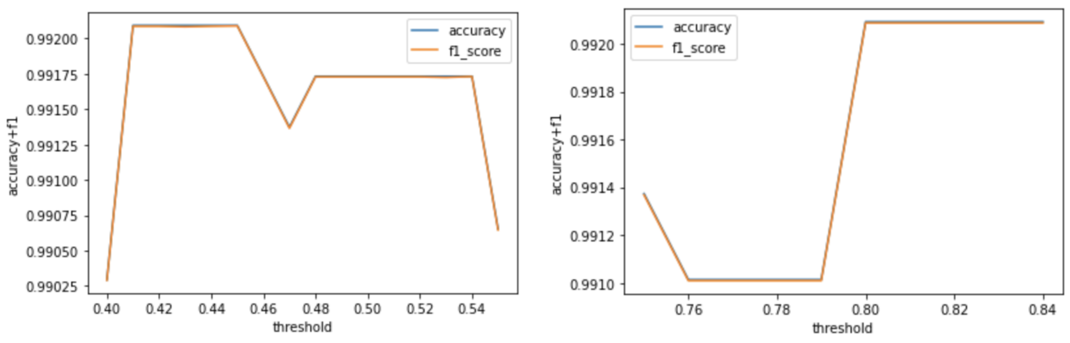

Fig.6 The accuracy and F1 score in range of 0.40-0.55(0.01++) and 0.75-0.8(0.01++)

From the plots above, we can see the average accuracy in the first threshold range are higher than the second one (the average f1_score for the first range is 0.9916651689432063, for the second range is 0.9915887850467289). Besides, when threshold values are taken from the range of 0.79-0.85, the number of selected feature is 1. As a result, we selected 0.43 as the threshold value to filter features. (code of narrowing the searching range and getting value of 0.43 is commented out in the submitted code)

Finally, the number of features screened out is 22. The rows of deleting features are: [0, 2, 3, 8, 10, 11, 13, 14, 59, 64, 66, 67, 68, 69, 72, 74, 75, 76, 83, 91, 99, 115].

2) By feature-to-feature relation

We used stats.kstest to test each column of training data and found that each pvalue of KstestResult is lower than 0.05, which means that There is no data of one column is normally distributed. To verify the above conclusion, we also select a few columns randomly and draw their distribution.

From the picture of distribution of column 18, we could observe that the most values gather between 0 and 10, however, there was no negative value. Hence, it did not have symmetry, which meant that column 18 was not normal distribution. In a similar way, the column 45 was also a non-normal distribution.

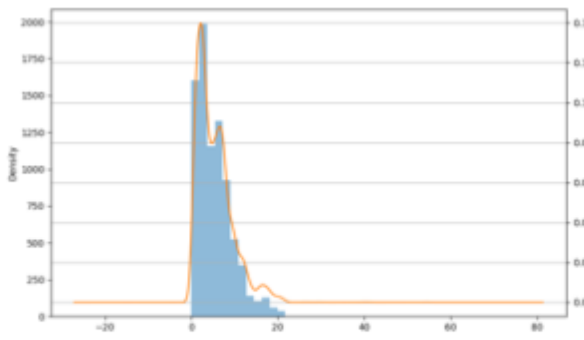
Fig.7 Distribution of column 18

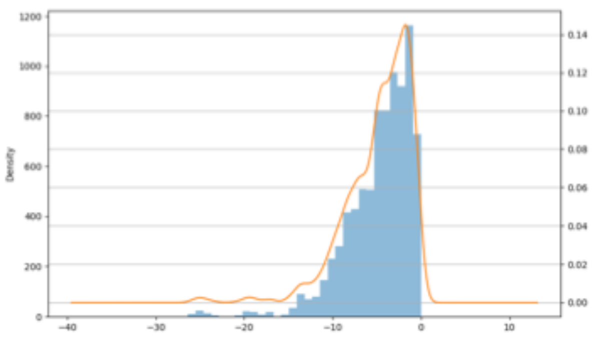
Fig.8 Distribution of column 45

Therefore, we decided to apply the Spearman's rank correlation coefficient to identify the correlation of different columns.

We use pandas.Dataframe.corr('spearman') to calculate the correlation and use f1_score and accuracy from the RandomForest classifier with random_state=0 and other parameters default to choose an appropriate threshold. Comparing with the results in different threshold as follows, we can see that when the threshold is set large, the number of correlation between features that higher than the threshold is less than the normal, therefore when the threshold decrease, there would be more features left; when the threshold is set excessive low, the frequency of occurrence of case that A, B, C above mentioned decrease so that the features left are less because it just remains one of A, B, C. The best accuracy and f1_score, however, do not correspond the peak of that.

Table II The accuracy and F1 score in different threshold

Threshold	Number of features left	Accuracy	f1_score
0.9	77	0.9906542056074766	0.9906533280707969
0.85	97	0.99137311286844	0.991366659093551
0.8	99	0.992451473759885	0.9924454988045878
0.75	103	0.9906542056074766	0.990651638868225
0.7	103	0.9906542056074766	0.990651638868225
0.5	100	0.9920920201294033	0.9920894237903874
0.3	86	0.9917325664989216	0.991727161020065

Finally, we choose the correlation threshold: 0.8.

In conclusion, with deleting the features that is unrelated to labels and related to other features, we have a new training set which contains 99 columns, that is [4, 5, 6, 7, 12, 15, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 33, 34, 35, 36, 37, 38, 40, 41, 42, 43, 44, 45, 46, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 60, 61, 62, 63, 65, 70, 71, 73, 77, 78, 79, 80, 81, 82, 84, 85, 86, 87, 88, 89, 90, 92, 93, 94, 95, 96, 97, 98, 100, 101, 102, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127].

D. Normalization

The Normalization of data is to scale the data to a small interval. Normalization can convert the data into a dimensionless pure value, which is convenient for comparison and weighting of features of different units or magnitudes. According to the figure1, we can find that there is a big gap in the magnitude of data from different features. This may affect the convergence and accuracy of the model. Therefore, we decide to normalize data. We choose three methods, Standard normalization, Robust normalization and Normalizer to normalize the data.

1) Standard normalization (StandardScaler).

Standard normalization based on the formula:

$$x' = \frac{x - \mu}{\sigma}$$

to rescale date into $[-1, 1]$. (μ denotes the mean of values of one feature, σ denotes their standard deviation.)
After implementation of this method, the accuracy and f1-score decrease.

2) Robust normalization (RobustScaler).

Robust normalization based on the formula:

$$x' = \frac{x - x_{median}}{Q3 - Q1}$$

to rescale data. (x_{median} denote the median of values of one feature, $Q3$ is the median of the upper half of the dataset, $Q1$ is the median of the lower half of the dataset.)
After implementation of this method, the accuracy and f1-score decrease.

3) Normalizer

This normalization based on the formula:

$$x' = \frac{x}{[x_1{}^2 + x_2{}^2 + x_3{}^2 + \ldots + x_n{}^2]^{\frac{1}{2}}}$$

After implementation of this method, the accuracy and f1-score increase.

Table III Model scores after different normalization of the data. (RandomForest model(random_state = 0))

	accuracy	f1	roc_auc	recall	precision
data without normalization	0.992451	0.992445	0.999137	0.992451	0.992478
StandardScaler	0.982746	0.98273	0.998555	0.982746	0.983253
RobustScaler	0.989216	0.989207	0.998971	0.989216	0.989271
Normalizer	0.995327	0.995318	0.999329	0.995327	0.995366

E. Model selection

1) Light gbm

Light gbm is a gradient boosting framework based on decision tree algorithm, mainly for high-performance analysis of large-scale of data. It uses the histogram algorithm to reduce memory consumption to improve efficiency, and the leaf-wise leaf growth strategy with depth limitation to reduce errors and control overfitting.

2) SVM

The basic model of support vector machine (SVM)[3] is to find the best separation hyperplane in the feature space. In this way, the interval between different classes on the training set is the largest. SVM is a supervised learning algorithm mainly used to solve two classification problems, and it can also be used for multi classification problems, which has become one of the most popular classifiers. Therefore, we select SVM and train a model on the given dataset to compare with other models to check its performance and try to get best result.

3) Decision Tree

Decision Tree: Decision tree is a nonparametric supervised learning method. It can summarize decision rules from a series of data with features and labels, and present these rules with the structure of tree graph to solve the problem of classification or regression. Decision tree algorithm is easy to understand, suitable for all kinds of data, and has good performance in solving all kinds of problems. Therefore, this method is used for training, and it is expected to get a better model and prediction results.

4) Random forest

Random forest: This is a very representative Bagging integration algorithm. All of its base evaluators are decision trees, and the forest composed of classification trees is called random forest classifier. It could increase diversity of trees in ensemble (n_estimators controls it) to reduce variance without effects on bias. Each tree only considers a subset of features (max features parameters), it might cost less time to train. Hence, we select it to train our model.

5) Xgboost

Xgboost stands for extreme gradient Boosting, that implements gradient boosting framework efficiently.

F. Selection of hyper-parameter

1) Light gbm

Firstly, choose the parameters {random_state, n_estimators, max_depth, num_leaves, learning_rate=, reg_alpha, reg_lambda} that have significant impact on the predict result. Take loop in a large range with large intervals and select an interval with better performance for the parameter. Secondly, use GridSearchCV to accurately search for the best value of the parameter. Repeat this step with update the parameter and get the final result of each parameter. Then substitute the obtained parameters, predict the valid data set and calculate the f1 score.

Finally, we have n_estimators=130, max_depth=16, num_leaves=50, learning_rate=0.1, reg_alpha=0.003, reg_lambda=0.1

2) SVM

Firstly, to narrow the range of parameters, we set three values to do the GridSearchCV, when the best parameter that GridSearchCV returns are the middle one, narrow the range. Otherwise, re-set the range which make the returned value in middle, and retry this step again. Try the step above several times until the range in minimize.

Secondly, using the minimized range to search the best parameters by GridSearchCV. The scoring criteria is ‘f1_weighted’. The best parameters that we find are: {'C': 3000000, 'decision_function_shape': 'ovo', 'gamma': 0.8, 'kernel': ‘rbf'}

Finally, the f1_score of SVM model by using the best parameters in ‘valset’ is: f1_score= 0.9895684883144912.

3) Decision Tree

First of all, select four parameters {min_samples_leaf, max_depth, min_samples_split, max_features}. The prediction results is defined by the parameters so that we can find the relatively high accurcy or f1_score through some tool functions. Next, according to sklearn packetage, find the GridSearchCV function to find the best parameters. During this section, design some function with smaller range, change the range and compare f1_scores to detect the best parameters. At last, the best parameters are: {min_samples_leaf = 1, max_depth = 20, min_samples_split = 2, max_features = 50}.

4) Random forest

There are several parameters in Random Forest Classifier: n_estimators, max_depth, min_samples_split, min_samples_leaf, max_features and so on. For each parameter of this model, we drew a figure showing the variation tendency of f1 score as the parameter changes. With figures, we could confirm the approximate location of the best parameters. Then, we could use GridSearchCV to find the exact values. The n_estimators is the most important parameters. Increasing it can reduce the variance of model without effects on bias, however, too large n_estimators might costly. After drawing and GridSearchCV, we found the best n_estimators is 153 and f1 score is 0.997002. Then, we used the similar way to search the max_depth. We found that f1 score increased with max_depth and it did not change anymore after the max_depth exceeded a certain value. The stable f1 score was still 0.997002, which meant that the max_depth should not be restricted and if we could use other parameters to increase the complexity of the model, the f1 score might continue to increase. However, after increasing the max_features, we found the f1 score reduced and the best max_features is the default value. Therefore, we could conclude that the parameters are already optimal. {'Random_state':0, 'n_estimators':153}

5) Xgboost

We use GridSearchCV combined with Xgboost classifier to find the optimal parameters. Although it is best to adjust all parameters at once, considering the performence of computer, we tuned one or two parameters at once. 'N_estimators' is the first one, followed by 'learning_rate', 'max_depth', 'gamma', 'subsample' and 'colsample_bytree'. Finally, we get the optimal parameters: {'n_estimators': 700, 'learning_rate': 0.25,'max_depth': 4, 'min_child_weight': 2, 'gamma': 0.1, 'subsample': 0.65, 'colsample_bytree': 0.15, 'eval_metric': 'merror', 'objective': 'multi: softprob'}.

IV. Results

The evaluation criteria we chose are the f1_score, accuracy, recall and precision. By traing five classifiers above, the model testing results we got show in the following table III.

A. Data analysis table

Table IV The data analysis table

Classifier			**F1_score**	**accuracy**	**recall**	**precision**
LightGBM classifier	Original dataset(Default parameters)		0.991730	0.991733	0.991733	0.991754
	Feature filter(Default parameters)		0.990646	0.990654	0.990654	0.990685
	Normalization(Default parameters)		0.992088	0.992092	0.992092	0.992123
	Feature filter +Normalization	Default parameters	0.994958	0.994968	0.994968	0.994989
		Best parameters	0.995317	0.995327	0.995327	0.995359

SVM	Original dataset(Default parameters)		0.857684	0.855500	0.855500	0.872912
	Feature filter(Default parameters)		0.708945	0.707405	0.7074047	0.7421679
	Normalization(Default parameters)		0.927560	0.928469	0.928469	0.9308150
	Feature filter +Normalization	Default parameters	0.726770	0.772825	0.772825	0.728302
		Best parameters	0.990288	0.990295	0.990295	0.990377
DecisionTree classifier	Original dataset(Default parameters)		0.966600	0.966571	0.966571	0.966707
	Feature filter(Default parameters)		0.958744	0.958663	0.958663	0.958915
	Normalization(Default parameters)		0.979158	0.979152	0.979152	0.979260
	Feature filter +Normalization	Default parameters	0.980538	0.980590	0.980590	0.980635
		Best parameters	0.984175	0.984184	0.984184	0.984283
RandomForest classifier	Original dataset(Default parameters)		0.991012	0.991014	0.991014	0.991054
	Feature filter(Default parameters)		0.992445	0.992451	0.992451	0.992478
	Normalization(Default parameters)		0.993519	0.993530	0.993530	0.993560
	Feature filter +Normalization	Default parameters	0.995318	0.995327	0.995327	0.995366
		Best parameters	**0.995320**	0.995327	0.995327	0.995373
XgBoost classifier	Original dataset(Default parameters)		0.991728	0.991733	0.991733	0.991777
	Feature filter(Default parameters)		0.989570	0.989576	0.989576	0.989628
	Normalization(Default parameters)		0.991364	0.991373	0.991373	0.991409
	Feature filter +Normalization	Default parameters	0.993878	0.993889	0.993889	0.995015
		Best parameters	0.994960	0.994968	0.994968	0.995015

V. Discussion

A. Methods Comparison

In order to compare the performance of different models after different data processing, the F1 score, accuracy, recall and precision were used as evaluation metrics（The selection of metrics is described in detail below). It can be seen from Table III that:

1) Analysis from the performance on the original dataset (by default parameters)

Compared with the five classifiers, the F1 score of xgboost and lightgbm classifiers are the highest. Their F1 score reached 0.991728 and 0.991730 respectively. However, the multi classification performance of SVM model is much worse than other models, whose F1 score only reached 0.857684.

2) Analysis from the performance of feature filter(by default parameters) :

After deleting the features that have less contribution to the prediction results, the four evaluation metrics of the other four models all declined except RandomForest model increased by 0.0014. Especially for SVM, its evaluation metrics decrease by about 0.15 on average. This might be due to the decline of feature dimension for training, or bad feature filter methods, which is where we need to improve.

3) Analysis from the normalization performance (by default parameters):

After normalizing all the data (without feature filter), except for the slight decline of xgboost, the evaluation metrics of other models all have increased. Among them, the average evaluation metrics of SVM increased by 0.05, which is the highest in all models. In addition, the average evaluation metrics of RandomForest model rose to 0.993519, which becoming the best performanced model in this step. It is also concluded that our normalization method is effective.

4) Analysis from the performance of Feature-filter + normalization(by default parameters):

After the process of Feature-filter and normalization, all models except SVM are relatively performed better when compared with that on the original dataset, only-Feature-filter dataset and only-normalization dataset. It might because that SVM classifier relies on more features to predict effectively. Among the five models, the RandomForest performed the best, whose F1 score reached 0.995318.

5) Analysis from the models in best parameters:

The performance of all models with searching the best parameters is better than that of the default parameters,which proves the effectiveness of our parameter selection. Among them, SVM improved the most, RandomForest became the model with the highest F1 score, and its F1 score, accuracy, recall and precision were 0.995320, 0.995327, 0.995327 and 0.995373, respectively. In addition, lightgbm also reached 0.995317, 0.995327, 0.995327 and 0.995359, becoming the model with the highest precision.

To sum up, after normalization and feature selection, the performance of the models all have been improved, which proves the effectiveness of our methods. Moreover, the process of finding the best parameters of the model pushed the performance of the models to the highest. Among the five models, SVM performs poorly, RandomForest and Lightgbm perform best in our experiments.

B. Appropriate Metrics

First of all, f1_score is an evaluation criteria in the GroupProjectSpec.pdf. F1_score is the calculation result which considers the precision. In other words, it is the harmonic average of precision and recall. Therefore, it is appropriate to evaluate trained models. secondly, during discussion, we decide to calculate precision and recall as verification to observe the f1_score.

In addition, accuracy is also a significant factor to evaluate the performance of trained model. In multilabel classification, accuracy returns the subset accuracy. If the entire set of predicted labels for a sample strictly match with the true set of labels, then the subset accuracy is 1.0; otherwise it is 0.0. Therefore, it can help us to observe the right prediction proportion.

In a short, according to discussion, we assume that f1_score is more appropriate to be a leading indicator and accuracy can also be a good indicator to evaluate the performance of trained models.

C. Future Improvement

In our processing of outliers (including the outliers and null values in each column of the box chart), we used the maximum value, minimum value, average value, and mode of the corresponding column to replace the outliers. But the actual effect is not as well as the performance before processing. Since we have not been informed of the attributes of each column in the data, we guess that the outliers we think are actually some special true values, so we omit the processing of outliers in subsequent experiments. However, for the huge amount of data, the existence of outliers should be inevitable. We cannot accurately distinguish outliers from special truth values, or use other algorithms to replace these outliers. This part can be a direction in future research.

When judging the required features, we deleted the features that are not highly correlated with the label and are related to each other, but the criterion for judging when we select the threshold is the default parameter result of the RandomForest, because the initial data without any processing is used RandomForest has the best performance among the results obtained by training of different classifiers. This makes when using the data after the step of only deleting features to perform different classifier experiments, only the results of the RandomForest are improved. In view of the good performance of the RandomForest on this data set, even if the threshold is adjusted for each classifier, the result of the RandomForest may still be the best in the end, which has no effect on the submitted results, but we must pay attention to the importance of rigorous scientific research. We were not aware of this problem at the time and hoped that we could add it in the follow-up research and warn ourselves.

VI Conclusion

This report focusses on how to build a machine learning model for multi-classification based on unknown dataset and how to evaluate the performance of this model. In this project, we implement numerous methods- Outlier detection, Feature exploration and Normalization for data analysis and re-processing, SVM, Decision Tree, RandomForest, Xgboost and light gbm combined with Grid Search for model selection and parameters tuning. Comparing different metrics, we think that ‘f1 score’ is an appropriate one among ‘accuracy’, ‘recall’ and ‘precision’ for model evaluation. Based on this metrics, we ensure that RandomForest performs better. Although, we have built a good model, two aspects still need to be improved. One is outlier detection, the other is feature exploration, which can be our future research directions.

VII References

[1] Jiang, S. and Wang, L., 2016. Efficient feature selection based on correlation measure between continuous and discrete

features. Information Processing Letters, 116(2), pp.203-215.

[2] Akoglu, H., 2018. User's guide to correlation coefficients. Turkish Journal of Emergency Medicine, 18(3), pp.91-93.

[3] A. J. Smola and B. Schölkopf, A tutorial on support vector regression, vol:14, Stat Comput,2004, pp.199–222.