A comparative study on car evaluation dataset using different machine-learning methods

Paper Recurrence and Other Techniques

Shiyu Wang sw3601, Yingda Tao yt2767, Hao Zeng hz2759 and Yuqing Jin yj2679

Electrical Engineering
Columbia University
New York, USA
Email: {sw3601, yt2767, hz2759, yj2679}@columbia.edu

ABSTRACT

This is the EECS 6690 report of final project in Statistical Learning for Biological and Information Systems. The objective of this report, is to reproduce the results of a machine learning paper with R code, use different ML methods to further explore their performances, and make a comparison of their results and reasons behind them in a specific case. Car plays a vital role in everyday life. People would make an evaluation of a car based on different characteristics of the car. This study is based on the car evaluation dataset from UCI data repositories. In the original paper, it used Naïve Bayes and multi-layer perceptron methods to evaluate the acceptability for customers considered of six characteristics of a car. We reproduced their paper ending up with the same accuracy result; the accuracy is about 80% and 93%, for Naïve Bayes and multi-layer perceptron respectively. Additionally, we use SVM, Decision Tree and other methods to build classification model and achieve successful results; the highest average accuracy is 97.7% by KNN method.

Key Words: Car Evaluation Dataset, Multiple Layer Perceptron, Naïve Bayesian, Decision Tree, SVM

1. INTRODUCTION

Price, safety and space size are some of the first factors people should consider when considering buying a car. The price, space, safety performance, and number of doors of a car vary according to the type, driving performance, and manufacturer of it. These factors can be divided into two main aspects, extrinsic and potential factors. Price, space size, number of doors, number of seats can be classified as external factors, while safety, maintenance costs can be classified as potential factors. These two factors mainly determine the acceptability of a car. In addition, different types of drivers have different requirements for the car, and the application scenarios of the car are different. As a result, different characteristics may have different results for different types of audiences. From the predictive studies of the data set, we can draw a general conclusion: A car with what kind of features would be more popular among different types of drivers. This conclusion is considerably meaningful for automobile manufacturers. The result helps manufacturing industries and business companies learn the preference of their buyers, thereby increases the value of their goods and industries.

For such large data and classification analysis issues, it is necessary and helpful to be able to help forecast the outlook in the manufacturing, healthcare, business and education fields. SVM, Naive Bayes, multi-layer perceptron neural network and other methods, are able to help dig out the hidden rules behind the numerical data. Based on these results, we can analyze the reasons to some extent and make a prediction for the future direction of industries.

In this study, the original article used two methods, Nave Bayes and MLP, to classify and analyze the evaluation of vehicles. Mainly based on the accuracy and speed of prediction as a measure, to determine whether the model has a better prediction ability and faster prediction speed. These two standards are also critical in practical application.

This study is mainly constructed in the following order: the first section is the introduction part; then the second part is the literature review. Section 2 included the dataset description, the methods used in the original paper (Naïve Bayes and MLPNN) and their results. The third part of this paper is the experiment and results that we have done in data-mining of the car evaluation dataset. Section 3 incorporates data visualization, data preprocessing and different classification method experiments. Different classification method includes KNN, SVM, Logistic Regression, Decision Tree and Random Forest. Section 4 is the conclusion and future work of this study. Comparing different kinds of classification method, we draw a conclusion of which models better outperform than others and analyze the potential reasons behind the results.

2. LITERATURE REVIEW

2.1 Description of the dataset

All the machine learning and deep learning experiments is based on the car evaluation dataset. There is total 1728 instances in the data. In each instance, it has six predicting attributes and one determining attribute. The task of this dataset is to make classification (four levels of acceptability of a car) according to six predicting attributes (exterior and potential factors of each car). The car evaluation dataset is from UCI machine learning dataset repository and is denoted by Marco Bohance. Table 1 shows the dataset categorically.

Table1. Car Evaluation Dataset Description

Data Set Characteristics:	Multivariate	Number of Instances:	1728
Attribute Characteristics:	Categorical	Number of Attributes:	6
Associated Tasks:	Classification	Missing Values:	No

The predictors in this dataset are Price, Maintenance, Boot Space, Safety, Doors and Seats. Boot space, seats and safety are at 3 levels whereas price, maintenance, doors and acceptability are at 4 levels. There is only one response in this dataset, which is acceptability. This is at 4 levels.

The names of their specific levels are stated in the following:

- Price is divided into 4 levels, including low, med, high, vhigh
- Maintenance is divided into 4 levels, including low, med, high, vhigh

- Seats is divided into 3 levels, including two, four, more
- Doors is divided into 4 levels, including two, three, four, five & more
- Safety is divided into 3 levels, including low, med, high
- Boot Space is divided into 3 levels, including small, med, big.
- Acceptability is divided into 4 levels, including unacc, acc, good, vgood, where unacc-unacceptable, acc-acceptable, vhigh- very high, med-medium, and vgood-verygood.

2.2 Methodology of the original paper

First of all, data-cleaning is used for car evaluation data in the original paper. The objective to employ it is to remove the inconsistency in training data and convert nominal attributes in data into numerical value. Without data-cleaning, the accuracy and result may be highly affected.

In order to narrow data in an appropriate range, it used the min-max normalization method to generate data between range 0 to 1. This helps our models to enhance the understanding of the association between inputs and outputs. With the range limits, data are densely located in an adequate range, making the learning process easy.

For the dataset split, the paper used 90%-10%, 66%-34%, 50%-50% data for training and testing in each machine-learning method. Furthermore, it uses 10-fold cross validation in study to decrease the possibility of overfitting and selection bias. In 10-fold cross validation, we divided data into 10-fold (each has 10% data). Choose the first 10% for testing and the rest for training; next time, select the second 10% for testing and the rest for training. Repeat the procedure and the average of their accuracy is the accuracy for training.

For data-mining methods, this paper mainly compares and contrasts Naïve Bayesian and Multilayer Perceptron Artificial Neural Network in terms of accuracy and time.

2.3 Results of the original paper[1]

This study is mainly based on the accuracy and time cost of predictions as metrics, to determine whether these models have a better fitting performance and faster forecast speed. These two standards are quite critical in practical applications.

In their research, the accuracy of Naïve Bayes is about 79-84% and of multi-layer perceptron algorithms is about 92-95%. Multi-Layer Perceptron neural network models have a relatively better performance than Naïve Bayesian (NB) in every data-split method employed in the experiment. Even though MLPNN requires more time to build the model and set parameters, the time cost is worthy for its much higher accuracy output.

3. METHODOLOGY & EXPERIMENTS AND RESULTS

3.1 Data Preprocessing

3.1.1 Data Visualization

With the help of graphical means, data visualization can clearly and effectively convey and communicate information, making the data vivid. For the car evaluation data set, there is a large amount of information, and data visualization is a good way of data expression. Figure1 is the heatmap of car evaluation data set. It is the clustered heatmap of six attributes and shows how dispersed each attribute is. Most of them are evenly distributed. and it shows that we have different levels of attributes in different case. Figure2 can clearly express the proportion of each class in our data set, from which we can see that class “unacceptable” occupies most while class “very good” and class “good” have the least proportion. Figure3 is the relationship between features. The values between different features in this figure are obviously small, this indicates the independence of features in our data set. Figure4 shows the density distribution of different classes.

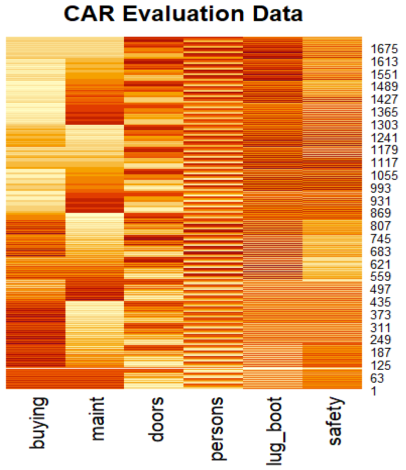


Figure1. Heatmap of car evaluation data set

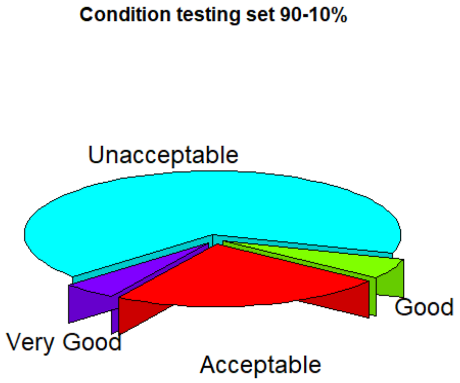


Figure2. Distribution of 4 classes

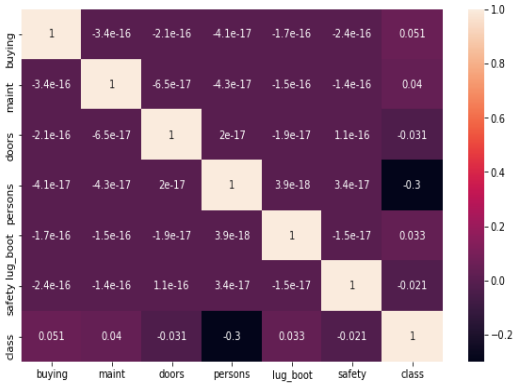


Figure3. Correlation between different features

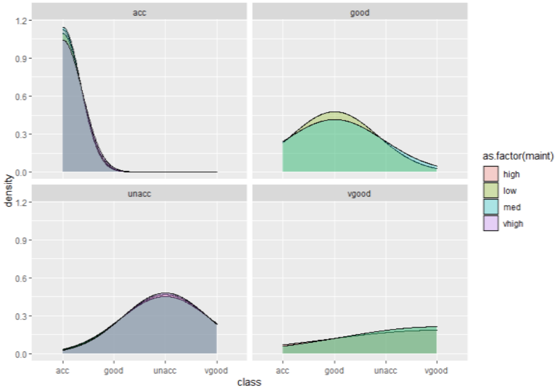


Figure4. Density distribution of 4 classes

3.1.2 Data Cleaning

Dataset we deal with has qualitative features instead of quantitative features. So, it is necessary to transform the input data to numbers. This transformation is a guarantee for the further data processing. Our transformation detail is in the following table.

Table2. Qualitative Features to Quantitative Features Conversion

Feature Name	Qualitative Features	Quantitative Features
buying	low	1
	med	2
	high	3

	vhigh	4
maint	low	1
	med	2
	high	3
	vhigh	4
doors	NA	2
	NA	3
	NA	4
	5more	5
persons	NA	2
	NA	4
	more	5
lug_boot	small	1
	med	2
	big	3
safety	low	1
	med	2
	high	3

NA in the table means the feature attribute itself in this place is the numeric feature, its value is in its right quantitative box.

3.1.3 Data Transformation

After converting data from qualitative attribute to numeric attribute, we implement data standardization, centralization and normalization in the data transformation process. We firstly scale and center data, then execute data normalization using min-max technique. In the original paper, it only pre-process data by normalizing. Our experiment result shows that implementing data standardization and centralization before data normalization can improve about 1% accuracy.

3.1.4 Data Split

It is essential to split data into training data and test data. Training data is used for building the model and test data is used for test the model. Splitting the dataset into 2 parts guarantees the “pure” model. In our project, we split data in 4 ways.

Table3. Dataset Split

Data split (training data-test data)
90%-10%
66%-34%
50%-50%
10-fold cross validation

3.2 Classification Methods and Experiment Results

3.2.1 MLPNN

Artificial neural networks (ANN) are inspired by the early models of sensory processing by the brain. It can simulate the structure, implementation mechanism and functions from human brain neural networks. By employing the algorithm in order to imitate the process how real neurons work computationally, our computers are able to learn and solve different types of tasks. [2]

In our project, as is shown in Figure 5, we have six units at input layer as predictors and four units at output layers as labels representing four classes of acceptability. In hidden layers, three layers were constructed, with 30, 60, 30 units respectively. Each layer employs Rectified Linear Unit, ReLU, as the activation function, followed by a dropout layer with a frequency of rate=0.4 to prevent overfitting.

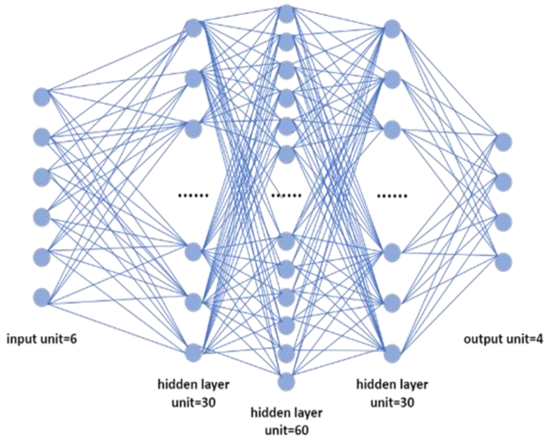


Figure5. The structure of ANN

Dataset was split into training data and test data with percentages 90-10, 66-34, 50-50, respectively. 10-folds cross validation is also employed in the experiment. The neural network was trained by 100 epochs with batch size=5. Thus, we have several different models generated by different training data and testing data.

In the original paper, the result achieved by the model is 94.79%, 93.19%, 92.7% and 94.09% with respect to data split 90%-10%, 66%-34%, 50%-50%, and 10-fold cross-validation, which offered good performance. However, the original paper did not specify the detailed structure of the multilayer neural network model. Thus, we constructed a new neural network to achieve higher accuracy. The accuracy and running time achieved by MLP-ANN with different data split is shown in Figure 6(a) and Figure 6(b) respectively.

From Figure 6 we could observe that under the same model structure, 10-fold cross-validation data split achieved the highest accuracy with 98.38%. Compared to other train/test split, 10-fold cross-validation use input data more sufficiently since every observation is used for both training testing where every data points get to be tested exactly once and is used in training 9 times. It also has less bias toward overestimating the true expected error but higher variance due to bias-variance tradeoff. However, by comparing running time, we could observe that 10-fold cross validation requires approximately ten times higher running time than train/test split because the training process must be rerun for 10 times while for each time it has to be run for 100 epochs.

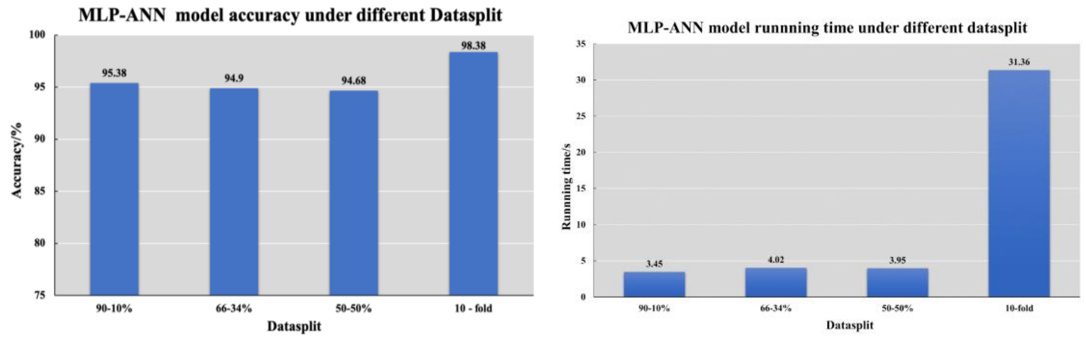


Figure6. (a)Testing accuracy with different data split (left figure); (b)Running time under different data split (right figure)

In our model, as is shown in the chart below, the result achieved by the model is 95.38%, 94.90%, 94.68% and 98.38% with respect to data split 90%-10%, 66%-34%, 50%-50% and 10-fold cross validation. Our model outperforms the original one by adding more layers in hidden layer and tuning parameters according to dataset. In the experiment, we found that the number of epochs and batch size both have a great influence on the performance of the model. Testing accuracy with regards to different number of epochs and batch size is also shown as follows.

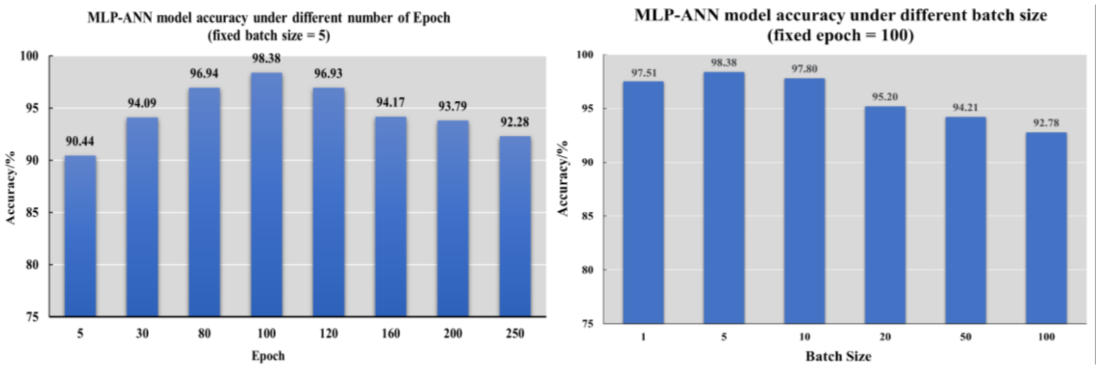
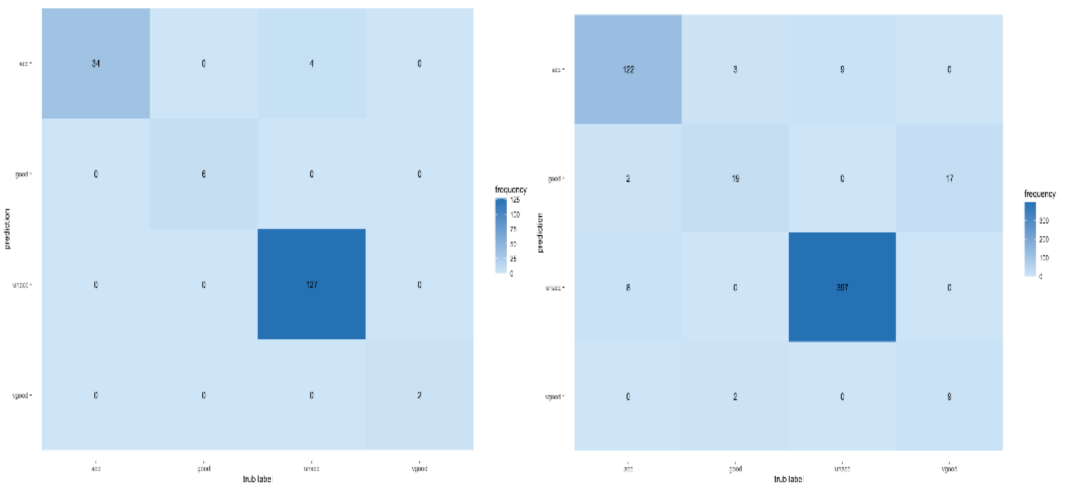


Figure7. (a) Accuracy under different number of epochs (left figure); (b) Accuracy under different batch size (right figure)



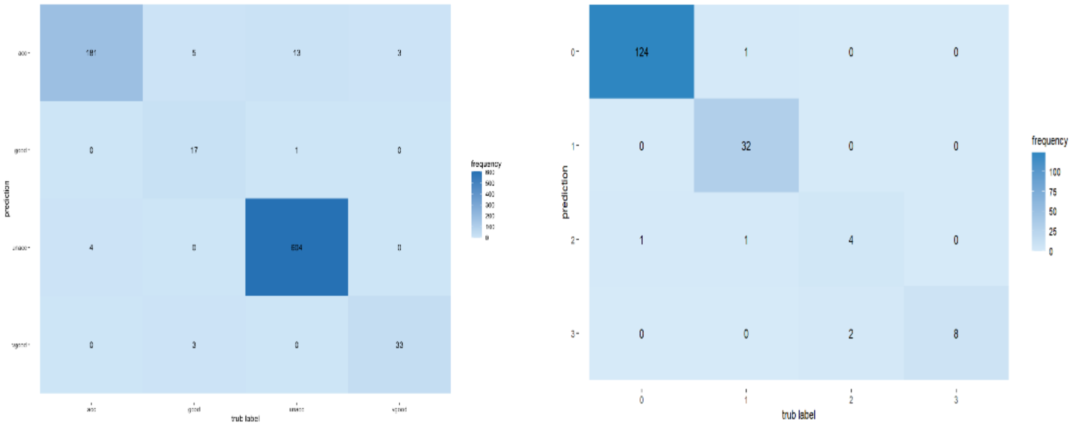


Figure8. Confusion Matrix using MLPNN method for 90%-10% data split, 66%-34% data split, 50%-50% data split and 10-fold cross validation respectively.

The comparative study shows that the accuracy is higher with an increasing number of epochs when batch size is fixed, which could be explained by the fact that the model is trained more sufficiently with more epochs. However, when the epoch reached 120, a decrease in testing accuracy can be observed. The phenomenon is caused by overfitting where the model is trained for too long on training data and fits too closely to training set learning noise. The model cannot generalize the testing data well.

When the number of epochs is fixed to 100, the accuracy dropped when increasing batch size. Since the number of steps is defined by the number of batches, which is associated with batch size, an increase in batch size resulted in a decrease in steps during each epoch. The model trained by fewer steps can result in underfitting, which caused lower accuracy. Same logic can be applied to the case when batch size=1, the model was trained by more steps, which caused overfitting and therefore resulted in a lower accuracy.

3.2.2 Naïve Bayes

The Naïve Bayes classifier is based on Bayes classifier which is based on Gaussian distribution with additional assumption which is the individual features are independent with each other given the class label. [3]

One advantage of Naïve Bayes classifier is that it is a simple model which can be computed efficiently. However, it can reach good results only when the dataset is in Gaussian distribution or features of the dataset are independent with each other.

In our project, we use the Naive Bayes function in e1071 library. There is a parameter called Laplace in this function whose role is to control Laplace smoothing to avoid zero probability problems. We tune this parameter and find it has no influence on the prediction of our dataset. This phenomenon corresponds to one of the characteristics of our dataset, that is, all 4 labels exist in the training dataset.

In original paper, for 90%-10% data split, 66%-34% data split and 50%-50% data split, the prediction accuracy is 79.76%, 83.84% and 83.68% respectively, while our prediction accuracy is 89%, 83.1% and 82.8% respectively. As for 10-fold cross validation, the accuracy of the original paper is 82.34% while our corresponding result is 79%. Our result is similar to the original paper. As we can see, Naïve Bayes performs not very well. From data visualization, we know that features of car evaluation data set are independent with each other. The only reason accounting for this bad result is that our dataset doesn't conform with Gaussian distribution.

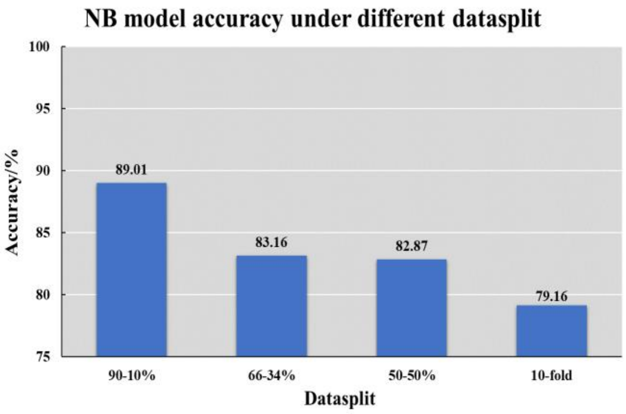
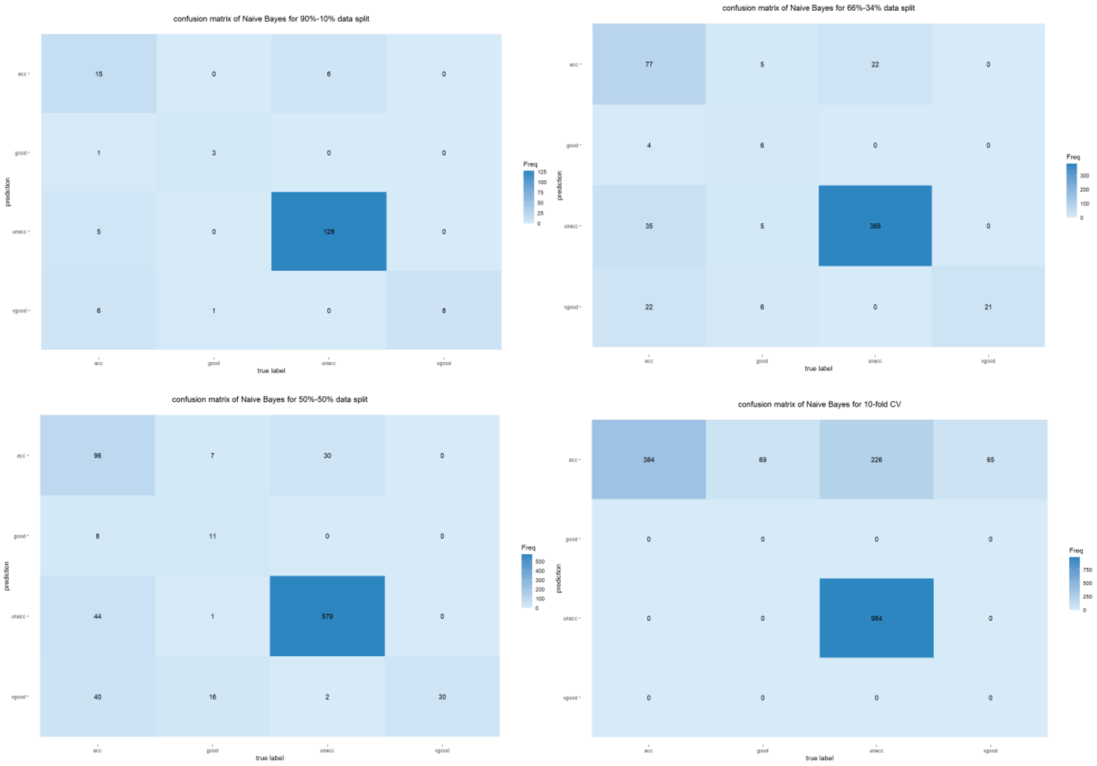


Figure9. Accuracy using Naïve Bayes method for different data split



confusion matrix of Naive Bayes for 90%-10% data split

	acc	good	unacc	vgood
acc	15	0	6	0
good	1	3	0	0
unacc	5	0	128	0
vgood	6	1	0	8

confusion matrix of Naive Bayes for 66%-34% data split

	acc	good	unacc	vgood
acc	77	5	22	0
good	4	6	0	0
unacc	35	5	385	0
vgood	22	6	0	21

confusion matrix of Naive Bayes for 50%-50% data split

	acc	good	unacc	vgood
acc	96	7	30	0
good	8	11	0	0
unacc	44	1	579	0
vgood	40	16	2	30

confusion matrix of Naive Bayes for 10-fold CV

	acc	good	unacc	vgood
acc	384	69	226	65
good	0	0	0	0
unacc	0	0	984	0
vgood	0	0	0	0

Figure10. Confusion Matrix using Naïve Bayes method for 90%-10% data split, 66%-34% data split, 50%-50% data split and 10-fold cross validation respectively.

3.2.3 Support Vector Machine

Support Vector Machine (SVM) is a kind of supervised learning methods. It can be classified into linear SVM and non-linear SVM. Linear SVM means its boundary is a hyperplane with the maximum margin[4]. And non-linear SVM means the input datasets don't need to be linear, but they can be transformed into high dimension space by feature mapping where they can be linearly separated by a hyperplane with the maximum margin as well. SVM can be used for solving both binary classification problems and multi-classification problems. In our project, we use SVM to resolve a 4-classification problem. Although mapping into high dimension space will lead to the increase of computational cost, we can use kernel tricks to accelerate computing progress implicitly.

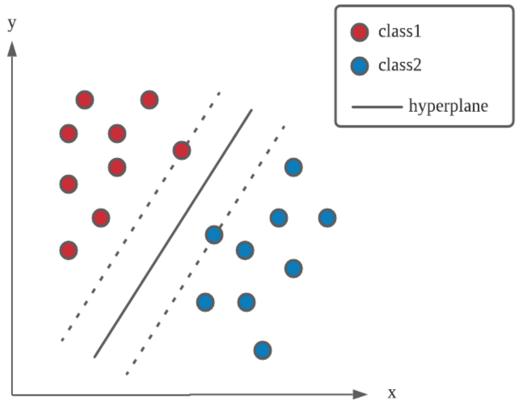


Figure11. SVM hyperplane with maximum margin

As said before, SVM can help increase computing speed, this is because it uses kernel tricks to conduct inner product implicitly in high dimensional space. In our project, we tried linear kernel, polynomial kernel, radial kernel and sigmoid kernel. Generally speaking, their performance ranking is radial kernel > polynomial kernel > linear kernel > sigmoid kernel. It is reasonable that the radial kernel performs best, since radial kernel maps low dimensional data into infinite high dimension and SVM can generate a hyperplane in high dimensional space to linearly separate data. The prediction result is as the following Figure, from which we can see that the radial kernel has the best performance.

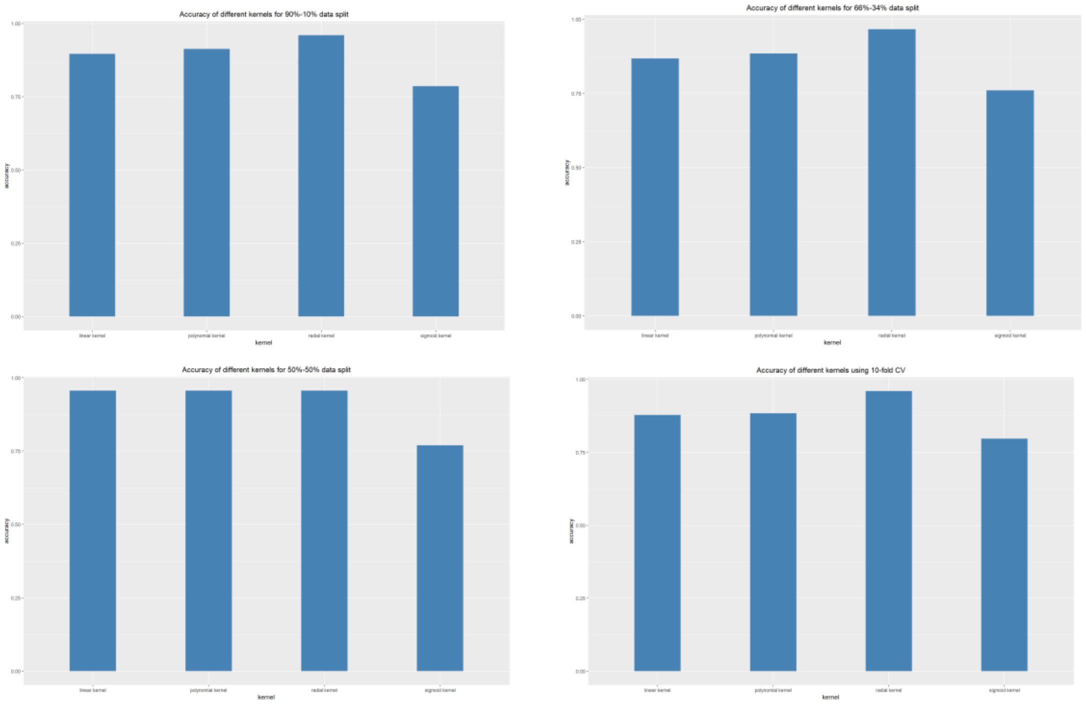


Figure12. Accuracy of different data split using different kernels

For hold-out validation, the result of using radial kernels is 95.9%, 96.5% and 95.6% respectively for data split 90%-10%, 66%-34% and 50%-50%, which is very high. For 10-fold cross validation, the result of using the radial kernel is 95.9%. As the result shows, SVM has a pretty good performance on our dataset.

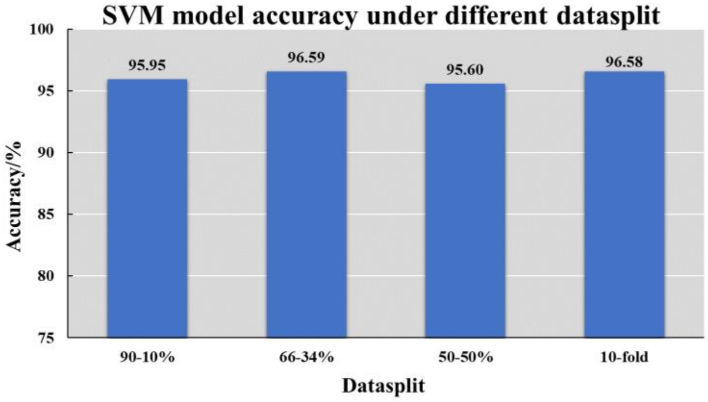
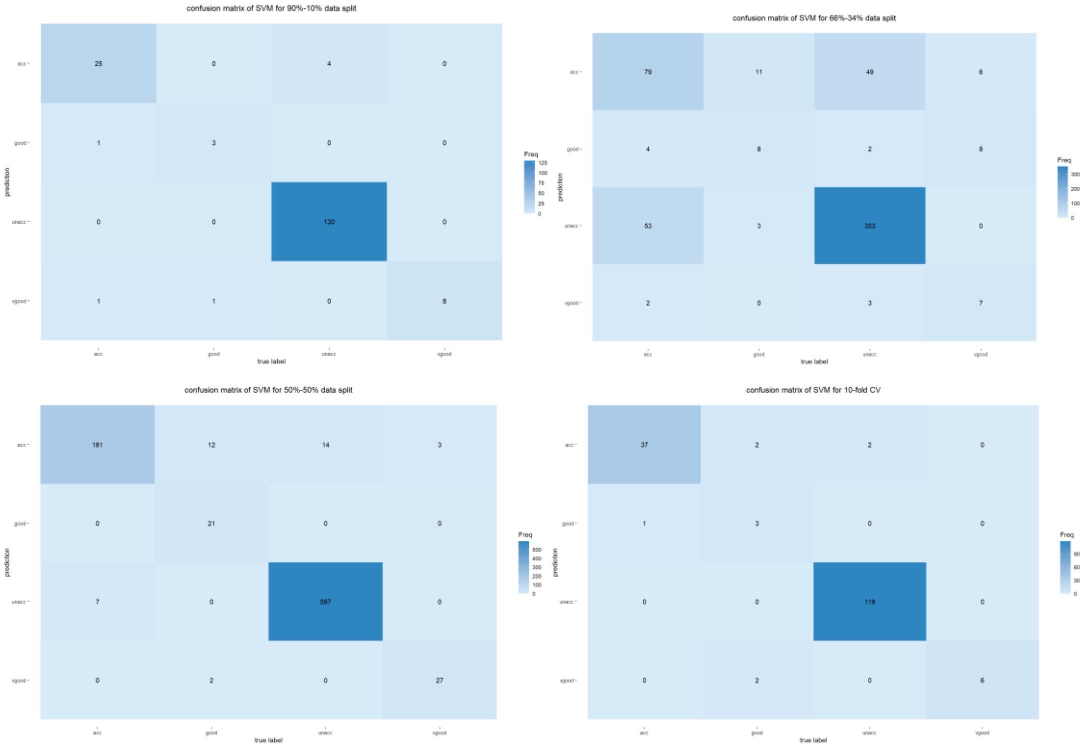


Figure13. Accuracy using SVM method for different data split



confusion matrix of SVM for 90%-10% data split

	acc	good	unacc	vgood
acc	25	0	4	0
good	1	3	0	0
unacc	0	0	130	0
vgood	1	1	0	8

confusion matrix of SVM for 66%-34% data split

	acc	good	unacc	vgood
acc	79	11	49	6
good	4	8	2	8
unacc	53	3	353	0
vgood	2	0	3	7

confusion matrix of SVM for 50%-50% data split

	acc	good	unacc	vgood
acc	181	12	14	3
good	0	21	0	0
unacc	7	0	597	0
vgood	0	2	0	27

confusion matrix of SVM for 10-fold CV

	acc	good	unacc	vgood
acc	37	2	2	0
good	1	3	0	0
unacc	0	0	119	0
vgood	0	2	0	6

Figure14. Confusion Matrix using SVM method for 90%-10% data split, 66%-34% data split, 50%-50% data split and 10-fold cross validation respectively.

3.2.4 KNN

K Nearest Neighbor (KNN) method is a kind of non-parametric classification, it is the simplest algorithm in machine learning. The principle of KNN is to find the k nearest training data of a given test data and then assign this test data to the class which majority of its k nearest training data belong to[5]. We use the relative distance to determine the k nearest training data of a test data point. The following figure shows an example of 3-NN for 2 classification problems.

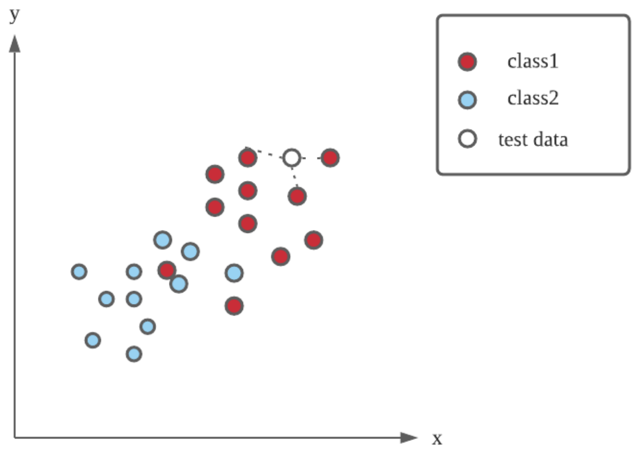


Figure15. An example of resolving binary classification problem using 3-NN method

How to determine the parameter K is very important in the KNN algorithm. If K is small, then our model will include more noises and then overfit, this means that test data will be very sensitive to its close data. If K is large, the accuracy will decrease and the computational complexity will be large, which will make the time of prediction very long. KNN is a good alternative to generative approaches like the naïve Bayes classifier, since it doesn't require us to pick a proper probability model. However, this also means using this kind of approach leads us to no understanding of the population.

We have mentioned that tuning the k parameter of KNN method is very important. In our project, we have tried several k values and finally found it achieves the highest accuracy when k is equal to 5. And we notice that when k is very big, the accuracy will decrease a lot. For example, when k increases to 200, the accuracy will decrease to around 80%.

Using KNN methods with k=5, the result is shown in the following figure. In detail, the accuracy can reach 98.2%, 99.3%, 94.9% and 98.8% respectively under the condition of 90%-10% data split, 66%-34% data split, 50%-50% data split and 10-fold cross validation. From the result, we can know that the performance of KNN for classification of this dataset is pretty good.

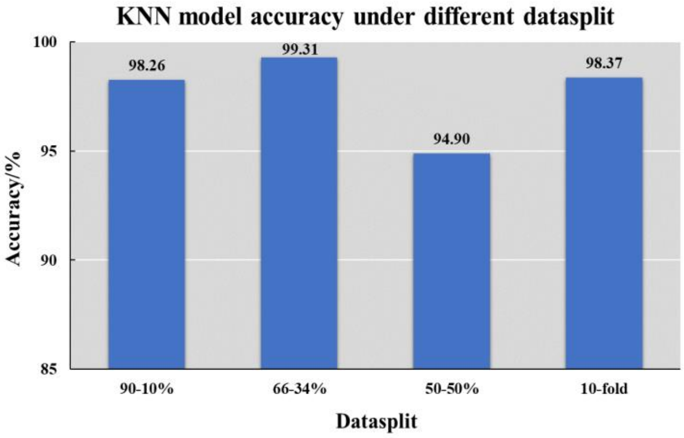


Figure16. Accuracy using KNN method for different data split

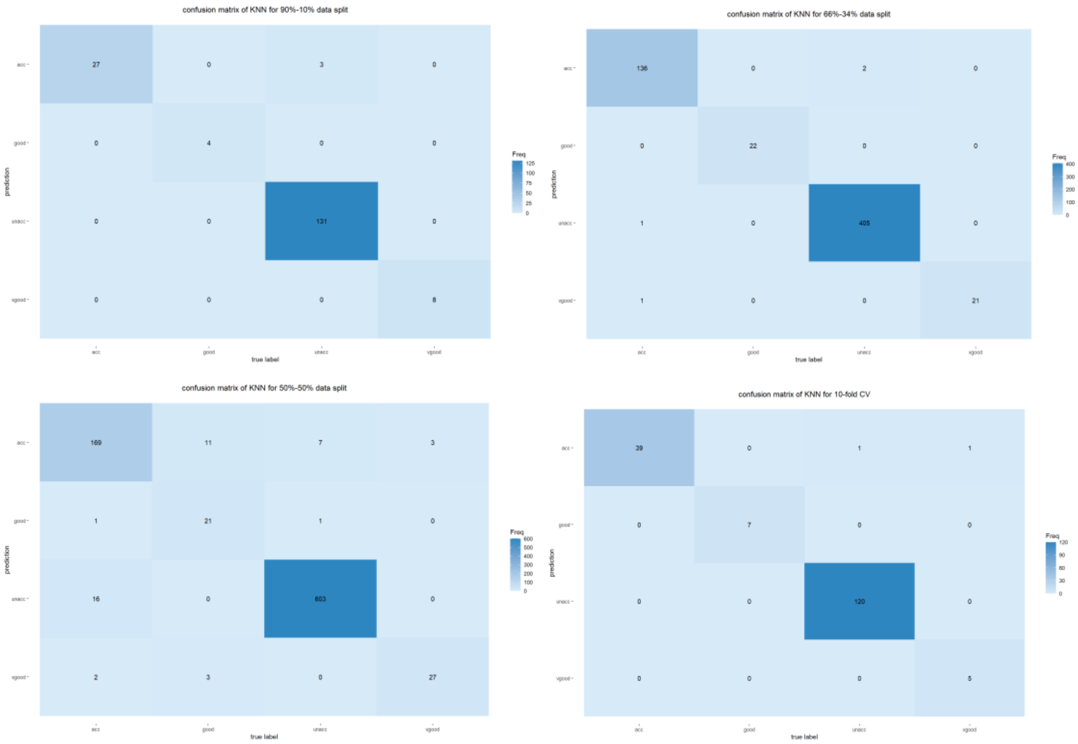


Figure17. Confusion Matrix using KNN method for 90%-10% data split, 66%-34% data split, 50%-50% data split and 10-fold cross validation respectively.

3.2.5 Logistic Regression

Logistic regression is a generalized linear model. Although it seems like a regression model according to its name, it is actually a classification model. Besides, logistic regression is a linear classifier, since the decision boundary that it creates is a hyperplane. It can not only be used for binary classification problem, but also can be utilized for multi-classification problem[6].

Logistic regression is a way to approximate the conditional probability $P(Y = j|X = x)$, which is probability of assigning an observation x to a class j. Our goal is to find a class j that can maximize this conditional probability. Linear regression models this conditional probability by formula $p(X) = \frac{e^{w \cdot X}}{1+e^{w \cdot X}}$, which limits the output value to [0,1]. To find the parameter w. We usually use the Maximum Likelihood estimation (MLE) method.

Logistic Regression method is usually used for solving binary classification problems. But we can still use multinomial logistic regression to do multi-classification. The result we get is as follows: For 90%-10% data split, the accuracy is 88.4%; For 66%-34% data split, the accuracy is 85.2%; For 50%-50% data split, the accuracy reaches 86.3%. When using the 10-fold cross validation method, the accuracy is 85.5%. As we can see, the classification result using logistic regression doesn't satisfy us, this is because logistic regression is a method mainly for binary classification problems. It is not appropriate for multi-classification problems.

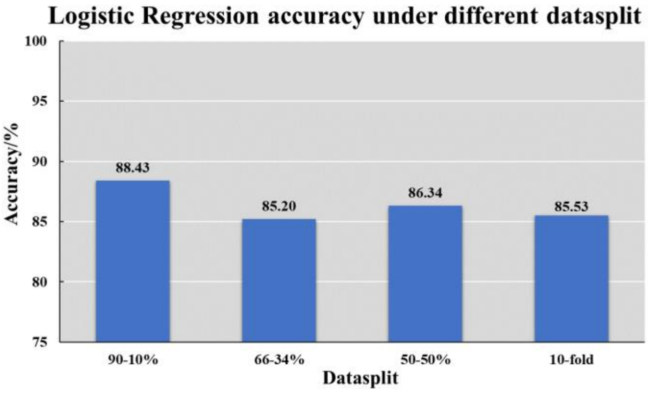


Figure18. Accuracy using Logistic Regression method for different data split

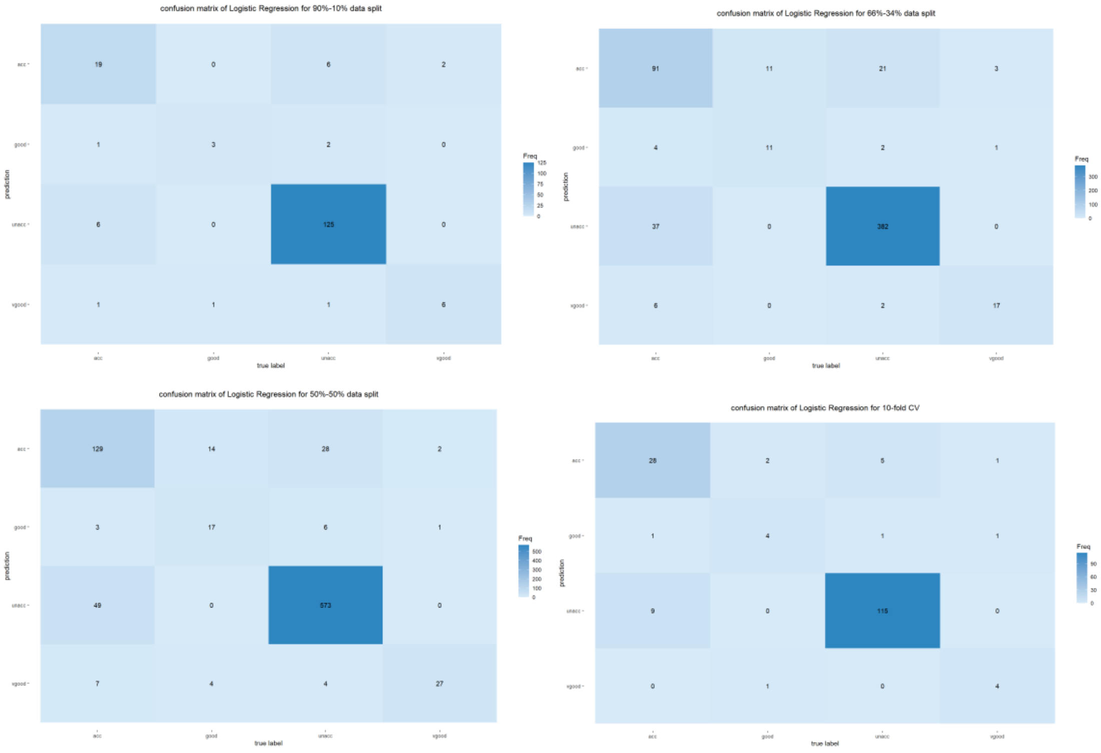


Figure19. Confusion Matrix using Logistic Regression method for 90%-10% data split, 66%-34% data split, 50%-50% data split and 10-fold cross validation respectively.

3.2.6 Decision Tree

Decision tree is a simple machine learning algorithm with tree structure. The decision tree passes through each branch of the tree structure (decision point) to classify data, and then enter the tree structure of the next layer until the end of the decision point is reached.[7]

It can be used for comparison with parametric methods such as naive Bayes. In this project, for creating a decision tree, we first create a root node, and then calculate the entropy of each attribute. After that, select an attribute that best classifies the train set and has the highest information gain, and bind the branch to the root node. Finally, we repeat this process until the entropy returns to zero or the information gain is zero or one.

Decision tree classification was used in this project as other Machine learning algorithms. It is used in this case study because it is a nonparametric method. The accuracy of the decision tree for 4 different data splitting are as shown in Figure20. It is clear that the accuracy of the decision tree is better than

Naive Bayesian and MLPNN from the original paper. In the decision tree, we use the CART algorithm, that is, the classification regression tree algorithm. Cart algorithm uses Gini index instead of information entropy and binary tree as the model structure, so it is not directly divided by attribute value. The algorithm should find the best binary division among all attributes. Cart algorithm continuously divides decision attributes through recursive operation, and optimizes the tree model by using validation data. The higher accuracy of the decision tree may be because the decision tree does not need preprocessing, does not need to be normalized in advance, and can deal with missing values. It can deal with both discrete values and continuous values. Many algorithms only focus on discrete or continuous values. At the same time, it can deal with the classification of multi-dimensional output, and has good fault tolerance and high robustness for outliers. The model can be selected by cross validation pruning, so as to improve the generalization ability.

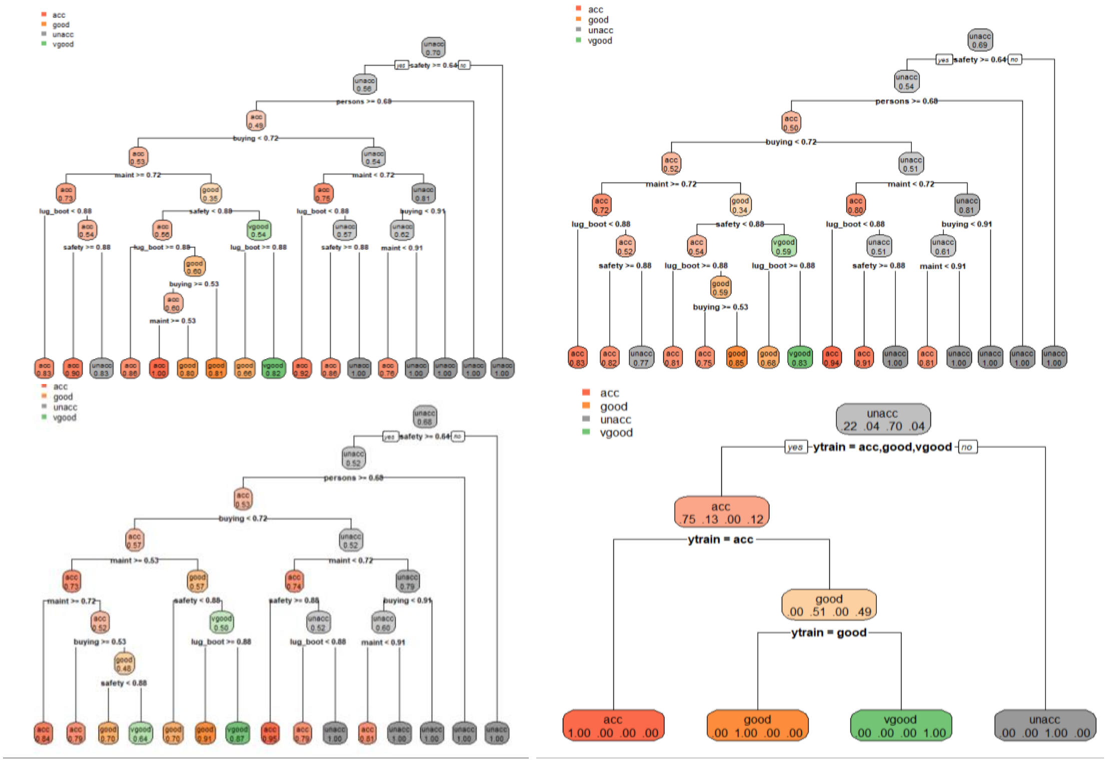


Figure20. visualization of Decision Tree for different data split

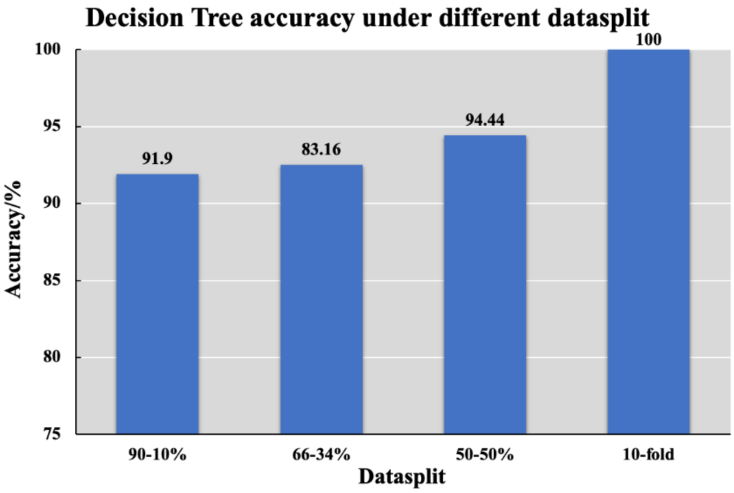


Figure21. Accuracy using Decision Tree method for different data split

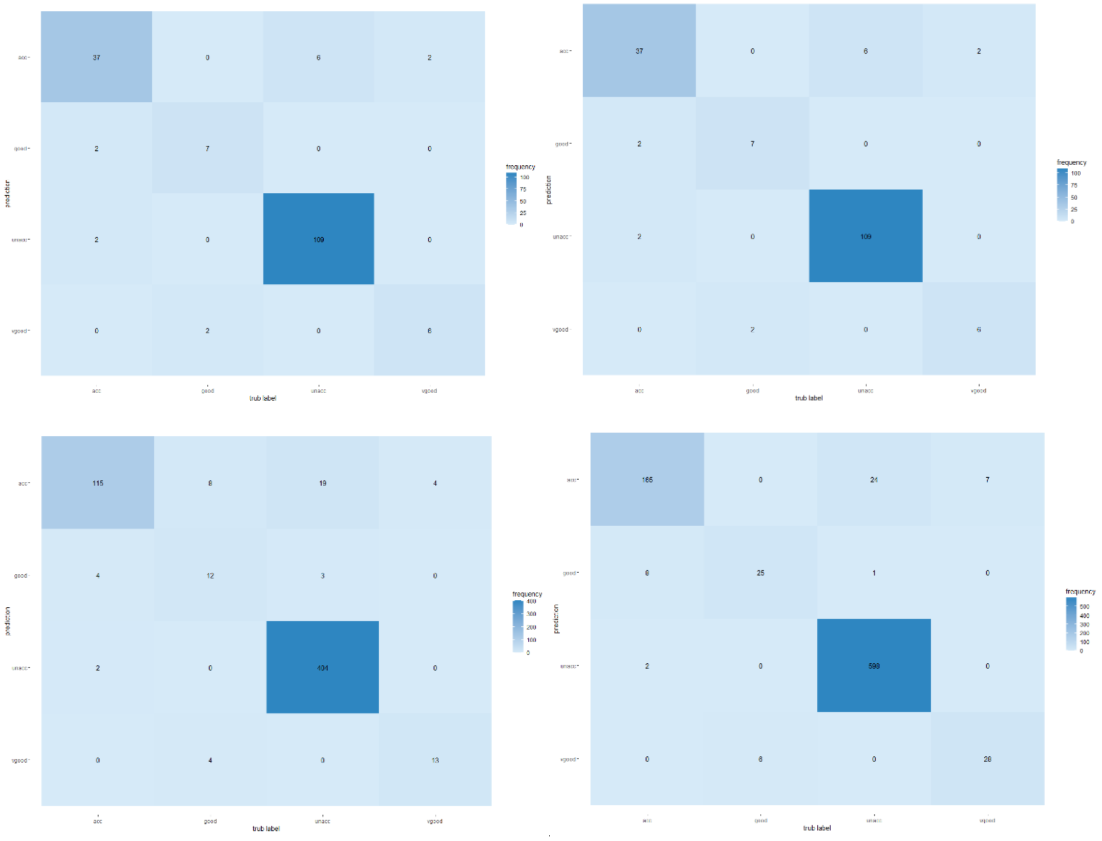


Figure22. Confusion Matrix using Decision Tree method for 90%-10% data split, 66%-34% data split, 50%-50% data split and 10-fold cross validation respectively.

3.2.7 Random Forest

The calculation of random forest is established through the principle of decision tree. Take the decision tree as a random forest containing a large decision tree. Different from the single result of the decision tree, a random forest can summarize the results of each decision tree to determine the final output result. In principle, a random forest has higher accuracy for data processing than a decision tree[8].

In this project, we establish a random forest integration model based on the decision tree to compare and improve the accuracy of the model. We construct a simple random forest classifier model containing 500 trees. The accuracy for random forest method is shown in Figure20, where the accuracy is 97.1%, 97.6%, 95.13% and 100% for data splitting 90%-10%, 66%-34%, 50%-50% and 10-fold cross validation. The accuracy is improved compared with the decision tree method. From the figure, we know that the accuracy of random forest is higher than that of decision tree in this project. Because the random forest contains a large number of decision trees, the random forest can correct the inaccurate prediction of data results caused by the influence of outliers of individual decision tree. But at the same time, because each decision tree contains its own unique feature samples, large-scale decision trees avoid the possibility of fitting the features of decision tree samples to each other.

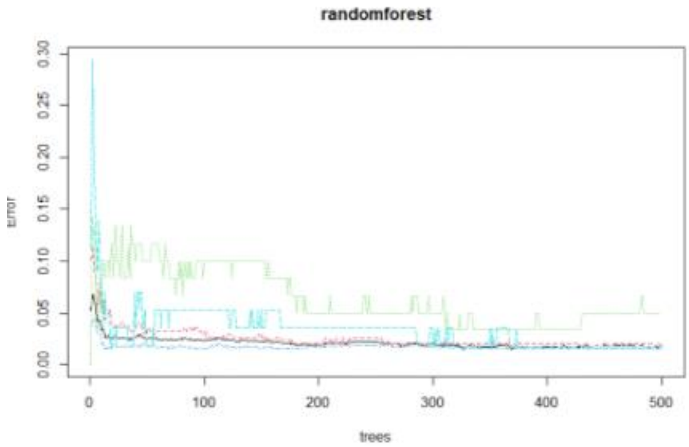


Figure23. Random Forest 500 trees error

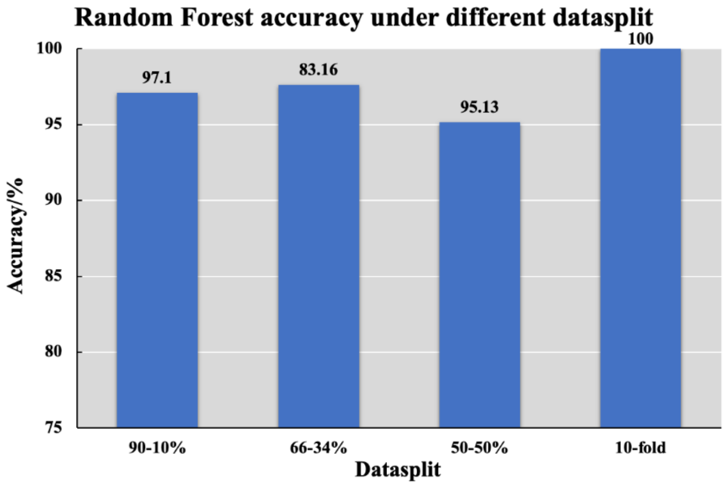
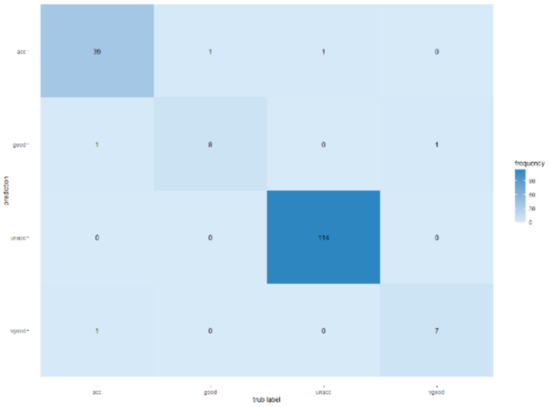
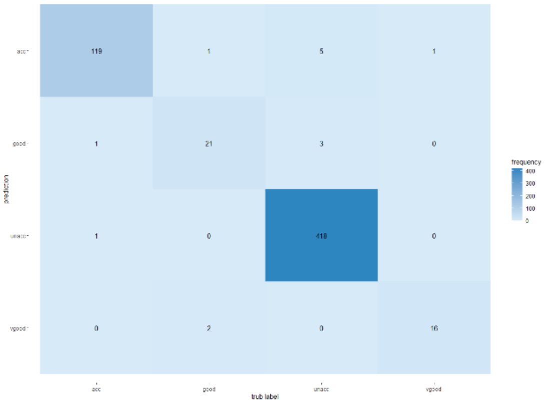


Figure24. Accuracy using Random Forest method for different data split





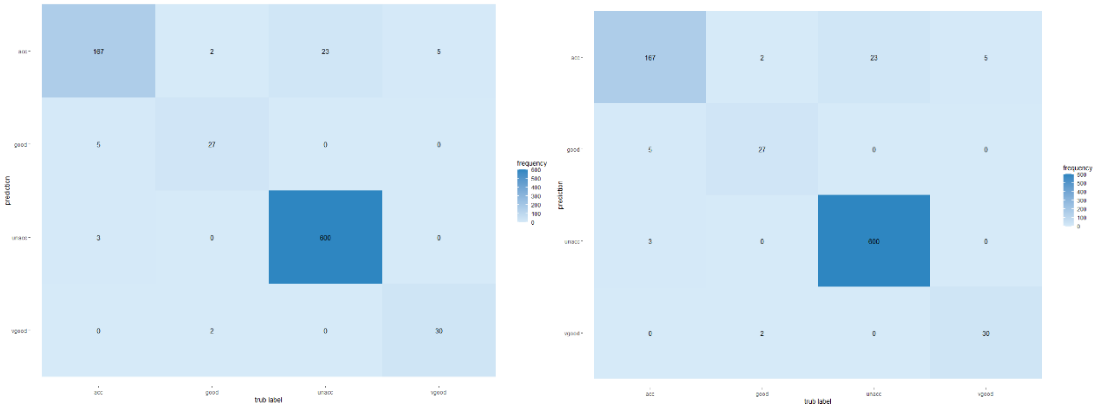


Figure25. Confusion Matrix using Random Forest method for 90%-10% data split, 66%-34% data split, 50%-50% data split and 10-fold cross validation respectively.

4. CONCLUSION&FUTURE WORK

4.1 conclusion

The fundamental task of this project is to compare the result of the classification models employing different methods: Naïve Bayes, multi-layer perceptron, SVM, Decision Tree and Random Forest in addition to reproducing the result of Naïve Bayes, multi-layer perceptron method in the original paper. The result of each method is demonstrated in the following chart.

Table 4. Performance of each method

Method	Average Accuracy	Average Running Time/s
MLPNN	95.84%	10.69
Naïve Bayes	83.55%	0.59
SVM	95.78%	0.05
KNN	97.71%	0.02
Logistic Regression	86.35%	0.25
Decision Tree	94.71%	0.38
Random Forest	97.46%	0.17

From the table, we could see all classifiers have relatively high accuracy above 94% except for Naïve Bayes and Logistic Regression. KNN and Random Forest performed the best, followed by SVM, multi-layer perceptron, and Decision Tree, while Logistic Regression and Naïve Bayes both gave accuracy below 90%. Compared to other methods, KNN is an easy and simple machine learning model that has fewer hyperparameters to tune. It has the highest accuracy and the shortest running time. It outperforms SVM when there are larger training data and lesser features.

Random Forest is a collection of Decision Tree, and the predicted output is selected by an average vote. Thus, it is less prone to overfitting and more robust than Decision Tree, which gave a more generalized solution, resulting in higher accuracy. Decision Tree, on the other hand, gives more understandable explanations over the prediction result compared to other classifiers and outperforms other methods when there is categorical value in training data. Both SVM and MLPNN are powerful classifiers and they both gave high accuracy results in our project. SVM can outperform MLPNN when there is a limited training dataset while the latter would give a better result when there is sufficient training data. However, MLPNN requires the longest running time.

Logistic Regression and Naïve Bayes, in our experiment, performed the second and last lowest accuracy respectively. Logistic Regression can only handle linear classification problems and it requires properly selected features. Naïve Bayes assumes that all features are independent while other classifiers work on dependent features.

Furthermore, we could conclude that KNN performed the best whether in terms of accuracy and efficiency. Logistic Regression and Naïve Bayes failed to achieve accuracy above 90%. Other method performed a satisfying accuracy and MLPNN has the longest running time.

4.2 Future Work

In this project, we did a comparative study on different methods for classification in terms of accuracy and running time under various data split. However, some classifiers did not achieve satisfying results due to the features of the dataset. In future work, we could preprocess the data more suitable for fitting the model. We could also try other methods like LOOCV, Boosting and Adaboost[9].

REFERENCE

[1] Rehman, Z. U., Fayyaz, H., Shah, A. A., Aslam, N., Hanif, M., & Abbas, S. (2018). Performance evaluation of MLPNN and NB: a comparative study on Car Evaluation Dataset. International Journal of Computer Science and Network Security, 18(9), 144-147.

[2] Awwalu, J., Ghazvini, A., & Bakar, A. A. (2014). Performance Comparison of Data Mining Algorithms: A Case Study on Car Evaluation Dataset. International Journal of Computer Trends and Technology (IJCTT), 13(2).

[3] Rish, I. (2001, August). An empirical study of the naive Bayes classifier. In IJCAI 2001 workshop on empirical methods in artificial intelligence (Vol. 3, No. 22, pp. 41-46).

[4] Hearst, M. A., Dumais, S. T., Osuna, E., Platt, J., & Scholkopf, B. (1998). Support vector machines. IEEE Intelligent Systems and their applications, 13(4), 18-28.

[5] Guo, G., Wang, H., Bell, D., Bi, Y., & Greer, K. (2003, November). KNN model-based approach in classification. In OTM Confederated International Conferences" On the Move to Meaningful Internet Systems" (pp. 986-996). Springer, Berlin, Heidelberg.

[6] de Souza, R. M., Cysneiros, F. J. A., Queiroz, D. C., & Roberta, A. D. A. (2008, October). A multi-class logistic regression model for interval data. In 2008 IEEE International Conference on Systems, Man and Cybernetics (pp. 1253-1258). IEEE.

[7] Gupta, P. (2017, November 12). Decision trees in machine learning. Medium. Retrieved December 19, 2021

[8] Maklin, C. (2019, July 30). Random Forest in R. Medium. Retrieved December 19, 2021

[9] Desarda, A. (2019, January 17). Understanding adaboost. Medium. Retrieved December 19, 2021