Machine Learning in WhiteWine Quality Prediction

Bohan Wu, and Victor Liu

***Abstract*—**We explore several current major machine learning models and use them to predict people's wine preferences on a large dataset that consists of *vinho verde* wine samples. We see this human wine preferences prediction as both regression and classification problem. Four regression techniques and four classification techniques were used to optimize prediction results. The Neural Network showed the best performance among all the methods considered. With high stability and prediction accuracy, Neural Network is helpful in acting as a guide for wine production and in providing accurate prediction of people's tasting preferences in similar domains with even larger dataset.

Keywords—Multiple Linear Regression, Support vector machine, Logistic Regression, k-nearest neighbor, Neural Network

1. Introduction

Our focus is to predict human wine preferences based on physicochemical data included in the Wine Quality dataset from UCI machine learning repository. The methods that we applied on the dataset in the order of their overall complexities include multivariate regression, k-nearest neighbor, logistic regression, support vector machine, and neural network. All of the methods we considered are listed in Machine Learning Methods section.

2. About Data and Overall Approach

The Wine Quality dataset includes 11 predictor variables and 1 target variable, which are shown in the figure 1 below. Firstly, we chose to predict the target ("quality") as a regression problem. And the error margin is set to 0.5 by rounding the predicted values into the nearest integers. Multi-linear regression was applied. And later we investigated the necessity of adding regularization to the model with the hope of reducing loss and improving accuracy by using Lasso and Ridge. The built-in Elastic-Net regression that comes with default 5-fold cross validation was applied. But the focus was not put on how k-fold cross validation impact prediction results. We found that the overall accuracy is not significantly improved from the baseline with linear regression models.

Classification provides us a way to tackle the problem without the need of rounding numbers. With a deeper understanding of the data, we relabeled the target variable into three new classes. More balanced class labels in term of the number of times they appear in the dataset can help improve the prediction accuracy and increase overall prediction stability. Model optimization is explored with the Logistic regression and Support vector machine by looping through lists of hyperparameters or using scikit-learn's built-in hyperparameter tuning function GridSearchCV.

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.0	0.27	0.36	20.7	0.045	45.0	170.0	1.0010	3.00	0.45	8.8	6
1	6.3	0.30	0.34	1.6	0.049	14.0	132.0	0.9940	3.30	0.49	9.5	6
2	8.1	0.28	0.40	6.9	0.050	30.0	97.0	0.9951	3.26	0.44	10.1	6
3	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	9.9	6
4	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	9.9	6
5	8.1	0.28	0.40	6.9	0.050	30.0	97.0	0.9951	3.26	0.44	10.1	6

Figure 1. First 6 lines of the dataset

Machine Learning Methods

A. Baseline: Most common label

We visualized the distribution of the class labels with a plot in figure 2 that shows around 44% of the class labels (about 2200 in counts) belongs to class 6. It is easily obtained and a decent baseline in this case.

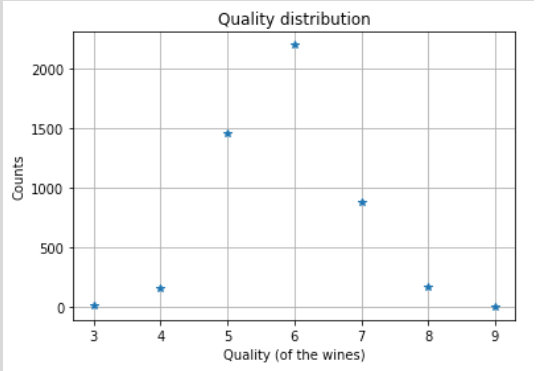

Figure 2. Class label distribution

B. Multi-linear regression

Multiple linear regression, or multiple regression, is widely used to explore the linear relationship between independent variable(s) and dependent variable, allowing people to predict unknown outcomes based on known inputs. Multi-linear regression model is easy to understand and implement, it is also the first method that we want to explore on the dataset. We implemented the multi-linear regression method by defining our own multi-linear function with the square loss, which is the function that we learned in class.

The dataset is randomly separated into training dataset, which includes some portion of the whole data, and testing dataset, which contains the rest of the samples in the data. The separated training and testing data were fed into the model with which we computed the train loss, test loss, and accuracy with varying training to testing sample size ratio. The train test ratio refers to the size of the training samples over the size of the testing samples. As it is shown in figure 3 below, train test ratio is varied between 0.01 and 0.99 with a step size of 0.01. When the train test ratio reaches about 0.75 to 0.85, we obtain a small test loss and the difference between train loss and test loss is also small. That means we were not overfitting.

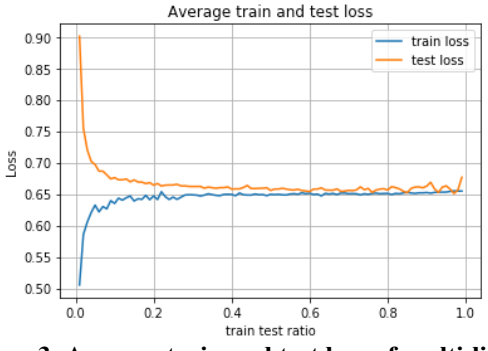

Figure 3. Average train and test loss of multi-linear regression

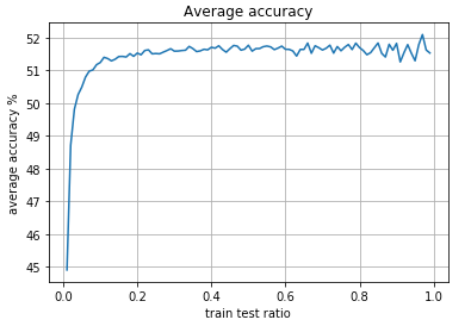

Figure 4. Testing accuracy (%)

With the train test ratio increased from 0.01 to 0.99, the prediction accuracy on the test set was increase by a decent amount from 45% to nearly 52%. The increase in accuracy agree with the decrease in train test difference in loss: the training dataset needs to be large enough (train test ratio > 0.5) to consider as many variations as possible and also cannot be too large (train test ratio <0.9) to prevent a good generalization of the model. A train test ratio of 0.75 is used for several variations of regression model later.

C. Ridge regression and Lasso regression

To confirm that there is no overfitting, we apply penalty terms to check if results can obtain statistically significant improvement.

Ridge regression applies a L2 regularization to the model, shrinking all parameters by the same factor. Alpha value decides the severity of the penalty term. A list of alpha values (from 0.01 to 1 with a step size 0.01) were used to explore how the severity of the L2 penalty affect the loss and accuracy of the data.

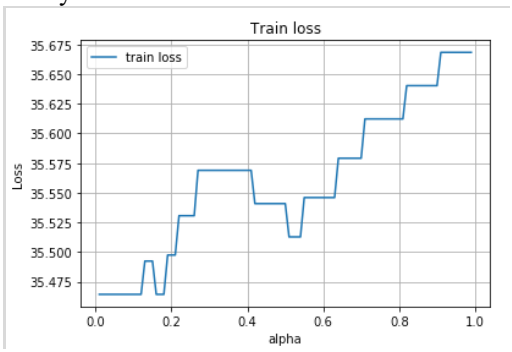

Figure 5. Train loss of Ridge regression

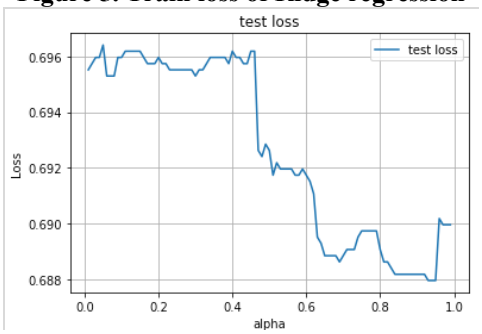

Figure 6. Test loss of Ridge regression

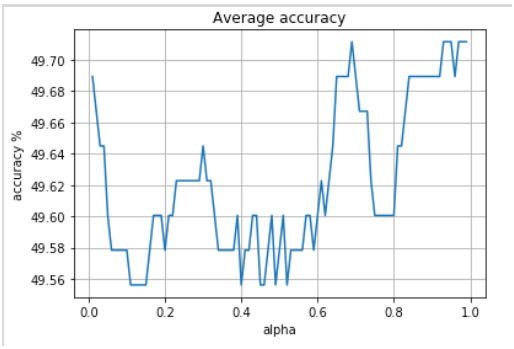

Figure 7. Test accuracy of Ridge regression

Train loss (figure 5 and 6) decreases as alpha increases while test loss decreases. The middle point where alpha = 0.5 provide both relatively low train loss and test loss. The accuracy does not change much (within 0.5%) with variation of alpha value. The L2 penalty term does not significantly improve the results.

Lasso regression, unlike Ridge regression, applies another penalty term, L1 penalty, to the model. It tends to create a more sparse, understandable model by eliminating some of the highly correlated predictor variables. We also varied the value of the penalty term (from 0.01 to 1 with step 0.01) to try to maximize accuracy.

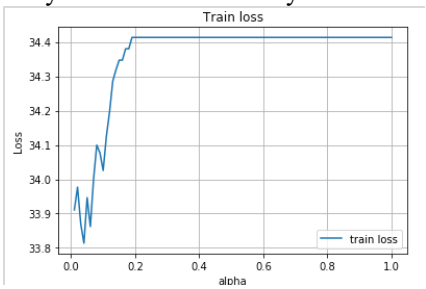

Figure 8. Train loss of Lasso regression

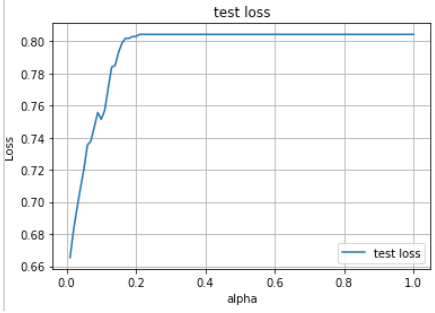

Figure 9. Test loss of Lasso regression

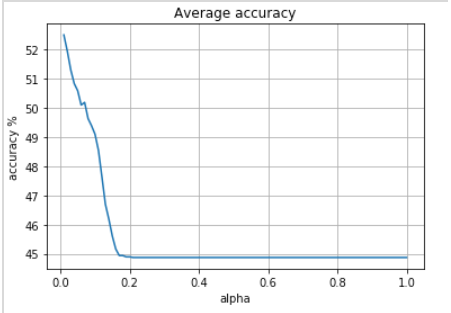

Figure 10. Prediction accuracy

With larger alpha, the loss actually increases and accuracy decreases. With experimenting with Ridge and Lasso regression, we realize it is better to not apply a severe regularization on the model. It agrees with the result obtained from using multi-linear regression without any

penalty terms. The result in multi-linear regression shows even without any penalty, we are basically not overfitting the model by having a small test loss.

D. Kernel Logistic regression

In the wine quality scenario with discrete class labels, classification algorithms like logistic regression has an advantage that linear regression does not, that is logistic regression does not need to set up a threshold based on which classification can be done.

A kernel logistic regression is implemented partially based on functions that we wrote in lab. The target variable was first relabeled into three new classes—bad, average and excellent—then they were fed into the logistic model with a list of gamma and a list of lambda values as parameters. The accuracy is plotted against a grid of gamma and lambda values shown in the figure 11 below.

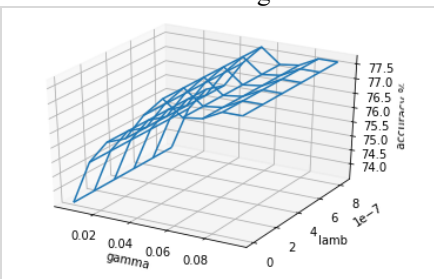

Figure 11. Accuracy of Logistic regression

The X and Y axis are gamma and lambda values, respectively. The Z axis is the accuracy. Lambda does not have a large influence on the change of accuracy in general. However, as gamma increases, the accuracy initially increases and it drops slightly after gamma larger than 0.07. The best gamma value is around 0.05.

E. Support vector machine

Support vector machine is another promising classification algorithm, which employ one-against-one approach for multiclass problems. We used the built-in SVM in the scikit-learn library with GridsearchCV function to facilitate automatic hyperparameter tuning, which we cannot do with kernel logistic regression. Overall, the accuracy can reach 75%.

F. Kth-Nearest Neighbor Algorithm

Also known as the 'Lazy Learning', Kth-Nearest Neighbor(KNN) is the another simple method we used to train and test based on finding a satisfied k value which is the parameter that refers to the number of nearest neighbors to include in the majority of the voting process.

Before we implemented the kNN algorithm, since each variable in our data have different ranges and units for example the sulfur dioxide property is usually over 100 but the chlorides mainly converge at around 0.05, we separated into two situations: 1. Training with the original data; 2. Training with the rescaled data. In the second situation, we rescaled all training data based on the original range into a new range from -1 to 1. For each situation we also compared with whether or not relabel the output.

The result verified our hypothesis that both rescale and relabel is necessary for Kth-Nearest Neighbor Algorithm. The following plots will easily demonstrate how rescale and relabel will improve the accuracy.

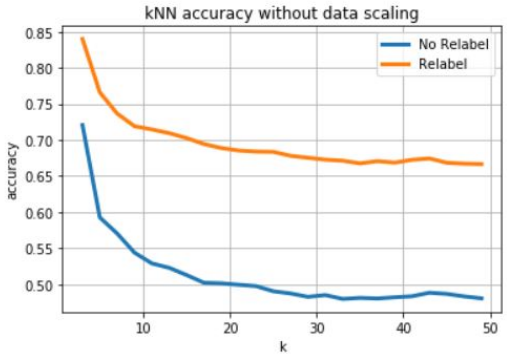

Figure 12. kNN Accuracy without Data Scaling

The above plot shows that the kNN model without data rescaling. The best k value for both No Relabel and Relabel model is 3 with the accuracies respectively 72.07% and 83.9%.

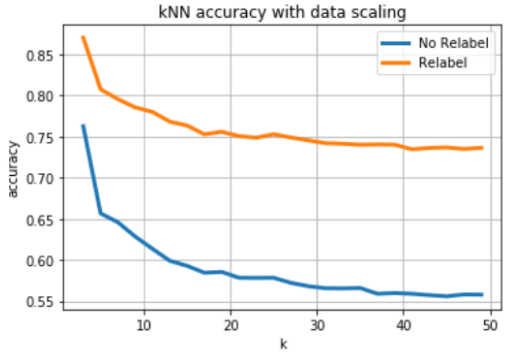

Figure 13. kNN Accuracy with Data Scaling

The above plot shows that the kNN model with data rescaling. The best k value for both No Relabel and Relabel model is 3 with the accuracies respectively 76.23% and 86.9%. The result make perfectly sense because if we didn't rescale the data, Euclidean distance calculated for each point will be unbalanced for certain properties. For example, large value will contribute more on the final distance while others will be too small to influence the final prediction. Also simplify the output classification by relabeling the quality into three categories helped to improve the overall accuracy about 20%.

However, from the plot, we also observed that while k increased a little bit, the performance dropped a lot at first and converged to a stable accuracy when k is greater than 30. At that time, even if we did both rescale and relabel job, the best accuracy is still less than 75%. Noting that choosing smaller values for K can be noisy, even if the plot shows preferable accuracy at k=3, it is still not reliable to determine the k value. In other hand, larger values of K will have smoother decision boundaries which means lower variance but increased the bias. Plus, computationally expensive for all cases (46 seconds for no scaling data; 42 seconds for scaling data) prevented us from choosing Kth-Nearest Neighbor Algorithm as a reliable methods in white wine quality prediction.

G. Neural Network

Neural Networks are a set of algorithms, modeled loosely after the human brain, that are designed to recognize patterns. In our project, we used Neural Networks helping us to classify the quality of white wine by trying to train and group unlabeled data according to similarities among the example inputs. In this report, I will show the comparison among using different network structures, different hidden layer activation functions and different kernel initializers.

The baseline situation in Neural Network training and testing is using original dataset to predict original wine quality which is in range zero to ten. The neural network structure for the baseline situation contains two layers, one hidden layer included 100 nodes with sigmoid activation function and one output layer included 10 nodes with softmax activation function which we kept using softmax as the output layer activation function for all rest of cases since softmax converts a real vector to a vector of categorical probabilities.

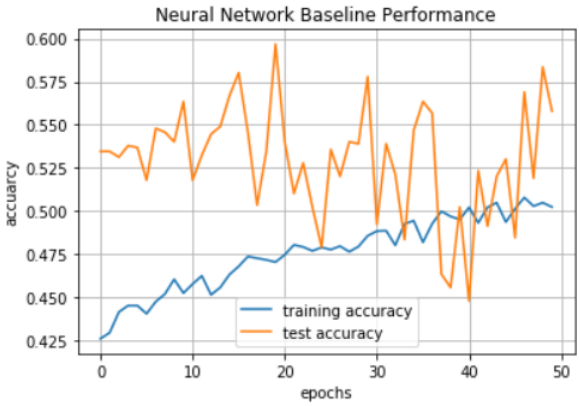

Figure 14. Neural Network Baseline Performance

The above figure demonstrates the performance of our baseline situation which is expected. While the training accuracy increases during the training process, the overall training accuracy is converged at around 50% and the test accuracy is highly vacillated. Next, we trained our dataset with rescaled input shown below but expect a little bit improve on training accuracy, overall performance remained unreliable.

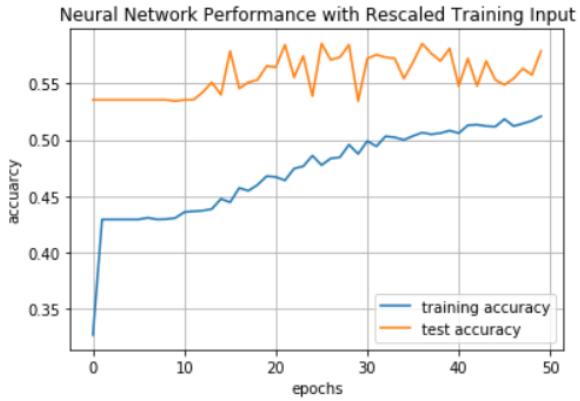

Figure 15. Neural Network Performance with Rescaled Training Input

Then, instead of predict the original quality, we relabeled the output into three categories: bad (0-5); average (6-7); good(8-10) and modified out neural network structure with only three output nodes. The results shown below help us visualize how relabel could improve our machine learning mode. Both training accuracy and test accuracy increased about 20%. And the final test evaluate accuracy is about 75.5%. In order not to overfit the model, we chose to use 50 epochs with 100 batch size. The train loss and test loss would also verify by showing both losses keep decreasing during the train/test processes.

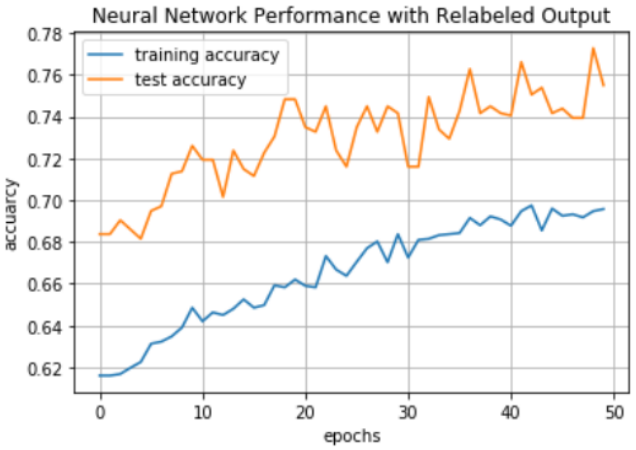

Figure 16. Neural Network Performance with Relabeled Output

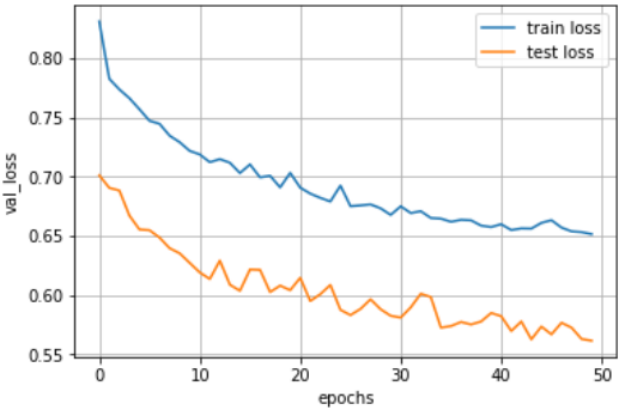

Figure 17. Train Loss and Test Loss in Relabeled Model

Then, in order to improve the rescaled neural network model, I tried four different method: 1. Adding more hidden layers; 2. Training with different hidden nodes; 3. Trying different activation functions for hidden layer; 4. Using different kernel initializer. The report will use tables to deliver those comparison results.

- Adding more hidden layers:
  In this method, we kept the number of nodes for each hidden layer as 100 and used sigmoid active function. From, Table1, we observed that there is no guarantee to improve accuracy by simply adding more layers.

Table 1. Comparison with different number of hidden layers

	1 Hidden Layer	2 Hidden Layer	3 Hidden Layer
Test Acc	75.5%	73.9%	74.1%

- Training with different hidden nodes:
  In this method, we did the experiment on the third hidden layer structure. For each case, we modified the number of nodes in each layer. The result is still not promising.

Table 2. Comparison with different number of nodes

	100,100,100 nodes for three hidden layers	100,100,50 nodes for three hidden layers	100,50,25 nodes for three hidden layers
Test Acc	74.1%	75.2%	68.3%

- Trying different activation functions for hidden layers:
  In this method, I chose five different activate functions for hidden layers: sigmoid, relu, tanh, selu, and exponential. Activate function is still softmax in order to calculate the highest possibility for each category.

Table 3. Comparison with different activate function for hidden layer

	Sigmoid	Relu	Tanh	Selu	Exponential
Test Acc	75.5%	76.1%	73.8%	73.9%	74.6%

- Using different kernel initializer
  In this method, I chose five different kernel initializers (Uniform, Random Normal, Truncated Normal, Glorot Normal, and Variance Scaling) putting into the model with only one hidden layer using sigmoid activate function.

Table 4. Comparison with different kernal initializers

	Uniform	Random Normal	Truncated Normal	Glorot Normal	Variance Scaling
Test Acc	75.5%	74.2%	73.2%	74.6%	74.0%

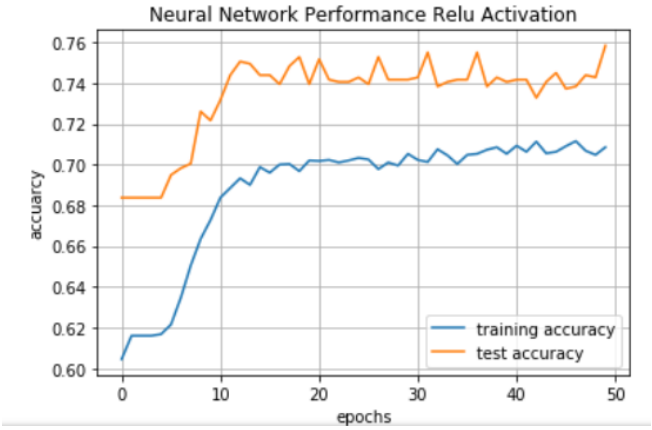

Figure 11. Neural Network Performance using Relu Activation Function

In conclusion, we did not find and obtain a better model which could significantly help us improve our prediction accuracy. For all attempts, the highest accuracy is 76.1% by using relu as our hidden layers' activation function which is similar as our origin result shown in Figure 18. But by trying out so many combinations, we found that some of the attempt may increase both train and test accuracy faster by using less epochs. For example, when we used relu, it only took less than 10 epochs to reach the highest accuracy and remained same for rest of epochs which is shown in Figure 18. This case might tell us by using relu in this project, the computational cost might be less expensive compared with other methods.

Conclusion:

In this machine learning project, we explored both regression and classification methods to make precise predictions of White Wine Quality. Overall, the classification methods including SVM, kNN and Neural Network model provide us a better performance than those of regression models. Our baseline, based on the distribution of all quality, is to simply predict 6 for all cases and achieves a 44% of accuracy. The final machine learning model we chose for this project is to use the Neural Network Algorithm by rescaling the output into three categories, which leads us to 76.1% if using the relu activation function. Although some of other methods may obtain higher accuracy, for example kNN when k is set as 3, the overall performance of kNN methods and the potential noisy prevented it from being our final model.

However, considering the results of all machine learning methods used in this project, the highest accuracy we obtained is not promising which partly because of the limited number of data in training set (only 5000 examples), or because of the highly subjective wine qualities. On the other side, we did not exhaustively play around our train model. In the future, we will try more options in Neural Network field by using different optimizer algorithm and building a more complex network structure implemented with various deep learning tools. Besides neural networks, SVM is also a very powerful technique, but it depends on correctly setting the hyperparameters and the changes of cross validation fold size could have influence on the accuracy and stability of the final result. We hope to spend more time on GridSearchCV and find out how it works to explore more about the impact of cross validation and hyperparameters in SVM.