Red Wine Quality Classification

Using Machine Learning Technique

NGUYEN Dang Hoa *(Author)*
ICT Department
Vietnam France University
Hanoi, Vietnam
hoazell41195@gmail.com

NGUYEN Gia Khang *(Author)*
ICT Department
Vietnam France University
Hanoi, Vietnam
nguyengiakhang1417@gmail.com

PHAM Ha An *(Author)*
ICT Department
Vietnam France University
Hanoi, Vietnam
an.phamhaan@gmail.com

***Abstract* - Based on the chemical and physical characteristic of wine, evaluation can be made to a high precision. This project aims at building and training a fully automated wine evaluating system on the base of the Java programming language with a view to correctly grade the quality of wines based on their chemical properties at the minimum cost with no human error.**

Keywords: dataset, training, testing, linear regression, k-NN, extreme learning machine, echo state networks, additive

I. INTRODUCTION

In this project, we consider a set of observations on a number of red wine varieties involving their chemical properties and grading by sommeliers. Wine industry shows a recent growth spurt as social drinking is on the rise. The price of wine depends on a rather abstract concept of wine appreciation by these professional wine tasters, opinion among whom may have a high degree of variability. Pricing of wine depends on such a volatile factor to some extent. Another key factor in wine certification and quality assessment is physiochemical tests which are laboratory-based and takes into account factors like acidity, pH level, presence of sugar and other chemical properties. For the wine market, it would be of interest if human quality of tasting can be related to the chemical properties of wine so that certification and quality assessment and assurance process is more controlled.

The project will thus focus on researching and implementing some methods of Artificial Intelligence and Machine Learning for the purpose of predicting the quality of red wine based on some chemical properties as mentioned above. The result will be the accuracy of each method and the possibility to apply them into reality for actual wine assessment process. The method that is able to handle the prediction accurately can be researched further in order to complete its performance in this problem particularly, and to be widely utilised in the wine production industry.

II. DATA ANALYSIS & PRE-PROCESSING

1. Data Analysis

The chemical properties of each wine sample is taken into account when grading its quality. We look into 1599 samples of Red Wine from the Wine Quality Dataset [1]. All wines are produced in a particular area of Portugal. These data are collected on 12 different properties of the wines one of which is quality, based on sensory data, and the rest are on chemical properties of the wines including density, acidity, alcohol content, etc. All chemical properties of wines are continuous variables. Quality is an ordinal variable with possible ranking from 1 (worst) to 10 (best). Each variety of wine is tasted by three independent tasters and the final rank assigned is the median rank given by the tasters.

Attribute information:

Input variables (based on physiochemical tests):

01 - fixed acidity
02 - volatile acidity
03 - citric acid
04 - residual sugar
05 - chlorides
06 - free sulphur dioxide
07 - total sulphur dioxide
08 - density
09 - pH
10 - sulphates
11 - alcohol

Output variable (based on sensory data):

12 - quality (score between 0 and 10)

Observing the Red Wine dataset, it is noticeable that all variables have the outliers – those values which "lies outside" (much smaller or larger than) most of the common values in the set of data.

- *fixed acidity*, *volatile acidity* and *citric acid* have outliers. If those outliers are eliminated distribution of the variables may be taken to be symmetric.
- *residual sugar* has a positively skewed distribution; even after eliminating the outliers distribution will remain skewed.
- Some of the variables, e.g. *free sulphur dioxide*, *density*, have a few outliers but these are very different from the rest.
- Mostly outliers are on the larger side.
- Most of the red wine has *quality* of 5, 6 or 7 whereas there were just a minor number of wine categorized into level 3, 4 or 8 and no instances are in the group 1, 2 or 9.
- Histogram illustrating the distribution of wine quality of the original dataset:

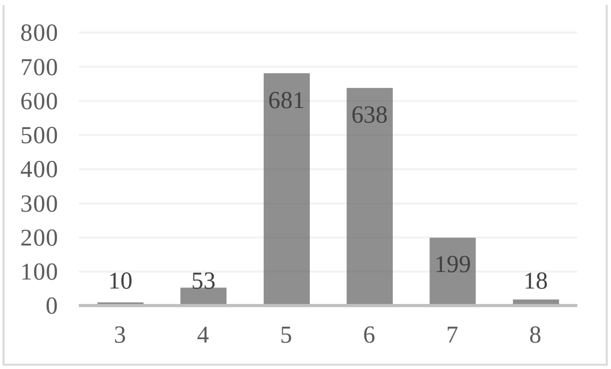

Figure 1: Wine quality (dataset: original)

2. *Pre-processing*

First of all, the dataset is divided by an 80-20 ratio based on a random pick-up. The former is used as a training set while the latter is used for testing the trained learning machine.

Since the scales of different attributes in the dataset are generally varied in several separated ranges, thus significantly affects the ability to learn as well as the accuracy of the program, dataset normalization is necessary before we move on to any method.

Clearing outliers is another crucial pre-processing step. However, in our case, the dataset are too evenly distributed to the point that the removal of outliers would result in more incorrect value than leaving the set intact. Therefore, we will keep the outliers.

III. METHODOLOGY

1. *Linear Model (Linear Regression)*

Linear regression is one of the algorithms used to predict an output ***y*** given an input ***x*** based on result of training examples. Linear regression is modelled as follow:

$$y_i = \beta_1 x_1^i + \beta_2 x_2^i + \cdots + \beta_p x_p^i + \varepsilon^i$$

It can also be rewritten as a vector form:

$$y = \beta x + \epsilon$$

where β is the model parameter and ϵ illustrates measurements or other noises. Our task is to compute the β from a training dataset before any predictions ***y*** can be made for provided ***x*,** such that the distance between predicted values and real ones is optimal (as low as possible)

How to find β

Given a set of inputs and outputs we would have to solve a set equations for β. The vector form can be solved using linear algebra; the solution takes the form of:

$$\beta = (X^T X)^{-1} X^T Y$$

This particular solution will find betas using ordinary least squares approach.

2. *k-Nearest Neighbours*
   a. *k-NN*

k-Nearest Neighbours algorithm is a non-parametric method in Machine Learning, which can be utilised for both cases regression and classification.

In the algorithm, we consider each instance in training set as a point in a N-dimension space (where N is the number of attributes). The prediction will be given by either way: majority vote or average.

- Majority vote: this method is applied for classification, in which the predicted value is computed by taking the class which has the largest number of instance among the considered k nearest points. In case there are several equal-sized classes, the prediction is taken randomly among those.
- Average: In regression case, the prediction is calculated as the average outcome of the considered k nearest points. This function was implemented in our program.

In this paper, we assess the performance of k-NN algorithm by altering k = 3, 5 and 7.

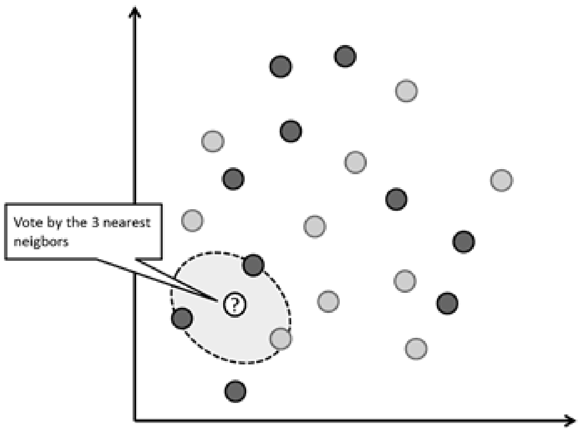

Figure 2: k-NN illustration

b. *k-NN – a variation*

k-NN variation acts almost the same way as the original algorithm. However, instead of considering k nearest neighbours, it divides the training space into cells by separating each column into k ranges. The prediction is computed by majority vote or average of all points in the considered cell.

For our datasets, we implement the algorithm in only 2 cases k = 2 and k = 3.

3. *Artificial Neuron Network (ANN)*
   a. *Feedforward:*

Extreme Learning Machine is a particular case of feed-forward ANN method.

- Feedforward is biologically inspired classification algorithm. It consist of a (possibly large) number of simple neuron-like processing *units* (also called *nodes*), organized in *layers*. Units in one layer will form connections with units in another layer, following a single direction. These connections are not all equal:

each connection may have a different strength or *weight*. The weights on these connections encode the knowledge of a network. [2]

- In this network, the information moves in only one direction, forward, from the input nodes, through the hidden nodes (if any) and to the output nodes. There are no cycles or loops in the network. [3]

In Extreme Machine Learning approach, there is only one single layer of hidden nodes where the weights between input and hidden nodes are arbitrarily selected and will not updated. We will, however, update the weights from hidden to output nodes.

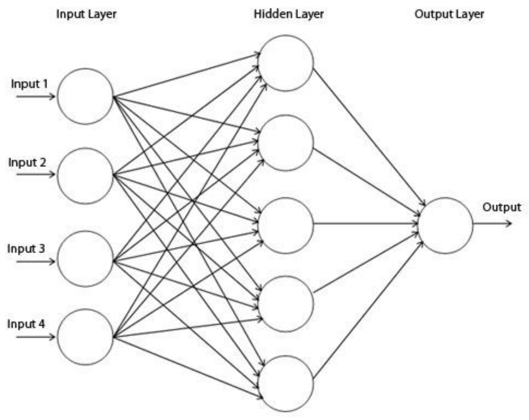

Figure 3: Extreme Machine Learning model

Networks generated from this method is far more efficient than those which trained from backpropagation.[4]

b. Recurrent Neural Networks: Echo State Network

Recurrent Neural Network (RNN) is a class of Artificial Neural Network where the connections between the units may create a directed cycle, which can be used to process a sequence of inputs. The methods belong to RNN is thus extremely effective for dependent datasets, such as handwriting and speech recognition.

Echo State Network is an idea to implement RNN. Unlike Extreme Learning Machine, Echo State Network uses several random hidden units to process the data. However, the weight between hidden layer (reservoir) and output layer is the only that needs computing, while the others are generated randomly.

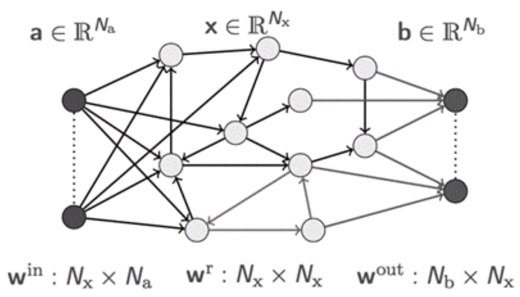
$\mathbf{a} \in \mathbb{R}^{N_a}$
$\mathbf{x} \in \mathbb{R}^{N_x}$
$\mathbf{b} \in \mathbb{R}^{N_b}$
$\mathbf{w}^{\text{in}} : N_x \times N_a$
$\mathbf{w}^{\text{r}} : N_x \times N_x$
$\mathbf{w}^{\text{out}} : N_b \times N_x$

Figure 4: Echo State Network model

In Echo State Network, the number of reservoir must be much larger that input units'. Therefore, they can be seen as the expansion functions of the input space.

4. Additive Models:

Additive Models is the generalization of the *Linear Models* method. In this approach, the additive nature is still be maintained whilst the simples terms of the linear equation $\beta_p x_p^i$ will be replaced with $\varphi_i(x^i)$ where φ_i is a non-parametric function of an input x^i. In particular, instead of using a typical coefficient for each variable (additive term) in the model, an unspecified (non-parametric) function is provided for each term in order to sufficiently estimate the output values $\boldsymbol{y}$.

IV. IMPLEMENTATION

1. Implementation

All the mentioned methods are implemented in Java environment.

Generally, the program will take the pre-generated "TrainingWineData" as the set used for "learning" process. It is randomly chosen from the original dataset as referred in II.2. Meanwhile, another available set named "TestWineData" will be used as input in order to compute the relevant predicted values. The *mean square error* will be calculated for each individual method for further comparison.

Again, it is noteworthy that as we will not remove any outliers from the training and testing datasets due to the reason mentioned in II.2.

2. Result Demonstration

To begin with, we would like to introduce the distribution of real wine quality of the testing dataset (from now on, all the chart relating to *Wine quality* will be implied using *TestWineData)*

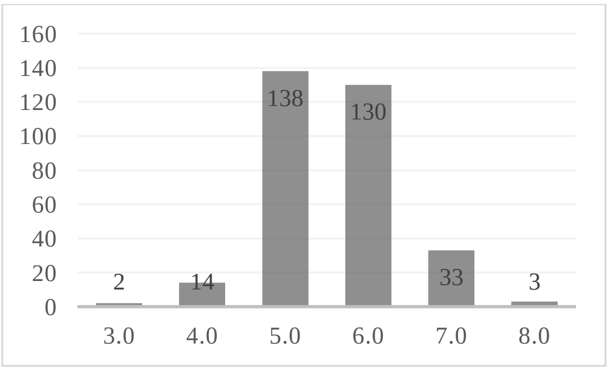

Figure 5: Wine quality – real value

This figure will be employed to compare with other predicted wine quality of the testing dataset based on a typical method. Now we should begin.

a. Linear Model

Mean Square Error: 0.411

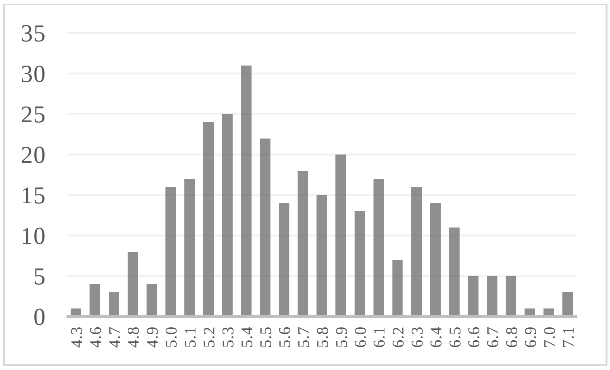

Figure 6: Wine quality – predicted value – method: LM

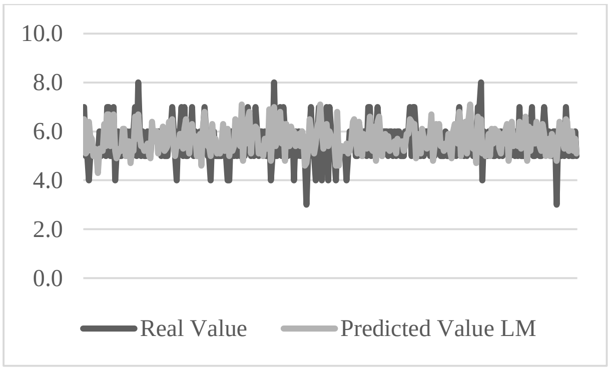

Figure 7: Comparison between real and predicted value (method: LM)

b. k-NN (without normalization)

i. with k = 3

Mean Square Error: 0.330

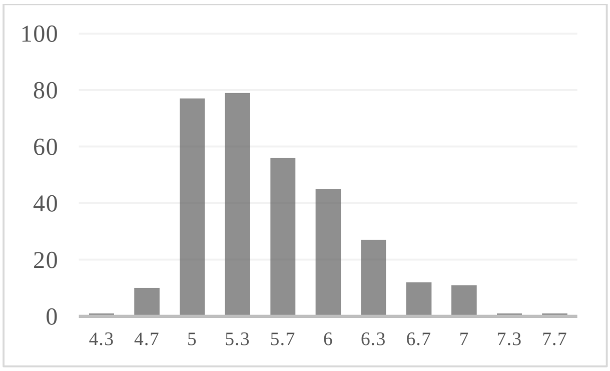

Figure 8: Wine quality – predicted value – method: k-NN with k =3

ii. with k = 5

Mean Square Error: 0.319

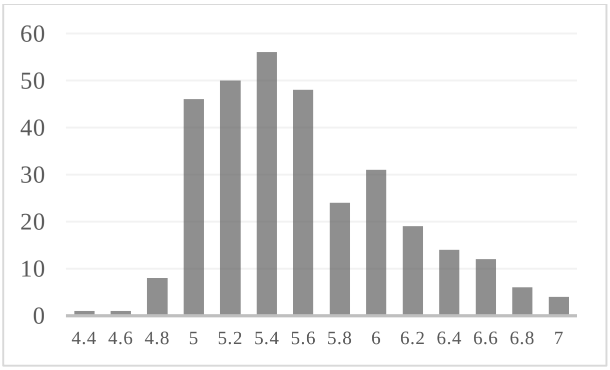

Figure 9: Wine quality – predicted value – method: k-NN with k =5

iii. with k = 7

Mean Square Error: 0.352

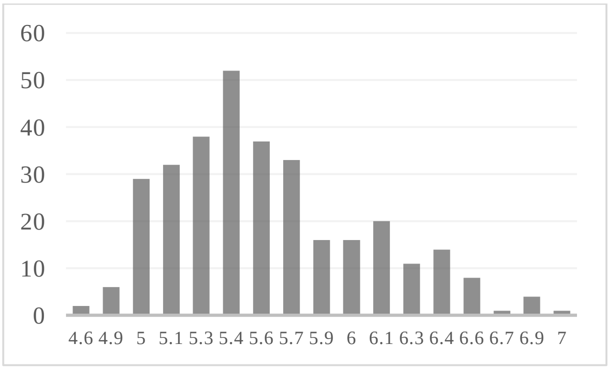

Figure 10: Wine quality – predicted value – method: k-NN with k = 7

c. k-NN variation (without normalization)

i. with k = 2

Mean Square Error: 0.431 (average)

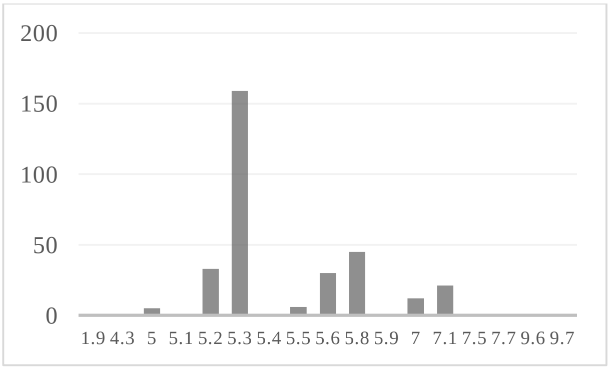

Figure 11: Wine quality – predicted value – method: k-NN variation with k = 2

ii. with k = 3

Mean Square Error: 0.228 (average)

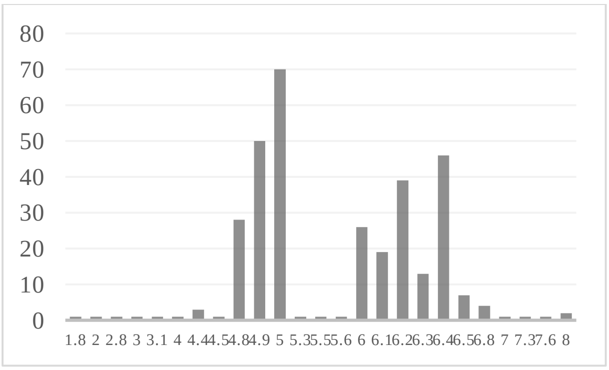

Figure 12: Wine quality – predicted value – method: k-NN variation with k = 3

d. ELM

i. With no. of hidden nodes (H) = 20

Mean Square Error: 0.512

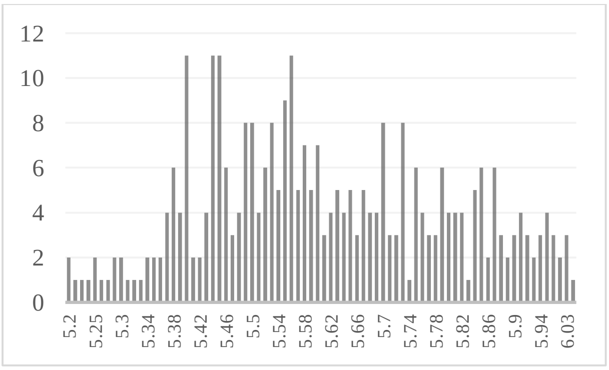

Figure 13: Wine quality – predicted value – method: ELM with no. of H = 20

ii. With no. of hidden nodes (H) = 100

Mean Square Error: 0.430

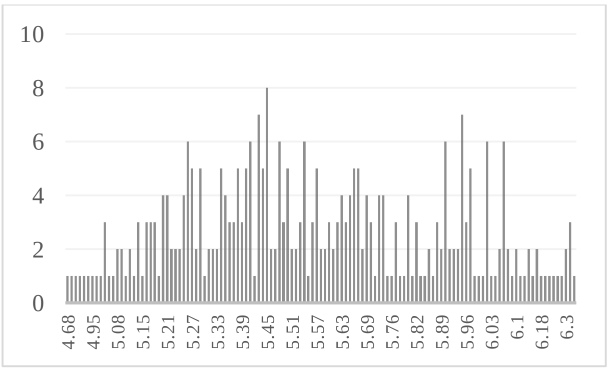

Figure 14: Wine quality – predicted value – method: ELM with no. of H = 100

iii. With no. of hidden nodes (H) = 500

Mean Square Error: 0.398

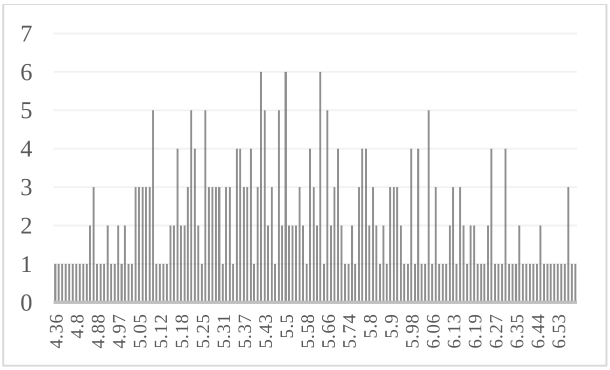

Figure 15: Wine quality – predicted value – method: ELM with no. of H = 500

iv. With no. of hidden nodes (H) = 5000

Mean Square Error: 0.403

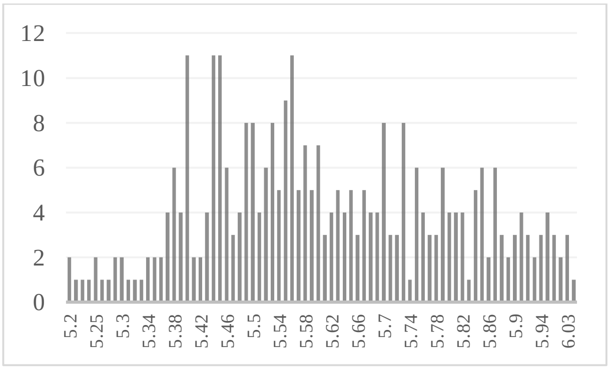

Figure 16: Wine quality – predicted value – method: ELM with no. of H = 5000

e. ESN

i. With no. of hidden nodes (H) = 20

Mean Square Error: 0.567

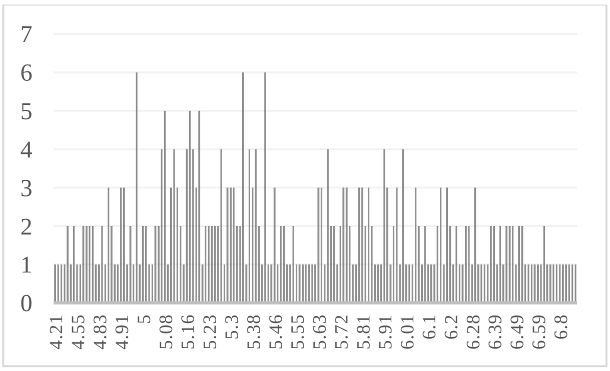

Figure 17: Wine quality – predicted value – method: ESN with no. of H = 20

ii. With no. of hidden nodes (H) = 100

Mean Square Error: 0.452

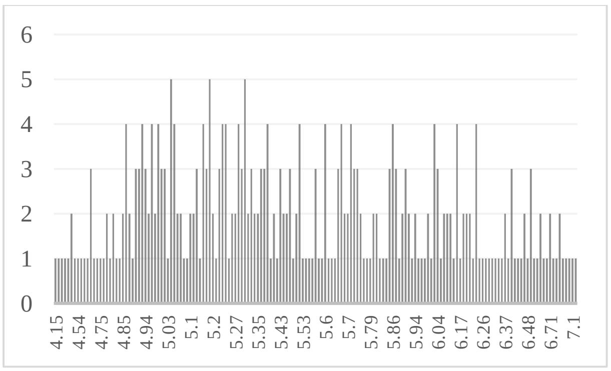

Figure 18: Wine quality – predicted value – method: ESN with no. of H = 100

iii. With no. of hidden nodes (H) = 500

Mean Square Error: 0.467

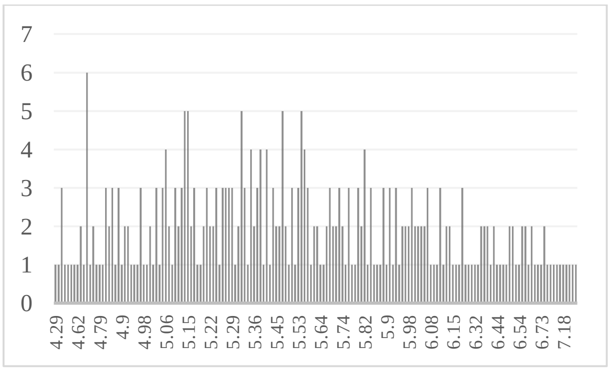

Figure 19: Wine quality – predicted value – method: ESN with no. of H = 500

f. Additive Model

Mean Square Error: 0.622

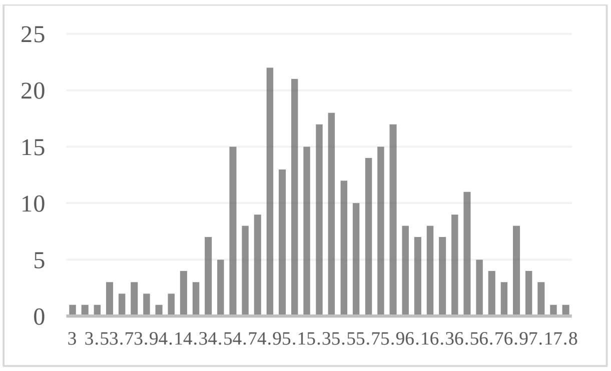

Figure 20: Wine quality – predicted value – method: Additive Models

V. DISCUSSION

1. Linear Models (Linear Regression)

Linear Regression: Although it is the most simple method in machine learning, linear regression gave the good predictions with an expected mean squared error at around 0.42 (0.41 with outliers and 0.42 without outliers). This can be explained by the fact that the values of our datasets has a small variation and number of outliers (outputs stand mostly at 5 and 6 as describe above), which consequently makes it better-fit the regression plane. Furthermore, the datasets can be assumed to be entirely independent to each other and thus does not cause the multicollinearity, which can give severe erroneous to the regression coefficient and negatively affects our predictions.

Runtime is another factor that promotes the linear regression method. In the program using Matrix computation of JAMA library and the environment as described in the table below, the measured runtime stands at approximately 300 to 400 milliseconds, which renders it the fastest methods in the project. This is the reason why linear regression is the most popular method.

Processor	2.7 GHz Intel Core i5
RAM	8 GB 1867 MHz DDR3
OS	OS X 10.11
Compiler	IntelliJ IDEA 14 CE

2. k-NN

With normalization		Without normalization		
k	MSE	k	MSE	Runtime
3	0.732	3	0.330	564 ms
5	0.547	5	0.319	551 ms
7	0.539	7	0.352	535 ms

These tables are the mean squared error for each value of k in two cases: with and without normalisation. As can be seen, normalisation before processing makes the program predict incorrectly with a large mean squared error. It is quite understandable since normalisation pre-process would considerably narrow the scale of input values, thus decrease the distance between each point in the feature space. Consequently, the selection of k nearest neighbours becomes extremely inefficient.

The k-NN method without normalisation gives much better predictions with smaller mean squared error. A k = 5 is the most appropriate value for this datasets as its corresponding mean squared error stands at 0.319. k = 3 and k = 7 also gives relatively accurate predictions with small erroneous. The program will perform less correctly as k increases larger than 7 because it may mistake other outcomes with the actual one.

k-NN algorithm also has a small complexity with the tested runtime at only about 550 milliseconds. This is the second fastest algorithm that was researched in this paper.

3. k-NN variation

With normalization		Without normalization		
k	MSE	k	MSE	Runtime
2	7.521	2	0.431	25060 ms
3	7.657	3	0.228	37714 ms

Similarly, the k-NN variation predicts much more correctly in case of no pre-normalisation. In the program, we only considered two values of k: k = 2 and k = 3.

As can be observed from the table, k = 3 gives the most expected results with the average mean squared error stands at only 0.228 (after 10 times running). Thus it can be concluded that k-NN variation with k = 3 is the most accurate algorithm for machine learning that was implemented in this project. However, there are some drawbacks that need considering before applying it into reality.

Firstly, the complexity of k-NN variation is extremely large since it creates k-NN cells in the training space. In our dataset, N is equal to 11, thus would force the program to process the total number of k11 cells and therefore severely affects the runtime as k increases. In our environment, the runtime when k = 2 is 29 seconds, when k = 3 is 32 seconds. This will becomes a remarkable problem in a larger dataset which has more attributes.

Other weakness of k-NN variation is the poor stability of the results. Since the predictions will return a random number if the searched cell is empty, the results will certainly depends on the random function and therefore vary at different time. However, this can be overcome by creating more non-empty cells by increase their size in our training space. For instance, in this project, k = 2 will give the more stable predictions than k = 3.

4. *ELM*

No. of hidden nodes (H)	MSE (avg)	Training time (ms)
20	0.512	8436
50	0.491	8256
100	0.430	8102
200	0.411	8241
500	0.398	13035
1000	0.399	19455
5000	0.403	62870

In this paper, we assess the performance of ELM on a pre-normalized datasets by comparing the mean squared error for different number of hidden nodes H.

The training time of ELM is certainly also of consideration, since the trade-off is the most important factor for us to make the final decision on choosing the number of hidden nodes. In this dataset, H = 500 would be the most appropriate as it gives the smallest mean squared error and an acceptable training time.

5. *ESN*

No. of hidden nodes (H)	MSE (avg)	Training time (ms)
20	0.567	8942
50	0.424	9148
100	0.452	8610
200	0.508	8676
500	0.467	15379
1000	0.560	38887
2000	0.891	127784

Similar to ELM, the performance of ESN is again tested by altering the number of hidden nodes and comparing the mean squared error in those cases. As can be perceived, the errors of this method varies significantly in different values of H. However, unlike ELM, the predictions are only acceptable in the case $H \in [50 ; 500]$. The increasing number of hidden nodes would cause a side effect that makes our prediction becomes strongly dependent to the trend of training set and thus totally incorrect to the test set.

Aside from that, the prediction of ESN can be concluded to have a lower accuracy than other methods' in this paper for the fact that: ESN is only useful for the sequential or time-series datasets because the computation of weights in this algorithm is closely correlated to the sequence of input data. Nevertheless, our dataset is entirely independent since we can assume that each sample of tested wine has no relationship with others. Therefore, its accuracy can be expected to be not as high as other methods mentioned in this paper. The large error of the prediction as can be observed in *Figures 17, 18 and 19* suggests the same conclusion as it is much higher than other methods'.

Furthermore, because the methods covers the computation of three matrices: W_{in}, W_r and W_{out}, the complexity of ESN is considerably higher than that of ELM (where there are only two matrices). Consequently, increasing the number of hidden nodes could severely raise the runtime of our program to an unexpected value, for example, more than 2 minutes to complete the process of 2000 hidden nodes in our test.

6. *Additive Models*

Unlike the original Linear Regression method, Additive Models works ineffectively in our datasets as it gives the mean squared error at 0.622.

The reason for this inefficiency is that adding new columns of second order into our datasets increases the variation of our training data, thus directly affects the process of finding the proper regression plane. Consequently, the prediction becomes no more accurate as the original one.

VI. CONCLUSION

It is certainly unreasonable to conclude the efficiency of all methods in this report by conducting the test on only one datasets. However, via the project, we want to make a general comparison between each method's performances in the datasets that have these similar properties:

1. Medium size
2. Small-variated data
3. One-dimension output
4. Independent data

The received result suggests that k-NN Variation with k = 3 gives us the most accurate prediction. However, the considerable complexity causes an imbalanced trade-off and thus makes it an ineffective method, especially for the large datasets.

Other than that, k-NN and ELM are two methods that are most highly recommended for this type of datasets for the fact that they are both able to compute a relatively accurate prediction. k-NN with k = 5 gives us the most second closest prediction at an acceptable runtime (~550 milliseconds for each execution). ELM with 500 to 1000 hidden nodes, on the other hand, only requires training time of approximately 10 seconds for the first execution, after that, the prediction process runtime is inconsiderable (~ 60 milliseconds for each prediction).

VII. REFERENCES

[1] Wine Quality Data Set. Retrieve from https://archive.ics.uci.edu/ml/datasets/Wine+Quality.

[2] djmw (April 26, 2004). Feedforward neural networks 1. What is a feedforward neural network? Retrieve from http://www.fon.hum.uva.nl/praat/manual/Feedforward_neural_networks_1__What_is_a_feedforward_ne.html

[3] Retrieve from https://en.wikipedia.org/wiki/Feedforward_neural_network

[4] Retrieve from https://en.wikipedia.org/wiki/Extreme_learning_machine

[5] David Poole and Alan Mackworth (2010). Artificial Intelligence. Foundation of computational agents. Retrieve from http://artint.info/html/ArtInt_179.html

[6] Retrieve from http://www.cs.cmu.edu/afs/andrew/course/15/381-f08/www/lectures/regression.pdf

[7] Tadas Vilkeliskis (August 14, 2013). Machine Learning Notes—Linear Regression. Retrieve from http://vilkeliskis.com/blog/2013/08/14/machine_learning_part_1_linear_regression.html

[8] Retrieve from https://en.wikipedia.org/wiki/Recurrent_neural_network#Echo_state_network

[9] Retrieve from http://documents.software.dell.com/Statistics/Textbook/Generalized-Additive-Models