ENGR 518 PROJECT REPORT

School of Engineering
Faculty of Applied Science
University of British Columbia

Project Title: Combustion Quality Classification
Group No.: 3
Members: Lev Slepchuk, Qingyu Wang, Shweta Antony (alphabetical order)
Date: 2023-10-11

Introduction

Combustion quality can be determined by measuring the 'heat signature' of a combustion chamber in addition to its 'pressure signature'. To perform the analysis, these two signals are then recorded over a certain time interval. A manufacturer wants to study the feasibility of determining the quality of combustion using one signal only, i.e., either pressure or heat. The manufacturer intends this to be a feasibility study, and the answer should help the manufacturer know whether pressure data alone can be used to classify combustion into these two states [1].

Theory

According to the given data, we have a dataset where every point or in our case, signal belongs to one of 2 classes based on the given pressure signal. The classes in our project are 0 (Bad) or 1 (Good), which refers to the combustion quality of the signal. The goal of our project is to check if the model correctly predicts the class based on given feature values. This problem is a case of a binary classification task in supervised learning as it requires the model to assign a label to the given parameters.

Data Preprocessing

Dataset Analysis

We are given two datasets: "*pressure_data.xls*" and "*pressure_data_raw.xls*". Each dataset contains 154 rows (data), with 77 labeled as positive and the rest as negative. The first column represents the label, indicating combustion quality with a binary value of 0 or 1. The rest 201 columns contain features, specifically pressure signals recorded over a time interval. Features in "*pressure_data.xls*" range from -0.076 to 0.076 and features in "*pressure_data_raw.xls*" range from -0.156 to 0.173. In the preprocessing stage, we split each dataset into feature set (chamber pressure data) and label set (chamber quality).

Data Visualization

In order to get a better understanding of our datasets, we visualized the dataset. We plotted each time series pressure feature as a polyline and visualized separately based on two categories of chamber quality. The aim is to observe how chamber pressure varies over time for each quality classification. Because the time interval of each time series datasets is not defined. Thus, the coordinate of the x-axis is the number of features, not the time.

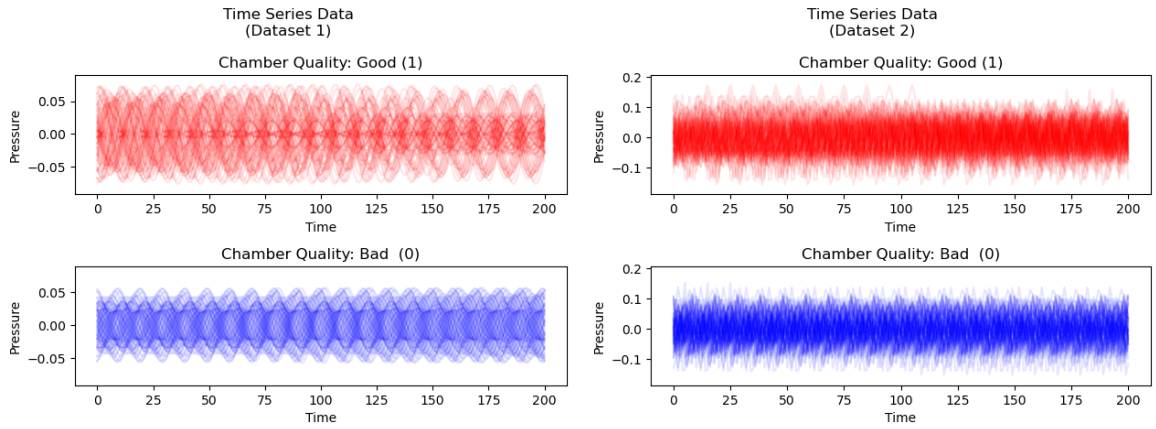

Figure 1. Visualization of the time series data from two datasets.

Upon the graphs of two datasets, we notice the frequency of the first dataset is significantly lower than the second. Since the features in two datasets have different frequencies and different value ranges, which means that the features of the two are not equivalent, we need to train and test the two data sets separately.

Feature Engineering

Intuitively, we feel that the similarities between different quality data are much greater than the differences. This could potentially cause the model to be unable to distinguish effectively. Therefore, we attempt to use feature engineering to extract significantly different features from the raw data for model learning. By observing, it can be seen that "good-quality" data has larger amplitudes compared to "bad-quality" data, while the waveform of "bad-quality" data is more uniform. Hence, we decide to plot histograms of the features based on the raw data [2]. We also try wavelet decomposition for feature extraction, but the result is the same as the time series data, the separability is not significant, so we do not use it in the following experiments.

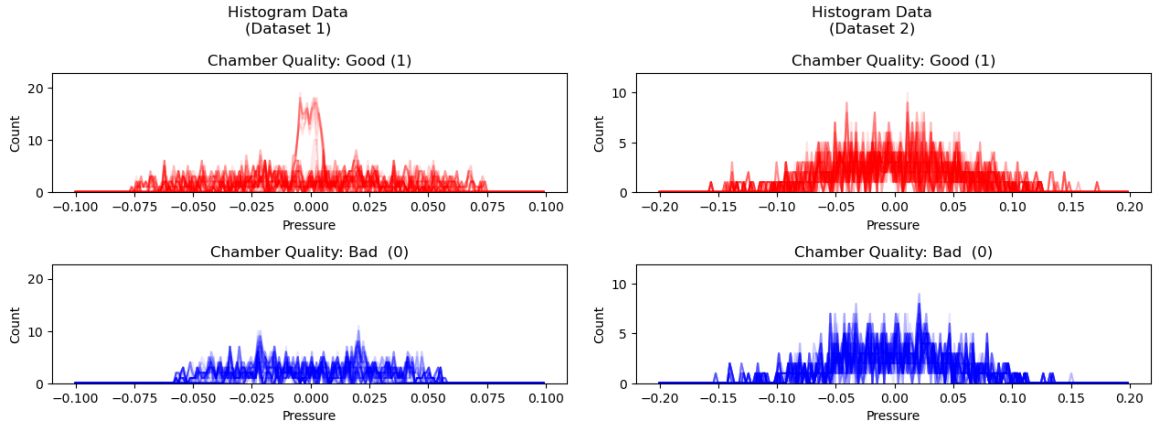

Figure 2. Visualization of the histogram data from two datasets.

Comparing the two histogram in each graph, we can see distinct differences in the feature distribution for the first dataset, the pressure range corresponding to “good-quality” encompasses a broader range, spanning from -0.08 to 0.08, whereas the “bad quality” pressure signals span from -0.06 to 0.06. And some of the “good quality” data have higher feature distribution around 0. In the graph of the second dataset, the "poor quality" data features are more evenly distributed.

Normalization

Before we start training our model, we need to normalize the datasets. Since the original datasets are time series data, we don't need to normalize each feature separately, we can scale it from -1 to 1. Similarly, the histogram data are also not independent data, so we just scale them to 0 to 1. This process improves the performance and training stability when the weights are initiated from 0 to 1.

Splitting Dataset

In this stage, the dataset is shuffled and then divided into training and testing sets at an 9:1 ratio. To maintain label balance, each label should follow the same ratio. Since our datasets are balanced there is no need to implement a resampling strategy.

Algorithm Implementation

Logistic Regression

To perform the classification task, we employ a supervised learning algorithm called Logistic Regression. Logistic Regression calculates the probability of a specific outcome occurring or not. It utilizes the sigmoid activation function to transform the values of the model, which are obtained by combining input features and their corresponding weights linearly, into a value between 0 and 1, representing the probability. The cross-entropy cost function is used to measure the discrepancy between the predicted class label and the actual value. This is optimized using the gradient descent algorithm, which adjusts the parameters in the direction of steepest descent, aiming to minimize the cost function and achieve a highly accurate model. Consequently, the error decreases with each iteration. The predict function is then used to assign class labels based on a threshold of 0.5. Values below 0.5 are assigned to class 0, while values above 0.5 are assigned to class 1. Regularization is also employed to prevent the weights from growing excessively as the number of iterations increases, thereby avoiding issues of overfitting or underfitting. First, we implemented the algorithm by autograd package, and then we used the Scikit Learn package to validate our result.

Neural Networks

In order to further explore valuable information in the dataset, we try to implement the Artificial Neural Network as well. The non-linear multi hidden layers structure in NN and the overall increased parameter quantity give the model a much stronger expression ability. We choose the academically popular deep learning framework PyTorch to build the NN structure. The model is constructed with 3 hidden layers and each layer contains 256 features. ReLU is used as an activation function with Kaiming uniform weight initialization for computational efficiency and better gradient propagation. And regularization is utilized to prevent overfitting or underfitting.

In order to prevent model overfitting, we also compared the performance of adding a dropout layer. Also we use MLPClassifier in Scikit Learn as a baseline for reference.

Evaluation Metrics

To evaluate model performance, we use a confusion matrix and calculate the accuracy to evaluate performance and compare results between individual algorithms.

Results and Discussion

In this section, we will present and discuss the results obtained from applying various algorithms to two datasets, which are represented in two formats: time series and histograms.

Accuracy (average of 100 results)	**Time Series Dataset 1**	**Time Series Dataset 2**	**Histogram Dataset 1**	**Histogram Dataset 2**
Logistic Regression Autograd	0.3856 STD 0.1072	0.3019 STD 0.1086	1.0000 STD 0.0000	1.0000 STD 0.0000
Logistic Regression Scikit Learn	0.2906 STD 0.0872	0.2894 STD 0.1006	1.0000 STD 0.0000	1.0000 STD 0.0000
Neural Network PyTorch	0.9425 STD 0.0557	0.4419 STD 0.1229	1.0000 STD 0.0000	1.0000 STD 0.0000
Neural Network PyTorch (dropout=0.5)	0.9663 STD 0.0445	0.4088 STD 0.1194	1.0000 STD 0.0000	1.0000 STD 0.0000
Neural Network Scikit Learn	0.7450 STD 0.2236	0.4550 STD 0.0977	0.7800 STD: 0.2482	0.7550 STD: 0.2499

Table 1. The accuracy of the models.

Training Time (s) (average of 100 results)	**Time Series Dataset 1**	**Time Series Dataset 2**	**Histogram Dataset 1**	**Histogram Dataset 2**
Logistic Regression Autograd	0.1170 300 iters	0.0424 100 iters	0.0422s 100 iters	0.0517 120 iters
Logistic Regression Scikit Learn	0.0043 44 iters	0.0038 32 iters	0.0021 22 iters	0.0024 24 iters
Neural Network PyTorch	0.5572 1000 iters	0.5892 1000 iters	0.1141 200 iters	0.1171 200 iters
Neural Network PyTorch (dropout=0.5)	0.7302 1000 iters	0.7466 1000 iters	0.1463 200 iters	0.1524 200 iters
Neural Network Scikit Learn	2.9370 732 iters	2.6057 705 iters	2.3134 631 iters	2.477 638 iters

Table 2. The training time of the models.

Time Series Data

We trained a logistic regression model on time series data, but the performance is bad. When we change the algorithm to a neural network, for the dataset 1, the model performance is greatly boosted, but the standard deviation of the accuracy in 100 experiments is around 0.05. Because different randomly split datasets will bring different results, the model's performance is unstable and good results may be overfitting. By adding a dropout layer, the accuracy can be further improved, but it still cannot be 100%. But for dataset 2, despite using the NN model, the results are still unacceptable.

Histogram Data

When we use histogram data for model training, we will find that even using a linear regression model can directly increase the accuracy to 100%, and the standard deviation of the accuracy for 100 experiments is 0. It can be considered that the model has sufficient stability.

Discussion

The results demonstrate that when time series data are used as input for model training, the linear regression model is completely unable to learn effective information, and the classification effect is very poor. Even if a more expressive nonlinear NN classifier is used, the model only can achieve an ideal accuracy for dataset 1, but still not be able to distinguish the dataset 2. On the contrary, if a histogram is used as input, even a linear regression model can completely classify and give perfect results. In terms of computational cost, linear regression model using histogram data has the lowest computational cost.

Although using validation sets in training is a good strategy to do the early stop and prevent overfitting, in our case, due to the lack of the data, it is even a trade off for us to decide what ratio should be used in split training and testing sets. So we try another way to prove the model's stability. We executed the code 100 times with different random split training and testing sets and calculated the standard deviation of the accuracy, which can be regarded as a cross validation process. According to the ideal result and zero standard deviation from our best model, we can conclude that this model is not overfitting.

Conclusions

In this project, we can conclude that we can achieve great results in classifying combustion quality by using only pressure data. Furthermore, feature engineering plays a significant role in machine learning. The data presented in different ways can give us very different results. When we aim to solve a practical problem using machine learning, it is essential to not only consider algorithms and models but also spend time analyzing and understanding the data that needs to be processed. By observing the data, analyzing it, and extracting key features, we can solve complex problems even with a simple model. In addition, through feature engineering, replacing complex models can also greatly reduce computational overhead and improve model production efficiency.

References

[1] M. Ihme, W. T. Chung, and A. A. Mishra, “Combustion machine learning: Principles, progress and prospects,” Progress in Energy and Combustion Science, vol. 91, p. 101010, 2022.

[2] Watt, J., Borhani, R., & Katsaggelos, A. K. (2016). Machine Learning Refined: Foundations, Algorithms, and Applications (2nd ed.). Cambridge University Press. (Chapter 9.2 "Feature Engineering and Selection: Histogram Features").

Appendix

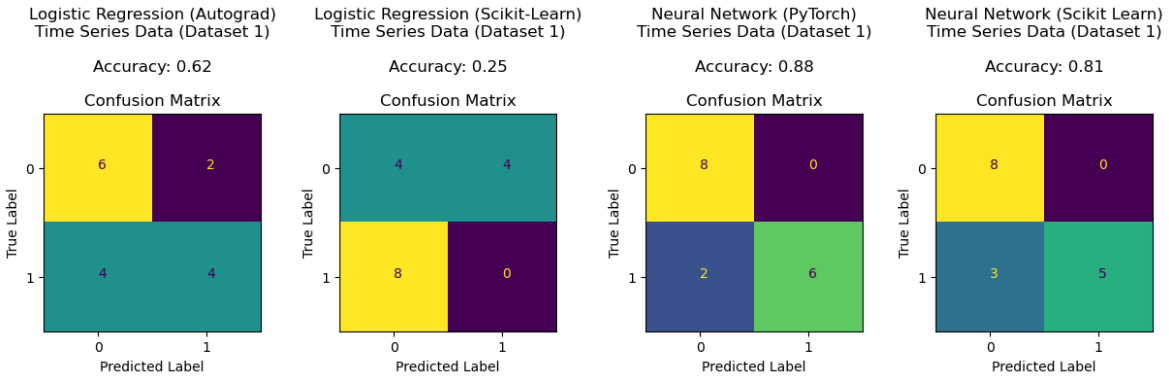

Figure 3. The performance of models for the time series data in dataset 1.

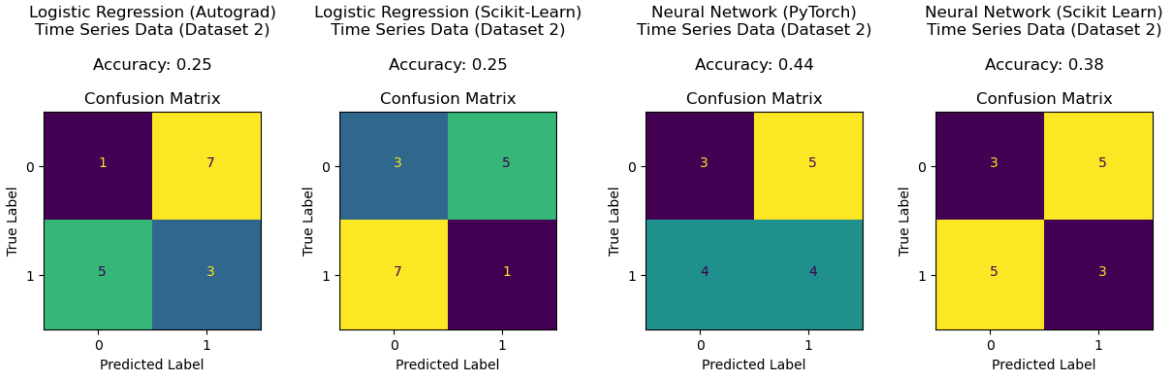

Figure 4. The performance of models for the time series data in dataset 2.

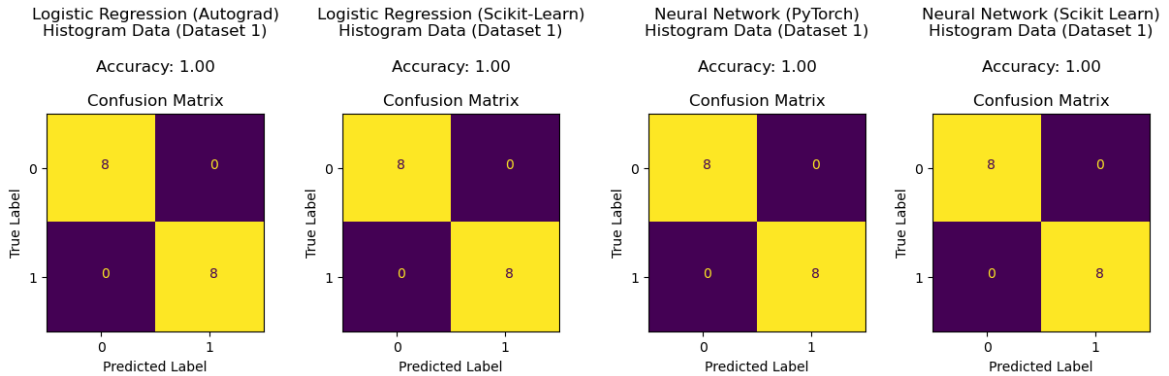

Figure 5. The performance of models for the histogram data in dataset 1.

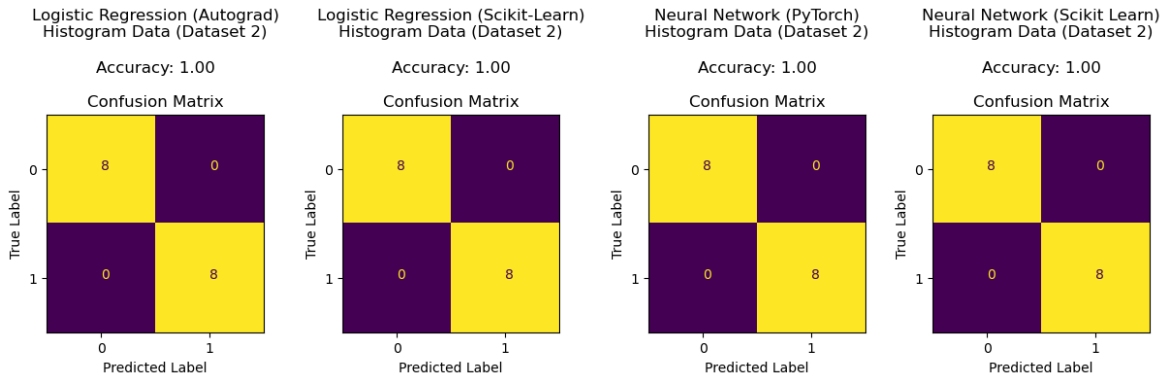

Figure 6. The performance of models for the histogram data in dataset 2.

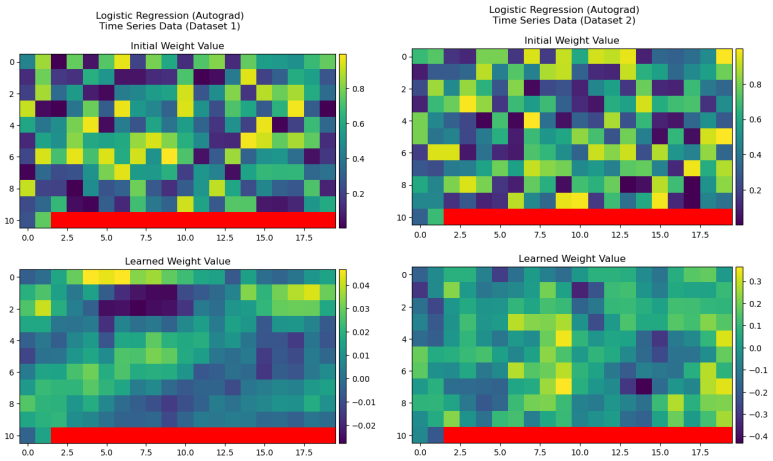

Figure 7. The weight of logistic regression models for the time series data.

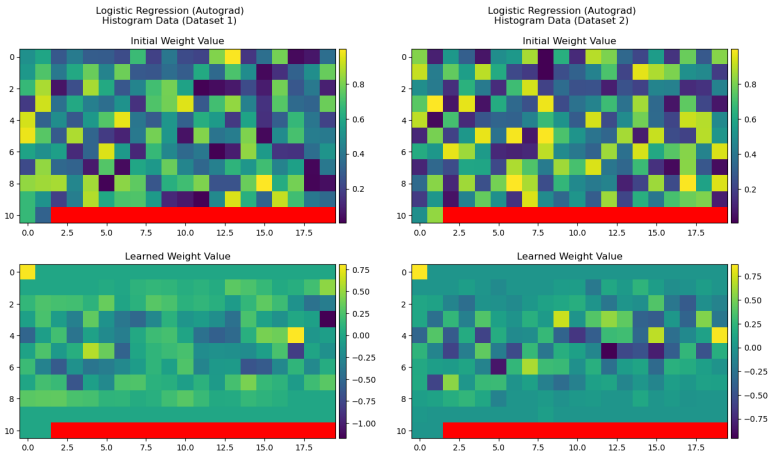

Figure 8. The weight of logistic regression models for the histogram data.

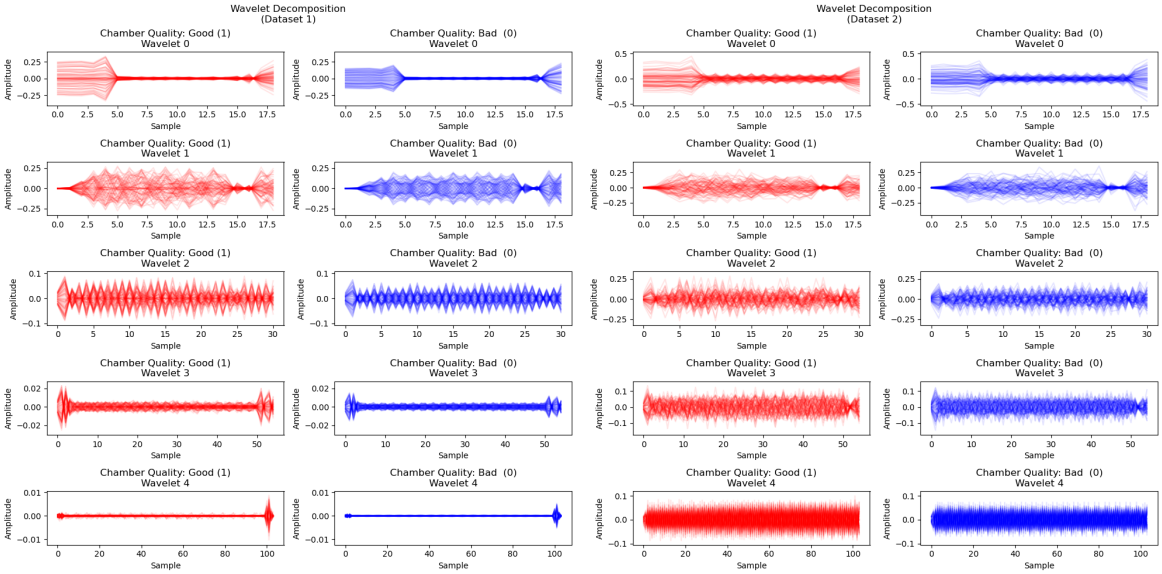

Figure 9. Wave decomposition of the time series data.

For more details, please look at the "*ENGR 518 Project Code.ipynb*".