Semisupervised Learning for FreshIndex

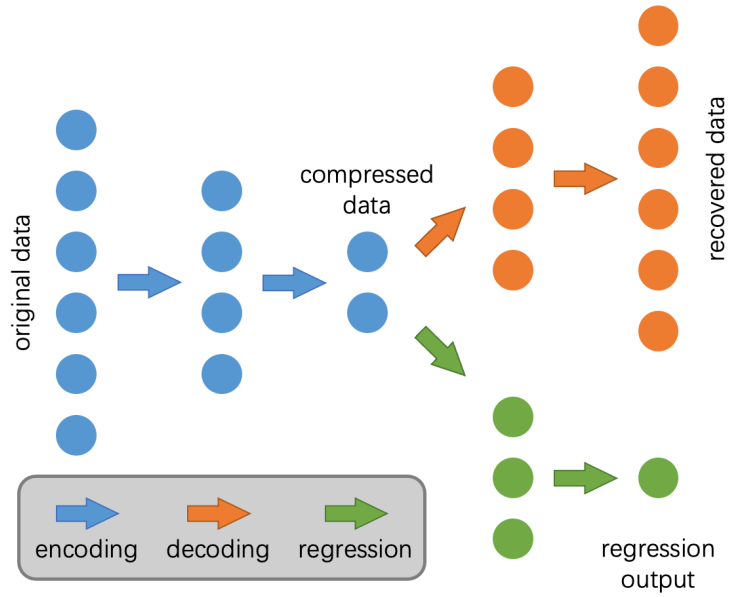
The FreshIndex project has amassed multiple large and meaningful datasets for fruit quality monitoring. However, the arduous process of data labeling has resulted in a considerable amount of unlabeled data in comparison to labeled data within the datasets. To make the most of this data, a regression network utilizing Autoencoder has been proposed. The encoder compresses the input data into a reduced set of features, which are then passed through a decoder to recreate the original input signal. Simultaneously, the encoded features are employed to predict the target data. This approach allows the unlabeled data to aid the encoder, enhancing the encoded features and preventing overfitting. In the experiments, the measurements are the spectrum of the target fruits, which is processed by the autoencoders. Meanwhile, the output data, i.e., the data for regression, are

- weight [g]
- acid [g/l]
- brix PCE [°]
- titration volume [ml]

for **grapes** and

- brix Atago [°]
- V_EP_strongBase
- V_EP_fix7.0
- V_EP_erf+trend

for **berry**. Moreover, for berry, we also tried the encoded features to classify their **cultivar**. In addition, the spectrum for Mangos are also involved in the experiment for the temporal prediction.



Non-Bayesian Network

Figure 1. Neural architecture for non-Bayesian autoencoder and regressor.

Figure 1 shows the exempary architecture of the proposed semisupervised machine learning mode. In this approach, the encoder operates as a feature extractor, which aims to compress the input data into a reduced set of features, whereas the decoder aims to recover a signal from the code through learned patterns. Thus, the quality of signal recovery can be utilized

as an anomaly detector to assess the plausibility of the encoded features. For instance, if an autoencoder fails to compress and recover a signal for grapes, it is plausible that the signal possesses a different pattern than that of a grape. This effect enables us to assess the **confidence level** of the regression by comparing the original signal with the one reconstructed by the decoder. Consequently, this methodology enables us to leverage the unlabeled data by utilizing the autoencoder's ability to learn meaningful features and prevent overfitting. The resulting encoded features can be used for the regression task to improve the performance of fruit monitoring systems.

Network Compression Deploying machine learning models in real-world applications often requires strict adherence to resource constraints. In the case of FreshIndex, the fruit quality detector designed for widespread distribution to numerous retailers must be deployed on inexpensive, resource-limited hardware. On the other hand, in customer-oriented scenarios, the design of the neural architecture of the machine learning model must prioritize quick response time, which is meaningful for customer to obtain the fruit information in their daily shopping. To achieve this, compressing neural networks has become a crucial technique for reducing the size and complexity of machine learning models while maintaining their performance. The compression process involves identifying and eliminating redundant or less important parameters while preserving the most informative features of the network. Essentially, compressing neural networks is a critical step in designing and optimizing machine learning models that can effectively tackle real-world challenges.

Neural Architecture Search for Ultra-Compact Models Our selection of Neural Architecture Search (NAS) among the various available network compression techniques was based on its ability to provide a wider search space and yield superior outcomes in comparison to alternative methods such as quantization. Even though one of the potential limitations of this technique is that it may result in a longer search process due to the vast search space, the advantages that neural structure search offers make it a highly valuable choice in this context.

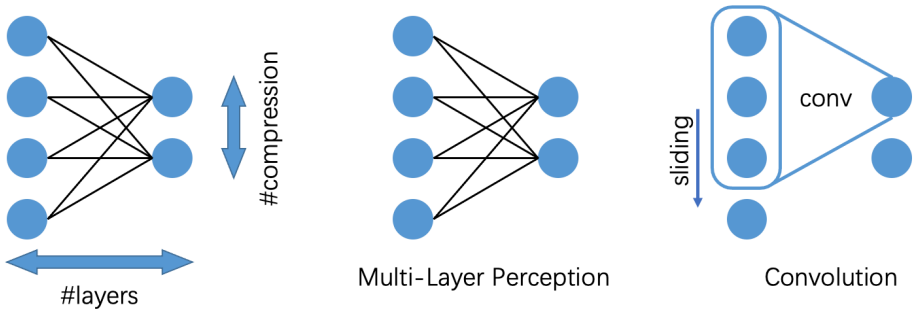


Figure 2. Search space of the Neural Archetecture Search.

In Figure 2, the search space for neural network search is illustrated, highlighting several primary search parameters. These parameters include:

1. the number of layers for the encoder, decoder, and regressor. As each layer of the neural network contains a nonlinear variation, the number of layers often represents the degree of nonlinearity for the problem. Thus, the search or optimization for the number of layers depends more heavily on the degree of nonlinearity for the specific problem.
2. the number of features being compressed is another critical search parameter. If using neural architecture search (NAS), the optimal (i.e., the minimum) number of features to be compressed can be determined. This number often depends on the degrees of

freedom for the target data, where a smaller number of compressible features indicates a lower degree of freedom for the problem.

3. the encoding & decoding mechanism is also an important search parameter. The encoder is responsible for compressing high-dimensional data into low-dimensional data, which can be accomplished in various ways. Depending on the nature of the problem, two dominant compression possibilities exist: fully connected layer and convolutional approach.

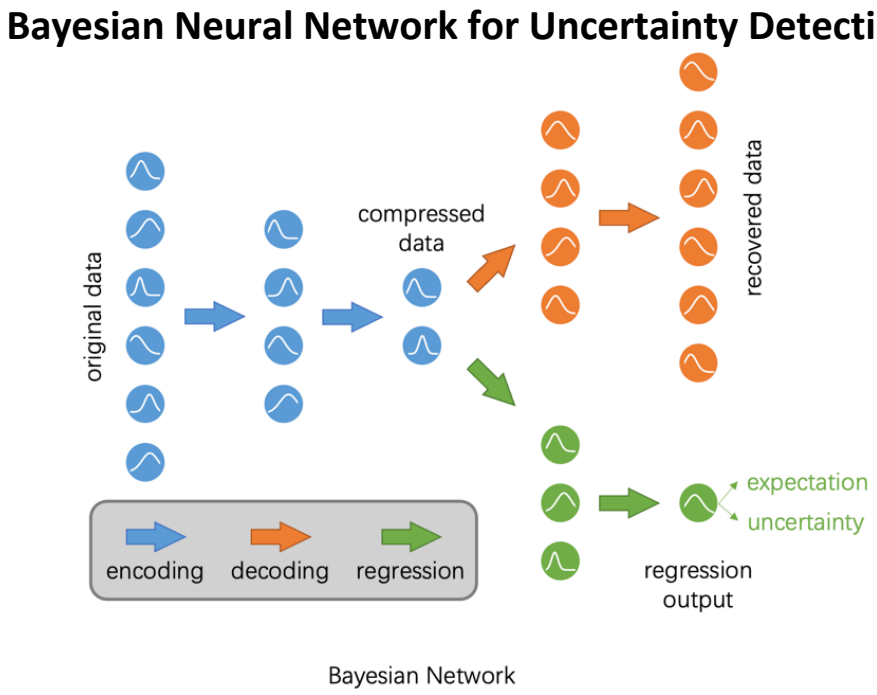


Figure 3. Neural architecture for Bayesian autoencoder and regressor.

The "confidence level" refers to the level of uncertainty in a prediction. Bayesian Neural Networks are a promising approach to address this issue. Unlike traditional neural networks, the parameters in Bayesian Neural Networks are represented by stochastic variables rather than deterministic values. Typically, these parameters are modeled to follow a Gaussian distribution with a mean value μ and standard deviation σ (as can be seen from Figure 3). As a result, the output of each neuron and layer is also stochastic, and the output of the entire network is a stochastic variable. The expected value of the output can be used as the predicted value, while the variation of the output indicates the uncertainty of the prediction. This approach allows us to quantify the "confidence level" or uncertainty of the network.

However, Bayesian Neural Networks are difficult to work with because the resulting distribution after multiple weighted-sum and activation functions has no closed form. To overcome this challenge, the law of large numbers is generally adopted for the numerical approximation. This involves sampling multiple (N) values from each stochastic variable, which enables us to embody the probability on the values of all samples.

In this project, the goal of training the Bayesian Neural Networks is not only to find an output that represents the target value, but also to capture the error of the prediction. This is

achieved by minimizing the difference between the target value and the expected output, as well as the difference between the "prediction error" and the "standard deviation". In other words, the standard deviation is trained to reflect the prediction error.

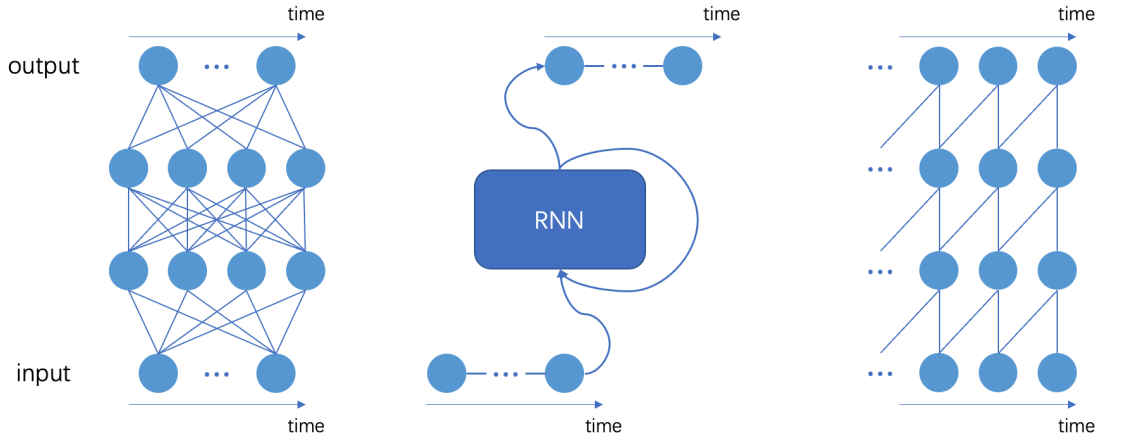


Figure 4. Candidates in temporal information processing.

Temporal Information Prediction

In addition to static spectral data and target regression data, temporal data predictions are also considered in the FreshIndex project. These time-varying data are derived from the spectral information of mangoes measured over multiple days. To simplify the problem, the mango spectra are compressed by an optimized encoder (from neural architecture search) before performing predictions, and the compressed spectral information is used for the prediction task. Specifically, the input of the prediction task is the compressed spectral information at one time stamp (i.e., one day), while the output is the spectral information at the next time stamp (i.e., the following day). To achieve this goal, several machine learning models can be utilized, such as fully connected layers, recurrent neural networks, or temporal convolutional neural networks. Figure 4 represents the mentioned networks that are adoptable for the temporal information processing. Similarly, neural architecture search and hyperparamter tuning are used to optimized the performance of the machine learning models.

Complementery

In addition to the aforementioned technologies, such as neural architecture search, Bayesian networks, etc., a series of rigorous experiments were conducted to enhance the performance of our neural networks. Various approaches were investigated, including the use of different input data, sensors, loss functions, and feature engineering techniques. The extensive experiments including:

1. The effect of different input data on network performance was explored. Specifically, four inputs were tested: raw I/I0, I/I0, A, and A_SNV. The goal was to determine which input provided the most informative features for our networks to learn from, thereby enhancing their ability to accurately predict outcomes.

2. The effect of different sensors on network performance was examined. Two types of sensors, namely near-infrared spectroscopy (NIR) and visible spectroscopy (VIS), were tested with the aim of identifying which sensor was better suited for our task. This experiment was particularly important because the choice of sensor could have a significant impact on the accuracy and robustness of our network.
3. Different loss functions were evaluated to guide the training of our networks. The performance of two commonly used loss functions, mean squared error (MSE) loss and coefficient of determination (R2) loss, was compared. The aim was to determine which loss function was better suited for our task and which could lead to a more accurate and robust network.
4. Feature engineering techniques were implemented to improve the performance of our networks. Specifically, the inverse Fourier Transformation was utilized, which may allow more meaningful and informative features to be extracted from the input data, leading to better network performance.
5. Multiple runs of the experiments using different random seeds, ranging from 1 to 10, were conducted to ensure the validity and reliability of the results. By conducting these experiments, the most effective approaches to enhance the performance of the networks were identified, and ultimately, our research goals were achieved.

Summary of all the optimizations

In summary, to achieve machine learning models that are optimized for use in embedded systems and offer high performance, we have optimized the following design parameters that might be critical for the project:

1. Tuning the learning rate, which is an important factor that can greatly affect the performance of a machine learning model.
2. Adjusting the number of layers for the encoder, decoder, regressor, or classifier, which can significantly impact the model's ability to capture complex patterns in the data.
3. Modifying the number of encoded features, which can have a direct effect on the model's predictive power.
4. Selecting the type of sensors to be used, which can impact the quality and type of data that is available to the model.
5. Choosing the type of input data to be used, which is a critical factor that can greatly impact the performance of a machine learning model.
6. Deciding whether to use feature engineering or not, which can be a powerful tool for improving the performance of a machine learning model.
7. Selecting either MSE or R2 loss as the loss function, which can help to measure the performance of the model and guide its optimization.
8. Choosing between a Bayesian approach or a nominal approach, which can impact the overall complexity of the model and its ability to generalize to new data.

By carefully optimizing these hyperparameters and factors, the optimal choice can be guaranteed to a great extent.

Results

This section provides a summary of the results obtained from the multiple optimization processes, as well as a comparison with the baseline method previously utilized, namely the partial least squares (PLS) method.

Grape

After conducting a comprehensive neural architecture search using hardware-aware neural networks, we ultimately opted for a fully connected Bayesian encoder, decoder, and regressor consisting of **two layers and three compressed features** to analyze the grape data. This choice not only provides an exceptionally fast inference speed, but also yields optimal outcomes. The regression results for the 4 objectives are presented in Table 1, indicating substantial improvements over traditional PLS techniques in terms of machine learning results. Additionally, the elimination of 10% of outliers through anomaly detection further enhances the regression results.

Figure 6 and 7 indicate the comparison of the regression from PS, ML, and ML with cleaned data.

Regression target	PLS		ML (with CV)		ML		ML (with clean)	
	R^2	θ	R^2	θ	R^2	θ	R^2	θ
weight [g]	0.36	0.17	0.45	0.16	0.66	0.13	0.79	0.10
acid [g/l]	0.36	0.16	0.38	0.16	0.54	0.14	0.67	0.11
brix PCE [°]	0.76	0.09	0.78	0.08	0.89	0.07	0.91	0.06
titration [ml]	0.31	0.18	0.36	0.16	0.59	0.15	0.60	0.12

Table 1. R^2 and θ of the regression performance from multiple methods.

Barry

During our investigation of berries, we conducted a cultivar identification classification task alongside regression analysis, which is analogous to grapes. Through an extensive exploration of neural network architectures, we obtained an optimized, fully connected Bayesian encoder, decoder, and classifier with eight compressed features, and each sub-model has two layers. Our classifier demonstrates noteworthy accuracy, achieving 88% and 73% on the training and test sets, respectively, across six distinct berry cultivars.

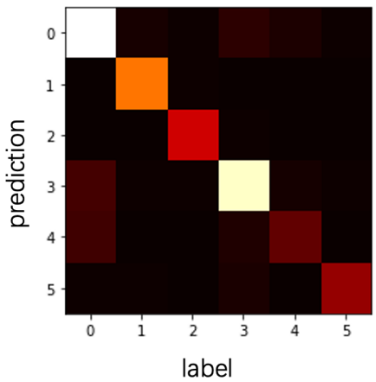


Figure 5. Heatmap of the confusion matrix of the classification.

Figure 5 analysis the result of the classification. It is evident that the majority of the data points have been correctly assigned to their respective classes. However, there exists a slight probability that class 0 may have been misclassified as class 3 or 4. We postulate that this could be due to the intrinsic spectral similarities shared between class 0 and classes 3 and 4.

Mongo

In our analysis of mango data, we shift our principal objective on temporal prediction. However, the original data comprises high spectral dimensions, totaling 126. This high dimensionality poses a significant challenge in terms of the complexity of the prediction task. Inspired by the success in grape and berry datasets, we propose to compress the spectral information of mangoes. Specifically, we posit that a suitable encoder can be employed to reduce the dimensionality of the data to a lower space, facilitating the prediction task. This compressed space can then be utilized for the temporal prediction task, streamlining the process, and improving the efficiency and effectiveness of the overall analysis. After employing the neural architecture search, a **2-layer** autoencoder with **10 compressed features** was found to optimally compress the spectral information of mangoes.

The other challenge of this task is the non-periodic and irreversible nature of fruit ripening, which makes it difficult to identify regularities in fruit properties over time without aligning the degree of ripening of each fruit w.r.t. ripening time. However, it is challenging to track the ripening or even picking time of individual fruits. To address this issue, we grouped mangoes in batches by their deliveries and assumed that, the mangoes in the same delivery might share similar degrees of ripeness. We then further reduced individual differences and ripeness variations within the batch by averaging their corresponding compressed features to create ripening curves. Upon analyzing Figure 6 (left), it is evident that the adoption of machine learning-based data preprocessing techniques renders the tendency of each feature more stable and robust over time. This approach facilitates accurate prediction of temporal information, mitigating the impact of inherent data variation. Figure 6 (right) shows an example of feature prediction of the 2nd compressed feature.

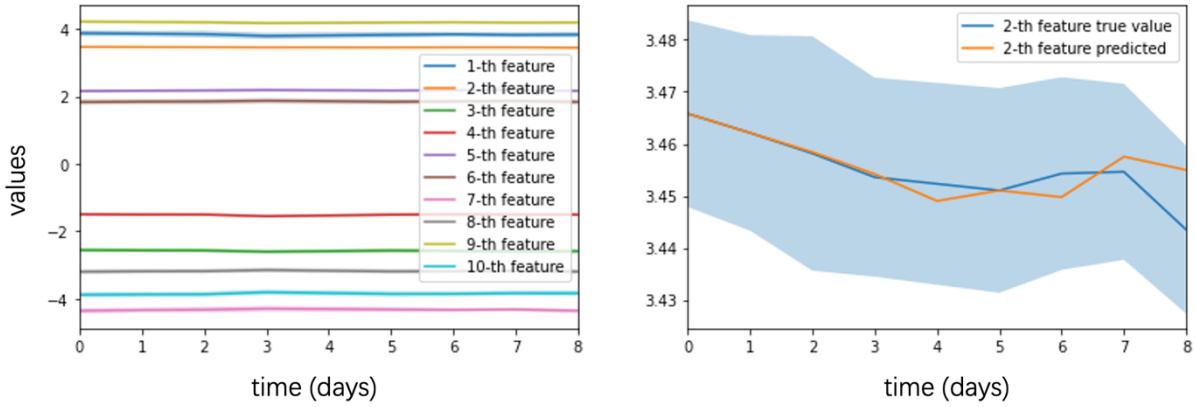


Figure 6. Left: mean (valid curves) and standard deviation (shadows) of compressed mango data with 10 features in 9 continuous days. Right: details of the 2nd feature as an example of prediction.

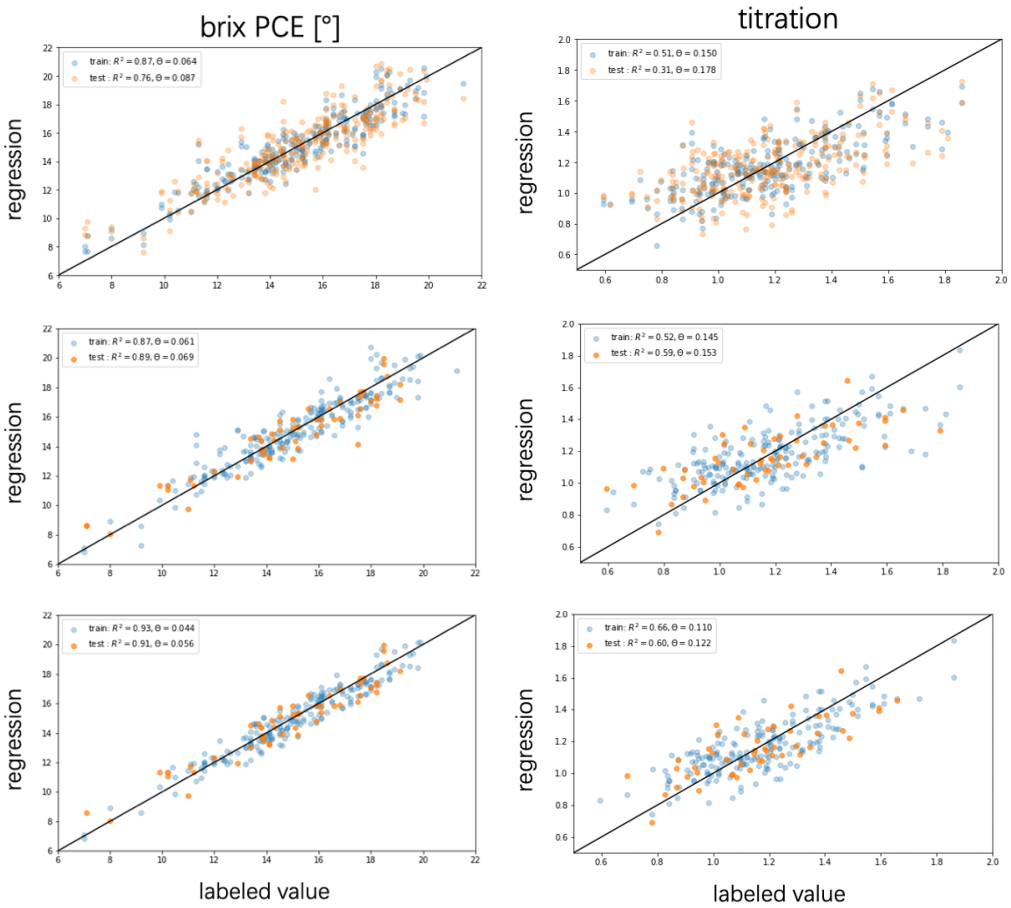


Figure 7. Comparison of different regressors for brix (left) and for titration (right), top for PLS, middle for ML, and bottom for ML with cleaned data.

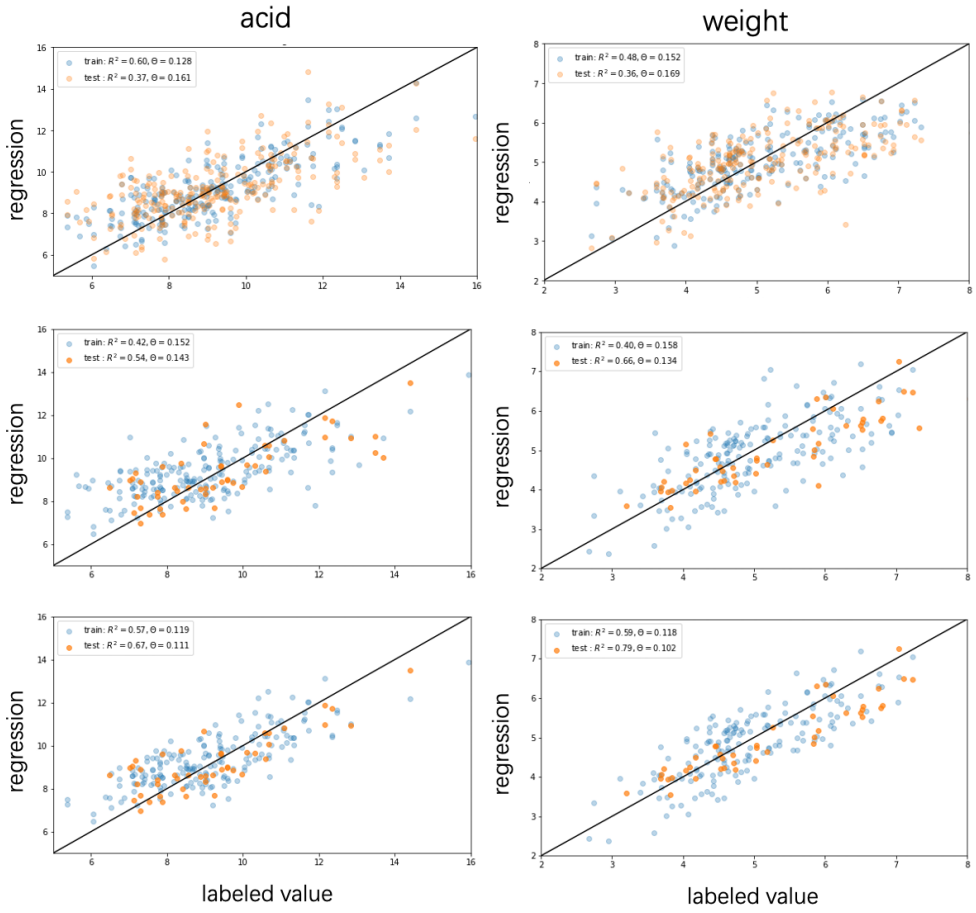


Figure 8. Comparison of different regressors for acid (left) and for weight (right), top for PLS, middle for ML, and bottom for ML with cleaned data.