# Interpolate and Extrapolate Machine Learning Models using An Unsupervised Method – An Approach for 2023 PHM North America Data Challenge

Peng Liu

*JMP Statistical Discovery LLC, Cary, NC, 27513, USA*
*Peng.Liu@jmp.com*

## ABSTRACT

The 2023 PHM North America Data Challenge is intriguing because it requires one to predict outcomes and use data patterns that training models do not see. Modern machine learning models based on gradient boosting and neural networks are not designed to address such issues in usually circumstances. Our final approach to address the challenge consists of five steps. In our approach, we use an unsupervised method besides machine learning models to address the challenge.

## 1. DETERMINE THE TYPE OF THE PROBLEM

The training data consists of multi-dimensional vibration time series of 2016 samples from gear pitting experiments. The testing data set has 800 samples, and the validation data set has 813 samples.

The testing subjects have the same gear parameters, but are under different pitting conditions, and tested under different settings. Pitting condition, also known as state in the data, is an ordinal response variable, which describes the degradation level of individual testing subjects. Testing settings consist of various levels of speed and torque combinations. The values are discrete numeric. The training data consists of seven pitting conditions and 78 testing settings. The testing and validation data consist of additional four pitting conditions and additional 15 testing settings. The task is to predict pitting conditions of samples in the testing and validation data.

We consider the problem as a classification modeling problem in general, but with additional requirement to predict unseen labels. Normally, classification modeling treats response as a categorical variable. For such models, interpolation and extrapolation are undefined.

Peng Liu. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 United States License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

The decision on the type of the problem is determined by the final methodology that we decide to use. Our choices of methodologies are two modern machine learning techniques: tree-based gradient boosting and neural networks. Both techniques are high performers in predictive modeling, given sufficient training data. But they are known to be unsuitable for interpolation and extrapolation in regular use cases.

Our final strategy is to use training data to fit a typical classification model, using either methodology, then use the fitted model to predict outcomes of new data. The predictions are then used to interpolate and extrapolate to unseen labels.

## 2. DETERMINE A CROSS-VALIDATION METHOD

First, we need to clarify some terminologies for the remaining discussion. Cross-validation is a framework to train modern machine learning models. It requires one to partition existing data for modeling in two pieces. In some literature, the two pieces are called training and validation sets respectively. The names, however, are confounded with three data sets in the competition. In the competition, we have three data sets: “training”, “test”, and “validation”. The data that we use for modeling is the “training”. For cross-validation, we need to partition the “training” in two pieces. To avoid confusion, we use “training-partition” and “validation-partition” for the partitions during cross-validation. Next, we describe how we apply a cross-validation strategy to model this data.

We notice the data set is huge, but the number of independent samples is small. Eventually, we decide to split long time series into shorter ones. But by such, we do not create more samples, and we need to avoid leaks. Therefore, when we train our model, we create cross-validation set based on the original sample identifications, not based on the shorter time series.

Such cross-validation faces a challenge in the current task because there are not enough data in the training data. To properly creating cross-validation sets, we need to consider the distinct level combinations of state, speed, and torque. In the end, there are not enough data for every combination to be split into training-partition and validation-partition. There are less than four samples on average for each testing setting. For example, a five-fold cross-validation split does not have enough data, because a five-fold cross-validation needs to put four fifth of samples in the training-partition, and one fifth of samples in the validation-partition. The partitions are iterated to have even opportunity to be either training-partition or validation-partition. But we still don't feel that one five-fold cross-validation set is sufficient. To address the problem, we use the repeated five-fold cross-validation, which creates multiple cross-validation sets. By such, each combination will have even opportunities to appear among all training-partitions and validation-partitions. A repeated five-fold cross-validation is an extension to simple five-fold cross-validation. Instead of creating one five-fold cross-validation split, the repeated version creates multiple random splits. We use a patented method which provides a rigorous approach to create such partitions that have the optimal properties such as independence, randomness, and balance. The method (Lekivetz, et.al. 2020) is implemented in the proprietary software JMP.

## 3. CREATE A DATA AND FIT A MODEL

We define our classification model as follows. The response variable is the state, which is the degradation condition. The predictors are features from time series. We only use the first three time series in each sample. From each sample, we split the long time series of each sample into shorter time series segment with length 1000, discarding irregular or redundant beginnings of the long series. A segment is still a multivariate time series. Time series in each segment are then normalized by centering by mean and scaling by standard deviation. From the processed segment, we extract features, which include univariate summary statistics of univariate time series, univariate time series features such as autocorrelation functions, multivariate summary statistics such as correlation, and multivariate time series features such as cross-correlation. We use extracted features and testing settings as predictors, together with the response state, we form a data for model fitting. The testing settings are treated as continuous variables. The data has about a quarter million rows and over five hundred columns.

We experimented two types of models: gradient boosting and multilayer perceptron neural networks. Their performances are similar. Our best performing model on the test data is a gradient boosting model. Gradient boosting is a general technique developed by Friedman (2001). The specific implementation that we use is known as the XGBoost developed by Chen (2016). XGBoost is usually used through either R or Python. We use XGBoost through a software developed by Wolfinger (2020).

To use XGBoost, one usually needs to tune hyperparameters to achieve the best performance. For this task, we did not do so but accept all default settings, except increasing the number of iterations. The reason is, given the default settings, the fitted model performs very well on the validation-partitions, which is independent of training-partitions for independent assessment of the model performance. The other reason that we did not put effort in tuning is the nature of this competition which requires predicting unseen labels. We anticipate that the uncertainty due to that nature will outpace improvement by hyperparameter tuning within the framework of our approach.

Our final model on the validation data is a multilayer perceptron neural networks model. We use the software PyTorch by Paszke et. al. (2019) through a developing interface in JMP (2023). Similarly, we did not spend effort on tuning hyperparameters, but choose an epoch value so the performance of the model on the validation-partition appears to converge and does not deteriorate. The main reason to use this method is because the speed of model fitting is much faster than fitting XGBoost models, while cross-validation performances are similar.

## 4. PRODUCE PREDICTIONS

The prediction outputs of our cross-validated classification models are probabilities of observed seven states, from individual short time series. Recall the short time series are segments of length 1000 from longer series, so we calculate the average of the probabilities by the original sample identifications. The results are averaged probabilities of seven possible states. For example, there are 800 independent samples in the testing data. The predictions on the testing data consist of 800 vectors of length 7, each entry of a vector is an averaged probability of being in the corresponding state Meanwhile, we have 816 vectors for the validation data.

## 5. PREDICTIONS TO UNSEEN LABELS

There are four states that are unseen in the training set and we need to predict them in the testing and validation sets. During our research, we use the training data to mimic the situation by intentionally excluding samples of a couple states and predict their outcomes by pretending they haven't been seen. And we used the learned knowledge to decide our extrapolation strategy.

We use the nearest neighbor clustering method, which is an unsupervised learning approach, to predict unseen labels. For example, the training data has 7 states. We exclude samples whose state is 3 from model fitting process. The fitted model is trained to predict just 6 states. The prediction is a vector of 6 entries, as we described in the previous section. We then apply the nearest neighbor clustering to cluster all predicted vectors into 7 clusters, which is the true number of levels of states. We use this approach to learn how the clustering result behaves to seen and unseen labels.

Figure 1 illustrates the result from such an experiment. The model was training by excluding samples whose state is 3.

The training process does not see samples whose state is 3. The predictions are done on the validation-partitions and the excluded samples whose state is 3. The predictions are clustered into 7 clusters. The plot in Figure 1 is known as the parallel coordinate plot. There are 7 sub-plots for 7 clusters. Each sub-plot has a cluster ID beneath it. In each sub-plot, a connected line draws the prediction vector of 6 entries. For example, in cluster 1, a red connect line connects 6 estimated probabilities values of predicting which state the corresponding sample is at. In cluster 1, the probability of state=0 remain highest for all samples in that cluster.

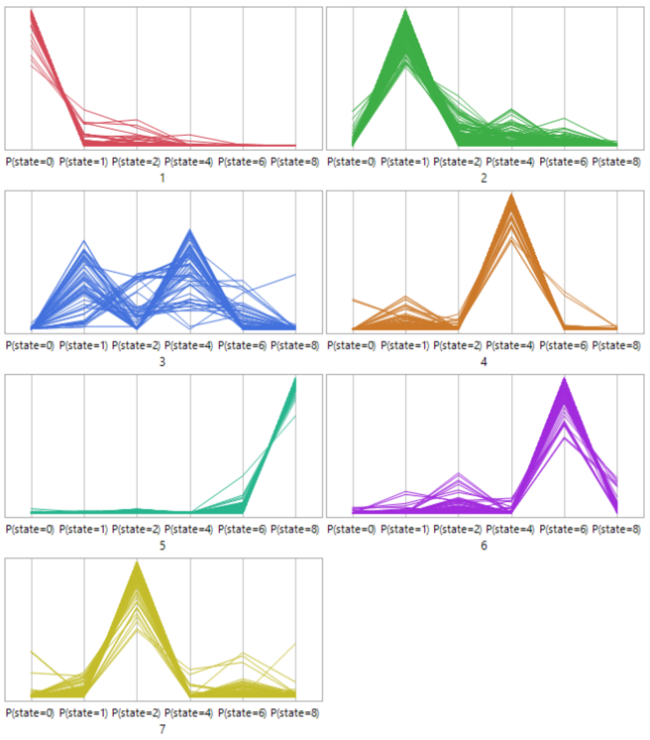

Figure 1. Clustering result with state=3 excluded.

Similarly, we can observe that samples in cluster 2 have highest probability prediction at state=1. And cluster 4, 5, 6, 7 have their clear highest probability prediction, corresponding to states 4, 8, 6, 2, respectively. Samples in cluster 3 appear to have multiple peaks. It is tempting to assign state = 3 to that cluster. If so, it seems that the prediction of the unseen label state = 3 spreads probabilities in the neighborhood of state = 3, such as 1, 2, 4, and 6. There is very little probability on state = 0 or 8, which are further away from state 3.

If we predict that samples in clusters 1, 2, 4, 5, 6, 7 are from state 0, 1, 4, 8, 6, 2, respectively, and samples in cluster 3 are from state 3, the accuracy can be seen from Table 1. The table tabulates the actual states of individual samples and their predicted states by clustering. The row labels are the actual states. The column titles are predicted states. The values in individual cells are the counts that are associated with corresponding row label and column title. For example, the value 287 in the upper left corner of the table represents the number of samples from state 0 and also correctly predicted. Another example, the cell with value 4 on the row of state = 3, under the column title where "state by cluster = 0", is the count of misclassification of 4 samples of state = 3 to state = 0.

Table 1. Accuracy of interpolation.

| | **state by cluster** | | | | | | |
|---|---|---|---|---|---|---|---|
| **state** | **0** | **1** | **2** | **3** | **4** | **6** | **8** |
| **0** | 287 | 0 | 0 | 0 | 0 | 0 | 0 |
| **1** | 0 | 294 | 1 | 0 | 0 | 0 | 0 |
| **2** | 0 | 0 | 291 | 0 | 0 | 0 | 0 |
| **3** | 4 | 114 | 5 | 59 | 85 | 0 | 0 |
| **4** | 0 | 0 | 0 | 4 | 300 | 0 | 0 |
| **6** | 0 | 0 | 0 | 0 | 0 | 276 | 0 |
| **8** | 0 | 0 | 1 | 1 | 0 | 0 | 294 |

From Table 1, we learned that assigning clusters with clear peaks to the corresponding states is a reasonable way to make prediction of seen labels. But probably due to unseen labels, there are misclassifications. For example, in the second column in Table 1, under the column title "state by cluster=1", there are 114 misclassified samples from "state = 3". The samples under "state by cluster = 4" include 85 misclassified samples from "state = 3".

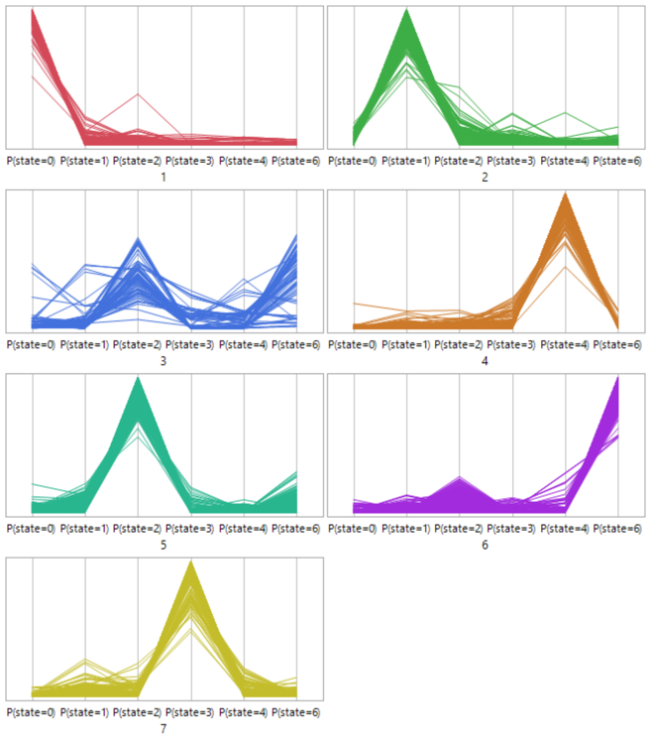

Figure 2. Clustering result with state=8 excluded.

Similarly, we studied the approach by excluding state = 8. And Figure 2 is the parallel coordinate plot of such a study. In this study, we assign predictions to individual clusters as follows. Assign cluster 1 to state = 0, cluster 2 to state = 1, cluster 4 to state = 4, cluster 5 to state = 2, cluster 6 to state = 6, and cluster 7 to state = 3. The assignment is based on the peak probabilities in individual sub-plots. The samples in cluster 3 are not assigned. Because we do not know how to handle the pattern in the study. We assign state = -1 to indicate that we are unable to assign a valid prediction. Table 2 is the tabulated misclassification table.

Table 2. Accuracy of extrapolation.

| | **state by cluster** | | | | | | |
|---|---|---|---|---|---|---|---|
| **state** | **-1** | **0** | **1** | **2** | **3** | **4** | **6** |
| **0** | 2 | 285 | 0 | 0 | 0 | 0 | 0 |
| **1** | 2 | 4 | 288 | 1 | 0 | 0 | 0 |
| **2** | 4 | 0 | 0 | 287 | 0 | 0 | 0 |
| **3** | 0 | 0 | 1 | 0 | 265 | 1 | 0 |
| **4** | 0 | 0 | 0 | 0 | 2 | 302 | 0 |
| **6** | 3 | 0 | 0 | 0 | 0 | 1 | 272 |
| **8** | 89 | 1 | 0 | 58 | 0 | 0 | 148 |

From Table 2, the unseen label state = 8 is misclassified into the state = -1, 2, and 6. But many congregate at state = 6. From what we learned, we can conclude that our approach to make prediction perform well on seen labels, but there are uncertainties on unseen labels. For unseen labels that require interpolation, the predicted probabilities appear to be around the truth. For the unseen labels that require extrapolation, a large number of samples congregate at the highest observed state.

Based on what we learned in the experiments, we tested different decision rules to assign clusters to predicted states. The main strategy is to decide threshold probabilities in individual clusters, such that the samples with predictions above the thresholds are assigned to corresponding predicted states. The ones under the thresholds are undetermined. The cluster that has the peak probabilities at state = 6 has two thresholds. The samples above the upper threshold are assigned to state = 6. The samples below the lower threshold are undetermined. The ones in the middle are assigned to state = 9. All the undetermined samples are assigned zero probabilities to all prediction entries and zero confidence. The process of finding the thresholds is a trial-and-error process by monitoring our scores on the leader board. We choose the result that had the highest score as our final result.

## 6. CONCLUSION AND FUTURE WORK

The data challenge is intriguing because it requires predicting unseen labels. Modern and power machine learning models are not designed for such tasks. Our approach combines modern machine learning models and an unsupervised method to address the challenge. The result is promising, but we also see a lot of room for improvement. We can think of at least following two directions to study further. The first one is how to make short time series segment from the longer time series based on the individual testing conditions. We use an arbitrary length of 1000 in our modeling process, but we think that different segmentation rules may result different results. The second direction is how to conduct the unsupervised learning more objectively. Maybe, there is a less arbitrary way.

## REFERENCES

Lekivetz, R.A., Morgan, J.A., Jones, B.A. and Wolfinger, R.D. (2020). Validation Sets for Machine Learning Algorithms. U.S. Patent 10,754,764. SAS Institute Inc.

Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5) 1189-1232

Chen, T., Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. arXiv: Learning, 785-794. Retrieved 3 27, 2023, from https://arxiv.org/abs/1603.02754

Wolfinger, R. D. (2020). XGBoost Add-In for JMP Pro. https://community.jmp.com/t5/JMP-Add-Ins/XGBoost-Add-In-for-JMP-Pro/ta-p/319383

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Chintala, S. (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*.

JMP (2023), Version 18. SAS Institute Inc., Cary, NC, 1989–2023.

## BIOGRAPHY

**Peng Liu** is a Principal Research Statistician Developer at JMP Statistical Discovery LLC. He holds a PhD in statistics from North Carolina State University. He is responsible for maintaining and developing products related to reliability data analysis, reliability engineering, time series, and time series forecasting. His work includes research in algorithm development, data analysis, graphical user interface design and implementation, and software architecting.