Lawrence Berkeley National Laboratory
LBL Publications

Title
Data Driven Dimensionality Reduction to Improve Modeling Performance∗

Permalink
https://escholarship.org/uc/item/0555v6rb

Authors
Chung, Joshua
De Prado, Marcos Lopez
Simon, Horst
et al.

Publication Date
2023-07-10

DOI
10.1145/3603719.3603744

Peer reviewed

eScholarship.org Powered by the California Digital Library
University of California

Chapter 1

Lessons on Hyperparameter Optimization from Data Driven Dimensionality Reduction

Joshua Chung*, Marcos Lopez de Prado*†‡, Horst D Simon†, and Kesheng Wu*

Machine learning algorithms excel at optimizing model parameters for intricate problem-solving. However, they rely on users to provide hyperparameters, which significantly impact performance. To avoid manual trial-and-error, automatic hyperparameter optimization (HPO) methods have been developed. Our recent exploration in dimensionality reduction reveals that HPO success depends not only on effective algorithms but also on a precisely defined objective function and robust parallelization strategies. Despite these efforts, capturing the essence of the application problem in the resultant model remains uncertain. In this paper, we provide a concise review of HPO algorithms and parallelization methods. Additionally, we share insights from our quest to establish a reliable quality measure for dimensionality reduction. Our findings emphasize the critical interplay between optimization algorithms, objective functions, and resource utilization strategies. Furthermore, we highlight the pressing need for automated detection of potential pitfalls in models forged through extensive hyperparameter optimization.

1.1 Introduction

Machine learning and data analysis are essentially optimization procedures, finely tuning model parameters to construct an effective model for specific problems [Carbonell *et al.* (1983); Jordan and Mitchell (2015); Mahesh

*Lawrence Berkeley National Lab, Berkeley CA, USA
†ADIA Lab, Abu Dhabi, UAE
‡ADIA, Abu Dhabi, UAE

(2020)]. In addition to these model parameters, there are hyperparameters that define the structure of the model, learning process, analysis algorithm, an so on [Bischl *et al.* (2023); Karl *et al.* (2023); Morales-Hernández *et al.* (2023)]. For instance, in training a regression model with a large neural network, trillions of model parameters might be adjusted to minimize an objective function like regression accuracy, while the shape of a neural network and the details of the learning process are defined by hyperparameters, numbering in dozens or more, must be provided. Given their influential role, selecting these hyperparameters requires meticulous consideration. Hyperparameter optimization (HPO), the intricate process of their selection, has garnered considerable attention [Diaz *et al.* (2017); Li *et al.* (2017); Yang and Shami (2020)]. While existing literature focuses on algorithms and tools, this work diverges by providing practical insights from our machine learning and feature extraction experiences [Chung *et al.* (2023); Lopez de Prado (2018a); Zhan *et al.* (2018)]. Our aim is to offer tangible guidance, highlighting potential pitfalls in real-world HPO applications.

For our discussions to be concrete, we will review the a specific hyperparameter optimization exercise to study its pros and cons [Chung *et al.* (2023)]. We then broaden the discussion to encompass a broad spectrum of feature engineering and machine learning tasks. The specific HPO exercise was to study a dimensionality reduction tasks on a set of obfuscated data named Numer.ai data [Numerai (2024)]. Dimensionality reduction involves extracting a limited set of features from high-dimensional data — a crucial step in data preparation for further analyses [Jia *et al.* (2022); Zebari *et al.* (2020)]. Given the array of algorithms and implementations available, choosing the right algorithm and its parameters poses a considerable challenge. Even with guidelines, such as those provided by [Nguyen and Holmes (2019)], the intricate choice often rests with users. Automating this selection has been very effective in simplifying the hyperparameter selection, but introduces its own challenges [Cooper *et al.* (2021); Eggensperger *et al.* (2019); Jacobs *et al.* (2022); Vento and Fanfarillo (2019)].

Taking the cost of evaluating the learning procedure in the above regression example as a unit of computational cost and call it an *evaluation* (or a *full evaluation* for emphasis), a brute-force evaluation of millions of different hyperparameter combinations (also known as hyperparameter configurations, or simply *configurations*) would require millions of *evaluations*, which would be very time-consuming. A state-of-the-art HPO procedure could examine millions configurations with the cost equivalent to several *full evaluations*, which makes HPO procedures high useful for large-scale data analysis tasks [Brandt *et al.* (2024); Falkner *et al.* (2018)]. However, the research community has also noticed serious pitfalls in these automatically optimized models [Cooper *et al.* (2021); Eggensperger *et al.* (2019); Jacobs *et al.* (2022); Vento and Fanfarillo (2019)]. Inspired by works like Nguyen and Holmes [Nguyen and Holmes (2019)], we share insights from our experiences with hyperparameter optimization [Chung *et al.* (2023); Lopez de Prado (2018b); Jacquier *et al.* (2022)]. The following is a quick summary of top two lessons.

Our first crucial observation emphasizes the importance of understanding the data thoroughly. This aligns with Tips 2, 3, 4, and 9 from [Nguyen and Holmes (2019)]. Knowing the data involves more than just handling different value types; it requires nuanced considerations, such as treating seemingly numeric values as categorical in specific contexts [Guo and Berkhahn (2016); Seger (2018)]. For scientific applications, incorporating known physics governing feature relationships can significantly enhance modeling efforts [Cuomo *et al.* (2022); Karniadakis *et al.* (2021); Meng *et al.* (2022)]. Proceeding to automated parameter tuning should follow a careful consideration of domain-specific information. Even though the automated HPO procedures are powerful optimization tools, selecting the right objective function for the optimizer is closely linked to the understanding of the data and context of the analysis task.

Another key observation is to test the analysis results for common pitfalls. Given the complexity of the potential traps and pitfalls, many of them could not be easily identified by typical user, therefore, it is necessary to engage automated tools [Jacobs *et al.* (2022); Park and Ho (2021); Webster *et al.* (2019); Werpachowski *et al.* (2019)]. There is no guarantee that these tools will capture all hidden problems, nevertheless, we strongly believe these tools are effective in identifying many problems involving overfitting [Webster *et al.* (2019); Werpachowski *et al.* (2019)] and underspecification [Jacobs *et al.* (2022)].

1.2 Review of Dimensionality Reduction (DR)

This section provides a short review of the dimensionality reduction algorithms and our experience of optimizing hyperparameters with two sets of different types of data [Chung *et al.* (2023)].

Feature Extraction	
Principal Component Analysis (PCA) [Maćkiewicz and Ratajczak (1993)]	Sci-kit Learn
Kernel PCA (KPCA) [Scholkopf *et al.* (1997)]	Sci-kit Learn
Locally Linear Embedding (LLE) [Roweis and Saul (2000)]	Sci-kit Learn
Isomap [Tenenbaum *et al.* (2000)]	Sci-kit Learn
UMAP [McInnes and Healy (2020)]	UMAP Learn
Feature Selection	
Mean Decrease Accuracy (MDA) [Han *et al.* (2016)]	Custom Code inspired by MLFin Lab
Shaply Values (SHAP) [Lundberg and Lee (2017)]	SHAP Values
Feature Clustering	
Variation of Information [Lopez de Prado (2020)]	MLFin Lab + Custom Code
Maximal Correlation [Lopez de Prado (2020)]	MLFin Lab + Custom Code
Distance Correlation [Lopez de Prado (2020)]	MLFin Lab + Custom Code

1.2.1 *DR Algorithms and the Need for Hyperparameter Optimization*

Table 1.1 shows three classes of dimensionality reduction algorithms examined by [Chung *et al.* (2023)]. Among the three classes, the first two, feature selection and feature extraction are well established in the literature [Anowar *et al.* (2021); Ayesha *et al.* (2020); Maaten *et al.* (2009); Reddy *et al.* (2020)]. [Chung *et al.* (2023)] introduced the third class named Feature Clustering to combined key features from the previous two classes. Feature Clustering techniques could be thought of as a generalization of a method known as Clustered MDA [Lopez de Prado (2020, 2018b)]. The class extends Clustered MDA in three ways: (1) any clustering method could be used for the clustering step, (2) any procedure to determine feature importance could be used to replace MDA, and (3) many feature extraction methods could be used to reduce the clusters into a lower-dimensional representation. In particular, the last extension allows this type of method to switch between feature selection and feature extraction seamlessly, which is the main motivation to give them a new name.

In general, a dimensionality reduction algorithm has far fewer hyperparameters than a deep learning method [Bischl *et al.* (2023); Brigato *et al.* (2021); Yang and Shami (2020)]. However, there might still be too many parameter combinations to explore by hand. An example involving Kernel PCA might has a half a dozen hyperparameters, but since several of the hyperparameters could take on any arbitrary integers, the total number number of parameter combinations could be easily reach thousands. [Chung *et al.* (2023)] showed that the different parameter combinations give rise to results with very different quality measures, therefore, it is critical to carefully explore the hyperparameter choices. Since a deep learning method typically has a lot more hyperparameters, with even more combinations to consider, it is even more necessary to carefully tune the hyperparameters [Akiba *et al.* (2019); Brigato *et al.* (2021); Waring *et al.* (2020)].

1.2.2 *Hyperparameter Optimization Algorithms*

One key challenge in hyperparameter optimization is that the relationship between a hyperparameter and the ultimate quality metric for the solution is often very complex, for example, how the number layers of a neural network affects a loss function is not clearly understood [Kingma and Ba (2017); Yang and Shami (2020); Yu and Zhu (2020)]. The hyperparameter optimization problem for a dimensionality reduction task could be simpler in some sense, but it is similarly complex. This complexity typically include: (1) the relationship between the features (i.e., the hyperparameters) to be optimized and the ultimate objective function is complex, generally, not-differentiable and not continuous; (2) some of the hyperparameters are non-numerical values; (3) the hyperparameters might be highly correlated; (4) the hyperparameter choices might be subject to many constraints. The commonly used optimization procedure is based on the multi-armed bandit approach with variations in several aspects, which we will touch on the top three [Diaz *et al.* (2017); Li *et al.* (2017); Kandasamy *et al.* (2017); Karl *et al.* (2023)]. (1) How to generate the hyperparameter combinations to examine, which commonly described as exploration-exploitation trade-off [Jamieson and Talwalkar (2016); Karnin *et al.* (2013)]. (2) How to reduce the cost of evaluating non-promising cases [Li *et al.* (2017, 2020)]. And, (3) how to parallelize the execution to make the best uses of the available parallel computing systems [Li *et al.* (2020); Meister *et al.* (2020); Gonzalez and Zavala (2023)]

There are a number of software tools for optimization hyperparameters of neural networks [Akiba *et al.* (2019); Feurer *et al.* (2015); He *et al.* (2021); Lindauer *et al.* (2022)]. In [Chung *et al.* (2023)], the Latin hypercube sampleing was used to generate hyperparameter combinations [Deutsch and Deutsch (2012)], a custom version of hyperband was developed in python [Li *et al.* (2017)], and Dask was used for parallelizing the evaluation of each hyperparameter combinations [Rocklin (2015)]. This approach has a relatively static set of hyperparameter configurations to examine at each stage of the optimization algorithm, which simplify the selection of which configurations to go into the next stage where more computational resources will be devoted to examining each configuration. For moderate-sized test cases, this is an effective approach. For larger problems, there are approaches that are more appropriate for making use of a large number of CPUs and CPUs with more asynchronous selection approaches [Bottou *et al.* (2018); Li *et al.* (2020); Meister *et al.* (2020)] and more sophisticated estimation of the quality of each configuration [Karl *et al.* (2023); Wu *et al.* (2019); Yang and Shami (2020)].

1.2.3 *Model Selection, Metrics, and Evaluation*

A dimensionality reduction task is usually used as a pre-processing step for another data analysis step, for example, a regression model [Mease and Wyner (2008)] or a classification model [Yu *et al.* (2023); Yoon (2021)]. For simple DR models, we might be able to rely on residual and related error measures to directly measure the effectiveness of the reduced models [Maćkiewicz and Ratajczak (1993); Scholkopf *et al.* (1997)], however, for more complex techniques, such error measures are no long appropriate [Anowar *et al.* (2021); Jia *et al.* (2022); Ray *et al.* (2021); Zebari *et al.* (2020)]. For example, for the Numer.ai contest, the preferred quality measure is the Spearman Rank Correlation (SRC), which is completely different from any Euclidean distance measure [Numerai (2024); Zar (1972)]. In tests conducted by [Chung *et al.* (2023)], a random forest based regression model is built as an example data analysis task for the dimensionality reduction results. This choice was made because a random forest is a relatively inexpensive procedure to build a useful model for original task motivated the study. Since this regression model building procedure also has its own hyperparameters, it only make sense for the overall hyperparameter optimization process to consider these hyperparameters in addition to the ones for DR algorithms.

Random Forests Regressors Next, we expand the discussion on choosing the regression model for data analysis task.

The original study [Chung *et al.* (2023)] was motivated by Numer.ai contest [Numerai (2024)], where each feature of the raw data given is quantized to five levels only. Such coarse quantization was partly to preserve the privacy of the data involved and partly to remove known dynamics from the contest data [Khanan *et al.* (2019); Narayanan and Shmatikov (2006); Price and Cohen (2019)]. Given the five levels are represented as floating-point values between 0 and 1, we could treat the overall modeling problem as either regression problem or a classification problem. The Numer.ai team stated that although both regression and classification have their merits when approaching this problem, however in their own research they have found that modeling through regression leads to less over-fitting of historical data and thereby reducing variance of the models. In a number of tests we found that classification models were not noticeably better than random guesses.

To build a regression model, [Chung *et al.* (2023)] chose to use a Random Forest Regressor. This choice was made for several reasons. First, the Numer.ai team noted that ensemble and gradient-boosted models tend to work the best with their data and meta-model. There are several different choices possibilities given this general advice, for example, Random Forests and Gradient-boosted Decision trees (GBDTs) are known to be effective, however Random Forest could be built far more effectively on a parallel computer as all ensemble members could be trained at the same time, rather than sequentially like GBDTs. In short, the random forest approach was chosen for it computational efficiency.

K-fold Cross-Validation Given that the process of hyperparameter optimization requires the evaluation of many hyperparameter combinations (or configurations), a key strategy to reduce the cost of these evaluations is to shortcut the evaluation process[Eggensperger *et al.* (2019); Hospedales *et al.* (2022); Yu and Zhu (2020)]. A concrete strategy for implementing this shortcut idea is known as successive halving, which known to be more easily parallelized for taking advantage of massive computer clusters [Li *et al.* (2020); Meister *et al.* (2020)]. A more dynamic alternative is the multi-armed bandit algorithm [Jamieson and Talwalkar (2016); Karnin *et al.* (2013); Lattimore and Szepesvári (2020)]. The common theme among these approaches is to control the budget allocated to early stages – presumably when most of the configurations evaluated are not very promising, so that these evaluations would not cost more than later evaluations. The budget could be described in different ways, [Chung *et al.* (2023)] decided to use the number of data records as a specific measure for computational budget. This is an approach used by many hyperparameter optimization tools [Kingma and Ba (2017); Zhou *et al.* (2023)].

Given that most of the evaluations in the HPO procedure will be only operate on a small fraction of the data, there is a concern about overfitting [Bailey *et al.* (2014); Park and Ho (2021); Wu *et al.* (2015)]. A common approach for addressing this concern is through cross validation [Ng *et al.* (1997); Stephen Bates and Tibshirani (2023)]. We are aware of reports that cross validation might not be sufficient in some cases [Keevers (2019)], however, we don't have a generic strategy that could replace it.

In [Chung *et al.* (2023)], a 5-fold cross validation is used throughout the configuration evaluation process. Empirical results showed that this approach was able to provide more consistent outcome for the hyperparameter optimization task. In earlier tests, we also saw that the k-fold cross validation was effective in many different application scenarios [Lopez de Prado (2018b)].

1.2.4 *Hyperparameter Configurations and Parallelization*

Earlier, we mentioned that the actual hyperparameter optimization process experimented by [Chung *et al.* (2023)] not only include parameters for dimensional reduction algorithms, but also those of a random forest algorithm used for evaluating the quality of the DR output. Next, we briefly review the sampling procedure for generating the hyperparameter configurations and the parallelization approach to manage the computation time.

Latin Hypercube Sampling The software implementation of the Random Forest algorithm used by [Chung *et al.* (2023)] has eight hyperparameters. The implementations of the dimensionality reduction algorithms mentioned in Table 1.1 has about three to 20 additional hyperparameters. Even if we only choose to search through 3 different options per hyperparameter, an extremely modest amount, this would give $3^1 1 \sim 177,000$ configurations if we chose to use a simple grid search. Given that the Numer.ai dataset has millions of row and over 300 features, the computational cost would be enormous. A simple strategy to limit the amount of configuration examined would be to perform a multi-dimensional random sample, a more systematic approach is Latin Hypercube Sampling [Deutsch and Deutsch (2012)].

In general, a multi-dimensional Latin Hypercube Sampling (LHS) is more efficient version of Random Sampling for hyperparameter selection [Deutsch and Deutsch (2012)]. LHS searches through a wider scope of configurations of hyperparameters by randomly sampling through separate strata of the combinatorial grid of hyperparameters, avoiding combinations where are similar to each other, which would repeat needless computation.

Utilizing Dask Parallel Computation Given the number of configurations to explore and the computationally intensity of DR algorithms' and Random Forest model's training, we need to utilize parallel computation. The work of [Chung *et al.* (2023)] utilize python as Jupyter notebooks. Jupyter notebooks are fast becoming a standard for data scientists as their flexibility for performing analysis and tuning models are far more suited for common data science tasks [Perez and Granger (2007)]. Within this ecosystem, Dask is an effective software package that can connect local or remote compute clusters as an Python object which can be used interactively in Jupyter notebooks [Rocklin (2015)]. Dask is partuclarly easy to use for data parallel tasks such as independent evaluations of different hyperparameter configurations from a HPO procedure. Dask also builds DAGs for parallel operations which require computation done in a particular order as well, allowing for further flexibility if necessary.

1.2.5 *Use Building Monitoring Data to Understand Numer.ai Data*

Since the Numer.ai data is obfuscated, we are attempting to understand it by drawing analogy with another one that has some similarity. For this purpose, [Chung *et al.* (2023)] used a set of building monitoring data [Luo *et al.* (2022)]. The building monitoring data includes weather station recording of outdoor weather conditions and many details of the building's Heating, Ventilation, and Air Conditioning (HVAC) system as well as the electricity usage. In particular, the tests from [Chung *et al.* (2023)] used the weather conditions to build a model to predict the total daily electricity usage.

How Building Monitoring Data Might Be Useful The building monitoring dataset is useful for two main reasons. The relationship between outdoor weather condition and electricity usage of a building is well understood, therefore the dimensionality reduction results could be verified easily. The building monitoring data is relatively small, and therefore it takes less time to perform hyperparameter optimization. In the daily record form, the building monitoring data has 529 rows and 192 columns. Though the number of columns is comparable with Numer.ai data (310 columns), the number of rows is much smaller.

Turning Building Monitoring Dataset into Daily Records Both the building monitoring data and the Numer.ai data are time series. A key observation about the Numer.ai data is that it has more than 300 columns, while the outdoor weather condition given by the weather station has only four variables: air temperature, Dew point, relative humidity, and solar radiation. The building monitoring data is recorded every half-an-hour. For each day, there are 48 values for each of the four variables. For a whole day, there are 192 different values associated with the four variables. By folding all 192 values into a single daily record, a new daily time series would have 192 features for each total daily electricity usage (the target variable of the regssion). This process creates a data table with nearly as many columns as Numer.ai data.

After turning the building monitoring data into a wider table, we also apply a quantization process similar to that used to generated the Numer.ai data. A simple description of this quantization is that each column (and each epoch) is processed separately; for a five-level quantization, the top fifth of the values are converted to 1, the fifth to 0.75, and so on. The details of this quantization process could be found in [Chung *et al.* (2023)]. This quantization process preserves some correlation between columns, however, from Figure 1.1 we see that it changes correlations quite noticeably.

Due to the data construction process, the building monitoring dataset shows high multicollinearity. Columns from the same weather feature show high correlation as apparent in Figure 1.1(a). Similarly, there are significant correlation among the columns of the Numerai data. Both of them could benefit from dimensionality reduction before model building [Mansfield and Helms (1982); Thompson *et al.* (2017)].

For the regression model, the weather features are used to predict the electricity usage that is dominated by the building's HVAC system. Due to strong variations within a day, the HVAC electricity usage does not follow

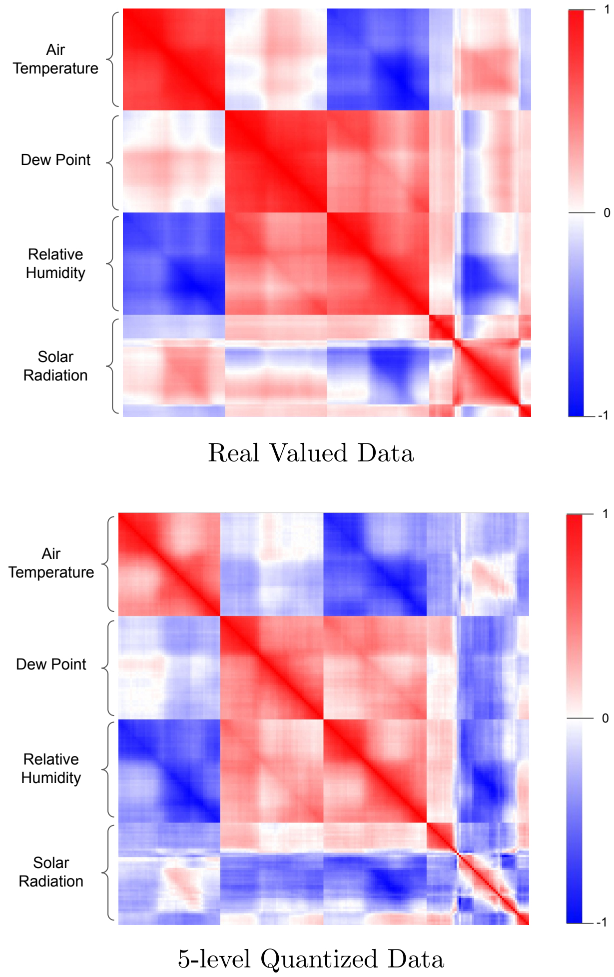

Fig. 1.1 Correlation heatmaps of both real-valued and quantized building monitoring data

the outdoor temperature or other weather features precisely, however, the overall electricity usage per day is primarily determined by the heat gain from outdoor weather. For this reason, the overall relationship between the weather condition and electricity usage per day could relatively easily modeled [Kim *et al.* (2018)]. In short, there is a reasonable expectation that the regression model for building monitoring data have a good solution, while the regression model for the Numer.ai data has no such expectation.

The best hyperparameter combination leads to a mean Spearman Rank Correlation of about 0.57 for the building monitoring data, while the same value remains below 0.05 for the Numer.ai data [Chung *et al.* (2023)].

1.3 Lessons from DR Study

After review the hyperparameter optimization for dimensionality reduction task in the previous section, next we outline the key lessons to take from the exercise.

1.3.1 *Understanding the Numer.ai data*

In the work by [Chung *et al.* (2023)], the authors attempted to understand the Numer.ai data by folding a building monitoring data set into daily records, which created a data set that resembles the Numer.ai one, but with known properties for the columns of the data table. Section 1.2.5 summarizes the similarities between the two data sets. Next, we describe the key differences.

After quantizing the building monitoring data, the regression model generated with the random forest is able to capture significant amount of information about the electricity usage as measured by the Spearman Rank Correlation [Chung *et al.* (2023)]. In contrast the regression model produced by the random forest from the Numer.ai data was only very weakly correlated with the target variable, where the observed mean SRC is less than 0.05. Extracting weak signal from data is an interesting challenge on its own. One opportunity we see in studying how the obfuscation process impacted the signal extraction. In particular, we see from Figure 1.1 that quantization noticeably affected the correlation among the columns of the feature table. Quantifying this impact could be a useful exercise for further understanding the Numer.ai data.

1.3.2 *Selecting the right objective function for HPO*

The dimensionality reduction task is typically a part of the feature engineering process for data analysis and machine learning [Mierswa (2016)]. As such, the quality of DR task might be better measured together with the down-stream analysis operations. Thus the hyperparameter optimization should use these measures that include the analysis tasks. In fact, several pitfalls associated with hyperparameter tuning are related to ineffectual target being used for optimization [Eggensperger *et al.* (2019); Vento and

Fanfarillo (2019)]. In [Chung *et al.* (2023)], a regression model was selected as the downstream task for dimensionality reduction.

Based on the discussion earlier, the regression model is well-suited for building monitoring use case, and the hyperparameter optimization was able to noticeably improve the Spearman Rank Correlation used to measure the quality of the regression models. However, for the Numer.ai test case, the situation is not as clear because the Spearman Rank Correlation values are always in a few percent (~ 0.05) and the dimensionality reduction techniques were not able to improve the results of regression according to the SRC measures. So far, the tests could not differentiate among the following possible causes: (1) SRC is not a good measure of quality, (2) regression models generated by the random forest approach are not effective for the use case – either the random forest approach is not able to build a good regression model or the regression model is not a suitable modeling; (3) the features contained in the Numer.ai data set are not capturing the relationship with the target variable.

In Section 1.2.3, an argument was made for performing k-fold validation to minimize the potential bias introduced by working with a small number of data records during the early rounds of the HPO process. This use of a smaller number of data records during HPO is generally described as early stopping in literature [Falkner *et al.* (2018); Karnin *et al.* (2013)] and is a critical to the overall effectiveness of HPO algorithms. In general, using less budget during the early rounds of an HPO process introduces higher uncertainty to the quality metric used. A systematic approach to addressing this issue is to estimate the confidence interval of the quality metric and only eliminate hyperparameter configurations that are sure to be not competitive [Jamieson and Talwalkar (2016); Karnin *et al.* (2013)].

1.3.3 *Efficient HPO algorithm makes a difference*

From the experience of optimizing the hyperparameters for dimensionality reduction [Chung *et al.* (2023)], we see that an effective optimization algorithm makes a big difference. This could be considered as a confirmation of the vast quantity of published work on hyperparameter optimization (HPO) [Bischl *et al.* (2023); Morales-Hernández *et al.* (2023); Karl *et al.* (2023)]. We see three key factors that are critical to an effective HPO approach:

- A high quality set of hyperparameter combinations for exploration. To reduce the complexity of selecting configuration for the next round, highly parallel HPO algorithms often use relatively static set of hyperparameter combinations. In general, an effective trade-off between exploration and exploitation is core to an HPO algorithm and there are extensive study on this topic [De Ath *et al.* (2021); Zhou *et al.* (2023)]. In [Chung *et al.* (2023)], a static choice is made with Latin Hypercube Sampling of the hyperparameter space. This choice is relatively simple to implement and is found to be effective for making uses of modest number of computing nodes. For an effective exploration of a large number of computing elements, a more dynamic approach would be more effective [Li *et al.* (2017); Falkner *et al.* (2018); Meister *et al.* (2020)].

- The second factor in an effective HPO algorithm is a robust early stopping strategy for evaluating not-so-promising hyperparameter configurations [Brandt *et al.* (2024); Falkner *et al.* (2018); Karnin *et al.* (2013)]. By allocating a small budget to an early round of evaluations, an HPO procedure could complete the the evaluations of a large number of test cases in the early rounds with modest cost and ensure the overall cost of the whole HPO procedure to be a small multiple of evaluating the full budget instances of the most promising hyperparameter configurations. In this process, the right objective function for optimization is critical, and so is the estimation of the uncertainty of the objective function [Bischl *et al.* (2023); Falkner *et al.* (2018); Karl *et al.* (2023); Jamieson and Talwalkar (2016); Karnin *et al.* (2013)].
- The 3rd factor in an effective HPO algorithm is its ability to take advantage of the parallel computing resources available. A large HPO problem might need to examine many thousands of configurations that could be evaluated independently [Li *et al.* (2020); Meister *et al.* (2020)]. At the same time, modern computing platforms typically have many computing elements available. [Chung *et al.* (2023)] have taken a relatively straightforward approach for parallelizing each round of an HPO process, while other forms of parallelization are also available in published tools [Falkner *et al.* (2018); Gonzalez and Zavala (2023); Li *et al.* (2020, 2022); Zhou *et al.* (2023)].

1.4 Observations with Broader Applications

Based on the lessons from the hyperparameter optimization for a set of dimensionality reduction use cases described earlier, we next draw a few observations for more general hyperparameter optimization problems. For those interested in the HPO algorithms and tools, please refer to recent review articles [Bischl *et al.* (2023); Karl *et al.* (2023); Morales-Hernández *et al.* (2023)].

Select the right tool In the study by [Chung *et al.* (2023)], the regression model with weather variables as input to predict electricity usage happens to work well because the total daily electricity usage is highly dependent on the total amount of heat needs to be moved in order to keep the indoor temperature constant for human comfort [Kim *et al.* (2018)]. Had we directly attempted to use the original time series, the regression model would work not nearly as well because there is a significant delay between the rise of the indoor temperature that triggers HVAC operation and the rise of outdoor air temperature. This delay is hard to capture in any regression model. The lesson here is that a data scientist needs to consider carefully what modeling tool is suitable for the problem at hand.

Design the objective function for optimization carefully In studying the two test cases in [Chung *et al.* (2023)], the authors selected to use a regression model to measure the quality of the dimensionality reduction task. We say that the regression model was a good fit for the building monitoring test case because the resulting regression model captures the relationship between daily weather condition and the total electricity usage. In many data analysis use case, there is a downstream task that makes use of the output of the current analysis results. In which case, it would make sense to consider the effect of the current task on the downstream one in constructing the objective of HPO. Some of the pitfalls observed in the literature are intimately related to optimizing with the wrong objective function [Eggensperger *et al.* (2019); Vento and Fanfarillo (2019); Park and Ho (2021)].

In some algorithms, a confidence interval or a uncertainty might be available without much extra computation. In such cases, one may insist on requiring an estimation of uncertainty or confidence interval from the modeling procedure [Pineda and Serpa (2021); Wang *et al.* (2023); Yang and Shami (2020)]. Such confidence information would be helpful in selecting the more promising hyperparameter configurations that deserve additional evaluation during HPO, which would lead to more robust answers from HPO.

Make effective uses of the parallel computing resources available Selecting the right HPO tool could make a big difference. There are a number of recently published reviews that could provide valuable information [Bischl *et al.* (2023); Karl *et al.* (2023); Morales-Hernández *et al.* (2023); Moosbauer *et al.* (2022); Shekhar *et al.* (2021)]. As observed in the previous section, the most important features of a HPO algorithm includes: effective hyperparameter space sampling, reliable early stopping strategy, and efficient parallel execution strategies. Several HPO tools implement state of the art options for all these three features [Akiba *et al.* (2019); Li *et al.* (2020); Zhou *et al.* (2023)].

Most popular machine linear frameworks have their own hyperparameter optimization or neural network architecture optimization tools. For example, Scikit Learn has Scikit-Optimize* and Ray has Ray Tune†. Large commercial AI frameworks also have their own optimization package, e.g., Microsoft has AutoML [He *et al.* (2021)], Google has Vizer, and Amazon AWS has Sage Maker.

To optimize in a large hyperparameter space, potentially multiple parallelization strategies might be needed. For example, during the early stages of HPO, each configuration is evaluated on a relatively small budget that might fit on a single compute node or a single GPU. However, during the later stages, each configuration is evaluated with a larger budget and might benefit from parallelization to reduce the execution time. Additionally, there are many more configurations to evaluate during the earlier stages of HPO to make use of a parallel computing environment, while the later stages might have far fewer configurations to evaluate, where it could be useful to breakup the evaluation of each configuration onto multiple computing elements to make better use of the computing resources. The existing HPO tools primarily parallelize the evaluation of multiple configurations, which might not fully utilize the available computing resources during the later stages of HPO process.

---

*`https://scikit-optimize.github.io/stable/auto_examples/hyperparameter-optimization.html`

†`https://docs.ray.io/en/master/tune/index.html`

Check for potential shortcomings of the optimized models It is important to select the right hyperparameters to make an effective use of an advanced analysis technique, however, it is just as appropriate to avoid the traps and pitfalls identified in the literature [Eggensperger *et al.* (2019); Vento and Fanfarillo (2019)]. A systematic approach for addressing these problems would be to detect them. As overfitting is often the first concern, there are a number of publications on detecting overfitting in various contexts [Bailey *et al.* (2014); Park and Ho (2021); Webster *et al.* (2019); Werpachowski *et al.* (2019)]. Recently, there was a tool developed for detecting a broader class of problems known as underspecification [Jacobs *et al.* (2022)]. It has not been a common practice to check for pitfalls in a data analysis result. We strongly recommend a data scientist to consider utilizing the automated tools being developed and published.

1.5 Summary and Observations

In this work, we reviewed the experience from optimizing the hyperparameters with two different dimensionality reduction use cases, a building monitoring use case and a use case from Numer.ai competition [Chung *et al.* (2023)]. Because the Numer.ai data is obfuscated, the authors attempted to use the building monitoring data to understand the Numer.ai data. Even though the original study was attempting to optimize the hyperparameters for the dimensionality reduction, the experiences from this and other related data analysis work [Lopez de Prado (2018b)] allowed us to draw a few lessons that are broadly applicable to different hyperparameter optimization problems.

In order to automatically optimize the hyperparameter choices, the appropriate optimization function is critical to the usefulness of the result of optimization. Our general advice is for the user to understand the most important downstream analysis task and take into account of this task in constructing the objective function for hyperparameter optimization. In the study by [Chung *et al.* (2023)], a regression model is selected as the downstream analysis for the dimensionality reduction task and the regression accuracy is used as the objective for minimization. From that study, we known that the regression model is effective for capturing the relationship between weather conditions and electricity usage of a building, but the not so for the Numer.ai test case. Further investigation is necessary to differentiate the different possible reasons for the observation on Numer.ai data.

There are a number of publications on various traps and pitfalls about HPO strategies currently available [Eggensperger *et al.* (2019); Vento and Fanfarillo (2019); Webster *et al.* (2019)]. There are some research tools for identify common issues such as overfitting [Bailey *et al.* (2014); Webster *et al.* (2019); Werpachowski *et al.* (2019)] and a broader class known as underspecification [Jacobs *et al.* (2022)], For future research, more tools for detecting a wider class of issues would be needed. In daily practice, we strong recommend every data scientist to utilize these existing tools to detect known issues from their analysis results.

Bibliography

Akiba, T., Sano, S., Yanase, T., Ohta, T., and Koyama, M. (2019). Optuna: A next-generation hyperparameter optimization framework, in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19 (Association for Computing Machinery, New York, NY, USA), ISBN 9781450362016, pp. 2623–2631, doi: 10.1145/3292500.3330701.

Anowar, F., Sadaoui, S., and Selim, B. (2021). Conceptual and empirical comparison of dimensionality reduction algorithms (pca, kpca, lda, mds, svd, lle, isomap, le, ica, t-sne), *Computer Science Review* **40**, p. 100378, doi: 10.1016/j.cosrev.2021.100378.

Ayesha, S., Hanif, M. K., and Talib, R. (2020). Overview and comparative study of dimensionality reduction techniques for high dimensional data, *Information Fusion* **59**, pp. 44–58, doi:10.1016/j.inffus.2020.01.005.

Bailey, D. H., Borwein, J. M., Lopez de Prado, M., and Zhu, Q. J. (2014). Pseudo-mathematics and financial charlatanism: The effects of backtest overfitting on out-of-sample performance, *Notices of the AMS* **61**, 5, pp. 458–471.

Bischl, B., Binder, M., Lang, M., Pielok, T., Richter, J., Coors, S., Thomas, J., Ullmann, T., Becker, M., Boulesteix, A.-L., Deng, D., and Lindauer, M. (2023). Hyperparameter optimization: Foundations, algorithms, best practices, and open challenges, *WIREs Data Mining and Knowledge Discovery* **13**, 2, doi:10.1002/widm.1484.

Bottou, L., Curtis, F. E., and Nocedal, J. (2018). Optimization methods for large-scale machine learning, *SIAM Review* **60**, 2, pp. 223–311, doi:10.1137/16M1080173.

Brandt, J., Wever, M., Iliadis, D., Bengs, V., and Hüllermeier, E. (2024). Incremental successive halving for hyperparameter optimization with budget constraints, `https://openreview.net/forum?id=Pa4hecILrt`.

Brigato, L., Barz, B., Iocchi, L., and Denzler, J. (2021). Tune it or don't use it: Benchmarking data-efficient image classification, in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, pp. 1071–1080.

Carbonell, J. G., Michalski, R. S., and Mitchell, T. M. (1983). 1 - an overview of machine learning, in R. S. Michalski, J. G. Carbonell, and T. M. Mitchell (eds.), *Machine Learning* (Morgan Kaufmann, San Francisco (CA)), pp. 3–23, doi:10.1016/B978-0-08-051054-5.50005-4.

Chung, J., Lopez de Prado, M., Simon, H., and Wu, K. (2023). Data driven dimensionality reduction to improve modeling performance, in *Proceedings of the 35th International Conference on Scientific and Statistical Database Management*, SSDBM '23 (Association for Computing Machinery, New York, NY, USA), doi:10.1145/3603719.3603744.

Cooper, A. F., Lu, Y., Forde, J., and De Sa, C. M. (2021). Hyperparameter optimization is deceiving us, and how to stop it, in M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan (eds.), *Advances in Neural Information Processing Systems*, Vol. 34 (Curran Associates, Inc.), pp. 3081–3095, `https://proceedings.neurips.cc/paper_files/paper/2021/file/17fafe5f6ce2f1904eb09d2e80a4cbf6-Paper.pdf`.

Cuomo, S., Di Cola, V. S., Giampaolo, F., Rozza, G., Raissi, M., and Piccialli, F. (2022). Scientific machine learning through physics–informed neural networks: Where we are and what's next, *Journal of Scientific Computing* **92**, 3, p. 88, doi:10.1007/s10915-022-01939-z.

De Ath, G., Everson, R. M., Rahat, A. A. M., and Fieldsend, J. E. (2021). Greed is good: Exploration and exploitation trade-offs in bayesian optimisation, *ACM Trans. Evol. Learn. Optim.* **1**, 1, doi:10.1145/3425501.

Deutsch, J. L. and Deutsch, C. V. (2012). Latin hypercube sampling with multidimensional uniformity, *Journal of Statistical Planning and Inference* **142**, 3, pp. 763–772, doi:https://doi.org/10.1016/j.jspi.2011.09.016.

Diaz, G. I., Fokoue-Nkoutche, A., Nannicini, G., and Samulowitz, H. (2017). An effective algorithm for hyperparameter optimization of neural networks, *IBM Journal of Research and Development* **61**, 4/5, pp. 9:1–9:11, doi:10.1147/JRD.2017.2709578.

Eggensperger, K., Lindauer, M., and Hutter, F. (2019). Pitfalls and best practices in algorithm configuration, *Journal of Artificial Intelligence Research* **64**, pp. 861–893, doi:10.1613/jair.1.11420.

Falkner, S., Klein, A., and Hutter, F. (2018). BOHB: Robust and efficient hyperparameter optimization at scale, in J. Dy and A. Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, *Proceedings of Machine Learning Research*, Vol. 80 (PMLR), pp. 1437–1446, `https://proceedings.mlr.press/v80/falkner18a.html`.

Feurer, M., Klein, A., Eggensperger, K., Springenberg, J., Blum, M., and Hutter, F. (2015). Efficient and robust automated machine learning, in C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, Vol. 28 (Curran Associates, Inc.), `https://proceedings.neurips.cc/paper_files/paper/2015/file/11d0e6287202fced83f79975ec59a3a6-Paper.pdf`.

Gonzalez, L. D. and Zavala, V. M. (2023). New paradigms for exploiting parallel experiments in bayesian optimization, *Computers & Chemical Engineering* **170**, p. 108110, doi:10.1016/j.compchemeng.2022.108110.

Guo, C. and Berkhahn, F. (2016). Entity embeddings of categorical variables, `arXiv:1604.06737 [cs.LG]`.

Han, H., Guo, X., and Yu, H. (2016). Variable selection using mean decrease accuracy and mean decrease gini based on random forest, in *2016 7th IEEE International Conference on Software Engineering and Service Science (ICSESS)*, pp. 219–224, doi:10.1109/ICSESS.2016.7883053.

He, X., Zhao, K., and Chu, X. (2021). Automl: A survey of the state-of-the-art, *Knowledge-Based Systems* **212**, p. 106622, doi:10.1016/j.knosys.2020.106622.

Hospedales, T., Antoniou, A., Micaelli, P., and Storkey, A. (2022). Meta-learning in neural networks: A survey, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **44**, 9, pp. 5149–5169, doi:10.1109/TPAMI.2021.3079209.

Jacobs, A. S., Beltiukov, R., Willinger, W., Ferreira, R. A., Gupta, A., and Granville, L. Z. (2022). AI/ML for network security: The emperor has no clothes, in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, CCS '22 (Association for Computing Machinery, New York, NY, USA), ISBN 9781450394505, pp. 1537–1551, doi:10.1145/3548606.3560609.

Jacquier, A., Kondratyev, O., Lipton, A., and Lopez de Prado, M. (2022). *Quantum Machine Learning and Optimisation in Finance: On the Road to Quantum Advantage* (Packt Publishing Ltd).

Jamieson, K. and Talwalkar, A. (2016). Non-stochastic best arm identification and hyperparameter optimization, in A. Gretton and C. C. Robert (eds.), *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, *Proceedings of Machine Learning Research*, Vol. 51 (PMLR, Cadiz, Spain), pp. 240–248, `https://proceedings.mlr.press/v51/jamieson16.html`.

Jia, W., Sun, M., Lian, J., and Hou, S. (2022). Feature dimensionality reduction: a review, *Complex & Intelligent Systems* **8**, 3, pp. 2663–2693, doi:10.1007/s40747-021-00637-x.

Jordan, M. I. and Mitchell, T. M. (2015). Machine learning: Trends, perspectives, and prospects, *Science* **349**, 6245, pp. 255–260, doi:10.1126/science.aaa8415, `https://www.science.org/doi/pdf/10.1126/science.aaa8415`.

Kandasamy, K., Dasarathy, G., Schneider, J., and Póczos, B. (2017). Multi-fidelity Bayesian optimisation with continuous approximations, in D. Precup and Y. W. Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning*, *Proceedings of Machine Learning Research*, Vol. 70 (PMLR), pp. 1799–1808, `https://proceedings.mlr.press/v70/kandasamy17a.html`.

Karl, F., Pielok, T., Moosbauer, J., Pfisterer, F., Coors, S., Binder, M., Schneider, L., Thomas, J., Richter, J., Lang, M., Garrido-Merchán, E. C., Branke, J., and Bischl, B. (2023). Multi-objective hyperparameter optimization in machine learning—an overview, *ACM Trans. Evol. Learn. Optim.* **3**, 4, doi:10.1145/3610536.

Karniadakis, G. E., Kevrekidis, I. G., Lu, L., Perdikaris, P., Wang, S., and Yang, L. (2021). Physics-informed machine learning, *Nature Reviews Physics* **3**, 6, pp. 422–440, doi:10.1038/s42254-021-00314-5.

Karnin, Z., Koren, T., and Somekh, O. (2013). Almost optimal exploration in multi-armed bandits, in S. Dasgupta and D. McAllester (eds.), *Proceedings of the 30th International Conference on Machine Learning*, *Proceedings of Machine Learning Research*, Vol. 28 (PMLR, Atlanta, Georgia, USA), pp. 1238–1246, `https://proceedings.mlr.press/v28/karnin13.html`.

Keevers, T. L. (2019). Cross-validation is insufficient for model validation, Tech. Rep. DST-Group–TR–3576, Joint and Operations Analysis Division, Defence Science and Technology Group: Victoria, Australia, `https://www.dst.defence.gov.au/sites/default/files/publications/documents/DST-Group-TR-3576.pdf`.

Khanan, A., Abdullah, S., Mohamed, A. H. H. M., Mehmood, A., and Ariffin, K. A. Z. (2019). Big data security and privacy concerns: A review, in A. Al-Masri and K. Curran (eds.), *Smart Technologies and Innovation for a Sustainable Future* (Springer International Publishing, Cham), pp. 55–61, doi:10.1007/978-3-030-01659-3_8.

Kim, T., Choi, J., Lee, D., Sim, A., Spurlock, C. A., Todd, A., and Wu, K. (2018). Predicting baseline for analysis of electricity pricing, *International Journal of Big Data Intelligence* **5**, 1/2, pp. 3–20, doi:10.1504/IJBDI.2018.10008133.

Kingma, D. P. and Ba, J. (2017). Adam: A method for stochastic optimization, `arXiv:1412.6980 [cs.LG]`.

Lattimore, T. and Szepesvári, C. (2020). *Bandit algorithms* (Cambridge University Press).

Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A., and Talwalkar, A. (2017). Hyperband: A novel bandit-based approach to hyperparameter optimization, *The Journal of Machine Learning Research* **18**, 1, pp. 6765–6816, `https://www.jmlr.org/papers/v18/16-558.html`.

Li, L., Jamieson, K., Rostamizadeh, A., Gonina, E., Ben-tzur, J., Hardt, M., Recht, B., and Talwalkar, A. (2020). A system for massively parallel hyperparameter tuning, in I. Dhillon, D. Papailiopoulos, and V. Sze (eds.), *Proceedings of Machine Learning and Systems*, Vol. 2, pp. 230–246, `https://proceedings.mlsys.org/paper_files/paper/2020/file/a06f20b349c6cf09a6b171c71b88bbfc-Paper.pdf`.

Li, Y., Shen, Y., Jiang, H., Zhang, W., Yang, Z., Zhang, C., and Cui, B. (2022). Transbo: Hyperparameter optimization via two-phase transfer learning, in *KDD '22* (Association for Computing Machinery, New York, NY, USA), pp. 956–966, doi:10.1145/3534678.3539255.

Lindauer, M., Eggensperger, K., Feurer, M., Biedenkapp, A., Deng, D., Benjamins, C., Ruhkopf, T., Sass, R., and Hutter, F. (2022). SMAC3: A versatile bayesian optimization package for hyperparameter optimization, *J. Mach. Learn. Res.* **23**, 1, `https://www.jmlr.org/papers/volume23/21-0888/21-0888.pdf`.

Lopez de Prado, M. (2018a). The 10 reasons most machine learning funds fail, *The Journal of Portfolio Management* **44**, 6, pp. 120–133, doi:10.3905/jpm.2018.44.6.120.

Lopez de Prado, M. (2018b). *Advances in financial machine learning* (John Wiley & Sons).

Lopez de Prado, M. (2020). Clustered feature importance (presentation slides), *SSRN Electronic Journal* doi:10.2139/ssrn.3517595.

Lundberg, S. M. and Lee, S.-I. (2017). A unified approach to interpreting model predictions, in I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 30* (Curran Associates, Inc.), pp. 4765–4774, `http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf`.

Luo, N., Wang, Z., Blum, D., Weyandt, C., Bourassa, N., Piette, M. A., and Hong, T. (2022). A three-year dataset supporting research on building energy management and occupancy analytics, *Scientific Data* **9**, 1, p. 156, doi:10.1038/s41597-022-01257-x.

Maaten, L. V. D., Postma, E., den Herik, J. V., *et al.* (2009). Dimensionality reduction: a comparative, *J Mach Learn Res* **10**, 66-71, p. 13.

Maćkiewicz, A. and Ratajczak, W. (1993). Principal components analysis (fpca), *Computers & Geosciences* **19**, pp. 303–342.

Mahesh, B. (2020). Machine learning algorithms - a review, *International Journal of Science and Research (IJSR)* **9**, 1, doi:10.21275/ART20203995.

Mansfield, E. R. and Helms, B. P. (1982). Detecting multicollinearity, *The American Statistician* **36**, 3a, pp. 158–160, doi:10.1080/00031305.1982.10482818.

McInnes, L. and Healy, J. M. J. (2020). Umap: Uniform manifold approximation and projection for dimension reduction, *arXiv* .

Mease, D. and Wyner, A. (2008). Evidence contrary to the statistical view of boosting. *Journal of Machine Learning Research* **9**, 2.

Meister, M., Sheikholeslami, S., Payberah, A. H., Vlassov, V., and Dowling, J. (2020). Maggy: Scalable asynchronous parallel hyperparameter search, in *Proceedings of the 1st Workshop on Distributed Machine Learning*, DistributedML'20 (Association for Computing Machinery, New York, NY, USA), ISBN 9781450381826, pp. 28–33, doi:10.1145/3426745.3431338.

Meng, C., Seo, S., Cao, D., Griesemer, S., and Liu, Y. (2022). When physics meets machine learning: A survey of physics-informed machine learning, `arXiv:2203.16797 [cs.LG]`.

Mierswa, I. (2016). The wisdom of crowds: Best practices for data prep & machine learning derived from millions of data science workflows, in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16 (Association for Computing Machinery, New York, NY, USA), p. 411, doi:10.1145/2939672.2945365.

Moosbauer, J., Binder, M., Schneider, L., Pfisterer, F., Becker, M., Lang, M., Kotthoff, L., and Bischl, B. (2022). Automated benchmark-driven design and explanation of hyperparameter optimizers, *IEEE Transactions on Evolutionary Computation* **26**, 6, pp. 1336–1350, doi:10.1109/TEVC.2022.3211336.

Morales-Hernández, A., Van Nieuwenhuyse, I., and Rojas Gonzalez, S. (2023). A survey on multi-objective hyperparameter optimization algorithms for machine learning, *Artificial Intelligence Review* **56**, 8, pp. 8043–8093, doi: 10.1007/s10462-022-10359-2.

Narayanan, A. and Shmatikov, V. (2006). How to break anonymity of the netflix prize dataset, *CoRR* **abs/cs/0610105**, arXiv:cs/0610105, http://arxiv.org/abs/cs/0610105.

Ng, A. Y. *et al.* (1997). Preventing" overfitting" of cross-validation data, in *ICML*, Vol. 97 (Citeseer), pp. 245–253.

Nguyen, L. H. and Holmes, S. (2019). Ten quick tips for effective dimensionality reduction, *PLoS computational biology* **15**, 6, p. e1006907, doi:10.1371/journal.pcbi.1006907.

Numerai (2024). Numerai tournament overview, https://docs.numer.ai/tournament/learn.

Park, Y. and Ho, J. C. (2021). Tackling overfitting in boosting for noisy healthcare data, *IEEE Transactions on Knowledge and Data Engineering* **33**, 7, pp. 2995–3006, doi:10.1109/TKDE.2019.2959988.

Perez, F. and Granger, B. E. (2007). Ipython: A system for interactive scientific computing, *Computing in Science Engineering* **9**, 3, pp. 21–29, doi:10.1109/MCSE.2007.53.

Pineda, L. R. and Serpa, A. L. (2021). Determination of confidence bounds and artificial neural networks in non-linear optimization problems, *Neurocomputing* **463**, pp. 495–504, doi:10.1016/j.neucom.2021.08.075.

Price, W. N. and Cohen, I. G. (2019). Privacy in the age of medical big data, *Nature medicine* **25**, 1, pp. 37–43, doi:10.1038/s41591-018-0272-7.

Ray, P., Reddy, S. S., and Banerjee, T. (2021). Various dimension reduction techniques for high dimensional data analysis: a review, *Artificial Intelligence Review* **54**, pp. 3473–3515, doi:10.1007/s10462-020-09928-0.

Reddy, G. T., Reddy, M. P. K., Lakshmanna, K., Kaluri, R., Rajput, D. S., Srivastava, G., and Baker, T. (2020). Analysis of dimensionality reduction techniques on big data, *IEEE Access* **8**, pp. 54776–54788, doi:10.1109/ACCESS.2020.2980942.

Rocklin, M. (2015). Dask: Parallel computation with blocked algorithms and task scheduling, in *Proceedings of the 14th python in science conference*, Vol. 130 (Citeseer), p. 136.

Roweis, S. T. and Saul, L. K. (2000). Nonlinear dimensionality reduction by locally linear embedding, *Science* **290**, 5500, pp. 2323–2326, doi:10.1126/science.290.5500.2323.

Scholkopf, B., Smola, A., and Muller, K.-R. (1997). Kernel principal component analysis, in W. Gerstner, A. Germond, M. Hasler, and J.-D. Nicoud (eds.), *Artificial Neural Networks — ICANN'97* (Springer Berlin Heidelberg, Berlin, Heidelberg), pp. 583–588.

Seger, C. (2018). An investigation of categorical variable encoding techniques in machine learning: binary versus one-hot and feature hashing, Tech. Rep. 2018:596, KTH, School of Electrical Engineering and Computer Science (EECS).

Shekhar, S., Bansode, A., and Salim, A. (2021). A comparative study of hyper-parameter optimization tools, in *2021 IEEE Asia-Pacific Conference on*

Computer Science and Data Engineering (CSDE), pp. 1–6, doi:10.1109/CSDE53843.2021.9718485.

Stephen Bates, T. H. and Tibshirani, R. (2023). Cross-validation: What does it estimate and how well does it do it? *Journal of the American Statistical Association* **0**, 0, pp. 1–12, doi:10.1080/01621459.2023.2197686.

Tenenbaum, J. B., de Silva, V., and Langford, J. C. (2000). A global geometric framework for nonlinear dimensionality reduction, *Science* **290**, 5500, pp. 2319–2323, doi:10.1126/science.290.5500.2319.

Thompson, C. G., Kim, R. S., Aloe, A. M., and Becker, B. J. (2017). Extracting the variance inflation factor and other multicollinearity diagnostics from typical regression results, *Basic and Applied Social Psychology* **39**, 2, pp. 81–90, doi:10.1080/01973533.2016.1277529.

Vento, D. D. and Fanfarillo, A. (2019). Traps, pitfalls and misconceptions of machine learning applied to scientific disciplines, in *Proceedings of the Practice and Experience in Advanced Research Computing on Rise of the Machines (Learning)*, PEARC '19 (Association for Computing Machinery, New York, NY, USA), ISBN 9781450372275, doi:10.1145/3332186.3332209.

Wang, X., Jin, Y., Schmitt, S., and Olhofer, M. (2023). Recent advances in bayesian optimization, *ACM Comput. Surv.* **55**, 13s, doi:10.1145/3582078.

Waring, J., Lindvall, C., and Umeton, R. (2020). Automated machine learning: Review of the state-of-the-art and opportunities for healthcare, *Artificial Intelligence in Medicine* **104**, p. 101822, doi:10.1016/j.artmed.2020.101822.

Webster, R., Rabin, J., Simon, L., and Jurie, F. (2019). Detecting overfitting of deep generative networks via latent recovery, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11273–11282.

Werpachowski, R., György, A., and Szepesvari, C. (2019). Detecting overfitting via adversarial examples, in H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, Vol. 32 (Curran Associates, Inc.), `https://proceedings.neurips.cc/paper_files/paper/2019/file/28f7241796510e838db4a1384ae1279d-Paper.pdf`.

Wu, J., Chen, X.-Y., Zhang, H., Xiong, L.-D., Lei, H., and Deng, S.-H. (2019). Hyperparameter optimization for machine learning models based on bayesian optimization, *Journal of Electronic Science and Technology* **17**, 1, pp. 26–40, doi:10.11989/JEST.1674-862X.80904120.

Wu, K. J., Ger, S., Sim, A., Bailey, D. H., and Lopez de Prado, M. (2015). Simple example of backtest overfitting (SEBO), doi:10.11578/dc.20210521.74.

Yang, L. and Shami, A. (2020). On hyperparameter optimization of machine learning algorithms: Theory and practice, *Neurocomputing* **415**, pp. 295–316, doi:10.1016/j.neucom.2020.07.061.

Yoon, J. (2021). Forecasting of real gdp growth using machine learning models: Gradient boosting and random forest approach, *Computational Economics* **57**, 1, pp. 247–265, doi:10.1007/s10614-020-10054-w.

Yu, T. and Zhu, H. (2020). Hyper-parameter optimization: A review of algorithms and applications, `arXiv:2003.05689 [cs.LG]`.

Yu, X., Chen, Z., and Lu, Y. (2023). Harnessing LLMs for temporal data - a study on explainable financial time series forecasting, in M. Wang and I. Zitouni (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: Industry Track* (Association for Computational Linguistics, Singapore), pp. 739–753, doi:10.18653/v1/2023.emnlp-industry.69.

Zar, J. H. (1972). Significance testing of the spearman rank correlation coefficient, *Journal of the American Statistical Association* **67**, 339, pp. 578–580, doi: 10.1080/01621459.1972.10481251.

Zebari, R., Abdulazeez, A., Zeebaree, D., Zebari, D., and Saeed, J. (2020). A comprehensive review of dimensionality reduction techniques for feature selection and feature extraction, *Journal of Applied Science and Technology Trends* **1**, 2, pp. 56–70, doi:10.38094/jastt1224.

Zhan, H., Gomes, G., Li, X. S., Madduri, K., Sim, A., and Wu, K. (2018). Consensus ensemble system for traffic flow prediction, *IEEE Transactions on Intelligent Transportation Systems* **19**, 12, pp. 3903–3914, doi:10.1109/TITS.2018.2791505.

Zhou, J., Shi, Q., Ding, Y., Wang, L., Li, L., and Zhu, F. (2023). AntTune: An efficient distributed hyperparameter optimization system for large-scale data, in X. Wang, M. L. Sapino, W.-S. Han, A. El Abbadi, G. Dobbie, Z. Feng, Y. Shao, and H. Yin (eds.), *Database Systems for Advanced Applications* (Springer Nature Switzerland, Cham), pp. 477–489, doi:10.1007/978-3-031-30678-5_35.