More is not always better

Exploring data valuation approaches

DIT892 Project Course - Group 7

Supervisor: Anna Tomberg

Group members:

Konstantin Hanlon, Noa Holmén, Anton Nordkvist, Yura Ueno

Table of Contents

Table of Contents **2**
1. Introduction **3**
2. Data Valuation use-cases **3**
2.1 Data marketplaces 3
2.2 Data Valuation in machine learning 4
3. Theoretical background **5**
3.1 Data Valuation principles 5
3.2 Current Data Valuation approaches for machine learning 6
4. Description and Implementation of the different Algorithms **7**
4.1 Leave-One-Out (LOO) 7
4.2 Truncated Monte Carlo Shapley 7
4.3 Distributional Shapley 9
4.4 KNN Surrogate Shapley values 10
5. Methodological Choices: Datasets and Models **12**
5.1 Regression task 12
5.1.1 Dataset 12
5.1.2 Feature Selection 12
5.1.3 Data Preparation 12
5.1.4 Model 13
5.2 Classification task 13
5.2.1 Dataset 13
5.2.2 Feature selection and reduction 13
5.2.3 Data preparation 14
6. Methods for evaluation **14**
6.1 Removing data points 14
6.2 Mislabeled data 15
6.3 Correlation and Computation Time 15
7. Results **15**
7.1 Removing data points 15
7.2 Finding mislabeled- or nonFunc data 17
7.3 Correlation and Computation Time 18
8. Discussion **19**
9. Conclusion **20**
References **21**
Appendix A **23**
Appendix B **26**
Appendix C **27**

1. Introduction

“Data is the new oil” – a saying that is often heard in a machine learning (ML) environment – but one that shows how valuable data has become as a resource, empowering both business and scientific research over the last years (Jia et al., 2018). As Ghorbani et al. (2020) describe it in a paper on the topic, data has become an “essential driver of innovation and service”. However, not all data sources are equally valuable (Ghorbani & Zou, 2019). Data needs to be of a high quality in order to actually enhance such processes (Jia et al, 2018). This gives rise to the question of how we can evaluate data and data sources. What data has a high quality? And what makes data have low quality?

With the rising importance of data itself, these are not just scientific or economic questions but questions that are also discussed in a political context (Ghorbani & Zou, 2019; Kwon et al., 2021). The DASHBOARD Act, for example, a bill introduced in the American Congress in 2022, demands that companies “disclose to users what types of user data are collected, and the usage and value of that data” (H.R.7120 2022). Today, data is recognized as more of a property (Ghorbani & Zou, 2019), and as such is becoming an important part of business models (Ghorbani et al., 2020) and helps companies earn revenue (Jia et al., 2019). Thus, it is becoming essential to be able to assign a certain value to this property and compensate individuals or data owners for their data (Ghorbani & Zou, 2019).

In this report, we conduct a comparative analysis of four data valuation methods. Our goal is to investigate the strengths and weaknesses of each approach when applied to both regression and classification tasks. We will be focusing on the Leave-one-out method as well as three Shapley Value (SV) approximations, namely TMC-Shapley, Distributional Shapley and KNN-Surrogate Shapley Values. The main research questions we want to answer are (1) how well do these methods perform in valuating data in a classification and regression task? (2) how do they compare in performance optimization with respect to computation time?

This report is organized as follows. In the next section, two specific use-cases for data valuation are introduced. Next follows an overview of the existing literature in the domain. We then do an in-depth examination of the four data valuation methods and their respective implementations. We proceed with a detailed exploration of the datasets used to carry out the comparative analysis. After that, different evaluation tests are presented, showcasing the results afterwards. The penultimate section discusses the observed results from the evaluation tests and the overall important factors for data valuation. The last section concludes the report.

2. Data Valuation use-cases

2.1 Data marketplaces

The “commoditization of data” (Jia et al., 2018) has led to the creation of so-called “data marketplaces” where users can buy or sell data and consumers can receive payments from companies if their data is used (Sim et al., 2022. In a medical context, for example, patients could submit their medical records onto such a marketplace and analysts could pay to use this data in certain analyses or to train ML models (Jia et al., 2019). This ML context makes the question of how to value data especially interesting. Data used to train a ML model might come from many different data owners (or many different patients in the medical example). What is a fair compensation for every data owner in order to encourage the sharing and utilization of this data (Sim et al., 2022)? We want to thus find a

way to quantify the contribution of each data point (or subset of data points) to the overall model, so that data owners with large quantities of important data should ideally be compensated higher than data owners of very little or rather unimportant data points. This process is called data valuation (Jia et al., 2018).

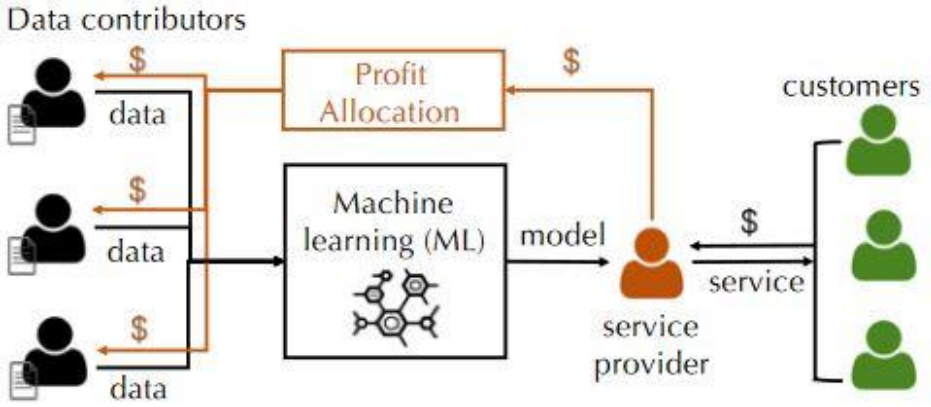


Fig.1 Visualization of a data marketplace (Jia et al., 2019)

2.2 Data Valuation in machine learning

Effective data valuation can have a large impact on many aspects of ML. It can increase the interpretability of created models, as we improve our understanding of how certain data points shape the result of the model (Rozemberczki et al., 2022). It can also greatly optimize data acquisition, as gathering or finding more high importance data can be very expensive. Data valuation can help to know where to focus limited resources (Ghorbani & Zou, 2019; Ghorbani et al., 2020). It can also improve overall model performance, as we understand which data points lead to good performance and which data points we might want to remove from the model as they hurt the performance instead (Sim et al., 2022; Tang et al., 2021).

In the following literature review, we will outline the theoretical background of data valuation in a ML context, compare data valuation methods used in current scientific literature, give practical examples of data valuation use-cases and finally present an outlook on data valuation topics that remain to be explored in greater detail.

Multiple papers show that the SV captures the actual importance of data points for models very well. Removing points with high SVs decreased model performance (Tang et al., 2021) while removing points with very low SVs either did not affect model performance at all (Tang et al., 2021) or even improved the performance of the model (Ghorbani & Zou, 2019). This way it is possible to, for example, find mislabelled data points, as those will be among the data points with the lowest overall value (Ghorbani & Zou, 2019; Tang et al., 2021). Another example in saving resources is illustrated when using auto labeling (which can create many mislabels) and letting experts review only those instances that are most likely mis-labeled (Jia et al., 2021).

Data valuation can also be used to improve data acquisition. Ghorbani & Zou (2019) show how they fit a regression model on the calculated SVs of the training points and then applied this model to a

new pool of patients in order to predict which patients would likely give the most valuable data in order to improve the performance of their model.

A different use could be when trying to adapt a model to new data, which can be especially useful when the real deployment data the model should be used upon is very rare or expensive to acquire. Training a model on available data that is just "close enough" but valuing these instances on a small test set of the real deployment data can help find those data points in the training set that lead to accurate predictions of the actual target data (Ghorbani & Zou, 2019).

Additionally, data valuation can improve the explainability of ML models, as Tang et al. (2021) show how they used data valuation in order to better understand what types of X-ray images were useful or harmful in predicting pneumonia cases. Other useful applications could include watermark removal in images (Jia et al., 2021) and data summarization, where one tries to select only a small subset of a massive dataset without losing much utility from models trained on it (Jia et al., 2021).

3. Theoretical background

3.1 Data Valuation principles

A data marketplace will have two types of actors: the sellers that provide training data instances (feature vectors with a corresponding label) and the buyers that want to use these instances to train ML models (Jia et al., 2018). The goal is thus often to find a way to fairly distribute payment between the sellers, which is analogous to distributing value between different training points. A supervised ML task requires three ingredients: a training dataset D, a learning algorithm A and a performance metric or utility function U that should have higher values for higher value models or datasets (Ghorbani & Zou, 2019; Sim et al., 2022). Data valuation is also dependent on these same three ingredients, changing any of them may change the determined value of a specific data point (Ghorbani & Zou, 2019). This seems intuitive for the learning algorithm and the performance metric, but it is important to understand that the value of a data point in D is also dependent on the other data points that are part of the dataset. Similar or redundant data should generally be worth less than more unique data (Sim et al., 2022).

Different data valuation strategies exist in order to use these ingredients to assign specific values to each data point. These strategies can have different properties (Sim et al., 2022) which will help us compare them. Key properties that are presented in scientific literature on the topic include:

1) **Low computational cost**: the total computation cost of the data valuation strategy should be "low enough to be efficient" (Sim et al., 2022). In a ML context, this can be very relevant, as evaluating a utility function usually requires the training of a model and is thus already computationally intensive – even more so if the valuation strategy requires repeated evaluations of the utility function (Jia et al., 2019).

2) **Fairness**: a fair data valuation strategy should accurately represent how much each owner's data has contributed to the value of the model (Sim et al., 2022) and be dependent on nothing else (Jia et al., 2018). This is captured by three sub-properties.

a. **Uselessness**: If including a certain subset D_i of the dataset D does not improve the performance score of any combination of subsets of D (also called a "coalition"), then D_i should be without value (Sim et al., 2022).

b. **Symmetry:** If including a subset D_i leads to the same improvement as including a different subset D_j to any coalition of D then D_i and D_j should receive the same value (Rozemberczki et al., 2022; Sim et al., 2022).

c. **Strict desirability:** Going beyond just symmetry, fairness requires subset D_i to receive a higher value than D_j if including D_i leads to a greater performance than including D_j for some coalitions, but the reverse is not true (so including D_j never leads to a greater performance than including D_i) (Sim et al., 2022).

3) **Group rationality**: this property states that a rational group of data owners are expected to distribute the full value gained by their coalitions. So, the value of the entire dataset should be distributed entirely between the data owners (Jia et al., 2018; Sim et al., 2022).

4) **Additivity**: if a ML model is used for different applications with different utility functions, the overall utility of a subset D_i should be the sum of the separate performance scores of this subset (Jia et al., 2018; Sim et al., 2022).

Based on these properties, we will now explore and compare commonly used and state-of-the-art data valuation methods and techniques.

3.2 Current Data Valuation approaches for machine learning

1) Leave-one-out method

The leave-one-out (LOO) method is a commonly used data valuation method in order to determine the importance of single points or data subsets (Tang et al., 2021). It is a very intuitive approach as it simply asks how much worse a model performs if we remove a subset (which can be a single data point) from the overall training set (Ghorbani & Zou, 2019). This approach has multiple problems however. First, it is computationally very expensive. If we are interested in valuating every data point in D, we need to train and evaluate the model |D| times (Jia et al., 2021; Sim et al., 2022). Secondly, while the LOO method can satisfy certain valuation properties as uselessness, symmetry, group rationality and additivity (Sim et al., 2022), it does not satisfy the strict desirability property as it does not look at all possible coalitions/subsets D but only ever at D - D_i . Thus, it violates the overall fairness property (Ghorbani & Zou, 2019; Rozemberczki et al., 2022; Sim et al., 2022).

2) Data Shapley – the game theory approach

The main question of data valuation ("how much is an individual data point worth") can also be approached using cooperative game theory (Ghorbani & Zou, 2019). Here, a cooperative game consists of a certain set of players, which for us can be the different data owners and their separate disjoint subsets of D, and a characteristic function that determines the value of certain coalitions of players (so a subset of all data owners), which in our context would be the performance metric or utility function of the model (Jia et al., 2019; Rozemberczki et al., 2022). The task in such a game is to find a way to distribute the collective value of a team across its individuals. This is very similar to

our data valuation objective. The SV is a key solution concept for such a cooperative game problem (Maleki et al., 2014; Sundararajan & Najmi, 2020).

Created by Lloyd Shapley (1953), the SV gives a way to uniquely divide the reward of a cooperation so that every player gets a fair share based on their contributions (Ghorbani & Zou, 2019). Based on the SV, the value attributed to subset D_i is its marginal contribution to every possible coalition C of D, where the marginal contribution is defined as the performance of the model using $C \cup D_i$ minus the performance of just using C (Sim et al., 2022). The SV is the only solution that satisfies the fairness, group rationality and additivity properties at the same time (Jia et al., 2018; Kwon et al, 2022; Sim et al., 2022). Because of this, it has been used and applied in many domains outside of ML (Maleki et al., 2014).

In a ML context, this method is often referred to as "Data Shapley" (Ghorbani & Zou, 2019) and seems to be the state-of-the-art method for data valuation used and explored currently (Ghorbani et al., 2020; Ghorbani & Zou, 2019; Jia et al., 2018, 2019, 2021; Kwon et al., 2021; Kwon & Zou, 2022; Liu et al., 2021; Rozemberczki et al., 2022; Sim et al., 2022; Sundararajan & Najmi 2020; Tang et al., 2021; Yan & Procaccia, 2021). Many studies also show that Data Shapley greatly outperforms the LOO method when it comes to finding useful valuations of individual data points or subsets (Jia et al., 2021; Kwon et al., 2022; Tang et al., 2021)

However, calculating SVs requires exponentially many evaluations of utility functions (as we need to evaluate the marginal contribution to every possible coalition of D) (Jia et al., 2018). This might be feasible in smaller cooperative games in other domains, but in a ML context we need to find ways to calculate SVs efficiently for millions or maybe even billions of data points – a scale that can be common in real world ML data valuation tasks (Jia et al, 2018). Current research focuses on different versions and approximations of Data SVs in order to try and satisfy the low computational cost property (Jia et al., 2018; Rozemberczki et al., 2022). The practical application of some of these approximations and versions will be the main topic of this research project.

4. Description and Implementation of the different Algorithms

4.1 Leave-One-Out (LOO)

The implementation of LOO is simple and straightforward; the value of the data owner k's dataset $D_k$is represented as the change in the performance metric of the output model after leaving out D_k from the overall aggregated data D_N. Formally, $\phi_k \triangleq v(N) - v(N \backslash \{k\})$, where v is the data value (Sim, Xu, and Low, 2022).

4.2 Truncated Monte Carlo Shapley

The truncated Monte Carlo approximation for the SV (TMC-Shapley) was first introduced by Ghorbani and Zou (2019). Given the three ingredients of a supervised ML task, their goal was to find an equitable measure of each data point, with respect to the learning algorithm and the performance metric used.

TMC-Shapley is composed of two components: the Monte Carlo Sampling technique followed by a truncation process. The Monte Carlo Sampling techniques for cooperative games were first proposed by Castro et al. (2009). It allows for approximation of the SV in linear time (Rozemberczki, B. et al., 2020), which is computationally more viable than calculating the exact SV. The sampling technique works by not considering all possible permutations of data points, instead it relies on random sampling of permutations. For each sampled permutation, the marginal contribution is computed for every new added data point, starting from the first. At the end of the process the marginal contributions of each data point is averaged, resulting in an estimated SV. The more permutations that are sampled, the closer the approximated SV will be to the true SV.

The truncation process can be likened to an early stopping procedure, with the goal of saving computational time. It works by "defining a "performance tolerance" based on the bootstrap variation in the predictive performance" (Ghorbani and Zou, 2019). The calculations of marginal contributions for a permutation are truncated whenever the predictive performance is within the performance tolerance. The data points that are truncated have their marginal contributions set to zero for that specific permutation.

In figure 2 the TMC-Shapley algorithm is described in pseudo code.

Algorithm 1 Truncated Monte Carlo Shapley

Input: Train data $D = \{1,\ldots,n\}$, learning algorithm $\mathscr{A}$, performance score V
Output: Shapley value of training points: $\phi_1,\ldots,\phi_n$
Initialize $\phi_i = 0$ for $i = 1,\ldots,n$ and $t = 0$
while Convergence criteria not met **do**
  $t \leftarrow t+1$
  π^t: Random permutation of train data points
  $v_0^t \leftarrow V(\emptyset,\mathscr{A})$
  for $j \in \{1,\ldots,n\}$ **do**
    if $|V(D) - v_{j-1}^t| <$ Performance Tolerance **then**
      $v_j^t = v_{j-1}^t$
    else
      $v_j^t \leftarrow V(\{\pi^t[1],\ldots,\pi^t[j]\},\mathscr{A})$
    end if
    $\phi_{\pi^t[j]} \leftarrow \frac{t-1}{t}\phi_{\pi^{t-1}[j]} + \frac{1}{t}(v_j^t - v_{j-1}^t)$
  end for
end while

Fig. 2 Pseudocode for the TMC-Shapley algorithm (Ghorbani and Zou, 2019)

The implementation of the TMC-Shapley algorithm used in this report largely mirrors the approach taken by Ghorbani and Zou (2019), with one notable distinction. In the Ghorbani and Zou implementation, a "truncation counter" keeps track of how often a data point fails to meet the desired performance tolerance level. If this counter exceeds a predefined limit (five by default), the computation of marginal contributions is stopped for this permutation, and subsequent data points are set to zero. As mentioned above, the performance tolerance was based on the bootstrapped variation in the predictive performance. In essence, if a data point underperforms in terms of its contribution, the truncation counter increases. If this underperformance recurs five times, the early stopping process is triggered.

The difference in the approach for this report was how to handle the performance tolerance. Rather than maintaining a truncation counter and evaluating the performance of an individual data point, we initially assessed the model's performance using the full training dataset. The early stopping was

triggered when the performance of the model trained on a subset of the data was within the performance tolerance of the full dataset performance.

Beyond the difference in the handling of the performance tolerance, we encountered a special case when using the TMC-Shapley algorithm for classification. Using the sklearn library for ML, it is required to have at least two distinct classes when training a model. However, with the first data point in the permutation, this isn't feasible, making it impossible to train a model and compute the marginal contribution. Depending on the class distribution of the dataset, this issue might persist for multiple iterations when adding one new data point each time. Consequently, when a subset finally has more than one class, the corresponding data point will be rewarded the entire marginal contribution up until that point. We decided to address this by splitting the marginal contribution evenly across all the previous data points in the permutation.

We encountered another problem not mentioned in the paper or the implementation by Ghorbani and Zou (2019) with using the TMC-Shapley algorithm for regression tasks. At the beginning of every permutation, there will be very few data points included in the training subset, which makes predictions on the validation dataset stochastic. This means that when we calculate the marginal contribution of a data point by subtracting the current iterations score from the previous iterations score, the marginal contribution could be extremely random. Using negative mean squared error as the metric, the score from one iteration to another could in some cases vary with more than one million, which would result in that particular data point getting a marginal contribution of one million. We found that starting with 2.5% of the training data in our subset before starting to calculate the marginal contributions significantly reduced the stochastic behavior.

We decided to run the TMC-Shapley algorithm for 1000 samples on the classification dataset with a performance tolerance level of 1%. For the regression dataset, we decided to run 100 samples and set the performance tolerance level to 5% as otherwise the computation time would be too great.

4.3 Distributional Shapley

The original algorithm, presented in "A Distributional Framework for Data Valuation" (Ghorbani et al., 2020) focuses on estimating the distributional SVs for data points within a given dataset. The algorithm, referred to as D-Shapley, is designed to provide unbiased and robust estimates for these values.

The algorithm works as follows. For each data point z in the input dataset Z, initialize the estimate for its distributional SV as $v_1(z)= 0$. This means that initially, we assume no contribution from any data point. Then we iterate T times. In each iteration, a random dataset S_t is sampled from the distribution D. D here is a distribution of the world ($\mathbb{Z}$). The size of S_t is chosen randomly from the set {1, 2, ..., m} based on the value of k ~ [m]. This simulates the process of randomly selecting a subset of data points from the distribution D.

For each data point z in the input dataset Z, compute the marginal contribution $\Delta_z U\ (S_t)$ of adding z to the sampled data set S_t. This is done by calculating the potential function U for the augmented set S_t

$\cup$ {z} and subtracting the potential of S_t. This step measures how much the addition of each point z contributes to the performance metric U. For each z, $\nu_t + 1(z)$ is updated using the previous estimate $\nu_t(z)$ and the new marginal contribution $\Delta_z U(S_t)$. The updated formula includes a weighting factor to ensure that the estimate becomes more accurate over time. After completing all iterations, the algorithm returns a set of pairs $\{(z, \nu_T(z))\}$ where z is a data point in the input dataset Z, and $\nu_T(z)$ is the estimated distributional SV for that point after T iterations.

D-Shapley

Fix: *potential* $U : \mathcal{Z}^* \to [0, 1]$*; distribution* $\mathcal{D}$*;* $m \in \mathbb{N}$
Given: *data set* $Z \subseteq \mathcal{Z}$ *to valuate; # iterations* $T \in \mathbb{N}$

for $z \in Z$ **do**
  $\nu_1(z) \leftarrow 0$ `// initialize estimates`
end for
for $t = 1, \ldots, T$ **do**
  Sample $S_t \sim \mathcal{D}^{k-1}$ for $k \sim [m]$
  for $z \in Z$ **do**
    $\Delta_z U(S_t) \leftarrow U(S_t \cup \{z\}) - U(S_t)$
    $\nu_{t+1}(z) \leftarrow \frac{1}{t} \cdot \Delta_z U(S_t) + \frac{t-1}{t} \cdot \nu_t(z)$
      `// update unbiased estimate`
  end for
end for
return $\{(z, \nu_T(z)) : z \in Z\}$

Fig. 3 Pseudocode for the D-Shapley algorithm.

In our implementation, *D* defines the distribution of the data available (*Z*), while the original algorithm assume that *D* is a distribution of the $\mathbb{Z}$. We assume that the distribution ($\mathbb{Z}$) is the same from how we estimate it from Z**.** The reason for this is that we had a small dataset. This approach may introduce a small bias into the valuation of data points compared to the original D-shapley (Ghorbani et al., 2020). The reason is that when we randomly select a sample of data points from *D* and evaluate the change in performance, z can be in the dataset. To avoid this, we check if $z \in S_t$, and if that is the case, we do not use that iteration.

4.4 KNN Surrogate Shapley values

As we have seen in the data valuation methods described above, reducing the needed computation time for generating SVs often relies on approximations. A different approach is to find efficient ways to calculate SVs for a given fixed family of ML models (Jia et al., 2018). Jia et al. (2021) show, for example, that calculating SVs for a K-Nearest-Neighbour (KNN) Classifier can be done in

O(|D|*log(|D|)) time for all data points in D. This relies on the fact that a KNN-Model does not require a real training phase and the difference in utility gained by two different points “translates linearly to the difference in the respective SVs”, allowing for recursive calculation of SVs (Jia et al., 2018). The main idea behind KNN Surrogate Shapley values is thus to calculate the exact SVs for a KNN model (either classification or regression) and use these values as proxies for the SVs of the more complex model. The calculated KNN SVs will satisfy the key concepts defined above (fairness, group

rationality and additivity) (Jia et al., 2018), but there is no guarantee that these are satisfied for the more complex model.

For a classification task, Jia et al. (2018) describe the algorithm to calculate the exact KNN SVs with following pseudocode:

Algorithm 1: Exact algorithm for calculating the SV for an unweighted KNN classifier.

input : Training data $D = \{(x_i, y_i)\}_{i=1}^{N}$, test data $D_{test} = \{(x_{test,i}, y_{test,i})\}_{i=1}^{N_{test}}$
output: The SV $\{s_i\}_{i=1}^{N}$

for $j \leftarrow 1$ **to** N_{test} **do**
  $(\alpha_1, ..., \alpha_N) \leftarrow$ Indices of training data in an ascending order using $d(\cdot, x_{test})$;
  $s_{j,\alpha_N} \leftarrow \frac{\mathbb{1}[y_{\alpha_N} = y_{test}]}{N}$;
  for $i \leftarrow N-1$ **to** 1 **do**
    $s_{j,\alpha_i} \leftarrow s_{j,\alpha_{i+1}} + \frac{\mathbb{1}[y_{\alpha_i} = y_{test,j}] - \mathbb{1}[y_{\alpha_{i+1}} = y_{test,j}]}{K} \frac{\min\{K,i\}}{i}$;
  end
end
for $i \leftarrow 1$ **to** N **do**
  $s_i \leftarrow \frac{1}{N_{test}} \sum_{j=1}^{N_{test}} s_{j,i}$;
end

Fig. 4 Pseudocode for the exact KNN SV for classification.

The exact time constraints with multiple test points is O(N log(N) N_{test}), as the training data needs to be sorted for every test point (Jia et al., 2018).

The same paper also describes the algorithm required for calculating the exact SVs for KNN-Regressor models as follows (Jia et al., 2018):

$$s_{\alpha_N} = -\frac{K-1}{NK} y_{\alpha_N} \left[\frac{1}{K} y_{\alpha_N} - 2y_{test} + \frac{1}{N-1} \sum_{l \in I \setminus \{N\}} y_{\alpha_l} \right] - \frac{1}{N} \left[\frac{1}{K} y_{\alpha_N} - y_{test} \right]^2 \tag{62}$$

$$s_{\alpha_i} = s_{\alpha_{i+1}} + \frac{1}{K} (y_{\alpha_{i+1}} - y_{\alpha_i}) \frac{\min\{K,i\}}{i} \left(\frac{1}{K} \sum_{l=1}^{N} A_i^{(l)} y_{\alpha_l} - 2y_{test} \right) \tag{63}$$

where

$$A_i^{(l)} = \begin{cases} \frac{\min\{K-1, i-1\}}{i-1} & \text{if } 1 \leqslant l \leqslant i-1 \\ 1 & \text{if } l \in \{i, i+1\} \\ \frac{\min\{K,l-1\} \min\{K-1,l-2\} i}{(l-1)(l-2) \min\{K,i\}} & \text{if } i+2 \leqslant l \leqslant N \end{cases} \tag{64}$$

Fig. 5 Pseudocode for the exact KNN SV for regression.

According to Jia et al. (2018), the time complexity for this algorithm should be the same as for the classification algorithm: O(N log(N) N_{test}). However, according to our understanding and implementation, this is not actually the case. The required sum in (63) is dependent both on the current test point and the current training point in question, resulting in an actual time complexity of O(N^2 log(N) N_{test}). Either our understanding (and subsequent implementation) of the presented

algorithm is wrong or the algorithm itself is. This has to be kept in mind when analyzing the results presented later on: the KNN Surrogate SVs for the regression task are likely not reliable.

5. Methodological Choices: Datasets and Models

5.1 Regression task

5.1.1 Dataset

The dataset we chose for the regression task is "Seoul Bike Sharing Demand Dataset" from the UC Irvine Machine Learning Repository. It contains the number of public bicycles rented per hour in the Seoul Bike Sharing System along with corresponding weather data and holiday information as shown below. As for the date (year-month-day) variable, we converted it into four variables; month of the year (January, February…), day of the month (1st, 2nd, … 31st), day of the week (Monday, Tuesday, ...), and is_weekend (weekday or weekend). The duration of the dataset is one year between December 2017 to November 2018, resulting in 24 hours x 365 days = 8760 data points. Using this dataset, we worked on a regression task to predict the number of bikes rented based on the datetime, holiday, and weather information (see Appendix A).

5.1.2 Feature Selection

Before training a model, we looked into whether there are any features that are better to be removed.

Firstly, we looked into what the variable "Functional Day" indicates and whether it is informational or not as we could not tell what this variable represented from the description. After looking into the difference between "functional day" data points and "non-functional day" data points, we concluded that the variable simply indicates hours when no bikes were rented due to out-of-service since the number of bikes rented were always 0 for "non-functional" data points. We also did not observe any apparent reason for the service being non-functional. Therefore, we decided to remove all the "non-functional day" data points and remove the variable "Functional Day" as these would rather confuse the model when predicting as they are not informational.

Secondly, we checked correlations among features to see if there are any highly correlated features. As a result, we found that "Temperature" and "Dew Point Temperature" were highly correlated (correlation coefficient of 0.91). In order to avoid multicollinearity, we decided to remove "Dew Point Temperature". The reason for removing "Dew Point Temperature" instead of "Temperature" was because "Temperature" was better correlated to the target variable.

5.1.3 Data Preparation

After removing the "Dew Point Temperature" variable and the data points that were labeled as "non-functional" according to the "Functional Day" variable, we shuffled the dataset and split it into train, validation, and test datasets with the ratio of 80%, 10%, 10% respectively. Thereafter, we validated that the three datasets share the same distribution by checking a Q-Q plot for each variable. We then used the train dataset and the validation dataset to calculate data values and we tested whether the data values were reasonable using the test dataset.

5.1.4 Model

For the regression task, we decided to use a gradient boosting regressor. Before choosing this model, we trained and compared the performance of linear regression, decision tree regressor, random forest regressor, knn-regressor, and gradient boosting regressor. The best performing model among these was a gradient boosting regressor with hyper-parameters (learning rate, max depth, number of estimators) tuned via a grid-search, and therefore, we decided to use the model for the data valuation.

5.2 Classification task

5.2.1 Dataset

The dataset we chose for the classification task is the "Chest X-Ray Images (Pneumonia)" dataset from Kaggle[1]. It contains 5,863 X-Ray images in JPEG format, divided into the categories "pneumonia" and "normal". The dataset is unbalanced regarding these categories, with 73% of the images depicting patients with pneumonia and only 27% depicting healthy patients (see Appendix A for example images). This Kaggle dataset is of a very high quality, with all the images screened to remove low quality x-rays and the diagnostic labels graded by multiple expert physicians.

We decided on this dataset as a recent paper by Tang et al. (2021) utilized a very similar approach by applying data valuation techniques on a large dataset of x-ray images for a classification task. This would allow us to use a similar approach in order to compare the different data valuation methods and SV approximations. In this paper, Tang et al. (2021) utilize CheXNet, a pre-trained 121-layer convolutional neural network (CNN) trained on over 100,000 x-ray images with 14 different diseases (Rajpurkar et al. 2017), to extract features from the images (by removing the last fully-connected layer) and then train a logistic regression model on these features to predict if the patient has pneumonia or not.

5.2.2 Feature selection and reduction

As we could not guarantee that the CheXNet CNN did not use any x-ray images in our dataset from Kaggle, we decided to utilize the pre-trained Densenet121 model form pytorch that was trained on the Imagenet 1K (trained to recognize a wide variety of objects and scenes but not x-ray image diagnostics) as our feature extractor in order to avoid data leakage. Similar to Tang et al. (2021), we removed the final fully connected layer from the model, which results in 65,536 features being extracted from a single image. These features would then be used as input for a Logistic Regression model with 1,000 maximum iterations.

As we suspected that computation time would be a limiting factor for our data valuation methods, we decided to run a Principle Component Analysis (PCA) algorithm on the extracted features to reduce their dimensions to a much smaller level before using them in the Logistic Regression model. Fitting the PCA to the standardized extracted features showed that about 700 features are necessary to capture approximately 50% of the original variance in the data (Appendix B, Fig. 1). This would likely still lead to very long computation times, so we instead decided to compare the actual performance of the Logistic Regression model on the data reduced to different dimensions. Fig. 2 in

[1] https://www.kaggle.com/datasets/paultimothymooney/chest-xray-pneumonia

Appendix B shows the F1-score and accuracy based on the number of dimensions the features were reduced to using the PCA. As can be seen, the performance takes a large jump when increasing the number of features to 10, after which the performance increase is more gradual. Because of this, we decided to reduce the number of dimensions of our extracted features to only 10. This is a very low number and captures only approximately 11% of the original variance. However, as our goal in this project is to compare the data valuation methods, our priority was on reducing computation time as much as possible so that these methods are able to produce useful results. As the performance drop from using only 10 dimensions was relatively small (less than a 0.02 drop in F1-score), we decided this was a trade-off we were willing to take.

5.2.3 Data preparation

As described above, the torchvision densenet121 model with default weights was used to extract features from the images. For this, the images needed to be compressed to a size of 256x256 pixels. In order to reduce the computation time later on, we decided to only use a random subset of all available images, performing a train-validation-test split of the dataset resulting in 2000 training images and 500 validation and test images respectively. The ratio of pneumonia images in the training, validation and training sets was 0.72, 0.71 and 0.71 respectively.

As the dataset had a very high quality, we flipped the label of 1% of the training instances (a total of 20 images), so that our data valuation methods had clear “bad” data to identify. Additionally, we created a version of the training data that also included low to very high levels of noise in the training images. We added increasing levels of noise to the training image pixels (before feature extraction), with 10% of the images receiving no noise, 10% receiving 0.1 standard deviations of noise, 10% receiving 0.2 standard deviations of noise and so on with the highest level of noise being 0.9 standard deviations added to the pixels. Examples of these noisy images can be seen in Appendix A. In this case, the mislabeled images were all taken from the images without added noise.

The extracted features from all training, validation and test images were then reduced to 10 dimensions using the PCA described above. The final result was two different training sets (one original and one with noise and mislabels), a validation set and a test set.

6. Methods for evaluation

We conduct three distinct tests to assess the various data valuation methods against one another. These tests are described further in the following sections.

6.1 Removing data points

After calculating the SV for each method, we identify which data points get the lowest and highest SV and remove a fraction of either the high valued-, low valued- or random data points. The model is then retrained on the remaining subset of data with the idea of seeing an increase in performance having removed data points with low SV, a decrease in performance after removing data points with high SV and no particular change in performance removing random data points. By removing random data points, we can assess the impact of removing both low and high SV data points, ensuring that the effects we have observed were not due to randomness.

The performance of the retrained model will be evaluated on a held-out test set by using the F1-Score metric for classification and the r^2 value for regression.

6.2 Mislabeled data

By intentionally mislabeling 20 of the images (1% of the data) in the classification dataset, we could evaluate how different methods assign SV to mislabeled data. A good data valuation method should in theory assign a low SV to mislabeled images as they should not improve performance in any way.

For the regression dataset, we use the variable 'non-functional days' as "mislabel", which adds 295 data points to the dataset where the target variable is zero irrespective of other features..

6.3 Correlation and Computation Time

To compare the calculated SVs between the methods, we will conduct a correlation analysis using Pearson correlation coefficient. Additionally, the computation time for the SV calculation by each method will be recorded.

7. Results

7.1 Removing data points

Classification - Original Dataset

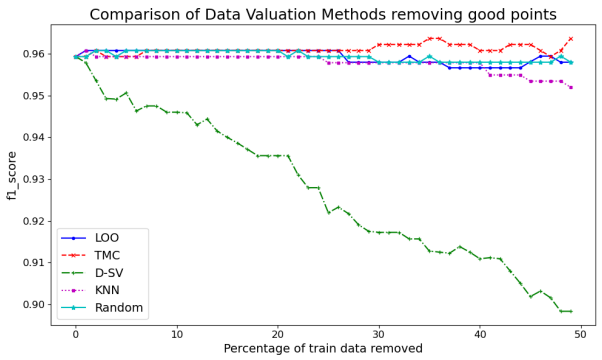


Fig. 6

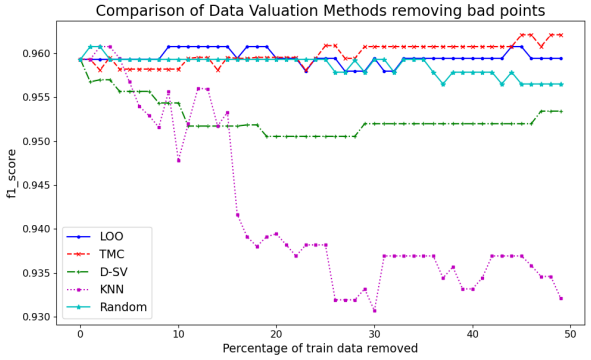


Fig. 7

Removing high valued data points (*fig. 6*) based on the LOO, TMC and KNN methods, the F1-score remains stable. Removing high valued data points based on the D-Shapley method, the F1-Score

drops steadily, reaching 0.90 after having removed 50% of the training data. Removing random data points the F1-score is stable, mirroring the LOO, TMC and KNN performance.

Removing low valued data points (*fig. 7)* based on the LOO and TMC method, the F1-score remains stable. For the KNN method, the F1-score improves slightly after removing 5% of the training data and then it drops to a low of 0.93 after removing 30% of the training data. For the D-Shapley method the F1-score drops slightly to 0.95.

Classification - Noisy Dataset

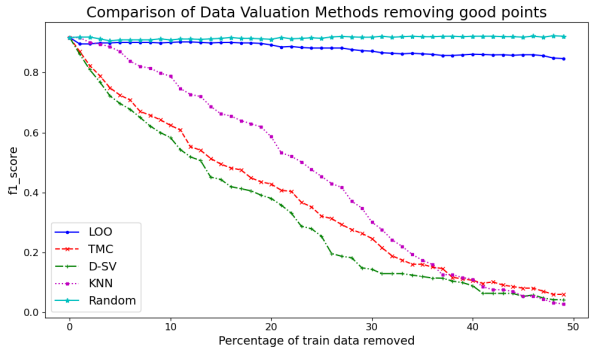


Fig. 8

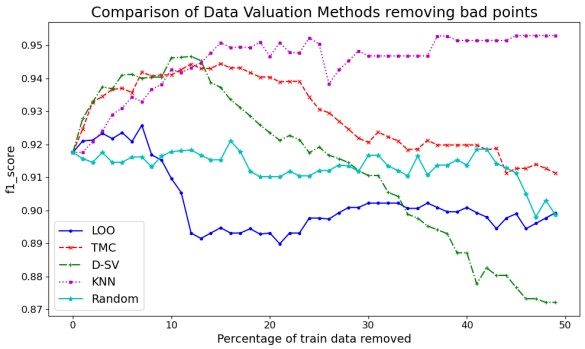


Fig. 9

Removing high valued points (*fig. 8)* from the noisy dataset based on the LOO method does not affect the F1-score, mirroring the effect of removing random data points. When removing points based on TMC, D-Shapley and KNN methods, the F1-score drops quickly and steadily, until it reaches zero after removing 50% of the training data.

When low value data points are removed (*fig. 9)* we see an increase in F1-score for all the methods, but mostly for TMC, D-Shapley, and KNN. After more than 10-20% of data is removed, both TMC and D-Shapley are decreasing in F1-score, while the performance drops quickly after removing 10% of the low value data based on LOO-values. Again, removing random data points does affect the F1-score.

Regression - Original Dataset

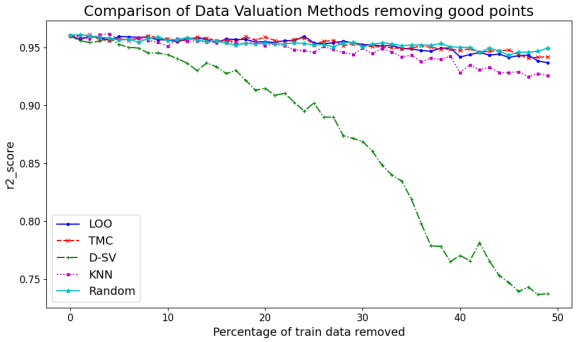


Fig. 10

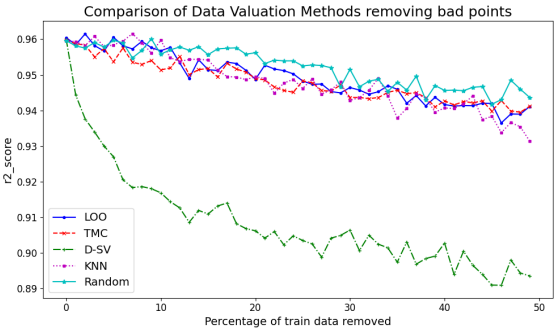


Fig. 11

Removing high or low valued data points (fig 10 & 11) based on the LOO, TMC and KNN method does not have a large effect on the R2 score and is similar to removing random data points. Removing data points based on the D-Shapley method has a clear negative effect on the R2 score in both cases, although the performance drop is larger when removing high value points than when removing low value points.

Regression - nonFunc Dataset

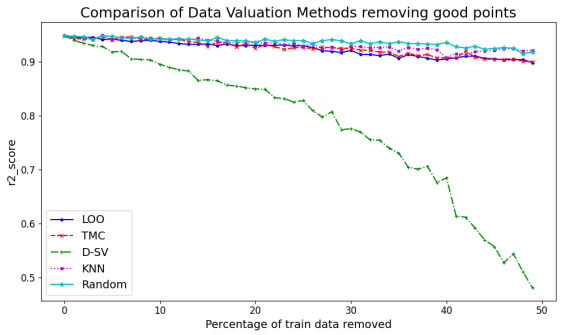


Fig. 12

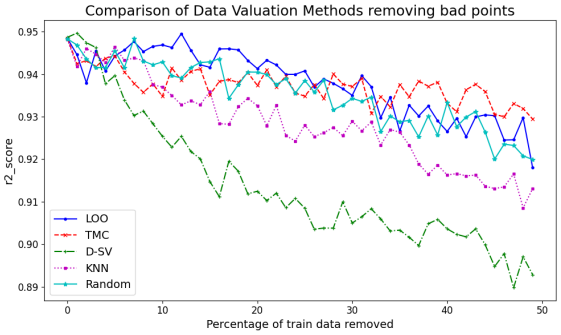


Fig. 13

The results are similar for the noisy dataset, except that the D-Shapley method reaches a lower R2 score of 0.5 when removing good points.

7.2 Finding mislabeled- or nonFunc data

In table 1 we present how the methods performed on finding mislabeled images for the classification dataset. First, how many of the mislabeled points were in the bottom 1%, i.e. the 1% of data points with lowest SV. Second, we look at how many mislabeled data points were in the bottom 10%.

Method	Mislabels in bottom 1% SV	Mislabels in bottom 10% SV
LOO	0 (0%)*	0 (0%)*
TMC	11 (55%)	17 (85%)
D-Shapley	6 (30%)	12 (60%)
KNN	12 (60%)	19 (95%)

Table 1 - Amount of mislabeled images in the 1%/10% bottom SV

KNN performed best with 12 of the mislabeled images in the bottom 1% of the SVs, closely followed by TMC which had 11 images in the bottom 1% SVs. LOO did not find any, which is because almost all data points have a marginal contribution of 0. A scatterplot of the SV from each of the methods can be found in Appendix C.

In table 2 we present how the methods performed on handling the nonFunc days. First by presenting how many of the nonFunc was given in the bottom 5%, which is the 5% of data points with lowest SV. Second, presenting how many of the nonFunc was given in the bottom 10%.

Method	nonFunc in bottom 5% SV	nonFunc in bottom 20% SV
LOO	5 (2%)	31 (11%)
TMC	45 (15%)	147 (50%)
D-Shapley	208 (71%)	284 (96%)
KNN	12 (4%)	50 (17%)

Table 2 - Amount of nonFunc data points in the 5%/20% bottom SV

The D-Shapley method performed best at giving data points with nonFunc a low SV. Again, LOO did not perform well.

7.3 Correlation and Computation Time

In order to compare the similarity between the different valuation methods, we calculated the Pearson correlation between the LOO and SV values. The different correlation matrices can be found in Appendix C. The most important observations will be described in the following.

For the original X-Ray data (without noise), only the TMC-Shapley and KNN-Shapley Values had a noteworthy correlation of 0.47. Regarding the noisy training data, the TMC-Shapley and D-Shapley values were highly correlated with each other (0.84) while also having a noteworthy correlation with the KNN-SVs (0.51 and 0.56 respectively). The LOO-Values are not correlated with any of the other values.

The correlations between the SVs are less pronounced for the regression dataset. Noteworthy correlations when using the original data without nonFunc days are between the TMC-Shapley and D-Shapley values (0.41) and TMC-Shapley and LOO-values (0.37). Similar results occur when including the nonFunc days in the training data set, with the correlation coefficient between TMC-Shapley and D-Shapley being 0.32 and between TMC-Shapley and LOO-values being 0.54. The KNN-Shapley values are not correlated with any of the other regression valuation methods. This fits to our skepticism regarding the utilized algorithm (or implementation) for KNN-Regression SVs. The graphs above have also shown that the regression KNN-SVs behave similarly to random values for data points, further showing that these valuations do not seem to be reliable for regression tasks.

Computation times in minutes for the methods can be found in table three and table four.

Method	Original	Noise	Workers
LOO	0.41	0.28	1
TMC	2.0	2.0	32
D-Shap	16.9	10.7	32
KNN-Surrogate	0.08	0.08	1

Table 3 - Computation time for the classification dataset

Method	Original	With nonFunc	Workers
LOO	28.16	31.98	32
TMC	1200.0	1400.0	32
D-Shap	781.4	819.3	32
KNN-Surrogate	113.6	122.8	32

Table 4 - Computation time for the regression dataset

8. Discussion

The results from the classification task are close to our expectations and show how data valuation methods can be very useful in certain settings. While the effect on model performance when removing high and low value data is relatively low for the original non-noisy dataset (even though D-Shapley shows a clear performance drop when removing high value data and KNN-Shapley shows an unexpected slight performance drop when removing low value data), all of TMC-Shapley, D-Shapley and KNN-Shapley are very effective at identifying the mislabeled training data points. Especially the KNN Surrogate method identifies these instances very efficiently, as its computation time is extremely low in comparison to TMC and D-Shapley. The results also show that the Logistic Regression model seems to have more than enough good data to work with to compensate for the few mislabeled instances, as model performance does not increase when removing low value data points.

This is very different when looking at the values for the noisy training data. The amount of useful training instances was drastically reduced, and removing high value points based on TMC, D-Shapley or KNN-Shapley reduces model performance drastically, as only very noisy data points remain. When removing low value data points based on these three methods, we can clearly see the model performance increase. It even gets close to the model performance of the original dataset without any noisy training images, which is very impressive when keeping in mind that only 10% of the original non-noisy images remain.

The LOO-method did not produce useful valuations for the classification training images in either case (original and noisy). This however, was to be expected. Removing a single image from 2000 training images is unlikely to influence the predictions a lot (if at all), resulting in most training instances being valued at or close to 0. This suggests that LOO is unable to find redundant points. Performance differences when removing certain data points and the identification of mislabeled instances is thus basically the same as random removals or guesses.

In the results for the regression dataset, all of the LOO, TMC and KNN approaches did not produce useful valuations. When removing high and low value data points, these methods all perform very similarly to removing random data points. Of the three, only the TMC approach identifies a reasonable amount of the nonFunc days as low value data points.

In comparison, the D-Shapley approach performs a lot better, even if the results are not completely aligned with expectations. When removing high and low value data points based on D-Shapley values in the dataset without nonFunc days, the performance drops in both cases. The performance drop

when removing high value data points is a lot larger than when removing low value data points, which shows that the approach succeeds in valuing more important data points higher than less important points. However, the performance drop when removing low value data points is larger than when removing random data points. We believe this might be because the D-Shapley approach values similar data points similarly. While each single point may have low importance, removing several points based on their value could lead to removing a lot of similar data points, like for example all points from Sunday evenings. This would result in the model performing a lot worse, as it loses all information about Sunday evenings.

Regarding the noisy dataset, the D-Shapley performs very well at identifying the NonFunc days which are bad for the model. This effect can actually be seen in the graph when removing low value data points, as the performance for D-Shapley increases slightly when removing 5% of the lowest value training points, which include 71% of the NonFunc days.

9. Conclusion

Data valuation is not guaranteed to improve the performance of a machine learning model. If we have a dataset that is noisy (which can often be the case in real-world situations), data valuation methods can be used in improving model performance by enabling the identification and removal of noisy, less informative, or mislabeled data points. Very clean datasets might not benefit as much, as all available data points carry useful information.

Computation time is a relevant factor when deciding what method of data valuation to use. The D-Shapley method performed the best over the different tasks and datasets, but also required a lot of computation time. Similarly, the TMC-Shapley approach would have likely produced better results if more computational resources were available for additional samples. If computation time is limited, the KNN-Surrogate approach can produce data valuations for classification tasks very efficiently, especially in regards to identifying potentially mislabeled data points. While the computation of the LOO values was also very quick, they did not prove to be useful valuations, both in performance difference and for identifying mislabels.

Future study in this field could focus on fine-tuning the algorithms to regression tasks, as these did not perform as well as for the classification task and required some adaptations in the implemented algorithms. Additionally, studying the different valuations of the data points in more detail is required as understanding what kind of points received high or low values can help to improve the interpretations of our results.

References

Castro, J., Gómez, D., & Tejada, J. (2009). Polynomial calculation of the Shapley value based on sampling. Computers & Operations Research, 36(5), 1726-1730.

Ghorbani, A., Kim, M. P., & Zou, J. (2020). *A Distributional Framework for Data Valuation* (arXiv:2002.12334). arXiv. https://doi.org/10.48550/arXiv.2002.12334

Ghorbani, A., & Zou, J. (2019). *Data Shapley: Equitable Valuation of Data for Machine Learning* (arXiv:1904.02868). arXiv. https://doi.org/10.48550/arXiv.1904.02868

H.R.7120 – 117th Congress (2022): DASHBOARD ACT of 2022, Bill Summary. https://www.congress.gov/bill/117th-congress/house-bill/7120?s=1&r=7

Jia, R., Dao, D., Wang, B., Hubis, F. A., Gurel, N. M., Li, B., Zhang, C., Spanos, C., & Song, D. (2018). Efficient task specific data valuation for nearest neighbor algorithms: 45th International Conference on Very Large Data Bases, VLDB 2019. *Proceedings of the VLDB Endowment*, *12*(11), 1610–1623. https://doi.org/10.14778/3342263.3342637

Jia, R., Dao, D., Wang, B., Hubis, F. A., Hynes, N., Gürel, N. M., Li, B., Zhang, C., Song, D., & Spanos, C. J. (2019). Towards Efficient Data Valuation Based on the Shapley Value. *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, 1167–1176. https://proceedings.mlr.press/v89/jia19a.html

Jia, R., Wu, F., Sun, X., Xu, J., Dao, D., Kailkhura, B., Zhang, C., Li, B., & Song, D. (2021). *Scalability vs. Utility: Do We Have to Sacrifice One for the Other in Data Importance Quantification?* (arXiv:1911.07128). arXiv. https://doi.org/10.48550/arXiv.1911.07128

Kwon, Y., Rivas, M. A., & Zou, J. (2021). Efficient Computation and Analysis of Distributional Shapley Values. *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, 793–801. https://proceedings.mlr.press/v130/kwon21a.html

Kwon, Y., & Zou, J. (2022). *Beta Shapley: A Unified and Noise-reduced Data Valuation Framework for Machine Learning* (arXiv:2110.14049). arXiv. https://doi.org/10.48550/arXiv.2110.14049

Liu, Z., Chen, Y., Yu, H., Liu, Y., & Cui, L. (2021). *GTG-Shapley: Efficient and Accurate Participant Contribution Evaluation in Federated Learning* (arXiv:2109.02053). arXiv. https://doi.org/10.48550/arXiv.2109.02053

Maleki, S., Tran-Thanh, L., Hines, G., Rahwan, T., & Rogers, A. (2014). *Bounding the Estimation Error of Sampling-based Shapley Value Approximation* (arXiv:1306.4265). arXiv. https://doi.org/10.48550/arXiv.1306.4265

Rozemberczki, B., Watson, L., Bayer, P., Yang, H.-T., Kiss, O., Nilsson, S., & Sarkar, R. (2022). *The Shapley Value in Machine Learning* (arXiv:2202.05594). arXiv. https://doi.org/10.48550/arXiv.2202.05594

Shapley, L. S. (1953). 17. A Value for n-Person Games. In H. W. Kuhn & A. W. Tucker (Eds.), *Contributions to the Theory of Games (AM-28), Volume II* (pp. 307–318). Princeton University Press. https://doi.org/10.1515/9781400881970-018

Sim, R. H. L., Xu, X., & Low, B. K. H. (2022). Data Valuation in Machine Learning: "Ingredients", Strategies, and Open Challenges. *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence*, 5607–5614. https://doi.org/10.24963/ijcai.2022/782

Sundararajan, M., & Najmi, A. (2020). *The many Shapley values for model explanation* (arXiv:1908.08474). arXiv. https://doi.org/10.48550/arXiv.1908.08474

Tang, S., Ghorbani, A., Yamashita, R., Rehman, S., Dunnmon, J. A., Zou, J., & Rubin, D. L. (2021). Data valuation for medical imaging using Shapley value and application to a large-scale chest X-ray dataset. *Scientific Reports*, *11*(1), Article 1. https://doi.org/10.1038/s41598-021-87762-2

Yan, T., & Procaccia, A. D. (2021). If You Like Shapley Then You'll Love the Core. *Proceedings of the AAAI Conference on Artificial Intelligence*, *35*(6), Article 6. https://doi.org/10.1609/aaai.v35i6.16721

Appendix A

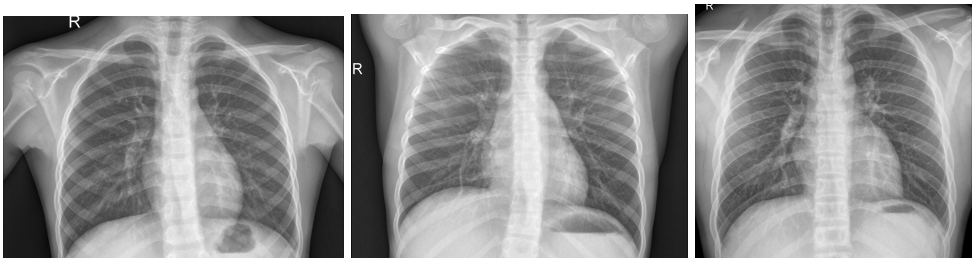
	Variables
1	Rented Bike count
2	Date (year-month-day)
	=> converted into • month of the year • day of the month • day of the week • is_weekend
3	Hour
4	Holiday (holiday or not)
5	Functional Day (functional hour or not)
6	Temperature (℃)
7	Humidity (%)
8	Wind Speed (m/s)
9	Visibility (10m)
10	Dew point temperature (℃)
11	Solar radiation (MJ/m2)

12	Rainfall (mm)
13	Snowfall (cm)
14	Seasons (winter, spring, summer, autumn)

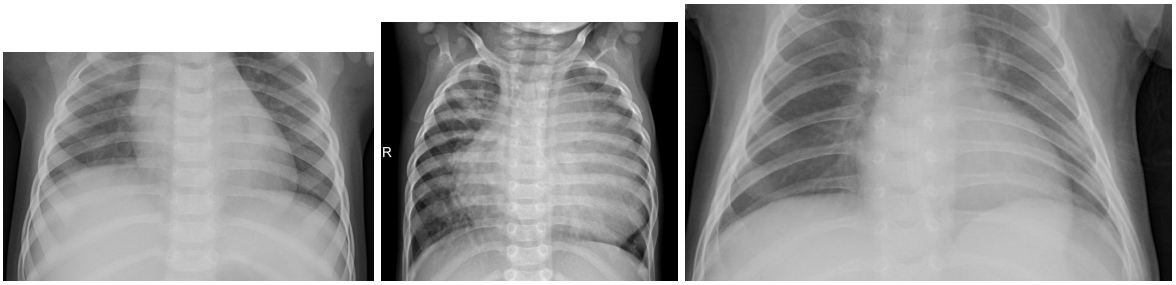
Table 1. Variables in "Seoul Bike Sharing Demand Dataset"

Example images of 'Pneumonia' and 'Normal'.

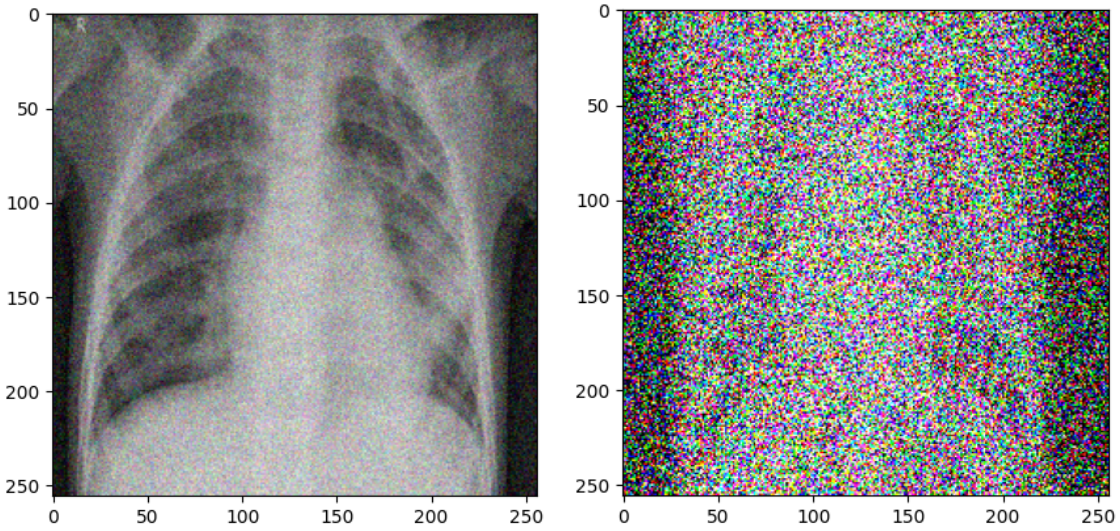
Normal image examples:



Pneumonia image examples:



Noisy image examples with 10% and 90% standard deviations of noise:


0
50
100
150
200
250
0
50
100
150
200
250
0
50
100
150
200
250
0
50
100
150
200
250

Appendix B

Fig. 1: Principal Component Analysis (PCA) of classification data set

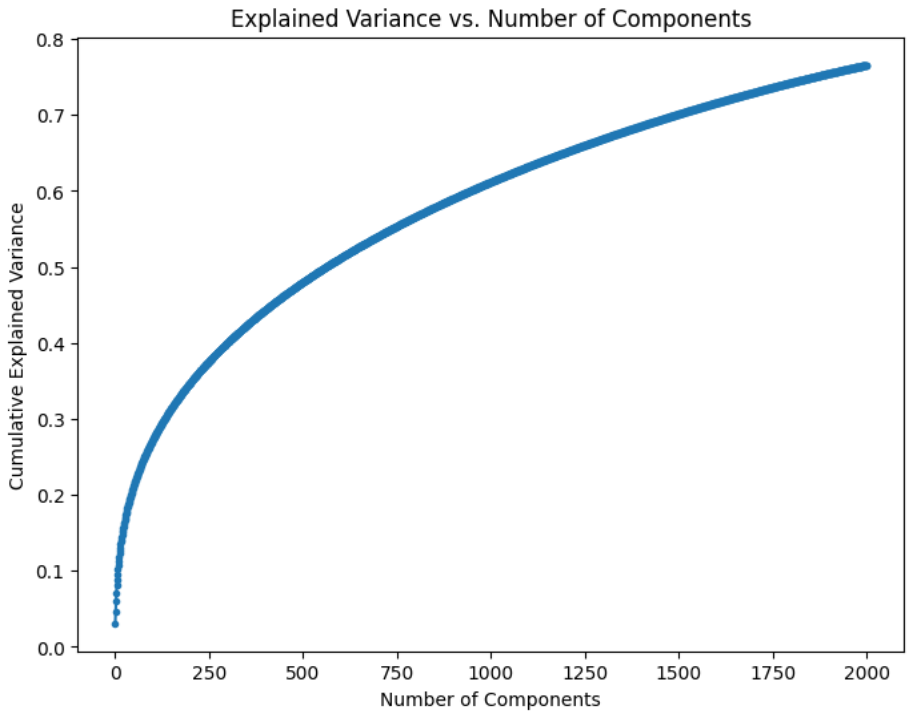
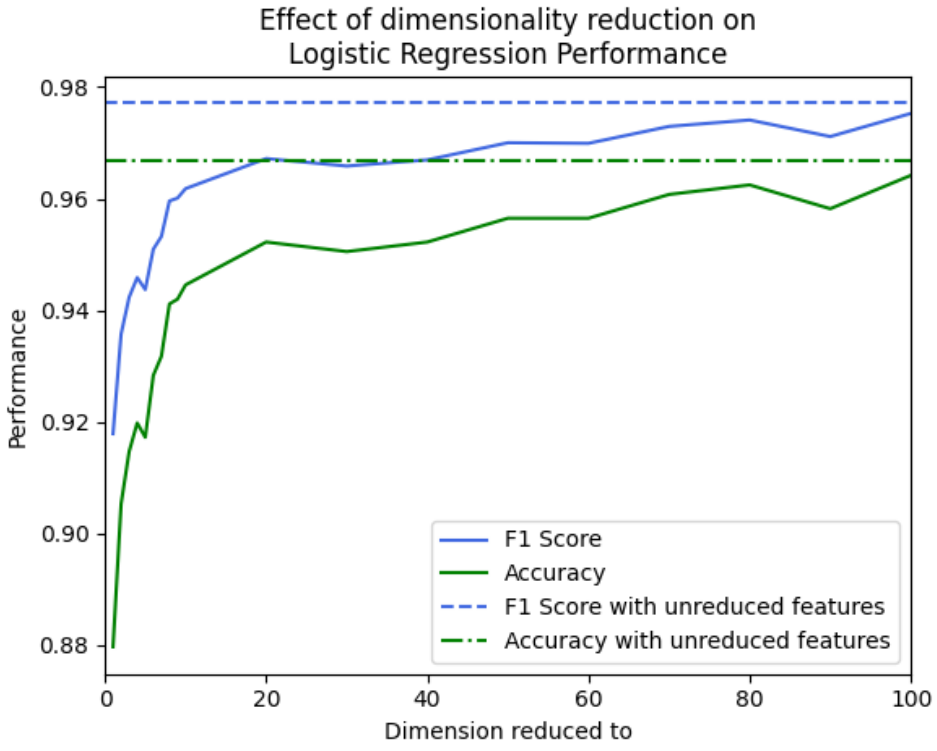
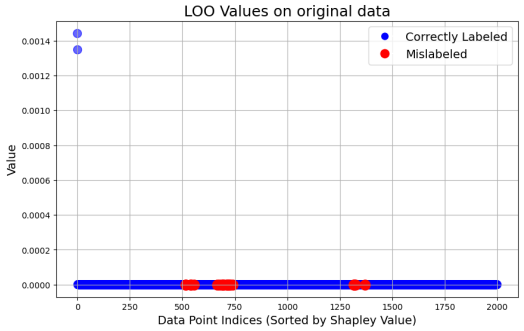


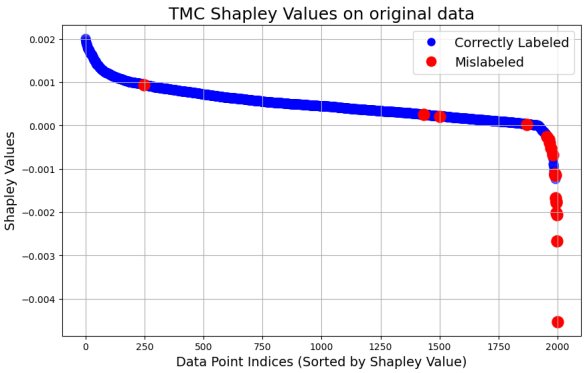
Fig. 2: Effect of dimensionality reduction on Logistic Regression Performance

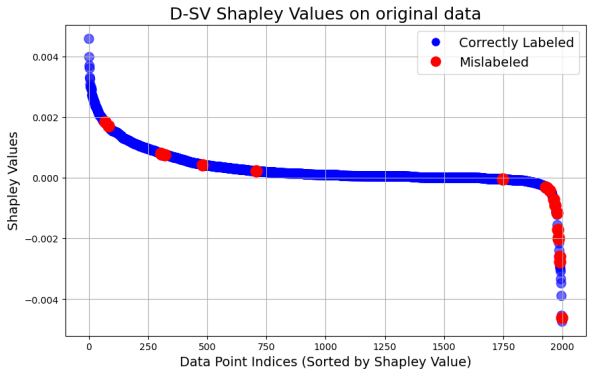


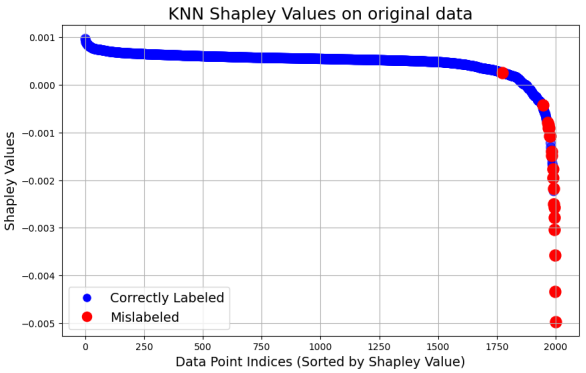
Appendix C

Classification SV - Original dataset

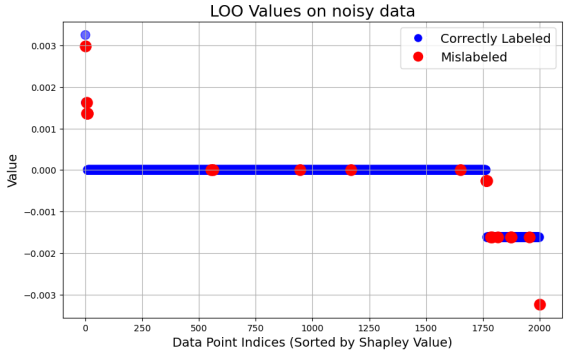


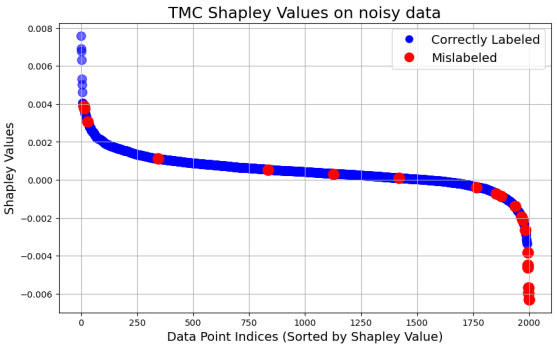


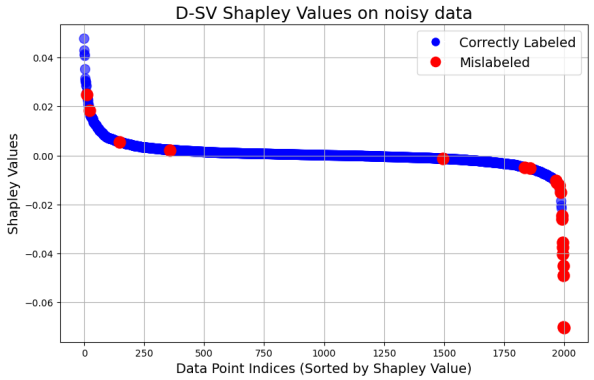


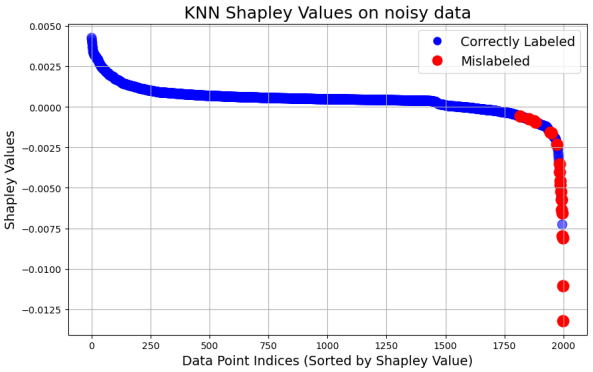


Classification SV - Noisy dataset

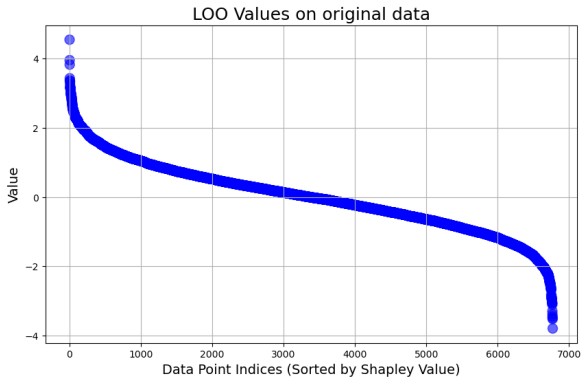


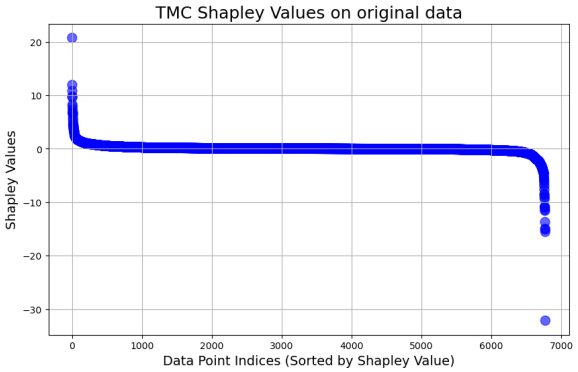


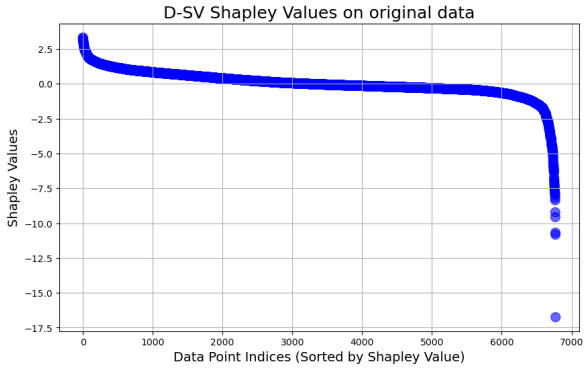


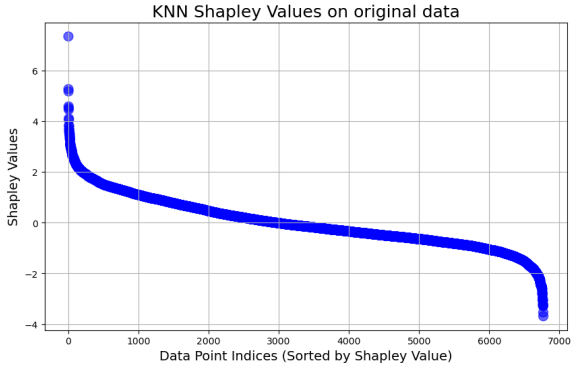


Regression SV - Original Dataset

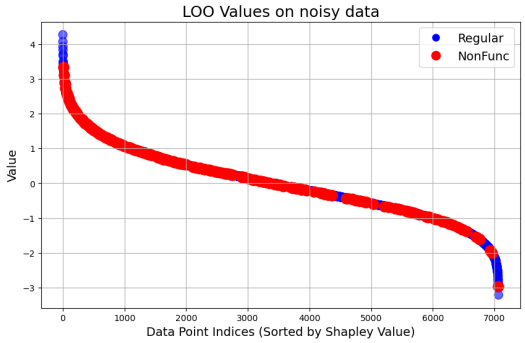


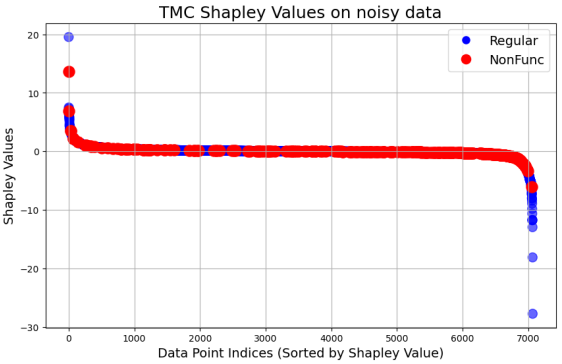


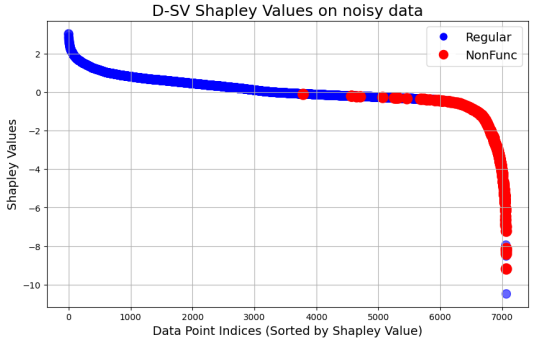


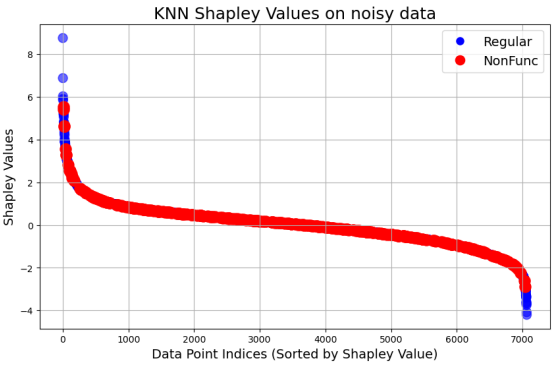


Regression SV - nonFunc dataset





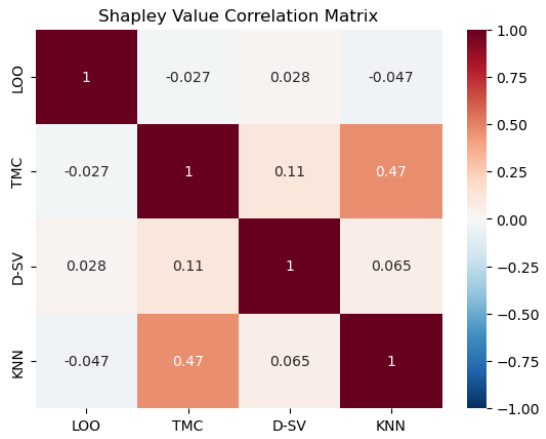




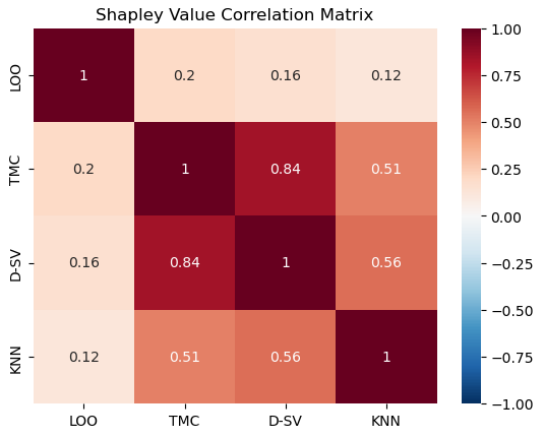
Correlation matrices:

Classification

Original dataset

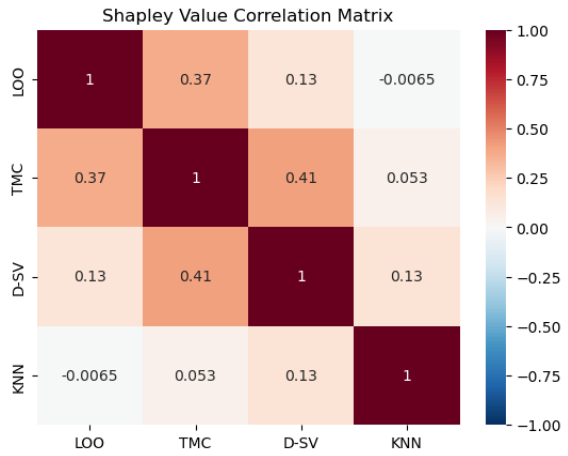


Noisy dataset



Regression

Original



nonFunc