Approximating the Power Absorption of PTO Settings of Wave Energy Converters using Surrogate Models for Optimisation

Constantina Pyromallis
Student ID: a1668648
Supervisor: Markus Wagner

Bachelor of Computer Science (Honours)
The University of Adelaide

Contents

Chapter 1

Introduction

1.1 Background and Motivation

The ocean is a natural source of wave energy, and covers 71% of the Earth's surface, yet it is a relatively untapped source of renewable energy [1, 2]. Wave energy can be harnessed and turned into electricity using a wave energy converter (WEC), of which there are many types. One such type of WEC is a point absorber buoy, which either floats on, or just below, the surface of the water, and moves with the waves to generate energy [3].

Carnegie Wave Energy developed a point absorber called CETO 5, which is a fully submerged buoy tethered to the hydraulic pump in a power take off system (PTO) on the sea floor [3, 4]. The up and down movement of waves causes the buoys to move up and down, which in turn drives the pump. The pump pressurises the water in a pipe that spans the distance from the buoys to the shore. When the water reaches the shore, it is either used to turn the turbines of an off the shelf generator to produce electricity, or to power a reverse osmosis desalination to create potable water [5]. A visual representation of this system can be seen in Figure 1.1.

An alternative to the single-tether buoys are three-tether buoys (depicted in Figure 1.2), in which there are 3 PTOs on the sea floor which each have a tether that connects to the buoy. This allows not only up and down movement (heave) to be captured and converted, but also surge and sway movement [6]. A single buoy can only produce a limited amount of electricity, so they are often deployed in large numbers as an array. In

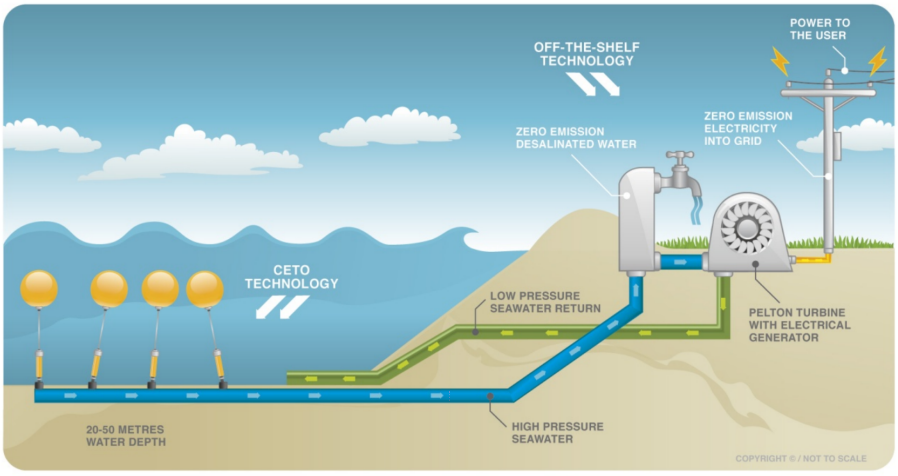


Figure 1.1: Operation of the CETO system [5].

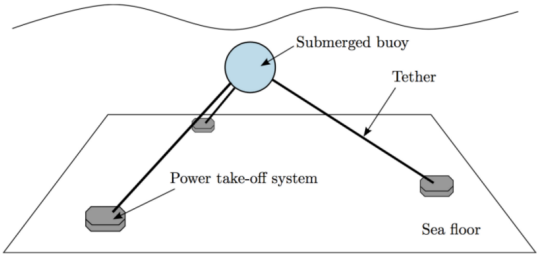


Figure 1.2: Representation of a three-tether WEC [8].

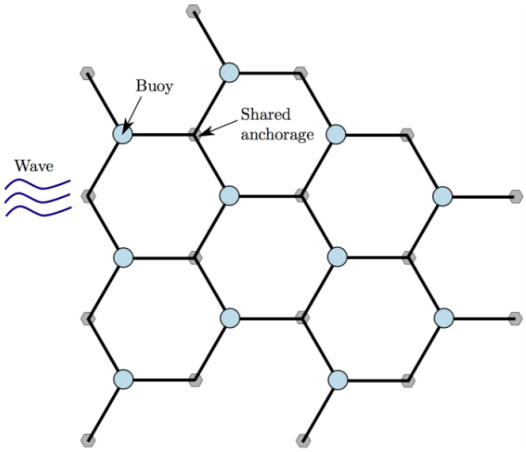


Figure 1.3: Bird's eye view of array with shared PTO [8].

the case of three-tether buoys, PTO can be shared by multiple buoys (see Figure 1.3).

1.1.1 Importance of PTO control

In addition to containing the hydraulic pump, the PTO is also used to control the movement of the buoy and absorb power from the waves. To achieve this, CETO contains a spring with spring constant k (in N/m/s), and damper (decreases amplitude) with damping factor d (in N/m). These are referred to as the PTO parameters.

The amount of power absorbed from an incoming wave is maximal if the frequency of the wave matches the natural frequency of the buoy [7]. Thus, we should force the buoy's natural frequency to match the wave's frequency. The natural frequency of the buoy is dependent on k, and thus we can control the buoy's natural frequency. The value of d decreases the amplitude of the buoy's movement. If d is too high, the buoy won't move very much with the waves, and hence doesn't absorb as much power. If d is too low, the buoy will move too much, and will lose energy. Note that d and k both alter and are dependent on the hydrodynamics of the buoy and cannot be optimised independently.

Additionally, to avoid damage, tether elongation cannot exceed 3 metres more than its natural length (between taut and slack). The tether elongation depends on both d and k. It is important to note that, for inconsistent sea states, there will be different wave frequencies, so for constant spring and damper values, the natural frequency will only match with one frequency of waves.

A series of buoys deployed near each other are referred to WEC arrays. They are often deployed in a grid or hexagonal pattern (see Figure 1.3). The primary purpose of WEC arrays is to capture energy, and thus it is something researchers aim to optimise. There are many aspects that can be individually optimised, but the key ones are the geometry of the buoy, the control via PTO, and the placement of buoys relative to one another. Our research focus is control via PTO. This is important because, as the incoming waves bounce off the first row of buoys, the wave properties change, so the waves hitting the second row of buoys are different. This means .

1.1.2 Model Runtime, and Surrogate Models

Many state-of-the-art models take large amounts of computational time to evaluate how much power a WEC array produces, given its settings. Many optimisation algorithms require many evaluations of different settings in order to find the optimal. Unfortunately, when these slow models are run multiple times, it can make running some optimisation algorithms infeasible. As such, it would be desirable to develop a model that is faster, even if it means it may be less accurate. Such approximations of the original model are called surrogate models.

1.2 Related Work

1.2.1 Optimisation of array parameters

Recall that the primary purpose of WEC arrays is to absorb power from waves. Arrays of WEC have many possible settings and parameters, including the number of buoys in the array, their locations, their PTO parameters d and k. Many papers aim to optimise these settings to find the values that absorb a maximal amount of power.

In an array of buoys, waves will be reflected off of buoys in various directions. This means that constructive and destructive interference can occur, and thus, in some locations within the array, energy absorption will be increased. This is the reason optimisation of placement of buoys is useful. Previous research into this area includes the development of a model for three-tether buoys in an array and the amount of energy it absorbs. This model is used as a fitness function for various genetic algorithms including (1+1)EA and CMA-ES which mutate the array by moving a single buoy [8].

1.2.2 PTO controllers

As mentioned in Section 1.1.1, inconsistent sea states will have waves with a variety of frequencies. If the PTO parameters are static, the natural frequency of the buoy will only match with one wave frequency. Thus, dynamic PTO controllers have been developed that adjust the parameters as time passes, and the waves change. Dynamic causal controllers do not explicitly require information about waves before they arrive. In [16], they note that the optimal causal controller can be found by solving a nonstandard linear-quadratic-Gaussian (LQG) optimal control problem.

In order to absorb maximum energy in inconsistent seas, the buoy can use explicit knowledge of incoming wave frequencies, in order to alter its own natural frequency. Controllers that use this information are acausal. Simple gradient-ascent algorithms have been used to alter PTO settings to match the incoming wave [9]. This is used as a max power point tracking (MPPT) control. The MPPT updates its PTO by a given step size at a given rate, and determines if the changes increased power absorption. It does this until the maximum power has been found.

1.2.3 Surrogate models

In regards to research into surrogate models, some approaches that have been looked into include a linear autoregressive with exogenous input (ARX) model, the Kolmogorov-Gabor polynomial (KGP), which is a natural polynomial extension of the ARX model and an artificial neural network (ANN) model [10]. These are applied to the behaviour

of a buoy over time. Some manual parameter tuning is applied to the ANN; the numbers of neurons in each layer, are increased incrementally, until a good balance is achieved between approximation of the training data and generalisation capability.

In [27], they use an Adaptive Network-based Fuzzy Inference System (ANFIS) to model non-buoyant WEC. Their training data was gathered via their own real experiments, and was therefore limited. Hence, they chose ANFIS for its predictive ability in uncertain environments. Their use case is similar to ours: to carryout further experiments to perform an optimisation upon.

Chapter 2

Approximating Power Absorption for Optimisation Purposes

2.1 Contribution

This research trains surrogate models to approximate state-of-the-art models; that is, the surrogates take the same input (array settings/parameters), and give similar output (power absorbed per buoy). The output of surrogates takes less time to evaluate than the original model when given the same array parameters.

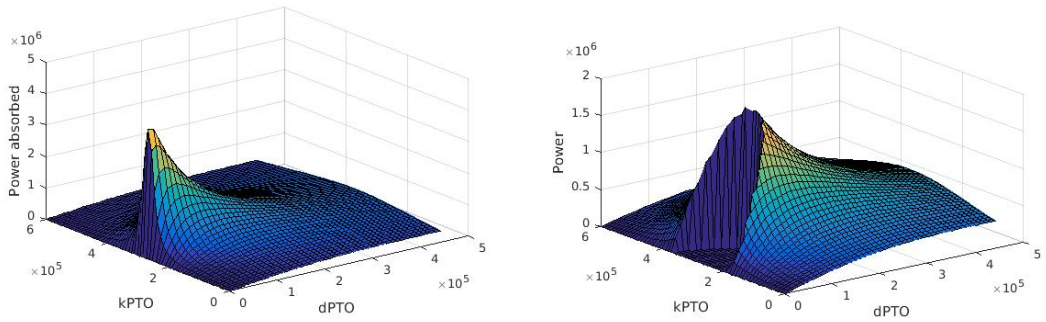
The primary purpose of this is to allow optimisation algorithms that require many model evaluations, to run in a shorter amount of time. The optimised WEC array parameters can be preprocessed by running the algorithms before array installation, or calculated in real time to make on-the-fly optimisations. As we wish to focus on optimising PTO control, we developed surrogates that can predict the power absorbed for various values of PTO parameters, d and k.

We assess the accuracy of the surrogate models, as well the effect of different surrogate parameters on the accuracy. The surrogate parameters often affect the training time, and so, we also assess the training time and the predicting time.

2.2 State-of-the-art models to approximate

We worked in collaboration with the School of Mechanical Engineering at The University of Adelaide to conduct our research. They provided an accurate, state-of-the-art mathematical model (written in Matlab) for a 3-tether CETO 5 given in [8]. For various WEC array settings and sea states, this can calculate the total power absorbed, and whether the settings are feasible (recall, settings are feasible if tether elongation does not exceed 3m). This is the model we approximate using surrogate models for optimisation purposes. When run on an Intel i7 core, the amount of time the model takes to determine the power absorption of a 2x2 array (with array settings given in Section 2.3) on average is 0.94 seconds (based on 100 samples). We therefore want the surrogate models to run faster than this.

As our focus is on the effect of PTO settings, we consider approximating the power absorption based on the PTO settings alone, and keeping all other array properties constant (see Section 2.3 for constants). One important constant is the number of buoys in the array; we consider an array with 4 buoys, as it is a small yet non-trivial model to approximate. Even with this constraint on approximation, there are still two different



(a) (b)

Figure 2.1: A visualisation of the PTO_{same} model. Figure 2.1(a) is the raw output of the mathematical model. The valley in the surface in Figure 2.1(b) is caused by infeasible parameter values. The power produced by these parameter values are set to 0. Infeasible parameters occur when the tether-elongation of the buoys exceeds 3m.

ways we can consider varying the PTO settings.

2.2.1 PTO_{same} model

The PTO_{same} model varies d and k, where d and k are the PTO settings and are the same for all buoys. Thus, this model has 2 parameters. This means, the settings and resulting power can be visualised in three-dimensional space (See Figure 2.1).

2.2.2 PTO_{indiv} model

The PTO_{indiv} model has 8 parameters, and will therefore be more difficult to learn. The parameters are d_1, d_2, d_3, d_4, k_1, k_2, k_3, k_4, where d_i and k_i correspond to the i^{th} buoy's PTO settings.

2.2.3 Discussion of sub-models: original, altered, ignorant

One thing to note is that, for optimization purposes, if we want to consider only feasible setting, it is not useful to approximate the original model (depicted in Figure 2.1(a)). For this reason, we also try to approximate the altered model, in which the infeasible powers are set to zero (depicted in Figure 2.1(b)). However, the steep cliff of the altered model can be hard for mathematically based approaches to learn.

In order to make approximations of the original model useful, we can also develop a classifier to determine whether PTO settings are infeasible or feasible. This would be utilised in optimisation algorithms to classify and ignore infeasible settings. If we ignore infeasible settings, we can also consider training surrogates based only on feasible data (ignorant model), since we don't care about the approximated power of thoe settings. In other words, we only train the surrogate on the feasible samples. See Figure 2.2 for visualisations of all 3 sub-models.

When classifying feasibility, we simply use the original sub-model. We could also classify the altered sub-model, since power is the only difference between the sub-models, and feasibility is the same for both. However. the ignorant sub-model would not be useful, as it removes infeasible samples. There is a very clear distinction between the PTO values that result in feasible power, and those that don't, at least for the PTO_{same} model. The

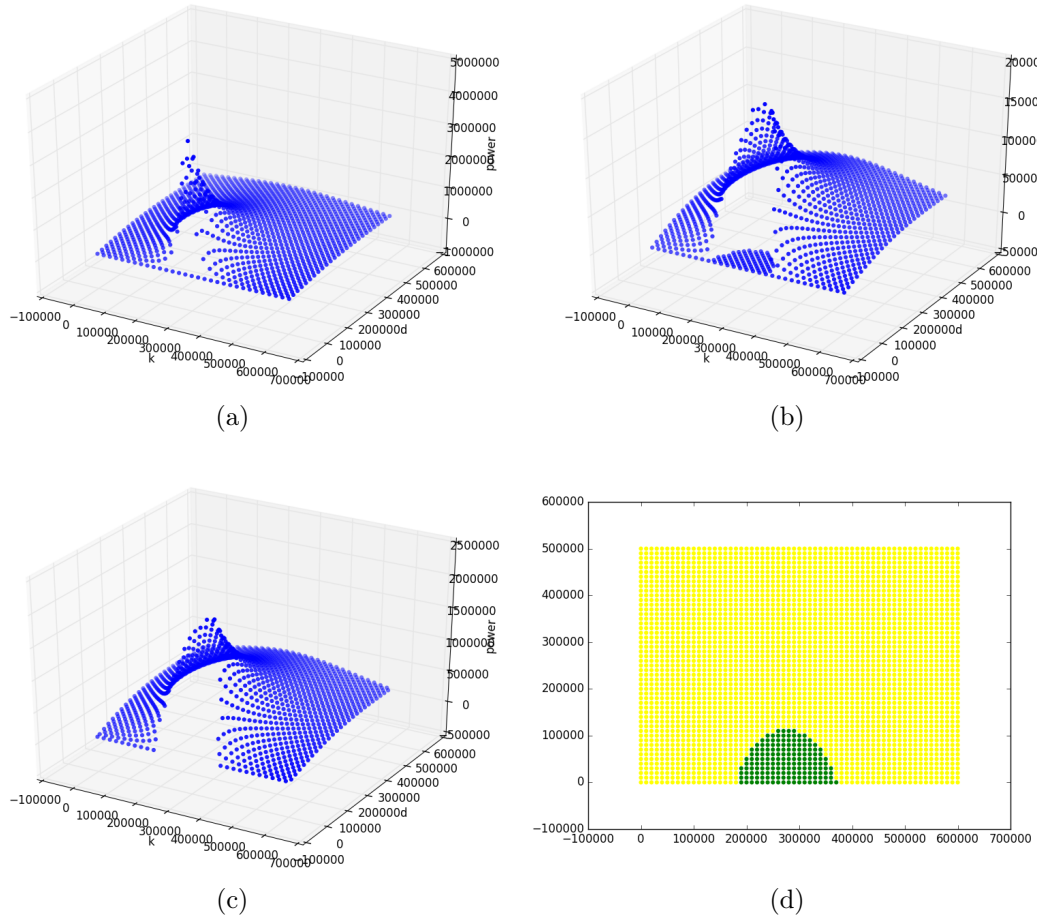


Figure 2.2: The data for the PTO_{same} sub-models. Figure 2.2(a) is the original sub-model, Figure 2.2(b) is the altered sub-model, and Figure 2.2(c) is the ignorant sub-model. These are what we aim to approximate. Figure 2.2(d) is the feasibility of the PTO_{same} sub-models. Yellow points are feasible, and green are infeasible. This is what we aim to classify.

classifier only has to develop a curve to separate these, as the regions of feasible data do not enclose any parts of the infeasible regions, and vice versa. See Figure 2.2(d) for a visualisation.

2.3 Collecting data

To train the surrogates, we need data. The data is a collection of samples that includes various input and associated output from the state-of-the-art model. We only want to consider inputs with varied d and k. All other array parameters were kept constant:

- Number of buoys: 4.
- Relative locations of buoys: (0, 0), (0, 50), (50, 0), (50, 50), in metres, where (0, 0) is the bottom-left corner.
- Wave frequencies: 0.7 rad/s.
- Wave direction: from left, with an incident angle of 0 radians.

- Buoy diameter: 5m.
- Water depth: 30m.
- Submergence depth (from the water level to the top of the buoy): 3m.

Each sample in a dataset has an input vector x and output vector y. The final form is "x_1, ... x_n, y_1, ... y_n". The input vector for PTO_{same} is [d, k], and for PTO_{indiv} it is [d_1, d_2, d_3, d_4, k_1, k_2, k_3, k_4]. For both, the output vector is [$power$, $feasibility$].

2.3.1 Collection methods

For each model PTO_{same} and PTO_{indiv}, we collected data using two different methods: grid and random. Note that the collection methods are primarily ways of varying the input variables. Each method then calculates the corresponding output determined by the mathematical model, and stores it as a sample.

We obtained 4 data sets in total: PTO_{same} samples generated from the grid method and the random method (3112 data points each), and PTO_{indiv} samples generated from the grid method and the random method (160000 data points each). The data was collected on a Xeon E5-2698v3 CPU. The PTO_{same} data sets took 27 minutes each to collect, while the PTO_{indiv} data sets took 22.5 hours. This leads to an average time of 0.5 seconds to calculate the power absorption for a given d and k.

Grid

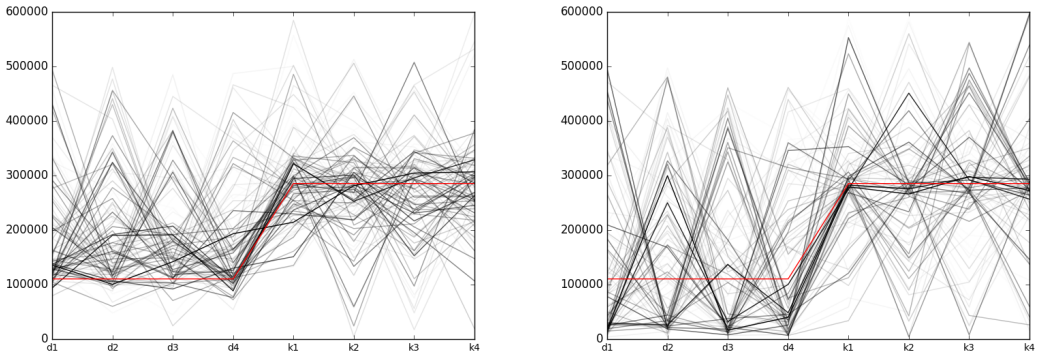
The first collection method used a grid search, so the input values were at uniformly spaced intervals. We considered a valid set of d and a valid set of k, and calculated the resulting power for every d and for every k. The valid set of d, or k, can be defined by a lower and upper bound for each, and a step size for the difference between each value in the set. The grid to search can thus be defined by the lower d and k, upper d and k, and the step size. We refer to these as lower_d, lower_k, upper_d, upper_k, and step_size, respectively. If D and K are the total number of valid d and k in each set, then the grid search takes $\Theta(DK)$ time to execute.

Data for the PTO_{same} model was generated using a grid that can be defined as follows:

- lower_d = 0
- lower_k = 0
- upper_d = 500000
- upper_k = 600000
- step_size = 10000

Data for the PTO_{indiv} model was generated using a grid that can be defined as follows:

- lower_d = 0
- lower_k = 0
- upper_d = 450000
- upper_k = 600000
- step_size = 150000



(a) (b)

Figure 2.3: A visualisation of a subset of the samples collected for the PTO_{indiv} model. Each sample is plotted as a line connecting input variable values that correspond to that same sample. Figure 2.3(a) contains the top 100 feasible samples by power absorption. Figure 2.3(b) contain the top 100 both feasible and infeasible. Each samples has an opacity which is equal to $(power - power_{min})/(power_{max} - power_{min})$, where power is the $power$ absorbed by that sample, $power_{min}$ and $power_{max}$ are the minimum power and maximum power absorbed across all plotted samples, respectively. The red line is the optimal settings for the PTO_{same} model.

Random

The second method we used was to, for each input variable, select a valid value, uniformly at random. Use this to generate input vectors until the desired number of samples is obtained.

The valid values are any integers in the range given by the same lower and upper d and k given above. This is repeated until the desired number of samples is obtained.

2.3.2 Visualisation

The grid data for PTO_{same} is visualised in Figure 2.1. A subset of the random data for PTO_{indiv} is visualised as a parallel plot in Figure 2.3. This depicts the first 100 samples when sorted in descending order by power absorption, as well as the first 100 samples that are all feasible.

2.4 Training Surrogate Models

Surrogate models are the product of some training approach; we consider supervised training approaches. All of the approaches can be used for both approximating the power absorption of PTO settings (regression) and classifying the settings as either feasible and infeasible. Note that the resulting regressors and classifiers will be distinct surrogates.

There are 4 distinct problems that we wish to solve using surrogate models: approximating PTO_{same}, classifying PTO_{same}, approximating PTO_{indiv}, and classifying PTO_{indiv}.

When approximating, either PTO_{same} or PTO_{indiv} models, we want to consider training on both the grid and random data sets. We also want to consider approximating the sub-models, altered, ignorant, and original (which involve altering the datasets as

mentioned in Section 2.2.3). When classifying the models, we want to consider training on both the grid and random data sets again. We don't need to consider any sub-model besides the original. This is because the altered sub-model only differs from the original in its values for power, which doesn't matter when classifying for feasibility. In addition. the ignorant sub-model doesn't contain infeasible samples, which is not useful for classifying feasibility.

Each data set is divided into two equally sized sets: one for training, and one for validating. To divide the grid data set, it is first sorted by its d value, and then its k value, and every second datum is put into the training set, and all others into the validation set. The random data sets are divided in half so the first half is the training set, and the second is the validation set. Both subsets are still collected uniformly at random.

All the surrogates are trained using the training set only, and the performance is assessed by using the surrogates to predict values for the validation set. The accuracy metric we use is R^2 (see Equation 3) and the precision metric is MSE (see Equation 5). All surrogate implementations are from the sklearn python library [31]. All experiments were run on a intel i7 core. Below, we describe the surrogates and their settings (hyper parameters), and the results are given in the 2 following chapters. Any hyper parameters that are not defined below are the defaults as defined by sklearn documentation.

2.4.1 K nearest neighbours

The k nearest neighbours method (KNN) for approximation involves storing all the training input vectors in a data structure, with their associated outputs. When approximating (the power) of a new input vector (PTO settings), the algorithm looks up the k nearest training vectors, and averages their associated outputs (power absorption). In this case, each neighbour contributes uniformly to the final prediction. However, a good heuristic is to use the weighted average of the nearest neighbour outputs, where the weight for each neighbour is the distance to the new input vector to approximate. This way, the closer neighbours influence the result more. When classifying data, the training input vectors are stored with their associated classification, rather than real valued output, and similar logic is applied.

When assessing the performance of this surrogate, we consider the weight heuristics, "uniform" and "distance", and try all k in the set {1, 2, 3, 4, 5, 7, 10, 15}.

2.4.2 Random forest

When given some input vector, a random forest regressor (RFR) utilises multiple random decision trees to guess the portion of the space, that the corresponding output should be in. Each node in a tree corresponds to an input value (in this case, the PTO parameters, d and k, or d_1, ... d_4, k_1, ... k_4), and has some threshold. If a value exceeds the threshold, we look at the node to the right (otherwise, to the left), and repeat the process. Leaf nodes contain the predicted output for the input. Random forests train a number of decision trees on random sample subsets. The output of the randomly trained decision trees is then averaged to produce the final output of the random forest regressor. For random forest classifiers, everything is the same, except leaf nodes store classes instead of output.

The hyper parameter of this surrogate that we are interested in is the number of decision trees used in the random forest. We consider the following number of trees: {1,

5, 10, 20, 50, 100}.

2.4.3 Multilayer perceptrons

Multilayer perceptrons (MLP) uses a back-propagation algorithm to update the weights between nodes in the network. The hyper parameters we consider are the number of layers, and the number of nodes per layer. We consider all combinations of number of layers {5, 10, 20}, and node per layer {5, 10, 20, 50, 100} for approximation. For regression of the PTO_{same} model, we add 30 to the set of numbers of layers.

2.4.4 Support vector machines

When classifying linearly separable data, we wish to find the best hyperplane $w^(T)x + b$ that separates the data. One way of defining the best hyperplane, is the separating hyperplane for which the distance to the nearest point for each class set is maximised. This is referred to as the maximum-margin hyperplane. Note that the maximum-margin hyperplane is completely dependent on the points closest to it (other points further away can be ignored). These points are called support vectors, and give SVMs their name. To use SVM for approximation, we instead try to fit a hyperplane to the data so we have a minimum margin hyperplane that all the training data lies within.

We can classify non-linearly separable data using different kernels. In our experiments, we consider radial basis function (RBF) kernels. We can also introduce slack variables to allow outliers in our training data. This introduces another hyperparameter, C. We consider values for C from the set {1e4,1e5,1e6,1e7}. Note that we do not use SVM for the PTO_{indiv} model due to excessive runtimes.

Chapter 3

Approximating Power Absorption of Arrays of WEC with the same PTO Settings

In this chapter, we look at the performance of each surrogate for both regression and classification on the PTO_{same} model. We look at 4 different performance metrics: the training time, predicting time, mean-squared error (MSE) and variance score. Training time is the total time (in ms) taken to train the surrogate on all training data, and predicting time is the total time (in ms) taken to predict all validation data. Mean-squared error is a measurement of precision (low MSE means high precision), given in Equation 5, and variance score is the R^2 value, a measurement of accuracy, given in Equation 3. All surrogates are trained and assessed twice; once on the grid data, and once on the random data. One thing to note is that the grid data predictions may be biased, since the validation set is also in a grid formation, and hence equidistant from all the training points. Thus, it may not be a good measure of how well the surrogates trained on grid data will perform when optimisations are run on them.

3.1 Regression

As mentioned previously, we wish to approximate 3 different sub-models: original, altered, and ignorant. This section assesses the approximation or regression results by each approach individually, and then compares the approaches.

3.1.1 By approach

Each subsection of this section looks at the performance of a surrogate when approximating all sub-models of PTO_{same} when trained on either the grid or random data sets.

KNN

The raw KNN results are in Tables 3.1 to 3.4. These are visualised in Figure 3.1. For all sub-model approximations trained on either grid or random datasets, the distance metric performs as well or better than the uniform metric in terms of both accuracy and precision. This is expected, since if the point to predict is closer to one training point than all the others, that training point will have more of an effect on the value when using the

k	1	2	3	4	5	7	10	15
original grid	0.92	0.96	0.98	0.98	0.97	0.96	0.96	0.95
alter grid	0.9	0.94	0.96	0.95	0.95	0.95	0.95	0.94
ignore grid	0.99	1	1	1	1	1	1	1
original random	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
alter random	0.92	0.95	0.96	0.96	0.95	0.95	0.95	0.95
ignore random	0.99	0.99	0.99	1	1	1	0.99	0.99

k	1	2	3	4	5	7	10	15
original grid	0.92	0.96	0.98	0.97	0.96	0.94	0.95	0.93
alter grid	0.9	0.94	0.96	0.95	0.94	0.94	0.94	0.92
ignore grid	0.99	1	1	1	1	0.99	1	0.99
original random	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.98
alter random	0.92	0.95	0.96	0.95	0.94	0.95	0.94	0.93
ignore random	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99

Table 3.1: R^2 of KNN regressors. The upper table uses the distance metric, while the lower one uses the uniform metric.

k	1	2	3	4	5	7	10	15
original grid	1.59099e10	9.24279e9	4.45886e9	4.51485e9	5.51423e9	8.26821e9	8.0757e9	1.07123e10
alter grid	1.14427e10	6.5418e9	4.89679e9	5.31547e9	5.57529e9	5.55176e9	6.01162e9	7.11602e9
ignore grid	1.09169e9	3.68104e8	1.58918e8	7.65866e7	1.60519e8	2.52988e8	2.18355e8	2.99627e8
original random	2.89739e9	2.13034e9	1.56615e9	1.58338e9	1.55676e9	1.32347e9	1.41633e9	1.78234e9
alter random	9.37492e9	5.56231e9	4.37329e9	4.9306e9	5.41526e9	5.32933e9	5.73802e9	6.1325e9
ignore random	8.88614e8	7.4022e8	5.9115e8	4.26556e8	4.22102e8	4.58563e8	5.82524e8	7.31973e8

k	1	2	3	4	5	7	10	15
original grid	1.59099e10	9.24279e9	4.45886e9	5.97394e9	8.34906e9	1.28564e10	1.08311e10	1.54856e10
alter grid	1.14427e10	6.5418e9	4.89679e9	5.32543e9	6.36685e9	6.51425e9	6.93442e9	8.81628e9
ignore grid	1.09169e9	3.68104e8	1.78398e8	1.23541e8	3.6937e8	5.73274e8	3.79189e8	5.52243e8
original random	2.89739e9	2.4319e9	1.90268e9	2.19934e9	2.17502e9	2.10159e9	2.58133e9	3.8784e9
alter random	9.37492e9	5.57133e9	4.20432e9	5.08706e9	6.2655e9	6.16941e9	7.12517e9	8.15028e9
ignore random	8.88614e8	8.03131e8	6.82602e8	5.64775e8	6.06999e8	7.25942e8	9.59868e8	1.28814e9

Table 3.2: MSE of KNN in ms. The upper table uses the distance metric, while the lower one uses the uniform metric.

k	1	2	3	4	5	7	10	15
original grid	0.001621	0.001307	0.001246	0.001326	0.001254	0.001262	0.001253	0.00136
alter grid	0.001485	0.001169	0.001149	0.001169	0.001361	0.001192	0.001136	0.001166
ignore grid	0.001764	0.001854	0.001747	0.001489	0.001495	0.001524	0.001374	0.001469
original random	0.001316	0.001372	0.001514	0.001426	0.001448	0.001333	0.001442	0.001344
alter random	0.001446	0.001375	0.001444	0.001265	0.001268	0.001374	0.001465	0.001505
ignore random	0.001354	0.001516	0.001332	0.001434	0.001391	0.001463	0.001463	0.001548

k	1	2	3	4	5	7	10	15
original grid	0.001607	0.001313	0.001258	0.001219	0.001193	0.001244	0.001235	0.001205
alter grid	0.001959	0.001369	0.001269	0.001226	0.001269	0.001225	0.001276	0.001244
ignore grid	0.00162	0.001488	0.001484	0.001482	0.001588	0.001499	0.001523	0.001553
original random	0.001376	0.001288	0.001433	0.00122	0.001425	0.001583	0.001463	0.001414
alter random	0.001486	0.002257	0.001195	0.001239	0.001216	0.001349	0.001182	0.00126
ignore random	0.001546	0.001415	0.001381	0.001347	0.002016	0.001716	0.00145	0.001428

Table 3.3: Training time of KNN regressors. The upper table uses the distance metric, while the lower one uses the uniform metric.

distance metric. In addition, the KNN surrogates trained on the random dataset perform better in terms of accuracy and precision.

Training time is constant in k, which is expected, since it just involves putting all the training samples into a data structure and is not dependent on the value of k. Testing time however, is dependent on k, and we can see it increases approximately linearly.

Across all sub-models, as the value of k increases up 4, the accuracy and precision increase. When k is set to 15, performance is less than or equal to the performance for the previous value of k (10), but not by much. This is expected, as training samples much further away from the point to approximate will contribute to the approximation. The peak in performance occurs at k=4, particularly for the surrogates trained on grid data. This is also expected, as the point will always lie between 4 training points, and more or

k	1	2	3	4	5	7	10	15
original grid	0.003515	0.003447	0.003717	0.003754	0.004348	0.004379	0.005117	0.006134
alter grid	0.003288	0.003259	0.003337	0.004016	0.00427	0.004168	0.004696	0.005868
ignore grid	0.003451	0.004333	0.003949	0.005284	0.004207	0.004553	0.005143	0.006578
original random	0.003336	0.003685	0.004048	0.004166	0.004489	0.004845	0.005519	0.006661
alter random	0.003423	0.003615	0.003962	0.004206	0.004347	0.004905	0.005647	0.006792
ignore random	0.00338	0.003633	0.004143	0.004128	0.00434	0.004864	0.005315	0.006915

k	1	2	3	4	5	7	10	15
original grid	0.003331	0.00333	0.003551	0.003449	0.004213	0.004793	0.004585	0.005737
alter grid	0.003944	0.003328	0.003235	0.003464	0.004009	0.004253	0.004801	0.005731
ignore grid	0.003401	0.003468	0.003488	0.003707	0.004128	0.004327	0.004743	0.006326
original random	0.003243	0.003861	0.003735	0.004258	0.004314	0.004648	0.005266	0.006455
alter random	0.003352	0.005378	0.003638	0.003784	0.004103	0.004501	0.005293	0.006112
ignore random	0.003366	0.003624	0.004054	0.003844	0.005741	0.004484	0.005519	0.005919

Table 3.4: Predicting times of KNN in ms. The upper table uses the distance metric, while the lower one uses the uniform metric.

less neighbours than 4 will cause the weighted average to be skewed in one direction.

The sub-model that the KNN regressor had the most success with is the ignorant sub-model, followed by the original, then the altered sub-model. See also Figure 3.2 for visualisations of the approximations.

Random Forest

num trees	1	5	10	20	50	100
original grid	0.96	0.99	0.99	0.99	0.99	0.99
alter grid	0.89	0.94	0.96	0.96	0.96	0.96
ignore grid	0.99	1	1	1	1	1
original random	0.96	0.99	0.99	0.99	0.99	0.99
alter random	0.84	0.89	0.91	0.92	0.92	0.92
ignore random	0.98	0.99	0.99	1	1	1

Table 3.5: R^2 of RFR.

num trees	1	5	10	20	50	100
original grid	7.64384e+09	2.89273e+09	2.31284e+09	1.40298e+09	1.11984e+09	1.05823e+09
alter grid	1.253e+10	6.28974e+09	4.72884e+09	4.98014e+09	4.49739e+09	4.54843e+09
ignore grid	8.33569e+08	2.44155e+08	1.55618e+08	1.27832e+08	1.03548e+08	8.67597e+07
original random	8.90851e+09	2.13923e+09	1.96747e+09	1.72835e+09	1.55564e+09	1.53271e+09
alter random	1.82213e+10	1.21795e+10	1.02594e+10	9.48874e+09	9.30552e+09	9.26547e+09
ignore random	1.61599e+09	7.36494e+08	5.18699e+08	4.44397e+08	3.95679e+08	3.51524e+08

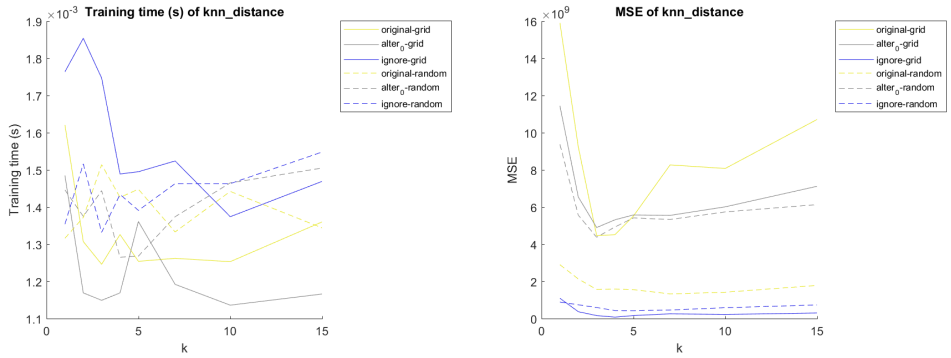
Table 3.6: MSE of RFR.

num trees	1	5	10	20	50	100
original grid	0.005672	0.055311	0.121438	0.183823	0.304527	0.469036
alter grid	0.006052	0.084349	0.075368	0.188461	0.278847	0.487295
ignore grid	0.012219	0.058148	0.043622	0.170536	0.353385	0.658575
original random	0.009694	0.031031	0.054756	0.1016	0.236981	0.485082
alter random	0.009146	0.030907	0.059732	0.101097	0.240046	0.482022
ignore random	0.013523	0.033059	0.055161	0.104946	0.3763	0.624334

Table 3.7: Training times of RFR in ms.

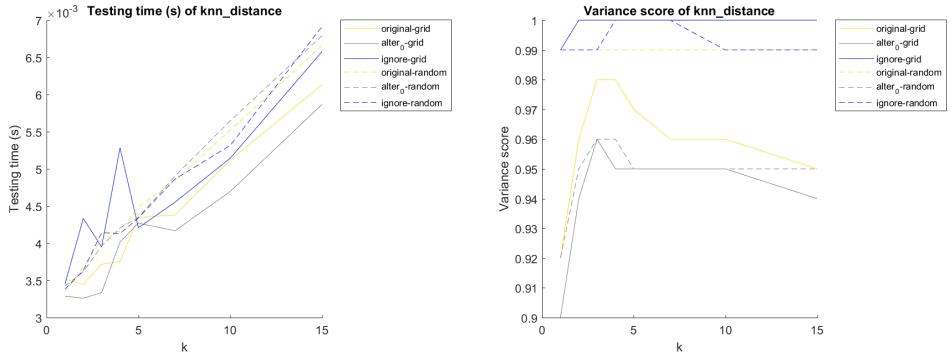
The raw random forest regressor (RFR) results are in Tables 3.5 to 3.8. These are visualised in Figure 3.3.

The RFR surrogates trained on the grid dataset perform slightly better in terms of accuracy and precision. Both training time and testing time are linear in the number of trees. This is expected, since there are more trees to train, and more trees to make an approximation and average.



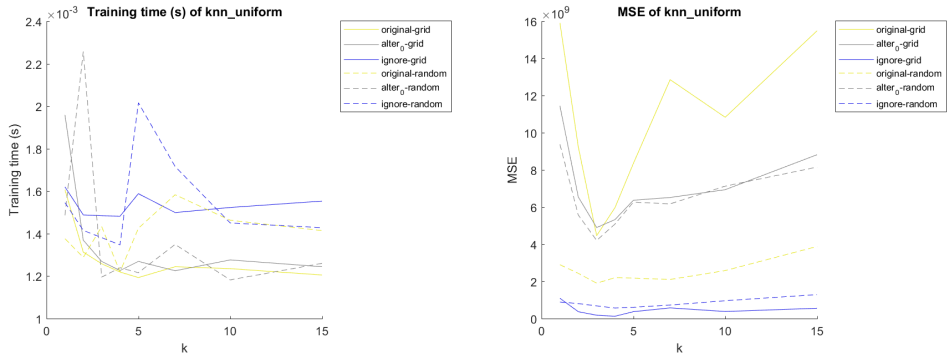
(a)

(b)



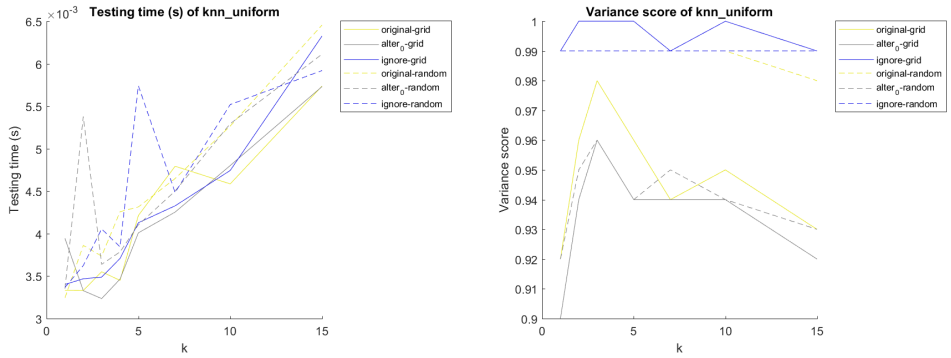
(c)

(d)



(e)

(f)



(g)

(h)

Figure 3.1: Performance of KNN regression surrogates on the PTO_{same} model.

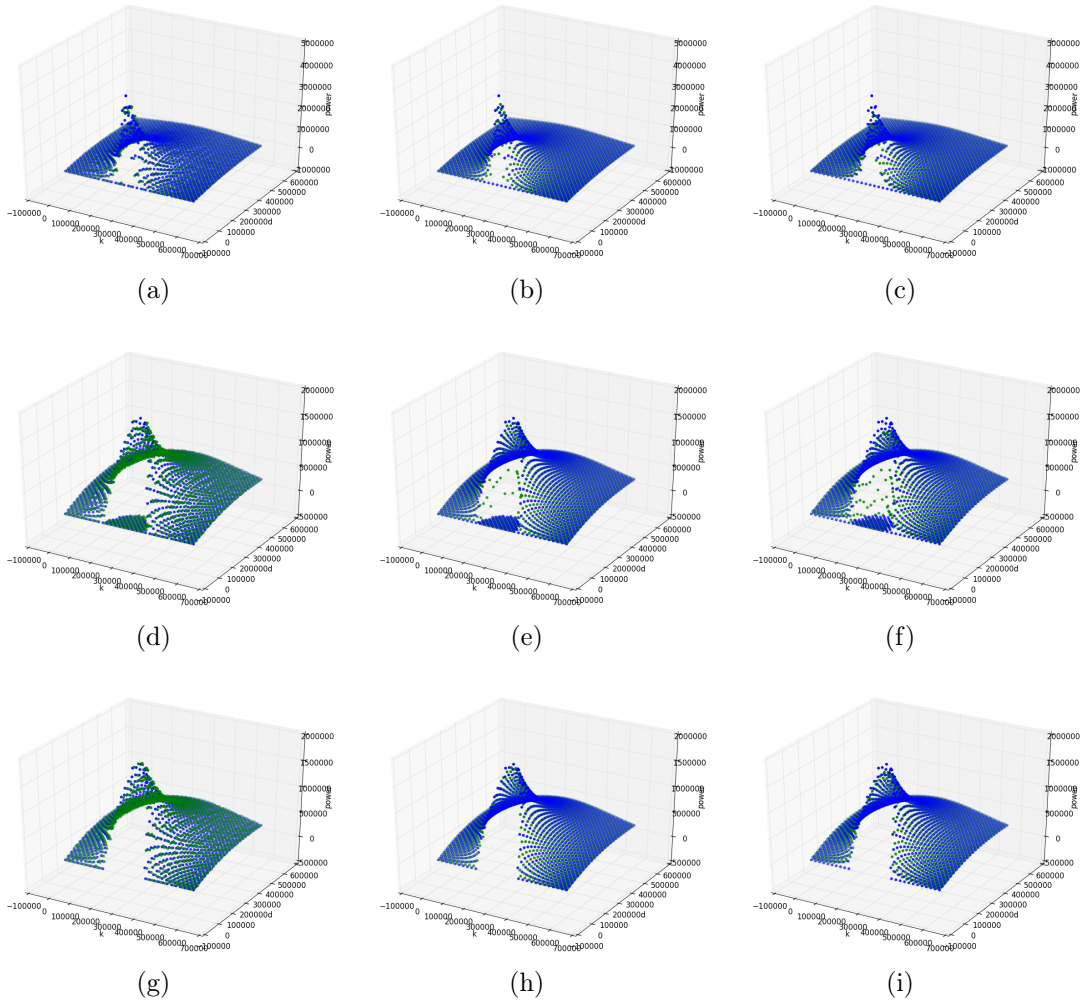


Figure 3.2: Plots of the KNN approximations of the validation data (green), and the training data (blue). The grid datasets were used for this visualisation. Each row corresponds to a different sub-model; the first is original, followed by altered, then ignorant. Each column corresponds to different values of k; in the first, it is set to 1, followed by 4, and finally 15.

num trees	1	5	10	20	50	100
original grid	0.001592	0.012048	0.01197	0.017782	0.03361	0.069045
alter grid	0.00453	0.016638	0.009443	0.01653	0.033832	0.066348
ignore grid	0.004384	0.004616	0.008472	0.045094	0.051173	0.17907
original random	0.004089	0.007867	0.008191	0.018693	0.050657	0.092739
alter random	0.002277	0.005202	0.009017	0.019371	0.049009	0.077156
ignore random	0.002097	0.005218	0.007825	0.020703	0.052509	0.092508

Table 3.8: Testing times of RFR in ms.

The sub-model that this regressor had the most success with is the ignorant sub-model, followed by the original, and finally, the altered sub-model.

Across all sub-models, as the number of trees increases, the accuracy and precision increase. This is expected based on the theory behind the approach. However, the improvement rate quickly decreases, and there is no difference in performance for 20 or more trees. See also Figure 3.4 for visualisations of the approximations.

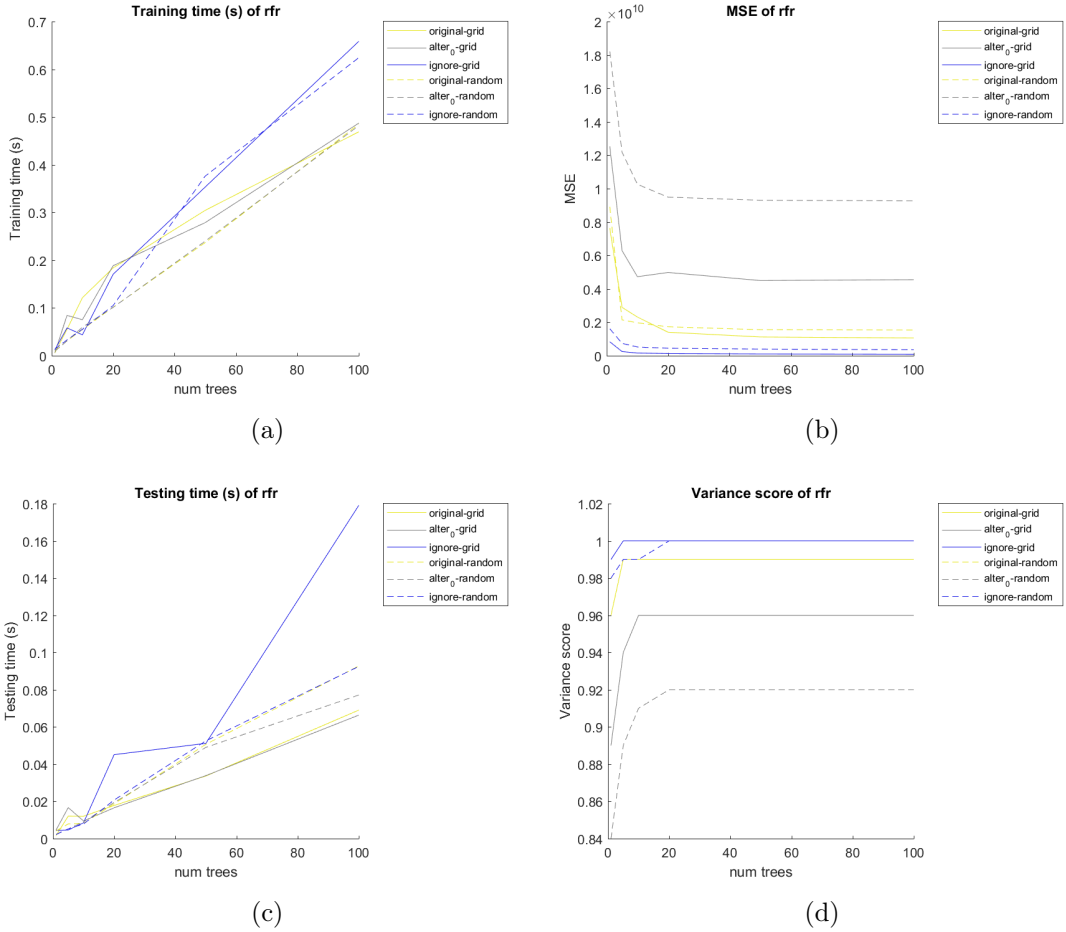


Figure 3.3: Performance of RFR surrogates on the PTO_{same} model.

num nodes per layers	5	10	20	50	100
original grid 5	-0.15	0.03	0.01	0.02	0.86
original grid 10	0.71	0.79	0.92	0.89	0.95
original grid 20	-0	0.91	0.91	0.86	0.94
alter grid 5	-0.13	0.04	0.05	0.18	0.88
alter grid 10	0.3	0.76	0.86	0.84	0.92
alter grid 20	0.8	0.83	0.88	0.91	0.79
ignore grid 5	-0.2	0	0.02	0.02	0.99
ignore grid 10	0.5	0.79	1	1	1
ignore grid 20	-0	0.99	1	1	1
original random 5	-0.22	-0	0.01	0.53	0.94
original random 10	0.76	0.77	0.98	0.99	0.99
original random 20	0.9	0.99	0.99	0.99	0.98
alter random 5	-0.15	0.03	0.1	0.19	0.89
alter random 10	0.32	0.72	0.93	0.94	0.96
alter random 20	0.81	0.82	0.94	0.95	0.97
ignore random 5	-0.21	-0.01	0.01	0.01	0.99
ignore random 10	0.64	0.68	1	1	1
ignore random 20	-0	0.99	1	1	1

Table 3.9: R^2 of MLP with various numbers of layers (specified in first column).

MLP

The raw MLP results are in Tables 3.9 to 3.12. These are visualised in Figure 3.5.

The MLP surrogates trained on the grid dataset perform slightly better in terms of accuracy and precision. Both training time and testing time are linear in the number of nodes per layer, as shown by the graphs. However, it is important to know the actual complexity is also linear in the number of layers as well.

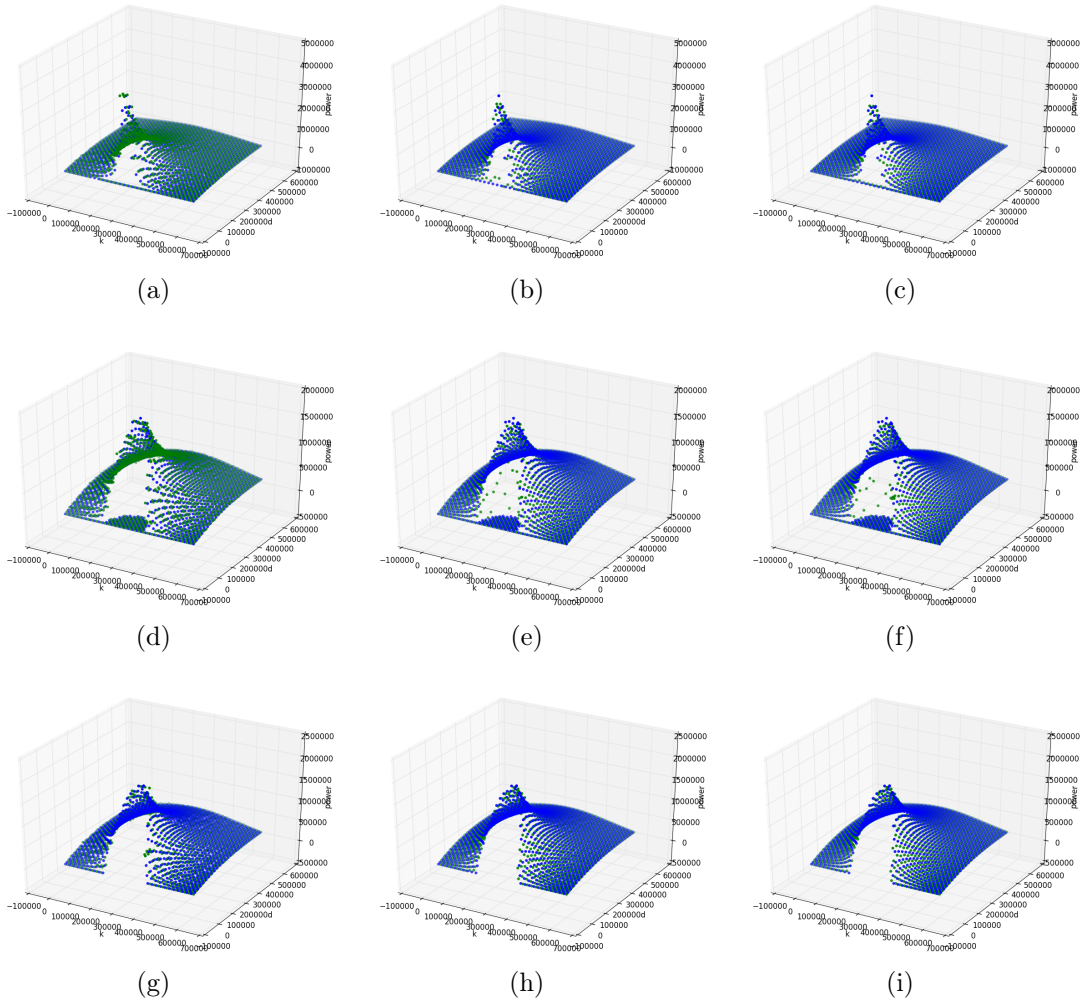


Figure 3.4: Plots of the random forest approximations of the validation data (green), and the training data (blue). The grid datasets were used for this visualisation. Each row corresponds to a different sub-model; the first is original, followed by altered, then ignorant. Each column corresponds to different number of trees; in the first, there is 1, followed by 10, then 100. We can see the smoothness of the plots increase from left to right, and the green points (approximations) appearing more equidistant from the blue points (training data).

This surrogate best approximated the ignorant sub-model, followed by the original, and finally, the altered sub-model.

Across all sub-models, as the number of nodes in each layer increases, so too does the accuracy and precision. However, some configurations of number of layers and number of nodes per layer do not result in significant increases in accuracy and precision when the number of nodes per layer is increased. See also Figure 3.6 for visualisations of the approximations.

SVM

The raw SVM results are in Tables 3.13 to 3.16. These are visualised in Figure 3.7.

The SVM surrogates trained on the grid dataset perform slightly better in terms of accuracy and precision. For the training times, the growth increases with C, when approximating the original model. For the ignorant model, the training time growth

num nodes per layers	5	10	20	50	100
original grid 5	2.39294e+11	2.01667e+11	2.05212e+11	2.04124e+11	2.873e+10
original grid 10	5.97037e+10	4.41498e+10	1.74466e+10	2.3624e+10	1.07633e+10
original grid 20	2.08273e+11	1.80781e+10	1.87061e+10	2.90721e+10	1.19604e+10
alter grid 5	1.25588e+11	1.06804e+11	1.05681e+11	9.09975e+10	1.30486e+10
alter grid 10	7.75311e+10	2.6365e+10	1.56186e+10	1.79391e+10	8.37981e+09
alter grid 20	2.2548e+10	1.9137e+10	1.35192e+10	1.02002e+10	2.31681e+10
ignore grid 5	1.2019e+11	1.00282e+11	9.85828e+10	9.85381e+10	7.34724e+08
ignore grid 10	5.05102e+10	2.14399e+10	4.40544e+08	8.92232e+07	1.54155e+08
ignore grid 20	1.00565e+11	1.36305e+09	2.64615e+08	1.20397e+08	1.34082e+08
original random 5	2.62565e+11	2.15797e+11	2.13906e+11	1.00291e+11	1.31174e+10
original random 10	5.19357e+10	4.88362e+10	4.13454e+09	1.88665e+09	1.26476e+09
original random 20	2.17746e+10	2.58905e+09	2.38425e+09	2.13216e+09	3.64767e+09
alter random 5	1.29721e+11	1.09927e+11	1.01252e+11	9.15376e+10	1.2603e+10
alter random 10	7.64544e+10	3.1176e+10	7.3405e+09	6.40211e+09	4.32406e+09
alter random 20	2.18184e+10	2.02714e+10	6.48499e+09	6.13344e+09	3.4753e+09
ignore random 5	1.2393e+11	1.03672e+11	1.01378e+11	1.01569e+11	1.0645e+09
ignore random 10	3.6624e+10	3.25515e+10	4.94003e+08	3.79261e+08	1.55871e+08
ignore random 20	1.02911e+11	9.83825e+08	5.03994e+08	2.19536e+08	1.69421e+08

Table 3.10: MSE of MLP with various numbers of layers (specified in first column).

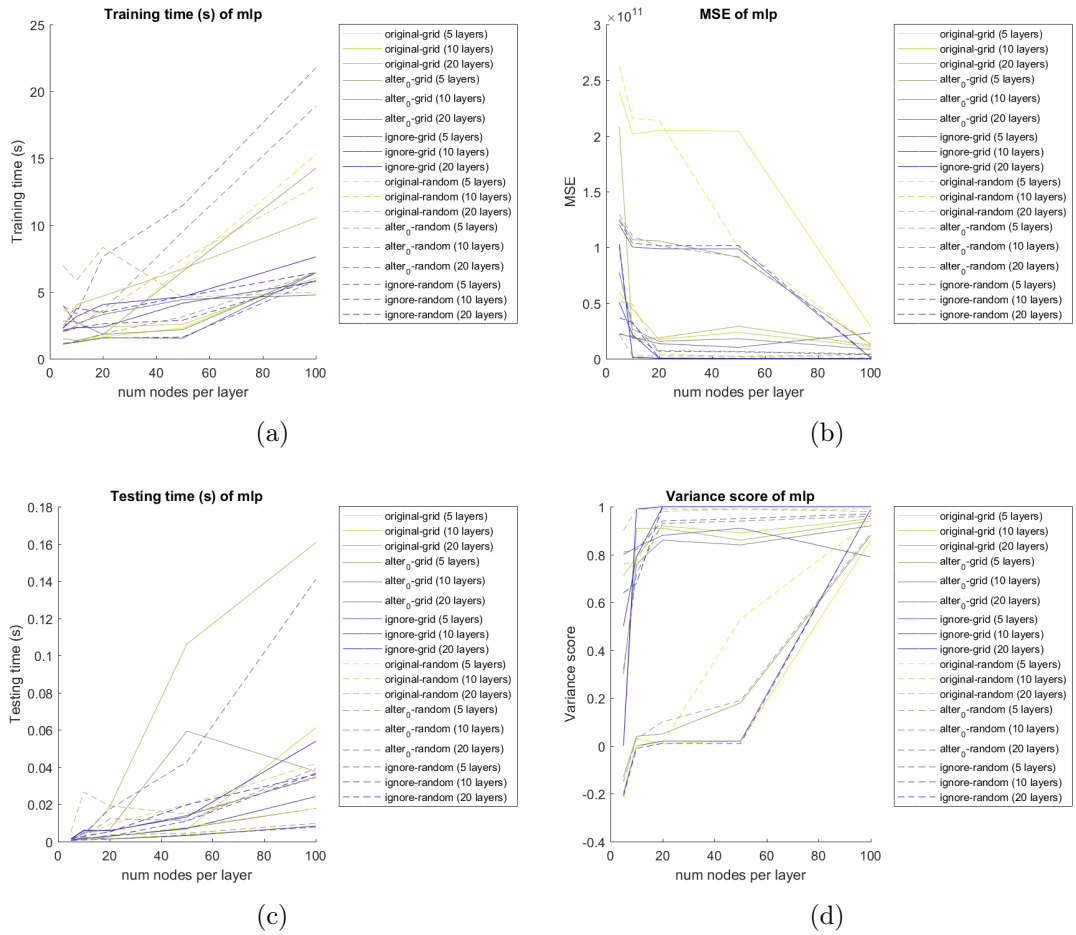
num nodes per layers	5	10	20	50	100
original grid 5	1.13132	1.28788	1.62567	2.29416	6.43111
original grid 10	2.00703	2.30235	2.37074	2.58748	6.02778
original grid 20	2.43812	4.03585	4.68558	6.7241	10.5244
alter grid 5	1.50853	1.35542	1.8109	6.53749	14.2515
alter grid 10	2.8082	2.73656	1.81653	2.17356	6.43677
alter grid 20	3.94352	2.62447	3.3053	4.42696	4.76675
ignore grid 5	1.08676	1.2338	1.5621	1.53438	6.41382
ignore grid 10	2.13843	2.33625	2.36547	4.15596	5.7734
ignore grid 20	2.31232	3.15169	4.05574	4.6303	7.61771
original random 5	1.14633	1.30538	1.75803	6.78704	15.2887
original random 10	3.61521	3.32493	3.46077	7.45793	12.8903
original random 20	6.95077	5.86094	8.32976	4.57936	4.98663
alter random 5	1.13404	1.30202	1.86683	3.13329	6.39289
alter random 10	2.0279	2.47492	3.60869	9.52572	18.8993
alter random 20	3.93068	3.36959	7.6448	11.4601	21.753
ignore random 5	1.1281	1.25704	1.53854	1.62059	5.85318
ignore random 10	2.04072	2.30651	2.61077	2.87703	5.88439
ignore random 20	2.26218	3.76167	3.47086	4.65255	6.43583

Table 3.11: Training times of MLP with various numbers of layers (specified in first column) in ms.

num nodes per layers	5	10	20	50	100
original grid 5	0.000603	0.000903	0.001517	0.00365	0.008255
original grid 10	0.000801	0.001574	0.0032	0.006582	0.061216
original grid 20	0.001358	0.002902	0.018401	0.106137	0.160671
alter grid 5	0.000548	0.004897	0.006359	0.059352	0.037784
alter grid 10	0.000789	0.001538	0.002792	0.007274	0.017855
alter grid 20	0.001442	0.006216	0.005667	0.01401	0.034686
ignore grid 5	0.000569	0.002224	0.001335	0.003148	0.008368
ignore grid 10	0.000815	0.001623	0.002759	0.007046	0.024204
ignore grid 20	0.00141	0.005738	0.005876	0.013202	0.053896
original random 5	0.000665	0.001015	0.002543	0.00793	0.017832
original random 10	0.000844	0.001504	0.009309	0.019832	0.041984
original random 20	0.005239	0.026566	0.019263	0.014728	0.033988
alter random 5	0.00077	0.001096	0.00353	0.004492	0.009771
alter random 10	0.000814	0.002217	0.012238	0.011358	0.039301
alter random 20	0.001434	0.004267	0.017338	0.042809	0.140851
ignore random 5	0.000575	0.000891	0.001311	0.003415	0.007812
ignore random 10	0.000759	0.001602	0.002835	0.011025	0.036706
ignore random 20	0.001435	0.002928	0.005149	0.019722	0.036042

Table 3.12: Testing times of MLP with various numbers of layers (specified in first column) in ms.

rate is roughly constant. For the altered model, the training time growth rate decreases. Testing times are constant, since once trained, the SVM is simple mathematic function



(a) (b)

(c) (d)

Figure 3.5: Performance of MLP surrogates on the PTO_{same} model.

C	5e5	1e6	5e6	1e7
original grid	0.7	0.72	0.75	0.77
alter grid	0.77	0.77	0.78	0.78
ignore grid	0.97	0.97	0.98	0.99
original random	0.75	0.78	0.81	0.84
alter random	0.81	0.81	0.82	0.83
ignore random	0.96	0.97	0.98	0.99

Table 3.13: R^2 of SVM.

C	5e5	1e6	5e6	1e7
original grid	6.21474e+10	5.92491e+10	5.11408e+10	4.81844e+10
alter grid	2.53475e+10	2.57161e+10	2.44768e+10	2.43387e+10
ignore grid	3.51817e+09	2.75464e+09	1.61033e+09	1.22708e+09
original random	5.44936e+10	4.80552e+10	4.0479e+10	3.54547e+10
alter random	2.16178e+10	2.12082e+10	1.9799e+10	1.90192e+10
ignore random	3.95997e+09	3.10422e+09	1.69121e+09	1.37044e+09

Table 3.14: MSE of SVM.

C	5e5	1e6	5e6	1e7
original grid	1.83465	4.57913	39.2633	225.665
alter grid	2.04234	1.74046	34.336	47.4818
ignore grid	1.30937	4.22366	22.6675	88.9345
original random	2.57153	4.34479	30.3103	164.272
alter random	1.52289	2.91651	24.9734	27.4269
ignore random	1.32607	3.14364	39.5588	98.7242

Table 3.15: Training times of SVM in ms.

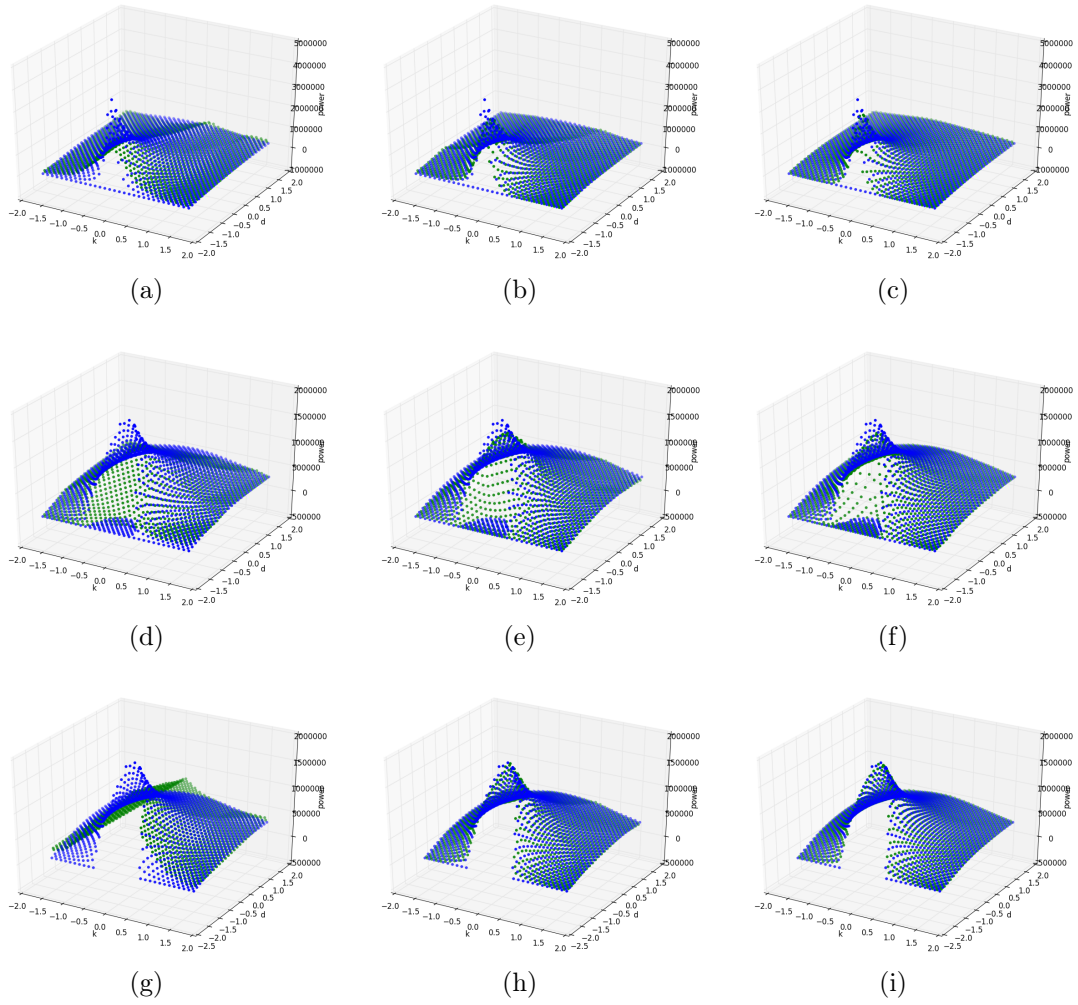


Figure 3.6: Plots of the MLP approximations of the validation data (green), and the training data (blue). The grid datasets were used for this visualisation. All MLP displayed have 10 layers. Each row corresponds to a different sub-model; the first is original, followed by altered, then ignorant. Each column corresponds to different number of of nodes per layer; in the first, there is 10, followed by 20, then 100. We can see the smoothness of the plots increase from left to right.

C	5e5	1e6	5e6	1e7
original grid	0.228935	0.067067	0.140404	0.067538
alter grid	0.22615	0.066508	0.132803	0.067334
ignore grid	0.21254	0.065019	0.061428	0.061067
original random	0.226605	0.068069	0.06996	0.067324
alter random	0.231063	0.22952	0.225929	0.228544
ignore random	0.202138	0.061581	0.061489	0.060348

Table 3.16: Testing times of SVM in ms.

that the sample to classify can be given to.

This surrogate best approximated the ignorant sub-model, followed by the altered sub-model, and finally, the original sub-model. This is different compared to the previous regression surrogates

Across all sub-models, as the value of C increases, so too does the accuracy and precision. Recall that C determines the robustness against outliers; the lower the value of C, the more it ignores outliers and less likely it is to overfit. Since the state-of-the-art

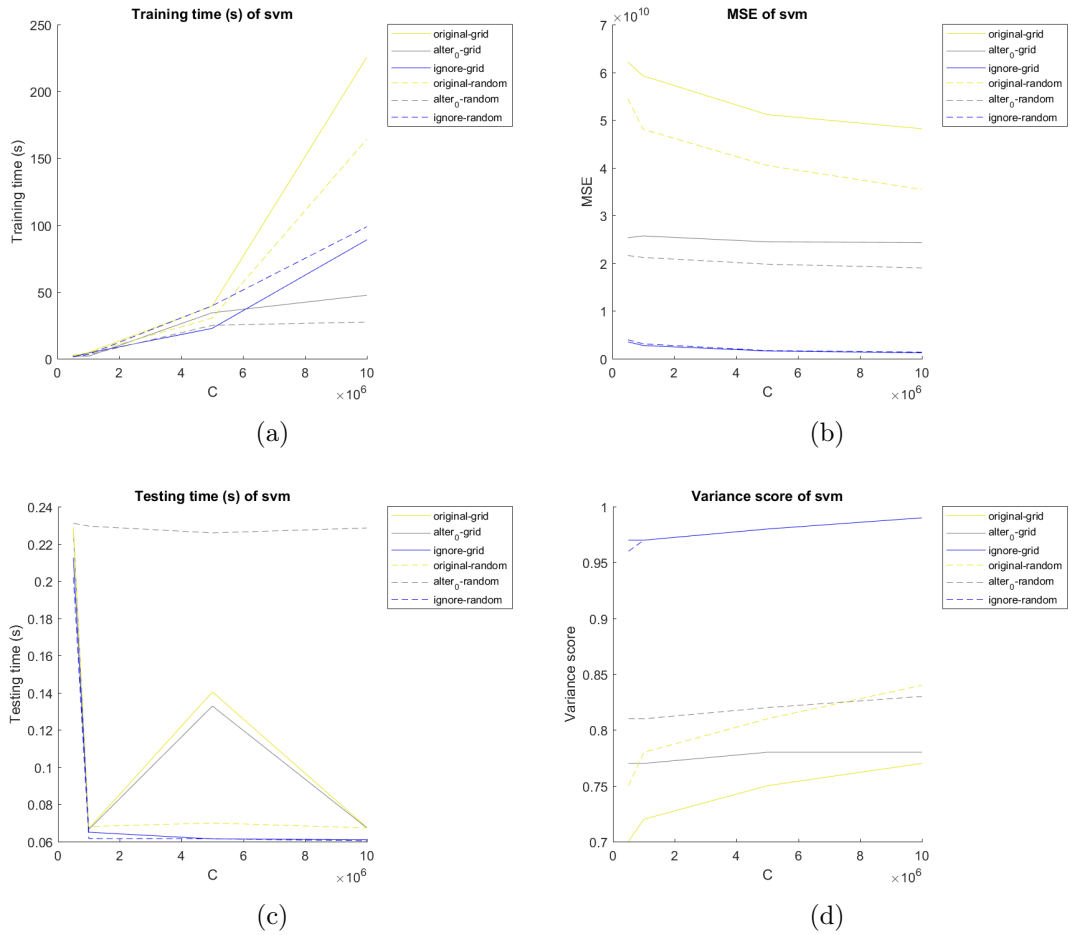


Figure 3.7: Performance of SVM surrogates on the PTO_{same} model.

model we are trying to approximate is mathematically based, and therefore completely accurate, it has no outliers. This is why we chose high values for C, and the reason an increase in C, causes an increase in performance. See also Figure 3.8 for visualisations of the approximations.

3.1.2 Comparing approaches

This section shows the surrogates best performing settings for a given sub-model, and compares them. The best settings for a surrogate were found by ordering them by MSE. It also gives us an idea of how easy each of the sub-models are to approximate; the ignorant sub-model has the lowest error, followed by the original, and finally the altered. This is expected as the smoothness of the sub-models decreases and complexity increases.

Original sub-model

approach and settings	MSE	variance	fitting time	predict time
knn distance random k=7	1.32347e+09	0.99	0.001333	0.004845
rfr grid num trees=100	1.05823e+09	0.99	0.469036	0.069045
mlp random num layers=10 num nodes per layer=100	1.26476e+09	0.99	12.8903	0.041984
svm random C=10000000	3.54547e+10	0.84	164.272	0.067324

Table 3.17: Performance of best surrogates of each approach on the original sub-model.

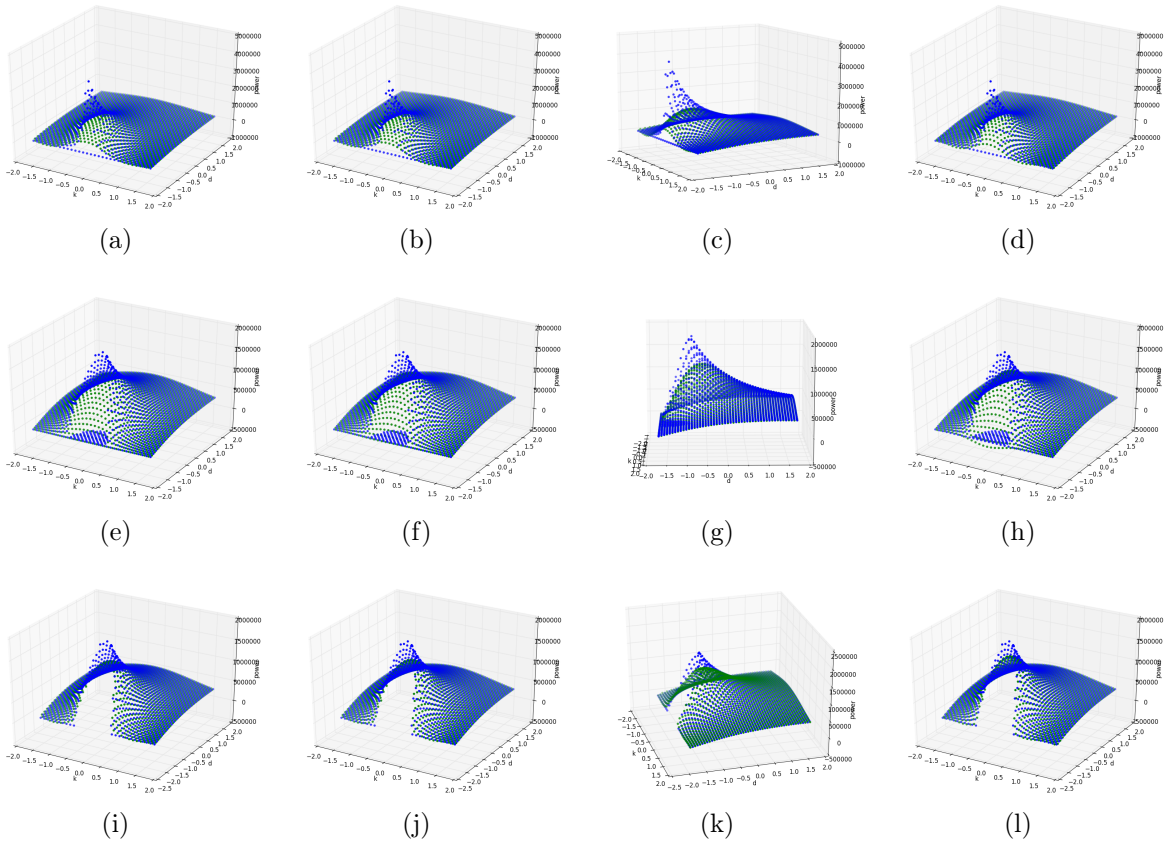


Figure 3.8: Plots of the SVM approximations of the validation data (green), and the training data (blue). The grid datasets were used for this visualisation. Each row corresponds to a different sub-model; the first is original, followed by altered, then ignorant. Each column corresponds to different value of C; in the first, there is 5×10^5, followed by 1×10^6, then 1×10^6, but at a different angle, and finally 1×10^7.

The results are depicted in Table 3.17. From there, we can see the best performing approach by both accuracy and precision was the random forest regressor, followed by MLP, KNN and then SVM. In addition, 3 out of the 4 surrogates performed better when trained on the random dataset. In terms of training time, the worst was SVM by far, followed by MLP. However, prediction times were similar across surrogates.

Ignorant sub-model

approach and settings	MSE	variance	fitting time	predict time
knn distance grid k=4	7.65866e+07	1	0.001489	0.005284
rfr grid num trees=100	8.67597e+07	1	0.658575	0.17907
mlp grid num layers=10 num nodes per layer=50	8.92232e+07	1	4.15596	0.007046
svm grid C=10000000	1.22708e+09	0.99	88.9345	0.061067

Table 3.18: Performance of best surrogates of each approach on the ignorant sub-model.

The results are depicted in Table 3.18. In this case, the surrogates trained on the grid data perform the best. The most accurate and precise approach was KNN, followed by random forest, MLP, and SVM. Training and prediction times are similar to those for the original sub-model.

Altered sub-model

approach and settings	MSE	variance	fitting time	predict time
knn uniform random k=3	4.20432e+09	0.96	0.001195	0.003638
rfr grid num trees=50	4.49739e+09	0.96	0.278847	0.033832
mlp random num layers=20 num nodes per layer=100	3.4753e+09	0.97	21.753	0.140851
svm random C=10000000	1.90192e+10	0.83	27.4269	0.228544

Table 3.19: Performance of best surrogates of each approach on the altered sub-model.

The results are depicted in Table 3.19. The surrogates trained on the random data perform the best. The most accurate and precise approach was MLP, followed by KNN, random forest, and SVM. Training and prediction times are similar to those for the original sub-model. The MLP's many degrees of freedom that come with the many weights in the layers, allow it to perform better than other surrogates when approximating the steep cliff.

3.2 Classification

When classifying feasibility, the original sub-model is the one we consider. We could also have classified the altered sub-model, since power is the only difference between the sub-models, and we consider feasibility, which is the same for both. However. the ignorant sub-model would not be useful, as it removes infeasible samples.

3.2.1 By approach

In the following section, we assess the experiment results by approach, as we did in the regression section. Some of the performance trends are similar to those seen in regression, and explanations of these are omitted. Refer to the regression section as appropriate for explanations.

KNN

k	1	2	3	4	5	7	10	15
original grid	0.99	0.99	1	1	1	1	1	1
original random	0.99	0.99	1	0.99	0.99	1	1	1

k	1	2	3	4	5	7	10	15
original grid	0.99	0.99	1	1	1	1	1	1
original random	0.99	0.99	1	0.99	0.99	1	0.99	1

Table 3.20: R^2 of KNN regressors. The upper table uses the distance metric, while the lower one uses the uniform metric.

k	1	2	3	4	5	7	10	15
original grid	0.01	0.01	0	0	0	0	0	0
original random	0.01	0.01	0	0.01	0.01	0	0	0

k	1	2	3	4	5	7	10	15
original grid	0.01	0.01	0	0	0	0	0	0
original random	0.01	0.01	0	0.01	0.01	0	0.01	0

Table 3.21: MSE of KNN in ms. The upper table uses the distance metric, while the lower one uses the uniform metric.

k	1	2	3	4	5	7	10	15
original grid	0.001884	0.001968	0.001962	0.001575	0.001763	0.001656	0.001383	0.002018
original random	0.001551	0.001524	0.001532	0.001511	0.001725	0.001483	0.0024	0.002258
k	1	2	3	4	5	7	10	15
original grid	0.001817	0.001483	0.001464	0.001538	0.001495	0.001501	0.001511	0.001502
original random	0.001624	0.001345	0.001551	0.001498	0.001454	0.001701	0.001513	0.0015

Table 3.22: Training time of KNN regressors. The upper table uses the distance metric, while the lower one uses the uniform metric.

k	1	2	3	4	5	7	10	15
original grid	0.005084	0.005302	0.004714	0.004065	0.007347	0.005582	0.007162	0.007623
original random	0.003475	0.003832	0.004056	0.004297	0.004779	0.005236	0.006037	0.008015
k	1	2	3	4	5	7	10	15
original grid	0.00388	0.003788	0.003885	0.004074	0.004471	0.004919	0.005389	0.006453
original random	0.003878	0.003963	0.004794	0.004098	0.004819	0.005018	0.005664	0.007032

Table 3.23: Predicting times of KNN in ms. The upper table uses the distance metric, while the lower one uses the uniform metric.

The raw KNN results are in Tables 3.20 to 3.23. These are visualised in Figure 3.9. When comparing training on the grid or random datasets, the grid datasets perform slightly better for some larger values of k, in terms of accuracy and precision. When comparing performance of surrogates using the distance metric or the uniform metric, the only case in which the distance metric performs better is when k=10. However, it is important to note that all precision errors are less than or equal to 0.01, and all accuracy values exceed 0.99.

The slight trends in accuracy and precision and k is similar to those in regression; there is a peak around 3 and then it drops and increases again. Note the accuracy and precision already start quite high, so there is little room for improvement.

Training time is constant in k, as with regression. Testing time however, is dependent on k, and we can see it increases approximately linearly, as with regression. See also Figure 3.10 for visualisations of the classifications.

Random Forest

num trees	1	5	10	20	50	100
original grid	0.99	0.99	0.99	0.99	1	1
original random	0.99	0.99	0.99	0.99	1	1

Table 3.24: R^2 of RFR.

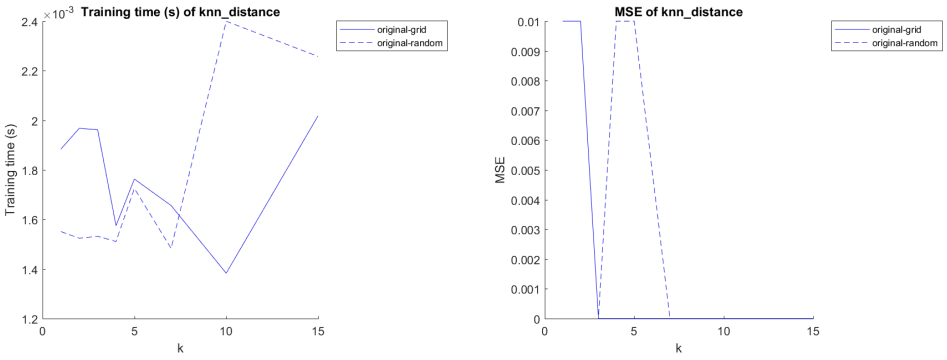
num trees	1	5	10	20	50	100
original grid	0.01	0.01	0.01	0.01	0	0
original random	0.01	0.01	0.01	0.01	0	0

Table 3.25: MSE of RFR.

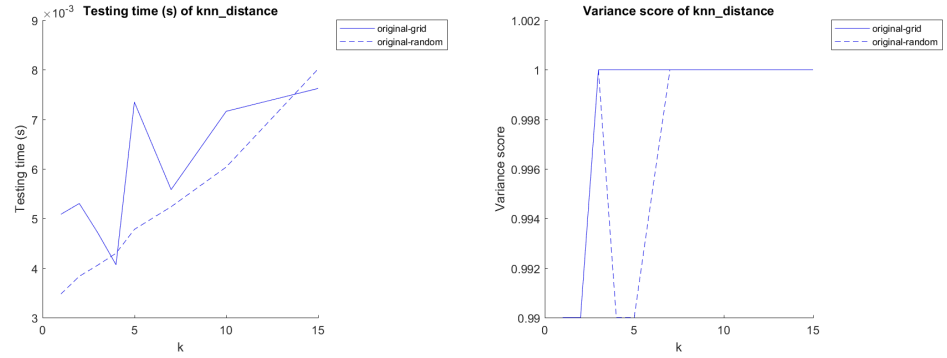
num trees	1	5	10	20	50	100
original grid	0.005678	0.041603	0.074327	0.144531	0.187692	0.370053
original random	0.004899	0.017483	0.035715	0.072112	0.180737	0.374436

Table 3.26: Training times of RFR in ms.

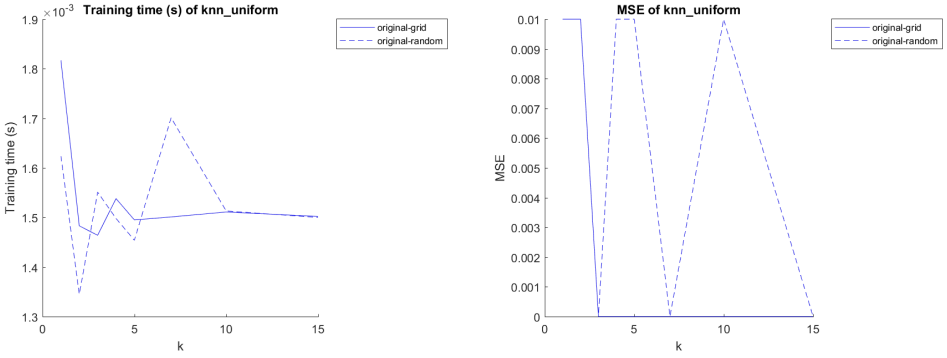
The raw random forest regressor (RFR) results are in Tables 3.24 to 3.27. These are visualised in Figure 3.11.



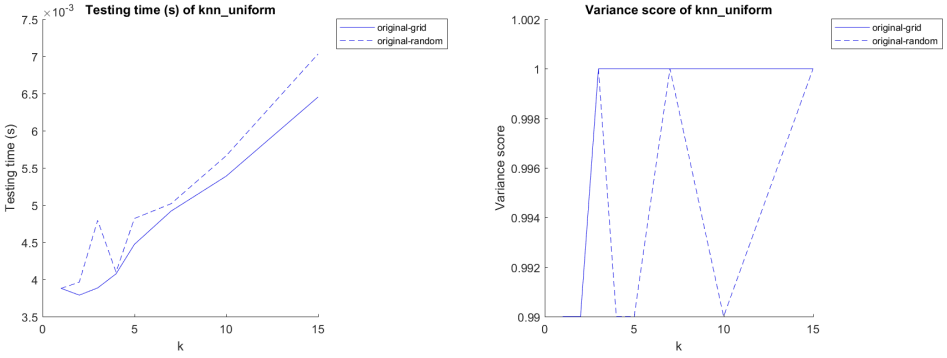
(a) (b)



(c) (d)

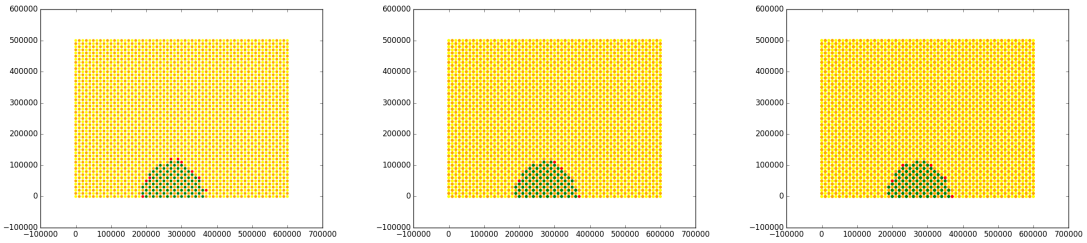


(e) (f)



(g) (h)

Figure 3.9: Performance of KNN regression surrogates on the PTO_{same} model.

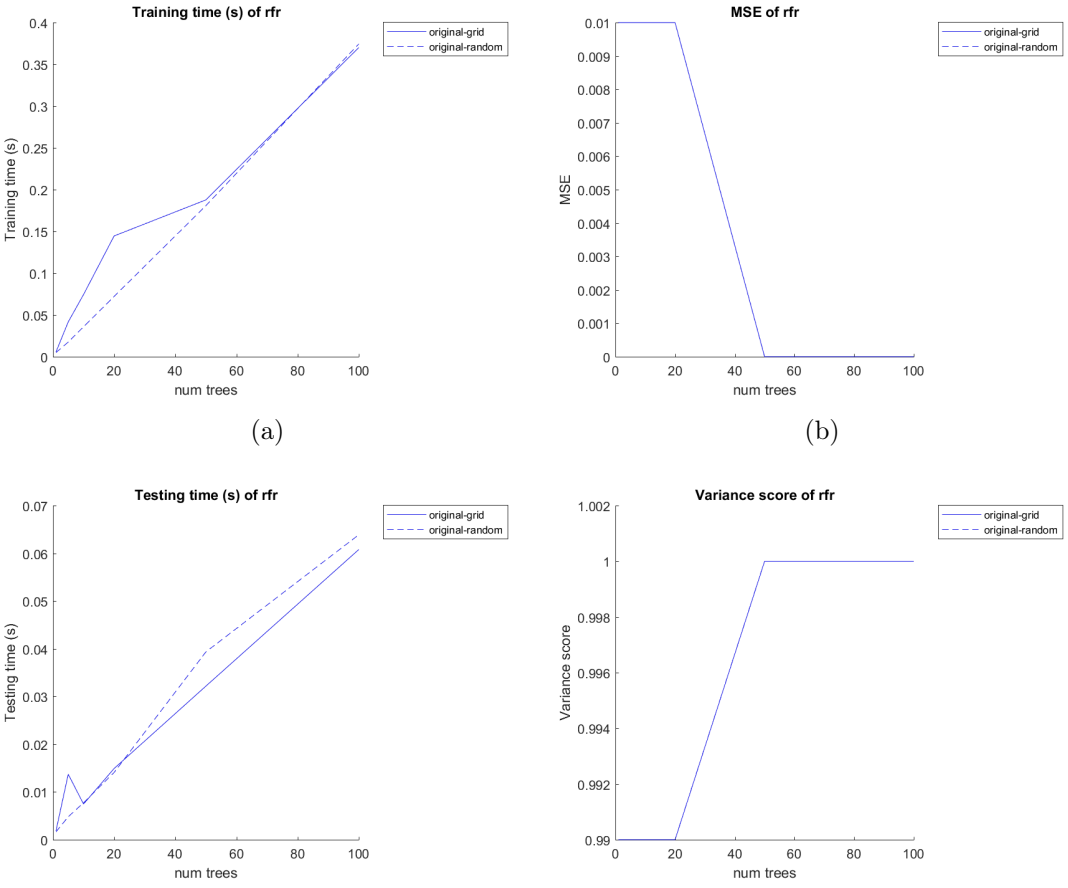


(a) (b) (c)

Figure 3.10: Plots of the KNN classifications of the validation data (orange if correctly classified, red if incorrectly classified), and the training data (yellow if feasible, green if infeasible). The grid datasets were used for this visualisation. Each column corresponds to different value of K; in the first, there is 1, followed by 3, then 7.

num trees	1	5	10	20	50	100
original grid	0.001852	0.013712	0.007468	0.01491	0.032168	0.060786
original random	0.00164	0.004796	0.007725	0.014077	0.039337	0.063883

Table 3.27: Testing times of RFR in ms.



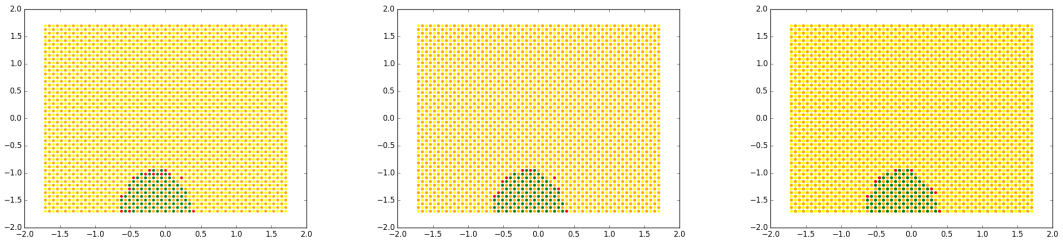
(a) (b)

(c) (d)

Figure 3.11: Performance of RFR surrogates on the PTO_{same} model.

The RFR surrogates trained on the grid dataset or random data set perform the same, in terms of accuracy and precision. The runtimes for both are similar.

As the number of trees increases, the accuracy and precision increase, as with regres-



(a) (b) (c)

Figure 3.12: Plots of the random forest classifications of the validation data (orange if correctly classified, red if incorrectly classified), and the training data (yellow if feasible, green if infeasible). The grid datasets were used for this visualisation. Each column corresponds to different numbers of trees; in the first, there is 1, followed by 5, then 20.

sion. However, the trend is slight, as with KNN when classifying, as the precision and accuracy are already high for a small number of trees. Both training time and testing time are linear in the number of trees, as with regression. See also Figure 3.12 for visualisations of the classifications.

MLP

num nodes per layer	5	10	20	50	100
original grid 5	1	1	1	1	0.99
original grid 10	1	1	1	1	0.99
original grid 20	0.95	1	1	0.99	0.99
original random 5	0.99	1	1	0.98	1
original random 10	1	1	1	0.99	1
original random 20	0.95	0.99	0.99	0.99	0.99

Table 3.28: R^2 of MLP with various numbers of layers (specified in first column).

num nodes per layer	5	10	20	50	100
original grid 5	0	0	0	0	0.01
original grid 10	0	0	0	0	0.01
original grid 20	0.05	0	0	0.01	0.01
original random 5	0.01	0	0	0.02	0
original random 10	0	0	0	0.01	0
original random 20	0.05	0.01	0.01	0.01	0.01

Table 3.29: MSE of MLP with various numbers of layers (specified in first column).

num nodes per layer	5	10	20	50	100
original grid 5	1.17321	0.703144	0.476445	0.560678	1.25484
original grid 10	1.02293	0.934789	0.565577	1.86965	3.07215
original grid 20	0.704494	1.70507	1.54192	1.43353	2.60192
original random 5	1.21675	0.64858	0.405209	0.460994	1.0156
original random 10	1.1043	1.16102	0.759024	0.774866	1.31842
original random 20	0.694574	1.19609	1.56808	1.38738	2.49439

Table 3.30: Training times of MLP with various numbers of layers (specified in first column) in ms.

The raw MLP results are in Tables 3.28 to 3.31. These are visualised in Figure 3.13.

num nodes per layer	5	10	20	50	100
original grid 5	0.000596	0.000945	0.001964	0.003492	0.01907
original grid 10	0.000909	0.001343	0.002268	0.014631	0.034571
original grid 20	0.00146	0.006437	0.007369	0.01385	0.037102
original random 5	0.000677	0.001057	0.001988	0.003757	0.009085
original random 10	0.000874	0.001332	0.002397	0.007346	0.016844
original random 20	0.001481	0.002455	0.00513	0.013837	0.041012

Table 3.31: Testing times of MLP with various numbers of layers (specified in first column) in ms.

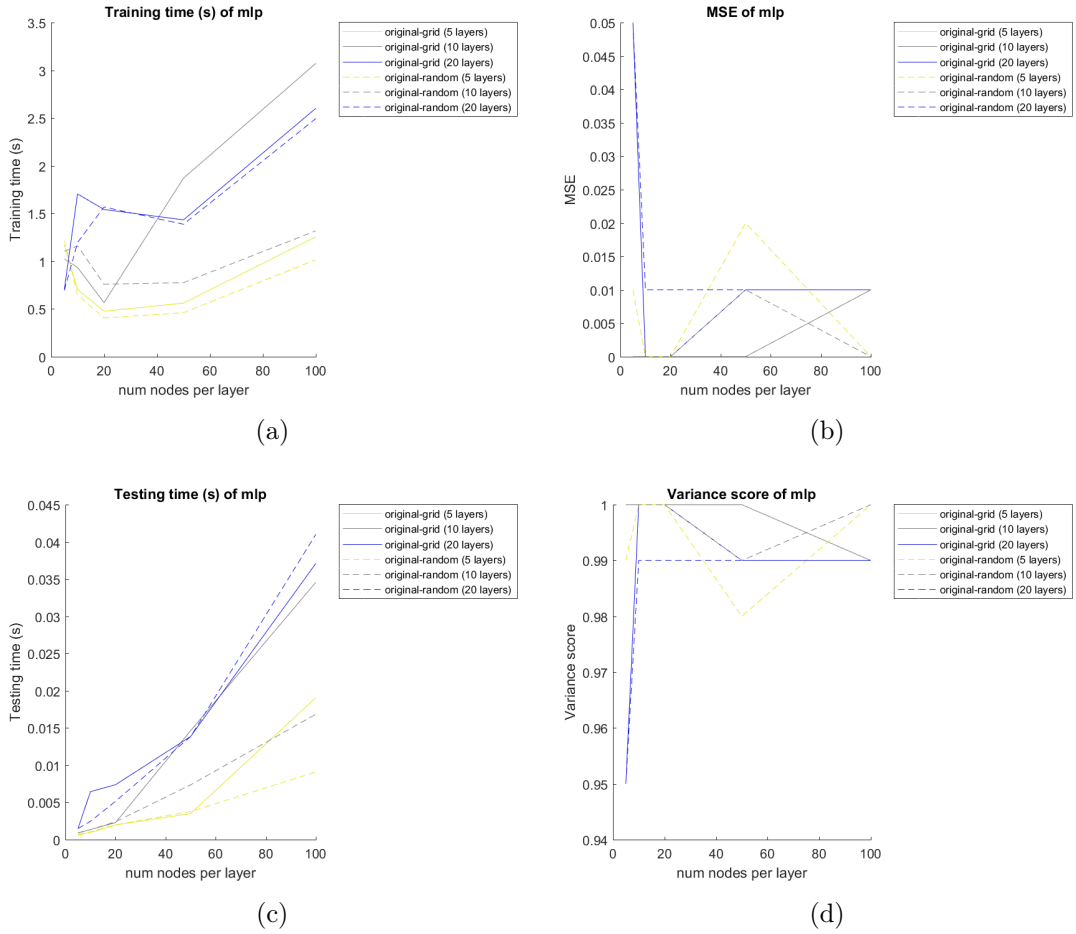


Figure 3.13: Performance of MLP surrogates on the PTO_{same} model.

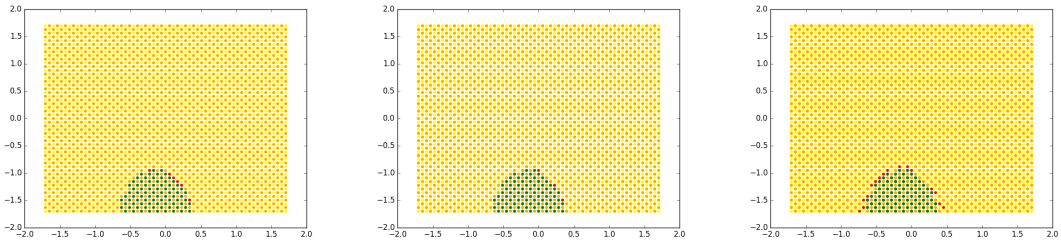
The MLP surrogates trained on the grid dataset perform slightly better in terms of accuracy and precision, but they are comparable. Both training time and testing time are linear in the number of nodes per layer, as with regression.

In regard to the number of nodes in each layer, the accuracy and precision peak at around 10 to 20 nodes per layer. This is likely due to overfitting when more nodes are used. See also Figure 3.14 for visualisations of the classifications.

SVM

C	10000	100000	1000000	10000000
original grid	1	1	1	1
original random	1	1	1	1

Table 3.32: R^2 of SVM.



(a) (b) (c)

Figure 3.14: Plots of the MLP classifications of the validation data (orange if correctly classified, red if incorrectly classified), and the training data (yellow if feasible, green if infeasible). The grid datasets were used for this visualisation. All MLP surrogates displayed have 5 layers. Each column corresponds to different numbers of nodes per layer; the first has 10, followed by 20, then 100.

C	10000	100000	1000000	10000000
original grid	0	0	0	0
original random	0	0	0	0

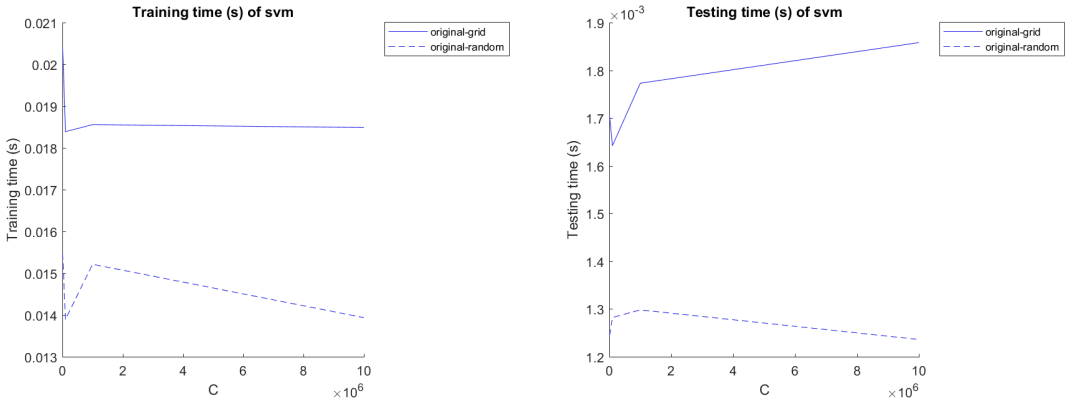
Table 3.33: MSE of SVM.

C	10000	100000	1000000	10000000
original grid	0.020391	0.018388	0.018557	0.018487
original random	0.01547	0.013892	0.015212	0.013935

Table 3.34: Training times of SVM in ms.

C	10000	100000	1000000	10000000
original grid	0.001704	0.001642	0.001773	0.001858
original random	0.001244	0.001282	0.001298	0.001236

Table 3.35: Testing times of SVM in ms.

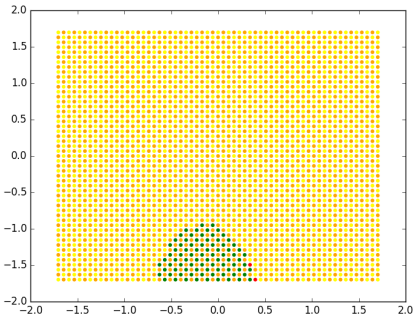


(a) (b)

Figure 3.15: Performance of SVM surrogates on the PTO_{same} model.

The raw SVM results are in Tables 3.32 to 3.35. Only the runtimes are visualised in Figure 3.15.

All SVM classifiers with various C achieved perfect precision and accuracy. Both the training times and testing times are constant in C. See Figure 3.16 for visualisations of the classifications.



(a)

Figure 3.16: Plots of the SVM classifications of the validation data (orange if correctly classified, red if incorrectly classified), and the training data (yellow if feasible, green if infeasible). The grid datasets were used for this visualisation. All values of C resulted in the same classifications.

3.2.2 Comparing approaches

approach and settings	MSE	variance	fitting time	predict time
knn uniform grid k=3	0	1	0.001464	0.003885
rfr grid num trees=50	0	1	0.187692	0.032168
mlp grid num layers=5 num nodes per layer=5	0	1	1.17321	0.000596
svm grid C=10000	0	1	0.020391	0.001704

Table 3.36: Performance of best surrogates of each approach.

As mentioned previously, we only approximate the original sub-model. The results can be found in Table 3.36. There we see that all surrogates, when using the best settings, achieve perfect accuracy and precision. All fitting and predicting times are comparable.

Chapter 4

Approximating Power Absorption of Arrays of WEC with individual PTO Settings

In this chapter, we look at the performance of each surrogate for both regression and classification on the PTO_{indiv} model. The metrics are the same as in the previous chapter.

4.1 Regression

Below are the results when trying to approximate all 3 sub-models, with various surrogates.

4.1.1 By approach

KNN

k	1	2	3	4	5	7	10	15
original grid	0.7	0.73	0.76	0.78	0.76	0.73	0.71	0.71
alter grid	0.6	0.66	0.73	0.76	0.73	0.69	0.67	0.67
ignore grid	0.8	0.83	0.83	0.83	0.82	0.82	0.82	0.82
original random	0.39	0.57	0.62	0.64	0.65	0.66	0.67	0.66
alter random	0.08	0.35	0.43	0.47	0.48	0.5	0.51	0.5
ignore random	0.69	0.8	0.83	0.85	0.86	0.86	0.86	0.86

k	1	2	3	4	5	7	10	15
original grid	0.7	0.73	0.76	0.77	0.74	0.71	0.69	0.7
alter grid	0.6	0.66	0.73	0.74	0.71	0.66	0.64	0.65
ignore grid	0.8	0.83	0.82	0.82	0.82	0.82	0.82	0.81
original random	0.39	0.56	0.61	0.63	0.65	0.66	0.66	0.66
alter random	0.08	0.34	0.42	0.46	0.47	0.49	0.5	0.49
ignore random	0.69	0.8	0.83	0.84	0.85	0.86	0.86	0.85

Table 4.1: R^2 of KNN regressors. The upper table uses the distance metric, while the lower one uses the uniform metric.

The raw KNN results are in Tables 4.1 to 4.4. These are visualised in Figure 4.1. There isn't a large disparity between the surrogates using the distance weighting compared to the uniform weighting, though the distance metric is slightly better, as it was when approximating the PTO_{same} model.

Training time is constant in k, and testing time is linear in k, as it was with the $PTOsame$ model.

k	1	2	3	4	5	7	10	15
original grid	1.24107e10	1.13505e10	9.96254e9	9.30821e9	1.00307e10	1.11268e10	1.19381e10	1.19489e10
alter grid	2.454e10	2.0733e10	1.63091e10	1.45883e10	1.60989e10	1.85356e10	2.01107e10	2.0217e10
ignore grid	8.05377e9	7.11374e9	7.06364e9	6.91478e9	7.13934e9	7.24377e9	7.21462e9	7.41612e9
original random	4.36651e10	3.09726e10	2.74873e10	2.5756e10	2.48314e10	2.40554e10	2.37825e10	2.40517e10
alter random	6.83335e10	4.85439e10	4.23357e10	3.97844e10	3.85267e10	3.72096e10	3.66805e10	3.7016e10
ignore random	8.40183e9	5.47637e9	4.53573e9	4.13861e9	3.92787e9	3.72109e9	3.6941e9	3.83715e9
k	1	2	3	4	5	7	10	15
original grid	1.24107e10	1.13505e10	1.00851e10	9.67918e9	1.07067e10	1.20099e10	1.27596e10	1.24874e10
alter grid	2.454e10	2.0733e10	1.65678e10	1.55001e10	1.76018e10	2.04569e10	2.18067e10	2.13706e10
ignore grid	8.05377e9	7.11374e9	7.15917e9	7.11438e9	7.41543e9	7.49317e9	7.38014e9	7.53033e9
original random	4.36651e10	3.11855e10	2.78329e10	2.6136e10	2.52405e10	2.45107e10	2.42746e10	2.45892e10
alter random	6.83335e10	4.89459e10	4.27955e10	4.0329e10	3.91367e10	3.78762e10	3.73969e10	3.78124e10
ignore random	8.40183e9	5.57407e9	4.64682e9	4.26135e9	4.05785e9	3.85776e9	3.84785e9	4.01348e9

Table 4.2: MSE of KNN in ms. The upper table uses the distance metric, while the lower one uses the uniform metric.

k	1	2	3	4	5	7	10	15
original grid	0.568326	0.566752	0.569379	0.570734	0.589429	0.569621	0.562823	0.643006
alter grid	0.570077	0.542975	0.539788	0.542766	0.55781	0.544841	0.543328	0.538397
ignore grid	0.27196	0.273296	0.263386	0.265752	0.256417	0.25355	0.280436	0.286804
original random	0.178001	0.162985	0.17932	0.16239	0.158882	0.155285	0.150468	0.150177
alter random	0.185248	0.155642	0.163516	0.152239	0.154827	0.144981	0.182292	0.168721
ignore random	0.144506	0.148761	0.12933	0.17538	0.145261	0.141728	0.133345	0.130344
k	1	2	3	4	5	7	10	15
original grid	0.550793	0.562233	0.563579	0.569762	0.566039	0.563299	0.573152	0.563819
alter grid	0.580746	0.574492	0.570568	0.536818	0.551801	0.537634	0.573264	0.533546
ignore grid	0.250488	0.267567	0.266577	0.276841	0.275514	0.272745	0.271444	0.265103
original random	0.185477	0.193783	0.181851	0.170315	0.169567	0.163685	0.161948	0.170302
alter random	0.153918	0.177575	0.177419	0.17879	0.190427	0.172294	0.176629	0.173302
ignore random	0.134419	0.130394	0.144958	0.131171	0.133648	0.146501	0.131114	0.127096

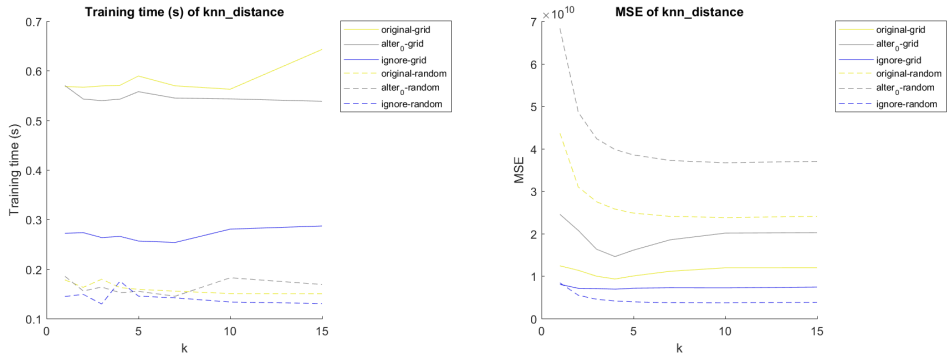
Table 4.3: Training time of KNN regressors. The upper table uses the distance metric, while the lower one uses the uniform metric.

k	1	2	3	4	5	7	10	15
original grid	5.84546	6.01509	6.83683	9.33861	12.0302	11.3855	11.8065	12.2284
alter grid	5.81105	5.97373	6.86082	9.31841	11.3173	11.333	12.005	12.18
ignore grid	4.87106	5.03382	6.6224	8.62464	9.365	9.57345	10.0276	10.5153
original random	5.32886	7.35426	8.48218	8.93328	9.80013	10.9362	12.5509	14.5269
alter random	5.97034	7.71617	7.78287	8.71848	9.80329	10.993	18.9494	20.2217
ignore random	4.28692	5.05943	5.79354	7.20465	7.92758	9.49188	9.56854	10.7764
k	1	2	3	4	5	7	10	15
original grid	5.79381	6.0031	6.83393	9.32618	11.1408	11.4016	11.594	12.1368
alter grid	5.78197	5.99227	6.82167	9.30563	11.0706	11.361	11.6345	11.9363
ignore grid	4.49933	4.84903	6.09515	8.98017	9.12933	9.34446	9.53268	10.2404
original random	7.40443	9.73348	10.8984	11.5555	9.75052	12.4954	14.2054	15.6363
alter random	5.4918	7.43429	8.1023	9.2243	10.9648	11.4665	13.7523	15.185
ignore random	3.99444	5.42671	6.19026	7.1388	8.2663	9.04232	10.3495	12.467

Table 4.4: Predicting times of KNN in ms. The upper table uses the distance metric, while the lower one uses the uniform metric.

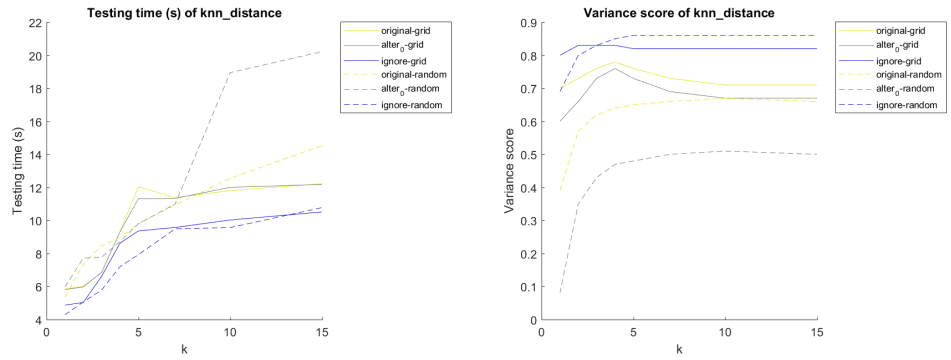
Across all sub-models, as the value of k increases up to a point, the accuracy and precision increase. When k is set to 15, performance is less than or equal to the performance for the previous value of k (10), but not by much. This is expected, as training samples much further away from the point to approximate will contribute to the approximation. There is a peak in performance at k=4, particularly for the surrogates trained on grid data, as with the PTO_{same} model. The peaks for the random data appear at around k=10. This is because of the higher dimensional space that the points exist in; we need more neighbours in order for the resulting power to not be skewed in one dimension.

The sub-model that the KNN regressor had the most success with is the ignorant sub-model, as with the *PTOsame* model.



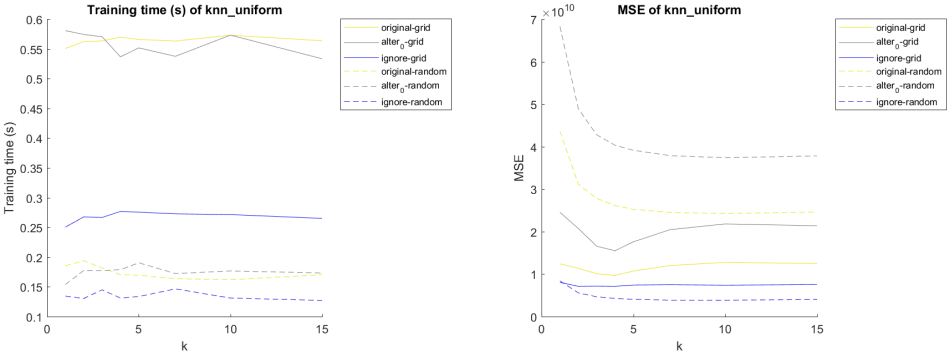
(a)

(b)



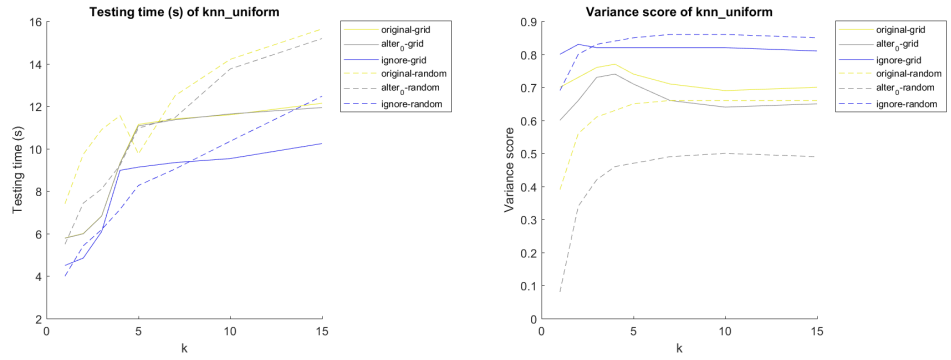
(c)

(d)



(e)

(f)



(g)

(h)

Figure 4.1: Performance of KNN regression surrogates on the PTO_{indiv} model.

Random Forest

num trees	1	5	10	20	50	100
original grid	0.88	0.89	0.89	0.89	0.89	0.89
alter grid	0.72	0.73	0.73	0.73	0.73	0.73
ignore grid	0.87	0.89	0.89	0.89	0.89	0.89
original random	0.66	0.85	0.88	0.89	0.9	0.9
alter random	0.65	0.83	0.85	0.86	0.87	0.87
ignore random	0.68	0.88	0.9	0.92	0.92	0.92

Table 4.5: R^2 of RFR.

num trees	1	5	10	20	50	100
original grid	5.07919e+09	4.64168e+09	4.60884e+09	4.56122e+09	4.56976e+09	4.5654e+09
alter grid	1.67868e+10	1.63301e+10	1.62687e+10	1.62384e+10	1.62101e+10	1.6206e+10
ignore grid	5.1149e+09	4.61757e+09	4.53606e+09	4.50566e+09	4.47553e+09	4.46791e+09
original random	2.46367e+10	1.05795e+10	8.8107e+09	7.89957e+09	7.33555e+09	7.1085e+09
alter random	2.62914e+10	1.25926e+10	1.08524e+10	1.01162e+10	9.65361e+09	9.56546e+09
ignore random	8.85889e+09	3.28787e+09	2.62165e+09	2.31897e+09	2.1271e+09	2.05818e+09

Table 4.6: MSE of RFR.

num trees	1	5	10	20	50	100
original grid	0.201784	0.76741	1.42661	2.88161	7.39967	14.8157
alter grid	0.217717	0.655286	1.31034	2.57029	6.48872	12.4097
ignore grid	0.144887	0.546366	1.12368	2.15438	5.24833	10.6879
original random	0.696134	3.25916	6.426	13.5476	32.5941	66.2652
alter random	0.742124	3.44088	7.11721	13.3224	33.5619	66.1411
ignore random	0.536155	2.55839	5.36377	10.0812	24.8115	50.2532

Table 4.7: Training times of RFR in ms.

num trees	1	5	10	20	50	100
original grid	0.05716	0.113602	0.168912	0.299709	0.692808	2.43139
alter grid	0.065095	0.1032	0.195944	0.288331	0.577283	1.03815
ignore grid	0.049466	0.094716	0.165032	0.24896	0.535664	1.07542
original random	0.083092	0.205781	0.328623	0.609478	1.47084	3.03617
alter random	0.072603	0.168479	0.26853	0.484716	1.15009	2.20084
ignore random	0.058948	0.141848	0.258347	0.412792	1.00858	1.93987

Table 4.8: Testing times of RFR in ms.

The raw random forest regressor (RFR) results are in Tables 4.5 to 4.8. These are visualised in Figure 4.2.

Across all sub-models, as the number of trees increases, the accuracy and precision increase. This is expected based on the theory behind the approach. However, the improvement rate quickly decreases, and there is not much performance disparity for greater than 10 trees. Furthermore, there is no increase in accuracy for 50 or more trees.

The RFR surrogates trained on the random dataset have a better increase in performance (accuracy and precision) as the number of trees increases when compared to those trained on the grid dataset.

Both training time and testing time are linear in the number of trees, as with the *PTOsame* model.

The sub-model that this regressor had the most success with is the ignorant sub-model, followed by the original, and finally, the altered sub-model.

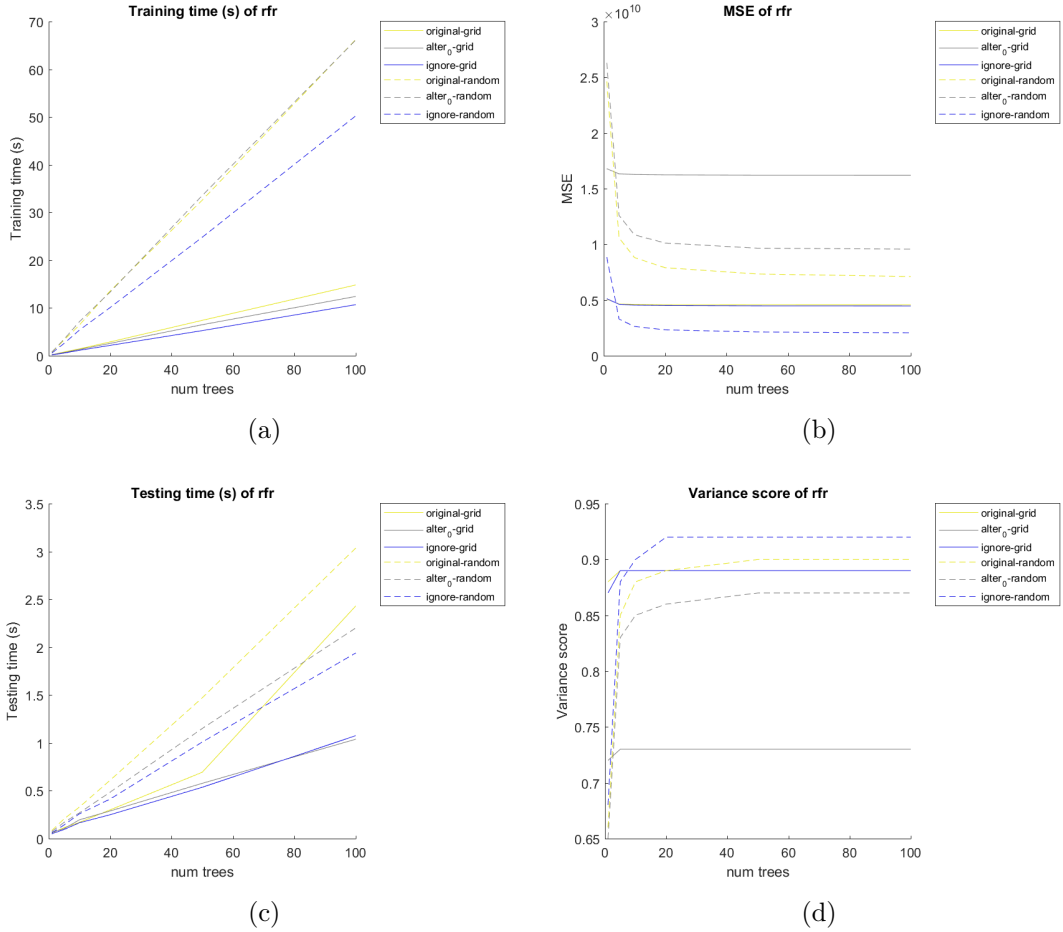


Figure 4.2: Performance of RFR surrogates on the PTO_{indiv} model.

num nodes per layers	5	10	20	50	100
original grid 5	0.09	0.09	0.75	0.85	0.85
original grid 10	0.26	0.67	0.85	0.85	0.86
original grid 20	-0.01	0.83	0.86	0.84	0.86
original grid 30	-0.01	0.75	0.83	0.79	0.84
alter grid 5	-0.01	-0.03	-0.42	0.57	0.59
alter grid 10	-0.09	0.07	0.76	0.61	0.82
alter grid 20	-0.04	0.49	0.7	0.7	0.62
alter grid 30	-0.04	0.44	0.74	0.68	0.51
ignore grid 5	0.07	0.08	0.75	0.87	0.85
ignore grid 10	0.49	0.58	0.87	0.85	0.86
ignore grid 20	0.26	0.77	0.83	0.84	0.86
ignore grid 30	-0	0.76	0.83	0.82	0.85
original random 5	0.03	0.03	0.03	0.97	0.96
original random 10	0.22	0.46	0.94	0.98	0.98
original random 20	-0	0.84	0.93	0.97	0.95
original random 30	-0	0.83	0.94	0.96	0.96
alter random 5	0.08	0.08	0.65	0.81	0.89
alter random 10	0.26	0.6	0.87	0.87	0.89
alter random 20	0.31	0.58	0.88	0.9	0.89
alter random 30	-0	0.69	0.9	0.87	0.89
ignore random 5	0.01	0.01	0.9	0.98	1
ignore random 10	0.22	0.66	0.96	0.97	0.99
ignore random 20	-0	0.89	0.99	0.99	0.98
ignore random 30	-0	0.81	0.98	0.99	0.97

Table 4.9: R^2 of MLP with various numbers of layers (specified in first column).

MLP

The raw MLP results are in Tables 4.9 to 4.12. These are visualised in Figure 4.3.

num nodes per layers	5	10	20	50	100
original grid 5	3.78052e+10	3.77344e+10	1.02513e+10	6.43484e+09	6.08582e+09
original grid 10	3.06631e+10	1.38241e+10	6.29317e+09	6.33701e+09	6.00875e+09
original grid 20	4.21469e+10	7.02344e+09	6.01832e+09	6.82331e+09	5.8697e+09
original grid 30	4.20482e+10	1.04068e+10	6.90021e+09	8.86907e+09	6.54052e+09
alter grid 5	6.12076e+10	6.23281e+10	8.65035e+10	2.58498e+10	2.46468e+10
alter grid 10	6.61898e+10	5.64941e+10	1.45178e+10	2.39765e+10	1.10008e+10
alter grid 20	6.29605e+10	3.09283e+10	1.84818e+10	1.83667e+10	2.30521e+10
alter grid 30	6.28834e+10	3.37889e+10	1.6058e+10	1.91652e+10	2.98589e+10
ignore grid 5	3.76034e+10	3.75957e+10	1.01009e+10	5.23944e+09	5.97691e+09
ignore grid 10	2.05321e+10	1.70562e+10	5.34941e+09	6.12782e+09	5.525e+09
ignore grid 20	3.00258e+10	9.30194e+09	6.8952e+09	6.51672e+09	5.89231e+09
ignore grid 30	4.07292e+10	9.70201e+09	6.87766e+09	7.15589e+09	5.97142e+09
original random 5	6.91272e+10	6.9181e+10	6.92914e+10	2.37056e+09	3.13527e+09
original random 10	5.57412e+10	3.88042e+10	4.16028e+09	1.41697e+09	1.28462e+09
original random 20	7.1585e+10	1.16782e+10	4.70864e+09	1.97033e+09	3.58683e+09
original random 30	7.15867e+10	1.19937e+10	4.53898e+09	3.15834e+09	3.13206e+09
alter random 5	6.85851e+10	6.85742e+10	2.57777e+10	1.44714e+10	7.85333e+09
alter random 10	5.53154e+10	3.01115e+10	1.00182e+10	9.71173e+09	8.24052e+09
alter random 20	5.1216e+10	3.15497e+10	8.96838e+09	7.59641e+09	8.28987e+09
alter random 30	7.43811e+10	2.31647e+10	7.33759e+09	9.95089e+09	8.53794e+09
ignore random 5	2.69663e+10	2.69658e+10	2.74148e+09	4.23814e+08	8.98119e+07
ignore random 10	2.11818e+10	9.34452e+09	1.21216e+09	7.56092e+08	3.83444e+08
ignore random 20	2.73211e+10	3.1084e+09	3.85575e+08	1.50943e+08	5.08294e+08
ignore random 30	2.73228e+10	5.16568e+09	4.27999e+08	2.02722e+08	9.04811e+08

Table 4.10: MSE of MLP with various numbers of layers (specified in first column).

num nodes per layers	5	10	20	50	100
original grid 5	34.1325	31.2733	472.424	842.945	941.627
original grid 10	56.0679	82.8897	369.538	1278.53	1012.52
original grid 20	18.6754	569.332	837.088	331.123	853.822
original grid 30	72.4169	303.843	351.608	434.267	340.846
alter grid 5	68.1349	67.6928	473.385	568.054	294.299
alter grid 10	144.894	156.003	380.112	238.788	428.067
alter grid 20	16.885	209.58	731.483	207.952	327.515
alter grid 30	114.346	158.095	325.939	344.238	539.077
ignore grid 5	15.8412	14.2907	146.657	120.825	154.623
ignore grid 10	62.011	60.7221	100.253	44.4589	89.5415
ignore grid 20	33.4368	136.978	115.159	125.122	135.705
ignore grid 30	50.5847	132.69	95.7118	81.7195	110.832
original random 5	27.6446	25.6622	24.9502	933.002	1452.89
original random 10	106.423	279.161	494.719	1129.06	2317.01
original random 20	14.4323	377.239	428.581	2237.29	775.611
original random 30	142.162	490.933	855.583	1505.25	1238.47
alter random 5	24.6806	33.0212	489.694	396.603	1069.05
alter random 10	306.644	284.423	260.937	372.574	977.292
alter random 20	182.791	195.099	676.151	1413.77	1197.43
alter random 30	126.089	335.68	1241.03	1605.36	2051.28
ignore random 5	7.75703	8.55562	118.739	122.372	278.531
ignore random 10	50.334	163.231	99.7826	239.668	344.404
ignore random 20	5.89693	146.183	282.601	97.4504	140.332
ignore random 30	49.3983	88.6905	156.334	161.193	143.052

Table 4.11: Training times of MLP with various numbers of layers (specified in first column) in ms.

The MLP surrogates trained on the grid dataset perform slightly better in terms of accuracy and precision. Both training time and testing time are linear in the number of nodes per layer, as shown by the graphs, as with the PTO_{same} model.

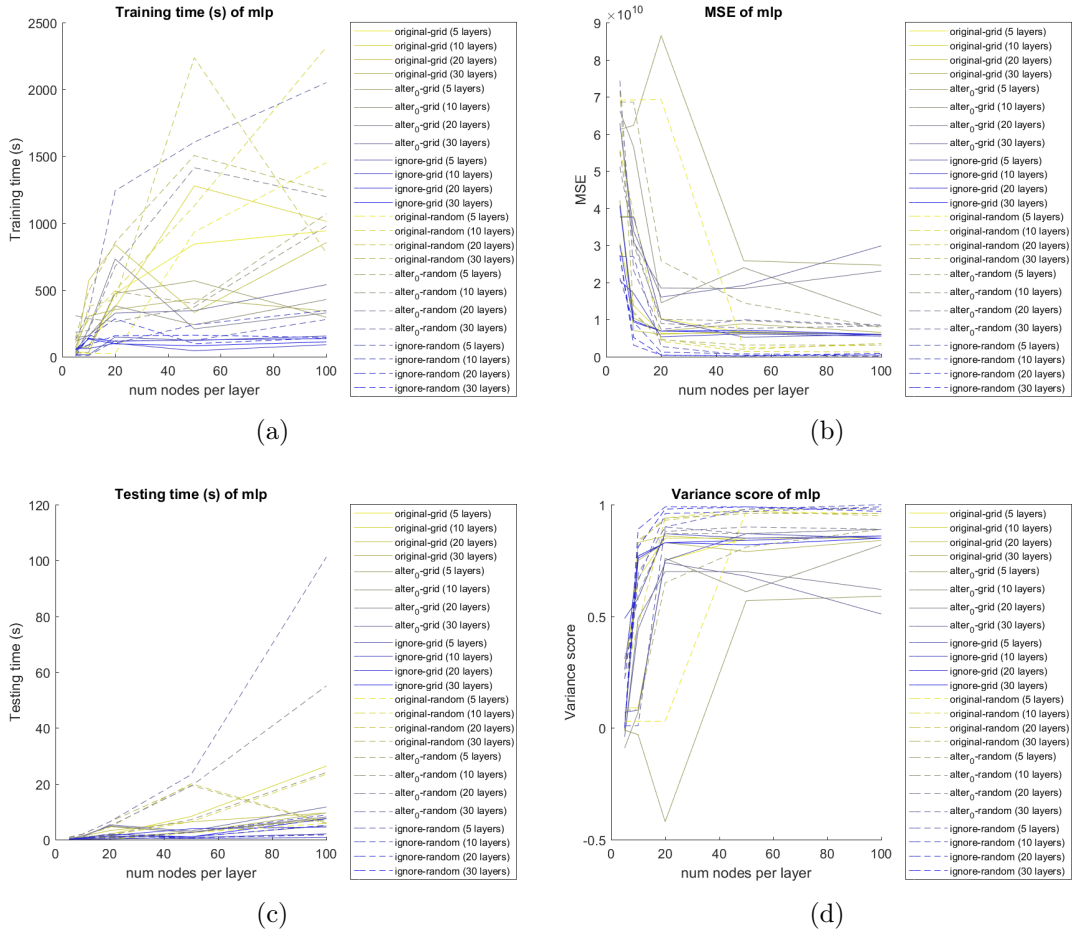
This surrogate best approximated the ignorant sub-model, followed by the original, and finally, the altered sub-model.

Across all sub-models, as the number of nodes in each layer increases, so too does the accuracy and precision. However, some configurations of number of layers and number of nodes per layer to not result in significant increases in accuracy and precision when the number of nodes per layer is increased.

The MLP regressor is prone to overfitting. Consider for example the R^2 of the MLP

num nodes per layers	5	10	20	50	100
original grid 5	0.036641	0.107019	0.729677	2.6989	7.53933
original grid 10	0.140285	0.78556	1.89934	8.25969	26.3505
original grid 20	0.19294	1.2203	3.21047	6.37275	9.53987
original grid 30	0.133366	1.40973	4.62041	2.56611	9.5443
alter grid 5	0.128472	0.368235	0.670431	2.57432	7.64013
alter grid 10	0.209381	0.800362	1.94215	0.980079	4.93754
alter grid 20	0.211439	1.32801	4.91054	2.47458	8.10823
alter grid 30	0.286467	0.81597	5.27002	3.04842	11.6736
ignore grid 5	0.020494	0.06789	0.172078	0.723508	2.03347
ignore grid 10	0.046825	0.240458	0.584301	0.315724	0.795353
ignore grid 20	0.08554	0.382402	1.42722	3.86009	4.47466
ignore grid 30	0.223625	0.613518	1.24874	0.96095	7.48051
original random 5	0.03973	0.149165	0.174436	2.43626	5.778
original random 10	0.110337	0.702319	1.51649	6.36162	23.3873
original random 20	0.161434	1.308	4.40915	19.985	6.02768
original random 30	0.901517	2.00715	6.33292	19.4132	5.68078
alter random 5	0.081493	0.332485	0.915917	2.58996	7.07269
alter random 10	0.202835	0.705126	1.67552	7.1121	24.0158
alter random 20	0.39113	1.32648	4.81347	19.1056	55.0744
alter random 30	0.707571	1.89664	6.3476	23.1924	101.246
ignore random 5	0.023603	0.120255	0.195697	0.700106	1.99808
ignore random 10	0.048605	0.247311	0.518466	2.54806	8.81281
ignore random 20	0.053158	0.396992	1.19894	0.638236	1.674
ignore random 30	0.165294	0.550224	1.97903	0.979731	5.09982

Table 4.12: Testing times of MLP with various numbers of layers (specified in first column) in ms.



(a) (b)

(c) (d)

Figure 4.3: Performance of MLP surrogates on the PTO_{indiv} model.

training on the altered using grid data, using 30 layers in the MLP; the accuracy peaks when the number of nodes in each layer is 20. Similar cases can also be found in Table

4.9.

4.1.2 Comparing approaches

This section shows the surrogates best performing settings for a given sub-model, and compares them. As with the PTO_{same} model, the easiest sub-model to approximate was the ignorant sub-model, followed by the original, and then the altered. For all of the sub-models, the most accurate and precise surrogate was the MLP, followed by random forest, and finally KNN. MLP is the best again, likely due to it's many degrees of freedom, especially useful in this higher dimensional space.

Original sub-model

approach and settings	MSE	variance	fitting time	predict time
knn distance grid k=4	9.30821e+09	0.78	0.570734	9.33861
rfr grid num trees=20	4.56122e+09	0.89	2.88161	0.299709
mlp random num layers=10 num nodes per layer=100	1.28462e+09	0.98	2317.01	23.3873

Table 4.13: Performance of best surrogates of each approach on the original sub-model.

The results are depicted in Table 3.17. 2 out of the 3 surrogates performed better when trained on the grid dataset. However, the best performing surrogate (MLP) was trained on the random dataset. In terms of runtime, the slowest was MLP, but, it performs the best, and the prediction time for all validation samples is still within milliseconds.

Ignorant sub-model

approach and settings	MSE	variance	fitting time	predict time
knn distance random k=10	3.6941e+09	0.86	0.133345	9.56854
rfr random num trees=100	2.05818e+09	0.92	50.2532	1.93987
mlp random num layers=5 num nodes per layer=100	8.98119e+07	1	278.531	1.99808

Table 4.14: Performance of best surrogates of each approach on the ignorant sub-model.

The results are depicted in Table 3.18. In this case, the surrogates trained on the random data perform the best. The MLP has fewer nodes for the ignorant model than the original model, so prediction time is comparable to the other surrogates. Training time is still slower, but recall we are interested in beating the state-of-the-art model prediction time, so training time is less important.

Altered sub-model

approach and settings	MSE	variance	fitting time	predict time
knn distance grid k=4	1.45883e+10	0.76	0.542766	9.31841
rfr random num trees=100	9.56546e+09	0.87	66.1411	2.20084
mlp random num layers=30 num nodes per layer=20	7.33759e+09	0.9	1241.03	6.3476

Table 4.15: Performance of best surrogates of each approach on the altered sub-model.

The results are depicted in Table 3.19. The training times of MLP are far worse than the other surrogates. However, prediction times are comparable across approaches. The two best surrogates for this sub-model were both trained on the random dataset.

4.2 Classification

As in the previous chapter, we consider classifying the original sub-model.

4.2.1 By approach

KNN

k	1	2	3	4	5	7	10	15
original grid	0.94	0.87	1	1	0.99	0.99	1	1
original random	0.88	0.88	0.9	0.9	0.91	0.91	0.92	0.91

k	1	2	3	4	5	7	10	15
original grid	0.94	0.87	1	0.95	0.99	0.99	0.98	1
original random	0.88	0.87	0.9	0.9	0.91	0.91	0.92	0.91

Table 4.16: R^2 of KNN regressors. The upper table uses the distance metric, while the lower one uses the uniform metric.

k	1	2	3	4	5	7	10	15
original grid	0.06	0.13	0	0	0.01	0.01	0	0
original random	0.12	0.12	0.1	0.1	0.09	0.09	0.08	0.09

k	1	2	3	4	5	7	10	15
original grid	0.06	0.13	0	0.05	0.01	0.01	0.02	0
original random	0.12	0.13	0.1	0.1	0.09	0.09	0.08	0.09

Table 4.17: MSE of KNN in ms. The upper table uses the distance metric, while the lower one uses the uniform metric.

k	1	2	3	4	5	7	10	15
original grid	0.788818	0.589703	0.947651	1.04882	0.666512	1.58263	1.02418	0.700085
original random	0.234229	0.356449	0.173214	0.306106	0.294727	0.431518	0.186965	0.269616

k	1	2	3	4	5	7	10	15
original grid	0.589609	0.548593	0.534803	0.531672	0.529308	0.530205	0.532592	0.53386
original random	0.250505	0.187973	0.222601	0.220832	0.34436	0.17531	0.290727	0.342471

Table 4.18: Training time of KNN regressors. The upper table uses the distance metric, while the lower one uses the uniform metric.

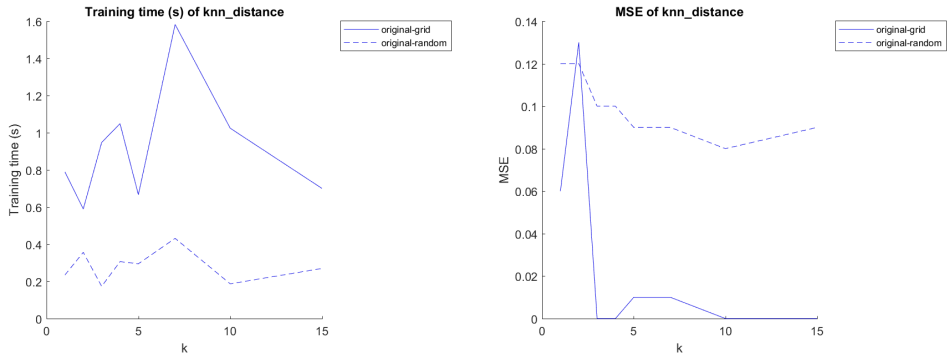
k	1	2	3	4	5	7	10	15
original grid	9.97197	9.28824	10.5266	15.6886	17.1014	18.3018	16.9544	19.7655
original random	9.36913	16.5803	13.9828	16.0773	19.0071	19.5132	23.712	28.8204

k	1	2	3	4	5	7	10	15
original grid	5.87339	5.91036	6.67761	8.87527	10.8543	10.9453	11.2046	11.4808
original random	10.2391	12.1528	14.1633	17.3348	17.7454	19.2503	23.8036	25.7417

Table 4.19: Predicting times of KNN in ms. The upper table uses the distance metric, while the lower one uses the uniform metric.

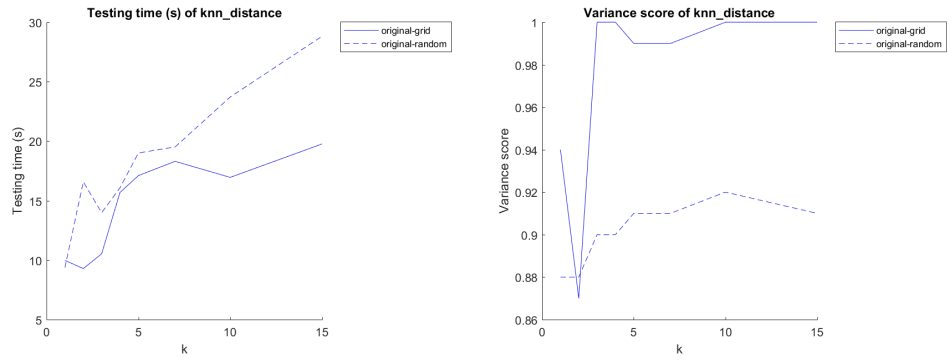
The raw KNN results are in Tables 4.16 to 4.19. These are visualised in Figure 4.4. When comparing training on the grid or random datasets, the grid datasets perform slightly better, in terms of accuracy and precision. When comparing performance of surrogates using the distance metric is also slightly better than the uniform metric. These are the same patterns seen when classifying the PTO_{same} model.

Training time is constant in k, as with regression. Testing time is linear in k, as with regression.



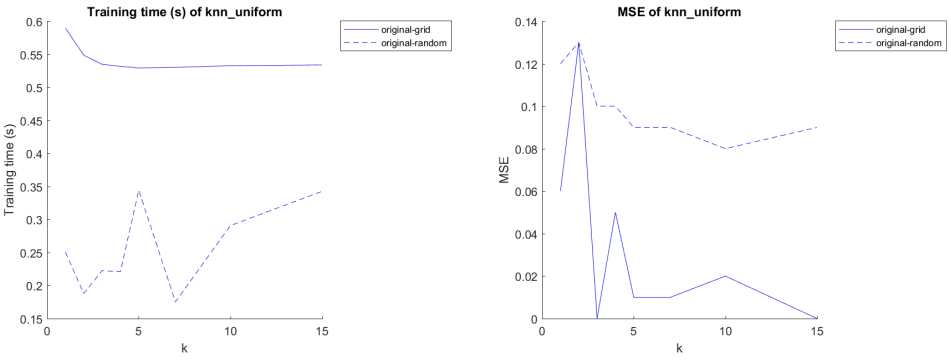
(a)

(b)



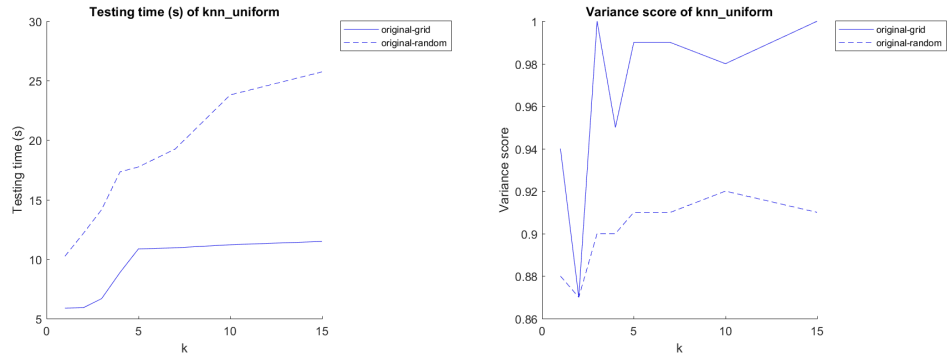
(c)

(d)



(e)

(f)



(g)

(h)

Figure 4.4: Performance of KNN regression surrogates on the PTO_{indiv} model.

There is a slight trend in accuracy and precision; they increase slightly as k increases. Note the accuracy and precision already start quite high, so there is little room for improvement. The accuracy and precision peak at around k=3, 4, and k=10, 15 for the distance metric.

Random Forest

num trees	1	5	10	20	50	100
original grid	0.91	0.93	0.92	0.91	0.91	0.91
original random	0.95	0.97	0.98	0.98	0.98	0.99

Table 4.20: R^2 of RFR.

num trees	1	5	10	20	50	100
original grid	0.09	0.07	0.08	0.09	0.09	0.09
original random	0.05	0.03	0.02	0.02	0.02	0.01

Table 4.21: MSE of RFR.

num trees	1	5	10	20	50	100
original grid	0.098114	0.353793	0.479375	1.02072	1.9987	4.82449
original random	0.248677	1.25062	2.50751	5.61306	13.2223	27.858

Table 4.22: Training times of RFR in ms.

num trees	1	5	10	20	50	100
original grid	0.019132	0.042562	0.08073	0.161043	0.513603	0.846903
original random	0.034289	0.072012	0.13616	0.245265	0.754791	2.02057

Table 4.23: Testing times of RFR in ms.

The raw random forest regressor (RFR) results are in Tables 4.20 to 4.23. These are visualised in Figure 4.5.

Unlike when classifying the PTO_{same} model, the RFR surrogates trained on the random dataset has better accuracy and precision than the grid dataset (for PTO_{same} they were very similar).

As the number of trees increases, the accuracy and precision increase, as with regression. However, the trend is slight, as with as with PTO_{same}, since the precision and accuracy are already high for a small number of trees. Both training time and testing time are linear in the number of trees, as with regression.

MLP

num nodes per layers	5	10	20	50	100
original grid 5	0.73	0.91	0.85	0.88	0.92
original grid 10	0.8	0.93	0.82	0.9	0.93
original grid 20	0.8	0.87	0.9	0.88	0.94
original random 5	0.9	0.98	0.98	0.98	0.99
original random 10	0.9	0.97	0.98	0.98	0.98
original random 20	0.83	0.94	0.98	0.98	0.98

Table 4.24: R^2 of MLP with various numbers of layers (specified in first column).

The raw MLP results are in Tables 4.24 to 4.27. These are visualised in Figure 4.6.

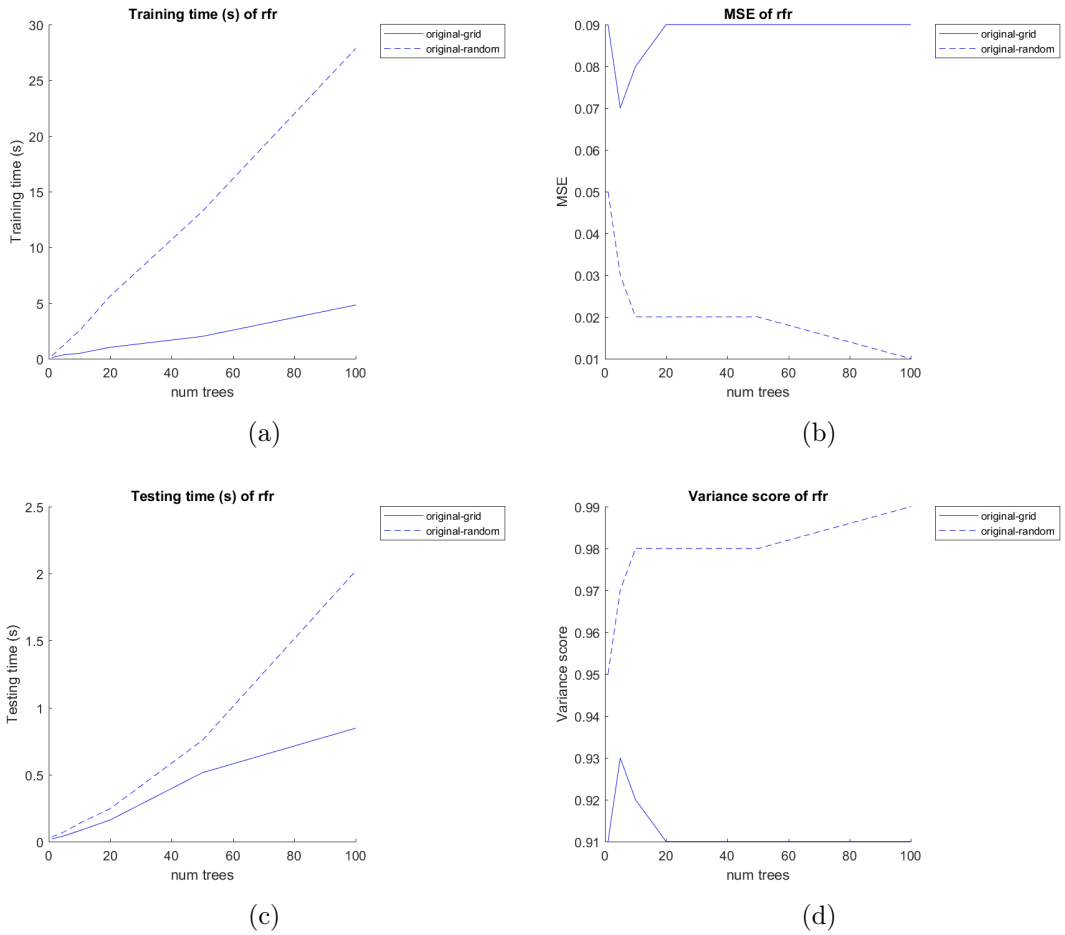


Figure 4.5: Performance of RFR surrogates on the PTO_{indiv} model.

num nodes per layers	5	10	20	50	100
original grid 5	0.27	0.09	0.15	0.12	0.08
original grid 10	0.2	0.07	0.18	0.1	0.07
original grid 20	0.2	0.13	0.1	0.12	0.06
original random 5	0.1	0.02	0.02	0.02	0.01
original random 10	0.1	0.03	0.02	0.02	0.02
original random 20	0.17	0.06	0.02	0.02	0.02

Table 4.25: MSE of MLP with various numbers of layers (specified in first column).

num nodes per layers	5	10	20	50	100
original grid 5	16.8441	16.8702	8.75277	28.2789	35.7798
original grid 10	28.6763	54.7532	21.8737	61.5676	122.391
original grid 20	95.9174	58.3485	31.3819	87.0249	232.356
original random 5	29.4466	89.8225	43.1875	790.757	336.361
original random 10	46.1404	185.624	390.413	683.754	832.16
original random 20	105.67	164.075	153.472	736.071	1019.19

Table 4.26: Training times of MLP with various numbers of layers (specified in first column) in ms.

The MLP surrogates trained on the random dataset perform slightly better in terms of accuracy and precision. There is little difference in the performance of surrogates with different numbers of layers. As the number of nodes in each layer increases, so too does the accuracy and precision.

Both training time and testing time are linear in the number of nodes per layer, as with regression.

num nodes per layers	5	10	20	50	100
original grid 5	0.038614	0.088353	0.193444	0.42103	0.940947
original grid 10	0.079009	0.6943	0.342222	0.789252	1.94248
original grid 20	0.487069	0.302677	0.490789	1.64391	4.37556
original random 5	0.071882	0.33466	0.834585	3.86037	8.11662
original random 10	0.149875	1.0664	2.34074	10.8391	20.244
original random 20	0.238437	1.77272	0.862259	17.7204	13.2056

Table 4.27: Testing times of MLP with various numbers of layers (specified in first column) in ms.

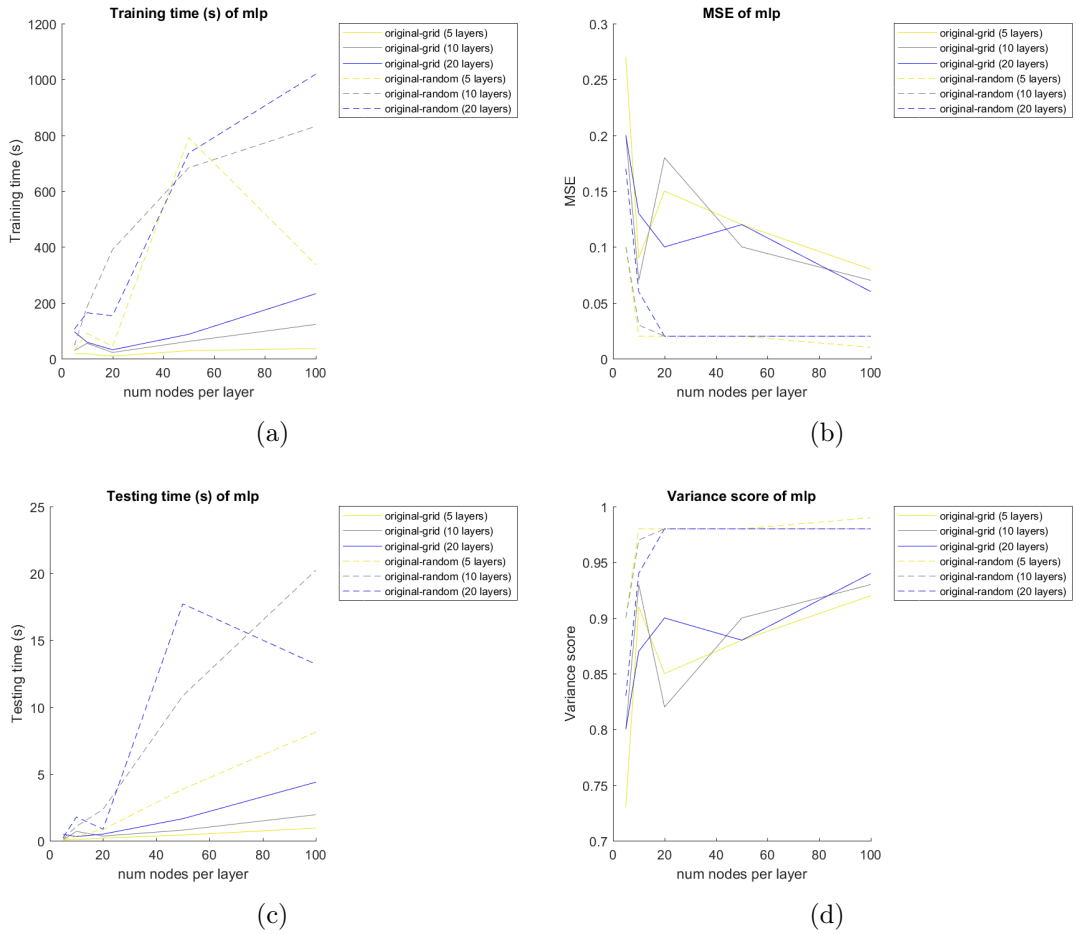


Figure 4.6: Performance of MLP surrogates on the PTO_{indiv} model.

4.2.2 Comparing approaches

As mentioned previously, we only approximate the original sub-model. The results can be found in Table 4.28. The best settings for all surrogates achieve an very high precision and accuracy ($R^2 \geq 0.99$, $MSE \leq 0.01$). KNN achieved perfect precision and accuracy. The worst training time belongs to MLP, though all predicting times are comparable.

approach and settings	MSE	variance	fitting time	predict time
knn uniform grid k=3	0	1	0.534803	6.67761
rfr random num trees=100	0.01	0.99	27.858	2.02057
mlp random num layers=5 num nodes per layer=100	0.01	0.99	336.361	8.11662

Table 4.28: Performance of best surrogates of each approach.

Chapter 5

Future Work

5.1 Future Work

5.1.1 Data collection

We have already collected data in which the PTO parameters, d and k, are varied, and we use the same d and k for every buoy (2 array parameters). We also have data in which there are different d and k for each buoy (8 array parameters). To make the approximations, (and hence the optimal PTO settings based on the approximations) more robust, it would be useful to collect another data set which varies d, k, as well as wave frequency and wave angle (4 array parameters). In addition, we another data set to collect could vary d and k for each buoy, and also wave angle and frequency (10 array parameters).

We wish to use these data sets to see how much longer surrogates training will take, as well as prediction. We can then check if the extra information allows us to achieve greater power absorption.

Another aspect we wish to assess is the effect of the training set size on the performance of a surrogate. To assess this, it would be easier to use a data set with inputs (d and k) that have been randomly and uniformly generated. This is because any randomly chosen subset would also be uniformly distributed.

5.1.2 Performance metrics

Currently, the accuracy performance metric we use is R^2, and the precision metric is MSE. However, we would like to use some other accuracy metrics (see Appendix .1) and precision metrics (see Appendix .2). We want to do this, just to reveal whether the best surrogate under the metrics we have already tests is still the best surrogate for other metrics.

5.1.3 Parameter tuning

As mentioned previously, the training methods for the surrogates themselves have parameters, which can be tuned to ensure the surrogate is performing at its full potential. Currently, we select subsets of values that interest us: either using exponential back-off, or values that we believed would yield good results. When there is more than parameters, we try all values for both in a grid search like approach to find the best. However, there are more sophisticated algorithms that can automatically tune parameters, including IRACE [32] and SMAC [33].

5.1.4 Surrogates

Other training methods we would like to use include genetic programming, in which we would have an underlying mathematical model, with unknown coefficients (similar to mathematical regression problems), and evolve the coefficients that best fit the data. There is also ridge regression, and least squares regression, however, the kernels need to be modified like they were for SVMs.

5.1.5 Assessing optimality of PTO settings resulting from optimising surrogates

Our research assesses the surrogates' abilities to approximate the original model. However, one key performance metric to assess are the surrogates' abilities to yield optimal PTO settings when optimisations are applied to them. This would require running optimisation algorithms on top of both the surrogates and the state-of-the-art models, and comparing the optimal PTO settings obtained. Metrics would be based on the resulting power, and whether the settings are feasible. If the settings are infeasible, we may also consider how much they exceed the feasible threshold by (i.e, how far beyond safe do the tethers elongate). In such a case, data may need to be recollected in a non-binary way, so that instead of denoting feasible or infeasible, it denotes feasible or a measurement of infeasibility.

Chapter 6

Conclusion

This thesis attempted to solve 4 problems: regression and classification of the PTO_{same} model, as well as regression and classification of the PTO_{indiv} model. The regression problems can be broken down by sub-model: original, ignorant, and altered.

For the PTO_{same} model, the best surrogate for the regression of the original sub-model was the random forest with 100 trees, resulting in an R^2 accuracy of 0.99. The best surrogate for the ignorant sub-model was KNN with k of 4, which achieved an R^2 accuracy of 1 (perfect). This occurred when trained on the dataset collected using a grid search. Hence, the best value of k was 4, since any new samples to predict will always fall perfectly between a square of 4 training points. The best surrogate for the altered model was the MLP with 20 layers and 100 nodes in each. This achieved an R^2 accuracy of 0.97. When classifying the PTO_{same} model, all surrogates achieved perfect accuracy and precision

For regression of the PTO_{indiv} model, the MLP achieved the best accuracy and precision for all 3 sub-models. The best hyperparameters for the original sub-model were 10 layers, and 10 nodes per layer, which achieved an R^2 accuracy of 0.98. For the ignorant sub-model, 5 layers with 100 nodes in each achieved perfect accuracy. For the altered sub-model, 30 layers with 20 nodes in each achieved an R^2 accuracy of 0.9. When classifying the PTO_{indiv} model, all surrogates achieve R^2 of at least 0.99.

All PTO_{same} surrogates were able to predict thousands of samples in under a millisecond, and the slowest PTO_{indiv} surrogate was still able to predict hundreds of thousands of samples in under a second. Therefore, the surrogate prediction times are far smaller than those of the state-of-the-art model. Now that we've shown we can train fast and reasonably accurate surrogates, we can test the surrogates for optimisation purposes. This would involve running optimisations on both the surrogates and the state-of-the-art model, and comparing them. This would be a substantial step forward in the optimisation of PTO settings of Wave Energy Converters.

Bibliography

[1] Drew B, Plummer AR, Sahinkaya MN. *A review of wave energy converter technology.* Proceedings of the Institution of Mechanical Engineers, Part A: Journal of Power and Energy, 223(8). 2009; 887–902.

[2] Esteban M, Miguel E, David L. *Current developments and future prospects of offshore wind and ocean energy.* Applied Energy. 2012;90: 128–136.

[3] Lagoun MS, Benalia A, Benbouzid MEH. *Ocean wave converters: State of the art and current status.* 2010 IEEE International Energy Conference. 2010. doi:10.1109/energycon.2010.5771758.

[4] Mann LD, Burns A, and Ottaviano M. *CETO, a carbon free wave power energy provider of the future.* In Proceedings of the 7th European Wave and Tidal Energy Conference. 2007.

[5] Mann LD. *Application of Ocean Observations & Analysis: The CETO Wave Energy Project.* Operational Oceanography in the 21st Century. 2011. pp. 721–729.

[6] Scruggs JT, Lattanzio SM, Taflanidis AA, Cassidy IL. *Optimal causal control of a wave energy converter in a random sea.* Applied Ocean Research. 2013;42: 1–15.

[7] Falnes J. *Ocean Waves and Oscillating Systems: Linear Interactions Including Wave-Energy Extraction.* Cambridge University Press. 2002.

[8] Cazzalato BS, Ding B, Neumann F, Sergiienko N, Shekh S, Wagner M, Wu J. *Fast and effective optimisation of arrays of submerged wave energy converters.* Accepted by GECCO. 2016.

[9] Ding B, Cazzolato BS, Arjomandi M, Hardy P. *Sea-state Based Maximum Power Point Tracking Damping Control of a Fully Submerged Oscillating Buoy.* Under review by Journal of Ocean Engineering. 2016.

[10] Giorgi S, Davidson J, Ringwood JV. *Identification of Wave Energy Device Models From Numerical Wave Tank Data—Part 2: Data-Based Model Determination.* IEEE Transactions on Sustainable Energy, VOL. 7, NO. 3, JULY 2016.

[11] Yaochu J. *Surrogate-assisted evolutionary computation: Recent advances and future challenges.* Swarm and Evolutionary Computation Volume 1, Issue 2, Pages 61–70. June 2011.

[12] Banos R, Manzano-Agugliaro F, Montoya FG, Gil C, Alcayde A, Gómez J. *Optimization methods applied to renewable and sustainable energy: A review.* Renewable and Sustainable Energy Reviews, 15(4), 1753-1766. 2011.

[13] Ringwood J, Butler S. *Optimisation of a wave energy converter*. In IFAC Conference on Control Applications in Marine Systems-CAMS (pp. 7-9). Maynooth University. 2004.

[14] Nolan GA, Ringwood JV, Leithead W, Butler S. *Optimal damping profiles for a heaving buoy wave energy converter*. In Proceedings of the Fifteenth International Offshore and Polar Engineering Conference (ISOPE) (pp. 477-485). 2005.

[15] Nunes G, Valério D, Beirao P, Da Costa JS. *Modelling and control of a wave energy converter*. Renewable Energy, 36(7), 1913-1921. 2011.

[16] J.T. Scruggs, S.M. Lattanzio, A.A. Taflanidis, I.L. Cassidy *Optimal causal control of a wave energy converter in a random sea*. Applied Ocean Research 42 (2013) 1–15

[17] Hals J, Falnes J, Moan T. *A comparison of selected strategies for adaptive control of wave energy converters*. Journal of Offshore Mechanics and Arctic Engineering, 133(3), 031101. 2011.

[18] Korde UA, Ertekin RC. *Wave energy conversion by controlled floating and submerged cylindrical buoys*. Journal of Ocean Engineering and Marine Energy, 1-18. 2015.

[19] McCabe AP, Aggidis GA, Widden MB. *Optimizing the shape of a surge-and-pitch wave energy collector using a genetic algorithm*. Renewable Energy, 35(12), 2767-2775. (2010).

[20] Borgarino B, Babarit A, Ferrant P. *Impact of wave interactions effects on energy absorption in large arrays of wave energy converters*. Ocean Engineering, 41, 79-88. 2012.

[21] Bacelli G, Ringwood J. *Constrained control of arrays of wave energy devices*. International Journal of Marine Energy, 3(4), 53-69. 2013.

[22] Fitzgerald C, Thomas G. *A preliminary study on the optimal formation of an array of wave power devices*. In Proceedings of the 7th European Wave and Tidal Energy Conference, Porto, Portugal. 2007.

[23] Lettenmaier T, von Jouanne A, Brekken T. *A new maximum power point tracking algorithm for ocean wave energy converters*. International Journal of Marine Energy Volume 17, Pages 40–55. April 2017.

[24] Ghasemi A, Anbarsooz M, Malvandi A, Ghasemi A, Hedayati F. *A nonlinear computational modeling of wave energy converters: A tethered point absorber and a bottom-hinged flap device* Renewable Energy Volume 103, Pages 774–785. April 2017.

[25] Zou S, Abdelkhalik O, Robinett R, Bacelli G, Wilson D. *Optimal control of wave energy converters* Renewable Energy Volume 103, Pages 217–225. April 2017.

[26] O'Sullivan ACM, Lightbody G. *Co-design of a wave energy converter using constrained predictive control* Renewable Energy Volume 102, Part A, Pages 142–156. March 2017.

[27] Amarkarthik A, Sivakumar K. *Investigation on modeling of non-buoyant body typed point absorbing wave energy converter using Adaptive Network-based Fuzzy Inference System* International Journal of Marine Energy, Volume 13, Pages 157–168. April 2016.

[28] González-Gorbeña E, Qassim RY, Rosman PCC. *Optimisation of hydrokinetic turbine array layouts via surrogate modelling* Renewable Energy, Volume 93, Pages 45–57. August 2016.

[29] Sarkar D, Contal E, Vayatis N, Dias F. *Prediction and optimization of wave energy converter arrays using a machine learning approach* Renewable Energy, Volume 97, Pages 504–517. November 2016

[30] Halder P, Rhee SH, Samad A. *Numerical optimization of Wells turbine for wave energy extraction* International Journal of Naval Architecture and Ocean Engineering Volume 9, Issue 1, Pages 11–24. January 2017.

[31] Pedregosa, F. and Varoquaux, G. and Gramfort, A. and Michel, V. and Thirion, B. and Grisel, O. and Blondel, M. and Prettenhofer, P. and Weiss, R. and Dubourg, V. and Vanderplas, J. and Passos, A. and Cournapeau, D. and Brucher, M. and Perrot, M. and Duchesnay, E. *Scikit-learn: Machine Learning in Python*, Journal of Machine Learning Research, vol. 12, Pages 2825–2830. 2011

[32] López-Ibáñez M, Cáceres LP, Dubois-Lacoste J, Stützle T, Birattari M. *The irace package: Iterated racing for automatic algorithm configuration* Operations Research Perspectives, vol. 3. pp. 43-58. 2016.

[33] Hutter F, Hoos H, Leyton-Brown K. *Sequential model-based optimization for general algorithm configuration* Proc. of LION'11, pp 507-523. 2011.

Appendices

.1 Calculating accuracy

The error, e_i (Equation 1) between the predictor's output (f_i) and original model's output (y_i), given the input of the i$^{\text{th}}$ datum in the set.

$$e_i = y_i - f_i \tag{1}$$

In all of the following, e_i is given by Equation 1.

$$ME = \frac{1}{n}\sum_{i=1}^{n} e_i \tag{2}$$

$$R^2 = 1 - \frac{\sum_{i=1}^{n} e_i^2}{\sum_{i=1}^{n}(\bar{y} - y_i)^2} \tag{3}$$

Where:

y_i: is the output of the original model being approximated.

$\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$

.2 Calculating precision

In all of the following, e_i is given by Equation 1.

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |e_i| \tag{4}$$

$$MSE = \frac{1}{n}\sum_{i=1}^{n} e_i^2 \tag{5}$$

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n} e_i^2} \tag{6}$$