# Comparison between three regression algorithms by applying them on two different datasets

Computer Aided Solution Processing, Spring 2021

Mohamed Rawash, 210ADB004

# 1. Introduction

To describe the relationships between a set of independent variables and a dependent variable, regression analysis is used in which models can be developed to predict the value of the dependent variable (the output) of new inputs based on a previous dataset used to train the model [5]. Two different datasets will be used in this project, and multiple regression techniques will be applied to them to compare the efficiency (speed) and the effectiveness (predictive performance) of these regression algorithms. The first dataset is about Combined Cycle Power Plant in which the features consist of hourly average ambient variables Temperature and other factors to predict the net hourly electrical energy output of the plant. While the second dataset is about prediction of residuary resistance of sailing yachts based on the basic hull dimensions. The three regression algorithms that will be applied on these two datasets are multiple linear regression, random forest regression, and K-nearest neighbors.

# 2. Theoretical Explanation

This section provides theoretical explanation of multiple linear regression, random forest regression, and K-nearest neighbors that will be applied on the two datasets. Also, it provides an explanation about the k-fold cross validation method and the testing criteria of the generated regression models.

## 2.1. Multiple Linear Regression

Multiple linear regression is a method of fitting a linear equation to observed data to model the relationship between two or more explanatory variables (inputs) and a response variable (output). Any value has a contribution by an effect factor to correspond to a value in the dependent variable [6]. Formally, the model for multiple linear regression is:

, (1)

where is the output or the response variable; is a constant y-intercept and is the effect factor for each input where and is the number of inputs; is the input where and is the number of inputs; is the model deviations.

As the input (explanatory variables) and the output (response variable) are given in the provided datasets, the aim is to estimate the effect factors . These factors can be estimated by many ways, one of them is the least square method and the solution can be represented by the following formula:

, (2)

where is a matrix of the output (response variable); is a matrix of the input (explanatory variables); is the matrix transpose of the input (explanatory variables) in the training dataset.

## 2.2. Random Forest regression

Random Forest Regression is a supervised learning algorithm that uses ensemble learning method which is a technique that combines predictions from multiple machine learning algorithms to produce a more accurate prediction than a single model [3]. As shown in figure 1, random forest regression is made up of independent decision trees each with random k data points from the training set that run in parallel, and the result is the average across all the predicted y values from all trees. Since it considers all the results generated by each decision tree, this approach overcomes the disadvantage of the decision trees' high variance.

Diagram

Description automatically generated

**Fig. 1.** Random Forest Regression Algorithm

## 2.3. K-Nearest Neighbors regression

The KNN regression algorithm predicts the values of new data points based on feature similarity. This implies that a value is given to the new point based on how similar it is to the points in the training set. So, it predicts the output based on the k nearest neighbors and returns their average y as the result. This algorithm is called "lazy" as the training stage consists of just storing the record. While in the prediction stage, the Euclidian distance is used to determine the k nearest neighbors to the query point so that the predicted ŷ is the average y of the k nearest neighbors that have the least Euclidean distance [2]. The Euclidean distance is calculated using the following formula:

, (3)

where is the query point for which the nearest neighbors are found; is the training record to which the distance is calculated; is the feature index; is the total number of features.

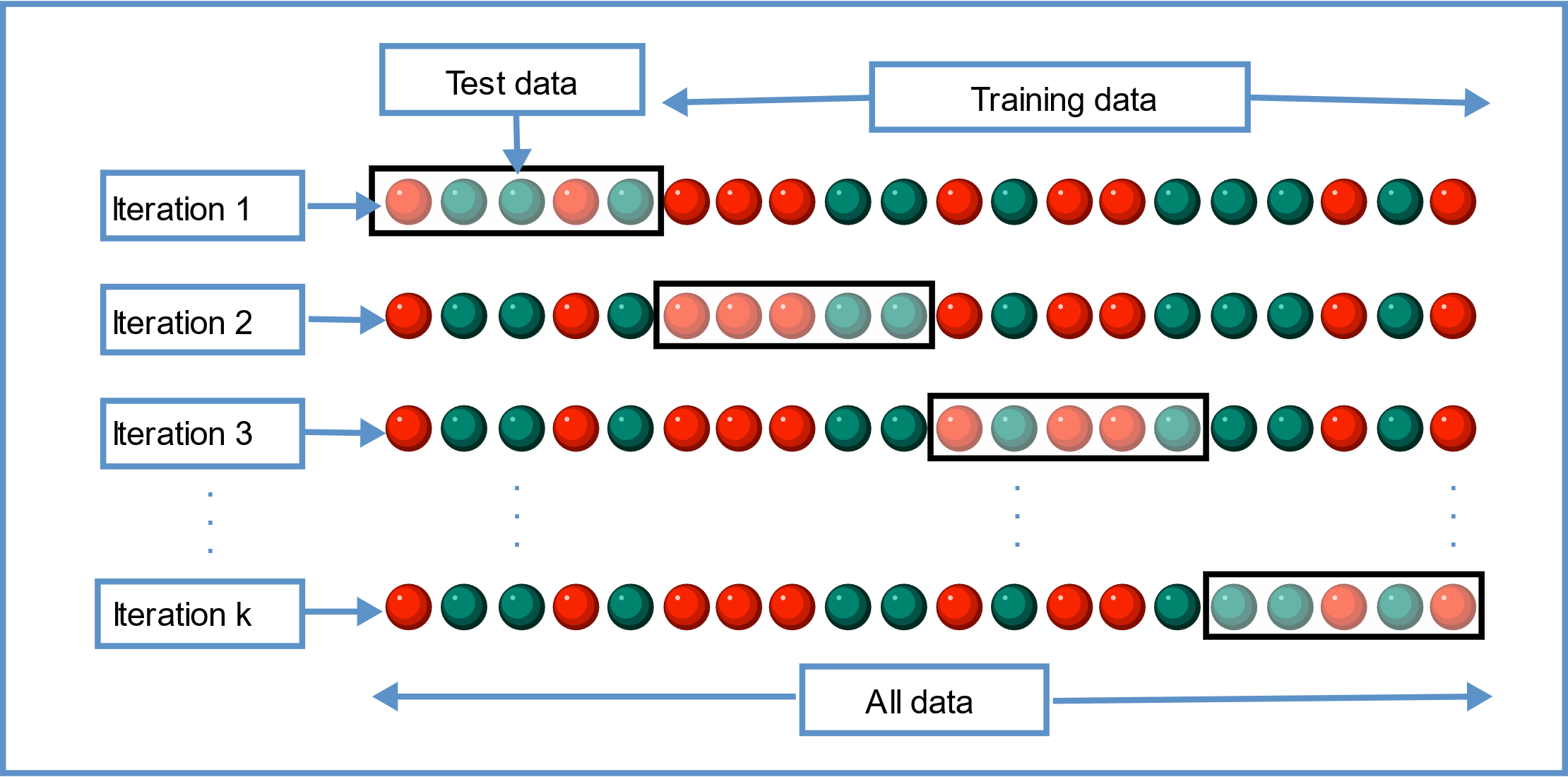
After using the Euclidean distance to find the k nearest neighbors, the predicted output is calculated as the average of the output of the k nearest neighbors without weights (uniform) or with weight for each neighbor as in the following formula:

, (4)

where is the weight: without weights (uniform): ,; with weights: will equal the inverted distance, , .

## 2.3. K-Fold Cross Validation

All records of the full dataset are randomly reordered and divided in subsets (folds) of equal size. iterations are done, in each iteration subset is used as a validation set to test the generated model by calculating its accuracy and the other subsets together are used as a training set to generate the model. The accuracy of each generated model is calculated by calculating the average of the k-scores from each iteration. Figure 2 illustrates the divisions of the dataset into test data and training data for k iterations [4].



**Fig. 2.** K-fold cross validation

## 2.4. Regression model testing criteria

There are many testing criteria to evaluate the generated models, this paper will use two ways. The method will be used in the cross validation to calculate the accuracy of each generated model in each iteration, and the Mean Squared Error to calculate the accuracy of the final selected model. The following equations illustrates the way to calculate the parameters of the two methods [1]:

, (5)

Where TSS is the total sum of squares (total variation):

, (6)

And RSS is the residual sum of squares (model squared error):

, (7)

, (8)

where is the output of the record; is the predicted output of the record; is the mean of the outputs; n is the number of the tested records.

# 3. Overview of the used datasets

## 3.1. First Dataset (Combined Cycle Power Plant)

According to the information provided from UCI Machine learning repository, 9568 records with zero null values collected from a Combined Cycle Power Plant over 6 years (2006-2011), when the power plant was set to work with full load [7]. The dataset size: 47840. As clarified in table 1, number of features are 4 hourly average ambient variables. The output variable is the net hourly electrical energy output (PE) of the plant in the range 420.26-495.76 MW.

**Table 1**

List and explanation of first dataset features

|  |  |  |
| --- | --- | --- |
| **Feature Nr.** | **Feature Symbol** | **Feature Explanation** |
| 1 | AT | Temperature in the range 1.81°C and 37.11°C |
| 2 | AP | Ambient Pressure in the range 992.89-1033.30 milibar |
| 3 | RH | Relative Humidity in the range 25.56% to 100.16% |
| 4 | V | Exhaust Vacuum in the range 25.36-81.56 cm Hg |

## 3.2. Second Dataset (Yacht Hydrodynamics)

According to the information provided from UCI Machine learning repository, 308 records with zero null values performed at the Delft Ship Hydromechanics Laboratory to predict the residuary resistance of sailing yachts at the initial design stage for evaluating the shipment's performance and for estimating the required propulsive power [8]. The dataset size: 1848. As clarified in table 2, number of features are 6 adimensional hull geometry coefficients and the Froude number. The output variable is the residuary resistance per unit weight of displacement.

**Table 2**

List and explanation of second dataset features

|  |  |  |
| --- | --- | --- |
| **Feature Nr.** | **Feature Symbol** | **Feature Explanation** |
| 1 | X1 | Longitudinal position of the center of buoyancy |
| 2 | X2 | Prismatic coefficient |
| 3 | X3 | Length-displacement ratio |
| 4 | X4 | Beam-draught ratio |
| 5 | X5 | Length-beam ratio |
| 6 | X6 | Froude number |

# 4. Experiments

To prepare the datasets to compare the efficiency (speed) and the effectiveness (predictive performance) of the algorithms, each dataset is split into two parts: 80% for training and 20% for testing. Then, a cross validation model with 10 folds is prepared. Now, by selecting the regression algorithm, the cross-validation scores can be obtained. To measure the efficiency (speed) of each algorithm, the Implementation Time will be measured which is the whole time taken for all calculations from setting the best parameter options to selecting the final model, and the Fitting Time which is the time taken by each model to produce the cross-validation scores. To measure the effectiveness (predictive performance) of the algorithms, the average of cross-validation scores is calculated. Table 3 lists all the used libraries to perform the experiments and their purposes [9] [10].

**Table 3**

All used libraries and their purposes

|  |  |
| --- | --- |
| **Library and Functions** | **Purpose** |
| pandas | To read the dataframe |
| matplotlib.pyplot | To plot |
| time | To measure the time |
| math | To call mathematical functions |
| tabulate | To draw tables |
| train\_test\_split | To split the dataset into training set and testing set |
| cross\_val\_score & KFold | To run the k-cross validation |
| LinearRegression | For linear regression |
| RandomForestRegressor | For random forest regression |
| neighbors | For knn regression |
| metrics | For evaluation |

## 4.1. Experiments with the first dataset

In each algorithm, several possible values of hyperparameters are changed to evaluate each model to find the best final model.

## 4.1.1. Linear Regression Algorithm

In linear regression function, there is only one parameter that may affect the performance which is the normalize option that could take “True” or “False”. However, after trying the two options, the performance and speed was the same. So, it was left as default which is “False”. Table 4 shows the results obtained from the final linear regression model.

**Table 4**

Linear Regression first dataset final model results

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Measured Parameter** | **Result** |
| Efficiency (Speed) | Implementation Time | 0.06470149999999997 |
| Fitting Time | 0.06460709999999992 |
| Effectiveness (Predictive performance) | Performance | 0.928134965279164 |

## 4.1.1.1 Analysis

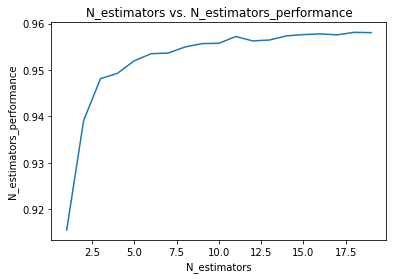
As expected from linear regression algorithm as it is simple, the implementation time and fitting time are approximately the same as there are not many hyperparameters to change and the implementation time consists of the fitting time. Also, the performance is not too high for the same reason of the simplicity of the linear regression algorithm.

## 4.1.2. Random Forest Regression Algorithm

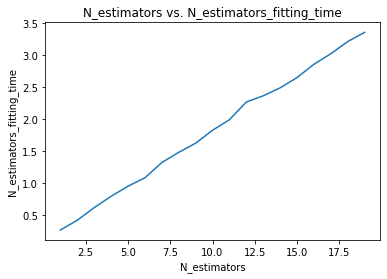
In Random Forest function, there are many hyperparameters options. However, only three of these parameters have significant effect on the performance and speed. Thus, they were selected to change their options and observe their effect on the performance and speed. To be able test each parameter, one parameter is changed with fixing all the other parameters to be default.

## 4.1.2.1 N-estimators: number of decision trees in the forest

By increasing the n-estimators, the time increases linearly which results in a large number in the implementation and the fitting time. Therefore, an efficient number of trees could be chosen by considering the speed and the performance of the algorithm together. A for loop is implemented to try different values of n-estimators from 1 to 20 and store the speed and performance parameters of each number of estimators in an array to be able to create a graph and compare between the estimators as shown in figure 3 and 4.



**Fig. 3.** N-estimators vs. N-estimators Performance



**Fig. 4.** N-estimators vs. N-estimators Fitting Time

Figure 3 illustrates that the performance is increasing with increasing the n-estimators until it reaches a specific n-estimators and then the performance does not significantly change. Figure 4 illustrates that time is increasing linearly with increasing the n-estimators. So, an efficient number of trees could be 8 where the performance is approaching the highest value and the fitting time is reasonable. In the following experiments to choose the other parameters, n-estimators of 8 will be chosen and fixed.

## 4.1.2.2 Criterion: quality of splitting trees error

This parameter selects the function to measure the quality of a split for the trees. It has two options: the mean absolute error (MAE) and the mean squared error (MSE). Table 5 lists the difference between the two options regarding the performance and speed.

**Table 5**

Splitting Quality Criterion Results

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter** | **Measured Parameter** | **MAE Result** | **MSE Result** |
| Efficiency (Speed) | Fitting Time | 27.283622200000003 | 1.3292043000000007 |
| Effectiveness (Predictive performance) | Performance | 0.9537716228159686 | 0.9544909669500921 |

The obtained results clarifies that this parameter has high effect on the speed rather than the performance as the MAE criterion takes more than 20 times the time taken by MSE criterion to reach to highest possible performance. This could be happened due to the nature of the dataset and the number of the datapoints as the MSE is better than the MAE in highlighting the differentiation of errors in each split as MSE squares the error, so the difference appears well. So, it took less time for MSE to find the best model with less splits than MAE criterion. In the following experiment to choose between parameter options, n-estimators of 8 and MSE criterion will be chosen and fixed.

## 4.1.2.3 Max\_features

This parameter selects the number of features to consider when looking for the best split. It has three options: auto which is the default and sets the number of considered features to be all the n-features, sqrt which sets the number of considered features to be sqrt(n-features), and log2 which sets the number of considered features to be log2(n-features). Table 6 lists the difference between the three options regarding the performance and speed.

**Table 6**

Number of Max Considered Features Results

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Parameter** | **Measured Parameter** | **Auto Result** | **Sqrt Result** | **Log2 Result** |
| Efficiency (Speed) | Fitting Time | 1.317477199999999 | 0.8319592999999941 | 0.8274352000000107 |
| Effectiveness (Predictive performance) | Performance | 0.9542193770479453 | 0.9565688657942127 | 0.9561693285896832 |

The above results show that the Sqrt and Log2 options have the highest speed as they do not consider all the features in splitting, and the highest performance too. By comparing the Sqrt and Log2 options, the sqrt will be the best option as it has the highest performance with a reasonable fitting time.

## 4.1.2.4 Final Model Results and Analysis

After selecting the above-mentioned parameters best options: N-estimators = 8, Criterion = MSE, and Max-features = sqrt, the Random Forest Regression algorithm had the results listed in Table 7.

**Table 7**

Random Forest Regression first dataset final model results

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Measured Parameter** | **Result** |
| Efficiency (Speed) | Implementation Time | 67.46766729999999 |
| Fitting Time | 0.8408417999999926 |
| Effectiveness (Predictive performance) | Performance | 0.9561266976694925 |

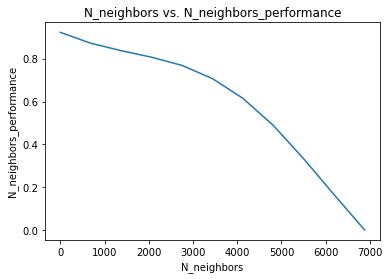
From the final model results, it can be noticed that the performance of Random Forest regression is reasonable, and the implementation time, taken to decide on the best parameters’ options and evaluating each model using the 10-fold Cross-Validation, is long because of the many parameters and options.

## 4.1.3. K-Nearest Neighbors Regression Algorithm

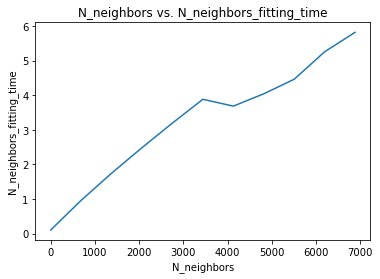
In KNN regression function, there are only two hyperparameters options that may influence the performance and speed. To be able test each parameter, one parameter is changed with fixing all the other parameters to be default.

## 4.1.3.1 N-neighbors

This parameter sets the number of neighbors to use by default for k-neighbors queries. To save time, a for loop is implemented to try different values of n-neighbors from 1 to the maximum possible number of neighbors which is 90% of the training set (as 90% of the data is for training and 10% is for the cross-validation) equals to 6888 and with a step of 688 and store the speed and performance parameters of each number of n-neighbors in an array to be able to create a graph and compare between the estimators as shown in figure 5 and 6.

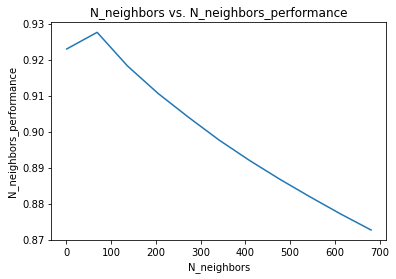


**Fig. 5.** N-neighbors max vs. N-neighbors Performance

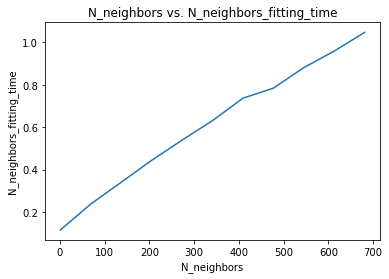


**Fig. 6.** N-neighbors max vs. N-neighbors Fitting Time

As can be observed from figure 5, the performance decreases when the k increases. So, the same loop should be repeated by taking the first part (from 1 to 688) and decreasing the step to be 68 to have better results.



**Fig. 7.** N-neighbors first segment vs. N-neighbors Performance



**Fig. 8.** N-neighbors first segment vs. N-neighbors Fitting Time

Figure 7 and 8 shows that the best option for n-neighbors parameter is 69 selected by getting the max performance from the array of performances and using its index to get the number stored in the same index in the array of number of neighbors. So, the performance for this best K was 0.9275905750944343 and the fitting time for this best K was 0.23851430000000562 which is reasonable as the time is rising linearly as shown in figure 8. In the following experiment, the K will be the best K = 69.

## 4.1.3.2 weights

This parameter determines whether to give a weight for each neighbor used in prediction or not based on the Euclidian distance. The two options are uniform (no weights and all neighbors are equal in weights) and distance where each neighbor has a weight in the prediction process so that the nearer points will have more effect on the predicted output. Table 8 lists the comparison between the two options regarding the speed and the performance.

**Table 8**

Weighting Criterion Results

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter** | **Measured Parameter** | **“Uniform” Result** | **“Distance” Result** |
| Efficiency (Speed) | Fitting Time | 0.24329290000000015 | 0.24806139999999743 |
| Effectiveness (Predictive performance) | Performance | 0.9275905750944343 | 0.9359595190357208 |

The obtained results clarifies that this parameter has small effect on the speed, but a considerable effect on the performance. As noticed from the table, choosing weighting criterion as distance is the best option.

## 4.1.3.3 Final Model Results and Analysis

After selecting the above-mentioned parameters best options: N-neighbors = 69 and weighting criterion = distance, the K-Nearest Neighbors Regression algorithm had the results listed in Table 9.

**Table 9**

K-Nearest Neighbors Regression first dataset final model results

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Measured Parameter** | **Result** |
| Efficiency (Speed) | Implementation Time | 43.810478200000006 |
| Fitting Time | 0.24882030000000555 |
| Effectiveness (Predictive performance) | Performance | 0.9359595190357208 |

From the final model results, it can be noticed that the performance of K-Nearest Neighbors is reasonable, and the implementation time, taken to decide on the best parameters’ options and evaluating each model using the 10-fold Cross-Validation, is a bit long because of the many parameters and options.

## 4.1.4 Selecting Best Model Among All Algorithms

In this subsection, the final three models generated from applying the three different algorithms are compared according to their speed and performance. Table 10 shows the comparison between the final models.

**Table 10**

Final Models Comparison

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Parameter** | **Measured Parameter** | **Linear Regression** | **Random Forest Regression** | **K-NN Regression** |
| Efficiency (Speed) | Implementation Time | 0.0647015 | 67.4677 | 43.8105 |
| Fitting Time | 0.0646071 | 0.840842 | 0.24882 |
| Effectiveness (Predictive performance) | Performance | 0.928135 | 0.956127 | 0.93596 |

By observing the comparison in table 10, it is found that the Linear Regression algorithm has the best speed as it has the least implementation time, and the linear regression final model has the best speed as its fitting time is the minimum. The Random Forest Regression algorithm has the best performance but a high implementation time and therefore slow speed. The K-NN regression algorithm has a good performance and a bit high implementation time. So, if we are interested in the best performance, Random Forest algorithm is the best option. If we are interested in the fastest speed, Linear regression algorithm is the best option. If we need a good performance with a moderate speed, K-NN regression is the best option.

## 4.1.4 Testing the Best Model

Using the whole training set to fit the final models and the test set which is 20% of the dataset, the best models from each algorithm will be tested by measuring their accuracy by comparing the predicted output by each final model to the original output in the test set. This comparison is made by using the mean squared error (MSE) criterion. Table 11 shows the accuracy of each final model.

**Table 11**

Final Models Testing Accuracy

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter** | **Linear Regression for best speed** | **Random Forest Regression for best performance** | **K-NN Regression for good performance and moderate time** |
| MSE | 20.27370599968745 | 11.213581298164849 | 17.142891341908364 |

## 4.2. Experiments with the second dataset

In each algorithm, several possible values of hyperparameters are changed to evaluate each model to find the best final model. As the columns' names are numbers and the last row has no records, it could be expected that the dataset was shifted. So, we need to fill the null row with the records of the first row.

## 4.2.1. Linear Regression Algorithm

As in the first dataset linear regression function has no parameter that may affect the performance or speed. Table 12 shows the results obtained from the final linear regression model.

**Table 12**

Linear Regression second dataset final model results

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Measured Parameter** | **Result** |
| Efficiency (Speed) | Implementation Time | 0.049430899999999944 |
| Fitting Time | 0.049351399999999934 |
| Effectiveness (Predictive performance) | Performance | 0.6252617150369819 |

## 4.2.1.1 Analysis

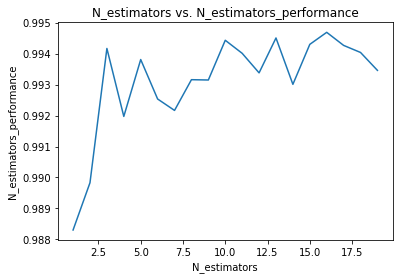
As the first dataset, linear regression is a simple algorithm, the implementation and fitting time are approximately the same as there are no hyperparameters to change. The performance is low because the model is simple and there are a few datapoints in this dataset.

## 4.2.2. Random Forest Regression Algorithm

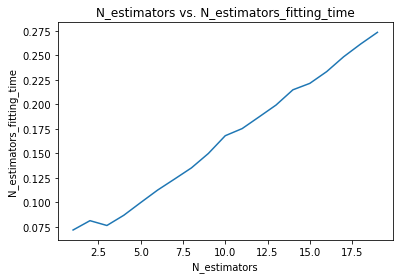
Only three hyperparameters have significant effect on the performance and speed. Thus, they were selected to change their options and observe their effect. To be able test each parameter, one parameter is changed with fixing all the other parameters to be default.

## 4.2.2.1 N-estimators: number of decision trees in the forest

As in the first dataset, a for loop is implemented to try different values of n-estimators from 1 to 20 and compare between them as shown in figure 9 and 10.



**Fig. 9.** N-estimators vs. N-estimators Performance dataset2



**Fig. 10.** N-estimators vs. N-estimators Fitting Time dataset2

Figure 9 illustrates that the performance does not vary significantly with increasing the n-estimators. Figure 10 illustrates that time is increasing linearly with increasing the n-estimators. So, an efficient number of trees could be 5 where the performance is approaching the highest value and the fitting time is reasonable. In the following experiments to choose the other parameters, n-estimators of 5 will be chosen and fixed.

## 4.2.2.2 Criterion: quality of splitting trees error

This parameter selects the function to measure the quality of a split for the trees. It has two options: the mean absolute error (MAE) and the mean squared error (MSE). Table 13 lists the difference between the two options regarding the performance and speed.

**Table 13**

Splitting Quality Criterion Dataset2 Results

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter** | **Measured Parameter** | **MAE Result** | **MSE Result** |
| Efficiency (Speed) | Fitting Time | 0.17978570000000005 | 0.09959380000000007 |
| Effectiveness (Predictive performance) | Performance | 0.9934542592294815 | 0.9922939510376498 |

The obtained results clarifies that this parameter has high effect on the speed rather than the performance as the MAE criterion takes 2 times the time taken by MSE criterion to reach to highest possible performance. This could be happened due to the nature of the dataset and the number of the datapoints as the MSE is better than the MAE in highlighting the differentiation of errors in each split as MSE squares the error, so the difference appears well. So, it took less time for MSE to find the best model with less splits than MAE criterion. In the following experiment to choose between parameter options, n-estimators of 5 and MSE criterion will be chosen and fixed.

## 4.2.2.3 Max\_features

This parameter selects the number of features to consider when looking for the best split. It has three options: auto which is the default and sets the number of considered features to be all the n-features, sqrt which sets the number of considered features to be sqrt(n-features), and log2 which sets the number of considered features to be log2(n-features). Table 14 lists the difference between the three options regarding the performance and speed.

**Table 14**

Number of Max Considered Features Dataset2 Results

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Parameter** | **Measured Parameter** | **Auto Result** | **Sqrt Result** | **Log2 Result** |
| Efficiency (Speed) | Fitting Time | 0.10507160000000049 | 0.1042879000000001 | 0.0933900999999997 |
| Effectiveness (Predictive performance) | Performance | 0.993427621281293 | 0.9449279213000432 | 0.935839313373636 |

The above results show that the auto option has the highest performance as it considers all the 6 features in splitting.

## 4.2.2.4 Final Model Results and Analysis

After selecting the above-mentioned parameters best options: N-estimators = 5, Criterion = MSE, and Max-features = auto, the Random Forest Regression algorithm had the results listed in Table 15.

**Table 15**

Random Forest Regression first dataset final model dataset2 results

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Measured Parameter** | **Result** |
| Efficiency (Speed) | Implementation Time | 4.2592351 |
| Fitting Time | 0.1038553000000002 |
| Effectiveness (Predictive performance) | Performance | 0.9929998832251329 |

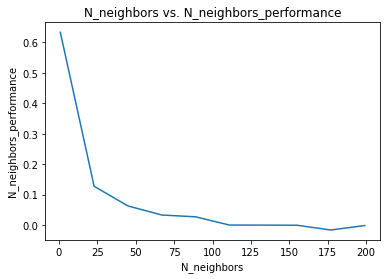
From the final model results, it can be noticed that the performance of Random Forest regression is high, and the implementation time, taken to decide on the best parameters’ options and evaluating each model using the 10-fold Cross-Validation, is reasonable.

## 4.2.3. K-Nearest Neighbors Regression Algorithm

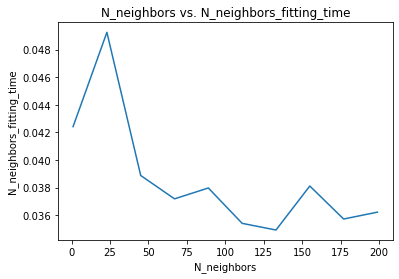
In KNN regression function, there are only two hyperparameters options that may influence the performance and speed. To be able test each parameter, one parameter is changed with fixing all the other parameters to be default.

## 4.2.3.1 N-neighbors

This parameter sets the number of neighbors to use by default for k-neighbors queries. To save time, a for loop is implemented to try different values of n-neighbors from 1 to the maximum possible number of neighbors which is 90% of the training set (as 90% of the data is for training and 10% is for the cross-validation) equals to 221 and with a step of 22 and store the speed and performance parameters of each number of n-neighbors in an array to be able to create a graph and compare between the estimators as shown in figure 11 and 12.

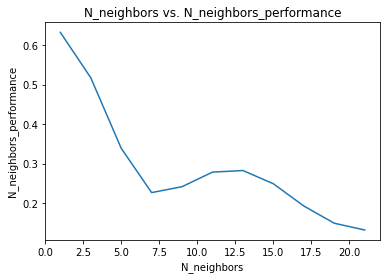


**Fig. 11.** N-neighbors max vs. N-neighbors Performance Dataset2

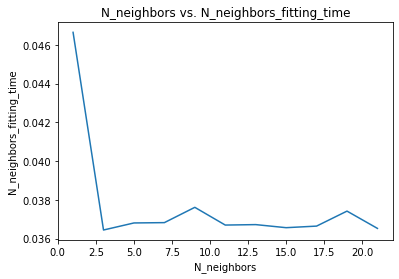


**Fig. 12.** N-neighbors max vs. N-neighbors Fitting Time Dataset2

As can be observed from figure 11, the performance decreases when the k increases. So, the same loop should be repeated by taking the first part (from 1 to 22) and decreasing the step to be 2 to have better results.



**Fig. 13.** N-neighbors first segment vs. N-neighbors Performance Dataset2



**Fig. 14.** N-neighbors first segment vs. N-neighbors Fitting Time Dataset2

Figure 13 and 14 shows that the best option for n-neighbors parameter is 1 selected by getting the max performance from the array of performances and using its index to get the number stored in the same index in the array of number of neighbors. So, the performance for this best K was 0.6329262355500526 and the fitting time for this best K was 0.04664980000000085 which is reasonable. In the following experiment, the K will be the best K = 1.

## 4.2.3.2 Weights

This parameter determines whether to give a weight for each neighbor used in prediction or not based on the Euclidian distance. The two options are uniform (no weights and all neighbors are equal in weights) and distance where each neighbor has a weight in the prediction process so that the nearer points will have more effect on the predicted output. Table 16 lists the comparison between the two options regarding the speed and the performance.

**Table 16**

Weighting Criterion Results Dataset2

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter** | **Measured Parameter** | **“Uniform” Result** | **“Distance” Result** |
| Efficiency (Speed) | Fitting Time | 0.0473081999999998 | 0.04255599999999937 |
| Effectiveness (Predictive performance) | Performance | 0.6329262355500526 | 0.6329262355500526 |

The obtained results clarifies that this parameter options have the same performance as the best k is already 1 so that the distance weight is the same as uniform (only one neighbor). As there is no difference noticed from the table, choosing any weighting criterion will be good. Distance option was chosen.

## 4.2.3.3 Final Model Results and Analysis

After selecting the above-mentioned parameters best options: N-neighbors = 1 and weighting criterion = distance, the K-Nearest Neighbors Regression algorithm had the results listed in Table 17.

**Table 17**

K-Nearest Neighbors Regression first dataset final model results dataset2

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Measured Parameter** | **Result** |
| Efficiency (Speed) | Implementation Time | 1.6778560000000002 |
| Fitting Time | 0.04417129999999858 |
| Effectiveness (Predictive performance) | Performance | 0.6329262355500526 |

From the final model results, it can be noticed that the performance of K-Nearest Neighbors regression is a bit low, and the implementation time, taken to decide on the best parameters’ options and evaluating each model using the 10-fold Cross-Validation, is reasonable.

## 4.2.4 Selecting Best Model Among All Algorithms

In this subsection, the final three models generated from applying the three different algorithms are compared according to their speed and performance. Table 18 shows the comparison between the final models.

**Table 18**

Final Models Comparison Dataset2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Parameter** | **Measured Parameter** | **Linear Regression** | **Random Forest Regression** | **K-NN Regression** |
| Efficiency (Speed) | Implementation Time | 0.0494309 | 4.25924 | 1.67786 |
| Fitting Time | 0.0493514 | 0.103855 | 0.0441713 |
| Effectiveness (Predictive performance) | Performance | 0.625262 | 0.993 | 0.632926 |

By observing the comparison in table 18, it is found that the Linear Regression algorithm has the best speed as it has the least implementation time, and the K-NN regression final model has the best speed as its fitting time is the minimum. K-NN regression is similar to linear regression in the fitting time as the number of datapoints is small. The Random Forest Regression algorithm has the best performance and a reasonable implementation time. The K-NN regression algorithm has a low performance. So, if we are interested in the best performance, Random Forest algorithm is the best option. If we are interested in the total implementation time, Linear regression algorithm is the best option. If we need a best fitting time, K-NN regression is the best option.

## 4.2.4 Testing the Best Model

Using the whole training set to fit the final models and the test set which is 20% of the dataset, the best models from each algorithm will be tested by measuring their accuracy by comparing the predicted output by each final model to the original output in the test set. This comparison is made by using the mean squared error (MSE) criterion. Table 19 shows the accuracy of each final model.

**Table 19**

Final Models Testing Accuracy Dataset2

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter** | **Linear Regression for best implementation time** | **Random Forest Regression for best performance** | **K-NN Regression for best fitting time** |
| MSE | 54.009119041137104 | 0.6761338709677411 | 31.32443548387096 |

## 4.3. Comparison of results

As the two datasets are different and their size is different, it is expected that the results obtained from implementing the same algorithms should be different. Table 20 shows a comparison between the final models testing accuracy of the two datasets.

**Table 20**

Comparison between the two datasets

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter** | **Linear Regression for best implementation time** | **Random Forest Regression for best performance** | **K-NN Regression for best fitting time** |
| MSE for dataset 1 | 20.27370599968745 | 11.213581298164849 | 17.142891341908364 |
| MSE for dataset 2 | 54.009119041137104 | 0.6761338709677411 | 31.32443548387096 |

## 4.3.1 Similarities

In both datasets, the linear regression algorithm had the least implementation time and the lowest performance which is logic due to the simplicity of the algorithm. In the Random Forest Regression algorithm, both datasets had the best number of n-estimators in between 5 and 10 decision trees. This is because both performance and speed were considered in the selection of n-estimators. In both datasets, the performance was satisfactory as in the first dataset the performance was 0.956, and in the second dataset the performance was 0.993. In both datasets, when increasing the k neighbors of the K-NN algorithm (after a specific threshold which was approximately 100 neighbors in both datasets), the performance decreases. This is because more neighbors may cause overfitting and may smooths out the prediction. In both datasets, the MSE criterion for the quality of splitting the tree in random forest algorithm was better than MAE regrading the fitting time as MSE differentiates between the errors well as it takes the square of the errors.

## 4.3.2 Differences

In K Nearest Neighbors algorithm, the first dataset had a k of 69 neighbors to get the best performance and the weight parameter was best when it was distance not uniform. While in the second datasets, the best k was only 1 neighbor and there is no difference between using uniform and distance weight which results in a fitting time like the fitting time of linear regression algorithm. In the second dataset the number of records was less than the first dataset, which resulted in not calculating the Euclidean distance for many numbers of datapoints and the fitting time of the K-Nearest Neighbors regression in the second dataset was smaller than the fitting time in the first dataset. In random forest algorithm, using the max-features parameter as auto in the second dataset resulted in higher performance than using sqrt and log2 as the number of features was 6, while in the first dataset sqrt option was the best for time and performance consideration as the number of features was only 4 and there is no big difference between 4 and sqrt (4) features consideration in the performance.

## 4.3.3 Suggestion

For the first dataset, the best algorithm is the random forest if we considered the performance more than considering the time. For the second dataset as it has a smaller number of records, the random forest regression has a great speed. So, for dataset 2, the best algorithm is random forest regression regarding speed and performance.

# References

1. A. Chugh, “MAE, MSE, RMSE, Coefficient of Determination, Adjusted R Squared - Which Metric is Better?,” Medium, 08-Dec-2020. [Online]. Available: <https://medium.com/analytics-vidhya/mae-mse-rmse-coefficient-of-determination-adjusted-r-squared-which-metric-is-better-cd0326a5697e#:~:text=Mean%20Squared%20Error(MSE)%20and,Mean%20Absolute%20Error%20(MAE).&amp;text=MAE%20is%20more%20robust%20to,R%20square%20is%20considered%20desirable>. [Accessed: 02-May-2021].
2. A. SinghAn, “K-Nearest Neighbors Algorithm: KNN Regression Python,” Analytics Vidhya, 25-May-2020. [Online]. Available: <https://www.analyticsvidhya.com/blog/2018/08/k-nearest-neighbor-introduction-regression-python/>. [Accessed: 02-May-2021].
3. C. Bakshi, “Random Forest Regression,” Medium, 09-Jun-2020. [Online]. Available: <https://levelup.gitconnected.com/random-forest-regression-209c0f354c84>. [Accessed: 02-May-2021].
4. J. Brownlee, “A Gentle Introduction to k-fold Cross-Validation,” Machine Learning Mastery, 02-Aug-2020. [Online]. Available: <https://machinelearningmastery.com/k-fold-cross-validation/#:~:text=Cross%2Dvalidation%20is%20a%20resampling,k%2Dfold%20cross%2Dvalidation>. [Accessed: 02-May-2021].
5. Hastie T., Tibshirani R., Friedman J., The elements of statistical learning: Data mining, inference and prediction, 2nd edition, Springer, 2009, 763 p. <http://web.stanford.edu/~hastie/ElemStatLearn/>
6. M. Lacey, Multiple Linear Regression. [Online]. Available: <http://www.stat.yale.edu/Courses/1997-98/101/linmult.htm>. [Accessed: 02-May-2021].
7. P. Tüfekci and H. Kaya, UCI Machine Learning Repository: Combined Cycle Power Plant Data Set. [Online]. Available: <http://archive.ics.uci.edu/ml/datasets/Combined+Cycle+Power+Plant#>. [Accessed: 02-May-2021].
8. R. Lopez, UCI Machine Learning Repository: Yacht Hydrodynamics Data Set. [Online]. Available: <http://archive.ics.uci.edu/ml/datasets/Yacht+Hydrodynamics>. [Accessed: 03-May-2021].
9. scikit-learn developers, “sklearn.linear\_model.LinearRegression,” scikit. [Online]. Available: <https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html>. [Accessed: 02-May-2021].
10. scikit-learn developers, “sklearn.neighbors.KNeighborsRegressor,” scikit. [Online]. Available: <https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsRegressor.html>. [Accessed: 02-May-2021].

# Directly used references

1. 1. A. Chugh, “MAE, MSE, RMSE, Coefficient of Determination, Adjusted R Squared - Which Metric is Better?,” Medium, 08-Dec-2020. [Online]. Available: <https://medium.com/analytics-vidhya/mae-mse-rmse-coefficient-of-determination-adjusted-r-squared-which-metric-is-better-cd0326a5697e#:~:text=Mean%20Squared%20Error(MSE)%20and,Mean%20Absolute%20Error%20(MAE).&amp;text=MAE%20is%20more%20robust%20to,R%20square%20is%20considered%20desirable>. [Accessed: 03-May-2021].
2. J. Brownlee, “A Gentle Introduction to k-fold Cross-Validation,” Machine Learning Mastery, 02-Aug-2020. [Online]. Available: <https://machinelearningmastery.com/k-fold-cross-validation/#:~:text=Cross%2Dvalidation%20is%20a%20resampling,k%2Dfold%20cross%2Dvalidation>. [Accessed: 02-May-2021].
3. M. Lacey, Multiple Linear Regression. [Online]. Available: <http://www.stat.yale.edu/Courses/1997-98/101/linmult.htm>. [Accessed: 02-May-2021].
4. P. Tüfekci and H. Kaya, UCI Machine Learning Repository: Combined Cycle Power Plant Data Set. [Online]. Available: <http://archive.ics.uci.edu/ml/datasets/Combined+Cycle+Power+Plant#>. [Accessed: 02-May-2021].
5. R. Lopez, UCI Machine Learning Repository: Yacht Hydrodynamics Data Set. [Online]. Available: <http://archive.ics.uci.edu/ml/datasets/Yacht+Hydrodynamics>. [Accessed: 03-May-2021].