Regression Modelling of Power Plant Electrical Energy Output Based on Ambient Variable Features

Finlo Heath
MSc Robotics
Bristol University, University of the West of England
Bristol, England
finlo.heath.2022@bristol.ac.uk

***Abstract*—This project compares mean-based, linear and KNN regression for the purpose of predicting the hourly electrical energy output of a power plant. It was found that KNN performed with the lowest error, using K = 8 neighbours, distance weighting and a ball tree algorithm, with MSE = 13.66 across all test data.**

***Index Terms*—AI, Regression, KNN, MSE, R Squared, Energy Prediction**

I. Introduction

In this project, we consider a combined cycle power plant which outputs electricity. Our aim is to predict the sum electrical energy output per hour PE, using the following measurements:

- Temperature (AT)
- Ambient Pressure (AP)
- Relative Humidity (RH)
- Exhaust Vacuum (V)

To do this, we will use regression algorithms, as the target variable falls in a continuous range of values. Our regression algorithms will take our four feature variables and attempt to map them to the *Real Line* ($\mathbb{R}$) [1]; the range of values that PE takes. This task is a supervised learning problem, as we have labelled data to train our algorithms. Therefore we will not use clustering, which is only appropriate for unlabelled data. Classification algorithms are only appropriate if the outcome possibilities are separate and discrete, which is not the case here. Some regression algorithms make use of hyper-parameters, which are variables which cannot be optimised by looking at the data alone, as they are set before training the model [2]. They must be optimised via validation, choosing hyper-parameter values to maximise algorithm performance on validation datasets.

II. Methods

Three regression algorithms have been developed for comparison: A simple Baseline, Linear Regression and K-Nearest Neighbours Regression. The labelled dataset was split into two subsets, where 80% was used as a training dataset, and the remaining 20% was held out for testing.

Performance of regression algorithms is best quantified using mean squared error (MSE), or root mean squared error. The coefficient of determination, R^2, is equally useful, where the closer to $R^2 = 1$, the better the prediction of the model. However, the R^2 equation is simply a normalised rearrangement of MSE, given it can be derived from the equation for MSE. Therefore they are functionally equivalent and, barring mathematical error will always agree. Hence, we consider primarily MSE in our evaluation of these algorithms. Accuracy, Recall and Precision apply only to classification problems, where we can clearly define a prediction as true positive/negative or false positive/negative. This cannot be performed for regression algorithms, where the prediction will differ from the target by a degree of error as opposed to being correct or incorrect.

The baseline uses mean-based regression. It takes the target values of PE provided in the training dataset, calculates their average and predicts every unseen data point as mapping to this single average value. This is incredibly reductive, given our feature space is four-dimensional and none of this is used to calculate the one-dimensional output space. A mean-based baseline was chosen over a median-based baseline due to its superior performance in preliminary testing. We aim for both competitive algorithms to significantly outperform the baseline.

The first competitive algorithm is a linear regression model. As with the baseline, linear regression does not make use of hyperparameters. An advantage of this is that it is quicker to implement and test, requiring no re-calibration to optimise performance for new data. This can be useful for quickly investigating how features map to outputs in preliminary investigations of new data, potentially indicating which of the more complex regression algorithms would be the best choice for the optimal model. Linear regression applies the assumption that each dimension of the feature space is independent and varies linearly with the target variable PE. This allows the relationship between feature space X and output space $\vec{y}$ to be written as $\vec{y} = X\theta$. Here θ is the vector of coefficients which maps our features to our targets and can be calculated with the pseudo-inverse of X, multiplied by $\vec{y}$: $\theta = (X^T X)^{-1} X^T \vec{y}$. Once θ is calculated, predictions on unseen data are simply made by performing $\vec{y}_{pred} = X_{test}\theta$.

The second competitive algorithm is a K Nearest Neighbours model. Inherently, the key assumption of this

model is that data points near to each other will map to similar target values. We optimise three hyper-parameters to maximise the performance of this model: the weighting type (*distance* or *uniform*), the algorithm (*ball tree*, *kd tree* or *brute*) and the number of neighbours K. To optimise weight and algorithm, loops were used to train and predict the regressor using each weight with each algorithm, outputting the MSE as a function of the number of neighbours accounted for (K), on a validation dataset. This is shown in Fig. 1.

Weight is the proportion each neighbour contributes to the prediction. *Uniform* weighting means this is equal for each neighbour considered, whereas *Distance* means closer neighbours contribute more significantly to the prediction. Therefore, if feature data is a good predictor of target values, *Distance* is usually more appropriate. Algorithm is the method by which the nearest neighbours are computed. *Auto* is a fourth option, but this simply selects from the other three based on the data. Changing the algorithm had a small effect compared to changing the weighting, hence the lines using the same weights with different algorithms are indistinguishable. The lowest MSE is given using distance weighting with either ball tree or kd tree algorithms.

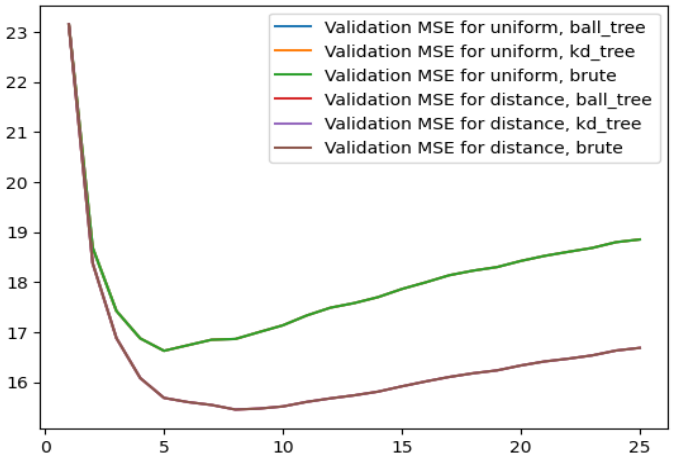
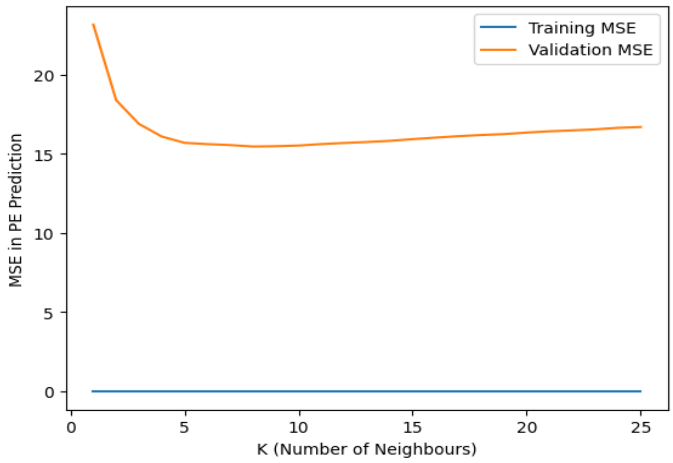
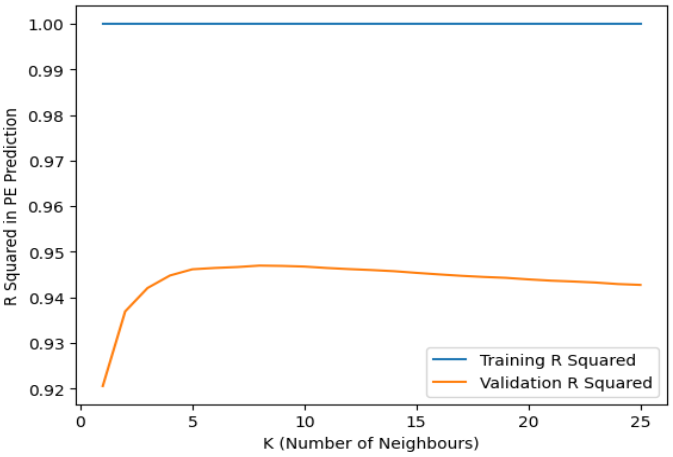


Fig. 1. Plotting MSE as a function of k neighbours for each weight and algorithm choice. The change in weight has a much greater effect than the change in algorithm. The lowest MSE is given using distance weighting with either ball tree or kd tree algorithms.

Using these optimal hyper-parameters, we then repeat the predictions across K and measure training and validation MSE and R^2, selecting the value of K which minimises MSE and maximises R^2. Since these two are mathematically equivalent metrics, it was found that $K = 8$ is optimal for both. This is displayed in Fig. 2. All three hyper-parameters were thus optimised for our dataset. *Leaf Size* is another hyper-parameter which could have been considered for optimisation. However, changing this variable significantly affects the running speed of the algorithm, thus looping over many variations of leaf size is a slow process, especially when nested with variations in other hyper-parameters; the number of loop iterations increases exponentially with each additional hyper-parameter considered.



(a) MSE as a function of K



(b) R^2 as a function of K

Fig. 2. (a) & (b) display how the choice of K affects algorithm performance. As K increases from 1 to 8, the error in the algorithm decreases to a minimum on the validation dataset. Beyond 8 neighbours, the error starts to increase as the neighbours being accounted for become more distant.

It was decided that optimising leaf size offered negligible gain considering the computational cost.

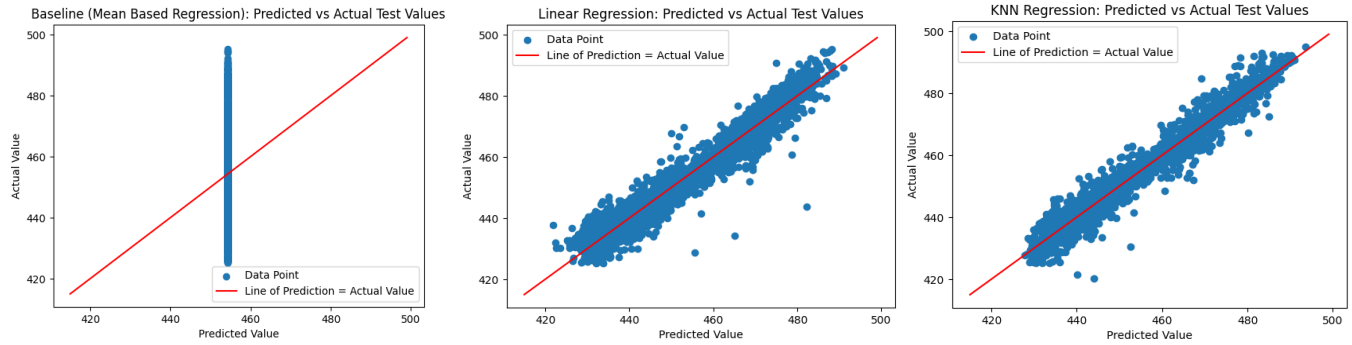
III. Results & Analysis

Algorithm	**MSE**	R^2
Baseline (Mean-Based Regression)	285.72	0.00
Linear Regression	21.90	0.93
KNN Regression	13.66	0.95

TABLE I
TABLE DISPLAYING MSE & R^2 FOR EACH ALGORITHM.

Table I displays the final MSE for each algorithm, clearly showing that our optimised KNN algorithm provides superior performance. As expected, the R^2 value for the baseline algorithm is $R^2 = 0$. Given that its prediction for targets $\vec{y}$ is $\vec{y}_{pred} = \bar{y}_{training}$ (the mean of the training targets).

$$\bar{y}_{training} \approx \bar{y}_{test} = \bar{y}$$



(a) MSE as a function of K (b) R^2 as a function of K (c) R^2 as a function of K

Fig. 3. (a), (b) & (c) display the performance of the algorithm. The red line $y = x$ marks a perfect prediction.

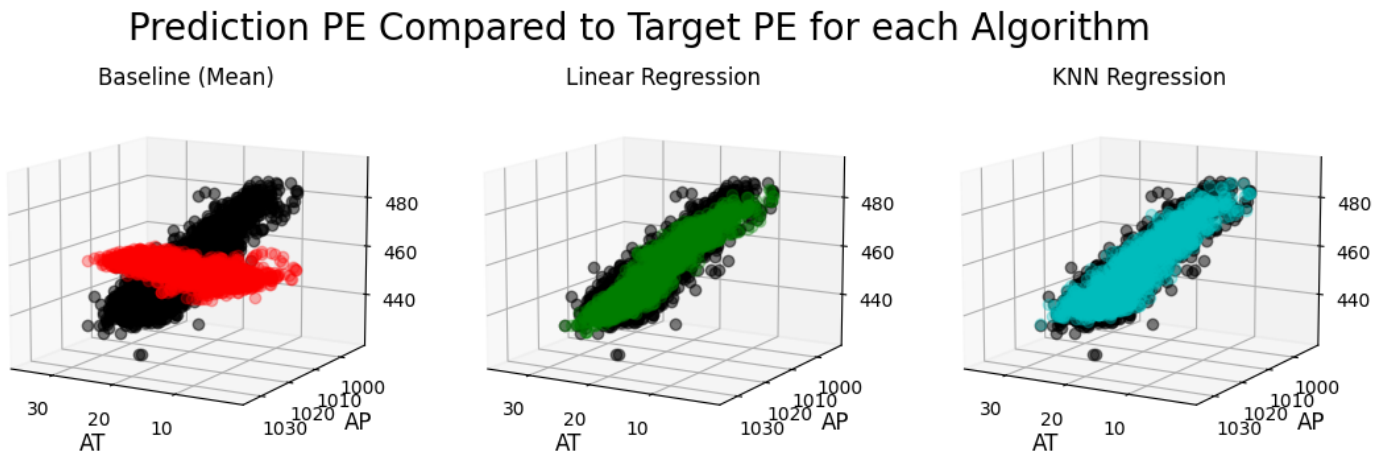


Fig. 4. 3D plots comparing the target data (black) with the predictions of each algorithm (coloured). We can see the KNN algorithm matches the targets best.

$$R^2 = 1 - \frac{\sum \vec{y} - \vec{y}_{pred}}{\sum \vec{y} - \bar{y}} = 1 - \frac{\sum \vec{y} - \bar{y}}{\sum \vec{y} - \bar{y}} = 1 - 1 = 0$$

R^2 is a measure of how much better the algorithm predictions fit the data compared to simply taking the mean. With values of 0.93 and 0.95 respectively, linear and KNN regression are both far superior predictors than the mean. When Uniform weighting was used over distance for KNN regression, MSE in predictions was significantly higher, indicating that this hyper-parameter has been chosen correctly. The fact that the MSE of the linear regression algorithm is higher than that of KNN is an indication that the independence of features assumption - or the assumption that they each vary linearly with PE - does not hold. To test the linear variance assumption, future work should investigate the use of polynomial regression to explore whether it can achieve better predicting power. In comparison, KNN makes no assumptions about the data or relationships between features, therefore it generalises better to variations in data, resulting in a lower MSE.

The fit of each algorithm's prediction to the targets is visualised in Fig. 3. This figure plots the targets against the predictions, with the red line $y = x$ indicating a perfect prediction. Points below the red line are therefore overestimated predictions, points above the red line are underestimated predictions. The baseline (a) performs poorest, as it predicts all target values to the mean. Both linear and KNN regression clearly predict the data with a measure of intelligence, their data points correlating strongly with the line $y = x$. This shows that these algorithms are good predictors of PE using the feature data. Linear regression has overestimates significantly on certain points which land far below the line of perfect prediction. KNN has fewer extreme outliers compared to linear regression, resulting in a lower overall MSE. KNN appears to underestimate and overestimate predictions with roughly equal frequency.

The fit of the predictions can also be explored as a function of the feature variables. Plotting the target PE on the vertical z-axis, a three-dimensional graph allows for PE as a function of two of the four feature variables. Here, pressure and temperature have been arbitrarily chosen. Fig. 4 displays PE as a function of pressure AP and temperature AT, in a three-dimensional point cloud. The black points are the target data, overlaid with predictions in varying

colours. The baseline predicts the same value of PE for all input data, hence the red cloud matches poorly to the target cloud. KNN again outperforms linear regression as it better predicts the spread of the data, where variances in pressure and temperature can result in similar PE values. The linear algorithm predicts too tightly clustered PE outputs, resulting in a higher comparative error. These point clouds can be viewed in more detail in the appendix A. The mapping of KNN predictions to targets is evidence that this model can generalise to unseen test data.

Based on these results, we conclude that tuned KNN is the best algorithm for predicting the hourly electrical energy output of the plant. This is because it does not make incorrect assumptions about the data and thus is better able to map the breadth of input feature values to concentrated output PE values. In contrast, both the baseline and linear regression models were too simplistic to predict as accurately on this dataset, though the results using linear regression have an impressively low error considering the input data is four-dimensional. Improvements can still be made to the KNN model, considering Fig. 4, we see that this model does not account for outlier points in the targets. This suggests the model needs further tuning or that further dimensions to the feature space are required to fully capture all of the causal variables affecting PE. Ultimately, as a relatively simple algorithm to implement, the KNN model has utility as a predictor that power-plant managers can use to assess with reasonable accuracy the delivery capabilities of the plant on an hourly basis. Being able to predict this capability allows for the mitigation of energy shortages, reducing the chance of blackouts and disruption to safety-critical services.

REFERENCES

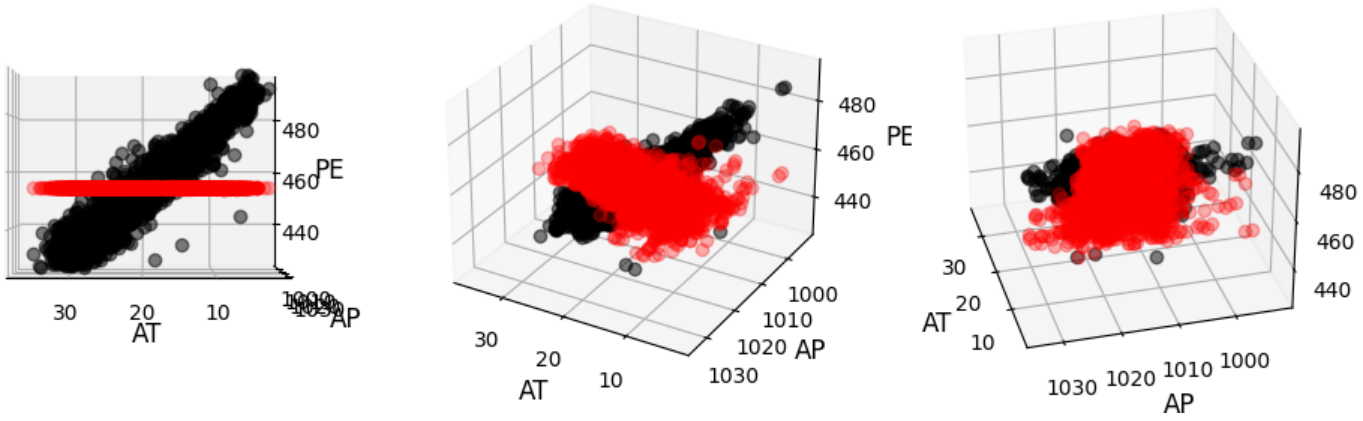
[1] L. H. Kahane, "Regression basics," 2008.

[2] P. Probst and B. Bischl, "Tunability: Importance of Hyperparameters of Machine Learning Algorithms," *Journal of Machine Learning Research*, vol. 20, pp. 1–32, 2019. [Online]. Available: http://jmlr.org/papers/v20/18-444.html..

APPENDIX

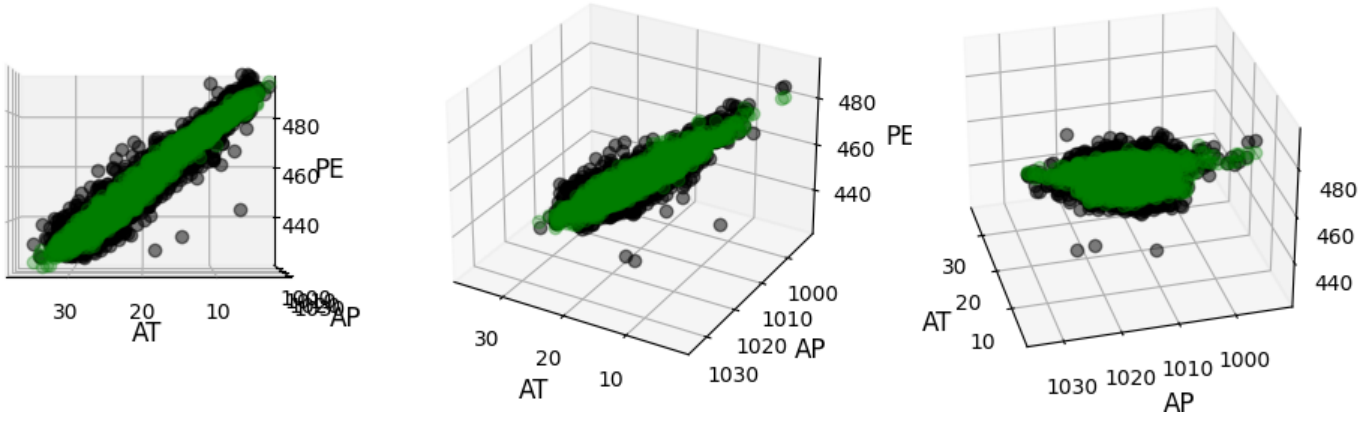
On the following page, the reader can view the 3D result plots for each algorithm from multiple angles, shown in Fig. 5.

$MSE = 285.721$



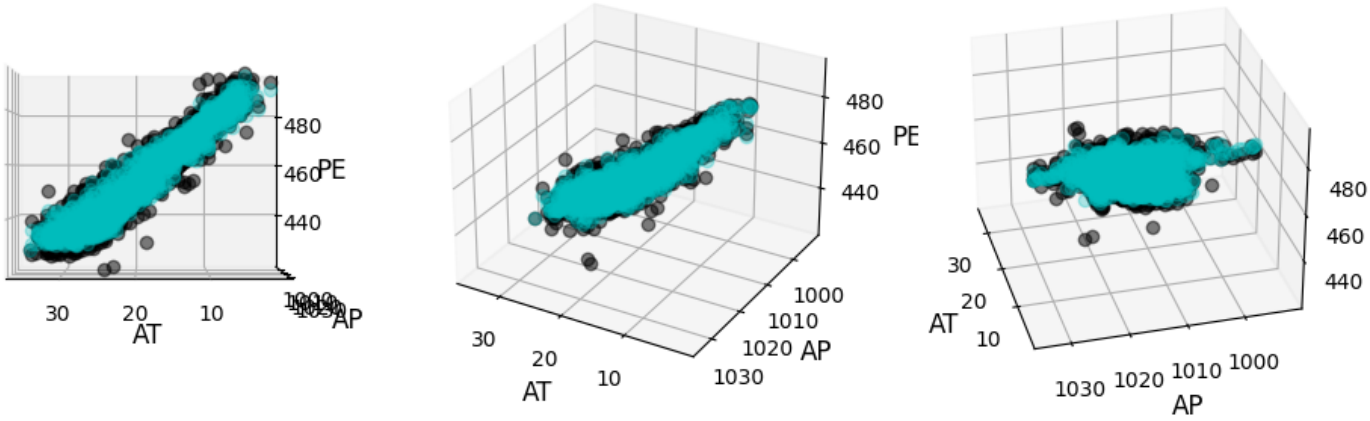
(a) Baseline (Mean-Based Regression) Prediction

$MSE = 21.089$



(b) Linear Regression Prediction

$MSE = 13.657$



(c) KNN Regression Prediction

Fig. 5. 3D plots comparing the target data (black) with the predictions of each algorithm (coloured), as functions of ambient pressure (AP) and temperature (AT). We can see the KNN algorithm matches the targets best.