EMATM0044: Question 1

James Reddaway (ID: 2190286)

I. Introduction

The dataset used presents the information for the number of rental bikes hired out per hour, combined with the statistics for several other variables including temperature, humidity and rainfall. From inspection of the data, the majority of the variables follow a continuous range of values, which would indicate that the problem may benefit from being treated as a regressive task. However, the variables such as seasons and holidays present data that can be defined by discreet class labels, which would best fit with the use of a classification algorithm.

As the majority of the variables presented in the dataset follow a continuous scale, the problem has been treated as a regressive problem, with supervised regressive algorithms applied to solve it and make predictions. Although, the values identified as belonging to discreet classes have not been excluded, but instead encoded for improved use with the selected type of algorithm.

II. Methods

A. Algorithm choice

To build models to fit the dataset, two regressive algorithms have been implemented and evaluated both against each other and a baseline. The two chosen algorithms are k-nearest neighbour (K-NN) and a multi-layer perception (MLP), which have both been applied with the use of the scikit-learn python libraries [1].

B. Evaluation metric

To evaluate the performance of the algorithms, the R^2 metric has been used (also known as the coefficient of determination metric [2]), with the equation presented in Fig.1. This particular evaluation metric has been used firstly, as it is suitable for use with these and other types of regression algorithms and models, as opposed to a classification metric which would not provide sufficient evaluation in this case. Additionally, it has been identified that other regressive metrics such as mean-squared error (MSE) can often overfit to noisy data, ignoring the most important information in some cases. This has been highlighted, as the dataset in use, does contain several data points that do not fit a constant trend and could be considered noise. Finally, it has been suggested that metrics such as mean absolute error (MAE), MSE and root mean squared error cannot be used to show the real quality of a regressive model, whereas R^2 can [3].

C. Baseline

As means of generating a baseline model to compare the algorithms against, the dummy function from scikit-learn has been evaluated. However, the performance of the model generated either a negative score of 0 with the R^2 metric, indicating that the model provided a worse fit than the average line [3]. Therefore, the scikit-learn library has been used to generate a decision tree (DT) model, which has been found to fit the data with much greater efficiency and therefore provides a more competitive baseline to be used. For hyperparameter selection of the baseline, the default values as provided in the scikit-learn documentation for the DecisionTreeRegressor function have been applied.

D. Data Processing

Prior to the training and evaluation of the models, the dataset has been processed to achieve a more suitable format for some variables. Namely, those that represent discreet data such as seasons, have been encoded using One-Hot encoding. This converts the variable's data into a binary string which has been identified as providing increased performance in many machine learning models over their original representation or an integer value [4]. Normalisation has been applied to the full dataset, which has again proven advantageous for a number of machine learning models due to a reduced range of the data [5]. Additionally, the train and test data have been divided using k-fold cross-validation with a total of 5 splits, with the intent of reducing the chance of model bias. However, due to the hyperparameter search process and associated training times, cross-validation has only been applied during the evaluation process. Finally, the variable for date has been excluded from the dataset, due to the additional complexities and requirements that would be required for encoding.

E. Hyperparameter selection

For the selection of hyperparameters, a genetic algorithm (GA) has been applied again using the scikit-learn python libraries [1], [6]. In this process, generations of values are passed to the regression algorithms, with the R^2 value returned and used as the candidate's fitness. For the selection of parameters that have been used to train for the K-NN and MLP models, the candidate with the maximum fitness has been

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

Fig. 1: R^2 metric equation

taken from running the evolutionary algorithm multiple times. The use of multiple runs has been employed in an attempt to reduce the chance of converging to a non-global optima (5 times for both K-NN and MLP).

To select the parameters for the evolutionary algorithm (shown in TABLE I), a simple trial and error test was used, resulting in values that provide good performance, although likely non-optimal.

To select the hyperparameters that were evolved within the GA, the documentation for the functions has been used to identify all parameters that were not dependent on others. For example, the 'power_t' parameter for the MLPRegressor function only becomes applicable when the 'sgd' solver is used, and therefore has not been evolved with an aim to reduce the search space and complexity of the problem. Therefore, for those parameters that have not been evolved, the default values described in the documentation have been applied where applicable.

The results of the genetic algorithm optimisation are presented in Fig.4, showing the mean overall fitness, as well as the mean of the maximum fitness from all 5 runs that were conducted for each algorithm. The plots indicate that MLP achieved an overall higher average fitness with the highest maximum value achieved from all generations of both algorithms.

As verification that the GA was successful in identifying suitable hyperparameters, the effect of varying some of the key parameters has been explored, with an example being the number of neighbours applied to the K-NN model. The plot in Fig. 2 presents how the R^2 score changes as the number of neighbours (K) is varied, illustrating that the highest score was achieved with a value of 5, with a value of 7 producing a very similar result. For the K-NN model, this aids in confidence that the GA has converged to an effective solution, as a value of 7 was used in the candidate with the greatest fitness (as shown in TABLE II).

A similar process has been completed for the number of layers and nodes for the MLP model, with the results plotted in Fig. 3. This test provides a more unexpected result, as, after 15 nodes, all architectures achieved scores in a similar range, suggesting selection becomes somewhat arbitrary. These results may be caused by the more limited selection of architectures that have been applied for choice in the search process, in an attempt to reduce the search space and aid in training times. This, therefore, highlights an area that could be investigated further in any future work that is completed.

From completing the evolutionary algorithm for both methods, the hyperparameters in TABLE II have been selected, as they have produced the highest R^2 values during training.

K-NN parameters: For the K-NN model, a value of 7 has been selected by the GA, meaning that the closest 7 neighbours to the point being predicted are queried to generate the most likely prediction. For deciding the weights of the neighbouring data points, the distance from the current test point is used, with the closest neighbours carrying the greatest weighting and therefore highest influence on the prediction. When calculating

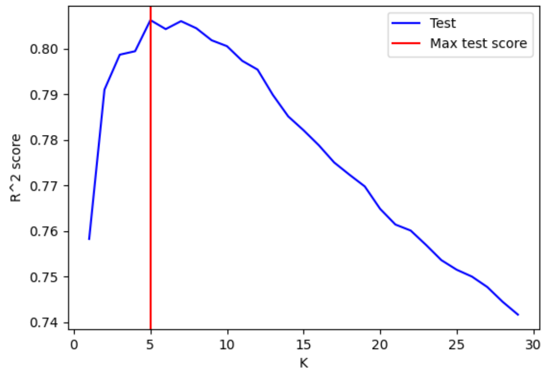


Fig. 2: Average testing data performance of K-NN model with varying K values

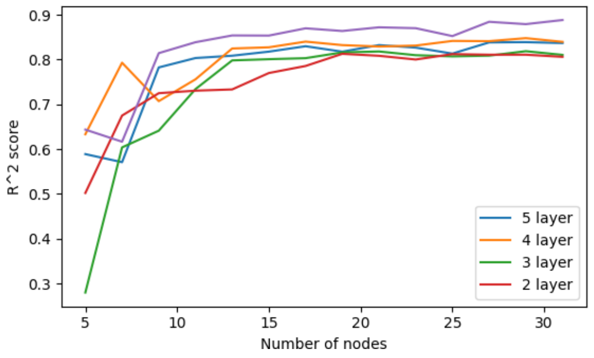


Fig. 3: Average testing data performance of MLP model with varying architecture

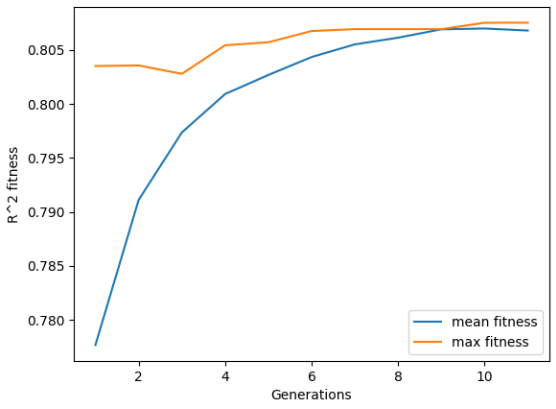
the nearest neighbours, a brute-force search is applied. This method involves calculating the distance between the test point and all other points in the set, creating an inefficient but complete search. However, for a relatively small dataset such as this, the inefficiency of the method provides less of a constraint. The leaf size parameter dictates the value used in the search for neighbours algorithms. However, as this candidate from the GA has selected brute-force for neighbour selection, leaf size is not directly applicable and bears no significance to performance in this case. For all GA candidates, the Minkowski metric has been applied to calculate distances within the neighbour search trees, with the value of p dictating the power parameter for this metric. Concisely, a value of 1 for p is equivalent to Manhattan distance.

MLP Parameters

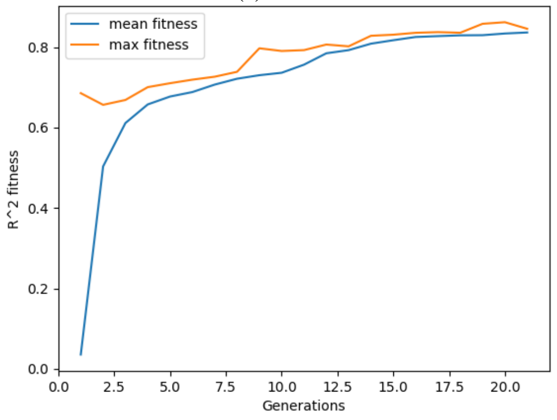
For the hyperparameters of the MLP, the hidden layer size dictates the architecture of the model, defining the number of layers and nodes that are applied. From the GA search, the candidate with the maximum fitness has used a 5 layer network with 20 nodes per layer. The rectified linear unit (ReLU) function has been used for the activation of the hidden layers, removing all negative values and maintaining those that are positive and therefore most useful information. The solver parameter defines the algorithm used for node weight

Algorithm	Population size	Generations	Cross-over probability	Mutation probability
k-NN	10	11	0.1	0.9
MLP	10	21	0.5	0.5

TABLE I: Genetic algorithm values for hyperparameter selection



(a) K-NN



(b) MLP

Fig. 4: Evolution fitness plots

optimisation, with Limited-memory BFGS (LBFGS) providing the best results. The alpha value sets the L2 penalty that is applied as a regularisation term in training, with a value of 0.0001 being selected. Batch size sets the mini-batch size when stochastic optimisation is used for weight optimisation, therefore, this is not applicable in this case due to the use of the LBFGS algorithm. The maximum iteration value limits the number of iterations that can be run in the scenario when convergence is not achieved during training. For all candidates, this has been fixed to a value of 5000. Finally, the tol parameter defines the loss of value score that must be achieved in a set number of iterations (the default value of 10 has been applied throughout for this) to recognise that convergence has not yet been achieved.

F. Testing Method

To evaluate performance, the models for each algorithm have been trained using the values optimised through the GA search process. Specifically, each model has been trained a total of 10 times to provide the mean performance, with a different permutation of data applied each time in the distribution of 80% training and 20% for testing. Although varying data has been used for each training cycle, it is important to note that the data has been kept consistent for training cycles between models, namely, one permutation has been used for the first training cycle of all models. In addition to recording performance on the training and testing sets, a 10% holdout set has been generated and applied to the models, providing an unseen data series that can be used to assess the generalisation capabilities of the models. This dataset has remained constant between all evaluation cycles for each model. As previously noted, the R^2 metric has been used to provide a direct comparison between the models and baseline.

Algorithm	Hyperparameters applied
K-NN	k neighbors=7, weights='distance' algorithm='brute', leaf size=23, p=1
MLP	hidden layer sizes=(20, 20, 20, 20, 20) activation= 'relu', solver='lbfgs', alpha=0.0001, batch size=405, max iter=5000, tol=0.0001

TABLE II: Hyperparameters table

III. Results

A. K-NN Performance

The results of the training and evaluation have been shown in TABLE III, demonstrating that on average, K-NN achieved the maximum R^2 score of 1 when evaluated against the training data, with a standard deviation of 0. This would indicate that when considering the training data, the model has achieved exceptional performance. Alternatively, this may show that a level of overfitting was achieved for this data series, which would require additional testing to identify if this is a factor.

Evaluation of the K-NN test data scores provides a more accurate representation of the model's true performance. With a score of 0.806, the K-NN model achieved an improvement of 7.8% on the baseline, as illustrated in Fig.5. This increase demonstrates a reasonable improvement from using K-NN over DT, especially when considering similar training times were observed.

When considering the generalisation capabilities of the K-NN model, the score for the holdout data demonstrates comparable performance to that of the test data, with a score of 0.802, and a standard deviation of 0.006. Compared to the baseline, this provides an improvement of 5.7%. Although slightly lower, these scores would still suggest that the model has good generalisation for unseen data when considering the score for the test data. This is further illustrated in the plot shown in Fig. 6, showing the predictions that were made from both models against the ground truth values for the holdout data series.

	Mean R^2 score			R^2 Standard deviation		
Algorithm	Train	Test	Holdout	Train	Test	Holdout
DT baseline	1.000	0.748	0.759	0.000	0.027	0.019
K-NN	1.000	0.806	0.802	0.000	0.014	0.006
MLP	0.887	0.851	0.842	0.009	0.014	0.011

TABLE III: Mean training and test performance

B. *MLP Performance*

When tested, TABLE III illustrates that the MLP achieved a significantly lower training performance than both the baseline (an 11.3% decrease) and K-NN models. There is potential for this to be created by underfitting to the data, which may be improved by applying a more extensive search for hyperparameter selection. Although, the baseline R^2 score of 1 for the training data could also indicate possible overfitting, and therefore provide a poor comparison for this set.

Although the average training score showed a considerable reduction when compared to the baseline, test scores achieved an increase of 13.8%, which is found to be the best of the tested models. If the model is underfitting, there is potential for this performance to be further improved by parameter selection as mentioned previously. Therefore, it may be possible for this algorithm to provide highly accurate performance on datasets such as the one used for testing.

Finally, for the holdout set, the MLP again achieved a considerable improvement of 11% on the baseline. Like the K-NN model, this shows a slight decrease when compared to testing scores, but overall does demonstrate that the model can effectively generalise to new data (Fig. 6).

C. *Performance Comparison*

The data and evaluation of both models strongly suggest that the MLP was able to achieve greater overall performance, with higher average R^2 scores for all data sets apart from training. Although, when evaluated against the baseline, both models were able to produce a considerable performance increase. When extending the scope of the evaluation, the MLP model did require noticeably greater training times in comparison to both the baseline and K-NN models. This could provide an important point of consideration for larger and more complex datasets.

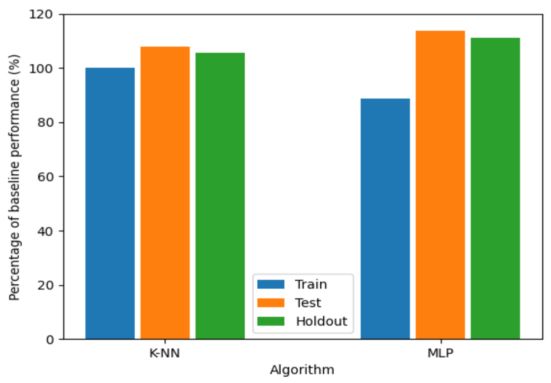


Fig. 5: Mean performance compared to the baseline

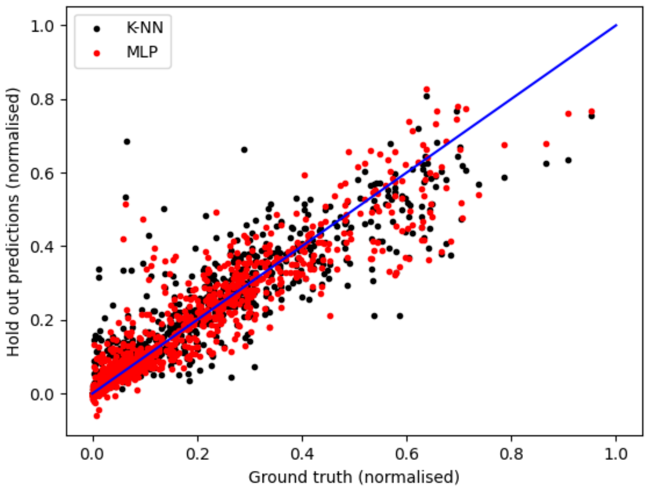


Fig. 6: Plot of the hold out set predictions against the ground truths

IV. Conclusions

In conclusion, for the task of predicting the number of rental bikes hired out per hour, an MLP model would provide better performance that a k-NN model. From evaluation, the MLP also provides the highest score with unseen data, suggesting a good ability to generalise. Furthermore, the use of a genetic algorithm has also achieved a good standard for hyperparameter selection when considering the possible number of permutations. Although, applying a more extensive search using this method would easily become computationally expensive and time-consuming.

To achieve greater performance for the dataset predictions, the dates of the bike rental could be encoded and included as an additional variable. This may provide additional information and indicate factors such as weekends, that may impact the number of rentals for those days. Furthermore, greater exploration of the MLP architecture may yield additional performance increases.

References

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

[2] A. Botchkarev, "Performance metrics (error measures) in machine learning regression, forecasting and prognostics: Properties and typology," *arXiv preprint arXiv:1809.03006*, 2018.

[3] D. Chicco, M. J. Warrens, and G. Jurman, "The coefficient of determination r-squared is more informative than smape, mae, mape, mse and rmse in regression analysis evaluation," *PeerJ Computer Science*, vol. 7, p. e623, 2021.

[4] Zach, "How to perform one-hot encoding in python - statology." Available at https://www.statology.org/one-hot-encoding-in-python/ (2022/20/04), Sep 2021.

[5] Available at https://datagy.io/pandas-normalize-column/ (2022/20/04), Nov 2021.

[6] R. Arenas, "Tune your scikit-learn model using evolutionary algorithms." Available at https://towardsdatascience.com/tune-your-scikit-learn-model-using-evolutionary-algorithms-30538248ac16 (2022/15/04), May 2021.