EMATM0044 Introduction to AI - Coursework Part 1

This report contains all the answers to Question 1 of the Summative Assessment for EMATM0044: Introduction to AI Coursework.

1 Introduction

The given problem statement is to predict the number of rental bikes out at each hour, given the values of other features of the dataset. A supervised machine learning algorithm uses labelled datasets to train algorithms to classify data or predict outcomes accurately. Since the number of rental bikes out at each hour are continuous values, this is a supervised regression problem where an unknown function $f: \Omega \rightarrow \mathbb{R}$ where Ω is n-dimensional feature space and $\mathbb{R}$ is an output space of real numbers. Therefore, regression algorithms such as Linear regression, K-Nearest Neighbor (KNN) Regressor, Decision tree regressor and Random Forest regressor can be applied. The data was divided into train, validation and test sets (80-10-10 %) for evaluating the models. The test set was only used to test the performance of the best model.

2 Methods

This section contains information regarding the implementation of machine learning models, performance metrics selected, plots chosen, and hyper-parameter tuning.

2.1 Performance Metrics

The performance metrics chosen to evaluate the regression models are:

- Mean Squared Error (RMSE): It is defined as the root over mean squared error, i.e., the average of the square of errors. Errors are the differences between actual and predicted values - the lesser the RMSE, the better the model. RMSE has been used as the primary metric to compare and select the best model.
- R-Squared (R2) Score (Coefficient of Determination): It is defined as the proportion of the variance for a dependent variable that can be explained by independent variables. Unlike the previous metric, the R2 score is not a measure of how accurate the predictions are but instead is a measure of fit. It shows how well the data fit the regression model. The R2 score of 1 indicates that the model is perfect, whereas a value of 0 indicates that the model will perform poorly on an unknown dataset.

2.2 Hyper-parameter Tuning

Based on the importance and impact on the model's performance, the hyper-parameters are chosen. The optimal values for a hyper-parameter can be selected by methods such as cross-validation and other hyper-parameter tuning tools. GridSearchCV has been used for hyper-parameter tuning the regression models in this report. It is a tool from the sklearn package to find the best hyper-parameter values from a given set of parameter grids. However, GridSearchCV becomes a complex and time-consuming task depending on the range of values and the number of hyper-parameters. Due to this reason, RandomSearchCV, another hyper-parameter tuning tool, has been used to find random hyper-parameter values from a more extensive range of values.

2.3 Plots

Plots used to visualise the results:

- Actual vs Predicted values plot: A scatter plot depicts how well the model fits the data. Ideally, all the data points need to be close to the regression line as possible.

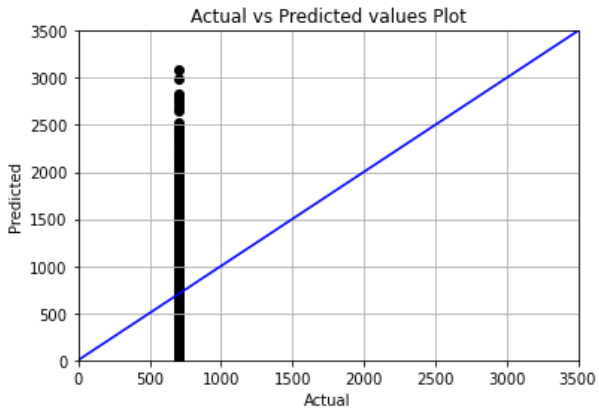
- Residual plot: A Residual is the difference between actual and predicted values. It measures how much a regression line vertically misses a data point. The residual clusters must be as close to the line as possible.

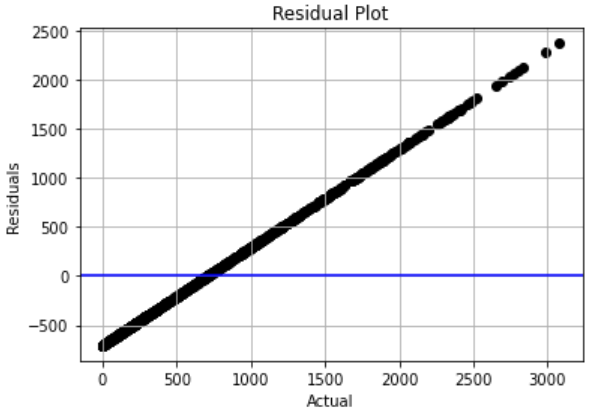
2.4 Machine Learning Models

Following machine learning models have been tested:

- KNN Regressor
- Decision Tree Regressor

Baseline model for comparison:
Sklearn's Dummy Regressor with default parameters has been chosen as a baseline model for comparison. It is a regressor that makes predictions on simple rules without any attention to the input data. The simple baseline predictions of the Dummy Regressor are compared with other regression models in this section.





As shown in the graphs above, the predictions made by the Dummy Regressor are very poor. The R-squared is very low, and RMSE is very high. As a result, the Dummy Regressor can test how well a conventional regression model fits a given dataset, but it can never be used to solve a real problem.

2.4.1 K-Nearest Neighbour (KNN) Regressor

The KNN algorithm is a supervised machine learning algorithm used to solve classification and regression problems. As the name suggests, it considers K Nearest Neighbours, i.e., data points, to predict the classes or continuous values for the new data points. KNN is a simple and easy to implement algorithm. There is no need to tune several hyper-parameters or make additional assumptions. Since the values ranged from 2000 to 0.0, the data was scaled for the distance-based KNN algorithm. The normalisation scaling technique was used to scale the data between 0 and 1.

2.4.1.1 Working

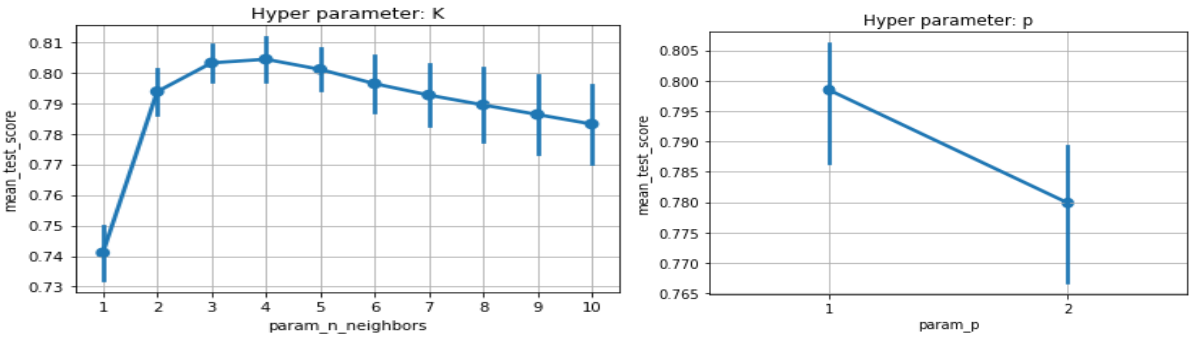
The KNN algorithm assumes that similar things exist near each other. Nearest neighbours are those data points with a minimum distance in feature space from the new data point. Closer neighbours have more significant influence than the ones that are further away. For regression, the average of the values of K-Nearest Neighbours from the training dataset is considered as the final prediction for the new data point. It calculates the distance between neighbours to make the predictions. This distance can be calculated depending on the problem statement.

2.4.1.2 Hyper-parameter tuning

- n_neighbors: represents the number of neighbours (K). Choosing the correct K value is one of the essential steps in implementing this model. As the K value decreases, the predictions become less stable. As the K value increases, the predictions become more stable up to a certain point; then, error increases.
- p: represents the power parameter for Minkowski distance. KNN regressor predicts the new data points based on the average of the values of nearest neighbours. This is done by calculat-

ing the K-th minimum distance between the train and test data points. The distance is Manhattan or Euclidean distance. Minkowski Distance can be defined as the generalised form of Euclidean and Manhattan distance. When p = 1, it is Manhattan distance (sum of absolute differences between points across all the dimensions). When p = 2, it is Euclidean distance (the shortest distance between 2 points).

Effects of various hyper-parameter values:



The optimal values for chosen hyper-parameters via GridSearchCV are n_neighbors = 4 and p = 1.

2.4.2 Decision Tree (DT) Regressor

Decision Tree is a tree-based supervised machine learning algorithm used for classification and regression problems. It is easy to interpret, understand and visualise. Compared to other algorithms, decision trees require minimal data preparation and do not require scaling of the data.

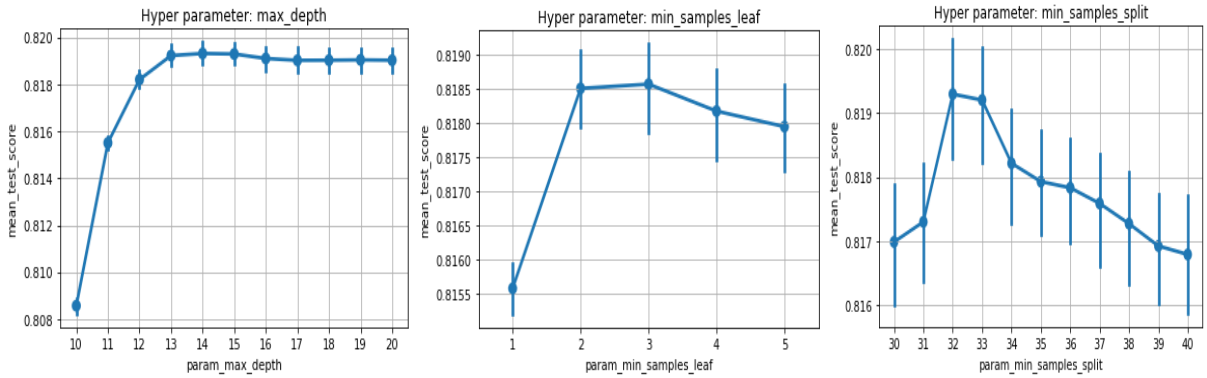
2.4.2.1 Working

It works by transforming the data into a tree structure and making predictions based on a set of binary rules. The tree structure is a simple model with branches, nodes, and leaves. Each binary rule narrows down the possible values until the model is ready to make a single prediction.

2.4.2.2 Hyper-parameter tuning

- max_depth: Represents the maximum depth of the decision trees. As the maximum depth of a tree increases, the model starts to overfit. Hence, to avoid overfitting or underfitting, optimal maximum depth is chosen.
- min_samples_split: Represents the minimum number of samples required to split an internal node.
- min_samples_leaf: Represents the minimum number of samples required to be a leaf node.

Effects of various hyper-parameter values:



Since the range of values and the number of hyper-parameters are high, RandomSearchCV was used to find the optimal range of hyper-parameter values. The optimal values for chosen hyper-parameters via GridSearchCV are max_depth = 14, min_samples_leaf = 3, min_samples_split = 32.

3 Results

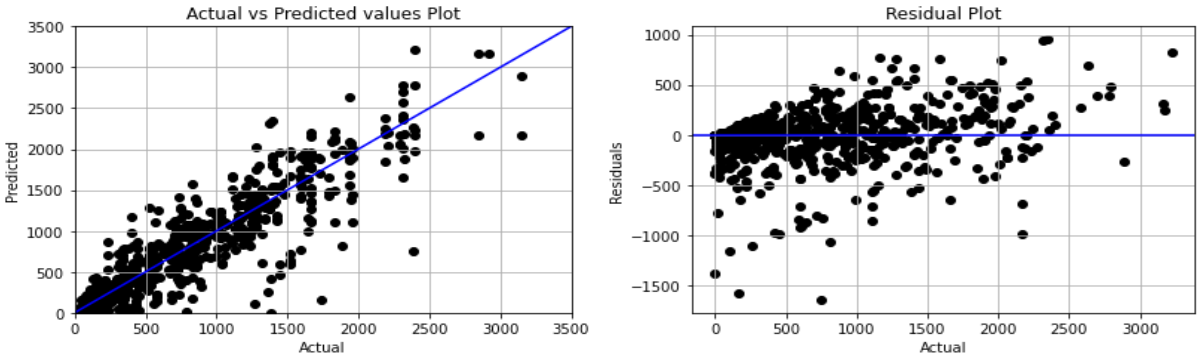
Results obtained across all models are summarised in the table below:

	Model	Train R2	R2	RMSE	Computation_Time
4	DT Regressor Tuned	0.909788	0.828131	274.534826	0.046901
2	KNN Regressor Tuned	0.900832	0.826153	276.109430	1.065586
0	KNN Regressor	0.877772	0.809068	289.359522	1.106256
3	DT Regressor	1.000000	0.794028	300.540235	0.049877
1	Dummy Model	0.000000	-0.001328	662.653312	0.000000

After tuning the hyperparameters, the RMSE decreased, i.e., the models performed well. The tuned models also performed better than the baseline model (dummy regressor) by a large margin. The unseen test set can now be evaluated with the tuned decision tree model as it has produced the best results with least the RMSE.

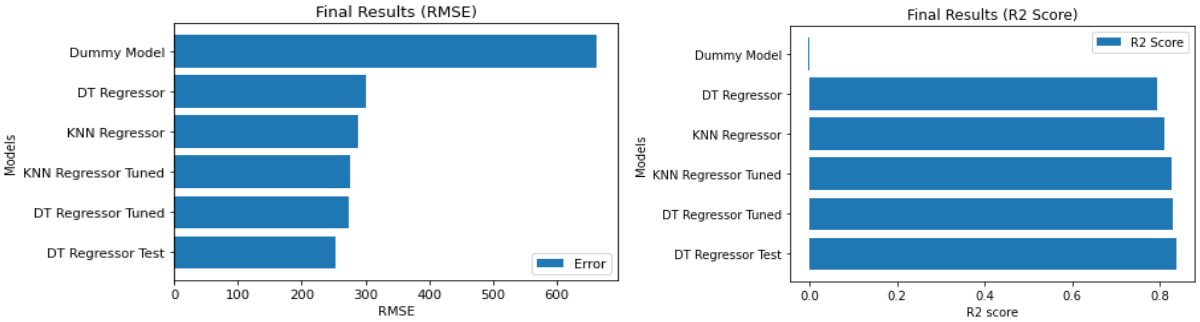
3.1 Evaluation on the test set

Results of the tuned decision tree model's performance on the test set are given below:



The data points and residuals are close to regression lines. From both the plots, it is clear that the model is generalising and performing well on the test set.

Comparison with all models:



The tuned (Decision Tree) DT Regressor model on the test set has performed the best among all other models. This implies that that model is able to generalise well to unseen data.

3.2 Conclusion

As part of this assessment, a thorough analysis of the KNN regressor algorithm and the Decision Tree regressor algorithm has been done on the given dataset. The models' working has been clearly explained, along with the impact of their hyperparameters. Observations were made on how the performance of the models varied with the change in hyperparameters. The performance of the models increased by tuning the hyperparameters.

Note: The attached jupyter notebook file contains references and further information on the problem statement.