Predicting Number of Bikes Rented using AI Models

1st Garvit Mathur
2072624
University of Bristol
Bristol, UK
qo21715@bristol.ac.uk

I. Introduction

This project aims to develop an AI-based model to predict the number of bikes rented by the hour given the required parameters. To develop a model rented bike dataset has been used, which contains a total of 13 features including the rented bike count. Fig 1 provides a snippet of how the dataset looks. The dataset comprises a combination of continuous datatype features (such as *Rented_Bike_Count, Hour, Temperature(°C), Humidity(%), Wind speed (m/s), Visibility (10m), Dew point temperature(°C), Solar Radiation (MJ/m2), Rainfall(mm), Snowfall (cm)*) and object datatype features (such as *Date Seasons, Holiday, Functioning Day*). And considering our aim, the rented_bike_count is the feature we seek to predict given other features, and hence is referred to as output sometimes in this report.

Before developing a model it is essential to classify

	Date	Rented_Bike_Count	Hour	Temperature(°C)	Humidity(%)	Wind speed (m/s)	Visibility (10m)	Dew point Temperature(°C)	Solar Radiation (MJ/m2)	Rainfall(mm)	Snowfall (cm)	Seasons	Holiday	Functioning Day
0	01/12/2017	254	0	-5.2	37	2.2	2000	-17.6	0.00	0.0	0.0	Winter	No Holiday	Yes
1	01/12/2017	204	1	-5.5	38	0.8	2000	-17.6	0.00	0.0	0.0	Winter	No Holiday	Yes
2	01/12/2017	173	2	-6.0	39	1.0	2000	-17.7	0.00	0.0	0.0	Winter	No Holiday	Yes
3	01/12/2017	107	3	-6.2	40	0.9	2000	-17.6	0.00	0.0	0.0	Winter	No Holiday	Yes
4	01/12/2017	78	4	-6.0	36	2.3	2000	-18.6	0.00	0.0	0.0	Winter	No Holiday	Yes
5	01/12/2017	100	5	-6.4	37	1.5	2000	-18.7	0.00	0.0	0.0	Winter	No Holiday	Yes

Fig. 1: Data set Snippet

the kind of algorithm required to solve the problem. The output variable provided here contains a continuous set of values ranging from 0 to 3556 and comprises of total 2166 unique values, making it a supervised regression problem instead of a unsupervised clustering or a sup classification problem which performs on a limited set of classes. To solve the model we will be using two distinct AI models i.e. a Multilayer Perceptron (MLP) Model and a Random Forest Regressor Model, and compare these models with a baseline model i.e. a Linear Regression model. The scikit-learn library is used to define mentioned models and to check their performances.

The following section will talk about the methodology used in developing the model. It will include the data engineering performed to prepare the data for training, data visualisation methods used to study the effects of variables, then discuss the model's implementation and, at the end discuss the hyperparameter tuning performed to improve the performance of the model. And after that will discuss the results in the last section.

II. Methodology

A. Pre-Processing

Data preprocessing is an essential stage in Machine Learning since the quality of data and the relevant information that can be extracted from it directly influence our model's capacity to learn. To begin with, the Date feature provided does not provide much of any information in the scenario but knowing the specific month could be helpful, so we split Date and kept the month as a category and drop the rest. Then another essential step is to check for any NA value in a given dataset, it could affect the training and hence any NA value row should be dropped. In the given dataset there was no NA value and hence 0 rows were dropped.

Next, we convert all the object datatype features into categorical data types and perform one-hot encoding. One-hot encoding One hot encoding is a process of converting categorical data variables to label integers so they can be provided to machine learning algorithms to improve predictions. The categories we encoded in the dataset are mentioned below:

- Season - Winter as 0; Spring as 1; Summer as 2; and Autumn as 3
- Holiday - No holiday as 0; and Holiday as 1
- Functioning Day - No Functioning Day as 0; and Functioning Day as 1

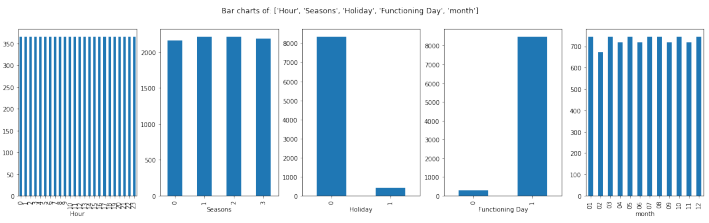


Fig. 2: Analysing Categorical Data

Further, we studied the continuous features and categorical features separately to better understand the quality of data provided. Figure 2 and figure 3 depict the spread of data points

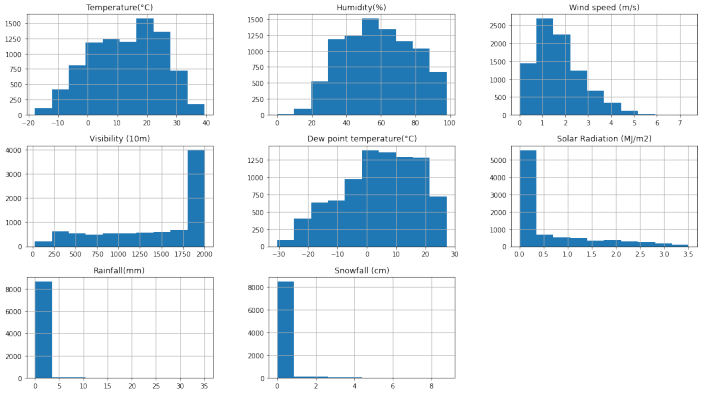


Fig. 3: Analysing Continuous Data

for both the categories for all the features. From the figures, we can say that features such as "Holiday", "Functioning Day", "Visibility", "Solar Radiation", "Rainfall", and "Snowfall" contain skewed values and hence do not help considerably in providing a generalise result for our model. To further examine the continuous features, Fig 4 shows the spread of data points. We can observe from the figure that apart from the temperature and dew point temperature, there is no visible trend in any other feature data points. This lack of trend affects the performance of the model.

Also from the covariance matrix between the categorical features, fig 5, one can say that the temperature feature is most correlated with the output variable and the rest not so much. Similarly, figure 6 shows the spread of datapoints for categorical features. It is evident from figure that most of the categorical features are spread uniformly except "Functioning Day".

Finally, standardisation was performed on the dataset using the StandardScalar module from sklearn.preprocessing library to bring every feature to a comparable scale. And the dataset was split into two separate sets for training and testing purposes using the train_test_split module from sklearn.model_selection. The split size kept here was 0.15 (i.e 85% training and rest testing) because for lower values the test set dimension was lower than 1000, which is insufficient for judging the performance of the model, and larger values reduce the training set dimension which could affect the performance of the model.

B. Model Training

1) Baseline - Linear Regression: Linear Regression is a well known and most basic regression model and is used as a baseline model here. Linear Regression assumes a linear relationship between the input variables and the dependent variables. The model used here is imported from LinearRegression from sklearn.linear_model library.

The R^2_Square metrics have been selected to compare and evaluate all the models in this study. In regression, the

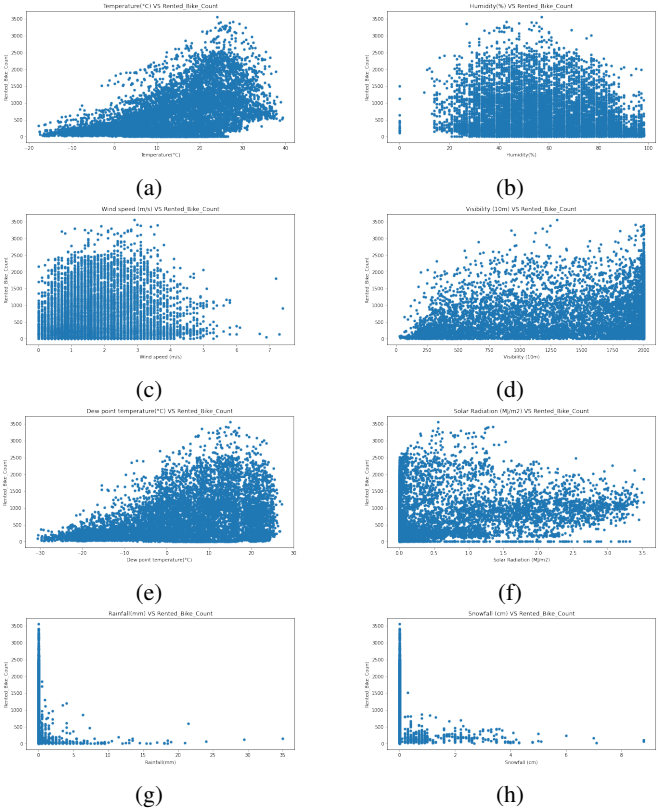


Fig. 4: Relationship exploration: Continuous Vs Continuous – Scatter Charts a) Temperature vs Rented_Bike_Count b) Humidity vs Rented_Bike_Count c) Wind_Speed vs Rented_Bike_Count d) Visibility vs Rented_Bike_Count e) Dew_Point_Temperature vs Rented_Bike_Count f) Solar_Radiation vs Rented_Bike_Count g) Rainfall vs Rented_Bike_Count h) Snowfall vs Rented_Bike_Count

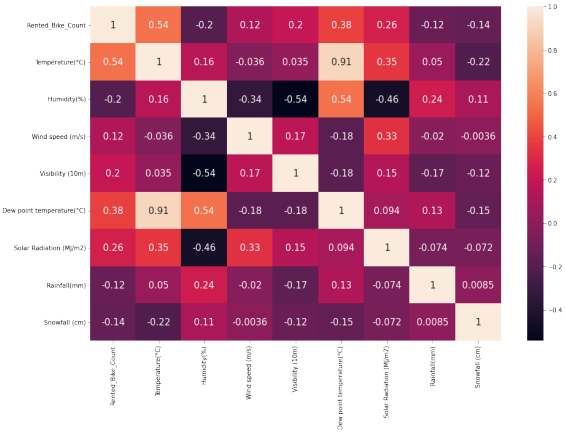


Fig. 5: Covariance Heat map of Continuous Features

R^2 coefficient of determination is a statistical measure of how well the regression predictions approximate the data points. An R^2 of 1 indicates that the regression predictions perfectly fit the data. Additionally, Root Mean Square Error (RMSE) has been evaluated for all models for comparison,

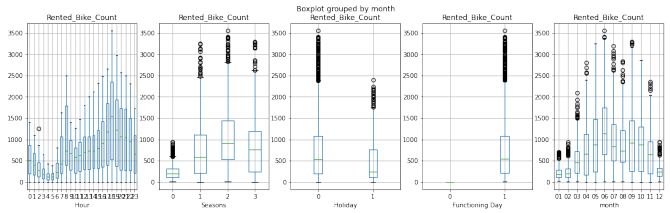


Fig. 6: Relationship exploration: Categorical Vs Continuous – Box Plots

as it provides the basic idea of how good the model predicts the value. The lower the RMSE, the better the model performance.

2) Multi-Layer Perceptron: A multilayer perceptron (MLP) is a type of feedforward artificial neural network (ANN). For training, MLP employs a supervised learning approach known as backpropagation. MLP is distinguished from linear perceptrons by its multiple layers and non-linear activation functions. It can identify data that cannot be separated linearly. The model has been trained using the MLPRegressor module from sklearn.neural_network library.

3) Random-Forest: Random Forest Regression is a supervised learning algorithm that uses the ensemble learning method for regression. The ensemble learning method is a technique that combines predictions from multiple machine learning algorithms to make a more accurate prediction than a single model. The random forest model has been imported from sklearn.ensemble library.

C. Hyper Parameter Tuning

1) Multi-Layer Perceptron: After defining a model it is important to tune hyperparameters to optimize the performance of the model. MLP contains a large number of hyperparameters and optimizing all using a Grid Search could be an expensive process. So to reduce the time expense, six hyperparameters were chosen to study their effect on the performance. The six hyperparameters were, ”hidden_layer_sizes” which defines the architecture of the neural network, ”activation function” which introduces the non-linearity to the model, ”solver” for that we choose sgd and adam as they are the most famous gradient descent optimizer solvers, ”alpha” which is a regularization term and combats overfitting, ”learning_rate” and ”learning_rate_init” which affects the rate of convergence oh the model. The table I summarizes the parameters values provided for grid search.

Fig 7 summarizes the result of the grid search on the

hidden_layer_sizes	(64,32,16,8), (32,16,16,8)
activation	tanh, relu
solver	sgd, adam
alpha	0.0001, 0.05
learning_rate	constant, adaptive
learning_rate_init	0.001, 0.0001

TABLE I: MLP Hyperparameter for Grid Search

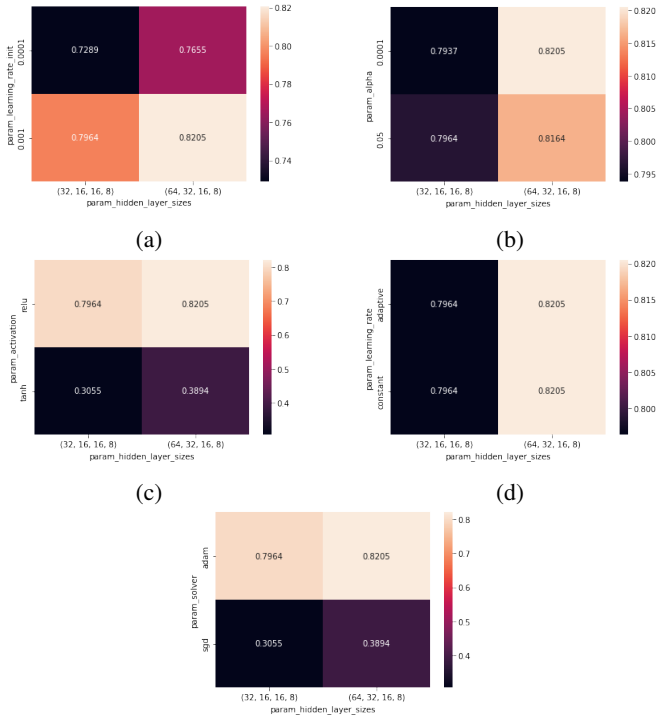


Fig. 7: Maximum R2 score on hidden_layer_size vs a) learning_rate_init b) alpha c) activation_function d) learning_rate e) solver

above-mentioned parameters. The grid search results suggests based on the given parameters search values **activation: ’relu’, alpha: 0.0001, hidden_layer_sizes: (64, 32, 16, 8), learning_rate: constant, learning_rate_init: 0.001, solver: ’adam’** is the most optimal selection of parameters for the model.

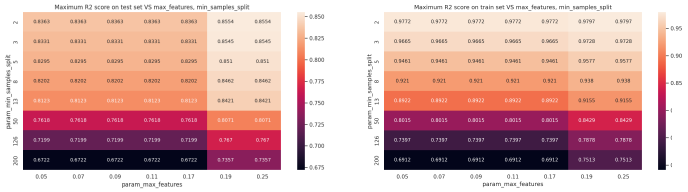
2) Random-Forest: The performance of the random forest largely depends on the number of trees in the selected forest i.e. n_estimators, a maximum number of features considered for splitting a node i.e. max_feature, and min number of data points placed in a node before the node is split i.e. min_samples_split. There are a few more hyperparameters linked to the random forest but the above-mentioned plays a significant role in impacting the performance of the model and hence these three are optimized here. The table II summarizes the parameters values given for grid search.

Fig 8 - 10 shows the results of the grid search

n_estimators	10, 18, 33, 60, 110, 200,250
max_features	0.05, 0.07, 0.09, 0.11, 0.17, 0.19, 0.25
min_samples_split	2, 3, 5, 8, 13, 50, 126, 200

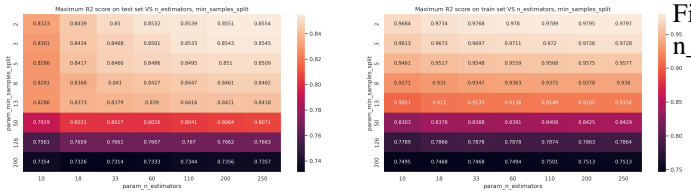
TABLE II: Random Forest Hyperparameter for Grid Search

for a combination of parameters. Based on the results **max_features: 0.19, min_samples_split: 2, n_estimators: 250** is the most optimized combination of the hyperparameters.



(a) (b)

Fig. 8: Maximum R2 score on a) Test and b) train set VS max_features, min_samples_split



(a) (b)

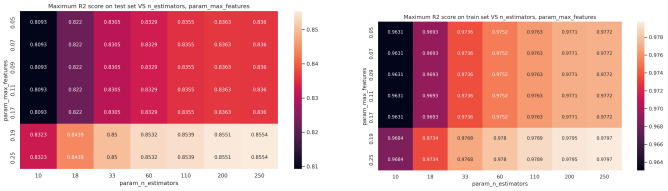
Fig. 9: Maximum R2 score on a) Test and b) train set VS n_estimators, min_samples_split

III. Results

Results	Baseline	MLP	Random Forest
R^2 Train	0.53	0.87	0.98
R^2 Test	0.54	0.84	0.86
RMSE Train	442.37	218.59	88.78
RMSE Test	472.26	239.09	232.65

TABLE III: Performance Comparison

To compute the results the common matrices were calculated for all models. Also to remove biases from the training Cross-Validation Fold = 4 was implemented which refers to the number of groups that a given data sample is to be split into and pick one by one for training and testing purposes. Table III summarizes the performance matrices results of all the three models for the training and testing set after tuning the hyperparameters of the model as discussed below. The better testing performance indicates the generalisation of the model for unseen data points. Additionally, figure 11 and 12 shows the comparison of predicted dataset and test dataset for both the MLP and random forests. The results conclude that Random Forest provides much better performance compared to our baseline approach and slightly better than the MLP method. The performance of R^2 greater than 0.8 in the test set makes the model reliable to use in the real scenario.



(a) (b)

Fig. 10: Maximum R2 score on a) Test and b) train set VS n_estimators, param_max_features

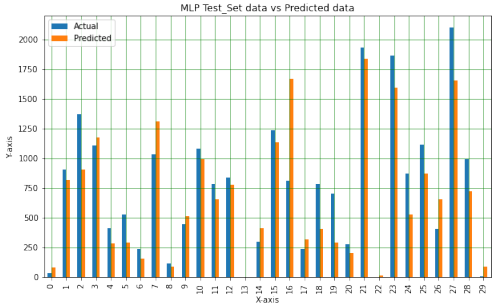


Fig. 11: MLP Test Set Data vs Predicted Data

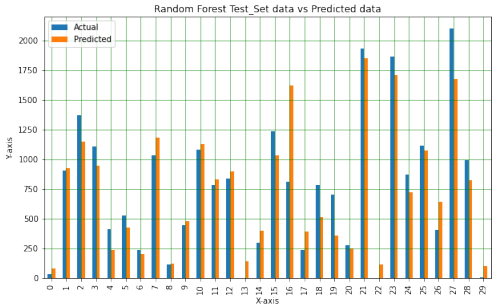


Fig. 12: Random Forest Test Set Data vs Predicted Data