Bike Sharing Demand Prediction

Vidit Ghelani,
Mahesh Lakum
Data Science Trainees,
AlmaBetter, Bangalore

Abstract:

When it comes to transportation and hiking business, there are a number of parameters that affect the business. A lot can be determined from the yearly data of bike renting of a certain region.

Our experiment aims to predict the demand of bike sharing in the best possible way, using Machine Learning Algorithms.

Keywords: EDA (Exploratory Data Analysis), Regression, Model Training, Machine Learning Algorithms.

1. Problem Statement

Currently Rental bikes are introduced in many urban cities for the enhancement of mobility comfort. It is important to make the rental bike available and accessible to the public at the right time as it lessens the waiting time. Eventually, providing the city with a stable supply of rental bikes becomes a major concern. The crucial part is the prediction of bike count required at each hour for the stable supply of rental bikes.

2. Introduction

The company keeping track of these bike bookings can alter the availability of the bikes based on basic observations of demand with respect to the season, hour of day, weekday/ weekend, etc parameters. However, with the help of Machine Learning Algorithms it's easy to understand the relevance/ importance of each feature in the data.

3. Understanding the features

The raw data contains various features mentioned in columns. Those features are as below:

1. Date: The date of the day.
   Value ranges from 01/12/2017 to 30/11/2018. Date Format: is DD/MM/YYYY.
   Note: We need to convert this into datetime format. dtype: str.
2. Rented Bike Count: Number of rented bikes per hour which is our dependent variable and we need to predict it. dtype: int
3. Hour: The hour of the day, starting from 0-23 it's in a digital time format. We need to convert it into category data type. dtype: int
4. Temperature (°C): Temperature in Celsius. dtype: Float
5. Humidity(%): Humidity in the air in %. dtype: int
6. Wind speed (m/s): Speed of the wind in m/s. dtype : Float
7. Visibility (10m): Visibility in m. dtype: int
8. Dew point temperature (°C): Temperature at the beginning of the day. dtype : Float
9. Solar Radiation (MJ/m2): Sun contribution. dtype: Float
10. Rainfall (mm): Amount of rain in mm. dtype: Float
11. Snowfall (cm): Amount of snow in cm. dtype: Float
12. Seasons: Season of the year. There are 4 seasons in this data. dtype: str
13. Holiday: If the day is a holiday period or not. dtype: str

14. Functioning Day: If the day is a Functioning Day or not, dtype: str

4. Approaching the Problem

The entire process to analyze the data, find out some useful conclusion was done in four steps:

4.1 Pre- Processing
4.2 Performing exploratory data analysis (EDA)
4.3 Training the Models
4.4 Identify the best model and conclude

4.1 Pre- Processing
4.1.0. View the data
4.1.0.1. Understand the Features
4.1.1. Inspect the data
Here first we viewed the information of data with the '.info()' command. However, isnull().sum() serves to be more appealing. Hence, we used it to spot the null values.

1. Inspecting the data for duplicate values and null values
2. Get the basics statistics for each feature. (i.e. use df.describe())

4.1.2. Clean the data
We will create a new dataframe and deal with the null values by replacing it with appropriate values or may drop them. This way we do not make any changes to the raw data.

1. Remove duplicate values (here none)
2. Replace null values (here none)
3. Remove irrelevant data (here none)
4. Dealing with the Outliers
5. Changing the data types of features where necessary.
6. Add derived features (changed the date feature to 'day', 'month', 'year' and then used the same data to get the 'weekend' column.)
7. Rename the columns.

Here, we observed that Date, Seasons, Holiday and Functioning Day are categorical variables.

4.2 Performing exploratory data analysis (EDA)
4.2.1 Summary of Observations
Here, the data was analyzed for various features like 'month', 'weekend', 'hour', 'functioning day', 'seasons' and 'holiday'.

1. Month: From the fifth month to tenth month the demand for the rented bike is high as compared to other months. These months correspond to the summer season.
2. Weekend:
   The week days which represent in blue color show that the demand of the bike is higher because of the office hours. Peak time are 7 am to 9 am and 5 pm to 7 pm. The orange color represent the weekend days, and it show that the demand of rented bikes are very low especially in the morning hour but when the evening start from 4 pm to 8 pm the demand slightly increases.
3. Hour: Generally people use rented bikes during their working hours from 7am to 9am and 5pm to 7pm.
4. Functioning Day: The use of rented bikes in functioning days or not, and it clearly shows that people don't use rented bikes on no functioning day.
5. Seasons: In the summer season the use of rented bikes is high and peak time is 7am to 9am and 7pm to 5pm. While in winter it may be low because of snowfall.
6. Holiday: The use of a rented bike on a holiday, and it clearly shows that on holiday people use the rented bike from 2pm to 8pm.

4.2.2 Normalizing the data
Next we do the regression plot for the numerical data and find some pattern in the data. Here we see that the feature

‘rented_bike_count’ is left skewed and hence it needs normalizing. We use square root transformation to achieve this.

4.2.3 Checking correlation

Lastly in this step we check for the correlation between the features of the data using the OLS (Ordinary Least Square) and Heatmap.

From the OLS model we observe that the features 'Temperature' and 'Dew_point_temperature' are highly correlated so we need to drop one of them. In order to drop them we check the (P>|t|) value from the table and we can see that the 'Dew_point_temperature' value is higher so we need to drop the ’Dew_point_temperature’ column.

From the heatmap we observe that on the target variable line the most positively correlated variables to the rented bike count are: ‘Temperature’, ‘dew point temperature’ and ‘solar radiation’. While the most negatively correlated variables are ‘Humidity’, ‘Rainfall’ and ‘Snowfall’.

The positive correlation between columns 'Temperature' and 'Dew point temperature' is 0.91. So even if we drop this column it does not affect the outcome of our analysis. Also they have the same variations, so we can drop the column 'Dew point temperature'.

4.3 Training the Models

Here, first we split the data for training and testing. We split the data into 75 : 25 ratio.

Next we train the model using the train data set and then test its performance on the test data. We undertake this process for various models. Based on that we will select the best suited model for our prediction.

The models used here are:

- Linear regression
- Lasso regression
- Ridge regression
- Elastic net regression
- Decision Tree regression
- Random forest regression
- Gradient Boosting regression

4.4 Identify the best model and conclude

Here, we find the errors or say the accuracy of each model. This is done under different matrices like MAE (Mean Absolute Error), MSE (Mean Squared Error), RMSE (Root Mean Squared Error), R2 score and Adjusted R2 for better understanding.

This is done for all the six models and based on this we identify the best suited model.

		Model	MAE	MSE	RMSE	R2_score	Adjusted R2
Training set	**0**	Linear regression	4.474	35.078	5.923	0.772	0.77
	1	Lasso regression	7.255	91.594	9.570	0.405	0.39
	2	Ridge regression	4.474	35.078	5.923	0.772	0.77
	3	Elastic net regression	5.792	57.574	7.588	0.626	0.62
	4	Dicision tree regression	5.477	54.194	7.362	0.648	0.64
	5	Random forest regression	0.802	1.593	1.262	0.990	0.99
	6	Gradient boosting regression	3.269	18.648	4.318	0.879	0.88
	7	Gradient Boosting gridsearchcv	1.849	7.455	2.730	0.952	0.95
Test set	**0**	Linear regression	4.410	33.275	5.768	0.789	0.78
	1	Lasso regression	7.456	96.775	9.837	0.387	0.37
	2	Ridge regression	4.410	33.277	5.769	0.789	0.78
	3	Elastic net regression Test	5.874	59.451	7.710	0.624	0.62
	4	Dicision tree regression	5.703	58.675	7.660	0.629	0.62
	5	Random forest regression	2.222	12.878	3.589	0.918	0.92
	6	Gradient boosting regression	3.493	21.289	4.614	0.865	0.86
	7	Gradient Boosting gridsearchcv	2.401	12.393	3.520	0.922	0.92

From the above table we see that Random Forest and Gradient Boosting have excellent performance. The performance of the Gradient Boosting model can still be improved by tuning a few hyper parameters.

We use Grid Search CV for this and create the eighth model with improved performance. However, this performance is still not superior to the Random Forest Model.

5. Challenges Faced:

Few of the challenges we faced were:

1. Need to plot a lot of graphs to analyze the data.
2. Feature selection and feature engineering.

3. Hyperparameter selection for optimizing the model.

6. Conclusion:

From this exercise we can conclude that:

1. No overfitting is seen.
2. Random forest Regressor and Gradient Boosting Grid Search CV gives the highest R2 score of 99% and 95% respectively for Train Set and 92% for th e Test set.
3. Upon applying GridSearchCV to Gradient Boosting, we observed that it resulted in a significant improvement in the performance of that model. However, it is not better than that of the Random Forest model.
4. Feature Importance value for Random F orest and Gradient Boost are different. Hence, we can choose either of these models for our prediction.

References:

1. GeekforGeeks
2. Stackoverflow
3. Python guide
4. Matplotlib guide
5. Seaborn guide
6. Scikit learn guide
7. Github
8. Kaggle