UNIVERSITÀ DI PISA

Artificial Intelligence and Data Engineering
Business and Project Management

Bike Rental Demand Forecaster

Project Documentation

Team Members:
Morucci Edoardo
Nello Enrico
Pierucci Matteo

Academic Year 2021-2022

GitHub Project: https://github.com/enricollen/bike-rental-demand-prediction

Summary

1 Introduction

Demand forecasting refers to the process of making estimations about future customer demand over a certain period. It uses historical data along with other information.

When demand forecasting is correctly implemented, businesses have valuable information about their potential in the current market as well as other markets so managers can make informed decisions about business growth strategies, pricing, and market potential.

Failing to use demand forecasting puts businesses at risk for making poor decisions about their target markets and services. These ill-informed decisions can have effects on supply chain management, inventory holding cost, and ultimately profitability.

Demand forecasts are then essential for any business to manage its different business activities but at the same time are difficult to do with good accuracy using traditional techniques. Huge volumes of data are usually generated by companies, and we can exploit them working on historical data to derive meaningful insights and provide better data-driven decisions.

Our project consists in providing AI-based logistics optimizations, the idea is to provide a system to support businesses in the field of rental, specifically for bike outdoor rental, but could be easily adapted also to other type of outdoor rental services like electric scooters, cars, or other means of transportation.

We will exploit regressors that learn from time series of past rental data and provide a prediction regarding demand for the days ahead. There is a huge profit to be made in this domain and hence accurate demand forecast can contribute to better planning, production, pricing, service to customer and other decision-making activities for the business.

Bike Rental is driven by external factors like different weather conditions, different calendar variables, etc. The project presents a study of different variables and their effect on the rental behavior of bikes.
Some determinants are mainly related to:

- weather (uncertain weather, rain, wind, humidity etc.)
- day of the week (working, holiday, weekend etc.)
- times within the day (night, day etc.).

In the following list we propose some use cases of our work:

- To provide a supporting tool to be able to estimate with a certain degree of probability the best days or hours for routine maintenance
- For dynamic provisioning of vehicles in case of periods with higher demands
- Intelligent distribution upon different rental locations
- References for all those rental startups which want to have some useful estimation for their investments.

So, the project aims to showcase better demand forecasting in an emerging bike sharing and Public Vehicle Rental Space. Two different forecasting models, Random Forest regressor and Artificial Neural Network are built, experimentally investigated, and then practicality discussed. The results will be used to investigate consumer behavior on demand forecasting and the overall impact on the business as a whole.

2 Dataset

The dataset is publicly available from Kaggle that is the primary source for our project. We found the '***Bike_Demand_Data***' dataset made available from Capital Bikeshare system, Washington D.C., USA and the corresponding weather and seasonal information is sourced from **freemeteo** website.

The core data set consists of two-year historical log corresponding to years 2011 and 2012 of the rental and weather records. The data set contains a total of 13 variables describing important factors like seasons, weather, etc. and we have total 10886 observations/records. According to some papers we found on the same argument, it turned out that the amount of data (two years) is sufficient to apply different techniques successfully.

The dataset consists of both categorical and numerical (also known as quantitative data) data types. **'Rentals'** is the dependent/target variable and the rest independent variables.

2.1 Data Attributes

The dataset is provided with hourly rental data spanning two years.
We are predicting the total count of bikes rented for each hour.
The dataset contains the following attributes:

- **datetime** – hourly date (mm/dd/yy) + timestamp (0 to 23 hr).
  It is a record index, time-series in nature containing the date (day, month, year) along with the hourly time. It is used as indexing feature in the python code.
- **hour** – hourly data extracted from datetime (0 to 23 hr)
  Numerical data type containing the hourly time for each record. Ranges from 0 (12:00 AM) to 23 (11:00 PM).
- **season** – seasonal data of four different seasons Categorical data type containing the information of current season for respective observation / record. Four types of seasons are used.
  They are:
    - 1 = Spring
    - 2= summer
    - 3=fall
    - 4=winter
- **holiday** – if the particular day is holiday
  Binary data type (Yes/No, 1/0). Tells if particular day is holiday or not.
    - 1=Holiday
    - 0=No Holiday
- **workingday** – whether the day is neither a weekend nor a holiday
  Binary data type (Yes/No, 1/0). Tells whether the day is neither a weekend nor a holiday.
    - 1= if day is neither weekend nor holiday
    - 0=otherwise
- **weather** – weather data of 4 different types of weather
  Categorical data type. Tells what weather condition is observed on a particular day. Four different types of weather conditions are considered. They are:
    - 1=Clear, Few clouds, Partly cloudy
    - 2= Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist
    - 3= Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds
    - 4= Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog
- **temp** – actual hourly temperature data
  Numerical data type. The temperature is in Celsius and the actual observed temperature, i.e., the physical amount of heat within a substance as measured by a thermometer.
- **atemp** – 'feels like' hourly temperature data

Numerical data type. This temperature is what we call as feels like temperature and is also measured in Celsius, i.e., specifically relating to when its values are greater than the actual temperature, is a measure of how hot it really feels for a human.

- **humidity** – relative humidity data
  Numerical data type. Relative humidity measures the amount of water in the air in relation to the maximum amount of water vapor (moisture). The higher the temperature, the more water vapor the air can hold.
- **windspeed** – hourly speed of wind
  Numerical data type. Tells us about the wind speed at a particular hour.
- **casual** – number of non-registered user rentals initiated
  Numerical data type. Non-registered users are those who have no account or any direct connection to the rental company. They may be viewed as guest users and does not talk about a particular user frequency. Example: Tourist, occasionally user, first time users, etc.
- **registered** – number of registered user rentals initiated
  Numerical data type. Registered users are those who have an established connection with the rental company, for example registered via an application (app), etc. Such users can be studied as their data is recorded on much higher level compared to the non-registered once.
- **rentals** – number of total rentals
  Numerical data type. This is our TARGET Variable, which we want to predict based on the above-mentioned independent variables or factors. Rentals is the total number of bikes rented or Bike users and is simply the addition of casual and registered variables.

3 Data Analysis & Manipulation

We analyzed the dataset to detect the presence of outliers and correlation between the features. Then we performed several techniques for data cleaning, preprocessing and outlier removal.
For sake of clarity, please refer to Ch. 6 Appendix, where every step is explained in detail.

4 Predictive Models

We performed a preliminary study on the state-of-the-art in prediction modeling, for both determining which kind of research have been already done and for establishing the most suitable models to work with. Also, we seek some references about the minimum size needed for the dataset.
We ended up choosing two of the most used AI approaches:

1. **Neural Network – MLP**
2. **Random Forest Regressor**

Using as references the indication inside the following papers:

- Knowledge Adaption for Demand Prediction based on Multi-task Memory Neural Network (2020, Li, Bao, Liu, Waller, Yao) [1]
- Utilizing Artificial Neural Networks to Predict Demand for Weather-sensitive Products at Retail Stores (2017, Taghizadeh) [2]
- Machine-Learning Models for Sales Time Series Forecasting (2019, Pavlyshenko) [3]

The first paper we choose, from September 2020, studies of a time series to make demand prediction for means of transport (bus, ferry etc.). regarding the dataset (chapter 3.1), they used one consisting of April-May-June hourly grain (ours is 2 years hourly grain). They used an ad hoc neural network to solve the problem. It is an interesting paper because it proposes to make predictions on request of means of transport, while we do it by means of transport bicycle for hire, so similar real-life scenarios.

The second choice is about a 2017 paper that deals with the forecasting of demand for weather sensitive products (umbrellas, and other 100 products. They used a dataset from 2012 to 2014 (ch.1) but daily grain (while our is 2 years hourly grain). Regarding the modeling choice, they used neural network as in first paper, specifically MLP with parameter tuning.

The third shows that the forecasting demand prediction can be considered as a time series problem, different models (for example ARIMA and SARIMA) have been developed to handle this type of problem, however regression approaches can often give better results compared to time series methods.
The results of predictions using different approaches like ARIMA, Lasso, Random Forest and Neural Network shows that the last two models have best performances.

4.1 MLP - Multilayer Perceptron

Multi-layer Perceptron (MLP) is a supervised learning algorithm that can learn a non-linear function approximator for classification or regression. Given a set of features $X = x_1, x_2, \ldots, x_m$ and a target y it can learn to associate specific configuration of input features to the target variable.
It is a multilayer approach because between the input and output layers there is one or more non-linear layers, called hidden layers.

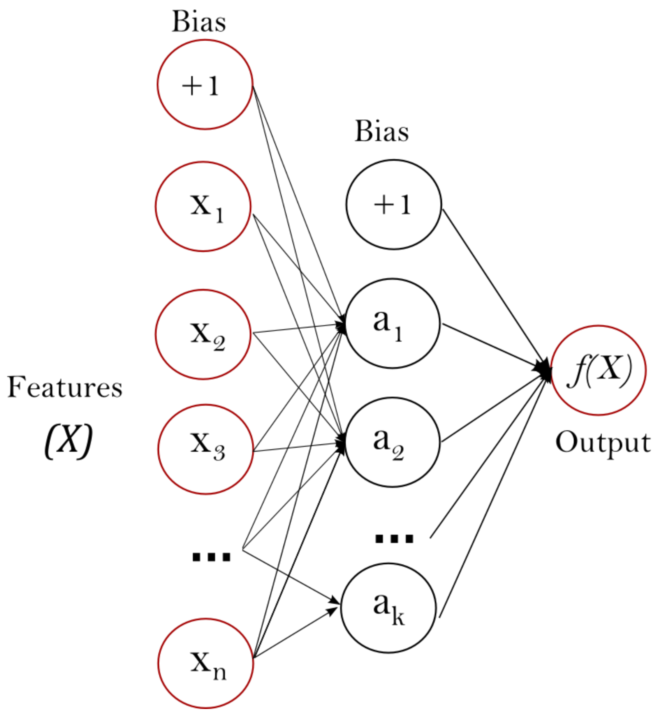


Figure 1 MLP with one hidden layer and scalar output

The Input layer consists of a set of neurons $\{x_i | x_1, x_2, \ldots, x_m\}$ representing the input features, each neuron in the hidden layer transforms the values from the previous layer with a weighted linear summation $w_1x_1 + w_2x_2 + \ldots + w_mx_m$ followed by a non-linear activation function $g(\cdot): R \rightarrow R$ like the hyperbolic tangent, rectified linear unit (ReLU), logistic Sigmoid functions.
The Output layer receives the values from the last hidden layer and transforms them into output values.

4.1.1 Parameter tuning

We import the preprocessed dataset from the storage with the normalized features and the dummy ones, then we set predictors features: the features used in the neurons of the input layer.
We try different combination of the neural network parameters, that include the choice of activation function, hidden layer size and number, the solver of the convergence method and the learning rate.

```
tuning_parameters = {
    'hidden_layer_sizes': [(24), (24, 12), (24, 12, 6)],
    'activation': ['identity', 'logistic', 'tanh', 'relu'],
    'solver': ['sgd', 'edam', 'lbfgs'],
    'alpha': [0.0001, 0.05],
    'learning_rate': ['constant', 'adaptive', 'invscaling'],
}
```

Then we iterate over the parameter combination to understand which one performs best.

```
reg_model = MLPRegressor (hidden_layer_sizes=hl_size, activation=act,
solver=solv, alpha=a, learning_rate=learn_rate, max_iter=600, random_state=0,
shuffle=False)
```

After this process we select the best parameter combination, that results in:

```
best_parameters = {
    hidden_layer_sizes: (24, 12),
    activation: 'relu',
    solver: 'adam',
    alpha: 0.05,
    learning_rate: 'constant'
}
```

4.1.2 Performance Metrics

In order to understand how well the model is performing we adopt the following metrics:

```
def get_MAE(y_test, predicted_labels):
    return metrics.mean_absolute_error(y_test, predicted_labels)

def get_MSE(y_test, predicted_labels):
    return metrics.mean_squared_error(y_test, predicted_labels)

def get_RMSE(y_test, predicted_labels):
    return np.sqrt(metrics.mean_squared_error(y_test, predicted_labels))

def get_MAPE(y_test, predicted_labels):
    return metrics.mean_absolute_percentage_error(y_test, predicted_labels)

def get_MAXE(y_test, predicted_labels):
    return metrics.max_error(y_test, predicted_labels)
```

Once the model, with a certain parameter configuration, ends its learning stage all the metrics for that instance is computed. After that the instances are sorted by Mean Absolute Percentage Error (MAPE), which is a metric that summarizes the overall accuracy of the model, as it normalizes the error with the target value magnitude. In the end we took the first instance of the model, that is the one with minimum MAPE.

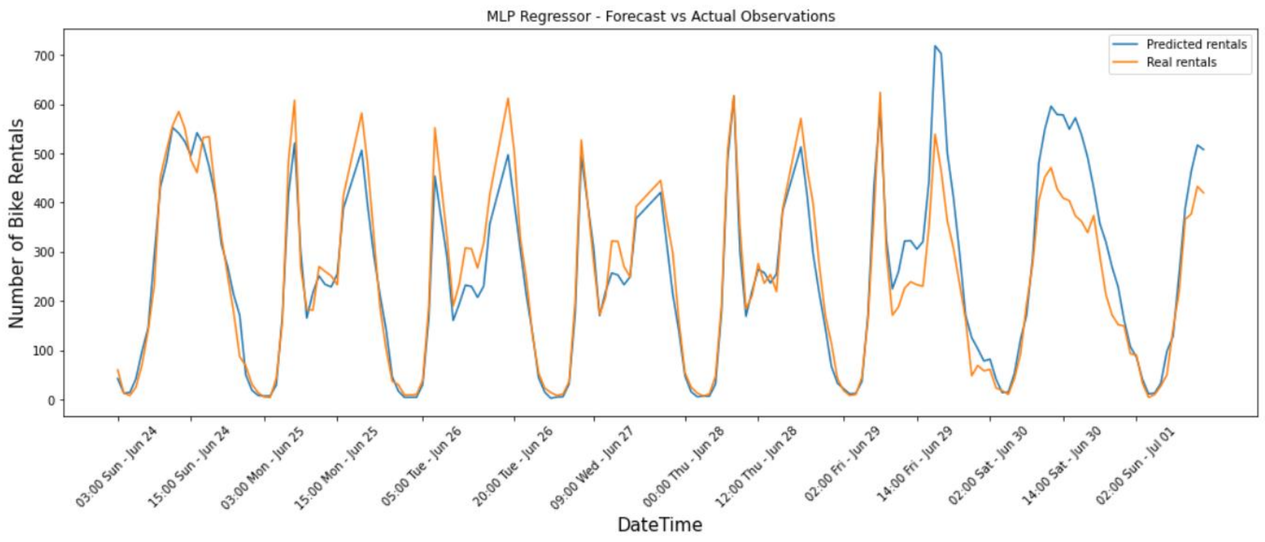
4.1.3 Performance Evaluation

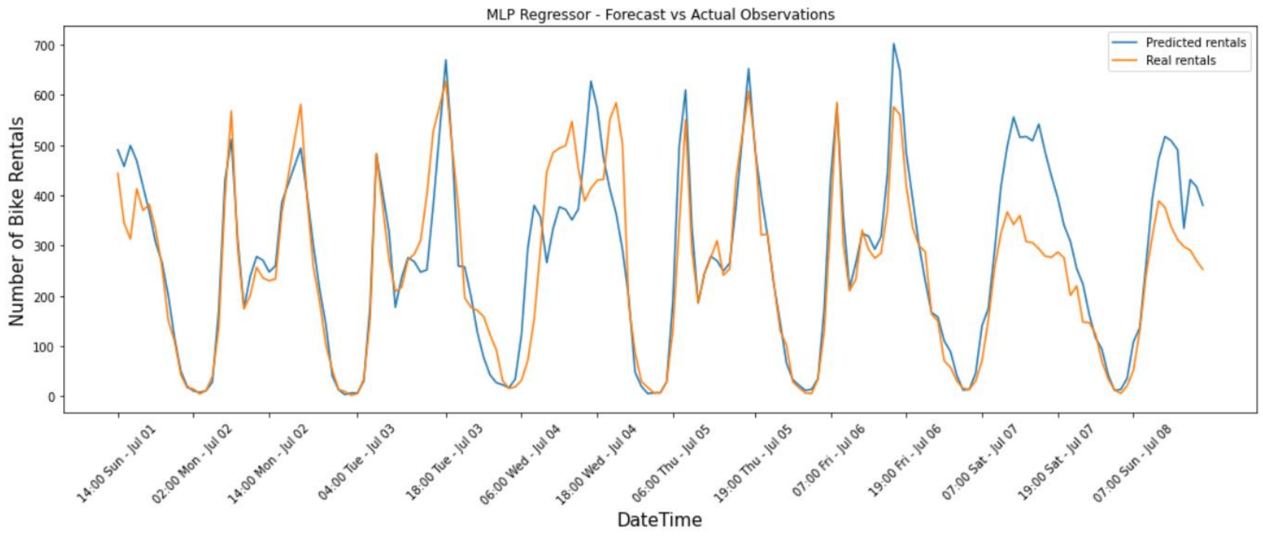
In the following table we summarize the performances of the final model, which is obtained after the parameter tuning:

Mean absolute error	40,9
Mean squared error	4152,6
Root mean squared error	64,4
Mean absolute percentage error	0,39
Max error	460,4

The error rate is quite high, but still acceptable considering the application domain and the not trivial task to resolve.

It is also interesting to visualize the difference between the actual plot of the rentals and the prediction provide by the model with best parameters, over a subset of the test set that covers approximately **two weeks**:





From the plots we can understand that the model is capable of predicting quite accurately the local minima, regarding the night period where the number of rentals is low. On the other hand, local maxima are sometimes badly estimated, in particular the ones during weekends. This inaccurate prediction of peaks could be addressed to the tendency of the model to generalize the behavior during weekends overestimating it.

4.2 Random Forest Regressor

Random Forest is a type of supervised machine learning algorithm based on ensemble learning. Ensemble learning is a type of learning where you join different types of algorithms or same algorithm multiple times to form a more powerful prediction model.
The Random Forest algorithm uses the several times the decision tree algorithm resulting in a forest of trees.
We built two models, in the first one we use default hyperparameters and for the second one we tuned those hyperparameter and then we built the model using them.

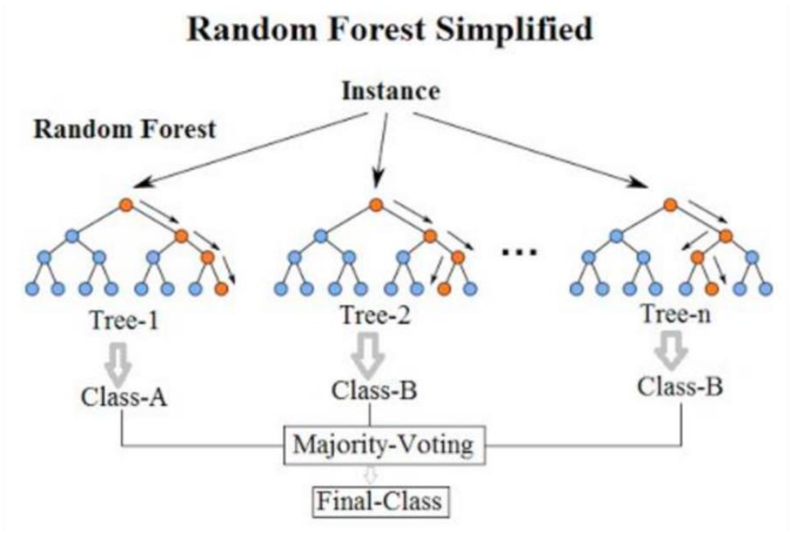


Figure 2: Random Forest Representation

4.2.1 Parameter Tuning

An important aspect of the model is the choice of the hyperparameter, to tune them we used a randomized cross validation with a random grid search and after we made another tuning using some nested for.

```
n_estimators = [x for x in range(10,300,30)]
max_depth = [x for x in range(10,200,30)]
min_samples_split = [2, 5, 10, 20]
min_samples_leaf = [1, 2, 4, 8]
bootstrap = [True, False]
random_grid = {'n_estimators': n_estimators,
               'max_depth': max_depth,
               'min_samples_split': min_samples_split,
               'min_samples_leaf': min_samples_leaf,
               'bootstrap': bootstrap}
```

```
{
    'n_estimators': [10, 40, 70, 100, 130, 160, 190, 220, 250, 280],
    'max_depth': [10, 40, 70, 100, 130, 160, 190],
    'min_samples_split': [2, 5, 10, 20],
    'min_samples_leaf': [1, 2, 4, 8],
    'bootstrap': [True, False]
}
```

The best parameters we obtained are the following ones:

N_estimators	160
Max depth	40

Min sample split	2
Min sample leaf	1
Bootstrap	True

Now we need to train the random forest regressor using the parameter just found.

```
random_forest_opt = RandomForestRegressor(n_estimators=90, min_samples_split=5,
min_samples_leaf=2, max_depth=50 ,random_state = 10, bootstrap=True)
random_forest_opt.fit(X_train, y_train)
```

And then predict rentals using the test set:

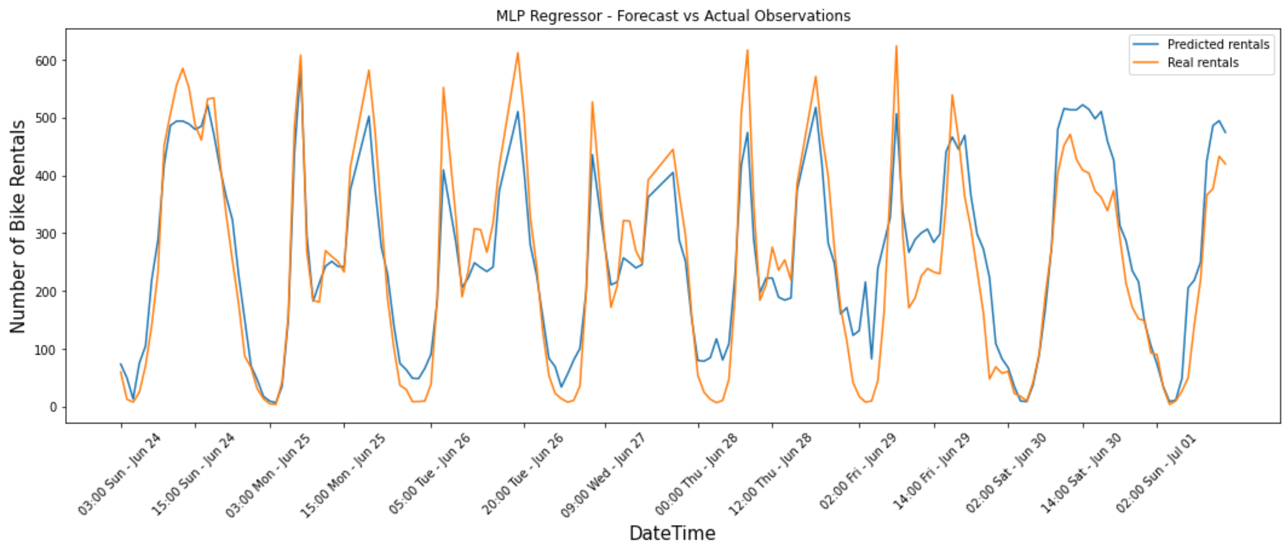
```
y_pred = random_forest_opt.predict(X_test)
```

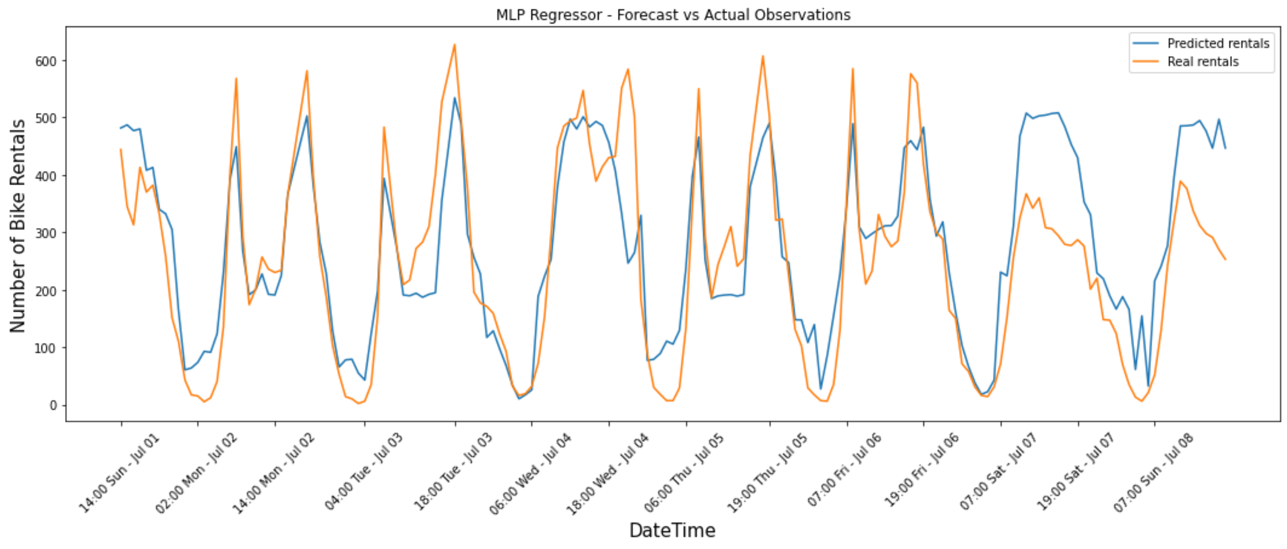
4.2.2 Performance Evaluation

Here we are going to evaluate the performances of the random forest model. We reported only the performances of the model with the hyperparameters tuned because it works better.
In the table below is shown different types of error of our model:

Mean absolute error	51,9
Mean squared error	5643,4
Root mean squared error	75,1
Mean absolute percentage error	0,51
Max error	429

We can see from the table that the error is quite high, this may happen for different reason.
A useful investigation could be seen through some graphs that reports the comparison between real and predicted rents. The graphs below show the first two week of prediction:





We can notice that the minimums of the rentals happen during the night, the model often overestimate those values. On the other hand, the peaks are often underestimating, this behavior may be due to the model that tend to generalize and accurately identify the peaks is not trivial.

5 Conclusions

Comparing the two trained models we can say that the Neural Network (MLP) approach outperforms Random Forest Regressor, in fact:

- MAPE is slightly lower in the MLP prediction evaluated on the whole dataset
- MLP estimates better local minima, even if both the approaches inaccurately predict some peaks

Considering that the demand forecasting task is not trivial, and the results could be improved, the models give an interesting snapshot of the behavior of rentals during the week and how its trend is influenced by the weather conditions. This kind of information are useful to a rental company in many ways:

- If that company has multiple rental shops, it can distribute the total number of bikes, or other means of transport, in order to match the demand in that specific day
- If a company need to optimize the maintenance of its vehicles, it can identify the working hours where the number of rentals is low, avoiding the risk of not serving some customers

Both previous use cases lead to an improved service that matches the needs of the customer, making the company use better its resources and reducing the overall costs.
In fact, some rental companies, in particular the ones in a city context, try to provide a low-cost service to promote the rental as an alternative to the traditional public means of transportation, in such environment the success of the business idea relies on the service cheapness achieved by resources optimization.

Talking about the weakness points of the model, we can say that some features, like weather conditions, can be affected by some uncertainty, the model exactness depends also on these forecasted attributes.

It is also important to mention the fact that in case of daily-grain rental prediction, the performances would be higher than the one for the hourly-grain prediction; anyway, we decided to maintain the rental prediction for hours as we thought it would be more useful to have a more fine-grained results for businesses in the field.

We also saw that our model suffers from the presence of holidays, in the middle of “common” days. In fact, for the models it is very hard to predict the peak of the demand for certain days of the year (e.g., Christmas). This issue could be fixed gathering more data for the training part of the models, so that they can learn more efficiently about sudden high/low peaks in demand for recurring days through the years.

6 Appendix

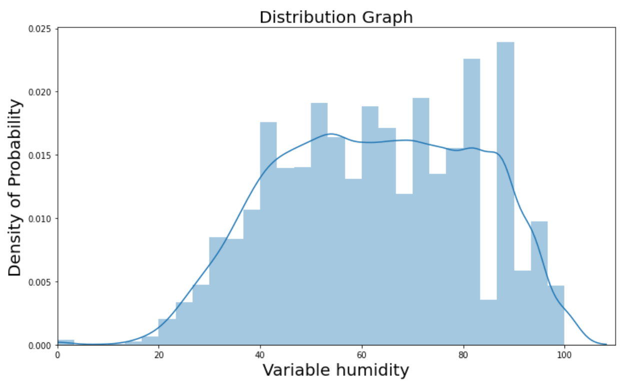
6.1 Data Analysis

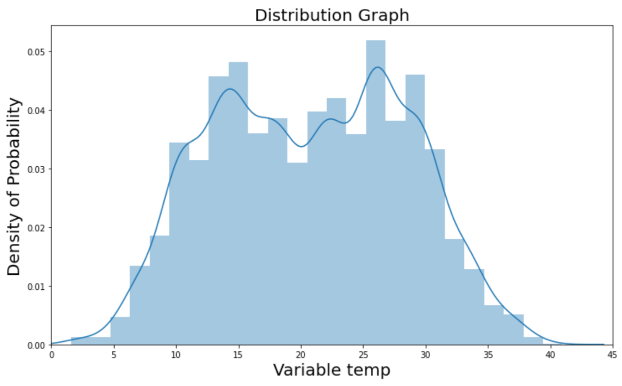
We start out our data exploratory analysis with plotting several types of plots to check the characteristics of each feature.

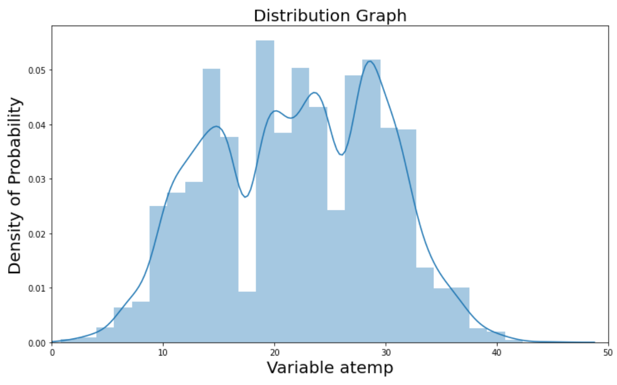
6.1.1 Distribution Plots

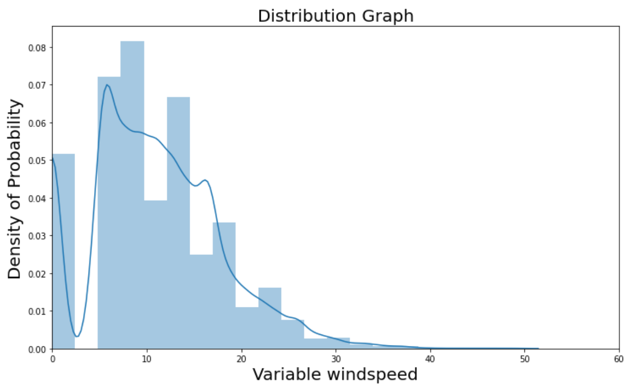
A distribution plot shows the possible values for a variable and how often they occur, and it is often used in data analysis to check whether they are normally distributed, uniformly or binomial distribution.
The normal or **Gaussian Distribution** is important for different reasons:

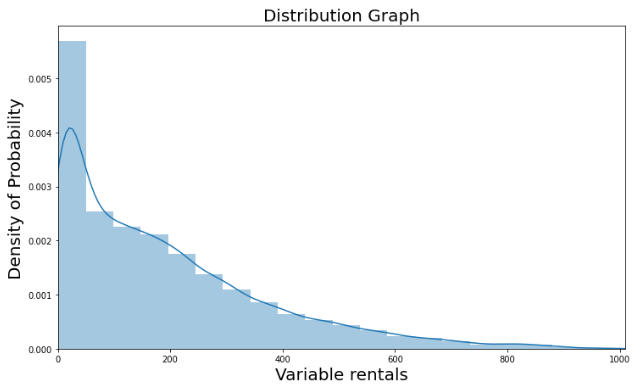
- Gaussian distribution is the most important probability distribution in statistics because it fits many natural phenomena like age, height, test-scores, etc.
- Datasets with Gaussian distributions makes applicable to a variety of methods that fall under parametric statistics. The methods such as propagation of uncertainty and least squares are applicable only to datasets with normal or normal-like distributions.
- Conclusions and summaries derived from such analysis are intuitive and easy to explain to audiences with basic knowledge of statistics.



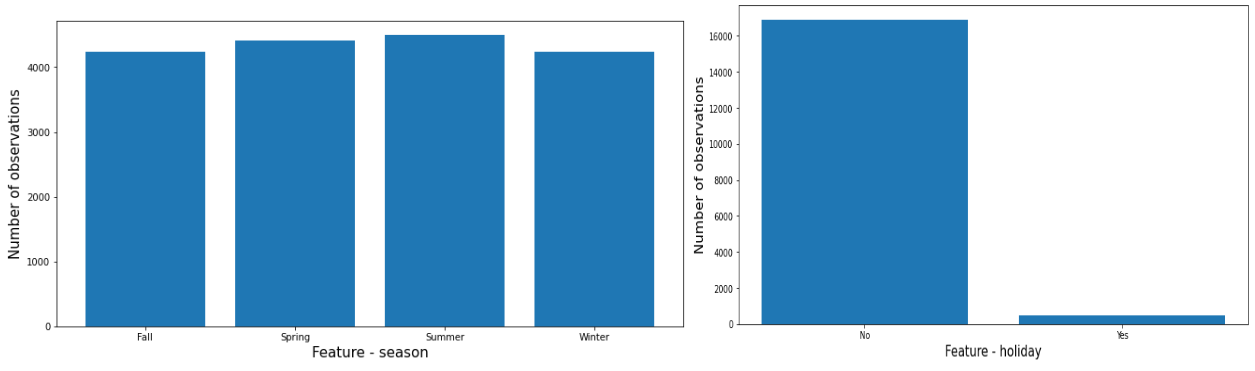


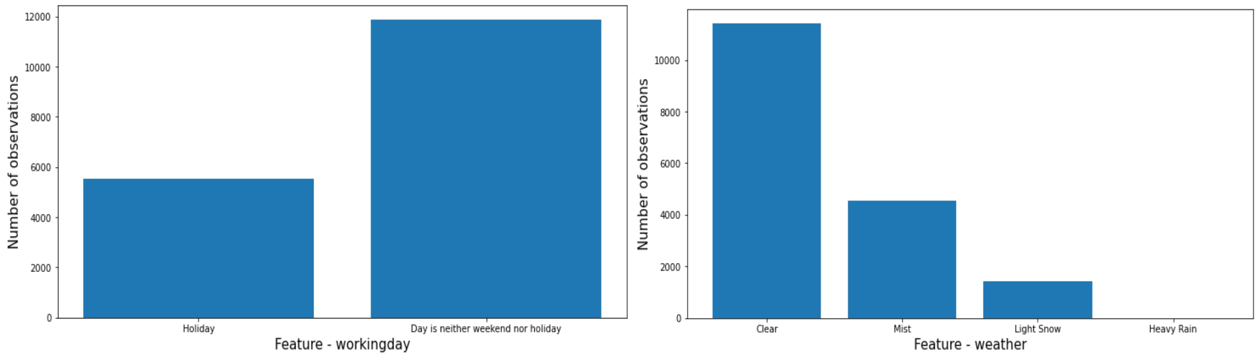






We can clearly see that the kernel density curve for variables *temp*, *atemp* and *humidity* follow a bell shape curve. Hence, they are normally distributed, while *windspeed* is not normally distributed.
The distribution plot for bike rentals shows that the probability of bikes rental is more likely to be higher between the range 0 to 100 and it decreases as number of rentals goes on increasing.



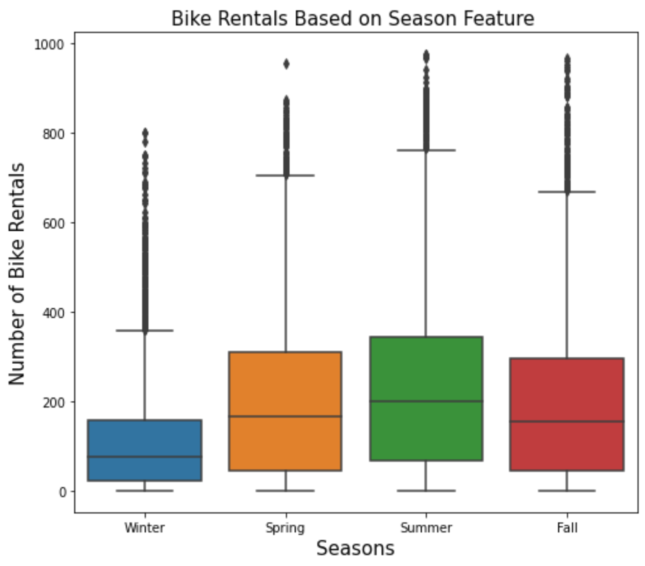


Season feature is divided into four different season and their distribution is uniform. All four seasons have the same number of observations spread across the two years dataset. We can also notice that there are more observations for no holiday and more observations for a working day in respective distribution plots.
The distribution plot for weather feature gives an idea that condition Clear, few clouds, partly cloudy, partly cloudy has more observations and condition Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog has the least observations, as we were expecting.

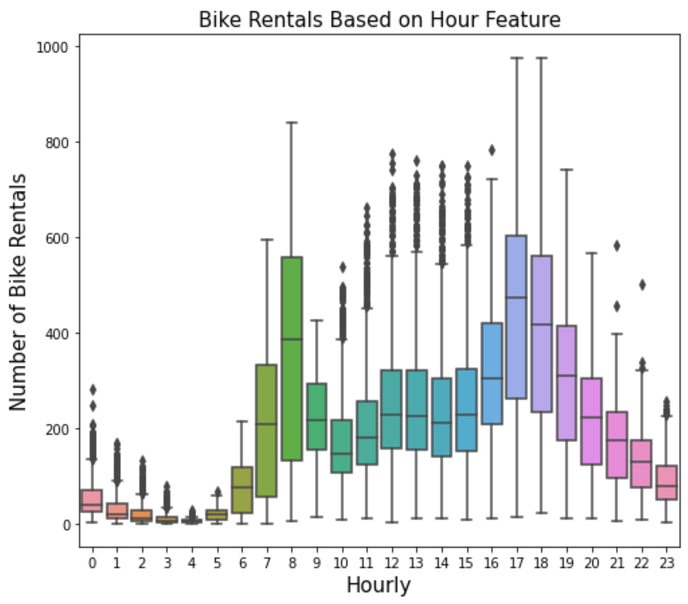
6.1.2 Bivariate Data Analysis

We also decided to perform Bivariate Analysis to study the relationship that exists between two variables. This has a lot of use in real life as it helps to find out if there is an association between the variables and if yes then what is the strength of association.

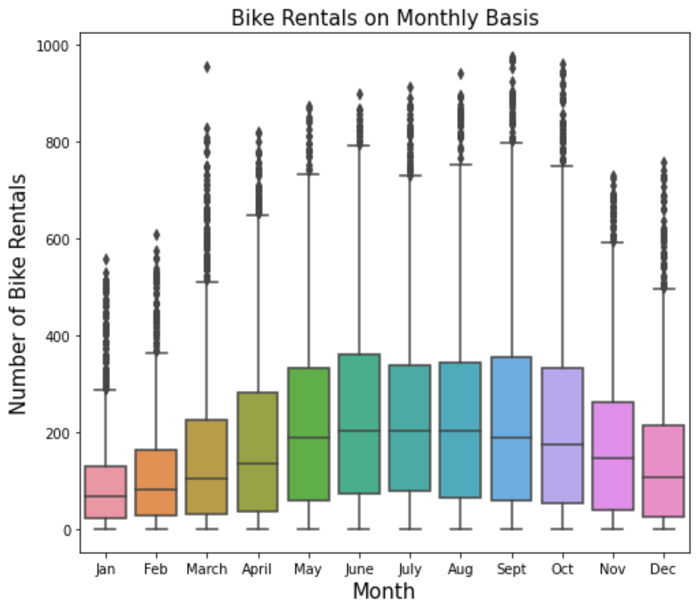
The bivariate analysis helps testing the hypothesis of casualty and association. It is useful to predict the value of a variable that is dependent based on changes that happen to an independent variable.
So, Boxplots gives a clear understanding of observations lying in the quartile percentages along with the outliers. Starting with the boxplot for bike rentals based on seasons we have:



We can clearly see that a greater number of bikes are rented in Summer and spring season when rain and cloudy weather is not present. Winter has the least number of bikes rented as we are speaking about Washington D.C area which experiences snowfall in winter.
Most outliers are present in Winter season as we can inspect maybe greater number of people rented bikes on holidays i.e., Thanksgiving or Christmas break. Regarding the couple hour-number of rentals we have:

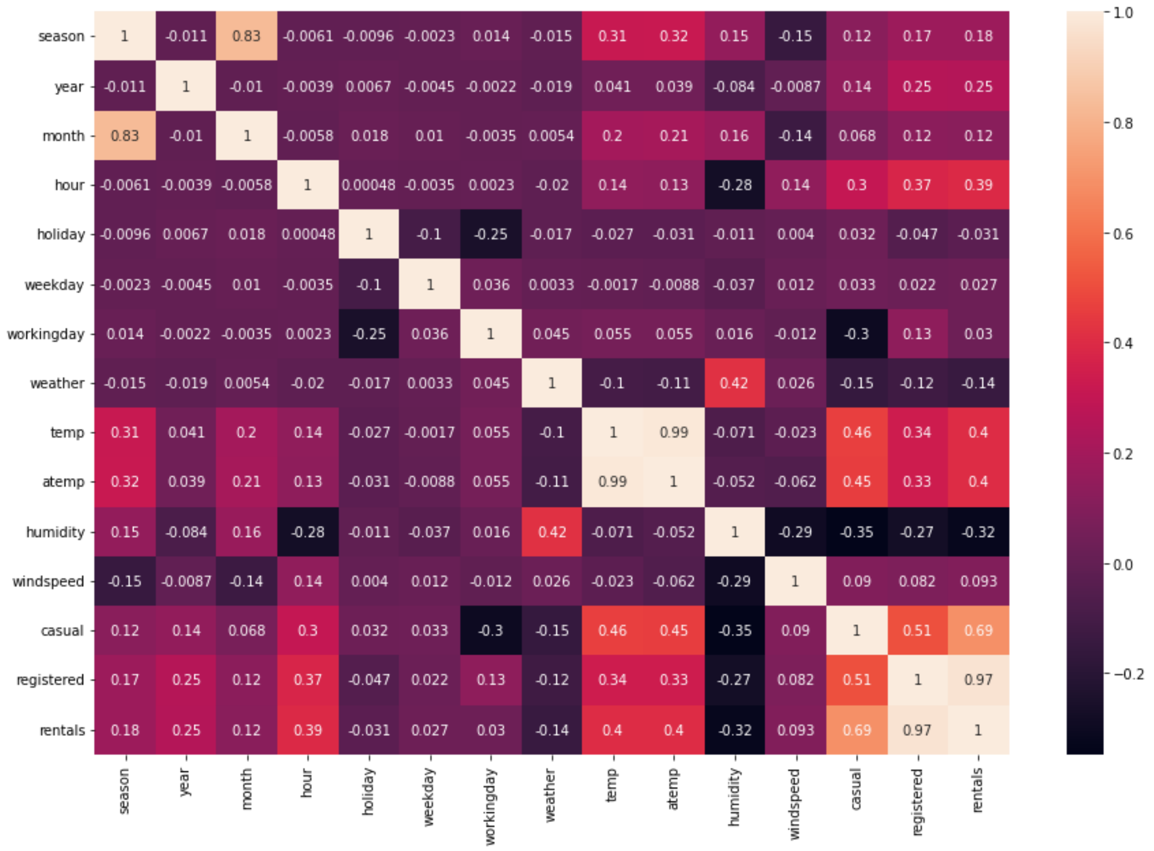


This boxplot gives an idea of how bikes are rented on hourly basis throughout the 24-hour cycle.
We can clearly see that in the morning during the peak office hours (7-8 AM) more bikes are rented and again in the evening during 5-6 PM when the sun comes down.
Finally, for bike rentals based on months:



This boxplot can be co-related directly with the season's boxplot. We see that from May to October we have the highest bike rental frequency which is in the Summer and Spring season.

6.1.3 Correlation Matrix



	season	year	month	hour	holiday	weekday	workingday	weather	temp	atemp	humidity	windspeed	casual	registered	rentals
season	1	-0.011	0.83	-0.0061	-0.0096	-0.0023	0.014	-0.015	0.31	0.32	0.15	-0.15	0.12	0.17	0.18
year	-0.011	1	-0.01	-0.0039	0.0067	-0.0045	-0.0022	-0.019	0.041	0.039	-0.084	-0.0087	0.14	0.25	0.25
month	0.83	-0.01	1	-0.0058	0.018	0.01	-0.0035	0.0054	0.2	0.21	0.16	-0.14	0.068	0.12	0.12
hour	-0.0061	-0.0039	-0.0058	1	0.00048	-0.0035	0.0023	-0.02	0.14	0.13	-0.28	0.14	0.3	0.37	0.39
holiday	-0.0096	0.0067	0.018	0.00048	1	-0.1	-0.25	-0.017	-0.027	-0.031	-0.011	0.004	0.032	-0.047	-0.031
weekday	-0.0023	-0.0045	0.01	-0.0035	-0.1	1	0.036	0.0033	-0.0017	-0.0088	-0.037	0.012	0.033	0.022	0.027
workingday	0.014	-0.0022	-0.0035	0.0023	-0.25	0.036	1	0.045	0.055	0.055	0.016	-0.012	-0.3	0.13	0.03
weather	-0.015	-0.019	0.0054	-0.02	-0.017	0.0033	0.045	1	-0.1	-0.11	0.42	0.026	-0.15	-0.12	-0.14
temp	0.31	0.041	0.2	0.14	-0.027	-0.0017	0.055	-0.1	1	0.99	-0.071	-0.023	0.46	0.34	0.4
atemp	0.32	0.039	0.21	0.13	-0.031	-0.0088	0.055	-0.11	0.99	1	-0.052	-0.062	0.45	0.33	0.4
humidity	0.15	-0.084	0.16	-0.28	-0.011	-0.037	0.016	0.42	-0.071	-0.052	1	-0.29	-0.35	-0.27	-0.32
windspeed	-0.15	-0.0087	-0.14	0.14	0.004	0.012	-0.012	0.026	-0.023	-0.062	-0.29	1	0.09	0.082	0.093
casual	0.12	0.14	0.068	0.3	0.032	0.033	-0.3	-0.15	0.46	0.45	-0.35	0.09	1	0.51	0.69
registered	0.17	0.25	0.12	0.37	-0.047	0.022	0.13	-0.12	0.34	0.33	-0.27	0.082	0.51	1	0.97
rentals	0.18	0.25	0.12	0.39	-0.031	0.027	0.03	-0.14	0.4	0.4	-0.32	0.093	0.69	0.97	1

A correlation matrix is a table showing correlation coefficients between variables. Each cell in the table shows the correlation between two variables. A correlation matrix is used to summarize data, as an input into a more advanced analysis, and as a diagnostic for advanced analyses. Key decisions to be made when creating a correlation matrix include choice of correlation statistic, coding of the variables, treatment of missing data, and presentation. We have created a heatmap correlation matrix using seaborn library in python.
We can see that *temp* and *atemp* variables/features are co-linear to each other.

Hence, we decided to drop the *atemp* variable in the actual model building phase. Also, casual and registered user's variables are co-linear and as rentals (target variables) is the addition of *casual* and *registered* variables we will drop *casual* and *registered* variables in further analysis.

6.2 Data Cleaning & Preprocessing

The data input fed to predictor models plays an important role and the accuracy of the output depends on it. If the input data is not cleaned and processed properly then even strong predictive models will yield unsatisfactory and inaccurate predictions.
The '***Bike_Demand_Data***' dataset available on Kaggle was clean.

After a first analysis the dataset contained:

- No null or missing values
- No duplicate values/observations
- No text values. Although weather, season are categorical data types, they were already encoded into numerical outputs and described in the data attributes section.

Data preprocessing is an important task. It is a data mining technique that transforms raw data into a more understandable, useful, and efficient format. Data preprocessing is an integral step in Machine Learning as the quality of data and the useful information that can be derived from it directly affects the ability of our model to learn; therefore, it is extremely important that we preprocess our data before feeding it into our model. We have done preprocessing in three steps as follows:

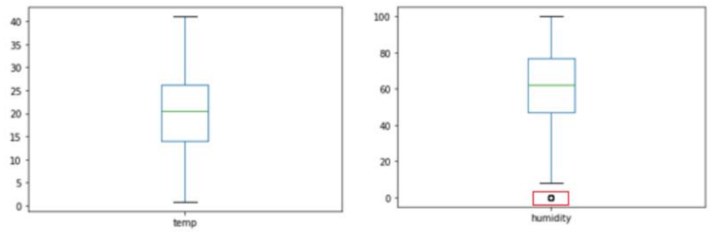
1. Outlier Removal
2. Feature Selection
3. Assigning Dummy Variables
4. Feature Scaling – Normalization

6.2.1 Outlier Removal

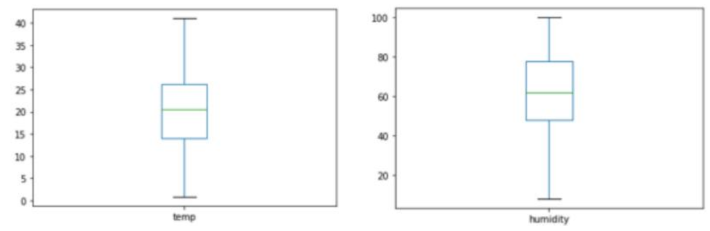
Outliers are unusual values in your dataset, and they can distort statistical analyses and violate their assumptions. Outliers increase the variability in your data, which decreases statistical power.

Consequently, excluding outliers can cause your results to become statistically significant. We have used the boxplot method to find and remove the outliers from our dataset. We will be discarding the values below 25th and above 75th percentile to make the data clean and increase our model's accuracy.

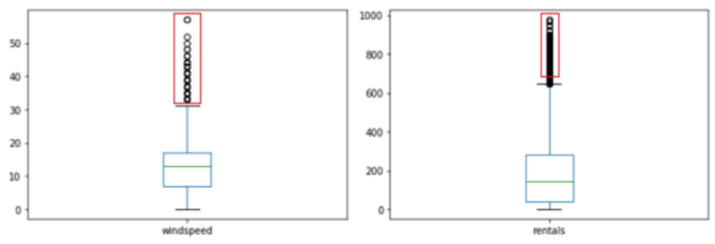
We have performed the outlier removal preprocessing on 4 variables namely: temp, humidity, windspeed and rentals. The following boxplots will give a proper and concise visualization of the entire process:



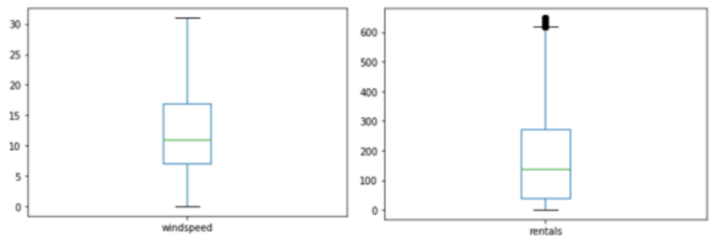
After Outliers Removed



As we can see the above boxplots contain the outliers we removed in the below boxplots. Interestingly temp did not have any absurd temperature values hence did not yield any outliers.



After Outliers Removed



Windspeed and rentals variables had the most outliers. After the outliers are removed the observations are reduced to 10348, i.e., 538 observations are outliers and discarded.
The processed file is saved as '*2_bike_rental_dataset_preprocessed.csv*' file and will be used for reading the data in Jupyter notebook for subsequent stages (predicting model building).

6.2.2 Feature Selection

Feature selection is the process of reducing the number of input variables when developing a predictive model. It is desirable to reduce the number of input variables to both reduce the computational cost of modeling and, in some cases, to improve the performance of the model.

We decided to exploit the *SelectKBest* algorithm with Mutual Info Regression that calculates the mutual information of each attribute. MI is an information theory concept; it basically quantifies the "amount of information" obtained about one random variable by observing the other random variable. It is equal to zero if and only if two random variables are independent, and higher values mean higher dependency, and so it let us obtain a list of the most important attributes for estimating the target variable. Here is the result of the attribute selection process that shows the features with the corresponding score:

```
     Specs     Score
      hour  0.630017
      temp  0.143119
  humidity  0.097018
     month  0.074361
    season  0.054894
      year  0.042504
   weekday  0.020206
   weather  0.014353
 windspeed  0.014237
workingday  0.011209
   holiday  0.010339
```

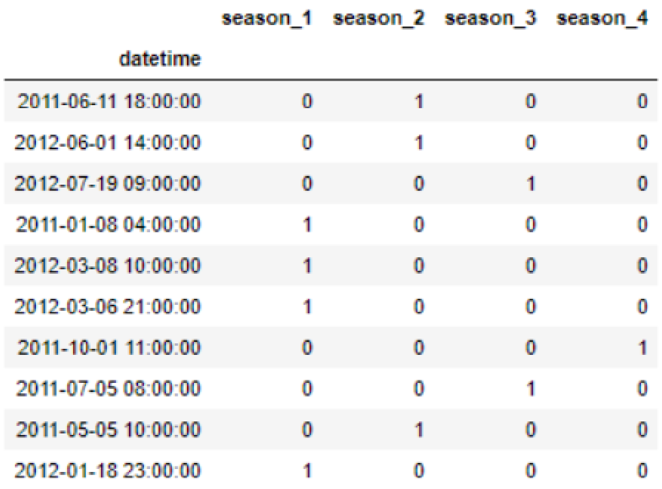
We then decided not to preserve the holiday attribute as it contributes in determining the target variable is negligible.

6.2.3 Dummy Variable Generation

We decided to transform categorical values into Binary Attributes, as in the majority of cases using dummy variables is more statistically meaningful than using a single numerical variable.
A single numerical variable does not accurately encode the information represented by a categorical variable, because of the relationships between numerical values it implicitly implies. On the number line, 1 is closer to 2 than it is to 3. However, by definition no pair of values assumed by categorical variables is more similar than

any other. Encoding three such values as 1, 2 and 3 will result in the model inferring incorrect relationships between those numbers.

The fact that real valued numbers fall on a number line that implies a greater degree of similarity between values that fall closer to one another along it; this notion is incompatible with categorical variables. Using dummy variables avoids this issue.
Most of the machine learning libraries expects all features to be numeric i.e., convert all categorical values to sensible numeric values. In this we have two different categories namely ‘Ordered’ and ‘Unordered’. We do not need ordered lists in our data while coding. Hence, we will be assigning dummy variables to features like ‘weather’, ’season’, and ’hour’.
Example on attribute season:



	season_1	season_2	season_3	season_4
datetime				
2011-06-11 18:00:00	0	1	0	0
2012-06-01 14:00:00	0	1	0	0
2012-07-19 09:00:00	0	0	1	0
2011-01-08 04:00:00	1	0	0	0
2012-03-08 10:00:00	1	0	0	0
2012-03-06 21:00:00	1	0	0	0
2011-10-01 11:00:00	0	0	0	1
2011-07-05 08:00:00	0	0	1	0
2011-05-05 10:00:00	0	1	0	0
2012-01-18 23:00:00	1	0	0	0

7 References

1. Knowledge Adaption for Demand Prediction based on Multi-task Memory Neural Network [https://arxiv.org/abs/2009.05777]
2. Utilizing Artificial Neural Networks to Predict Demand for Weather-sensitive Products at Retail Stores (2017, Taghizadeh) [https://www.researchgate.net/publication/321241806_Utilizing_artificial_neural_networks_to_predict_demand_for_weather-sensitive_products_at_retail_stores]
3. Machine-Learning Models for Sales Time Series Forecasting (2019, Pavlyshenko) [https://www.mdpi.com/2306-5729/4/1/15]
4. Neural network models (supervised) - Multi-layer Perceptron [https://scikit-learn.org/stable/modules/neural_networks_supervised.html]
5. Ensemble methods - Random Forests [https://scikit-learn.org/stable/modules/ensemble.html]
6. Metrics and scoring: quantifying the quality of predictions [https://scikit-learn.org/stable/modules/model_evaluation.html]