Demand Forecasting: Public Bike Rental Predictive Modeling

by

Aniket Amar Thopte

A capstone project submitted to the faculty of
The University of North Carolina at Charlotte
in partial fulfillment of the requirements
for the degree of Master of Science in
Engineering Management

Charlotte

2021

Approved by:

_____

Dr. Guanglin Xu

ABSTRACT

ANIKET AMAR THOPTE. Demand Forecasting: Public Bike Rental Predictive Modeling. (Under the direction of DR. GUANGLIN XU)

Demand forecasts are essential for any business to manage its different business activities but at the same time are difficult to do with utmost accuracy using traditional techniques. Big Data is part of any business acumen in today's world and the world needs to embrace it and make maximum use of it to increase productivity, economic benefits, etc. Huge volumes of data are generated, and many data analyst and scientists are working on historical data to derive meaningful insights and provide better data-driven decisions. In order to do so state of the art machine learning and artificial intelligence is used and perfected to outsmart the competitors. The aim of the project is to use the state-of-the-art machine learning techniques and improve their implementation to get the most accurate predictive models for our data and forecast better demand with least errors. The project aims to showcase better demand forecasting in an emerging bike sharing and Public Vehicle Rental Space. A number of forecasting models like Linear Regression, Clustering, Random Forest, Artificial Neural Nets are built, experimentally investigated, and their practicality discussed. The results will be used to investigate consumer behavior on demand forecasting and the overall impact on the business as a whole.

DEDICATION

To my supporting parents.

Your unconditional, endless love and unyielding support mean the world to me!

ACKNOWLEDGMENT

First and foremost, I wish to record my sincere gratitude to University of North Carolina at Charlotte for making available essential facilities to students like the library and other indirect facilities needed during the ongoing Covid-19 pandemic to make the masters journey smooth and successful.

I am extremely happy at the completion of this capstone project and would like to show my sincere gratitude to Dr. Guanglin Xu, Project Guide and Mentor, Assistant Professor, Systems Engineering & Engineering Management, University of North Carolina at Charlotte for giving me valuable insights and the much-needed guidance throughout the semester to successfully complete the capstone project.

My gratitude to Dr. Tao Hong, EMGT 6985 – Engineering Management Project Instructor, Associate Professor, Graduate & Research Director, Systems Engineering & Engineering Management, University of North Carolina at Charlotte for his guidance & technical support and instructions throughout the semester.

Finally, I would like to take this opportunity to thank all of the people who have directly and indirectly guided me in completing this project.

It was a wonderful experience working on this project topic. Overall, it has been a great personal learning experience and successfully accomplishing such a wonderful project where I could apply my knowledge and skills practically.

Table of Contents

List of Figures

1. Introduction

Over the past decades, people have taken a liking to bike rental transportation services over the traditional public transportation services like Bus, Taxi's, Cab services. Major metropolitan cities like San Francisco, New York, Boston and many other cities have seen the rise in the bike rental numbers along with the rise in companies and startups like Lime, Lyft, Uber to name a few, investing heavily in this sector of public rental space. Bike sharing systems have become popular means of travel in recent years, providing a green and flexible transportation scheme to citizens in metropolitan areas. Many governments in the world have seen this as an innovative strategy that could potentially bring a number of societal benefits. For instance, it could reduce the use of automobiles and hence reduce greenhouse gas emission and alleviate traffic congestion in city centers.

But while it all sounds too easy, technologically advanced and easy to implement, the capacity planning that goes into these rental services, the supply, demand of bike rental services is very much volatile and dynamic in nature. As the population is on the rise and many different forms of public transportation on the rise along with the real estate being limited this unpredictable nature of demand forecasting needs to be addressed and predicting the future rental demand and planning the supply accordingly is very important to establish a smooth, efficient rental space business. An average bike ride can cost you anything based on time or distance metric. 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.

But what do we actually mean when we say demand forecasting, is it just generating numbers? What it actually means is taking into account the historical generated data and the study of various variables connected to the target variable. Using this when we build a state-of-the-art predictive model to generate the future demand is what we can call demand forecasting.

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. The dataset is publicly available hence this project is transparent and reproducible. Finally, the report is organized as follows: Literature review contains the research papers reviewed for the project to gain insights. Information about the data set and each attribute is explained in section 3. Preliminary data analysis with valuable insights is explained in the section 4. Section 5 contains the methods implemented for data cleaning and preprocessing. Section 6 explains the evaluation metrics used for testing the performance of the predictive models being used. All the machine learning models implemented are explained with their individual result in section 7. Chapter 8 concludes the project report with the conclusions and the possible future work.

2. Literature Review

Many researchers have worked hard and conducted their experiments and made contributions to the demand forecasting field. The range of techniques, methodologies used differed in many ways but getting the accurate predictive model was the aim of each and everyone.

In the study done by Wen Wang [1], the New York Citibike data set was used. Citibike is the bike-share program in New York. Unlike other bike rental demand forecasting studies, the data preparation in this study involved data collection from three different sources: one for individual ride information, one for weather data and lastly one for official holidays in New York. The author proposed an improved multiple linear regression model, Neural Nets, Decision Trees & Random Forest to predict the hourly bike demand. The significance of this paper was the 'Random Forest' ensemble method which improved the accuracy. The significance was due to the authors introduction of new independent attributes and improvement in each independent attribute and transforming the dependent variable.

In 2017, YouLi Feng and ShanShan Wang [2] proposed a forecast model for bicycle rental demand based on random forests and multiple linear regression that got featured in the 2017's 16th International Conference on Computer and Information Science (ICIS) held in Wuhan, China. The key aspect of the forecasting model was the linear regression model was established by the conventional method, but the multiple linear regression (MLR) was obtained by using SPSS software. The prediction accuracy of MLR was too low, although a good linear relationship between factors. To improve the accuracy and cover the shortcomings of MLR the authors proposed a random forest model coupled with GBM packet to improve the decision trees prediction accuracy.

In the paper represented by Om Prakash Nekkanti [3], 'Prediction Of 2016 Rental Demand in Great Rides Bike Share Program' the research analyzed the effect of weather (average temperature, total daily precipitation, average wind speed and weather outlook) and other important factors effecting the demand in Great Rides Bike Share program in Fargo, North Dakota, U.S.A. The paper summarized the application of Bayesian methods (Naïve Bayes & Bayes Network) and Decision Trees. One of the highlights of this paper is the use of business analytics software 'Weka' for building the Decision Trees model. Clear and concise visualizations for the predictions make it easy to understand and finally the order of importance among the causal attributes captures the non-linear and non-parametric relationship between the factors.

3. Data Set

The data set used for this project is sourced from open-source data website 'Kaggle'. The 'Bike_Demand_Data' dataset is open-source and made available on Kaggle from Capital Bikeshare system, Washington D.C., USA and the corresponding weather and seasonal information is sourced from freemeteo (now rebranded as i-weather.com) 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. We have total 10886 observations/records. The amount of data 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. More descriptive information about the data variables is given in the 'Data Attributes Table'.

3.1 Data Attributes

Kaggle is the primary source of data for the project. 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 season 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, 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 particular direct connection to the rental company. They may be viewed as guest users and does not talk about a particular user frequency. Example:- Tourist, once in a while users, 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 bike rented or Bike users and is simply the addition of casual and registered variables.

4. Data Visualization & Exploratory Analysis

4.1. Distribution Plots.

We will be plotting distribution plots to check the distribution of each feature and see if they are normally distributed, uniformly or binomial distribution. A distribution plot shows the possible values for a variable and how often they occur.

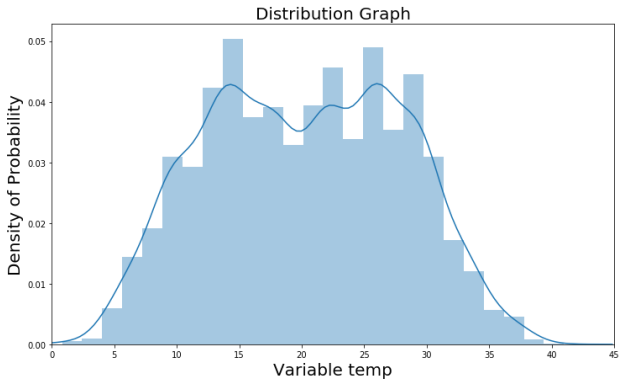


Fig.1: Distribution plot for actual temperature feature

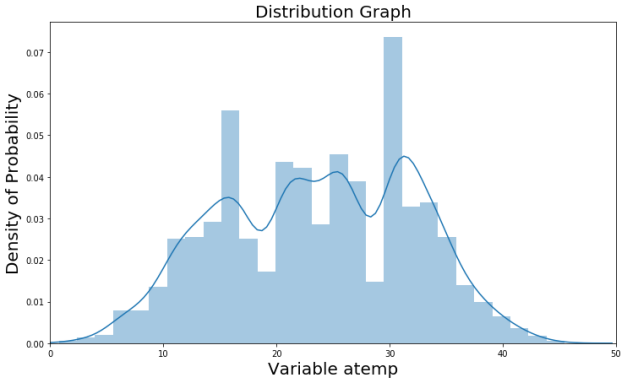


Fig.2: Distribution plot for feels like temperature feature

Figures 1 & 2 are the distribution plots for the actual and feels like temperature feature. We can clearly see that the kernel density curve for both follows a bell shape curve. Hence they are normally distributed.

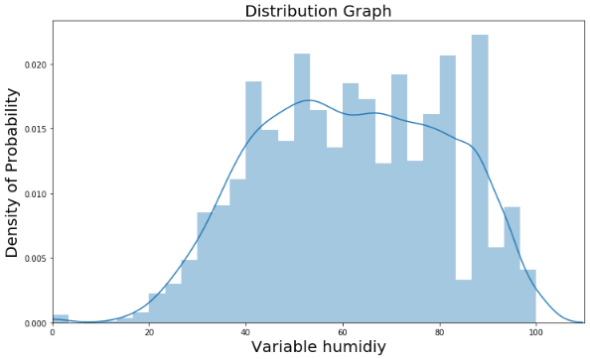


Fig.3: Distribution plot for humidity

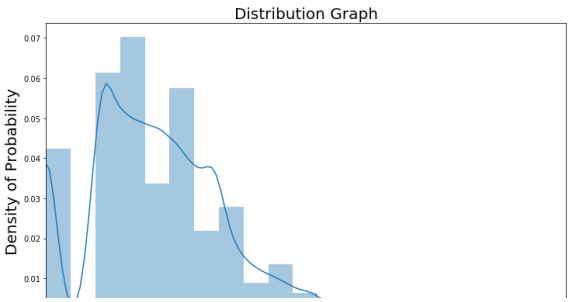


Fig.4: Distribution plot for windspeed

Figure 3 shows a bell curve hence it is a normal distribution for humidity feature. Windspeed is not normally distributed.

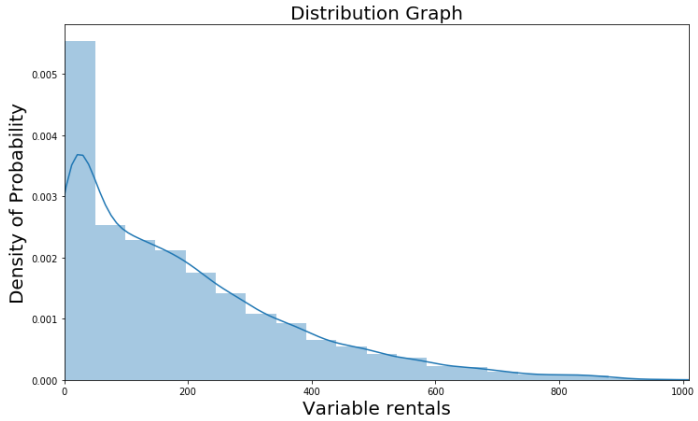


Fig.5: Distribution plot for target variable rentals.

The distribution plot for bike rentals shows that the probability of bikes rental is more between the range 0 to 100 and its dips down as number of rentals goes on increasing.

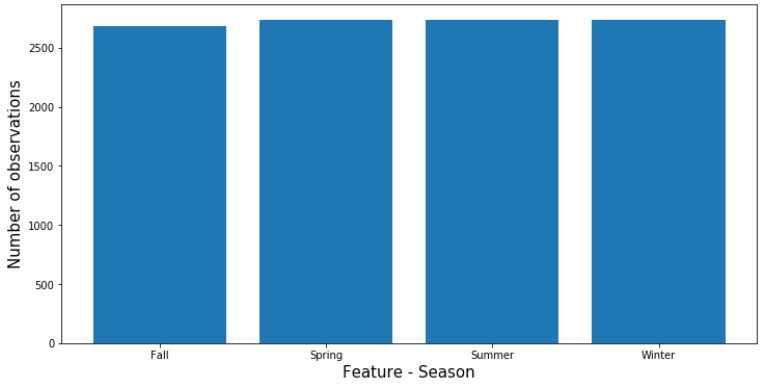


Fig.6: Distribution plot for season feature.

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.

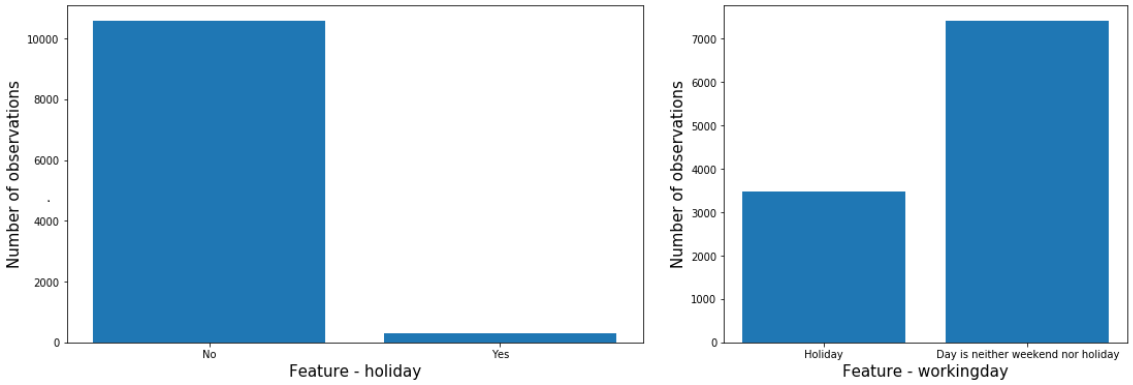


Fig.7: Distribution plot for holiday and workingday feature.

We have more observations for no holiday and more observations for a working day in respective distribution plots.

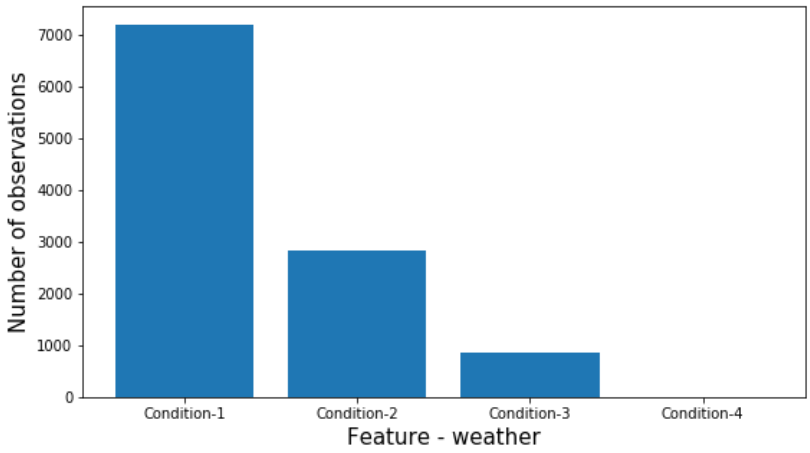


Fig.8: Distribution plot for weather feature.

The distribution plot for weather feature gives an idea that condition-1 i.e., Clear, few clouds, partly cloudy, partly cloudy has more observations and condition-4 (Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog) has the least observations.

4.2. Bivariate Data Analysis for Target Variable (rentals).

Another simple technique to draw insights from a dataset are boxplots. Boxplots gives a clear understanding of observations lying in the quartile percentages along with the outliers present.

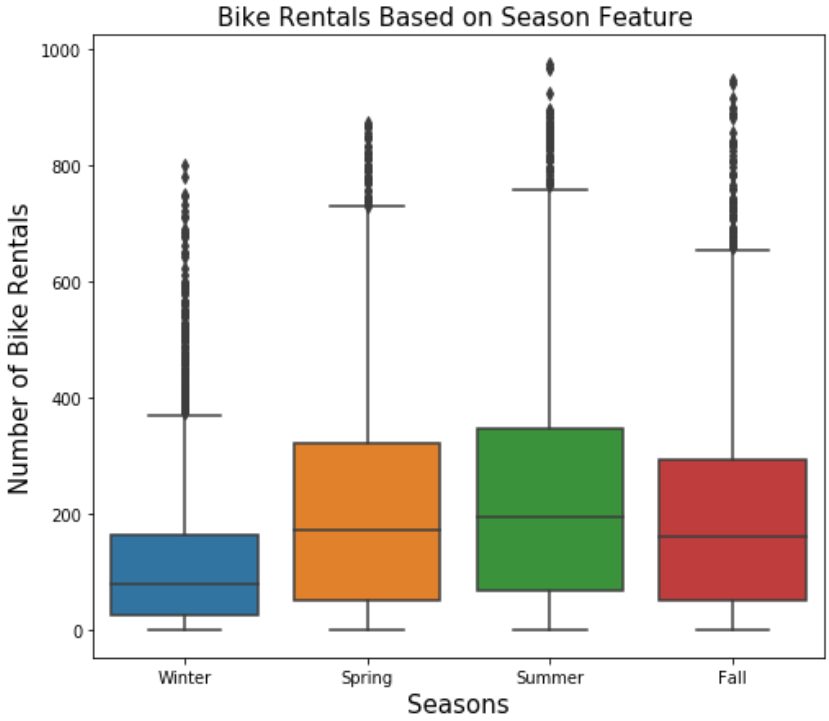


Fig.9: Boxplot for bike rentals based on seasons.

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. And a majority of outliers are present in Winter season as we can inspect maybe a greater number of people rented bikes on holidays i.e., Thanksgiving or Christmas break.

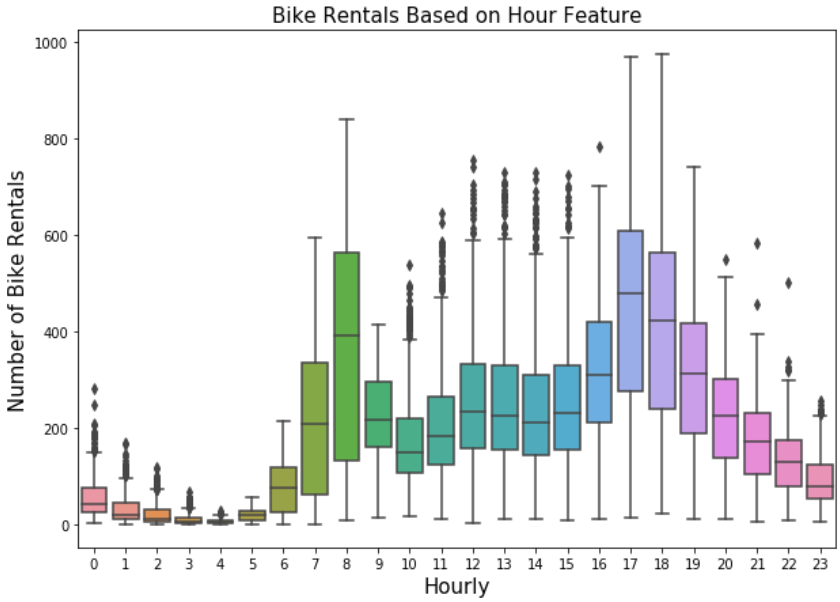


Fig.10: Boxplot for bikes rented based on time of the day.

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.

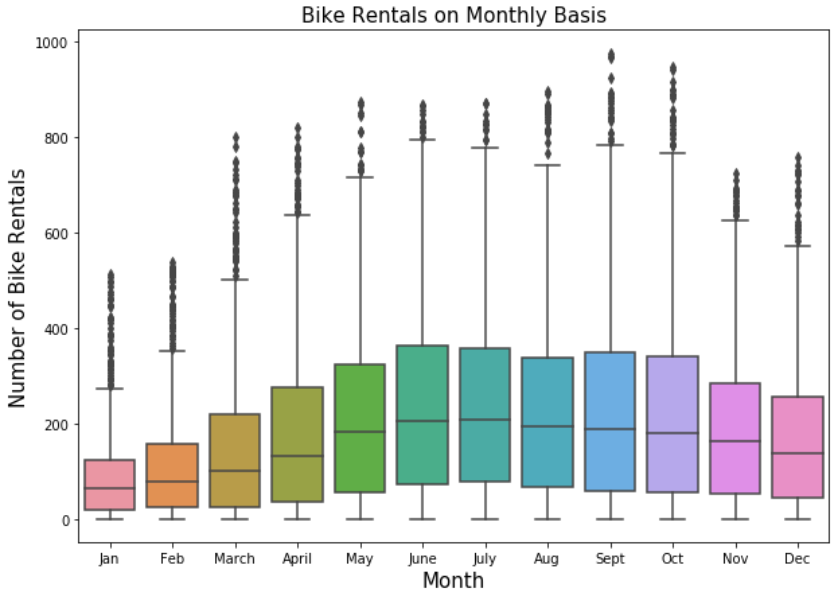


Fig.11: Boxplot for bikes rented on monthly basis.

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

4.3. Correlation Matrix (Heatmap)

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 **[4]**. 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 **[4]**. We have created a heatmap correlation matrix using seaborn library in python.

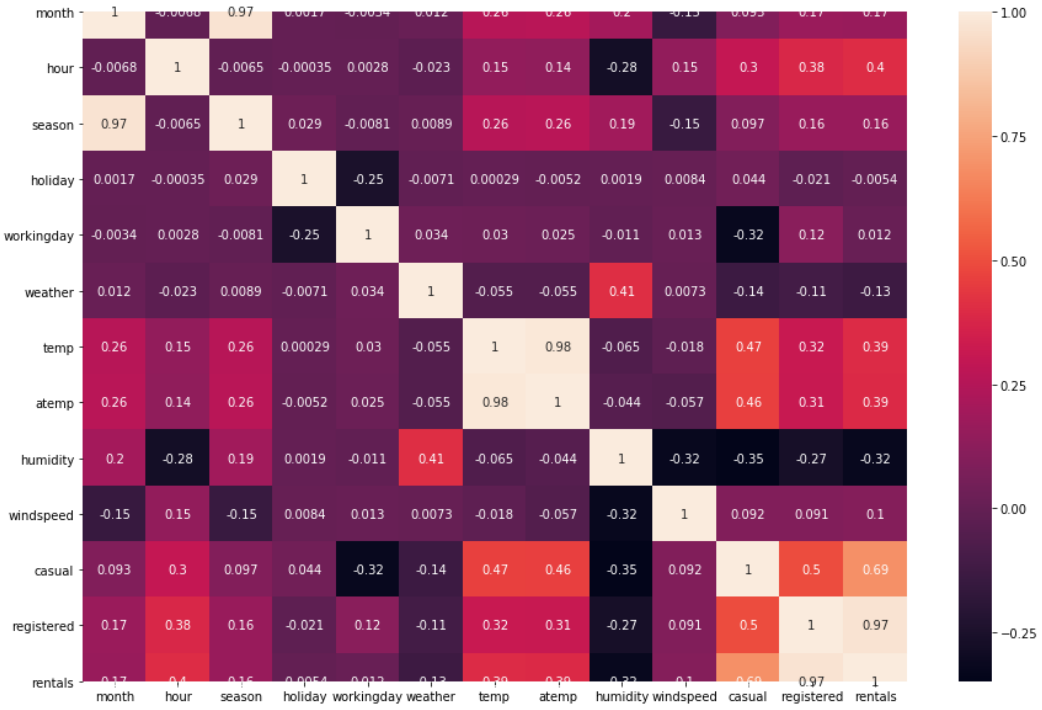


Fig.12: Heatmap Correlation Matrix.

We can see that temp and atemp variables/features are co-linear to each other. Hence we will 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.

5. Data Cleaning & Preprocessing

No matter how optimum or advanced your forecasting model is, the data input fed to it 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. Most of the data scientists and analyst spend almost 50% of their time cleaning and organizing the data, which plays a vital role in increasing the accuracy of your models and lowers the errors.

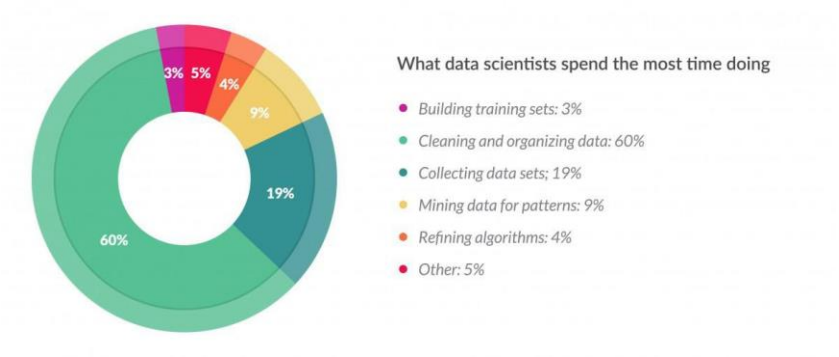


Fig.13: Pie Chart for time distribution of a Data Scientist's Project [5].

The 'Bike_Demand_Data' dataset available on Kaggle was clean. The dataset contained:

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

5.1. Data Preprocessing

Data preprocessing is an important task. It is a data mining technique that transforms raw data into a more understandable, useful and efficient format [6]. 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 [7]. We have done preprocessing in three steps as follows:

5.1.1. 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 [8]. We have used the boxplot method to find and remove the outliers from our dataset. We will be discarding the values below 25^{th} and above 75^{th} 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 process.

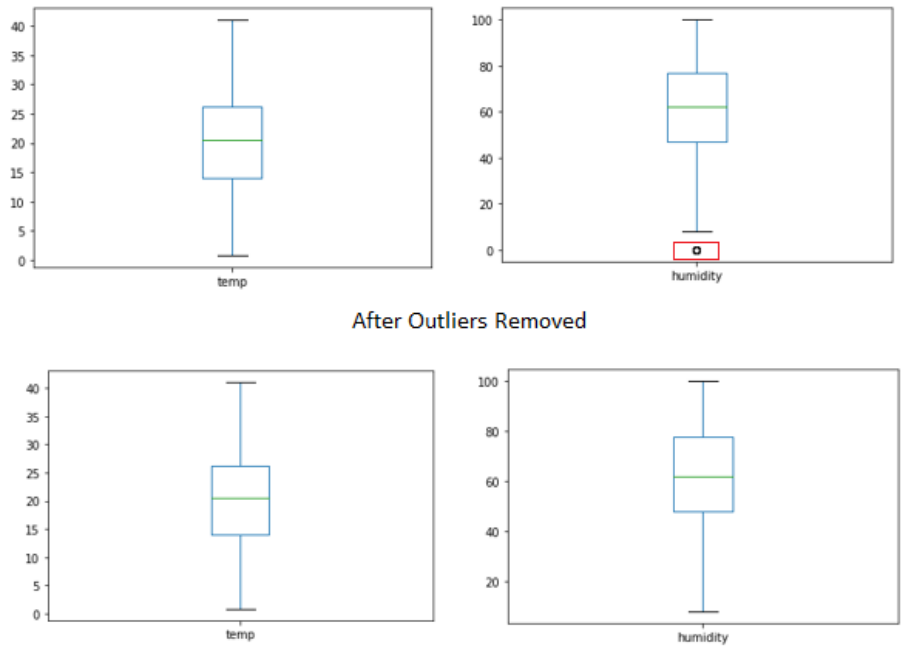


Fig.14: Box plots of Outliers for variables ‘temp’ & ‘humidity’ from left.

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

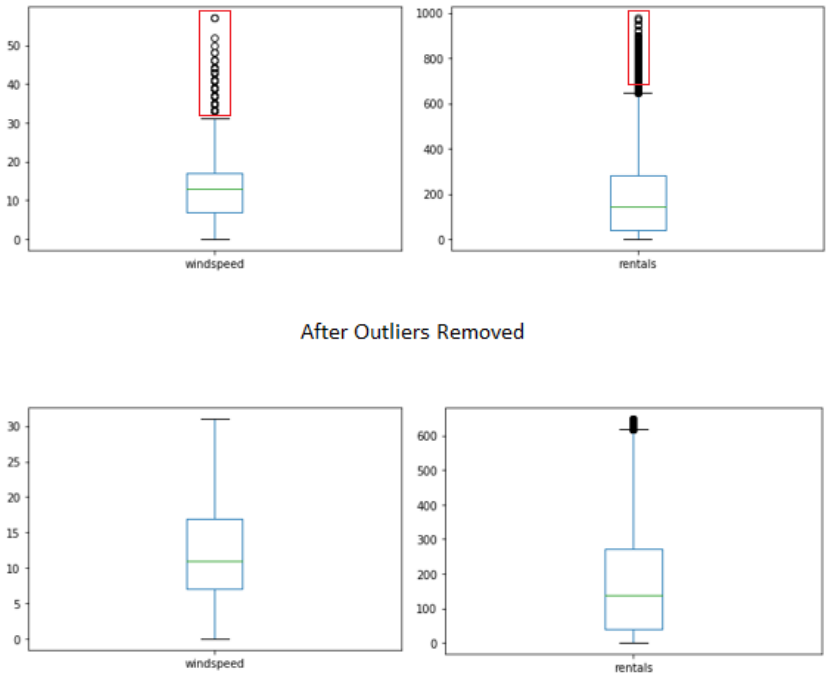


Fig.15: Box plots of Outliers for variables ‘windspeed’ & ‘rentals’ from left.

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 'cleandata.csv' file and will be used for reading the data in Jupyter notebook.

5.1.2. Assigning Dummy Variables.

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'. Hence we create 4 dummy variables for weather and season features, respectively. For hour feature 23 dummy variables are created starting from h0 (0 or 12AM) to h23 (23 or 11PM).

datetime	season_1	season_2	season_3	season_4
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

Fig.16: Dummy Variables for 'season'.

datetime	weather_1	weather_2	weather_3	weather_4
2011-06-11 18:00:00	0	1	0	0
2012-06-01 14:00:00	0	0	1	0
2012-07-19 09:00:00	0	1	0	0
2011-01-08 04:00:00	0	0	1	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	1	0
2011-07-05 08:00:00	1	0	0	0
2011-05-05 10:00:00	1	0	0	0
2012-01-18 23:00:00	1	0	0	0

Fig.17: Dummy Variables for 'weather'.

datetime	hour_0	hour_1	hour_2	hour_3	hour_4	hour_5	hour_6	hour_7	hour_8	hour_9	...	hour_14	hour_15	hour_16	hour_17	hour_18
2011-06-11 18:00:00	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	1
2012-06-01 14:00:00	0	0	0	0	0	0	0	0	0	0	...	1	0	0	0	0
2012-07-19 09:00:00	0	0	0	0	0	0	0	0	0	1	...	0	0	0	0	0

Fig.18: Dummy Variables for 'Hour'.

5.1.3. Feature Scaling: Normalization and Standardization.

Most of our data features have spanning varying degrees of magnitude, range and units. For example, temp has a short range of 0 to 41 values and calculated in Celsius, while humidity has max range of 100 and a different unit of measure. This is a significant obstacle as a few machine learning algorithms are highly sensitive to these features [9]. Some machine learning algorithms will achieve better performance if the data has a consistent scale or distribution, and different features can be used on the same level. For this purpose, two techniques that every data scientist or machine learning enthusiast uses are 'Normalization' or 'Standardization'.

What is Normalization?
Normalization is a scaling technique in which values are shifted and rescaled so that they end up ranging between 0 and 1. It is also known as Min-Max Scaling [9].

Here's the formula for normalization:

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}}$$

Fig.19: Mathematical formula for Normalization technique [9].

Here, Xmax and Xmin are the maximum and the minimum values of the feature, respectively.

What is Standardization?
Standardization is another scaling technique where the values are centered around the mean with a unit standard deviation. This means that the mean of the attribute becomes zero and the resultant distribution has a unit standard deviation [9].

Here's the formula for standardization:

$$X' = \frac{X - \mu}{\sigma}$$

Fig.20: Mathematical formula for Standardization technique [9].

We have tried both normalization and standardization for our data set and we see that the errors reduce for the standardize scaling. Hence we will be using the standardization scaling technique.

We have performed Normalization on features windspeed, humidity, temp. And we have performed standardization using standard scaler feature of sklearn library on entire dataset.

Normalization:

```
df['windspeed'] = df['windspeed'] / df['windspeed'].max()
df['humidity'] = df['humidity'] / df['humidity'].max()
col = ["temp", "atemp"]
df[col] = (df[col]-df[col].min())/(df[col].max()-df[col].min())
```

Fig.21: Python code Snapshot for Normalization

Standardization:

```
from sklearn.preprocessing import StandardScaler
X_scaled = StandardScaler().fit_transform(X)
```

Fig.22: Python code Snapshot for Standardization

6. Evaluation or Performance Metrics.

Evaluating a model is a core part of building an effective machine learning model. When building any model, we need to assess if its running at optimal level and giving us the highest expected accuracy. Hence it becomes very important to check how robust our predictive model is. We will be using cross validation method using sklearn library to find the best optimal hyperparameters for all the predictive models implemented.

Once the models are built and predictions are done on the testing data set, we will be using Mean Absolute Error (MAE), Mean Square Error (MSE) & Root Mean Square Error (RMSE) to evaluate the errors and the same metric will be used to assess and compare all the predictive models.

6.1. Performance Metrics:

MAE

The Mean absolute error represents the average of the absolute difference between the actual and predicted values in the dataset. It measures the average of the residuals in the dataset.

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}|$$

Where,

$\hat{y}$ – *predicted value of y*

$\bar{y}$ – *mean value of y*

Fig.23: Mathematical formula for Mean Absolute Error [10]

MSE

Mean Squared Error represents the average of the squared difference between the original and predicted values in the data set. It measures the variance of the residuals.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y})^2$$

Fig.24: Mathematical formula for Mean Square Error [10]

RMSE

Root Mean Squared Error is the square root of Mean Squared error. It measures the standard deviation of residuals.

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y})^2}$$

Fig.25: Mathematical formula for Root Mean Square Error [10]

Mean Squared Error (MSE) and Root Mean Square Error penalizes the large prediction errors vi-a-vis Mean Absolute Error (MAE). However, RMSE is widely used than MSE to evaluate the performance of the regression model with other random models as it has the same units as the dependent variable (Y-axis) [10]. The lower value of MAE, MSE, and RMSE implies higher accuracy of a regression model [10].

6.2. Cross Validation Technique

Cross Validation is one of the most important concepts in any type of data modelling. It simply says, try to leave a sample on which you do not train the model and test the model on this sample before finalizing the model [11]. K-Fold gives us a way to use every single datapoint which can reduce the selection bias to a good extent. Also, K-fold cross validation can be used with any modeling technique [11]. In this project we have used 3 different types of cross validation techniques to find the best optimal predictive model with optimal hyperparameters.

The first technique used is model validation technique from scikit-learn library called 'cross_val_score' to find the optimal value of alpha for the Linear Regression Ridge model.

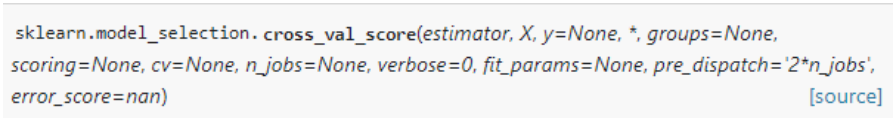


Fig.26: Code syntax for sklearn Cross Validation [12].

The second technique used is hyper-parameter optimizer called Randomized Search Cross Validation to find the best parameters for the random forest predictive model.

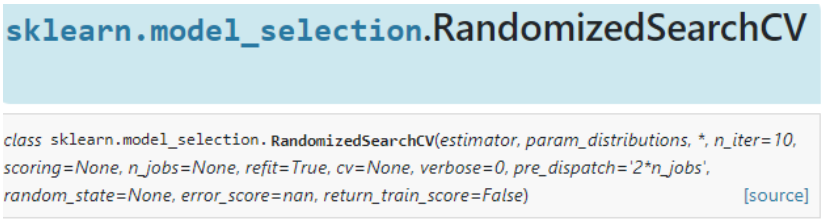


Fig.27: Code syntax for sklearn Random Search Cross Validation [13].

The third and final cross validation technique used is also a hyper-parameter optimizer called Grid Search Cross Validation to find the optimal number of hidden layers and other parameters of Neural Nets predictive model.

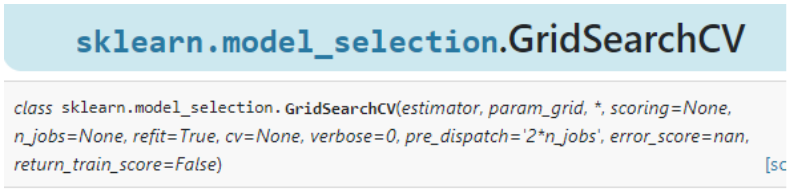


Fig.28: Code syntax for sklearn Grid Search Cross Validation [14].

7. Predictive Model Building

In this section we will see the four machine-learning techniques that we have built for predictive modeling to predict the bikes rented based on the available features. We will be studying different techniques like Linear Regression (Ridge Model), Clustering technique (K-Nearest Regressor), Ensemble technique called Random Forest and finally Deep learning i.e., Neural Networks and evaluate their performance and select the best technique.

7.1. Dataset Sampling Ratio

The Dataset containing 10348 records after data preprocessing is split into training and testing dataset using the 'sklearn.model_selection' library and its 'train_test_split' function into 75:25 ratio. 75% for training purpose on which cross validation is done and predictive model built and finally the last 25% for final prediction purpose.

7.2. Regression Model

The first model built is the Linear Regression model and to be precise 'Ridge Regression' a variation of Linear Regression. In order to have the balance of right bias and variance Ridge Regression is best preferred. In ridge regression we can tune the lambda parameter so that the model coefficients change **[15]**. Ridge regression is a special case of Tikhonov regularization in which all parameters are regularized equally. Ridge regression is particularly useful to mitigate the problem of multicollinearity in linear regression, which commonly occurs in models with large numbers of parameters. In general, the method provides improved efficiency in parameter estimation problems in exchange for a tolerable amount of bias-variance tradeoff. Along with this we have implemented two models, where one model has normalization and another standardization data preprocessing **[16]**.

Parameters considered in our model:
Alpha (α): Regularization strength; must be a positive float. Regularization improves the conditioning of the problem and reduces the variance of the estimates **[17]**.

To find the optimal value of alpha (α) which gives us the lowest error we have used k-Fold Cross Validation with an integral for loop that optimizes and runs the model to get the best value for (α).

The results are as follow:

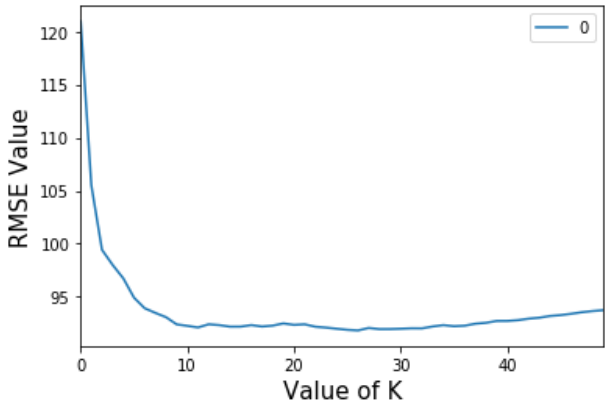
Ridge Regression Model	Normalization	Standardization
Value of Alpha (α)	1	41
MAE	70.76	70.32
MSE	9007.22	8948.82
RMSE	94.90	94.59

7.3.Clustering Model

KNN algorithm can be used for both classification and regression problems. The KNN algorithm uses 'feature similarity' to predict the values of any new data points. This means that the new point is assigned a value based on how closely it resembles the points in the training set **[18]**. The best choice of k depends upon the data; generally, larger values of k reduce effect of the noise on the classification but make boundaries between classes less distinct. A good k can be selected by various heuristic techniques (see hyperparameter optimization). The special case where the class is predicted to be the class of the closest training sample (i.e., when k = 1) is called the nearest neighbor algorithm.

Parameters considered in our model:
n_neighbors (k): Number of neighbors to use by default for kneighbors queries.

To find the optimal value of k which gives us the lowest error we have used a for loop that optimizes and runs the model to get the best value for (k) corresponding to the lowest error.




```
RMSE value for k=  24 is: 92.10298996594874
RMSE value for k=  25 is: 91.99609714919535
RMSE value for k=  26 is: 91.89613669519404
RMSE value for k=  27 is: 91.83549947982277
RMSE value for k=  28 is: 92.06779732561691
RMSE value for k=  29 is: 91.962069080297
RMSE value for k=  30 is: 91.96468416526425
RMSE value for k=  31 is: 91.99505505300243
RMSE value for k=  32 is: 92.03523968369531
```

Fig.29: Elbow Curve with RMSE value for corresponding K

The results are as follow:

k-nearest neighbor regressor	Values
Optimal value of k	27
MAE	63.67
MSE	8433.75
RMSE	91.83

7.4. Ensemble Model

For the next model we have implemented random forest machine learning technique. 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 combines multiple algorithm of the same type i.e., multiple decision trees, resulting in a forest of trees, hence the name "Random Forest". The random forest algorithm can be used for both regression and classification tasks **[19]**. We have created two separate models namely base model and hyperparameter tuned model to compare the results with each other.

In base model we have set the default hyperparameters and ran the model to get the preliminary base results. After that using Random Search Cross Validation we have created the hyperparameter tuned model.

 # Hyperparameter Tuning the Random Forest in Python

Random Search Cross Validation in Scikit-Learn [20].
We only have a vague idea of the best hyperparameters and thus the best approach to narrow our search is to evaluate a wide range of values for each hyperparameter.

Using Scikit-Learn's RandomizedSearchCV method, we can define a grid of hyperparameter ranges, and randomly sample from the grid, performing K-Fold CV with each combination of values.

- n_estimators = number of trees in the forest
- max_features = max number of features considered for splitting a node
- max_depth = max number of levels in each decision tree
- min_samples_split = min number of data points placed in a node before the node is split
- min_samples_leaf = min number of data points allowed in a leaf node
- bootstrap = method for sampling data points (with or without replacement)

```
{'bootstrap': [True, False],
 'max_depth': [10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, None],
 'max_features': ['auto', 'sqrt'],
 'min_samples_leaf': [1, 2, 4],
 'min_samples_split': [2, 5, 10],
 'n_estimators': [10, 31, 52, 73, 94, 115, 136, 157, 178, 200]}
```

Fig.30: RandomSearch Cross Validation Parameter Grid.

After the parameter grid has been created it is executed to get the optimal tuned hyperparameters for our model. The final parameters are:

The optimal Hyperparameters are:

```
rf_random.best_params_
```

```
{'n_estimators': 73,
 'min_samples_split': 10,
 'min_samples_leaf': 4,
 'max_features': 'sqrt',
 'max_depth': 50,
 'bootstrap': True}
```

Fig.31.: Optimal Hyperparameters for Random Forest Model.

The results are as follow:

Random Forest Model	Base Model	Hyperparameter Tuned
No of Decision Trees	10	73
Minimum Samples Split	2	10
Minimum Samples Leaf	1	4
Max Features	Auto	Square Root
Max Depth	None	50
MAE	67.02	62.06
MSE	9551.15	7966.47
RMSE	99.75	89.25

7.5. Neural Networks Model

We have built two predictive models using Multi-Layer Perceptron from sklearn library and Neural Network using Keras and TensorFlow. In MLP model we have built a base model and an optimized model using hyperparameter tuning by using Grid Search cross validation.

7.5.1. Multi-layer Perceptron regressor

Multi-layer Perceptron regressor model optimizes the squared-loss using LBFGS or stochastic gradient descent.

MLP using GridSearch Cross Validation
Hyper Parameter Tuning
We can calculate the best parameters for the model using "GridSearchCV". The input parameters for the GridSearchCV method are

- The MLP model
- A parameter dictionary in which we define various hidden layers, activation units, learning rates.

It trains the model and finds the best parameter.

BASE MODEL

```
MLPRegressor(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.9,
             beta_2=0.999, early_stopping=False, epsilon=1e-08,
             hidden_layer_sizes=(100,), learning_rate='constant',
             learning_rate_init=0.001, max_iter=100, momentum=0.9,
             n_iter_no_change=10, nesterovs_momentum=True, power_t=0.5,
             random_state=1, shuffle=True, solver='adam', tol=0.0001,
             validation_fraction=0.1, verbose=False, warm_start=False)
```

GridSearch Cross Validation Hyperparameter Tuning

```
MLPRegressor(activation='logistic', alpha=0.0001, batch_size='auto', beta_1=0.9,
             beta_2=0.999, early_stopping=False, epsilon=1e-08,
             hidden_layer_sizes=24, learning_rate='constant',
             learnirg_rate_init=0.001, max_iter=400, momentum=0.9,
             n_iter_no_change=10, nesterovs_momentum=True, power_t=0.5,
             random_state=1, shuffle=True, solver='sgd', tol=0.0001,
             validation_fraction=0.1, verbose=10, warm_start=False)
```

Fig.32.: Parameters used for MLP Regressor Model.

The results are as follow:

MLP Regressor Model	Base Model	Hyperparameter Tuned
Number of Layers	1	1
Number of Neurons	25	24
MAE	61.24	60.94
MSE	7916.41	7850.29
RMSE	88.97	88.60

7.5.2. Neural Networks (Keras Regressor)

Using Keras and TensorFlow we have built an ANN and used Keras regressor for the predictive model building. Just like MLP Regressor we have used GridSearch CV for finding the optimal number of hidden layers and the optimal number of neurons in each hidden layer.

```
[grid_result.best_score_, grid_result.best_params_]
```

```
[-62.239584915049065,
 {'activation': 'relu',
  'batch_size': 128,
  'epochs': 30,
  'layers': [1024, 256]}]
```

Fig.33.: Optimal hyper tuned parameters of Keras Regressor model.

We are not seeing any significant decrease in the errors and hence we can conclude that the MLP regressor is the best version of Neural Nets.

The results are as follow:

Keras Regressor Model	Values
Number of Hidden Layers	2
Number of Neurons	1024, 256
MAE	62.39
MSE	8266.63
RMSE	90.92

8. Results & Conclusion

The preliminary data analysis gives us a good explanation about the consumer behavior for bike rental purpose. There is nothing out of ordinary insights as we can clearly see the bike rental numbers are up in the peak hours during the day and night time as well as in favorable season and month. Good weather conditions demand more rentals. Which is expected and bad weather less.

After implementing four different forecasting models and applying different suitable cross validation techniques to get the optimal model with tuned hyperparameters we have definitely seen improvement in forecast accuracy and decrease in the errors model after model. The following table shows the decrease in prediction errors for all models.

	RIDGE	KNN	Random Forest	MLP Regressor	KERAS Regressor
MAE	**70.32**	**63.67**	**62.06**	**60.94**	**62.39**
MSE	**8948.82**	**8433.8**	**7966.47**	**7850.29**	**8266.63**
RMSE	**94.59**	**91.83**	**89.25**	**88.6**	**90.92**

Fig.34.: Performance Evaluation Criteria Comparison Table.

From the above table we can see that Ridge Regression model has performed the worst compared to other techniques and MLP regressor performed the best. Linear regression has its limitation, and the linear line is susceptible to outlier records. Though we have removed the outlier data in preprocessing phase, sometimes the linear model still cannot handle the spikes in the records. Compared to this K-nearest regressor performed better as it contains only one parameter and is easy to implement and intuitive.

Compared these two the ensemble learning technique i.e., Random forest gave us a much-improved forecast with huge decrease in errors as the algorithm is not biased and there are multiple trees, and each tree is trained on a subset of data. Basically, the random forest algorithm relies on the power of "the crowd"; therefore, the overall biasedness of the algorithm is reduced. The algorithm is very stable. Even if a new data point is introduced in the dataset the overall algorithm is not affected much since new data may impact one tree, but it is very hard for it to impact all the trees **[19]**.

The project clearly shows the prowess of Artificial Neural Networks as this technique gives us the best forecast and lowest error metrics. Where linear models fail neural networks solve those problems. MLP is flexible in addressing the non-linear shapes present in the data. We can say that the neural networks are typically black box in nature that is there is very less information and insights what happens in the hidden layers after the input is feed and output is obtained.

The following graphs show how the prediction line of each algorithm parses the actual data line, to get a clear understanding of the forecast being made.

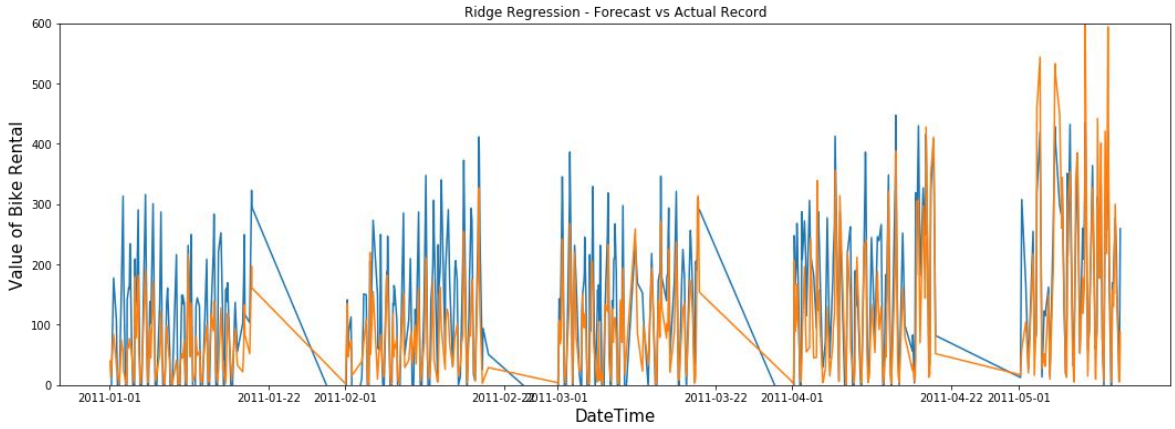


Fig.35.: Line Graph for Actual vs Forecast Values of Bike Rentals – Ridge Regression

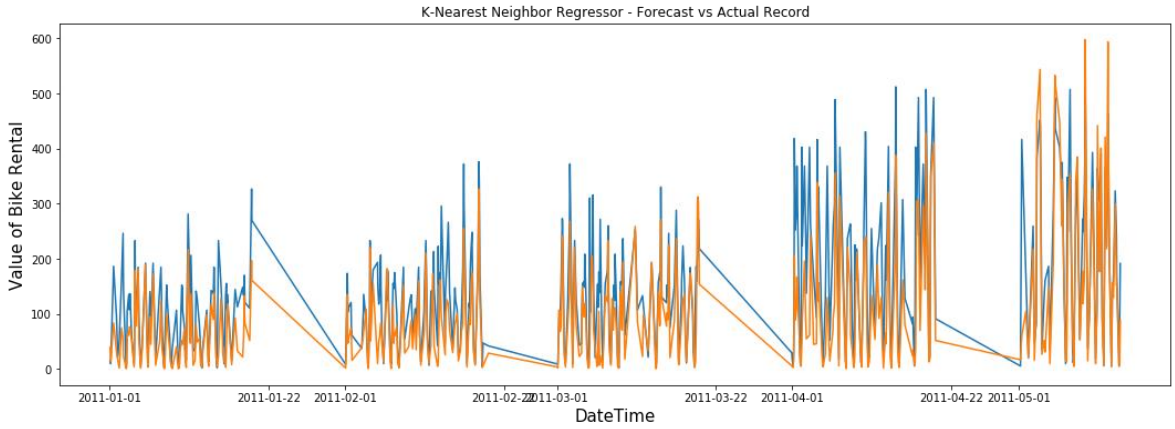


Fig.36.: Line Graph for Actual vs Forecast Values of Bike Rentals – KNN Regressor

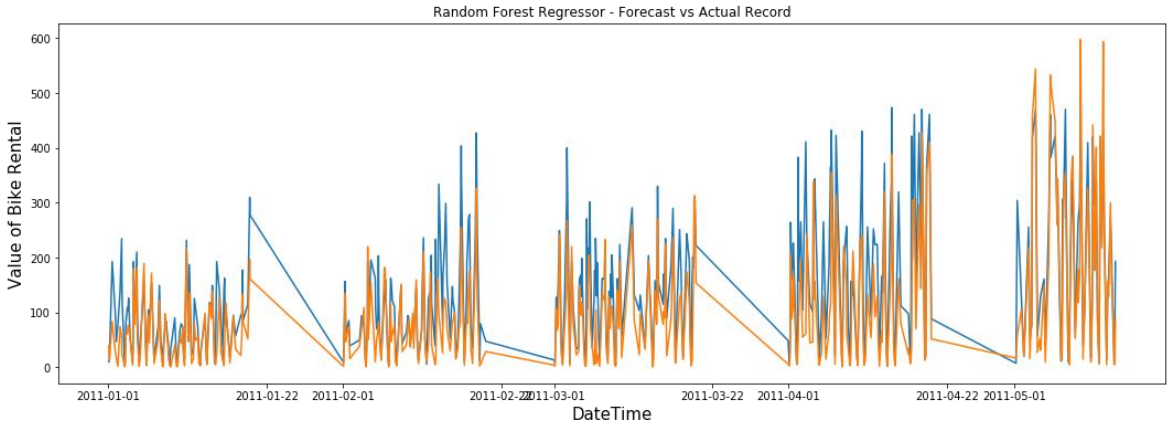


Fig.37.: Line Graph for Actual vs Forecast Values of Bike Rentals – Random Forest Regression

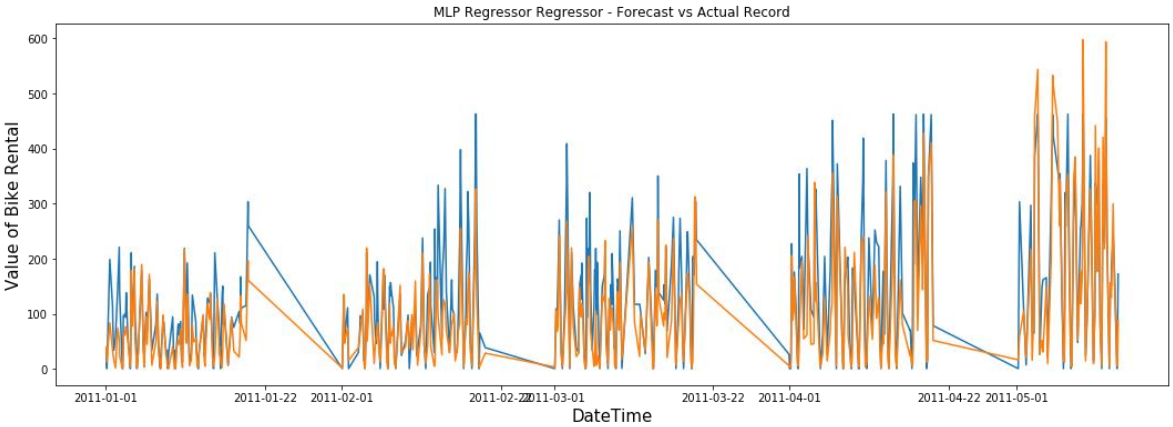


Fig.38.: Line Graph for Actual vs Forecast Values of Bike Rentals – MLP Regressor

From the above four figures we can clearly see that the MLP Regressor forecast line parses the actual data line more clearly, hence produces least errors.

To conclude, machine learning techniques have great potential and can be used to make accurate demands compared to the traditional spreadsheet and age-old forecasting methods. After analyzing and comparing the accuracy of the models the potential usefulness of those results can be studied and applied for future studies down the line. Forecast methods and particularly using machine learning techniques are seen very powerful and can be further used for various forecasts related to different operational functions of the bike sharing or rental system. For example, allocation of bike resources as per the peak demand at highly rented areas can be marked out and potential profits which could have been lost can be avoided, more-over with robust and accurate forecasts bike maintenance schedules can be improved and service supply chains can deliver significant competitive advantage. Hence along with mentioned benefits we can extract many other beneficial insights like reduced inventory investment for bikes, lowered operating costs, efficient service levels to customers and optimize the profit margins exponentially. The models have been tuned to find the optimal parameters and once optimized can be trained on historical data and deployed.

9. References

[1] W. Wang, "Forecasting Bik ecasting Bike Rental Demand Using New Y e Rental Demand Using New York Citi Bik ork Citi Bike Data," Technological University Dublin, Dublin, 2016.

[2] Y. Feng and S. Wang, "A forecast for bicycle rental demand based on random forests and multiple linear regression," in *2017 IEEE/ACIS 16th International Conference on Computer and Information Science (ICIS)*, Wuhan, China, 2017.

[3] O. p. Nekkanti, "PREDICTION OF RENTAL DEMAND FOR A BIKE-SHARE PROGRAM," North Dakota State University of Agriculture and Applied Science, Fargo, North Dakota, 2017.

[4] T. Bock, "https://www.displayr.com/," [Online]. Available: https://www.displayr.com/what-is-a-correlation-matrix/.

[5] G. Press, "Cleaning Big Data: Most Time-Consuming, Least Enjoyable Data Science Task, Survey Says," https://www.forbes.com/, 2015. [Online]. Available: https://www.forbes.com/sites/gilpress/2016/03/23/data-preparation-most-time-consuming-least-enjoyable-data-science-task-survey-says/?sh=a9280366f637.

[6] H. Singh, "Understanding Data Preprocessing," Towards Data Science, 13 May 2020. [Online]. Available: https://towardsdatascience.com/data-preprocessing-e2b0bed4c7fb#:~:text=Understanding%20Data%20Preprocessing&text=Data%20preprocessing%20is%20an%20important,understandable%20after%20performing%20data%20preprocessing.

[7] D. Kumar, "Introduction to Data Preprocessing in Machine Learning," Towards Data Science, 25 December 2018. [Online]. Available: https://towardsdatascience.com/introduction-to-data-preprocessing-in-machine-learning-a9fa83a5dc9d.

[8] J. Frost, "Guidelines for Removing and Handling Outliers in Data," Statistics By Jim, [Online]. Available: https://statisticsbyjim.com/basics/remove-outliers/#:~:text=Outliers%20are%20unusual%20values%20in,analyses%20and%20violate%20their%20assumptions.&text=Outliers%20increase%20the%20variability%20in,results%20to%20become%20statistically%20significant.

[9] A. BHANDARI, "Feature Scaling for Machine Learning: Understanding the Difference Between Normalization vs. Standardization," Analytics Vidhya, 3 April 2020. [Online]. Available: https://www.analyticsvidhya.com/blog/2020/04/feature-scaling-machine-learning-normalization-standardization/.

[10] A. Chugh, "MAE, MSE, RMSE, Coefficient of Determination, Adjusted R Squared — Which Metric is Better?," Medium, 8 December 2020. [Online]. Available: https://medium.com/analytics-vidhya/mae-mse-rmse-coefficient-of-determination-adjusted-r-squared-which-metric-is-better-cd0326a5697e.

[11] T. SRIVASTAVA, "11 Important Model Evaluation Metrics for Machine Learning Everyone should know," Analytics Vidhya, 6 August 2019. [Online]. Available: https://www.analyticsvidhya.com/blog/2019/08/11-important-model-evaluation-error-metrics/.

[12] scikit-learn.org, "sklearn.model_selection.cross_validate," [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.cross_validate.html.

[13] scikit-learn.org, "sklearn.model_selection.RandomizedSearchCV," [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.html.

[14] scikit-learn.org, "sklearn.model_selection.GridSearchCV," [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html.

[15] Qshick, "Ridge Regression for Better Usage," Towards Data Science, 2 January 2019. [Online]. Available: https://towardsdatascience.com/ridge-regression-for-better-usage-2f19b3a202db.

[16] Wikipedia, "Tikhonov regularization," [Online]. Available: https://en.wikipedia.org/wiki/Tikhonov_regularization.

[17] scikit-learn.org, "sklearn.linear_model.Ridge," [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Ridge.html.

[18] A. SINGH, "A Practical Introduction to K-Nearest Neighbors Algorithm for Regression (with Python code)," Analytics Vidhya, 22 August 2018. [Online]. Available: https://www.analyticsvidhya.com/blog/2018/08/k-nearest-neighbor-introduction-regression-python/.

[19] U. Malik, "Random Forest Algorithm with Python and Scikit-Learn," Stack Abuse, [Online]. Available: https://stackabuse.com/random-forest-algorithm-with-python-and-scikit-learn/.

[20] W. Koehrsen, "Hyperparameter Tuning the Random Forest in Python," Towards Data Science, 18 January 2018. [Online]. Available: https://towardsdatascience.com/hyperparameter-tuning-the-random-forest-in-python-using-scikit-learn-28d2aa77dd74.