PROJECT REPORT

On

BIKE RENTAL COUNT PREDICTION

Submitted by

Neeta Bharti

edWisor

25-08-2019

TABLE OF CONTENTS

Chapter 1
Introduction

1.1 Problem Statement

A bike rental is a bicycle business that rents bikes for short periods of time. Most rentals are provided by bike shops as a side line to their main businesses of sales and service, but some shops specialize in rentals. Bike rental shops rent by the day or week as well as by the hour, and these provide an excellent opportunity for people who don't have access to a vehicle, typically travellers and particularly tourists. Specialized bike rental shops thus typically operate at beaches, parks, or other locations that tourists frequent. In this case, the fees are set to encourage renting the bikes for a few hours at a time, rarely more than a day.

The objective of this Case is the Predication of bike rental count on daily based on the environmental and seasonal settings ,so that required bikes would be arranged and managed by the shops according to environmental and seasonal conditions.

1.2 Data

Our task is to build a regression model which will predict the bike rental count depending on multiple factors/variables. Given below is a sample of the data set that we are using to predict the bike rental count:

Table1.1: Bike Renting Sample data (Column:1-16)

	instant	dteday	season	yr	mnth	holiday	weekday	workingday	weathersit	temp	atemp	hum	windspeed	casual	registered	cnt
0	1	2011-01-01	1	0	1	0	6	0	2	0.344167	0.363625	0.805833	0.160446	331	654	985
1	2	2011-01-02	1	0	1	0	0	0	2	0.363478	0.353739	0.696087	0.248539	131	670	801
2	3	2011-01-03	1	0	1	0	1	1	1	0.196364	0.189405	0.437273	0.248309	120	1229	1349
3	4	2011-01-04	1	0	1	0	2	1	1	0.200000	0.212122	0.590435	0.160296	108	1454	1562
4	5	2011-01-05	1	0	1	0	3	1	1	0.226957	0.229270	0.436957	0.186900	82	1518	1600

As you can see in the table below we have the following 15 independent variables, using which we have to correctly predict the count of bike renting.

The details of data attributes in the dataset are as follows –

- instant: Record index
- dteday: Date
- season: Season (1:springer, 2:summer, 3:fall, 4:winter)
- yr: Year (0: 2011, 1:2012)
- mnth: Month (1 to 12)
- holiday: whether day is holiday or not (extracted from Holiday Schedule)
- weekday: Day of the week
- workingday: If day is neither weekend nor holiday is 1, otherwise is 0.
- weathersit: (extracted from Freemeteo)
  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: Normalized temperature in Celsius.
  The values are derived via (t-t_min)/(t_max-t_min),
  t_min=-8, t_max=+39 (only in hourly scale)
- atemp: Normalized feeling temperature in Celsius.
  The values are derived via (t-t_min)/(t_maxt_min),
  t_min=-16, t_max=+50 (only in hourly scale)
- hum: Normalized humidity. The values are divided to 100 (max)
- windspeed: Normalized wind speed. The values are divided to 67 (max)
- casual: count of casual users
- registered: count of registered users
- cnt: count of total rental bikes including both casual and registered

Chapter 2

Exploratory Data Analysis

Any predictive modelling requires that we look at the data before we start modelling. However, in data mining terms looking at data refers to so much more than just looking. Looking at data refers to exploring the data, cleaning the data as well as visualizing the data through graphs and plots. This is often called as Exploratory Data Analysis(EDA). Remember the quality of your inputs decide the quality of your output. So, once you have got your business hypothesis ready, it makes sense to spend lot of time and efforts here. During this project Data exploration, cleaning and preparation took up to 70% of my total project time.

Below are the steps involved to understand, clean and prepare the data for building the predictive model:

1. Variable Identification
2. Univariate Analysis
3. Bi-variate Analysis
4. Missing values treatment
5. Outlier treatment
6. Feature selection
7. Feature scaling
8. Variable transformation and dummy variable creation

2.1 Variable Identification

First, identify Predictor (Independent) and Target (dependent) variables. Next, identify the data type and category of the variables.

Table 2.1 : Independent Variables

S.No	Variables
1	instant
2	dteday
3	season
4	yr
5	mnth
6	holiday
7	weekday
8	workingday
9	weathersit
10	temp
11	atemp
12	hum
13	windspeed

Table 2.2 : Dependent Variables

S.No	Variables
1	registered
2	casual
3	cnt

- Cnt = Sum of rental bikes of casual and registered users (Target Variable)

Data type of the variables are as below:

```
bike_data.dtypes
```

```
instant          int64
dteday          object
season           int64
yr               int64
mnth             int64
holiday          int64
weekday          int64
workingday       int64
weathersit       int64
temp           float64
atemp          float64
hum            float64
windspeed      float64
casual           int64
registered       int64
cnt              int64
dtype: object
```

It is clear from data that the variables like ‘dteday’,’season’,’yr’,’mnth’,’workingday’, ’weathersit’ ’holiday,’weekday’are categorical data .So data types after converting those variables into category are as below

```
bike_data.dtypes
```

```
dteday         category
season         category
yr             category
mnth           category
holiday        category
weekday        category
workingday     category
weathersit     category
temp            float64
atemp           float64
hum             float64
windspeed       float64
cnt               int64
dtype: object
```

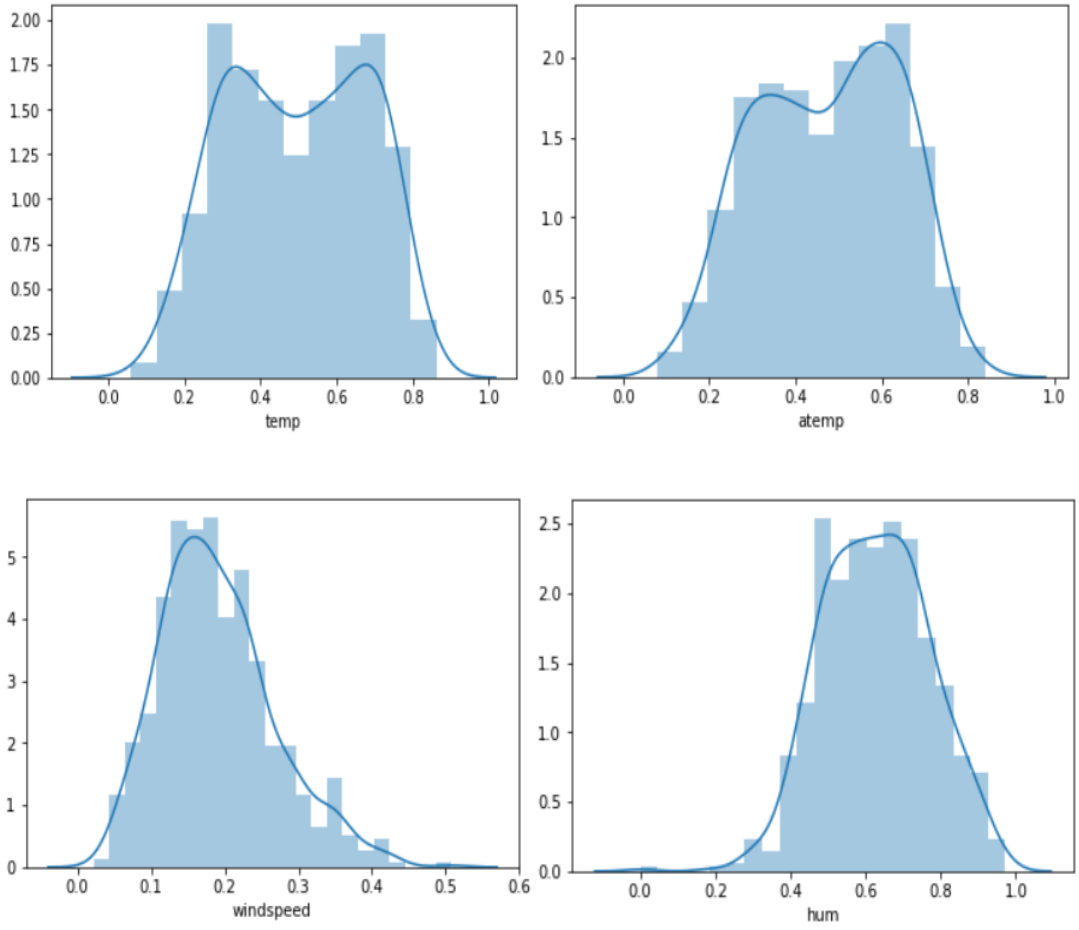
2.2 Univariate Analysis

At this stage, we explore variables one by one. Method to perform uni-variate analysis will depend on whether the variable type is categorical or continuous. Let's look at these methods and statistical measures for categorical and continuous variables individually:

Continuous Variables**:-** In case of continuous variables, we need to understand the central tendency and spread of the variable. These are measured using various statistical metrics visualization methods as shown below:

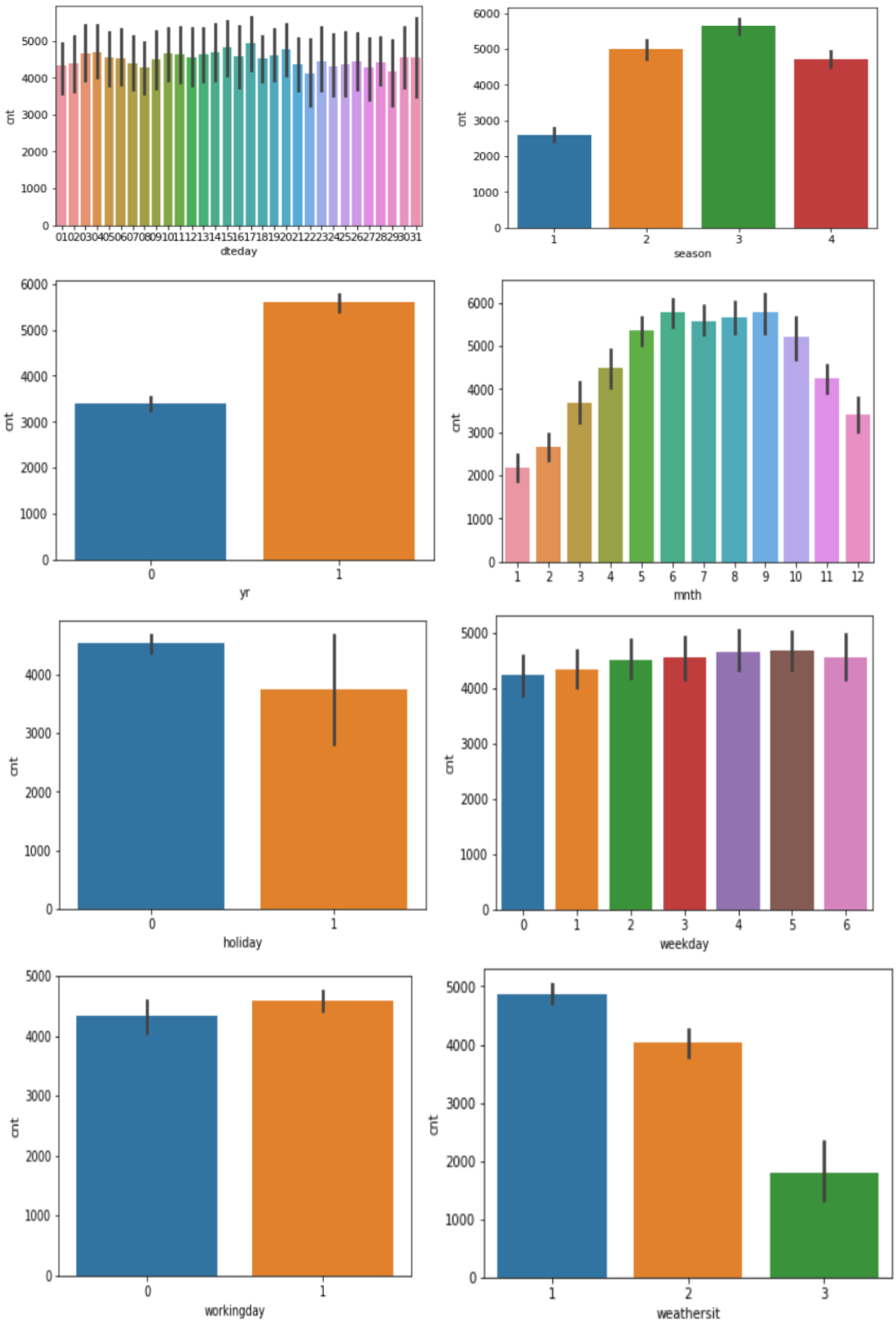
Categorical Variables:- For categorical variables, we'll use frequency table to understand distribution of each category. We can also read as percentage of values under each category. It can be be measured using two metrics, Count and Count% against each category. Bar chart can be used as visualization.

Fig: 2.1 Distribution plot of univariate set of observations(numerical variables)



Conclusion: All the numerical variables are somewhat normally distributed.

Fig 2.2 : Barplot of univariate set of observation(categorical variables)



Conclusion: Count of bike renting is more dependent on variables like weathersit ,yr, season, mnth than the variables like workingday, weekday, holiday, date.

2.3 Bi-variate Analysis

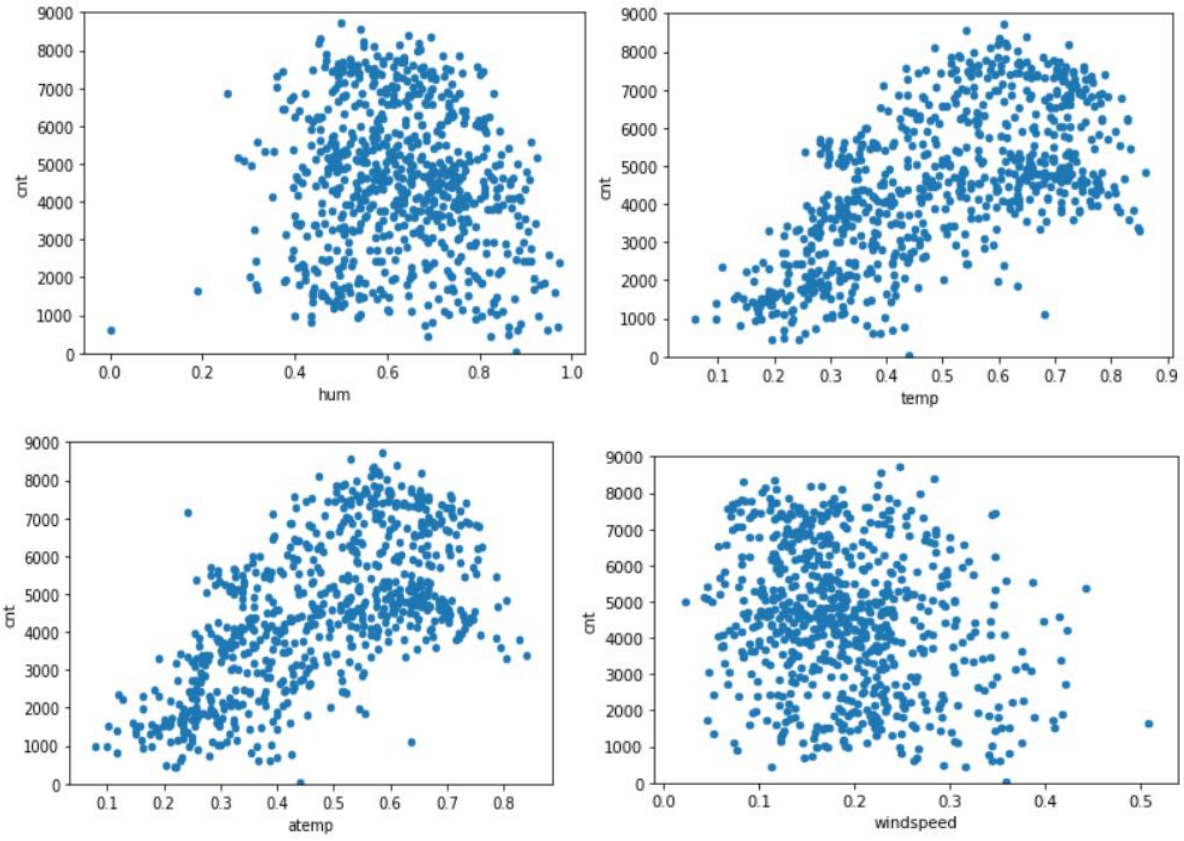
Bi-variate Analysis finds out the relationship between two variables. Here, we look for association and disassociation between variables at a pre-defined significance level. We can perform bi-variate analysis for any combination of categorical and continuous variables.
The combination can be: Categorical & Categorical, Categorical & Continuous and Continuous & Continuous. Different methods are used to tackle these combinations during analysis process. Let's understand the possible combinations in detail:

Continuous & Continuous**:** While doing bi-variate analysis between two continuous variables, we should look at scatter plot. It is a nifty way to find out the relationship between two variables. The pattern of scatter plot indicates the relationship between variables. The relationship can be linear or non-linear.

Scatter plot shows the relationship between two variable but does not indicates the strength of relationship amongst them. To find the strength of the relationship, we use Correlation. Correlation varies between -1 and +1.

- -1: perfect negative linear correlation
- +1:perfect positive linear correlation and
- 0: No correlation

Fig 2.3 : Scatter plot of Dependent vs Independent numerical variables



2.4 Missing Value Analysis

Missing data in the training data set can reduce the power / fit of a model or can lead to a biased model because we have not analysed the behaviour and relationship with other variables correctly. It can lead to wrong prediction or classification.

Data has missing values because of the following reason:

Let's identify the reasons for occurrence of the missing values. They may occur at two stages:

1. Data Extraction: It is possible that there are problems with extraction process. In such cases, we should double-check for correct data with data guardians. Some hashing procedures can also be used to make sure data extraction is correct. Errors at data extraction stage are typically easy to find and can be corrected easily as well.
2. Data collection: These errors occur at time of data collection and are harder to correct. They can be categorized in four types:

- Missing completely at random**:** This is a case when the probability of missing variable is same for all observations. For example: respondents of data collection process decide that they will declare their earning after tossing a fair coin. If an head occurs, respondent declares his / her earnings & vice versa. Here each observation has equal chance of missing value.
- Missing at random**:** This is a case when variable is missing at random and missing ratio varies for different values / level of other input variables. For example: We are collecting data for age and female has higher missing value compare to male.
- Missing that depends on unobserved predictors**:** This is a case when the missing values are not random and are related to the unobserved input variable. For example: In a medical study, if a particular diagnostic causes discomfort, then there is higher chance of drop out from the study. This missing value is not at random unless we have included "discomfort" as an input variable for all patients.
- Missing that depends on the missing value itself**:** This is a case when the probability of missing value is directly correlated with missing value itself. For example: People with higher or lower income are likely to provide non-response to their earning.

Methods to treat Missing value:

Missing values can be easily treated using various methods like mean, median method, knn method to impute missing value.

Missing Value in our dataset:

Code to get missing values is provided in Appendix.

cnt	0
windspeed	0
hum	0
atemp	0
temp	0
weathersit	0
workingday	0
weekday	0
holiday	0
mnth	0
yr	0
season	0
dteday	0

There is no missing value found in given dataset.

2.5 Outlier Analysis

Outlier is a commonly used terminology by analysts and data scientists as it needs close attention else it can result in wildly wrong estimations. Simply speaking, Outlier is an observation that appears far away and diverges from an overall pattern in a sample.

Methods to detect Outliers:

Most commonly used method to detect outliers is visualization. We use various visualization methods, like Box-plot, Histogram, Scatter Plot (above, we have used box plot and scatter plot for visualization). Some analysts also various thumb rules to detect outliers. Some of them are:

- Any value, which is beyond the range of -1.5 x IQR to 1.5 x IQR
- Use capping methods. Any value which out of range of 5th and 95th percentile can be considered as outlier
- Data points, three or more standard deviation away from mean are considered outlier
- Outlier detection is merely a special case of the examination of data for influential data points and it also depends on the business understanding

I have detected outliers using Box-plot. Below is the visualization:

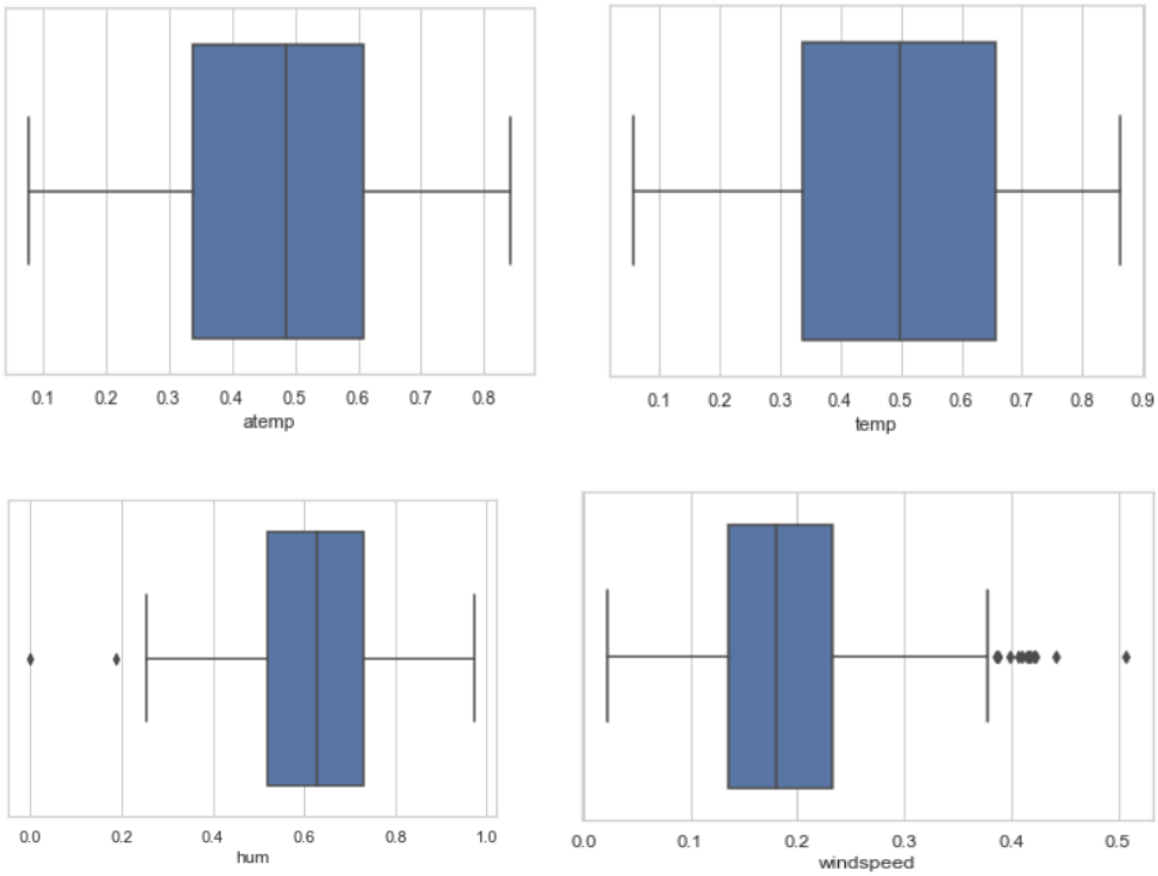


Fig 2.4 : Boxplot before Outlier Removal

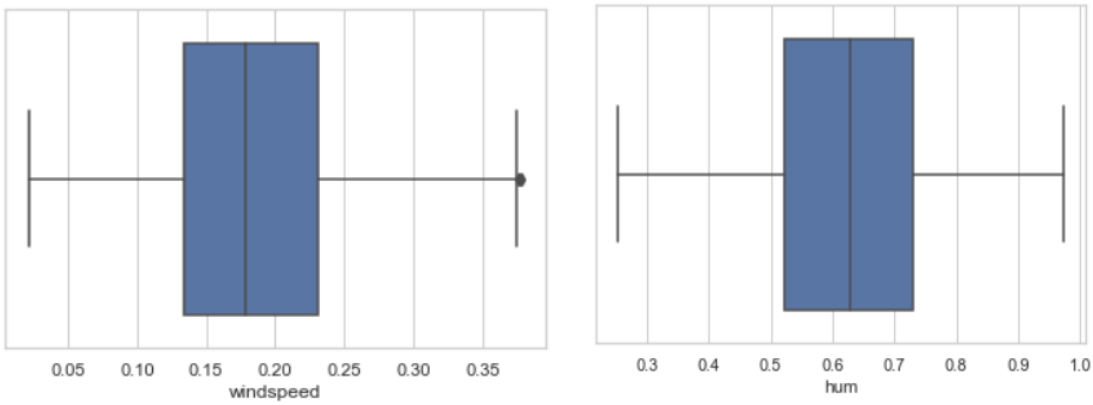


Fig 2.5 : Boxplot after removal of outliers

We can observe from the visualization that the variables like ‘windspeed’ and ‘humidity’ have outliers. So I have removed the outliers as below

Q1=first quartile
Q3=third quartile
IQR(Interquartile range)=Q3-Q1

A value is considered an outlier if it is less than 1.5 times the interquartile range below Q1 or more than 1.5 times the interquartile range above Q3.

Code for outliers removal is given in Appendix.

2.6 Feature selection

Feature Selection is the process where we automatically or manually select those features which contribute most to your prediction variable or output in which we are interested in.

Having irrelevant features in our data can decrease the accuracy of the models and make our model learn based on irrelevant features.
Some Feature selection techniques that are easy to use and also gives good results are as below:
1. Univariate Selection
2. Feature Importance
3.Correlation Matrix with Heatmap

1. Univariate Selection

Statistical tests can be used to select those features that have the strongest relationship with the output variable.The scikit-learn library provides the SelectKBest class that can be used with a suite of different statistical tests to select a specific number of features.

The example below uses the chi-squared (chi^2) statistical test for non-negative features to select 11 of the best features from the Bike Rent Prediction Dataset.

	variables	Importance Score
2	mnth	1268.982987
4	weekday	929.366575
3	holiday	655.928571
1	yr	351.049180
0	season	343.605595
5	workingday	219.773547
6	weathersit	144.588091
7	temp	47.612668
8	atemp	39.432053
9	hum	21.565272
10	windspeed	21.541518

3.Correlation Matrix with Heatmap

Correlation states how the features are related to each other or the target variable.Correlation can be positive (increase in one value of feature increases the value of the target variable) or

negative (increase in one value of feature decreases the value of the target variable).Heatmap makes it easy to identify which features are most related to the target variable, we will plot heatmap of correlated features using the seaborn library.

Below is the visualization that I have created:

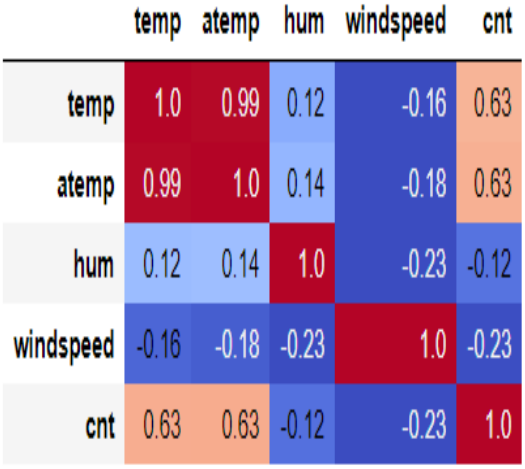


Fig 2.6 : Correlation map

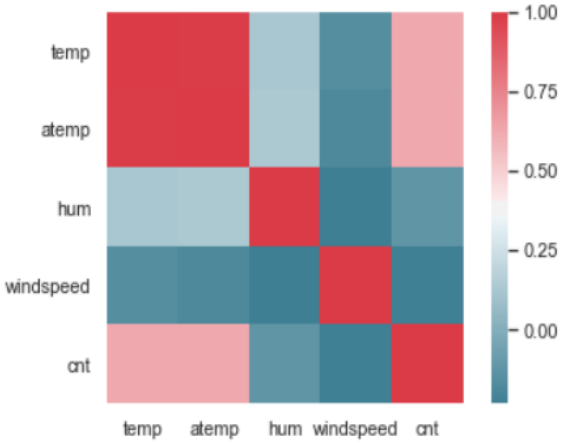


Fig 2.7 : Heat map

The above visualization shows that ‘temp’ and ‘atemp’ are highly correlated .Hence one variable is sufficient for our modelling. So I have dropped the variable ’atemp’.

2.7 Feature scaling

All the numerical variables like ‘temp’, ‘hum’ and ‘windspeed’ are already normalized value as described in the problem statement hence no need of scaling or normalization here.

2.8 Feature Engineering

Feature engineering is the science (and art) of extracting more information from existing data. You are not adding any new data here, but you are actually making the data you already have more useful.

Feature engineering itself can be divided in 2 steps:

2.8.1 Variable transformation
2.8.2 Variable creation

2.8.1 Variable Transformation

In data modelling, transformation refers to the replacement of a variable by a function. For instance, replacing a variable x by the square / cube root or logarithm x is a transformation. In other words, transformation is a process that changes the distribution or relationship of a variable with others.

2.8.2 Variable Creation

Feature / Variable creation is a process to generate a new variables / features based on existing variable(s). For example, say, we have date(dd-mm-yy) as an input variable in a data set. We can generate new variables like day, month, year, week, weekday that may have better relationship with target variable.

Creating dummy variables: One of the most common application of dummy variable is to convert categorical variable into numerical variables. Dummy variables are also called Indicator Variables. It is useful to take categorical variable as a predictor in statistical models.

Here I have created dummy variables for all the categorical data.

	temp	hum	windspeed	cnt	season_1	season_2	season_3	season_4	yr_0	yr_1
0	0.344167	0.805833	0.160446	985	1	0	0	0	1	0
1	0.363478	0.696087	0.248539	801	1	0	0	0	1	0
2	0.196364	0.437273	0.248309	1349	1	0	0	0	1	0
3	0.200000	0.590435	0.160296	1562	1	0	0	0	1	0
4	0.226957	0.436957	0.186900	1600	1	0	0	0	1	0

weekday_2	weekday_3	weekday_4	weekday_5	weekday_6	workingday_0	workingday_1	weathersit_1	weathersit_2	weathersit_3
0	0	0	0	1	1	0	0	1	0
0	0	0	0	0	1	0	0	1	0
0	0	0	0	0	0	1	1	0	0
1	0	0	0	0	0	1	1	0	0
0	1	0	0	0	0	1	1	0	0

Fig 2.8 : Dataset after dummy variable creation for all categorical data

Chapter 3

Modelling

3.1 Training and Testing Sets

There is one final step of data preparation: splitting data into training and testing sets. During training, we let the model 'see' the answers, in this case the actual bike rent count, so it can learn how to predict the bike rent count from the features. We expect there to be some relationship between all the features and the target value, and the model's job is to learn this relationship during training. Then, when it comes time to evaluate the model, we ask it to make predictions on a testing set where it only has access to the features (not the answers) Because we do have the actual answers for the test set, we can compare these predictions to the true value to judge how accurate the model is. Generally, when training a model, we randomly split the data into training and testing sets to get a representation of all data points .I am setting the random state to 60 which means the results will be the same each time I run the split for reproducible results.

3.2 Model Selection

In this case we have to predict the count of bike renting according to environmental and seasonal condition. So the target variable here is a continuous variable. For Continuous we can use various Regression models. Model having less error rate and more accuracy will be our final model. Models built are :

1. Decision tree
2. Random Forest
3. Linear regression

3.3 Decision tree

Decision tree regression observes features of an object and trains a model in the structure of a tree to predict data in the future to produce meaningful continuous output. Continuous output means that the output/result is not discrete, i.e., it is not represented just by a discrete, known set of numbers or values.

This model is also known a Decision tree for regression target variable. For this model we have divided the dataset into train and test part using random sampling. Where train contains 70%

data of data set and test contains 30% data and contains 36 variable where ‘cnt’ variable is the target variable.

Code for Model testing and prediction is given in Appendix.

3.4 Random Forest

A Random Forest is an ensemble technique capable of performing both regression and classification tasks with the use of multiple decision trees and a technique called Bootstrap Aggregation, commonly known as bagging. The basic idea behind this is to combine multiple decision trees in determining the final output rather than relying on individual decision trees.

In Random forest we have divided the dataset into train and test part using random sampling. For this model we have divided the dataset into train and test part using random sampling Where train contains 70% data of data set and test contains 30% data and contains 36 variable where ‘cnt’ variable is the target variable.

Code for Model testing and prediction is given in Appendix.

3.5 Linear regression

It is used to estimate real values (ex- bike rental counts.) based on continuous variable(s). Here, we establish relationship between independent and dependent variables by fitting a best line. This best fit line is known as regression line and represented by a linear equation

Y= a*X + b.

In this equation:

- Y – Dependent Variable
- a – Slope
- X – Independent variable
- b – Intercept

These coefficients a and b are derived based on minimizing the sum of squared difference of distance between data points and regression line.

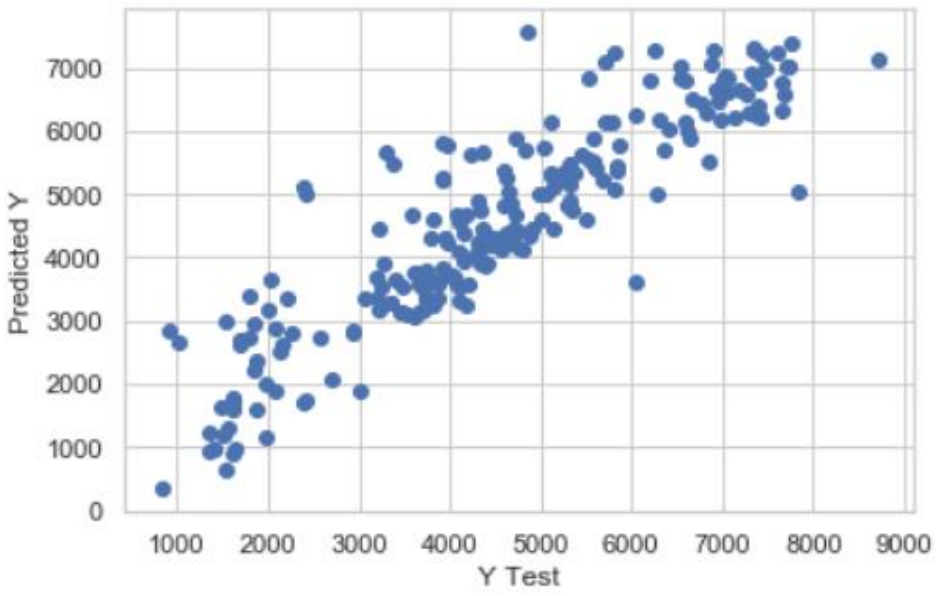
Linear Regression is mainly of two types: Simple Linear Regression and Multiple Linear Regression.

Simple Linear Regression is characterized by one independent variable. And, Multiple Linear Regression(as the name suggests) is characterized by multiple (more than 1) independent variables.

In my modelling I have used multiple linear regression as number of independent variables are more than one.

Code for Model testing and prediction is given in Appendix.

Below is the scatterplot of the real test values versus the predicted values.



In Linear regression we have divided the dataset into train and test part using random sampling. For this model we have divided the dataset into train and test part using random sampling Where train contains 70% data of data set and test contains 30% data and contains 36 variable where 'cnt' variable is the target variable.

Below are the coefficients of the model in linear regression :

```
Coefficients:
 [ 4.66478415e+03 -2.05719528e+03 -3.50904280e+03  1.20283045e+14
  1.20283045e+14  1.20283045e+14  1.20283045e+14 -7.76443835e+16
 -7.76443835e+16 -2.69015681e+15 -2.69015681e+15 -2.69015681e+15
 -2.69015681e+15 -2.69015681e+15 -2.69015681e+15 -2.69015681e+15
 -2.69015681e+15 -2.69015681e+15 -2.69015681e+15 -2.69015681e+15
 -2.69015681e+15 -2.08711030e+16 -1.07672819e+16  7.62691903e+15
 -2.47690208e+15 -2.47690208e+15 -2.47690208e+15 -2.47690208e+15
 -2.47690208e+15  7.62691903e+15 -5.05191056e+15  5.05191056e+15
  6.92601695e+15  6.92601695e+15  6.92601695e+15]
```

Chapter 4
Model Evaluation

Now that we have a few models for predicting the target variable, we need to decide which one to choose. There are several criteria that exist for evaluating and comparing models. We can compare the models using Predictive performance as the criteria to compare and evaluate models.

Predictive performance can be measured by comparing Predictions of the models with real values of the target variables, and calculating some average error measure.

Regression Metrics to evaluate our Model.

In this section will review 3 of the most common metrics for evaluating predictions on regression machine learning problems:

1. Mean Absolute Percentage Error (MAPE)
2. Mean Squared Error (MSE)
3. R^2 (R-square)

4.1 Mean Absolute Percentage Error (MAPE)

The mean absolute percentage error (MAPE) is a statistical measure of how accurate a forecast system is. It measures this accuracy as a percentage, and can be calculated as the average absolute percent error for each time period minus actual values divided by actual values. Where A_t is the actual value and F_t is the forecast value, this is given by:

$$\mathrm{M} = \frac{1}{n}\sum_{t=1}^{n}\left|\frac{A_t - F_t}{A_t}\right|$$

MAPE calculated for our model in Python is as below:

```
# MAPE with all variables
```

```
#Calculate MAPE
def MAPE(y_test, predictions_DT):
    mape = np.mean(np.abs((y_test - predictions_DT) / y_test))*100
    return mape
```

```
#MAPE for Decision tree model
print(MAPE(y_test, predictions_DT))
```

```
18.79115800450788
```

```
#MAPE for Random forest model
print(MAPE(y_test, predictions_RF))
```

```
13.365363239373215
```

```
#MAPE for Linear Regression model
print(MAPE(y_test, predictions_LR))
```

```
17.488732306566252
```

MAPE calculated for our model in Python with only important variables is as below:

```
# MAPE with only the important seven variables
```

```
#MAPE for Decision tree model
print(MAPE(y1_test, predictions_DT1))
```

```
19.88905509764415
```

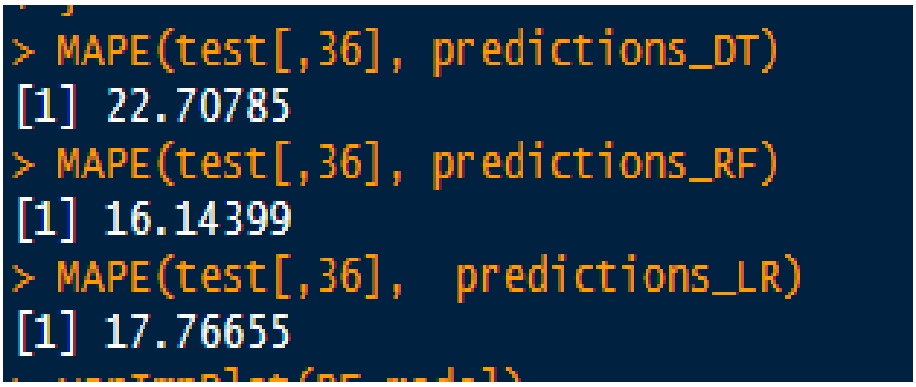
```
#MAPE for Random forest model
print(MAPE(y1_test, predictions_RF1))
```

```
14.762022151289331
```

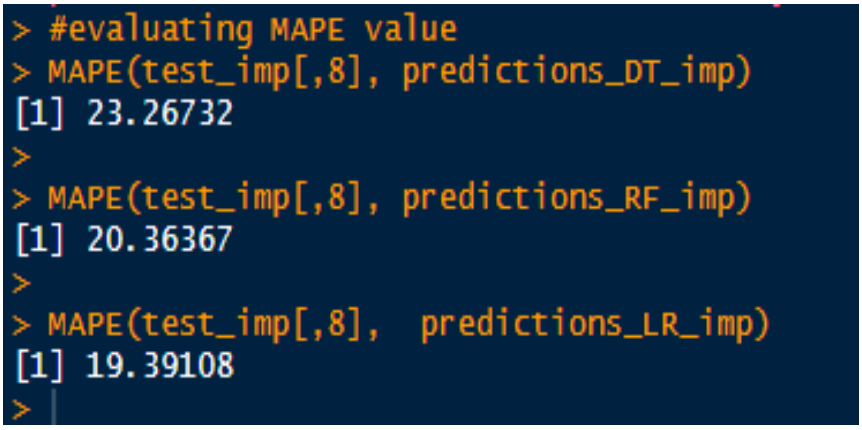
```
#MAPE for Linear Regression model
print(MAPE(y1_test, predictions_LR1))
```

```
19.650475354247654
```

MAPE calculated for our model in R is as below:



MAPE calculated for our model in R with only important variables is as below:



4.2 R-square value

The R^2 (or R Squared) metric provides an indication of the goodness of fit of a set of predictions to the actual values. In statistical literature, this measure is called the coefficient of determination.

This is a value between 0 and 1 for no-fit and perfect fit respectively.

R Squared value calculated for our model is as below:

```
#Evaluate R square value with all variables
```

```
from sklearn.metrics import r2_score
print(r2_score(y_test,predictions_DT))
from sklearn.metrics import r2_score
print(r2_score(y_test,predictions_RF))
from sklearn.metrics import r2_score
print(r2_score(y_test,predictions_LR))
```

```
0.7384058712544308
0.8774849112969362
0.805628391904126
```

```
#Evaluate R square value with important variables
```

```
from sklearn.metrics import r2_score
print(r2_score(y1_test,predictions_DT1))
from sklearn.metrics import r2_score
print(r2_score(y1_test,predictions_RF1))
from sklearn.metrics import r2_score
print(r2_score(y1_test,predictions_LR1))
```

```
0.743451878393203
0.8544532376582687
0.7577338164217754
```

4.3 Best Model Selection

We can see that from both R and Python Random forest model performs best out of Decision tree and linear regression. So random forest model is selected with 84% accuracy in R and with 86% accuracy in python.

Extracted predicted value of random forest model are saved with .csv file format in R and .xlsx file format in Python which is our Output sample.

Chapter 5
Visualization

We want to believe that “the data speaks for itself” . However, data visualization is an essential tool in a data scientist’s toolbox. Data visualization allows us to see patterns that would be invisible or inefficiently visible from looking at numbers alone. It can help us more efficiently make sense of data, facilitating good decision-making. This is true throughout the data science workflow. Beyond exploratory data analysis, data visualization can help us build better statistical and machine learning models, and it can speed up our understanding of the patterns in our data. And ultimately, there is no point in doing good science if you’re not able to communicate the results, whether you are a natural scientist contributing to the foundation of scientific understanding or a data scientist making insight actionable.

Earlier I have shown many visualization and below are some more visualizations to show the result:

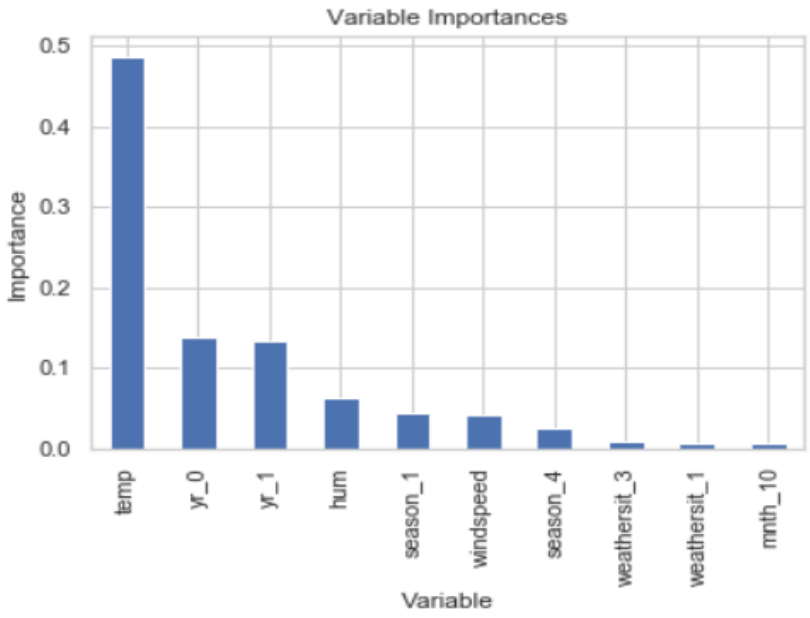


Fig 5.1 : Variable Importance in the RF model

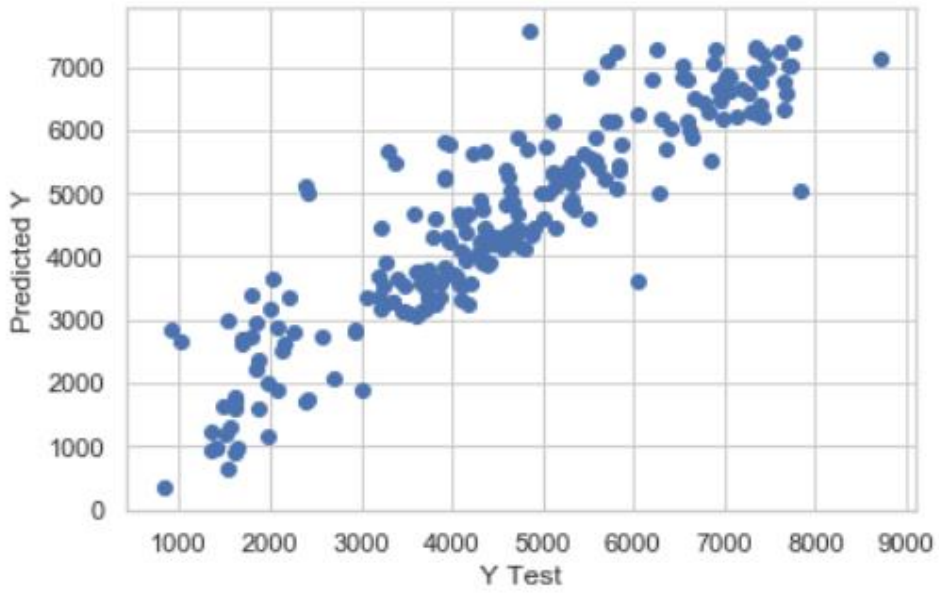


Fig 5.2 : Scatterplot to visualize Predicted_Y and Actual_Y

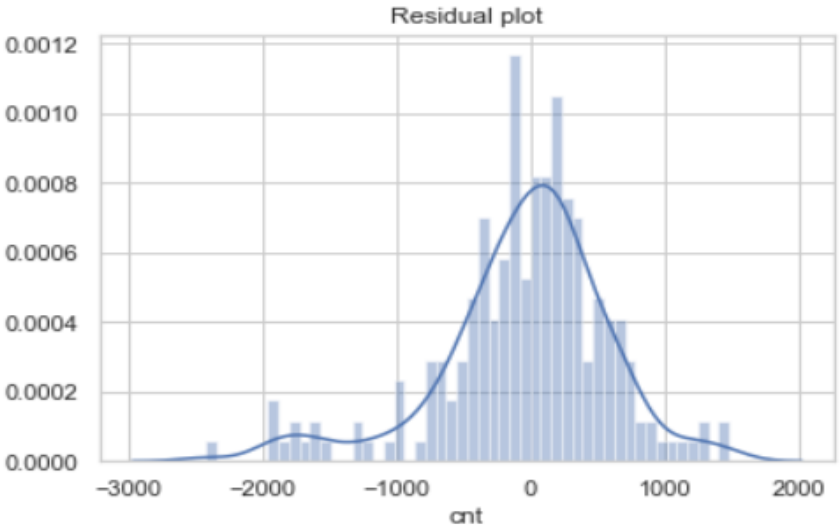


Fig 5.3 : Residual plot for the regression model

Chapter 6
Conclusion

After the data pre-processing ,model selection and model evaluation for the Bike Rent count project finally we can conclude below points:

- The random forest is the best method for our model as it has lowest MAPE(good accuracy) and good R square value .

- We can also conclude that the variables like 'temp' , 'year', 'hum', ' windspeed', 'season' alone can be a good predictor variable for the bike rent count.

- We can also conclude that our model is accurate model /good model as the residual plot of the model is normally distributed.

- We can conclude that the accuracy of Random Forest model with all (36) variables is 87% and the accuracy with only seven important variable is 86%.

- This tells us that we actually do not need all the data we collected to make accurate predictions. If we were to continue using this model, we could only collect those important variables and achieve nearly the same performance.

Appendix A- Python code

```
# importing libraries
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import datetime as dt
from  matplotlib import pyplot

#set working directory
os.chdir("C:/Users/NANINE/Desktop/EdWisor")

#loading dataset
bike_data = pd.read_csv("day.csv")
bike_data.head()

#Data pre processing
bike_data.shape
bike_data.dtypes
bike_data['season']= bike_data['season'].astype('category')
bike_data['yr']=bike_data['yr'].astype('category')
bike_data['mnth']=bike_data['mnth'].astype('category')
bike_data['holiday']=bike_data['holiday'].astype('category')
bike_data['workingday']=bike_data['workingday'].astype('category')
bike_data['weekday']=bike_data['weekday'].astype('category')
bike_data['weathersit']=bike_data['weathersit'].astype('category')
bike_data['dteday']=bike_data['dteday'].astype('category')
bike_data.dtypes
bike_data = bike_data.drop(['instant','dteday','casual', 'registered'], axis=1)
bike_data.head()
missing_values = bike_data.isnull().sum().sort_values(ascending=False)
missing_values
sns.distplot(bike_data['temp']);
sns.distplot(bike_data['atemp']);
sns.distplot(bike_data['hum']);
```

```
sns.distplot(bike_data['windspeed']);
sns.barplot(x='season',y='cnt',data=bike_data)
sns.barplot(x='yr',y='cnt',data=bike_data)
sns.barplot(x='mnth',y='cnt',data=bike_data)
sns.barplot(x='holiday',y='cnt',data=bike_data)
sns.barplot(x='weekday',y='cnt',data=bike_data)
sns.barplot(x='workingday',y='cnt',data=bike_data)
sns.barplot(x='weathersit',y='cnt',data=bike_data)
bike_data.plot.scatter(x='temp', y='cnt', ylim=(0,9000));
bike_data.plot.scatter(x='atemp', y='cnt', ylim=(0,9000));
bike_data.plot.scatter(x='hum', y='cnt', ylim=(0,9000));
bike_data.plot.scatter(x='windspeed', y='cnt', ylim=(0,9000));
sns.set(style="whitegrid")
ax = sns.boxplot(x=bike_data['temp'],orient ='h')
ax = sns.boxplot(x=bike_data['atemp'],orient ='h')
ax = sns.boxplot(x=bike_data['hum'],orient ='h')
ax = sns.boxplot(x=bike_data['windspeed'],orient ='h')
cnames = ['windspeed']
for i in cnames:
    Q3, Q1 = np.percentile(bike_data.loc[:,i], [75 ,25])
    IQR = Q3 - Q1

min = Q1 - (IQR*1.5)
max = Q3 + (IQR*1.5)

print(min)
print(max)

bike_data_out = bike_data.copy()

bike_data_out = bike_data_out.drop(bike_data_out[bike_data_out.loc[:,i] < min].index)
bike_data_out = bike_data_out.drop(bike_data_out[bike_data_out.loc[:,i] > max].index)
sns.set(style="whitegrid")
ax = sns.boxplot(x=bike_data_out['windspeed'],orient ='h')
cnames = ['hum']
for i in cnames:
    Q3, Q1 = np.percentile(bike_data.loc[:,i], [75 ,25])
    IQR = Q3 - Q1

min = Q1 - (IQR*1.5)
max = Q3 + (IQR*1.5)

print(min)
print(max)

bike_data_out = bike_data.copy()
```

```
bike_data_out = bike_data_out.drop(bike_data_out[bike_data_out.loc[:,i] < min].index)
bike_data_out = bike_data_out.drop(bike_data_out[bike_data_out.loc[:,i] > max].index)
sns.set(style="whitegrid")
ax = sns.boxplot(x=bike_data_out['hum'],orient ='h')
bike_data_out_num.corr(method='pearson').style.format("{:.2}").background_gradient(cmap
=plt.get_cmap('coolwarm'), axis=1)
corr = bike_data_out_num.corr()
sns.heatmap(corr, mask=np.zeros_like(corr, dtype=np.bool), cmap=sns.diverging_palette(220
, 10, as_cmap=True),square=True)
import pandas as pd
import numpy as np
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
X = bike_data_out.drop('cnt',axis=1)
y = bike_data_out['cnt']
bestfeatures = SelectKBest(score_func=chi2, k=10)
fit = bestfeatures.fit(X,y)
dfscores = pd.DataFrame(fit.scores_)
dfcolumns = pd.DataFrame(X.columns)
featureScores = pd.concat([dfcolumns,dfscores],axis=1)
featureScores.columns = ['variables','Importance Score']
print(featureScores.nlargest(12,'Importance Score'))
bike_data_new = bike_data_out.drop(['atemp'], axis=1)
cat_feats=['season','yr','mnth','holiday','weekday','workingday','weathersit']
final_data = pd.get_dummies(bike_data_new,columns=cat_feats)
from sklearn.model_selection import train_test_split
X = final_data.drop('cnt',axis=1)
y = final_data['cnt']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=101)

#Model selection
from sklearn.tree import DecisionTreeRegressor

dtree = DecisionTreeRegressor()

dtree.fit(X_train,y_train)

predictions_DT = dtree.predict(X_test)
from sklearn.ensemble import RandomForestRegressor

rfr = RandomForestRegressor(n_estimators=600,random_state=101)

rfr.fit(X_train,y_train)

predictions_RF = rfr.predict(X_test)

from sklearn.linear_model import LinearRegression
lm = LinearRegression()

lm.fit(X_train,y_train)

LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

```
print('Coefficients: \n', lm.coef_)
predictions_LR = lm.predict(X_test)

plt.scatter(y_test,predictions_LR)
plt.xlabel('Y Test')
plt.ylabel('Predicted Y')

print(rfr.feature_importances_)
feat_importances = pd.Series(rfr.feature_importances_, index=X.columns)
feat_importances.nlargest(10).plot(kind='bar')
plt.ylabel('Importance'); plt.xlabel('Variable'); plt.title('Variable Importances');
plt.show()
imp_data=final_data[['temp','yr_0','yr_1','hum','season_1','windspeed','season_4','cnt']]
X1 = imp_data.drop('cnt',axis=1)
y1 = imp_data['cnt']
X1_train, X1_test, y1_train, y1_test = train_test_split(X1, y1, test_size=0.30, random_state=
101)

from sklearn.tree import DecisionTreeRegressor

d1tree = DecisionTreeRegressor()

d1tree.fit(X1_train,y1_train)

predictions_DT1 = d1tree.predict(X1_test)

rfc1 = RandomForestRegressor(n_estimators=600, random_state=101)

rfc1.fit(X1_train,y1_train)

predictions_RF1 = rfc1.predict(X1_test)

from sklearn.linear_model import LinearRegression

lm1 = LinearRegression()

lm1.fit(X1_train,y1_train)

LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
print('Coefficients: \n', lm1.coef_)

predictions_LR1 = lm1.predict( X1_test)

plt.scatter(y1_test,predictions_LR1)
plt.xlabel('Y Test')
plt.ylabel('Predicted Y')
```

#model evaluation

```
def MAPE(y_test, predictions_DT):
    mape = np.mean(np.abs((y_test - predictions_DT) / y_test))*100
    return mape
print(MAPE(y_test, predictions_DT))
```

```
print(MAPE(y_test, predictions_RF))
print(MAPE(y_test, predictions_LR))
from sklearn.metrics import r2_score
print(r2_score(y_test,predictions_DT))
from sklearn.metrics import r2_score
print(r2_score(y_test,predictions_RF))
from sklearn.metrics import r2_score
print(r2_score(y_test,predictions_LR))
sns.distplot((y1_test-predictions_RF1),bins=50);
plt.title('Residual plot');
predicted_output=pd.DataFrame(predictions_RF1.round(0))
predicted_output.rename(columns={0:'Predicted_cnt'},inplace=True)
predicted_output.head()
predicted_output.reset_index(0,inplace=True)
y1_test_df=pd.DataFrame(y1_test)
y1_test_df.head()
input_df_bike_cnt=X1_test.join(y1_test_df,how='inner')
input_df_bike_cnt.head()
input_df_bike_cnt.reset_index(0,inplace=True)
sample_output=input_df_bike_cnt.join(predicted_output,how='inner',lsuffix='_left',rsuffix='_
right')
sample_output.drop(['index_left', 'index_right'],axis=1, inplace=True)
sample_output.to_excel('sample output.xlsx', index=False)
```

Appendix B- R code

```
#set working directory and load libraries
rm(list=ls(all=T))
setwd("C:/Users/NANINE/Desktop/EdWisor")
x=c("ggplot2", "corrgram", "DMwR", "caret", "randomForest", "unbalanced", "C50", "dummies", "e1071", "Information","MASS", "rpart", "gbm", "ROSE", 'sampling', 'DataCombine', 'inTrees','fastDummies')
lapply(x, require, character.only = TRUE)
rm(x)
#data pre processing
bike_data = read.csv("day.csv", header = T, na.strings = c(" ", "", "NA"))
View(bike_data)
bike_data$season=as.factor(bike_data$season)
bike_data$mnth=as.factor(bike_data$mnth)
bike_data$yr=as.factor(bike_data$yr)
bike_data$holiday=as.factor(bike_data$holiday)
bike_data$weekday=as.factor(bike_data$weekday)
bike_data$workingday=as.factor(bike_data$workingday)
bike_data$weathersit=as.factor(bike_data$weathersit)
bike_data=subset(bike_data,select = -c(instant,casual,registered,dteday))
str(bike_data)
View(bike_data)
missing_val = data.frame(apply(bike_data,2,function(x){sum(is.na(x))}))
missing_val
numeric_index = sapply(bike_data,is.numeric) #selecting only numeric
numeric_data = bike_data[,numeric_index]
cnames = colnames(numeric_data)
for (i in 1:length(cnames))
{assign(paste0("gn",i), ggplot(aes_string(y = (cnames[i]), x = "cnt"), data = subset(bike_data))+
        stat_boxplot(geom = "errorbar", width = 0.5) +
        geom_boxplot(outlier.colour="red", fill = "blue" ,outlier.shape=18,
                     outlier.size=1, notch=FALSE) +
        theme(legend.position="bottom")+
        labs(y=cnames[i],x="cnt")+
        ggtitle(paste("Box plot of count for",cnames[i])))}
gridExtra::grid.arrange(gn1,gn2,ncol=3)
gridExtra::grid.arrange(gn3,gn4,ncol=2)
corrgram(bike_data[,numeric_index], order = F, upper.panel=panel.pie, text.panel=panel.txt, main = "Correlation Plot")
bike_data= subset(bike_data,select = -c(atemp))
cnames= c("season","yr", "mnth","holiday","workingday","weekday","weathersit")
bike_data_new=bike_data[,cnames]
```

```
cnt=data.frame(bike_data$cnt)
names(cnt)[1]="cnt"
bike_data_new <- fastDummies::dummy_cols(bike_data_new)
bike_data_new=subset(bike_data_new,select=c(season,yr,mnth,holiday,workingday,weekday
,weathersit))
d3 = cbind(bike_data_new,bike_data)
d3= subset(d3,select = -c(season,yr,mnth,holiday,workingday,weekday,weathersit,cnt))
bike_data_new=cbind(d3,cnt)
```

#model selection

```
set.seed(1234)
train_index = sample(1:nrow(bike_data_new), 0.7 * nrow(bike_data_new))
train = bike_data_new[train_index,]
test = bike_data_new[-train_index,]
fit = rpart(cnt ~ ., data = train, method = "anova")
predictions_DT = predict(fit, test[,-36])
RF_model = randomForest(cnt ~ ., train, importance = TRUE, ntree = 200)
predictions_RF = predict(RF_model, test[,-36])
plot(RF_model)
lm_model = lm(cnt ~., data = train)
predictions_LR = predict(lm_model,test[,-36])
plot(lm_model)
```

#model evaluation

```
MAPE = function(y, yhat){
  mean(abs((y - yhat)/y))*100}
MAPE(test[,36], predictions_DT)
MAPE(test[,36], predictions_RF)
MAPE(test[,36],  predictions_LR)
varImpPlot(RF_model)
c_imp= c("temp","yr_0", "season_1","yr_1","hum","windspeed","mnth_1","cnt")
data_imp=bike_data_new[,c_imp]
View(data_imp)
set.seed(1234)
train_index_imp = sample(1:nrow(data_imp), 0.7 * nrow(data_imp))
train_imp = data_imp[train_index_imp,]
test_imp = data_imp[-train_index_imp,]
fit_imp = rpart(cnt ~ ., data = train_imp, method = "anova")
predictions_DT_imp = predict(fit_imp, test_imp[,-8])
RF_model_imp = randomForest(cnt ~ ., train_imp, importance = TRUE, ntree = 200)
predictions_RF_imp = predict(RF_model_imp, test_imp[,-8])
lm_model_imp = lm(cnt ~., data = train_imp)
predictions_LR_imp = predict(lm_model_imp,test_imp[,-8])
MAPE(test_imp[,8], predictions_DT_imp)
MAPE(test_imp[,8], predictions_RF_imp)
MAPE(test_imp[,8],  predictions_LR_imp)
```

References

- https://learning.edwisor.com/
- https://www.analyticsvidhya.com/
- https://www.coursera.org/learn/machine-learning
- https://ieeexplore.ieee.org/document/7959977
- https://library.ndsu.edu
- https://www.analyticsvidhya.com/blog/2016/07/practical-guide-data-preprocessing-python-scikit-learn/
- https://www.analyticsvidhya.com/blog/2016/01/guide-data-exploration/