

Project Name: Bike Rental Prediction

By: Garima

Date: 15 January 2018



| S.No. | Contents | Page no. |
|--------------|------------------------|-----------------|
| 1. | Introduction | |
| 1.1 | Problem Statement | 3 |
| 1.2 | Data | 3-4 |
| 2 | Methodology | |
| 2.1 | Missing Value Analysis | 5 |
| 2.2 | Data Visualisation | 6-10 |
| 2.3 | Outlier Analysis | 11-12 |
| 2.4 | Feature Scaling | 13 |
| 2.5 | Feature Selection | 13-15 |
| 2.6 | Data Modelling | 16-20 |
| 3 | Conclusion | |
| 3.1 | Model Evaluation | 21 |
| 4. | References | 22 |

Chapter 1

Introduction

1.1 Problem Statement :

Bike sharing has been gaining popularity all over the world. People can rent bike from one location and return at other. Thus such data can be captured to understand the mobility of bikes in the city and manage inventory better. We can thus construct a business model out of it by understanding the end user behaviour and usage pattern to maximise the organisation's profit.

In this problem statement our task is to predict bike rental count based on the historical data given for the year 2011 and 2012 .The usage pattern is affected by weather variables.

1.2 Data :

Dataset Details:

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

Table 1.1 Data set feature description

| S.no. | Feature | Description |
|-------|-------------|--|
| i. | instant: | Record index |
| ii. | dteday: | Date |
| iii. | season: | Season (1:springer, 2:summer, 3:fall, 4:winter) |
| iv. | yr: | Year (0: 2011, 1:2012) |
| v. | mnth: | Month (1 to 12) |
| vi. | holiday: | weather day is holiday or not (extracted from Holiday Schedule) |
| vii. | Weekday: | Day of the week |
| viii. | workingday: | If day is neither weekend nor holiday is 1, otherwise is 0. |
| ix. | weathersit: | 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 |
| x. | 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) |
| xi. | atemp: | Normalized feeling temperature in Celsius. The values are derived via $(t - t_{\min}) / (t_{\max} - t_{\min})$, $t_{\min} = -16$, $t_{\max} = +50$ (only in hourly scale) |
| xii. | Hum: | Normalized humidity. The values are divided to 100 (max) |
| xiii. | windspeed: | Normalized wind speed. The values are divided to 67 (max) |
| xiv. | casual: | count of casual users |
| xv. | registered: | count of registered users |
| xvi. | Count: | count of total rental bikes including both casual and registered |

Dimension of the data set (including the target variable): **731X16**

Table 1.2 daily data_set

| | 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 |
| 5 | 6 | 2011-01-06 | 1 | 0 | 1 | 0 | 4 | 1 | 1 | 0.204348 | 0.233209 | 0.518261 | 0.089565 | 88 | 1518 | 1606 |
| 6 | 7 | 2011-01-07 | 1 | 0 | 1 | 0 | 5 | 1 | 2 | 0.196522 | 0.208839 | 0.498696 | 0.168726 | 148 | 1362 | 1510 |
| 7 | 8 | 2011-01-08 | 1 | 0 | 1 | 0 | 6 | 0 | 2 | 0.165000 | 0.162254 | 0.535833 | 0.266804 | 68 | 891 | 959 |
| 8 | 9 | 2011-01-09 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0.138333 | 0.116175 | 0.434167 | 0.361950 | 54 | 768 | 822 |
| 9 | 10 | 2011-01-10 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0.150833 | 0.150888 | 0.482917 | 0.223267 | 41 | 1280 | 1321 |

Chapter 2

Methodology

On analysing the data set at hand, we see that :

- feature “*instant*” is just a serial number and does not contribute any information needed for predicting bike count.
- From feature “*dteday*”, we see that most of the useful information like year, month, weekday, holiday etc. are already extracted.
- Further, we see that features like “*casual*” and “*registered*” are leaky variables as it is given that total count is sum of these two variables.

Hence, before proceeding further we drop these 4 variables.

For simplicity of further operations, we categorise our data set features into numerical and categorical feature set consisting of following features in the data:

- **Numerical features set** : "cnt","temp","atemp","hum","windspeed"
- **Categorical features set** :
"season","yr","mnth","holiday","weekday","workingday","weathersit"

2.1 Missing Value Analysis:

As part of pre-processing step, we first check if there is any missing value present in our data set. Values can be missing from the data for various reasons like data entry error, no records available or at times there can even be some interesting behaviour hidden within the missing values. But when it comes to data modelling, generally algorithms would require a cleaner data, hence it is recommended to apply missing value analysis by imputing them with mean/median/mode or knn imputation methods and check the amount of variance explained by the data set before and after imputation. From the table below, we can see that our data set is free from missing value. Good for us!

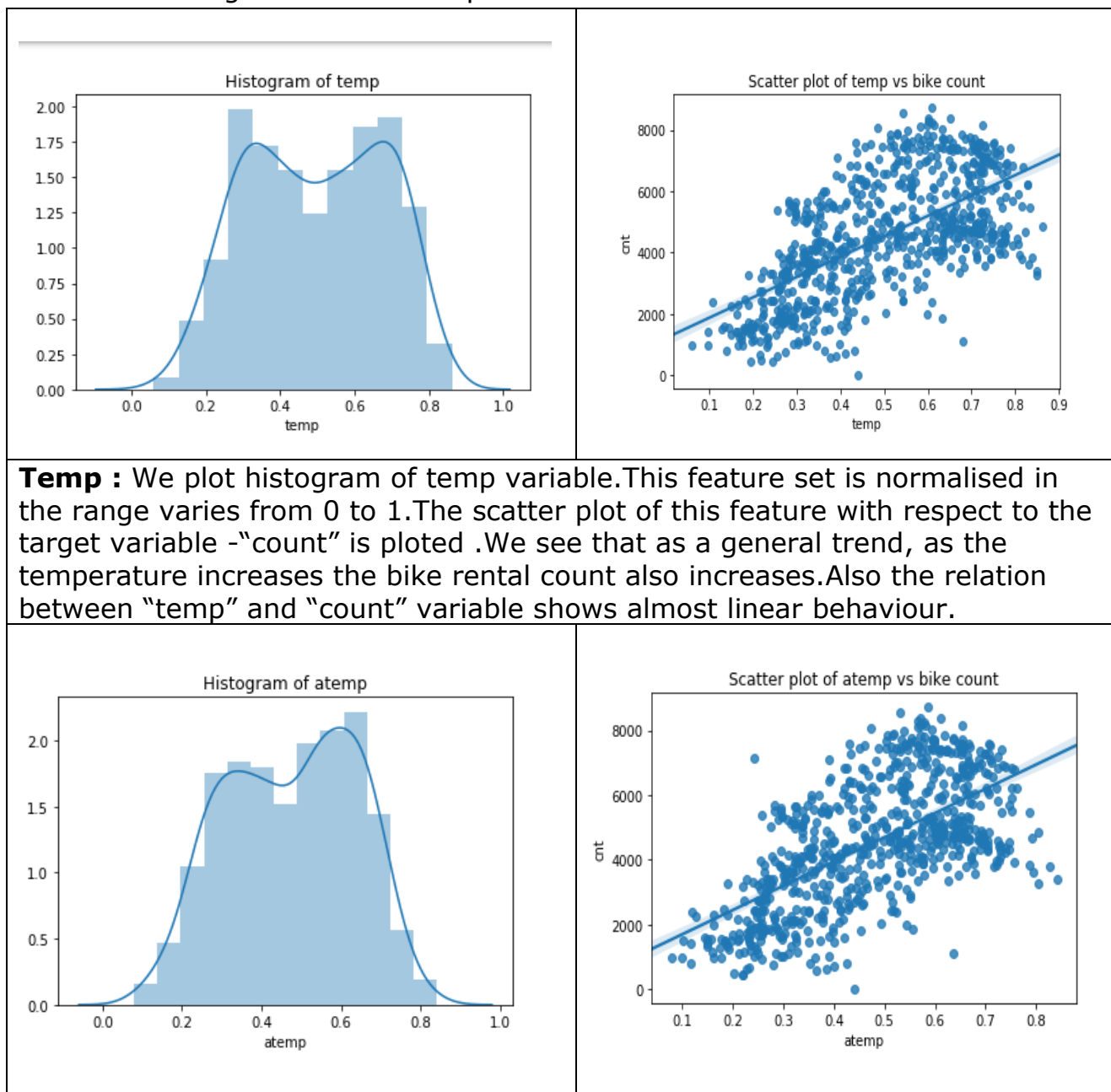
| | |
|------------|---|
| season | 0 |
| yr | 0 |
| mnth | 0 |
| holiday | 0 |
| weekday | 0 |
| workingday | 0 |
| weathersit | 0 |
| temp | 0 |
| atemp | 0 |
| hum | 0 |
| windspeed | 0 |
| cnt | 0 |

2.2 Data visualisation

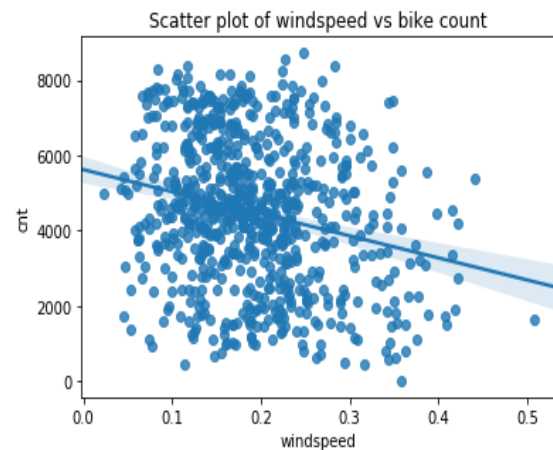
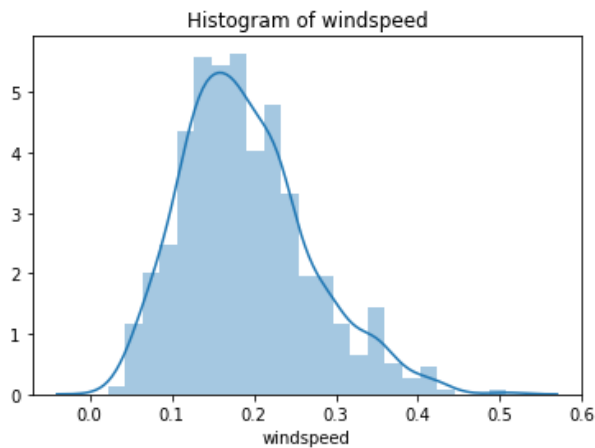
To assess distribution of data provided to us and to communicate information more clearly we go for data visualisation. It is also said "*a picture tells a thousand word*"! We would do univariate as well as bivariate visualisation of the data set for numerical and categorical features separately. Let us start with numerical features where we use histogram and scatter plot for visualisation and barplots for categorical feature set.

Numerical features:

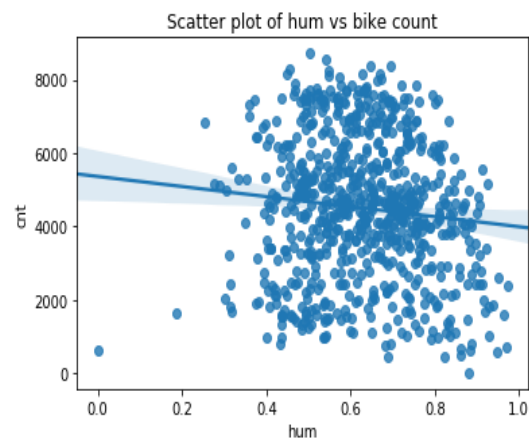
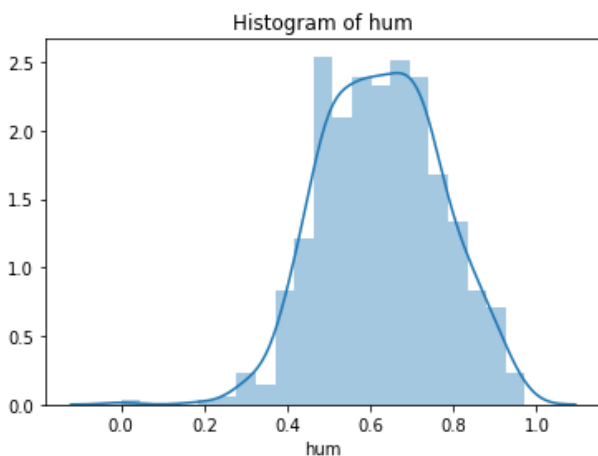
Table 2.2 Histogram and scatter plot of numerical features



Atemp : "atemp" is the normalised feeling temperature in Celsius. From the above distribution we see that "temp" and "atemp" shows similar distribution. This result was expected because the actual temperature and feeling temperature should not ideally deviate a lot. This variable also exhibits almost linear distribution with respect to the target variable – "count".



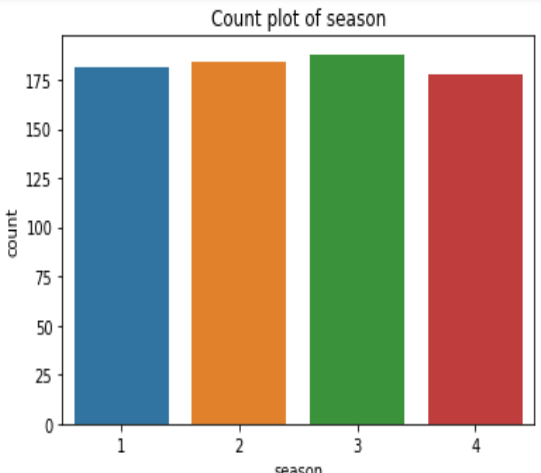
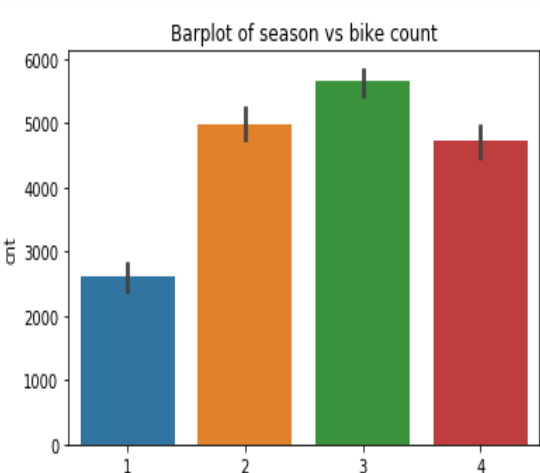
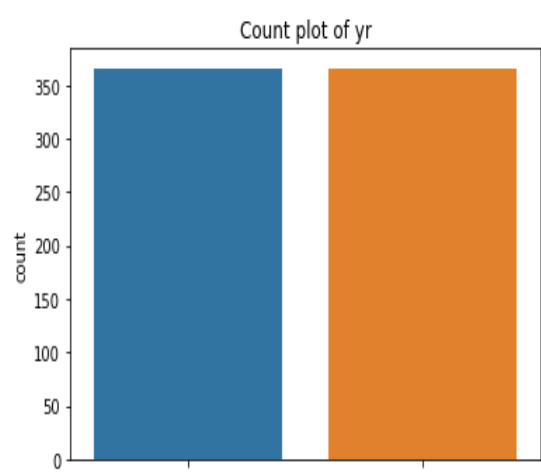
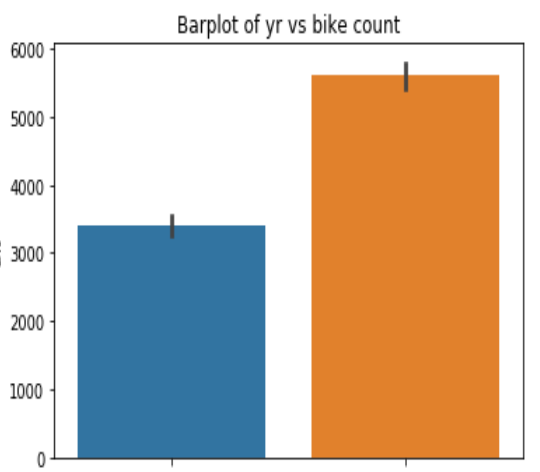
Windspeed : Histogram of windspeed shows slightly right skewed normal distribution. However, when we compared this feature alone with the target variable - bike count, the distribution is little scattered with concentration mainly on the lower side of the windspeed and so is the histogram distribution concentrated in the range of 0.1-0.3 units.



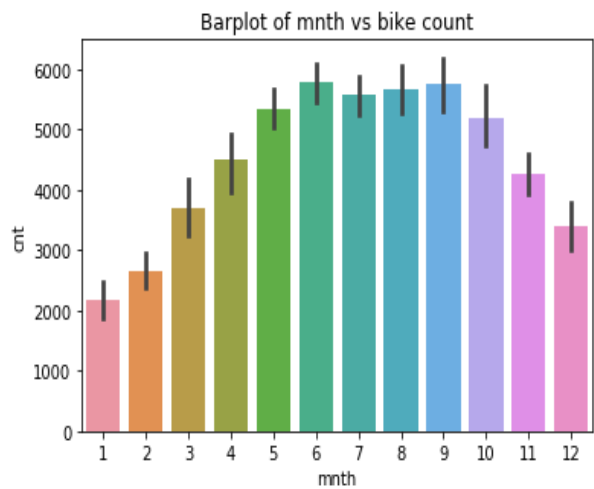
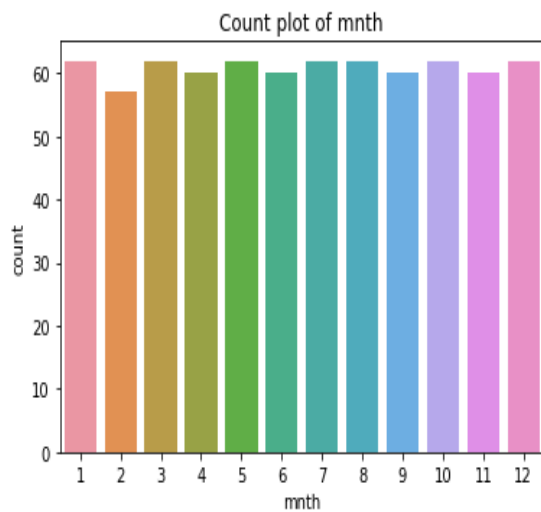
Hum : Humidity feature shows slightly left skewed histogram plot. The scatter plot of count vs humidity shows random distribution and no particular discernible pattern could be found out. The humidity concentration is typically in the range of 0.4-0.8 as shown in histogram as well as in the scatter plot. However, there are outliers seen around 0.0 and 0.2. However, the overall trend of bike count vs humidity is slightly downward.

Categorical features:

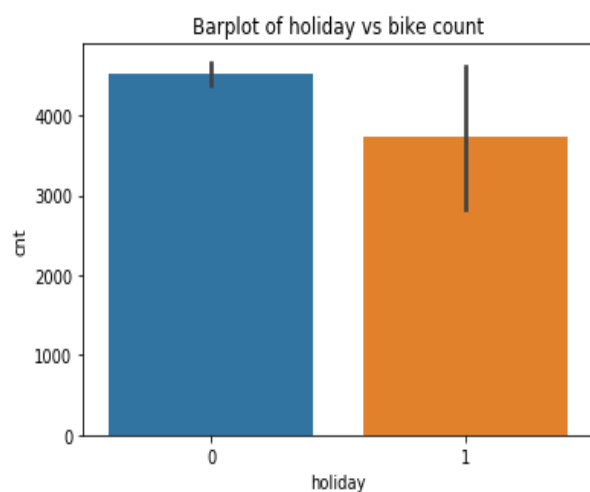
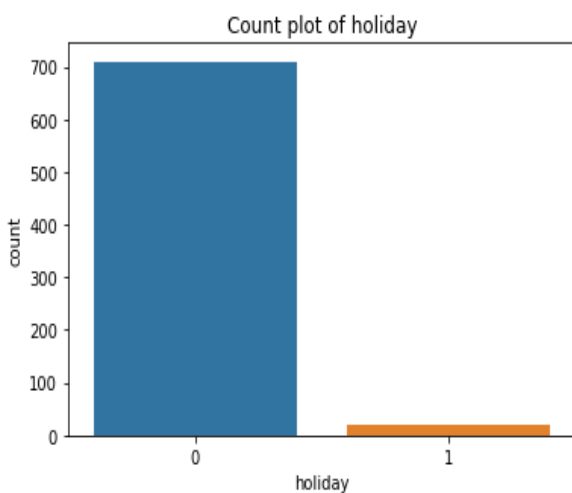
Table 2.3 Count plot and barplot of categorical features

|  <p>A count plot titled 'Count plot of season' showing the distribution of bike rental counts across four seasons. The x-axis is labeled 'season' with values 1, 2, 3, and 4. The y-axis is labeled 'count' with values from 0 to 175 in increments of 25. The bars are colored blue (1), orange (2), green (3), and red (4). The counts are approximately 180, 185, 190, and 175 respectively.</p> <table border="1"> <thead> <tr> <th>season</th> <th>count</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>180</td> </tr> <tr> <td>2</td> <td>185</td> </tr> <tr> <td>3</td> <td>190</td> </tr> <tr> <td>4</td> <td>175</td> </tr> </tbody> </table> | season | count | 1 | 180 | 2 | 185 | 3 | 190 | 4 | 175 |  <p>A barplot titled 'Barplot of season vs bike count' showing the mean bike rental count for each season. The x-axis is labeled 'season' with values 1, 2, 3, and 4. The y-axis is labeled 'cnt' with values from 0 to 6000 in increments of 1000. The bars are colored blue (1), orange (2), green (3), and red (4). Error bars are shown for each bar. The mean counts are approximately 2600, 5000, 5600, and 4700 respectively.</p> <table border="1"> <thead> <tr> <th>season</th> <th>cnt</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>2600</td> </tr> <tr> <td>2</td> <td>5000</td> </tr> <tr> <td>3</td> <td>5600</td> </tr> <tr> <td>4</td> <td>4700</td> </tr> </tbody> </table> | season | cnt | 1 | 2600 | 2 | 5000 | 3 | 5600 | 4 | 4700 |
|--|--------|-------|---|-----|---|-----|---|-----|-----|-----|--|--------|------|---|------|---|------|---|------|---|------|
| season | count | | | | | | | | | | | | | | | | | | | | |
| 1 | 180 | | | | | | | | | | | | | | | | | | | | |
| 2 | 185 | | | | | | | | | | | | | | | | | | | | |
| 3 | 190 | | | | | | | | | | | | | | | | | | | | |
| 4 | 175 | | | | | | | | | | | | | | | | | | | | |
| season | cnt | | | | | | | | | | | | | | | | | | | | |
| 1 | 2600 | | | | | | | | | | | | | | | | | | | | |
| 2 | 5000 | | | | | | | | | | | | | | | | | | | | |
| 3 | 5600 | | | | | | | | | | | | | | | | | | | | |
| 4 | 4700 | | | | | | | | | | | | | | | | | | | | |
| <p>Season: There are 4 seasons in our data set ,1: spring , 2: summer ,3:fall ,4:winter .Now the data set that we had has almost equal representation of all the four seasons.Hence , we are atleast sure that we have sufficient data in each of the season's categories.But the plot of seasonal variation of bike rental count reveals that bike renting is less in spring than other seasons.Also,renting reaches its maximum value in the fall season.</p> | | | | | | | | | | | | | | | | | | | | | |
|  <p>A count plot titled 'Count plot of yr' showing the distribution of bike rental counts across two years. The x-axis is labeled 'yr' with values 0 and 1. The y-axis is labeled 'count' with values from 0 to 350 in increments of 50. The bars are colored blue (0) and orange (1). The counts are approximately 360 and 370 respectively.</p> <table border="1"> <thead> <tr> <th>yr</th> <th>count</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>360</td> </tr> <tr> <td>1</td> <td>370</td> </tr> </tbody> </table> | yr | count | 0 | 360 | 1 | 370 |  <p>A barplot titled 'Barplot of yr vs bike count' showing the mean bike rental count for each year. The x-axis is labeled 'yr' with values 0 and 1. The y-axis is labeled 'cnt' with values from 0 to 6000 in increments of 1000. The bars are colored blue (0) and orange (1). Error bars are shown for each bar. The mean counts are approximately 3400 and 5600 respectively.</p> <table border="1"> <thead> <tr> <th>yr</th> <th>cnt</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>3400</td> </tr> <tr> <td>1</td> <td>5600</td> </tr> </tbody> </table> | yr | cnt | 0 | 3400 | 1 | 5600 | | | | | | | | |
| yr | count | | | | | | | | | | | | | | | | | | | | |
| 0 | 360 | | | | | | | | | | | | | | | | | | | | |
| 1 | 370 | | | | | | | | | | | | | | | | | | | | |
| yr | cnt | | | | | | | | | | | | | | | | | | | | |
| 0 | 3400 | | | | | | | | | | | | | | | | | | | | |
| 1 | 5600 | | | | | | | | | | | | | | | | | | | | |
| <p>Year: Countplot of "yr" : year feature .0: signifies year 2011 and 1 : signifies year 2012.Our data set sufficiently represents both the year 2010 and 2011.While plotting year vs count plot,we see that 2012 has more bike rental count than in year 2011.Probably the popularity of rental bike increased in</p> | | | | | | | | | | | | | | | | | | | | | |

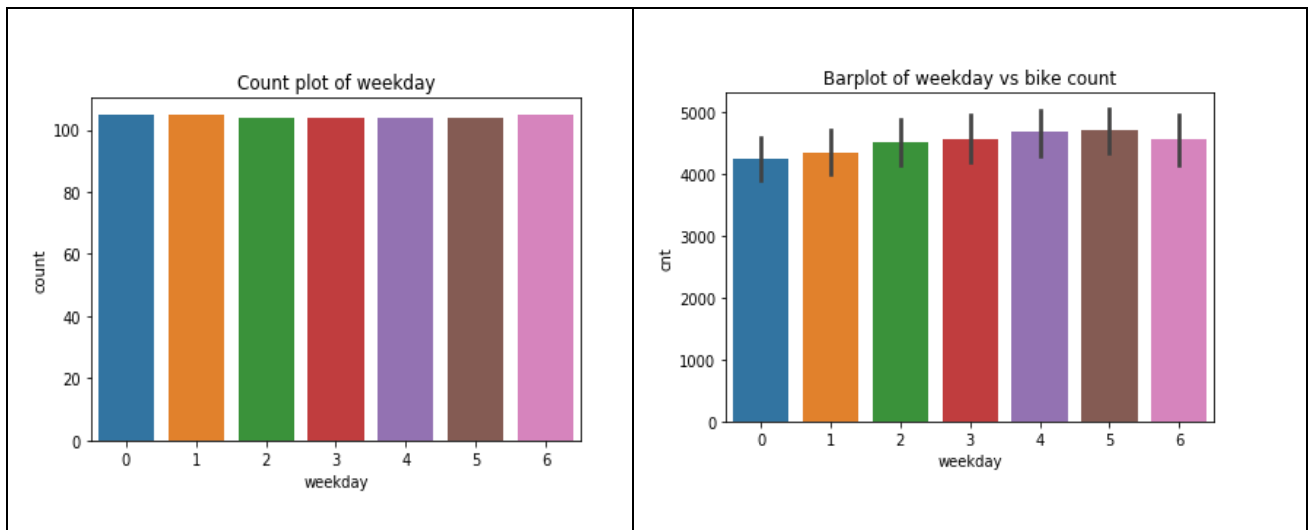
later years.



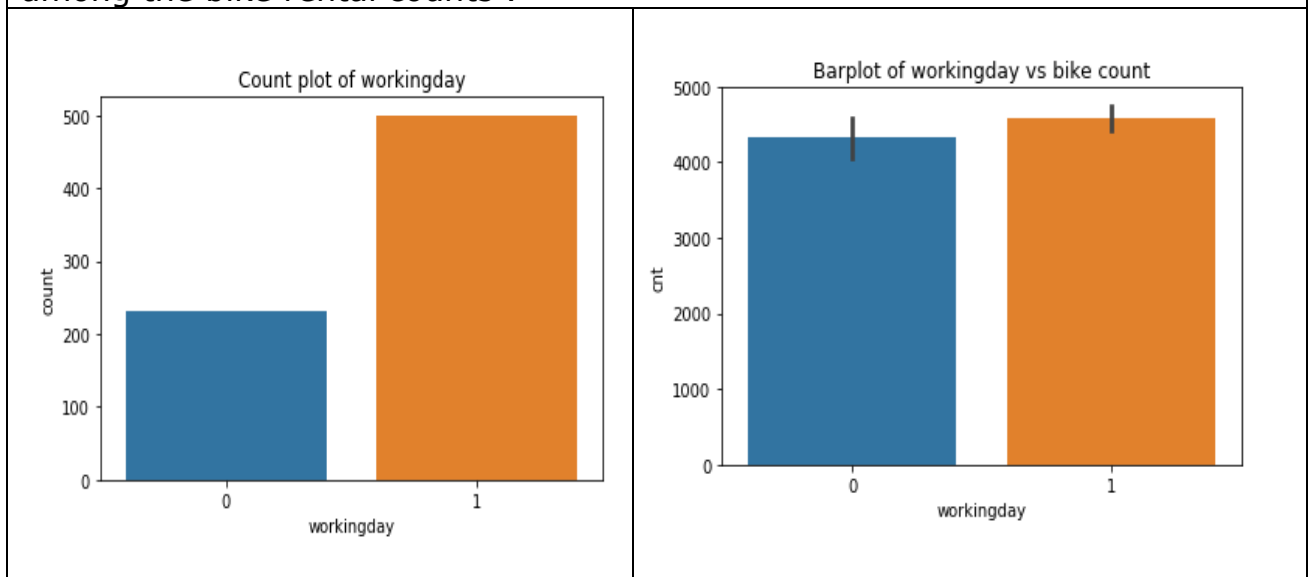
Month : The above plot shows barplot of month feature for all the 12 months. The countplot shows that each month is almost equally present in our data set. However, on comparing monthly count plot, we see that some seasons show higher bike rental count. Like for months 6-9 i.e June, July, August and September shows peak seasons with decreasing count on either side of the plot.



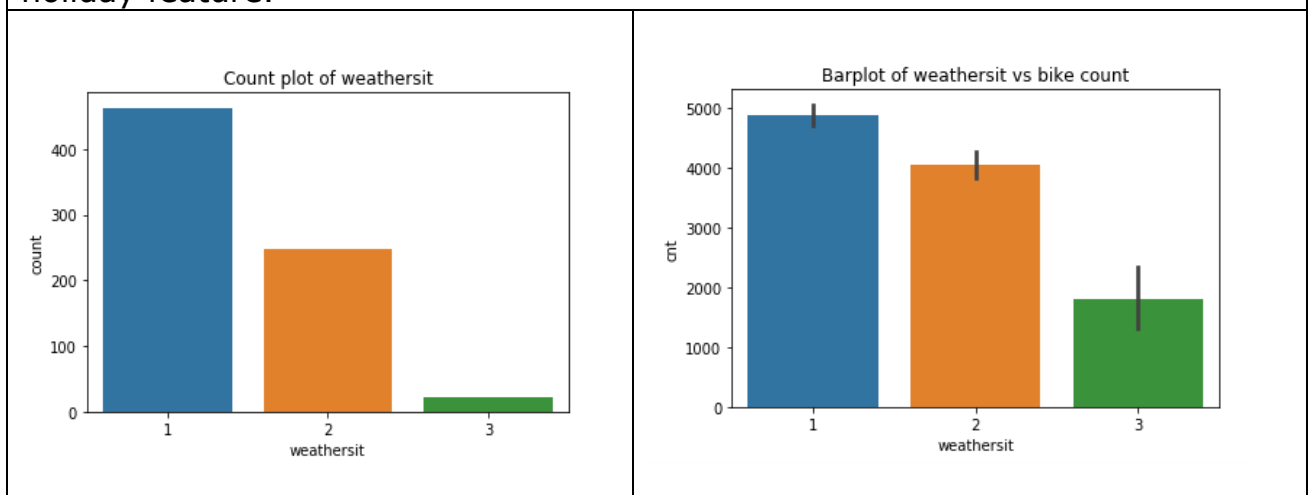
Holiday : Holiday barplot, as expected shows lower bar for holiday than for working days. From the barplot of count plot vs holiday shows that people use bike more on working days than on holidays. This suggests that our customer base would involve working professionals or college graduates. The usage pattern is more for weekday commutation and business purposes rather than leisure riding.



Weekday: Our data set has all the weekdays equally represented as evident from the barplot. Barplot of weekday vs bike count shows only slight variation among the bike rental counts .



Workingday : There are more working days than holidays, as expected. Barplot of the working day vs bike count shows count to be slightly higher than the non-working days. This customer behaviour was also evident in the holiday feature.



Weathersit : Weathersit 's countplot show that we have more data for 1: Clear, Few clouds, Partly cloudy, Partly cloudy then 2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist and least for 3 : Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds. Also no data for weather situation 4: Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog is present in our data set. Any ways during heavy rain ,no one will rent a bike

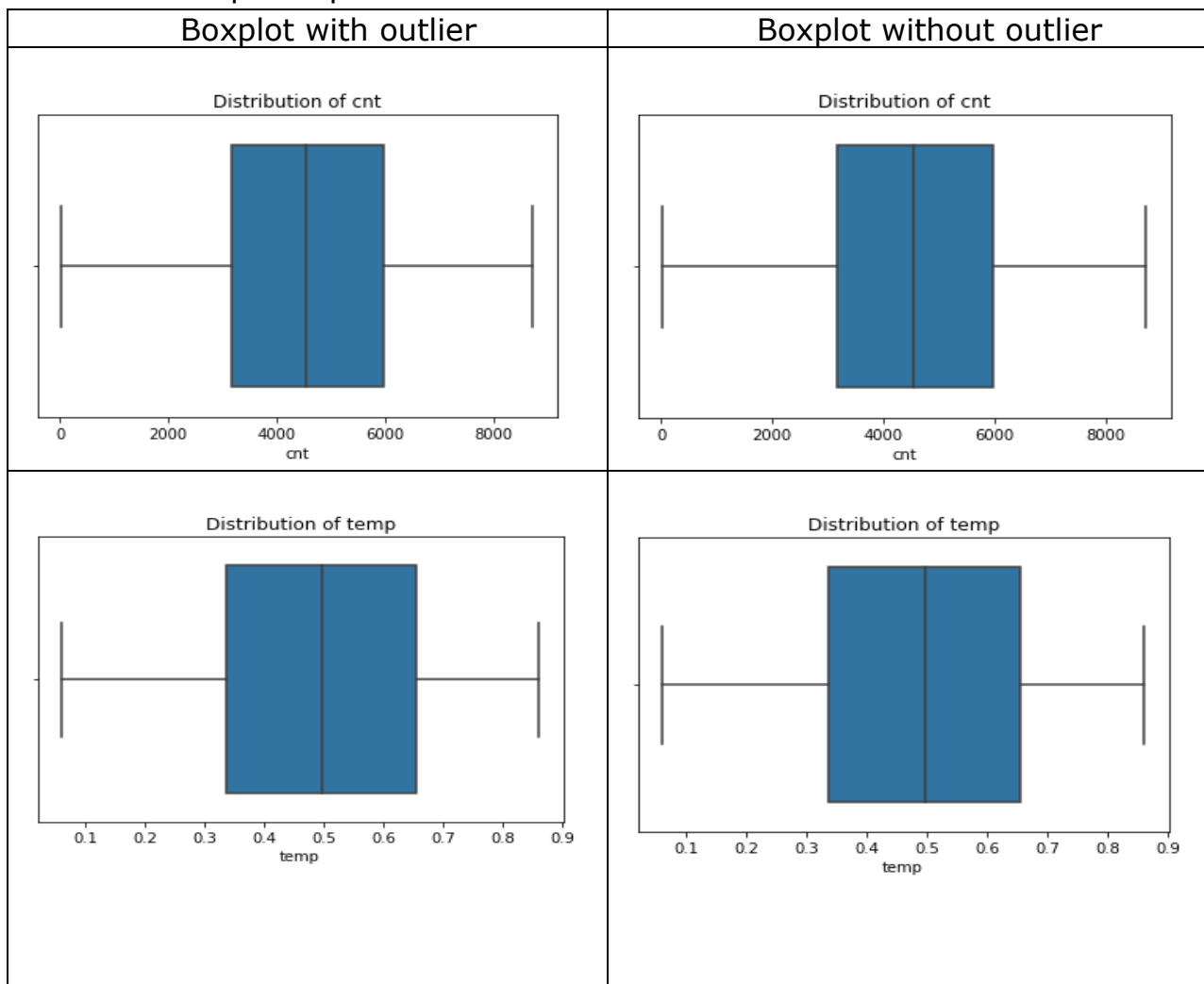
2.3 Outlier analysis:

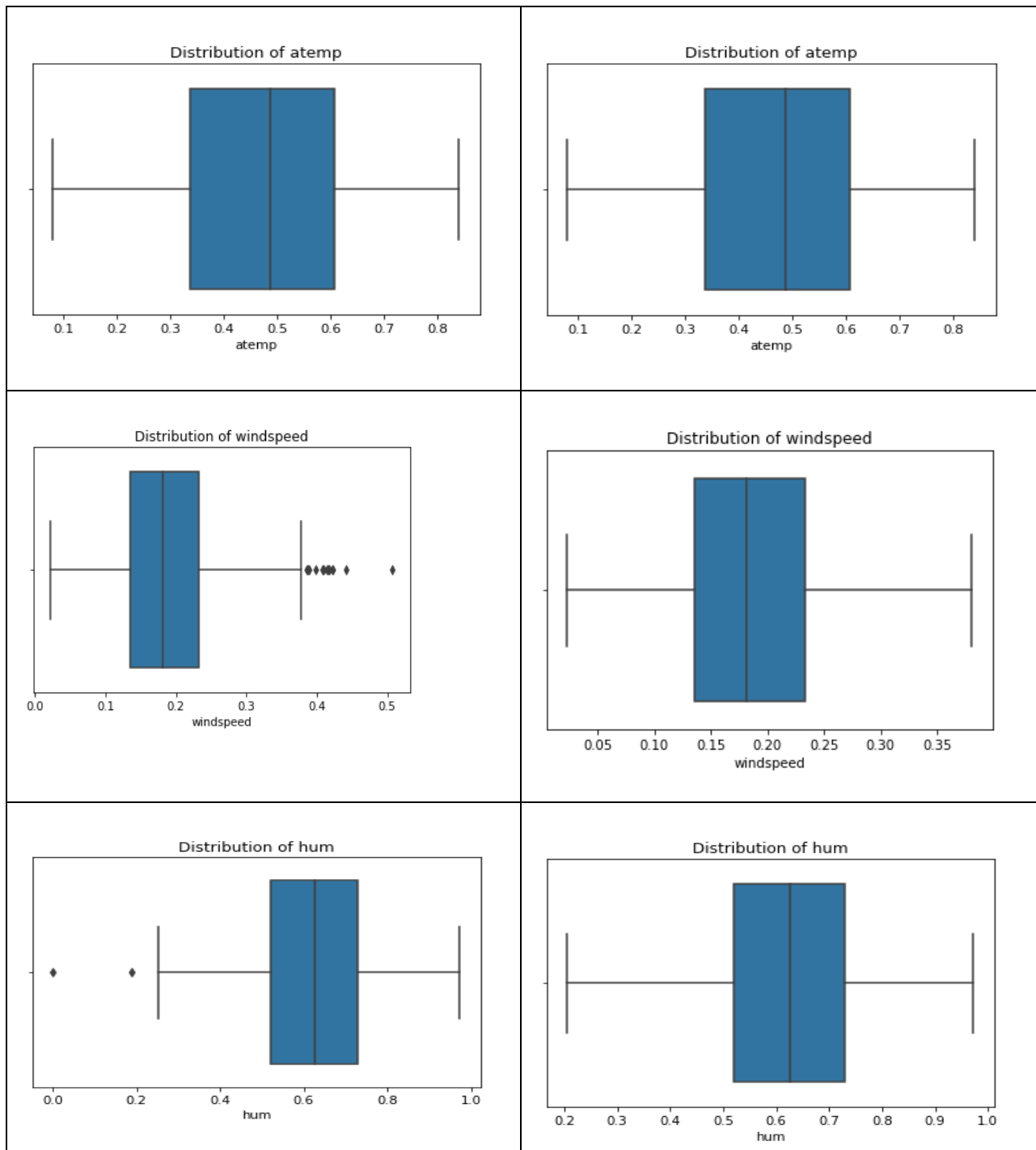
Outliers are atypical values in the dataset .One of the techniques of outlier analysis is boxplot method.

Here , any data point which is less than 1.5 inter quartile range times less than the 25th percentile or more than 1.5 inter quartile range times the 75th percentile, is to be treated as an outlier. In our analysis, we have replaced those values by capping them to lower fence or upper fence respectively.

We will visualise the boxplot of the numerical features with and without outliers.

Table 2.4 Boxplot representation of numerical features with and without outlier





2.4 Feature Scaling:

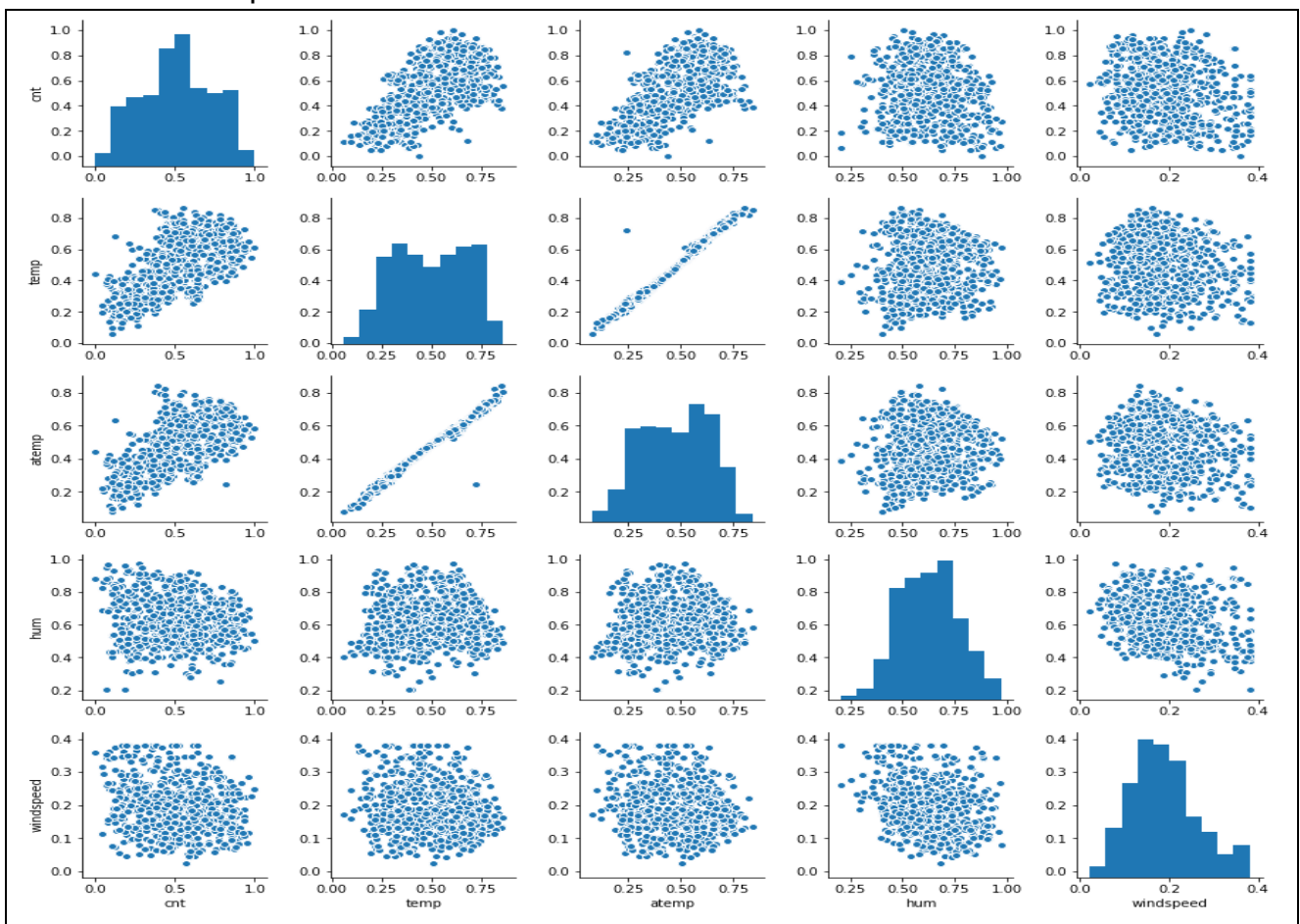
Before proceeding further ,we scale down the features .Most of the numerical features are normalised , however our target value is in the range of 100s and 1000s.Thus, we scale down the target variable count using normalisation .General formula for normalising any variable X is given by :

$$X_{norm} = (X - X_{min}) / (X_{max} - X_{min})$$

2.5 Feature Selection:

One of the key task in any data science operation is to choose right set of predictors. This is because ,although more number of features implies more knowledge of our dataset but high dimension in the data set can also lead to higher variance which might fail to generalise on the test data leading to higher test MSE(Mean Square Error) .This is also known as the **curse of dimensionality**. Apart from this, higher dimensional data in our model can also be computationally expensive. Thus we need to perform feature selection before supplying predictors to our model.

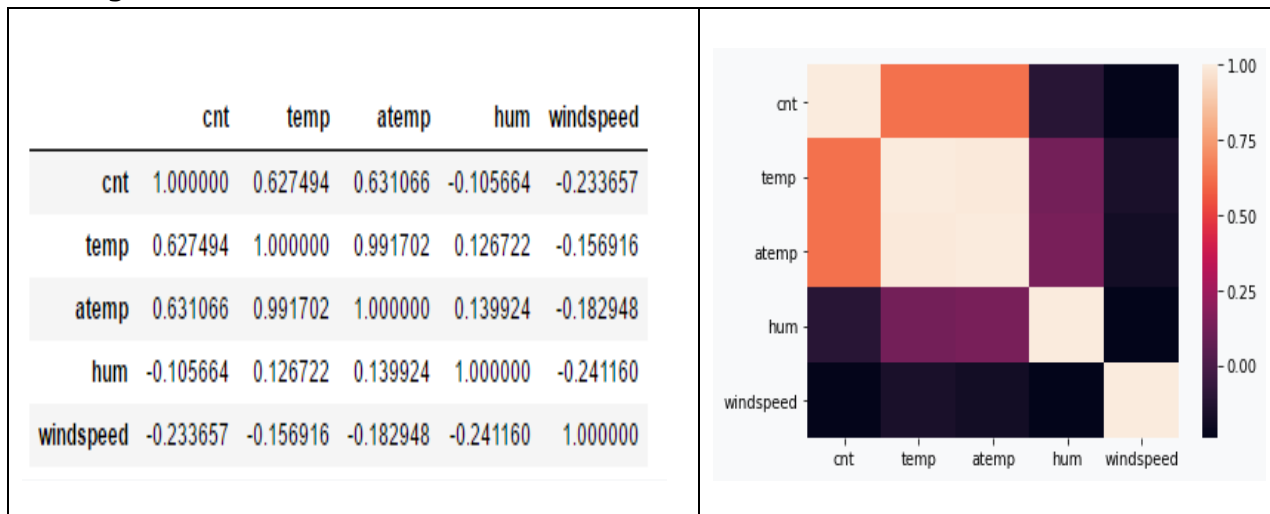
Table 2.5 Pair plot for all the numerical variables



Here from the above graph , we see correlation between pair of variable. Feature “temp” and “atemp” shows very high correlation. Let us check the degree of correlation between the two.

Numerical features :

Table 2.6 Correlation matrix for numerical features (left) and correlation plot on right



Multicollinearity:

As correlation verifies relation between each *pair of variables*, however when more than two features are present, multicollinearity is a better check.

Multicollinearity can be checked through VIF (Variance Inflation Factor) values. We obtain following VIF values for our data set. Variables with vif value greater than 10 can be safely dropped. Temp and atemp shows higher vif, thus we discard one of them (temp).

Table 2.7 VIF factor table for numerical features

| | |
|-----------|-----------|
| cnt | 1.870477 |
| temp | 63.123945 |
| atemp | 64.081148 |
| hum | 1.178937 |
| windspeed | 1.184046 |
| const | 53.690600 |
| dtype: | float64 |

Categorical features:

Anova(Analysis of variance) test : As target variable is continuous ,we perform ANOVA test for checking the variation in the target variable explained by the categorical feature set. Considering 95% confidence interval, feature variables

with pvalue more than 0.05 will be discarded. From the table we see that no such feature is present.

Table 2.8 Anova results on categorical data set

```
season
F_onewayResult(statistic=2234.7131036468481, pvalue=1.1812270473734066
e-296)
yr
F_onewayResult(statistic=0.54842192143257973, pvalue=0.459082290681055
99)
mnth
F_onewayResult(statistic=2202.3890187080228, pvalue=7.2351429161856712
e-294)
holiday
F_onewayResult(statistic=2233.4041277168758, pvalue=1.5301137528987259
e-296)
weekday
F_onewayResult(statistic=1106.3742871546431, pvalue=4.742367523581619e
-181)
workingday
F_onewayResult(statistic=77.813963359979383, pvalue=3.1624070118212319
e-18)
weathersit
F_onewayResult(statistic=1632.0856144299062, pvalue=3.56582852113021e-
240)
```

The Anova test shows “yr” feature set having p value greater than 0.05. Thus we remove “yr” variable.

Thus, we drop following feature set :

- instance
- Dteday
- Casual
- registered
- Temp
- yr

Considering total feature reduction that we have done till now , we have been able to reduce our data set dimension from 731*16 to 731*10.

2.5 Data Modelling

After all the pre-processing steps are done, we are now ready to model our data to predict bike rental count.

Let us first distribute our data into test set and train set. We chose 75% of the data as train data and 25% of the data as test data.

Code :

```
data_set_test , data_set_train = train_test_split(data_set,test_size = 0.25)
```

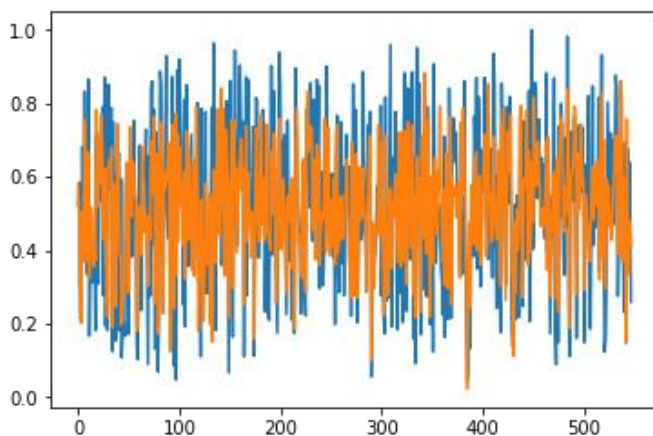
After feature selection we can now start using different regression models to predict .Let us start with the simplest model and then move towards more complex models if needed.

Linear Regression:

```
modelLR = sm.OLS(y_train,x_train,data = data_set_train).fit()  
modelLR.fit(x_train,y_train)  
predictLR = modelLR.predict(x_test)
```

Output :

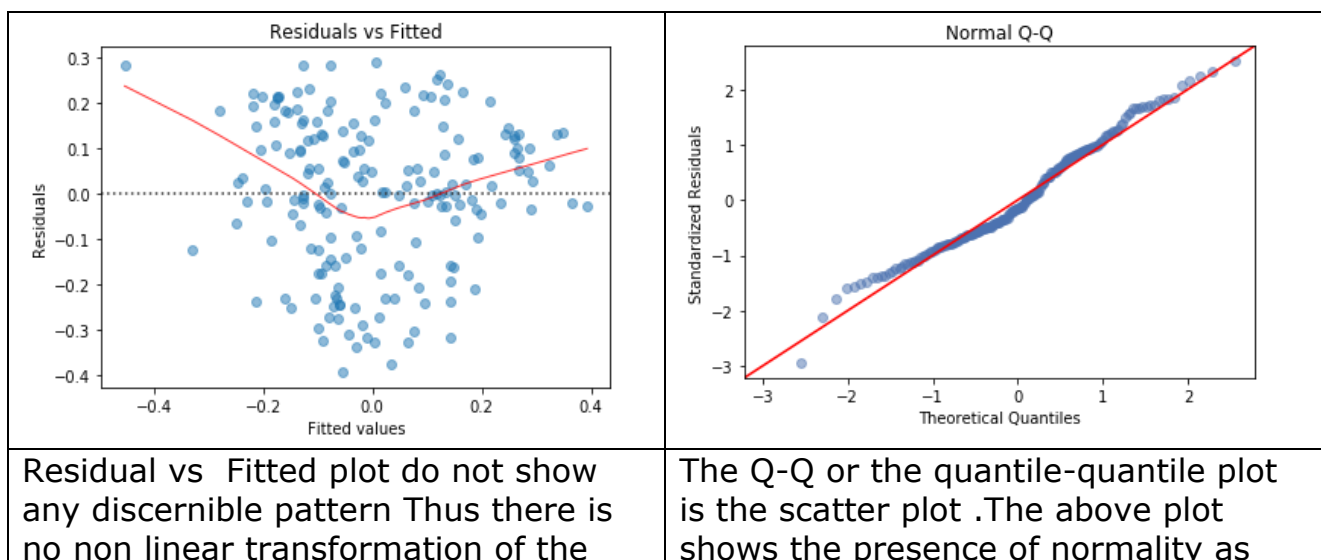
Actual(-) vs predicted plot(-)

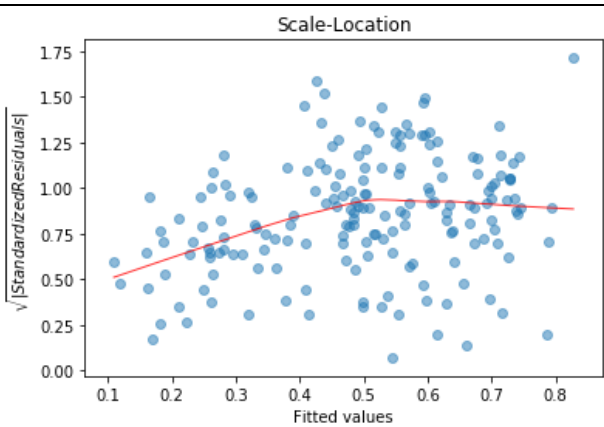
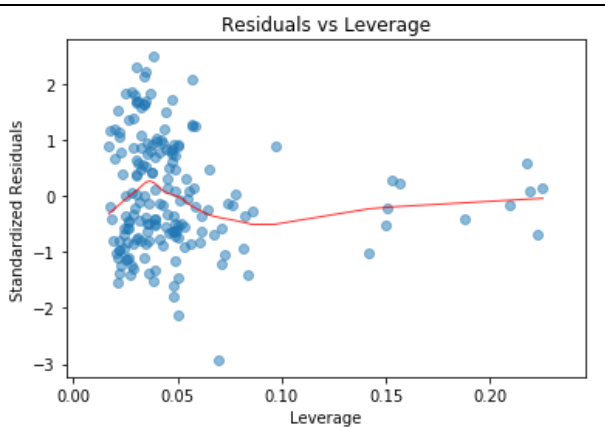


Summary :

| OLS Regression Results | | | | | | |
|------------------------|------------------|---------------------|----------|-------|--------|--------|
| ===== | | | | | | |
| Dep. Variable: | cnt | R-squared: | 0.919 | | | |
| Model: | OLS | Adj. R-squared: | 0.915 | | | |
| Method: | Least Squares | F-statistic: | 218.9 | | | |
| Date: | Mon, 28 Jan 2019 | Prob (F-statistic): | 5.53e-90 | | | |
| Time: | 22:14:18 | Log-Likelihood: | 80.943 | | | |
| No. Observations: | 183 | AIC: | -143.9 | | | |
| Df Residuals: | 174 | BIC: | -115.0 | | | |
| Df Model: | 9 | | | | | |
| Covariance Type: | nonrobust | | | | | |
| ===== | | | | | | |
| | coef | std err | t | P> t | [0.025 | 0.975] |
| ----- | | | | | | |
| season | 0.0812 | 0.019 | 4.240 | 0.000 | 0.043 | 0.119 |
| mnth | -0.0084 | 0.006 | -1.379 | 0.170 | -0.020 | 0.004 |
| holiday | 0.1050 | 0.075 | 1.408 | 0.161 | -0.042 | 0.252 |
| weekday | 0.0123 | 0.005 | 2.267 | 0.025 | 0.002 | 0.023 |
| workingday | 0.0118 | 0.025 | 0.468 | 0.640 | -0.038 | 0.061 |
| weathersit | -0.0967 | 0.031 | -3.158 | 0.002 | -0.157 | -0.036 |
| atemp | 0.7503 | 0.077 | 9.725 | 0.000 | 0.598 | 0.903 |
| hum | 0.0573 | 0.109 | 0.525 | 0.600 | -0.158 | 0.273 |
| windspeed | 0.2580 | 0.142 | 1.815 | 0.071 | -0.023 | 0.539 |
| ===== | | | | | | |
| Omnibus: | 3.486 | Durbin-Watson: | 2.242 | | | |
| Prob(Omnibus): | 0.175 | Jarque-Bera (JB): | 3.209 | | | |
| Skew: | 0.252 | Prob(JB): | 0.201 | | | |
| Kurtosis: | 2.592 | Cond. No. | 109. | | | |
| ===== | | | | | | |

Model Plot :



| | |
|--|--|
| predictors required. | the points almost passes through the straight line diagonal. However, some deviations are seen at the extremes. |
|  |  |
| Scale Location plot is similar to the residual plot. The only difference is, it uses square root of standardised residual errors. As no particular pattern can be seen in the plot, we can conclude that there is no heteroskedacity i.e. variance of the error term is equal. | The above plot is also known as Cook's distance plot. It is used to identify if some predictors' point influence our prediction errors more than the others. |

Regularisation :

Code :

```
for Model in [Ridge, Lasso]:
    gscv = GridSearchCV(Model(), dict(alpha=alphas),
    cv=5,scoring='r2').fit(x_train, y_train)
    print('Model %s:Best alpha: %s' % (Model.__name__, gscv.best_params_))
    print("R2:",gscv.best_score_)
    print("MSE:",math.sqrt(mean_squared_error(y_test,gscv.predict(x_test))))
```

Output :

```
Model Ridge:Best alpha: {'alpha': 0.10000000000000001}
R2: 0.710556844191
RMSE: 0.10207218992400499
Model Lasso:Best alpha: {'alpha': 0.001}
R2: 0.716058562664
RMSE: 0.10309449668243081
```

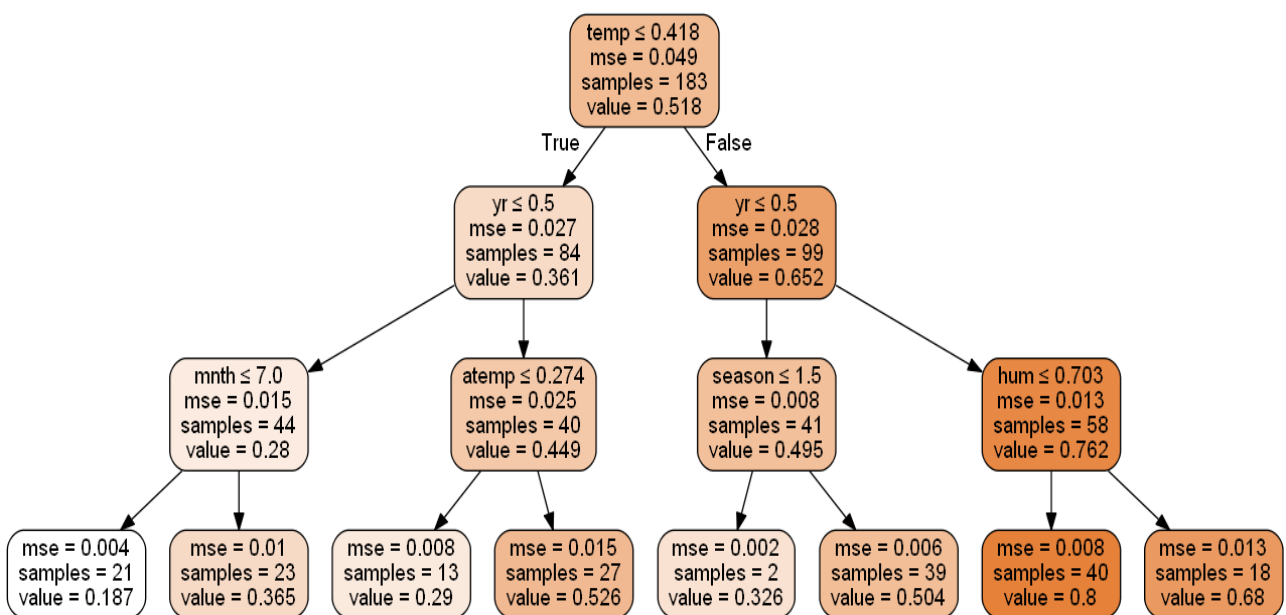
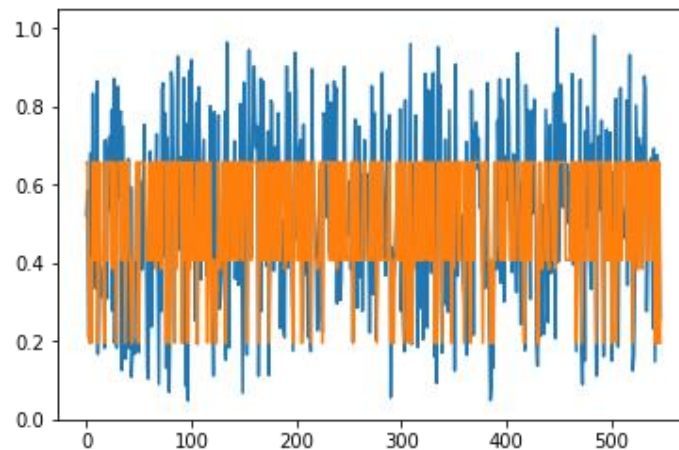
Decision Trees:

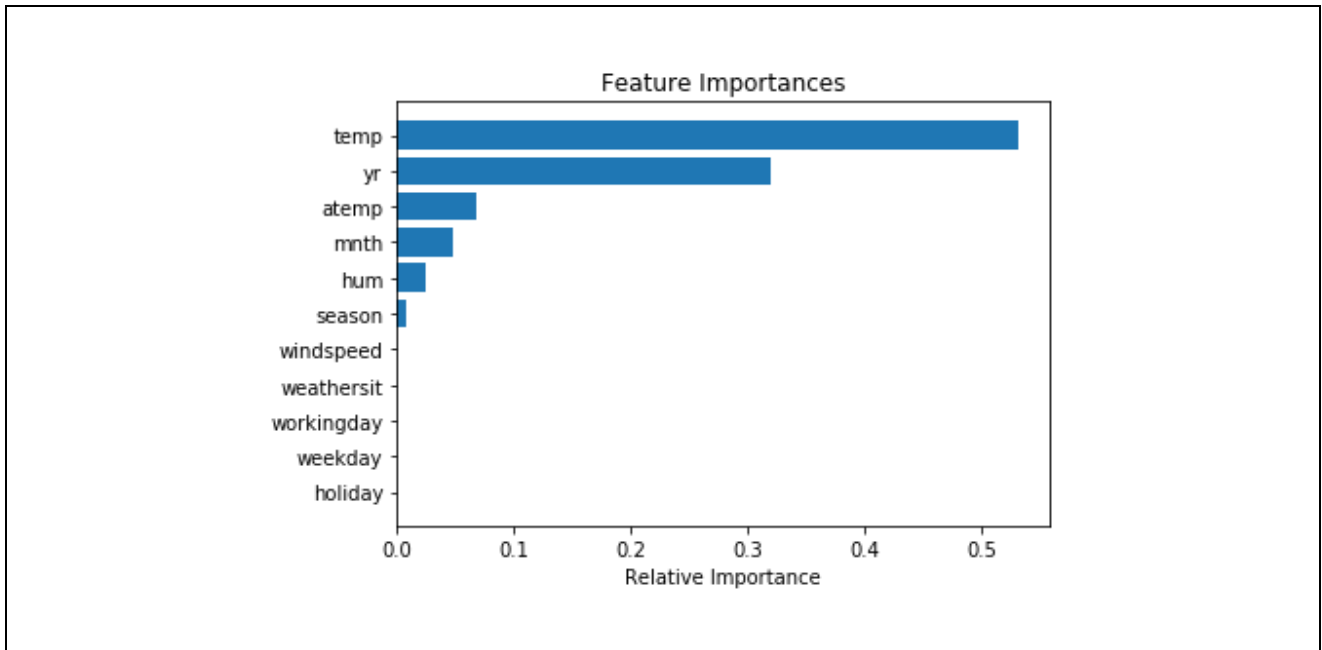
```
modelDT = DecisionTreeRegressor(max_depth=3,max_features='auto')
modelDT.fit(x_train,y_train)
predictDT = modelDT.predict(x_test)
print("RMSE:",math.sqrt(mean_squared_error(y_test,predictDT)),"R2:",model
DT.score(x_test,y_test))
```

Output:

RMSE: 0.11274794712287024 R2: 0.745404033844

Actual(-) vs predicted plot(-)





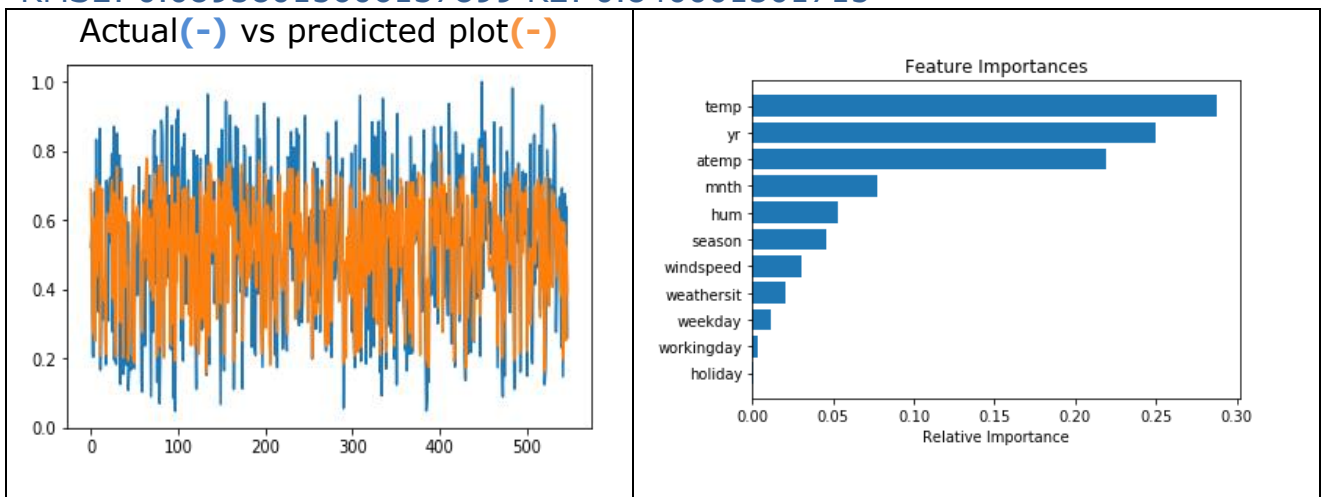
Random Forest:

Code:

```
modelRF = RandomForestRegressor(bootstrap =
'True',max_depth=8,n_estimators=
100,random_state=0,max_features=4,min_samples_leaf=2,min_samples_split
=8)
modelRF.fit(x_train,y_train)
predictRF = modelRF.predict(x_test)
print("RMSE:",math.sqrt(mean_squared_error(y_test,predictRF)),"R2:",modelR
F.score(x_test,y_test))
```

Output :

RMSE: 0.08938015606137899 R2: 0.840001301715



Chapter 3

Conclusion

3.1 Model Evaluation

In the earlier section, we had used following algorithms for modelling our dataset

- i. Linear Regression
- ii. Ridge Regression
- iii. Lasso Regression
- iv. Decision Trees
- v. Random Forest

We now compare the result by measuring RMSE (Root mean square error) of the actual vs predicted plot for the test set for all the three models used.

Table 2.6 RMSE of Linear Model, Decision Trees and Random Forest

| Model | RMSE(in R) | R-sq(R) | RMSE(in Python) | R-sq(Python) |
|-----------------------|-------------|-------------|-----------------|--------------|
| Linear Model | 0.27 | 0.64 | 0.10 | 0.79 |
| Ridge Regression | 0.28 | 0.74 | 0.10 | 0.71 |
| Lasso Regression | 0.28 | 0.74 | 0.10 | 0.71 |
| Decision Trees | 0.02 | 0.66 | 0.11 | 0.74 |
| Random Forest | 0.20 | 0.85 | 0.08 | 0.84 |

From the above RMSE and R-squared results of the five algorithms, we see that Random Forest performs best in Python whereas Decision Trees performs best in terms of RMSE and Random forest in terms of R – squared value .

References:

- i) <https://edwisor.com/>
- ii) <https://www.analyticsvidhya.com>
- iii) <https://towardsdatascience.com/>

Note: Figures and References are made from Python code outputs