BIKE RENTAL PREDICTION

April 1st, 2019

Prediction based on seasonal and environmental settings

BY VENKAT DEEPAK GARLAPATI

CONTENTS

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 behavior and usage pattern to maximize the organization's profit.

In this problem, our task is to predict the predict the bike rental counts based on seasonal and environment settings given for the year 2011 and 2012. The usage pattern is affected by weather variables.

1.2 DATA

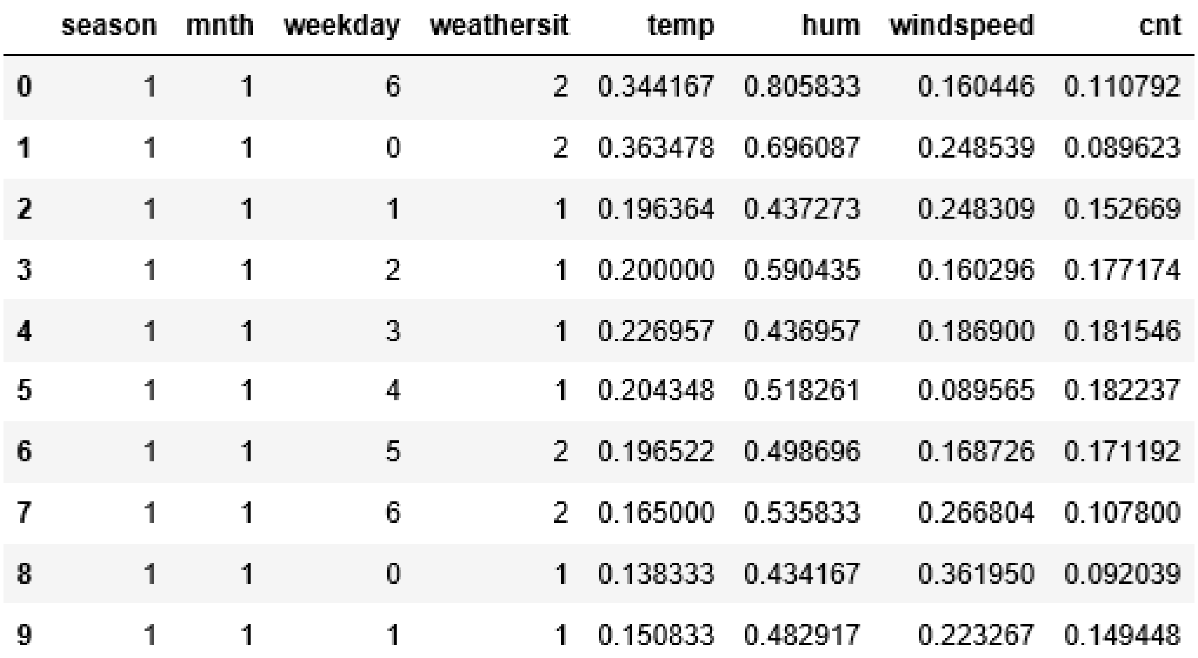
- The details of data attributes in the dataset are as follows

S. No.	Feature	Description
1	instant	Record index
2	dteday	Date
3	season	Season (1:springer, 2:summer, 3:fall, 4:winter)
4	yr	Year (0: 2011, 1:2012)
5	mnth	Month (1 to 12)
6	holiday	weather day is holiday or not (extracted fromHoliday Schedule)
7	weekday	Day of the week
8	workingday	If day is neither weekend nor holiday is 1, otherwise is 0.
9	weathersit	(extracted fromFreemeteo) 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
10	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)
11	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)
12	hum	Normalized humidity. The values are divided to 100 (max)
13	windspeed	Normalized wind speed. The values are divided to 67 (max)
14	casual	count of casual users
15	registered	count of registered users
16	cnt	count of total rental bikes including both casual and registered

Table 1.1 Data set feature description

- Dimension of the data set (including the target variable): 731 X 16



	season	mnth	weekday	weathersit	temp	hum	windspeed	cnt
0	1	1	6	2	0.344167	0.805833	0.160446	0.110792
1	1	1	0	2	0.363478	0.696087	0.248539	0.089623
2	1	1	1	1	0.196364	0.437273	0.248309	0.152669
3	1	1	2	1	0.200000	0.590435	0.160296	0.177174
4	1	1	3	1	0.226957	0.436957	0.186900	0.181546
5	1	1	4	1	0.204348	0.518261	0.089565	0.182237
6	1	1	5	2	0.196522	0.498696	0.168726	0.171192
7	1	1	6	2	0.165000	0.535833	0.266804	0.107800
8	1	1	0	1	0.138333	0.434167	0.361950	0.092039
9	1	1	1	1	0.150833	0.482917	0.223267	0.149448

Fig 1.1 Data set feature description

Chapter 2

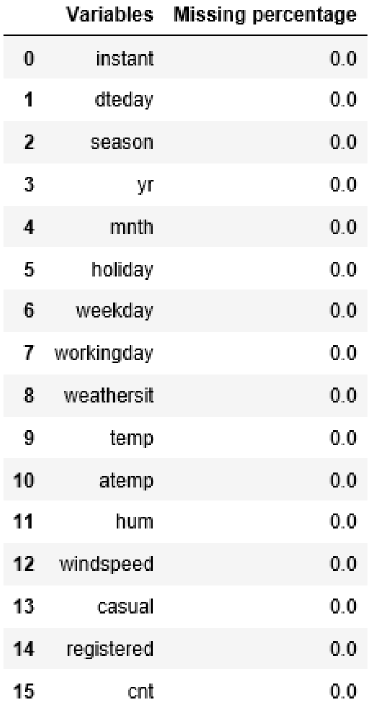
METHODOLOGY

2.1 PRE – PROCESSING

Any predictive modeling requires that we look at the data before we start modeling. 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. To start this process, we will first try and look at all the probability distributions of the variables. Most analysis like regression, require the data to be normally distributed. We can visualize that in a glance by looking at the probability distributions or probability density functions of the variable.

2.1.1 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.



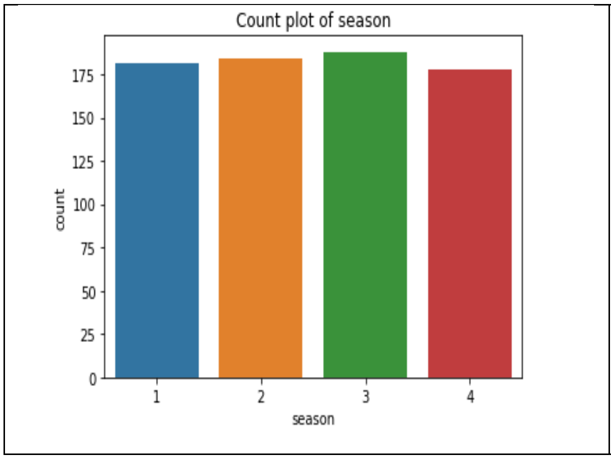
	Variables	Missing percentage
0	instant	0.0
1	dteday	0.0
2	season	0.0
3	yr	0.0
4	mnth	0.0
5	holiday	0.0
6	weekday	0.0
7	workingday	0.0
8	weathersit	0.0
9	temp	0.0
10	atemp	0.0
11	hum	0.0
12	windspeed	0.0
13	casual	0.0
14	registered	0.0
15	cnt	0.0

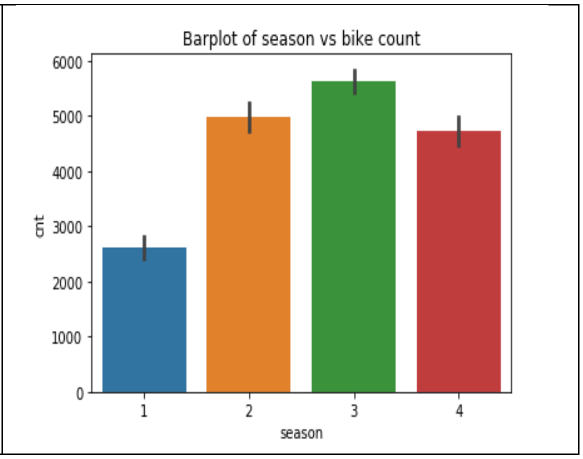
Figure 2.1 Missing value percentages

2.1.2 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.

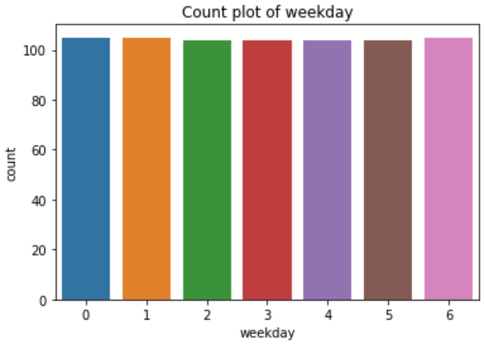
Categorical Features:

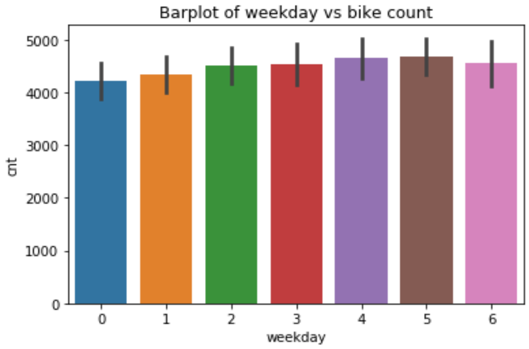




Season:

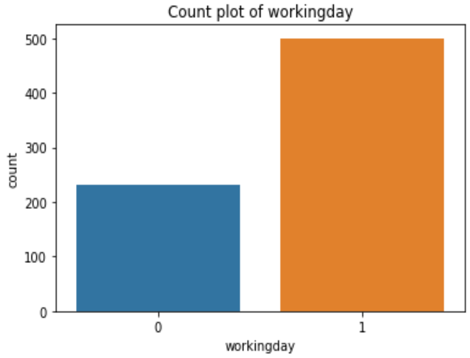
There are 4 seasons in our data set ,1: spring , 2: summer ,3:fall ,4:winter .Now the data set that we had has almost equal representation of all the four seasons. Hence , we are at least sure that we have enough data in each of the season's categories. But the plot of seasonal variation of bike rental count reveals that bike renting is less in spring than other seasons. Also, renting reaches its maximum value in the fall season.

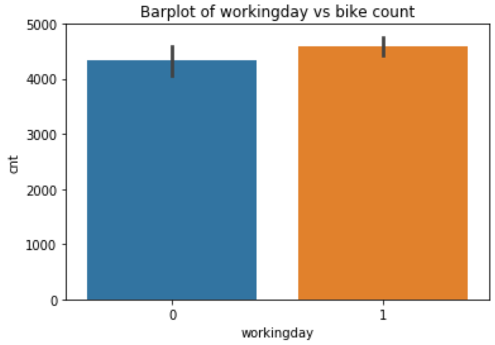




Weekday:

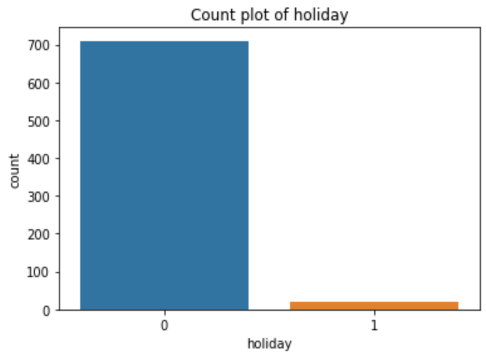
Our data set has all the weekdays equally represented as evident from the bar plot. Barplot of weekday vs bike count shows only slight variation among the bike rental counts

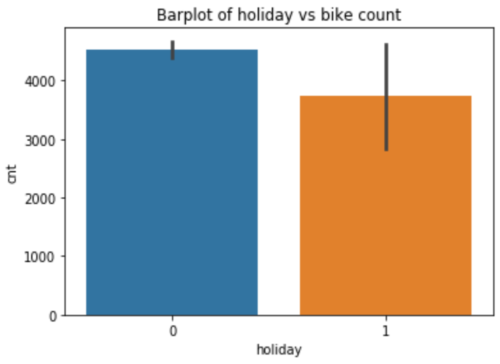




Workingday:

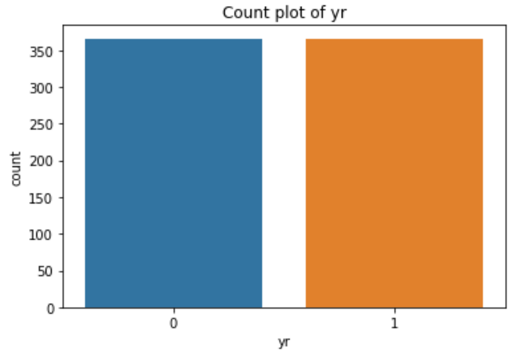
There are more working days than holidays, as expected. Barplot of the working day vs bike count shows count to be slightly higher than the non-working days. This customer behavior was also evident in the holiday feature.

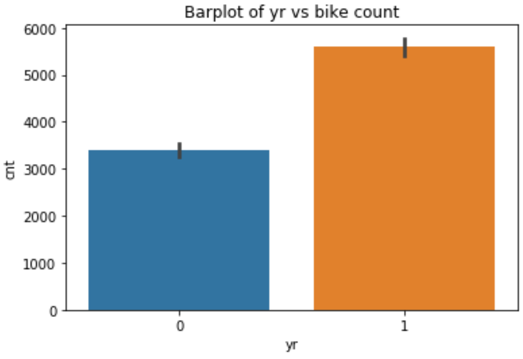




Holiday:

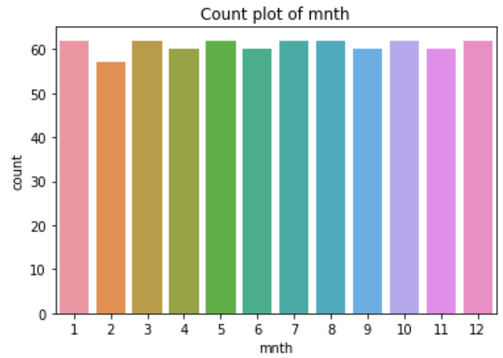
Holiday bar plot, as expected shows lower bar for holiday than for working days. From the bar plot of count plot vs holiday shows that people use bike more on working days than on holidays. This suggests our customer base would involve working professionals or college graduates. The usage pattern is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

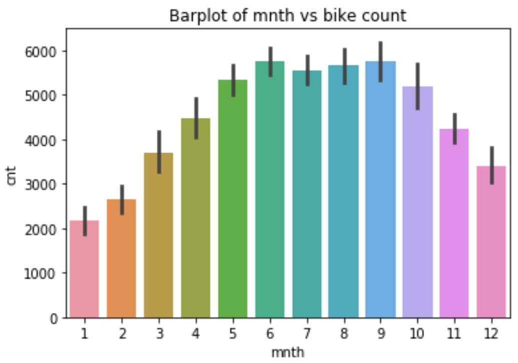




Year:

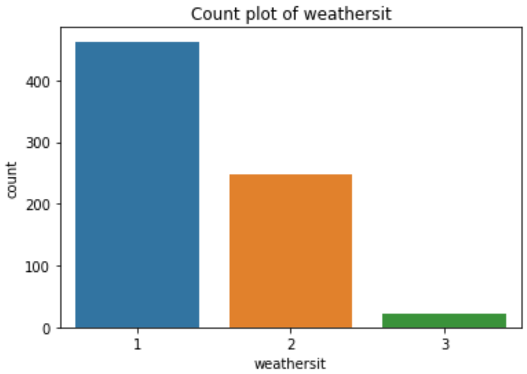
Count plot of “yr” : year feature .0: signifies year 2011 and 1 : signifies year 2012.Our data set sufficiently represents both the year 2011 and 2012.While plotting year vs count plot, we see that 2012 has more bike rental count than in year 2011.Probably the popularity of rental bike increased in later years.

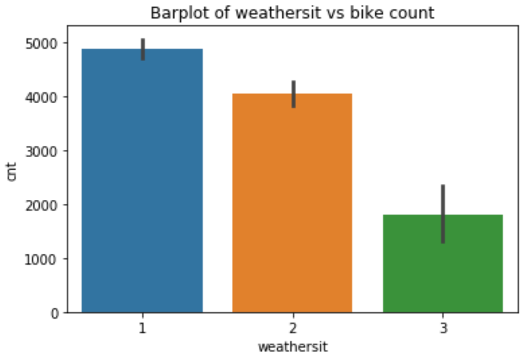




Month:

The above plot shows bar plot of month feature for all the 12 months. The count plot 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 show peak seasons with decreasing count on either side of the plot.



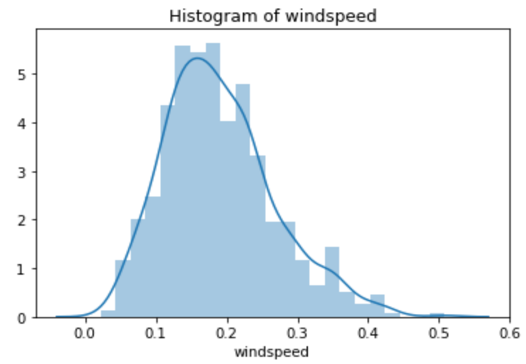


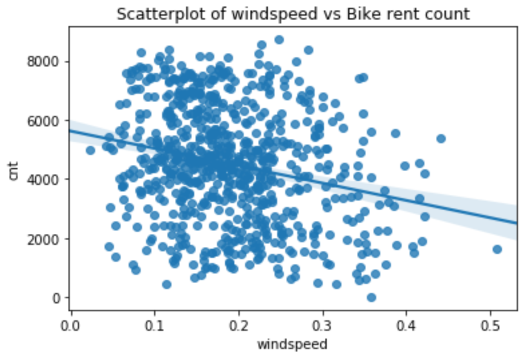
Weathersit:

Weathersit's count plot 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.

Figure 2.2 Count plot and bar-plot of categorical features

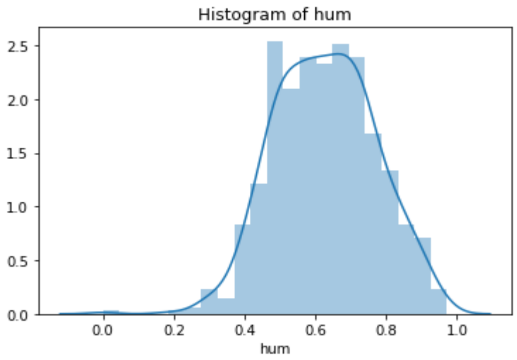
Numerical Features:

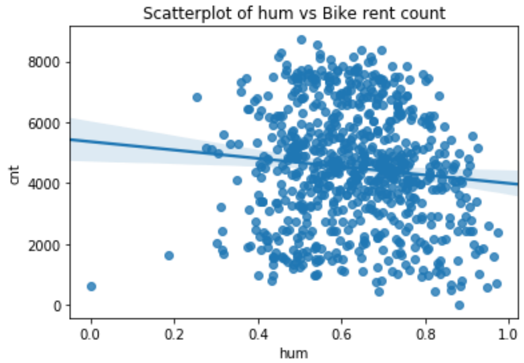




Windspeed:

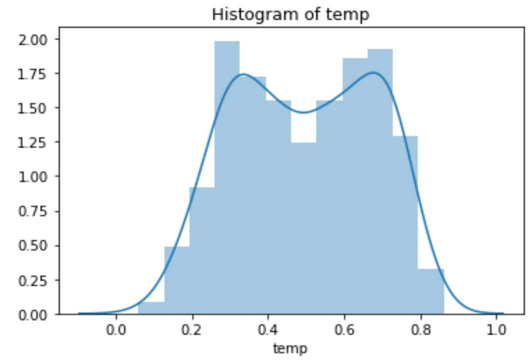
Histogram of windspeed shows slightly right skewed normal distribution. However, when we compared this feature alone with the target variable - bike count ,the distribution is little scattered with concentration mainly on the lower side of the windspeed and so is the histogram distribution concentrated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

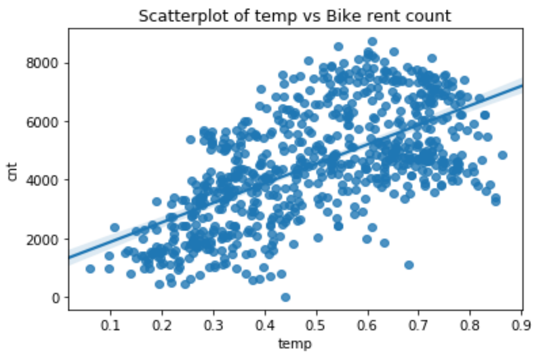




Hum:

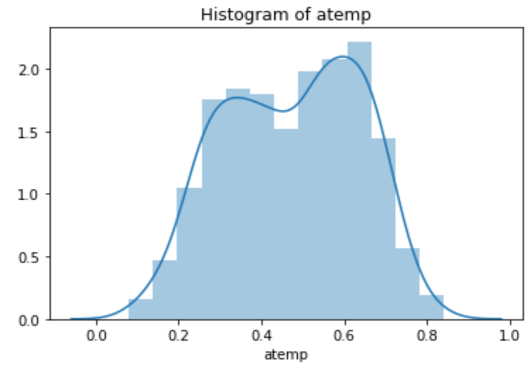
Humidity feature shows slightly left skewed histogram plot. The scatter plot of count vs humidity shows random distribution and no discernible pattern could be found out. The humidity concentration is typically in the range of 0.4-0.8 as shown in histogram as well as in the scatter plot. However, there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

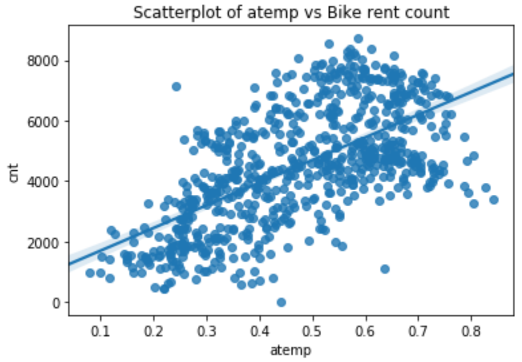




Temp:

We plot histogram of temp variable. This feature set is normalized in the range varies from 0 to 1.The scatter plot of this feature with respect to the target variable -“count” is plotted .We see that as a general trend, as the temperature increases the bike rental count also increases. Also, the relation between “temp” and “count” variable shows almost linear behavior.





Atemp:

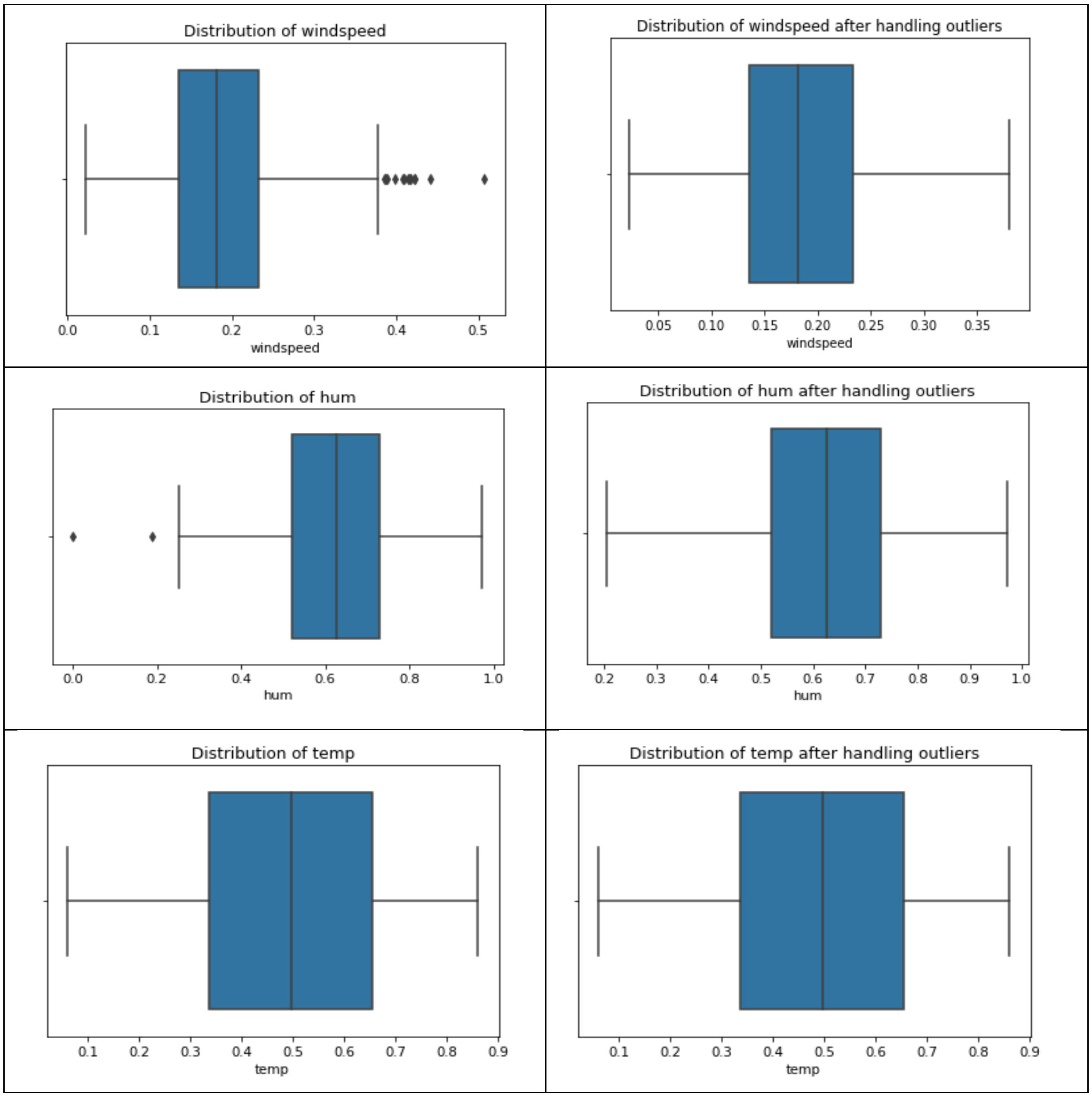
“atemp” is the normalized feeling temperature in Celsius. From the above distribution we see that “temp” and “atemp” shows similar distribution. This result was expected because the actual temperature and feeling temperature should not ideally deviate a lot. This variable also exhibits almost linear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

Figure 2.3 Count plot and bar-plot of numerical features

2.1.3 OUTLIER ANALYSIS

One of the other steps of pre-processing apart from checking for normality is the presence of outliers. In this case we use a classic approach of removing outliers, Tukey's method. 2 We visualize the outliers using boxplots.

Outliers are atypical values in the dataset. 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.



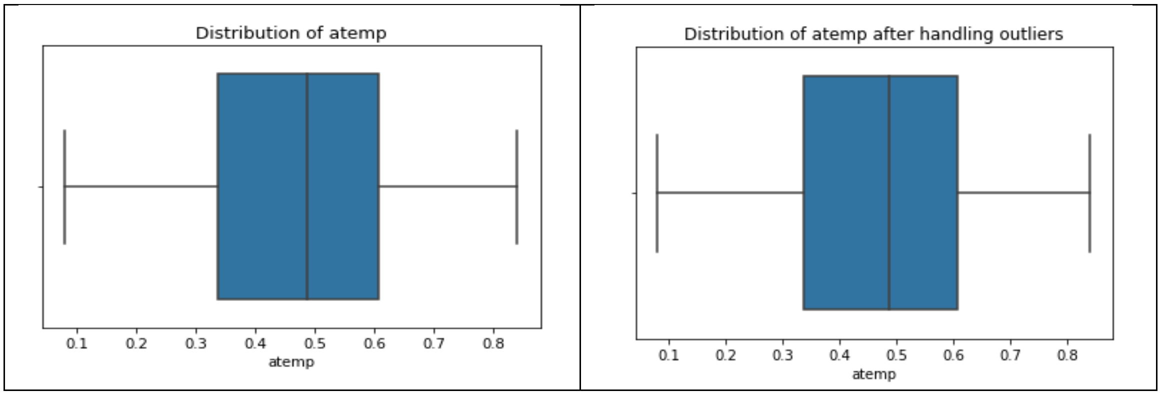


Figure 2.4 Boxplot representation of numerical features with and without outlier

2.1.4 FEATURE SCALING

Real world dataset contains features that highly vary in magnitudes, units, and range. Normalisation should be performed when the scale of a feature is irrelevant or misleading and not should Normalize when the scale is meaningful.

Before proceeding further ,we scale down the features .Most of the numerical features are normalized, 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 :

$$Xnorm = (X-Xmin)/(Xmax-Xmin)$$

2.1.5 FEATURE SELECTION

One of the key tasks in any data science operation is to choose right set of predictors. This is because ,although 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.

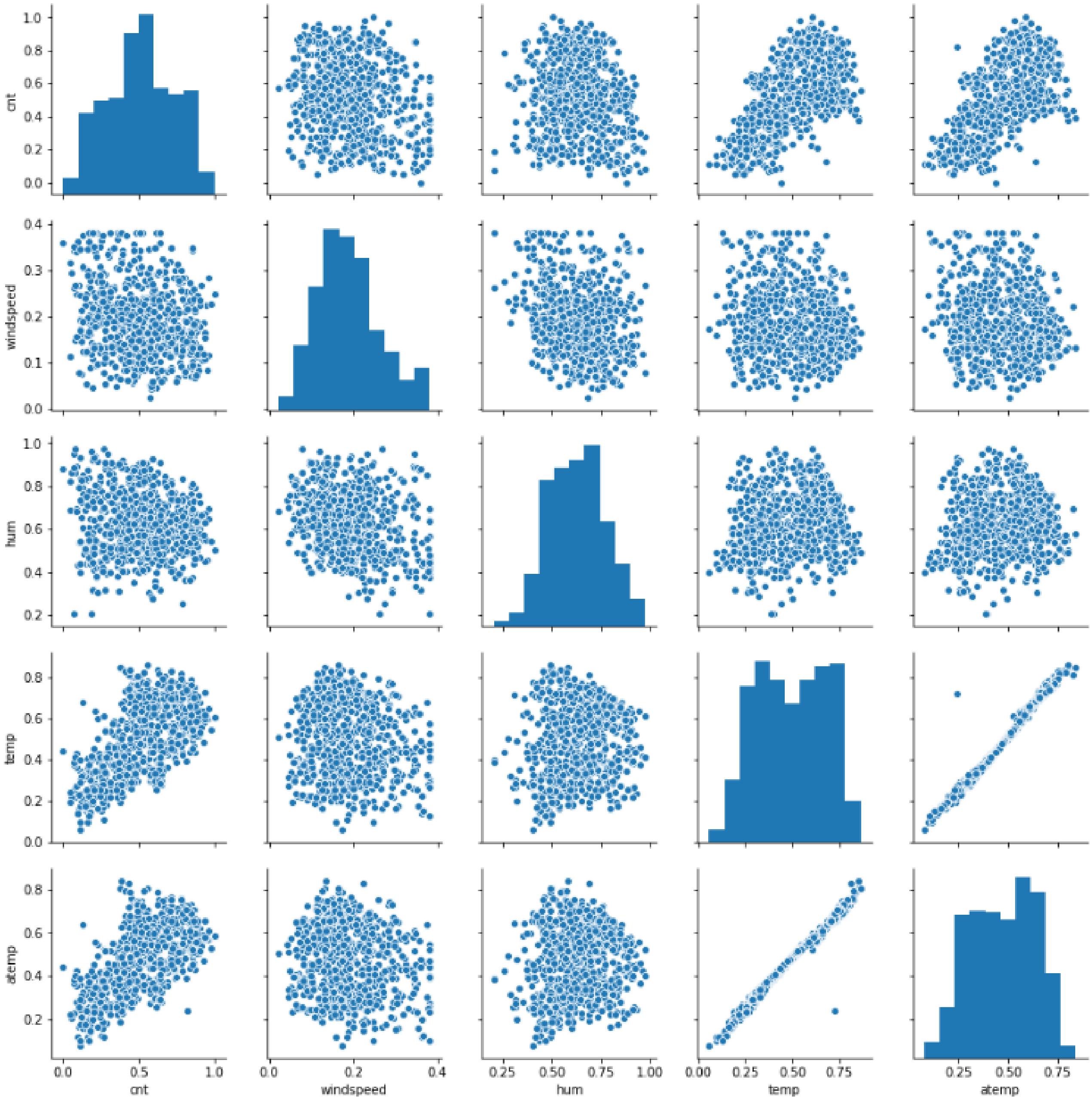
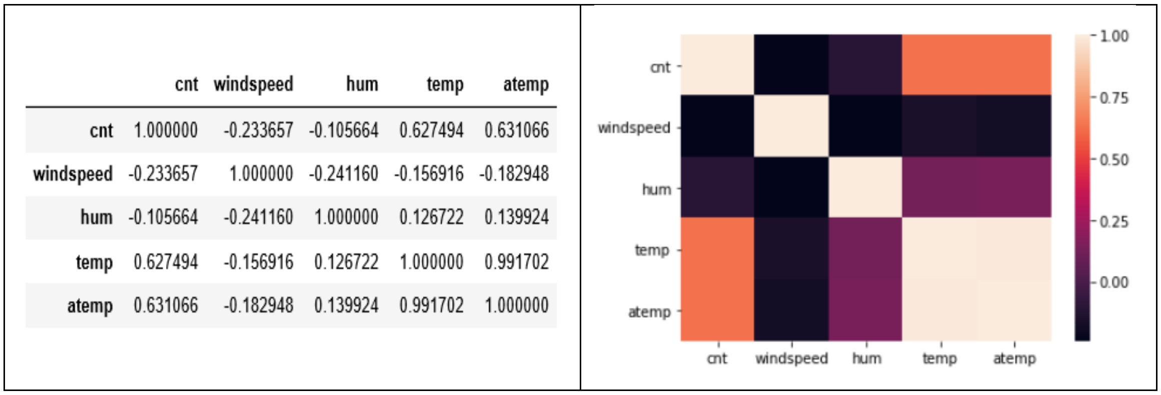


Figure 2.5 Pair plot for all the numerical variables

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

Numerical Features:



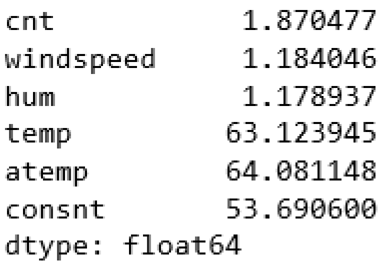
	cnt	windspeed	hum	temp	atemp
cnt	1.000000	-0.233657	-0.105664	0.627494	0.631066
windspeed	-0.233657	1.000000	-0.241160	-0.156916	-0.182948
hum	-0.105664	-0.241160	1.000000	0.126722	0.139924
temp	0.627494	-0.156916	0.126722	1.000000	0.991702
atemp	0.631066	-0.182948	0.139924	0.991702	1.000000

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

Multi - Collinearity:

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 (atemp).

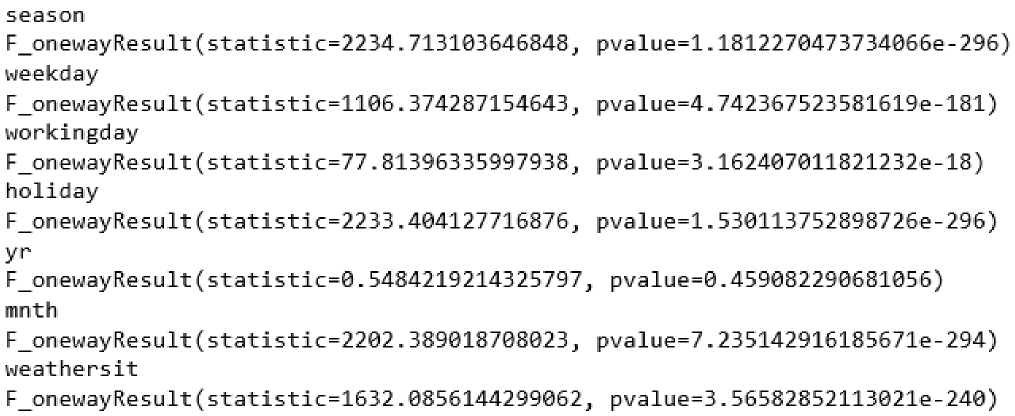


```
cnt           1.870477
windspeed     1.184046
hum           1.178937
temp         63.123945
atemp        64.081148
consnt       53.690600
dtype: float64
```

Figure 2.7 VIF factor table for numerical features

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 p-value more than 0.05 will be discarded. From the table we see that no such feature is present.



```
season
F_onewayResult(statistic=2234.713103646848, pvalue=1.1812270473734066e-296)
weekday
F_onewayResult(statistic=1106.374287154643, pvalue=4.742367523581619e-181)
workingday
F_onewayResult(statistic=77.81396335997938, pvalue=3.162407011821232e-18)
holiday
F_onewayResult(statistic=2233.404127716876, pvalue=1.530113752898726e-296)
yr
F_onewayResult(statistic=0.5484219214325797, pvalue=0.459082290681056)
mnth
F_onewayResult(statistic=2202.389018708023, pvalue=7.235142916185671e-294)
weathersit
F_onewayResult(statistic=1632.0856144299062, pvalue=3.56582852113021e-240)
```

Figure 2.8 Anova results on categorical data set

- The Anova test shows “yr” feature set having p-value greater than 0.05.Thus we remove “yr” variable.
- As cnt = casual + registered, so casual and registered gives no independent significance value to our model.
- From the visualizations holiday and weekday doesn't give extra significance compared to working day, so both can be dropped
- temp and atemp are highly correlation, so either of them can be dropped

Thus, from the analysis we drop following feature set :

  - instance
  - dteday
  - casual
  - registered
  - atemp
  - yr

2.2 DATA MODELLING

2.2.1 SPLITTING THE DATA

In statistics and machine learning we usually split our data into two subsets: training data and testing data (and sometimes to three: train, validate and test), and fit our model on the train data, in order to make predictions on the test data.

The data we use is usually split into training data and test data. The training set contains a known output and the model learns on this data in order to be generalized to other data later. We have the test dataset (or subset) in order to test our model's prediction on this subset.

For Huge data sets, split ratio: 50% train & 50% test

For Small data sets, split ratio: 80% train & 20% test

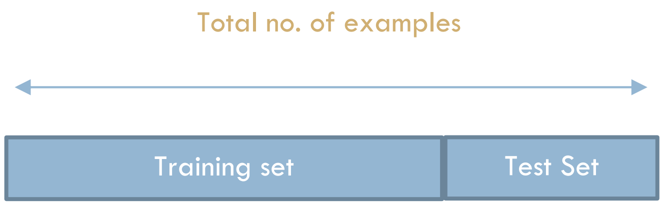


Figure 2.9 Training data and Test data

```
## Python Code

train_data, test_data = train_test_split(Bike_rent_data, test_size =  0.25)
```

2.2.2 LINEAR REGRESSION

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

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.

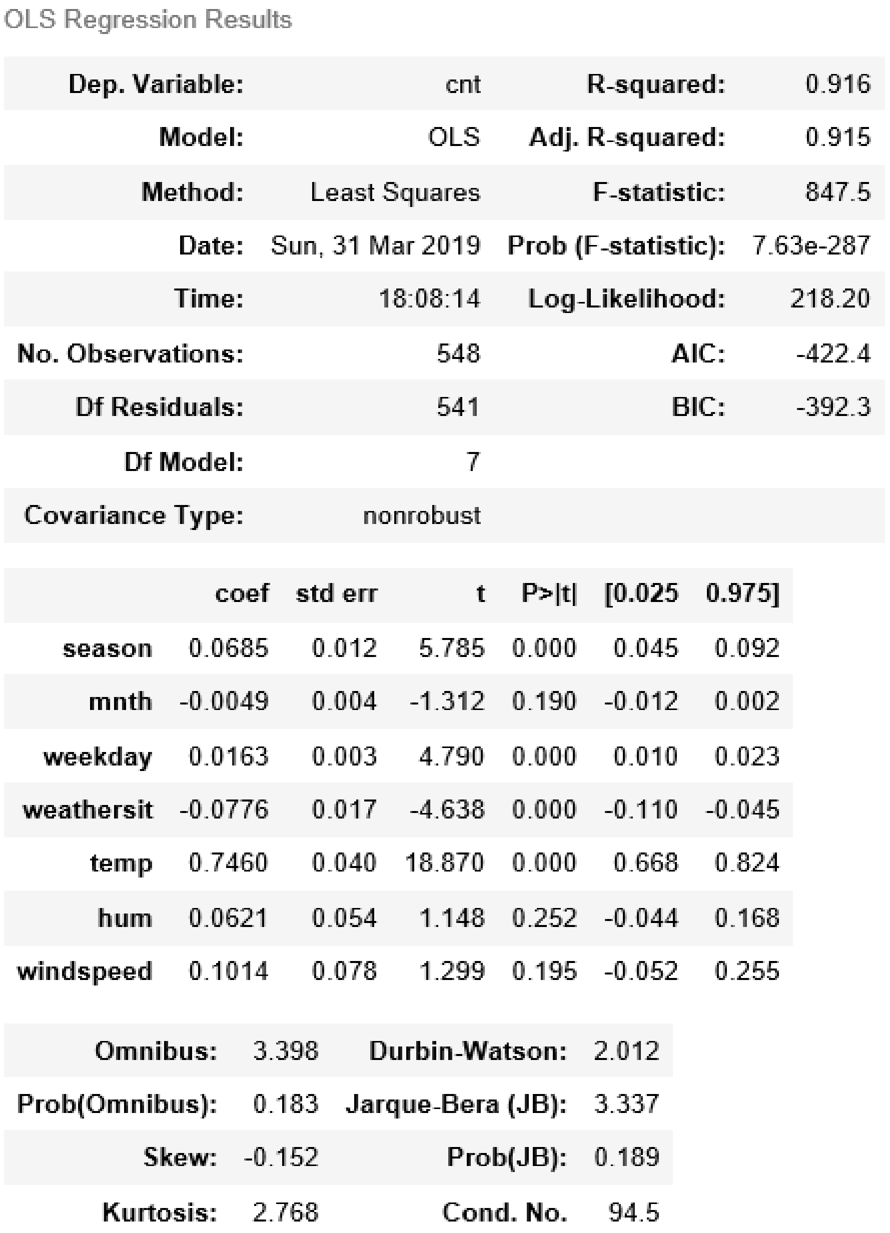
## Python Code:

```
from sklearn.linear_model import LinearRegression

Model_LR = sm.OLS(y_train,x_train).fit()

predictions_LR = Model_LR.predict(x_test)
```

Summary:



OLS Regression Results

Dep. Variable:	cnt	**R-squared:**	0.916
Model:	OLS	**Adj. R-squared:**	0.915
Method:	Least Squares	**F-statistic:**	847.5
Date:	Sun, 31 Mar 2019	**Prob (F-statistic):**	7.63e-287
Time:	18:08:14	**Log-Likelihood:**	218.20
No. Observations:	548	**AIC:**	-422.4
Df Residuals:	541	**BIC:**	-392.3
Df Model:	7		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
season	0.0685	0.012	5.785	0.000	0.045	0.092
mnth	-0.0049	0.004	-1.312	0.190	-0.012	0.002
weekday	0.0163	0.003	4.790	0.000	0.010	0.023
weathersit	-0.0776	0.017	-4.638	0.000	-0.110	-0.045
temp	0.7460	0.040	18.870	0.000	0.668	0.824
hum	0.0621	0.054	1.148	0.252	-0.044	0.168
windspeed	0.1014	0.078	1.299	0.195	-0.052	0.255

Omnibus:	3.398	**Durbin-Watson:**	2.012
Prob(Omnibus):	0.183	**Jarque-Bera (JB):**	3.337
Skew:	-0.152	**Prob(JB):**	0.189
Kurtosis:	2.768	**Cond. No.**	94.5

Output:

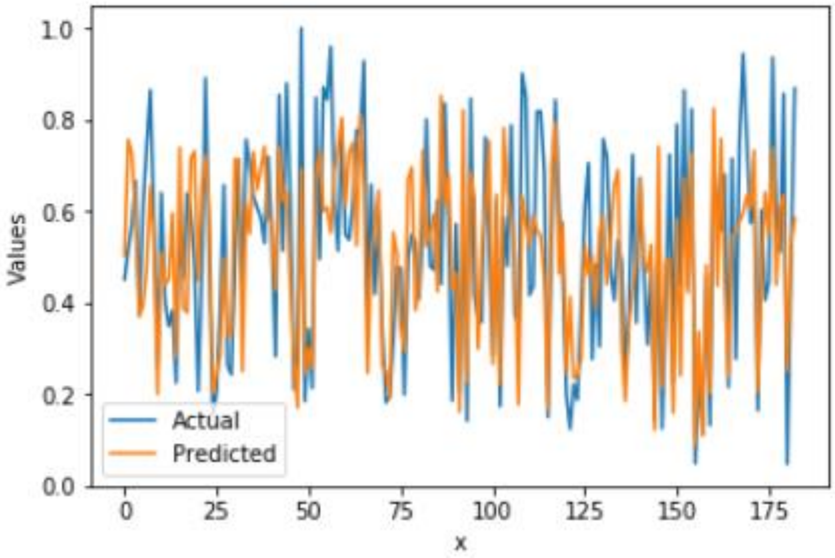


Figure 2.10 Linear Regression model Actual vs Predicted output

2.2.3 DECISION TREES

Now, we implement decision trees for the prediction of our target (cnt) variable.

##Python Code:

```
from sklearn.tree import DecisionTreeRegressor

Model_DT = DecisionTreeRegressor(max_depth = 6 ,max_features = 'auto')

Model_DT.fit(x_train,y_train)

Predictions_DT = Model_DT.predict(x_test)
```

Output:

```
R2: 0.4480515769262581 RMSE: 0.16630101287844298
```

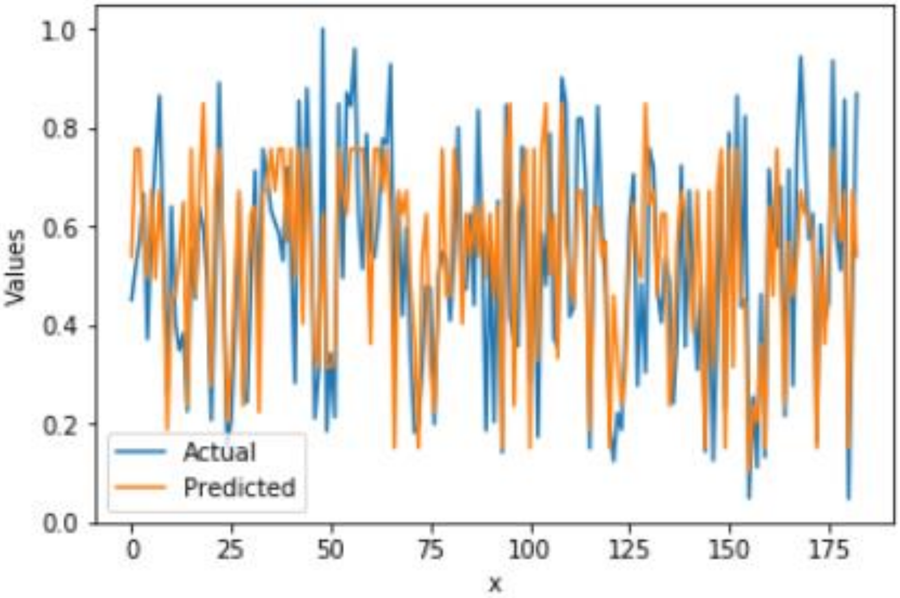


Figure 2.11 Decision Trees Actual vs Predicted output

2.2.4 RANDOM FORESTS

##Python code:

```
from sklearn.ensemble import RandomForestRegressor

Model_RF = RandomForestRegressor(n_estimators = 100,random_state=0).fit(x_train,y_train)

Predictions_RF = Model_RF.predict(x_test)
```

Output:

```
R2: 0.5948756794323142     RMSE: 0.1424753733686972
```

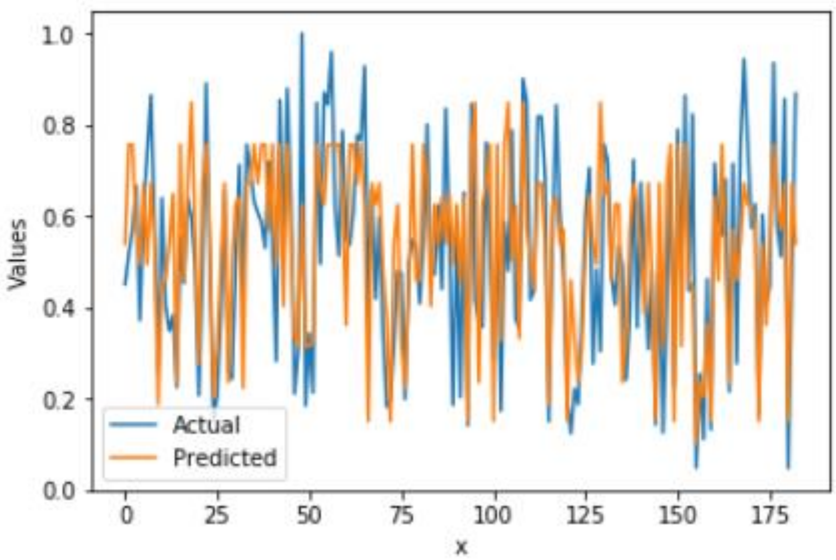


Figure 2.12 Random Forests Actual vs Predicted output

Chapter 3

Conclusion

3.1 MODEL EVALUATION

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

- Linear Regression
- Decision Trees
- Random Forests

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.

Model	RMSE (R)	R-sqr (R)	RMSE (Python)	R-sqr (Python)
Linear Regression	0.0984	0.8481	0.1714	0.4261
Decision Trees	0.0096	0.6813	0.1617	0.4894
Random Forest	0.0898	0.7853	0.1423	0.6049

Figure 3.0 RMSE of Linear Model, Decision Trees and Random Forest

From the above RMSE and R-squared results of the three algorithms, we see that Random Forest performs best in R 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

ii) https://towardsdatascience.com/

Note: Figures and References are made from Python code outputs