# Bike Renting

SWAROOP H

July 2019

## Contents



## Appendix A

# Chapter 1

# Introduction

## 1.1 Problem Statement.

The objective of this Case is to predict the number of bikes rent count based on environmental and seasonal settings.

We are provided with the daily data of bike renting count for two years (2010–2011) and we have to predict the number of bikes rented for a given seasonal and environmental settings.

## 1.2 Data Overview.

| | instant | dteday | season | yr | mnth | holiday | weekday | workingday | weathersit | temp | atemp | hum | windspeed | casual | registered | cnt |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 2011-01-01 | 1 | 0 | 1 | 0 | 6 | 0 | 2 | 0.344167 | 0.363625 | 0.805833 | 0.1604460 | 331 | 654 | 985 |
| 2 | 2 | 2011-01-02 | 1 | 0 | 1 | 0 | 0 | 0 | 2 | 0.363478 | 0.353739 | 0.696087 | 0.2485390 | 131 | 670 | 801 |
| 3 | 3 | 2011-01-03 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0.196364 | 0.189405 | 0.437273 | 0.2483090 | 120 | 1229 | 1349 |
| 4 | 4 | 2011-01-04 | 1 | 0 | 1 | 0 | 2 | 1 | 1 | 0.200000 | 0.212122 | 0.590435 | 0.1602960 | 108 | 1454 | 1562 |
| 5 | 5 | 2011-01-05 | 1 | 0 | 1 | 0 | 3 | 1 | 1 | 0.226957 | 0.229270 | 0.436957 | 0.1869000 | 82 | 1518 | 1600 |
| 6 | 6 | 2011-01-06 | 1 | 0 | 1 | 0 | 4 | 1 | 1 | 0.204348 | 0.233209 | 0.518261 | 0.0895652 | 88 | 1518 | 1606 |

Fig 1.1 : Sample data(1-16 columns)

As we look at the data , there are 16 variables and 731 observations. We have a total of 16 columns, in which we have 13 independent variables and 3 dependent variables. ‘casual’, ‘registered’ and ‘cnt’ (dependent variables) are the counting of renting bikes for a particular day. Casual counting is for the non-registered customers, registered counting is for registered customers and cnt is for total counting i.e. casual + registered.
The dependent variable or the target variable is cnt, i.e. the count of people renting bikes.

### 1.2.1 Detailed data attributes

**instant:** Record index

**dteday:** Date(In the format yyyy-mm-dd)

**season:** Season (1:springer, 2:summer, 3:fall, 4:winter)

**yr:** Year (0: 2011, 1:2012)

**mnth:** Month (1 to 12)

**holiday:** weather day is holiday or not (extracted fromHoliday Schedule)

**weekday:** Day of the week

**workingday:** If day is neither weekend nor holiday is 1, otherwise is 0.

**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

**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

As we can see here we have a time-series dataset of two years (2010 – 2011), where we have different variables including environmental factors (Temperature, season, humidity, windspeed, etc), which can determine if a person will rent the bike on that day or not. We'll analyse each variable dependency on the target variable as well as the other variables and then we'll make our machine learning model on those variable to predict the bike rental count of a particular day.

# Chapter 2

# Data Interpretation and Visualisations

## 2.1 Exploratory Data Analysis [EDA]

Exploratory Data Analysis refers to the critical process of performing initial investigations on data so as to discover patterns, to spot anomalies, to test hypothesis and to check assumptions with the help of summary statistics and graphical representations.

### 2.1.1 Variable Identification

First we need to identify **Predictor** (Input) and **Target** (output) variables. Next, identify the data type and category of the variables.

First, we take a look at the structure of data (***br is the bike renting data)

```
> str(br)
'data.frame':    731 obs. of  16 variables:
 $ instant   : int  1 2 3 4 5 6 7 8 9 10 ...
 $ dteday    : Factor w/ 731 levels "2011-01-01","2011-01-02",..: 1 2 3 4 5 6 7...
 $ season    : int  1 1 1 1 1 1 1 1 1 1 ...
 $ yr        : int  0 0 0 0 0 0 0 0 0 0 ...
 $ mnth      : int  1 1 1 1 1 1 1 1 1 1 ...
 $ holiday   : int  0 0 0 0 0 0 0 0 0 0 ...
 $ weekday   : int  6 0 1 2 3 4 5 6 0 1 ...
 $ workingday: int  0 0 1 1 1 1 1 0 0 1 ...
 $ weathersit: int  2 2 1 1 1 1 2 2 1 1 ...
 $ temp      : num  0.344 0.363 0.196 0.2 0.227 ...
 $ atemp     : num  0.364 0.354 0.189 0.212 0.229 ...
 $ hum       : num  0.806 0.696 0.437 0.59 0.437 ...
 $ windspeed : num  0.16 0.249 0.248 0.16 0.187 ...
 $ casual    : int  331 131 120 108 82 88 148 68 54 41 ...
 $ registered: int  654 670 1229 1454 1518 1518 1362 891 768 1280 ...
 $ cnt       : int  985 801 1349 1562 1600 1606 1510 959 822 1321 ...
```

Fig 2.1 structure of data

We have datatype as factor/object for dteday and rest others have numerical (int and Float). We can also see the unique values of each variables.

**Time-based variables:** *'dteday', 'yr', 'mnth', 'season', 'weekday'*
Now, what should be considered for them either continuous or categorical?

***'dteday'*** has a date of every day, It has 31 values, so consider it as a category.

***'yr'***: we have two value 0 for 2010 and 1 for 2011, so we can consider it as categorical.

***'mnth':*** can we consider a month like jan, feb, march…, dec. It has 12 values, so consider it as a category, will introduce 11 dimensions to our dataset. So, we will make bins for this column in the feature engineering section.

***'season'***: has four unique values, so we would consider it as a category.

***'weekday'***: Its same like mnth, so we will make bins for this in the feature engineering section.

***'holiday'*** and ***'workingday'*** having value 0 for nonholiday/ non-working day 1 for the opposite. So it would be our categorical variable.

***'weathersit'***: it containing 4 unique values, so we can consider it as categorical.

**Continuous Variables:** 'temp', 'atemp', 'hum', 'windspeed' are continuous values, and in our dataset, they are in a normalized format.

**Target variables:** 'casual', 'registered' and 'cnt' are our target variables and in continuous form. So our problem would be a regression problem.

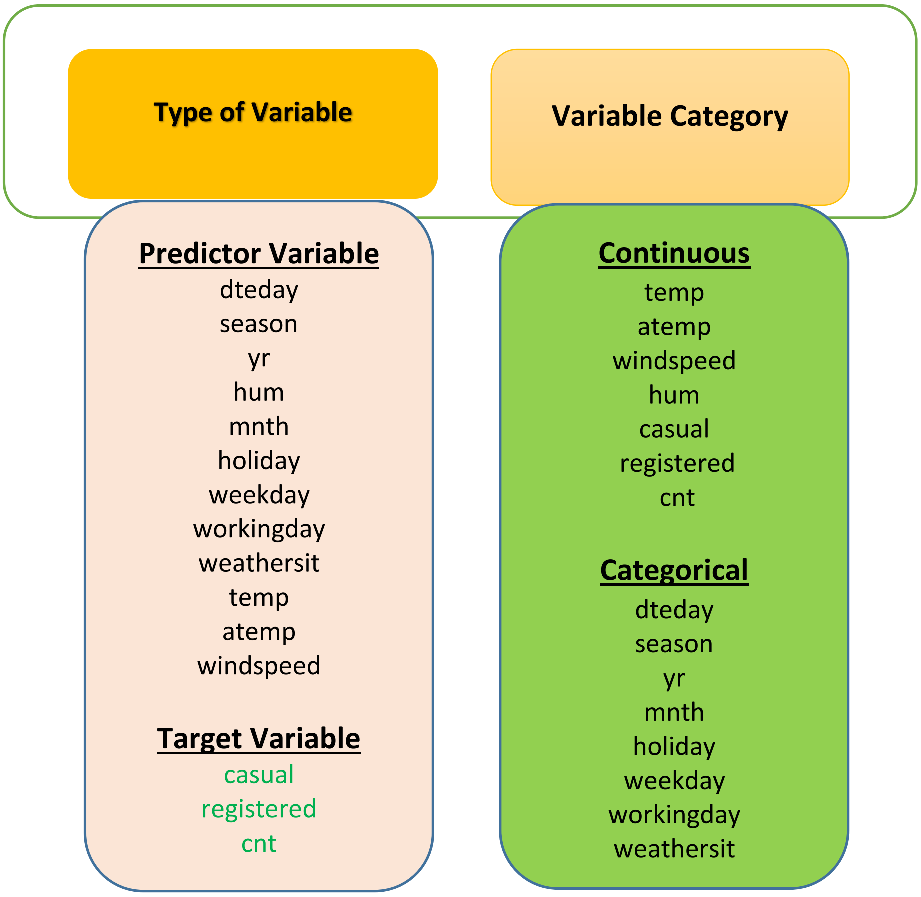



Fig 2.2 Variable identification

## 2.1.2 Univariate Analysis

Understand the distribution of numerical variables and generate a frequency table for numeric variables. Now, I'll test and plot a histogram for each numerical variables and analyze the distribution.

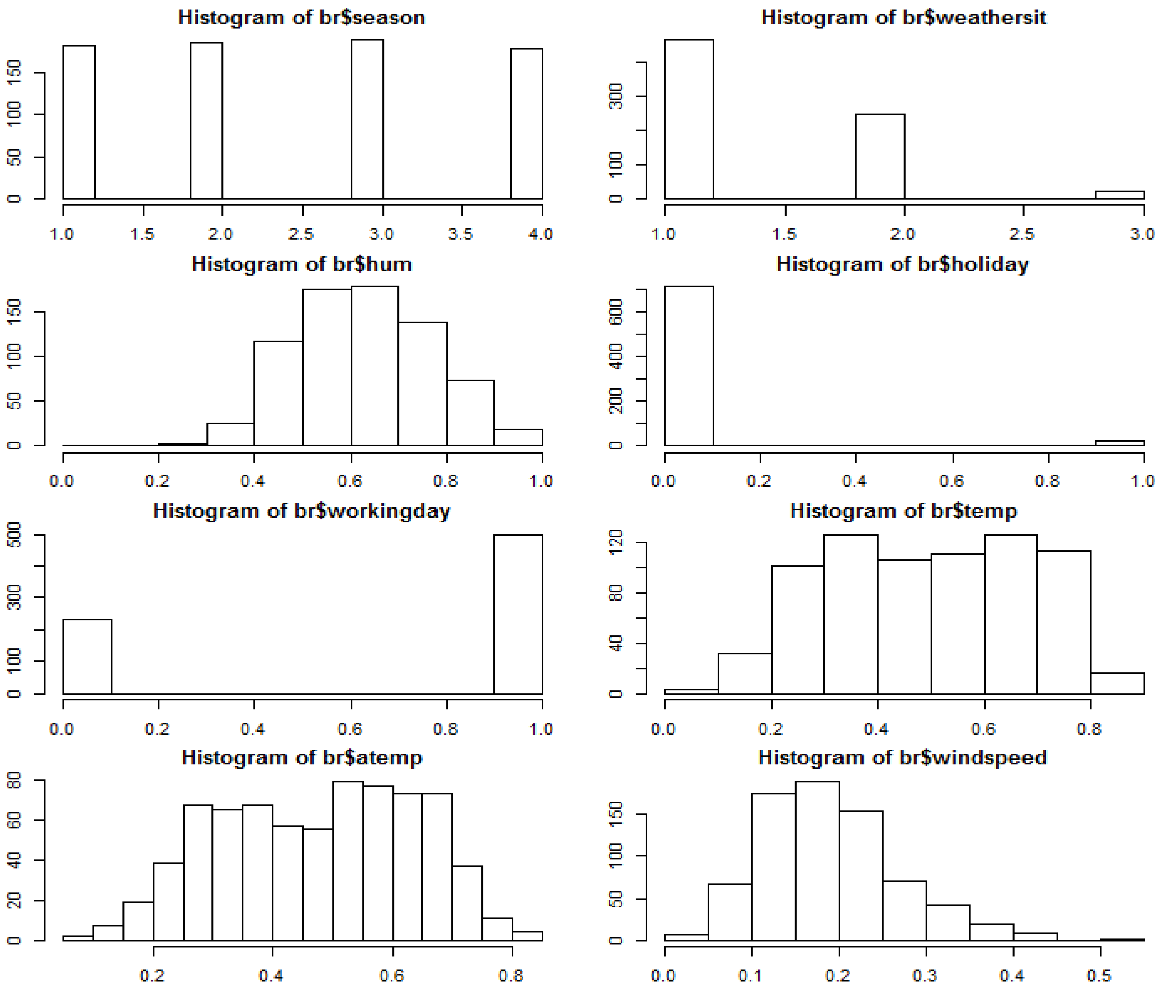



Fig 2.3 Numerical variable distribution

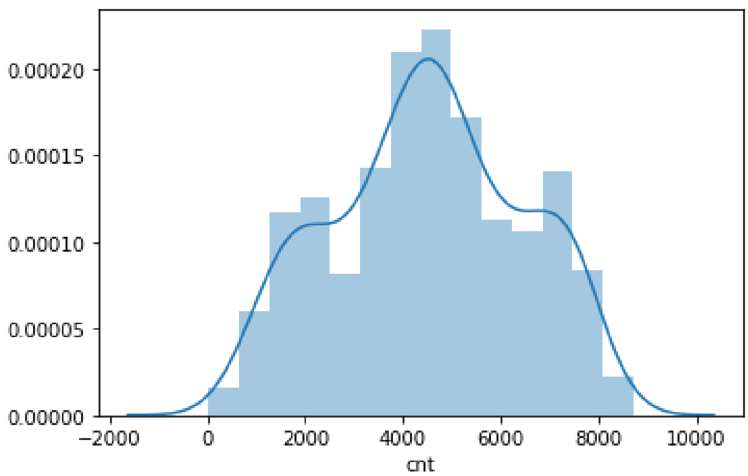
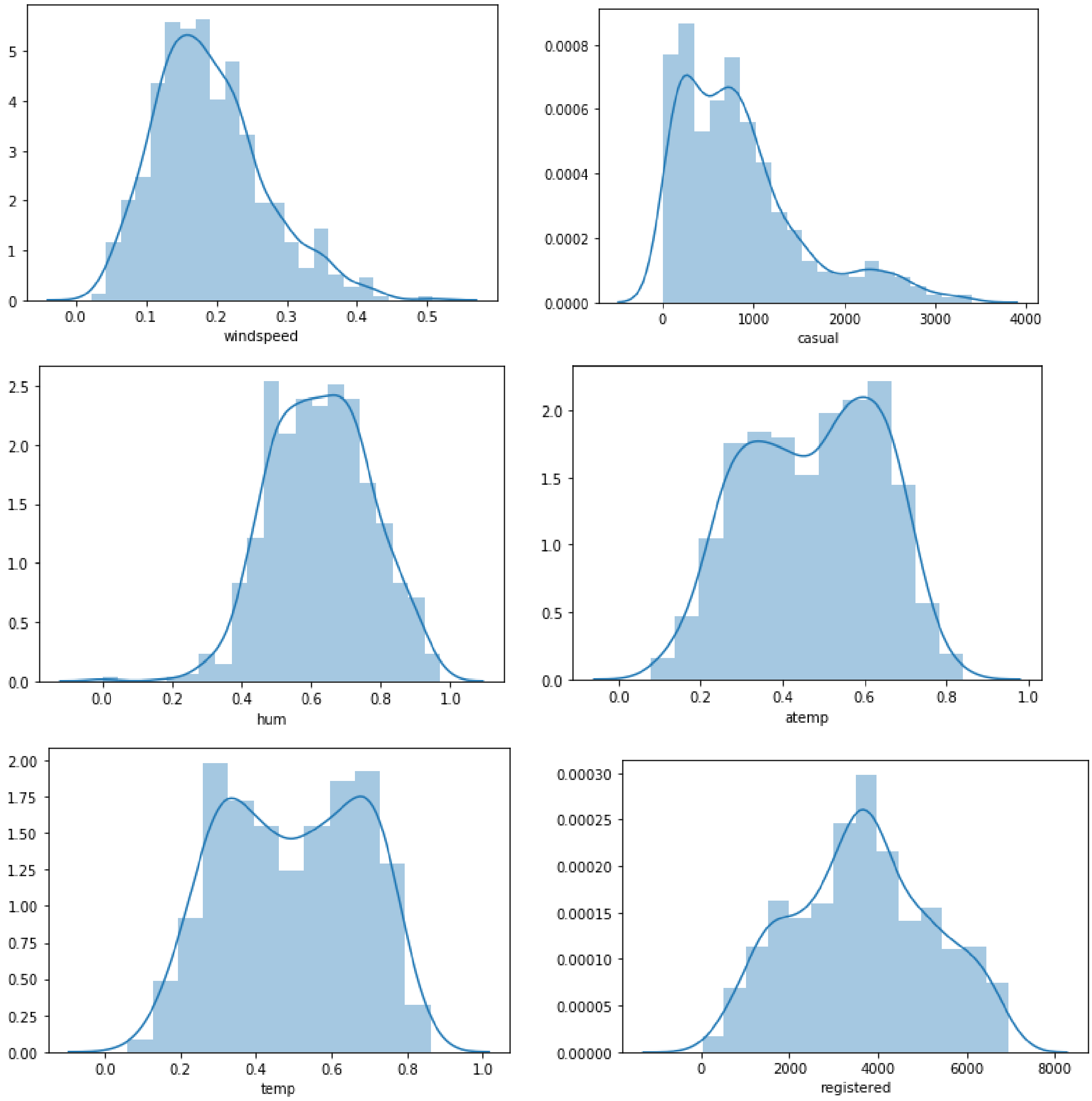



Fig 2.4 Distribution of target variable CNT

*Fig 2.5 probability density functions*




- ✓ *'Season'* has four categories of almost equal distribution
- ✓ *'Weather'* 1 has higher contribution i.e. mostly clear weather.
- ✓ As expected, mostly working days and variable holiday is also showing a similar inference.
- ✓ Variables temp, *atemp, humidity, windspeed* and *'registered'* looks naturally distributed.
- ✓ Target variable '*cnt*' is normally distributed.
- ✓ '*hum*' data is slightly skewed to the left , here data is already in normalize form so outliers are discarded
- ✓ *'casual'* data is slightly skewed to the right so, there is chances of getting outliers.
- ✓ Here we observe that the holiday variable is highly imbalanced, 21 holidays and 710 non-holidays which may prove to be non-important for our model development. We'll be looking forward to this in feature selection and engineering section.

| | temp | atemp | hum | windspeed | casual | registered | cnt |
|---|---|---|---|---|---|---|---|
| **count** | 731.000000 | 731.000000 | 731.000000 | 731.000000 | 731.000000 | 731.000000 | 731.000000 |
| **mean** | 0.495385 | 0.474354 | 0.627894 | 0.190486 | 848.176471 | 3656.172367 | 4504.348837 |
| **std** | 0.183051 | 0.162961 | 0.142429 | 0.077498 | 686.622488 | 1560.256377 | 1937.211452 |
| **min** | 0.059130 | 0.079070 | 0.000000 | 0.022392 | 2.000000 | 20.000000 | 22.000000 |
| **25%** | 0.337083 | 0.337842 | 0.520000 | 0.134950 | 315.500000 | 2497.000000 | 3152.000000 |
| **50%** | 0.498333 | 0.486733 | 0.626667 | 0.180975 | 713.000000 | 3662.000000 | 4548.000000 |
| **75%** | 0.655417 | 0.608602 | 0.730209 | 0.233214 | 1096.000000 | 4776.500000 | 5956.000000 |
| **max** | 0.861667 | 0.840896 | 0.972500 | 0.507463 | 3410.000000 | 6946.000000 | 8714.000000 |

Fig 2.6 Summary of the numerical variables.

### 2.1.3 Bi-variate Analysis

Bi-variate Analysis finds out the relationship between two variables**.**

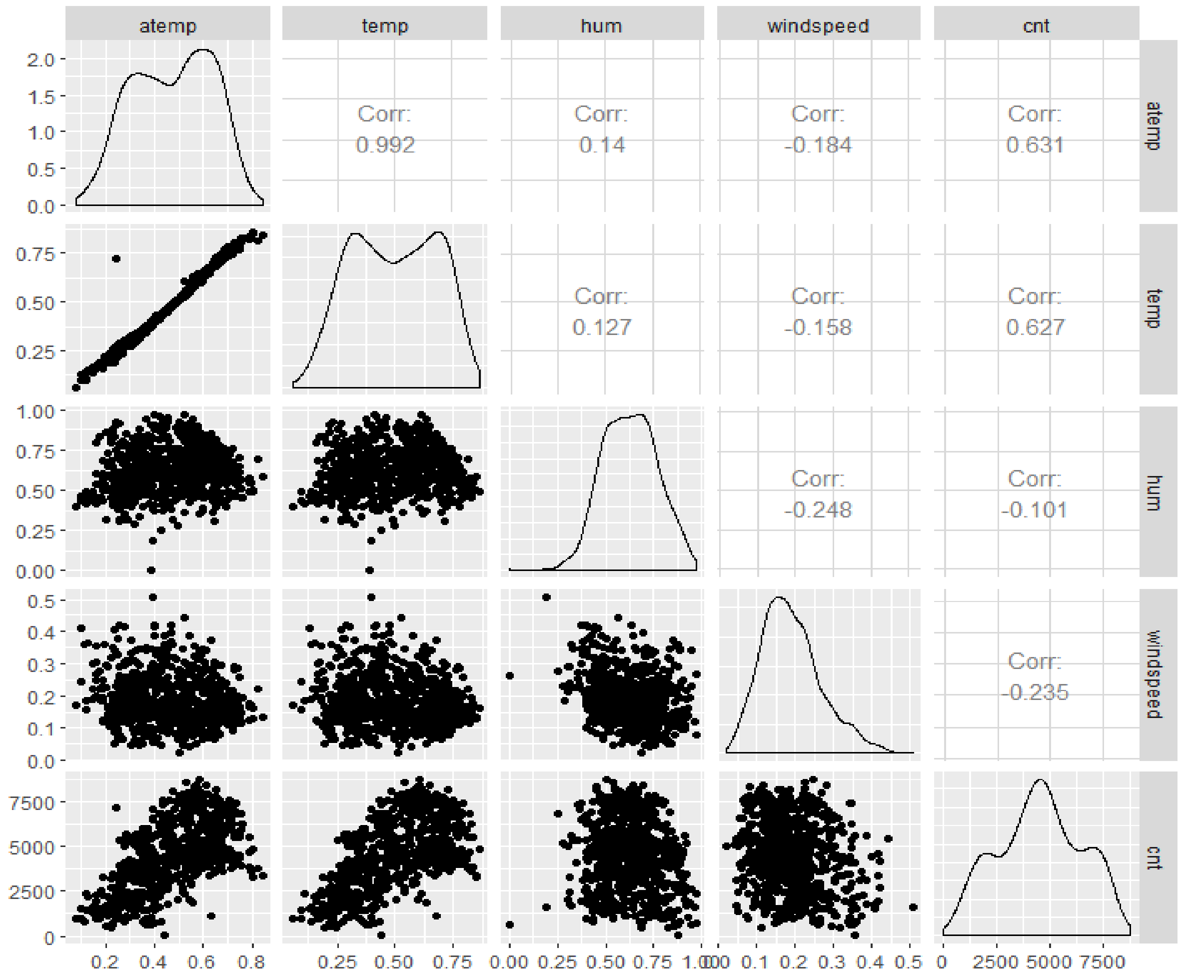



Fig 2.7 Pair plot

The above plot shows that less negative relationship between 'cnt'-'hum' and cnt-windspeed and there is strong positive relationship between temp- cnt and atemp-cnt.

Between independent variables ‘temp’ and ‘atemp’ there is strong positive relationship which creates multi-collinearity. Which should be avoided feature selection and engineering section.

The collinearity between ‘temp’ and ‘atemp’ can be observed below scatter plot.

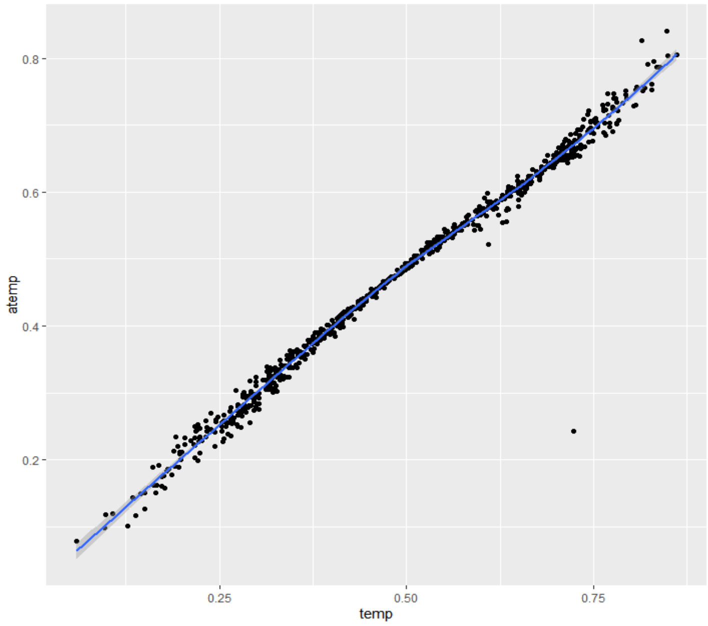
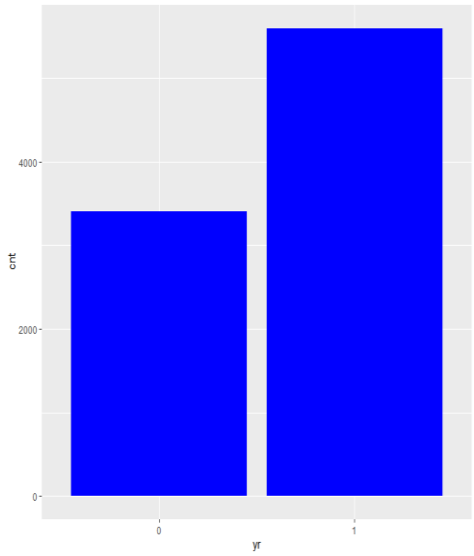
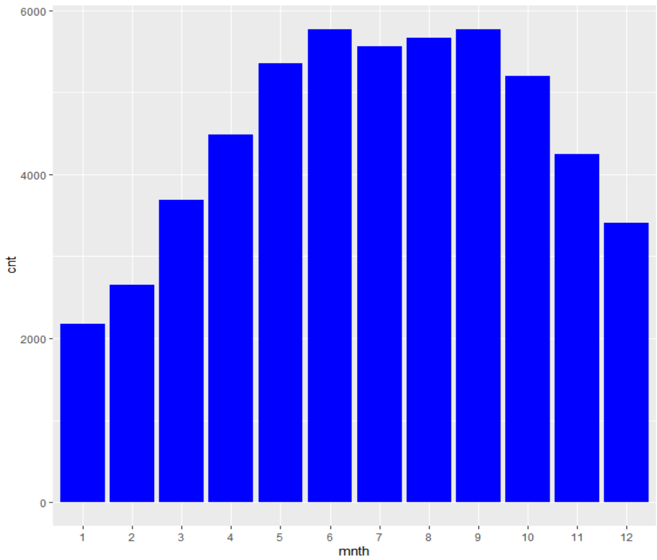



Fig 2.8

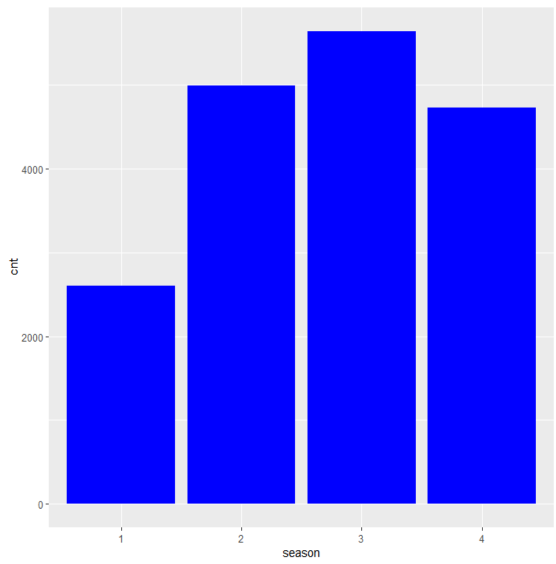


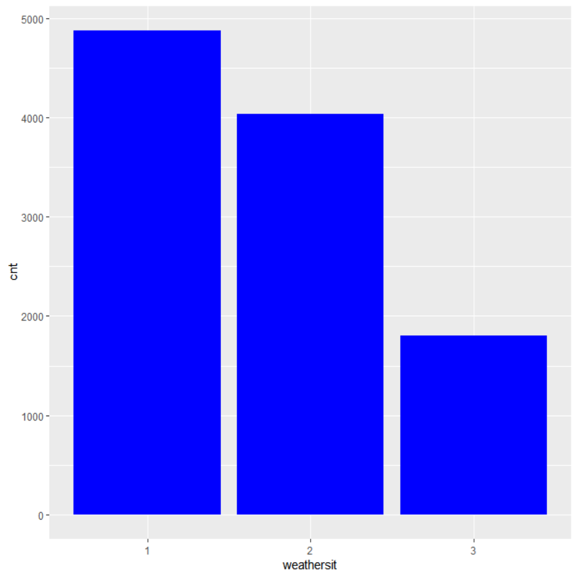

2.9(a) 2.9(b)

2.9(a) Here we see there are more Bike rentals in 2011 as compared to 2010. Now we’ll see if all the other categorical variables follow the same pattern for both years.

2.9(b), The Visualisations shows that an increase in the number of bike rentals in June, July, and August and following a decrease in the months of Winter season.




2.9(c)




2.9(d)

2.9(c) These plots show that people prefer to rent bikes mostly in fall season and least in the spring season and follow the same pattern for both years.

2.9(d) These visualizations clearly show that people prefer to rent a bike when the weather is clear and least when the weather is snowy and it is raining heavily.

**So, the conclusion from the visualizations and data analysis is that people prefer to rent a bike when the sky is clear or a little cloudy and most in the fall season. Weekdays / Weekends doesn't really affect the bike rental count.**

# Chapter 3

## Missing Value Analysis

Missing value analysis helps address several concerns caused by incomplete data. If cases with missing values are systematically different from cases without missing values, the results can be misleading.

These values can be computed by various methods like KNN, mean, median, mode etc. But here in our case luckily, there are no missing values, Therefore, we don't have to impute missing values.

Below output table illustrate no missing value present in the data.

| | Missing_Val |
|---|---|
| instant | 0 |
| dteday | 0 |
| season | 0 |
| yr | 0 |
| mnth | 0 |
| holiday | 0 |
| weekday | 0 |
| workingday | 0 |
| weathersit | 0 |
| temp | 0 |
| atemp | 0 |
| hum | 0 |
| windspeed | 0 |
| casual | 0 |
| registered | 0 |
| cnt | 0 |

Fig 3.1 output shows there are no missing values

# Chapter 4

# Outlier Analysis

Outlier detection and treatment is always a tricky part especially when our dataset is small. The box plot method detects outlier if any value is greater than (Q3 + (1.5 * IQR) ) or less than ( Q1 — (1.5 * IQR) ).
where
Q1 > 25% of data are less than or equal to this value
Q2 or Median -> 50% of data are less than or equal to this value
Q3 > 75% of data are less than or equal to this value
IQR (Inter Quartile Range) = Q3 — Q1

**All numeric variable are in normalize form so , no need to analysing Outliers, here the six numeric variables are present out of six four variables are in normalize form *temp, atem, hum, windspeed* are in normalize form no need to check outliers.**

Let us check for boxplot for curiosity.

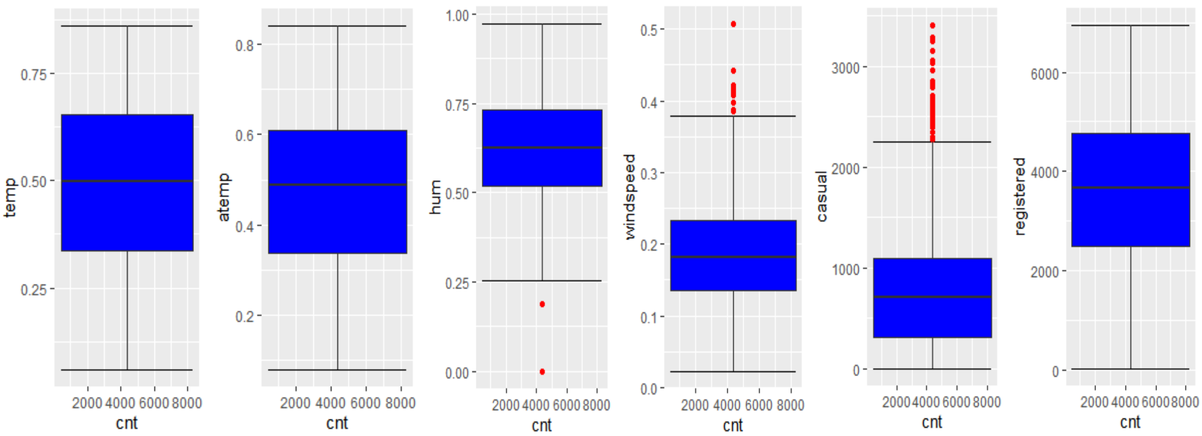



Fig 4.1 Boxplot to visualize outliers

The visualizations show no outliers in temp and atemp variables with few in humidity and quite a few in windspeed variable.
Usually, in the boxplot method, it only removes the 1% data from the whole dataset which seems to be an anomaly in the data. We assume that since the data is too small, it won't affect our whole dataset and modeling process.
The boxplot method would remove that data point, but that data point could be an important predictor.
***The numeric variables are normalized. So, removing outliers in these case won't be a good idea as we might be removing some important data from our dataset.***

## Chapter 5
## Feature Selection and Feature Engineering

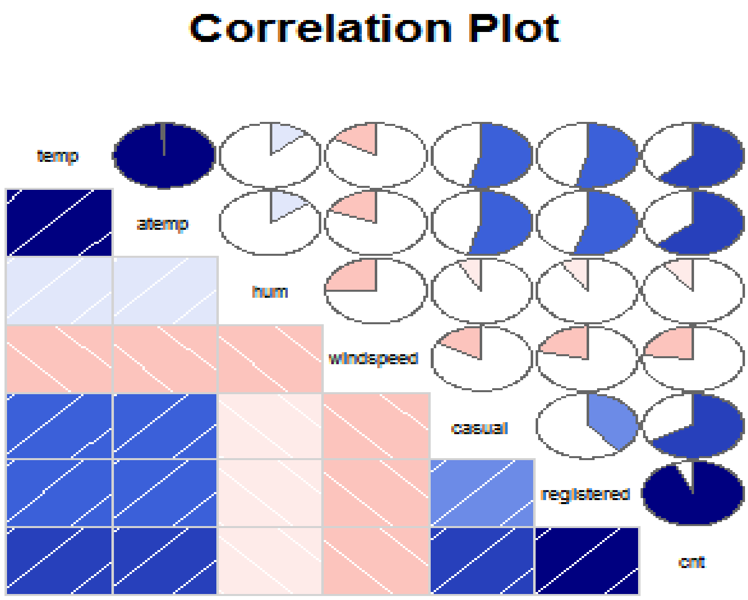



Fig 5.1 Correlation plot

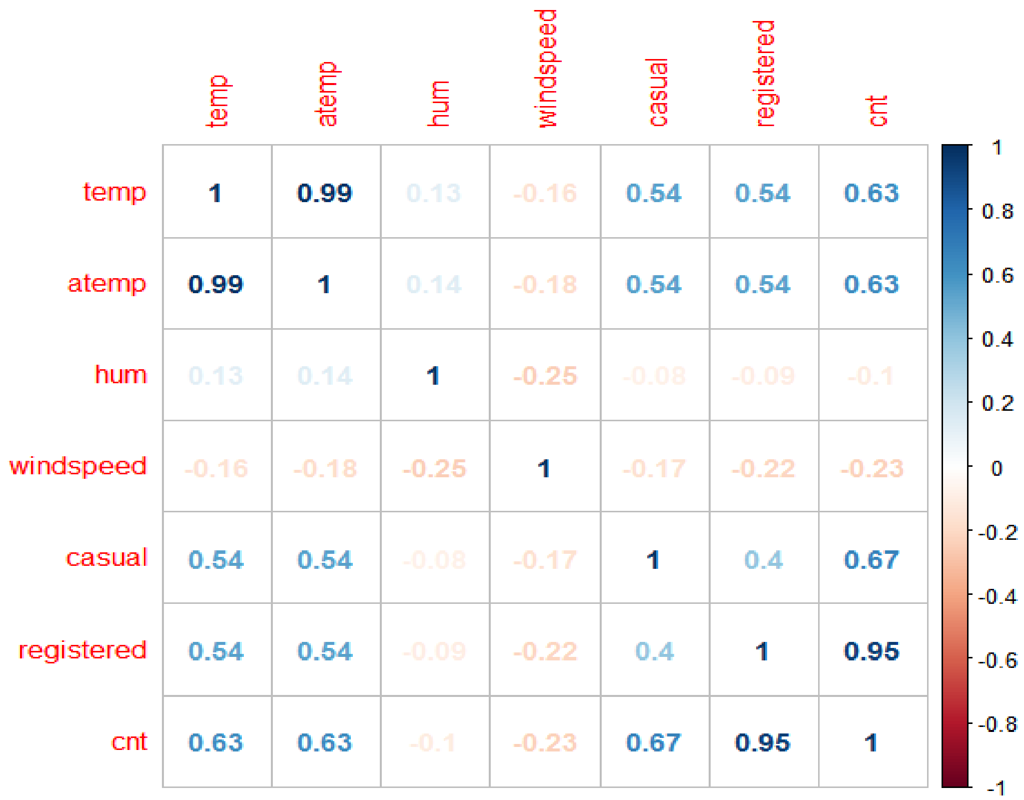



Fig 5.2 heatmap of correlation between the continuous variables

These plots show collinearity between
> *temp* and atemp,
> *cnt* and registered and casual

From heatmap and correlation plot we can see that there is multicollinearity present between *temp* and *atemp*, so we will drop one of those columns as per the importance of features.

Multicollinearity present between *registered* and *cnt*, but these are our target variable. We will only use total counting i.e. *cnt* as our target variable and would drop *casual* and *registered* column.

<u>Removed variables:</u>
*Instant* (non important variable)
*atemp* (colinearity with *temp*)
*registered / casual* (colinearity with *cnt*)

### 5.2 Feature Scaling

As discussed, already have numerical variables (temp, hum and windspeed) which are already in normalized form. By normalizing these variables are in the same range, So, we are not required to scale or normalize the data further.

## Chapter 6

# Modelling

Now, that our dataset is free from any anomaly, colinearity, and is scaled to the same range. We are ready to make machine learning models to predict bike rental count.

## Approach

We'll be using various different machine learning algorithms to make different models. We'll be comparing amongst themselves using different methodologies and after selecting the machine learning model we'll be tuning some parameter to make model best as possible.

The Models we build here are

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

## 6.1 Decision tree regression

In this model we have divided the dataset into train and test part using random sampling. Where train contains 80% data of data set and test contains 20% data and contains 12 variable where 12th variable is the target variable.

In R

```
###########Decision tree regression #################
fit = rpart(cnt ~ ., data = train, method = "anova")
predictions_DT = predict(fit, test[,-12])
```

In Python

```
#Decision tree for regression
fit_DT = DecisionTreeRegressor(max_depth=2).fit(train.iloc[:,0:11],train.iloc[:,11])
predictions_DT = fit_DT.predict(test.iloc[:,0:11])
```

```
> fit
n= 584

node), split, n, deviance, yval
      * denotes terminal node

 1) root 584 2221918000 4483.567
   2) temp< 0.432373 241  538630600 3018.900
     4) yr=0 130  134578400 2209.023
       8) season=1,2 91   29274800 1709.011 *
       9) season=4 39   29466600 3375.718 *
     5) yr=1 111  218922500 3967.405
      10) season=1 62   63496370 3150.000
        20) temp< 0.2804165 28   15397810 2445.500 *
        21) temp>=0.2804165 34   22757040 3730.176 *
      11) season=2,4 49   61584940 5001.673
        22) dteday=22,23,24,25,27 7   13697040 3093.571 *
        23) dteday=01,02,03,04,05,06,07,08,09,10,11,12,13,14,15,16,17,18
,19,20,21,26,28,29,30,31 42   18154270 5319.690 *
   3) temp>=0.432373 343  803022900 5512.676
     6) yr=0 158  113462300 4272.785
      12) hum>=0.886187 12    7482681 2767.500 *
      13) hum< 0.886187 146   76554190 4396.507 *
     7) yr=1 185  239214000 6571.611
      14) hum>=0.8322915 10   24398430 4274.700 *
      15) hum< 0.8322915 175  159042900 6702.863
        30) mnth=2,3,4,5,11,12 67   54942150 6210.418 *
        31) mnth=6,7,8,9,10 108   77773510 7008.361 *
```

Fig 6.1 Decision tree regression patterns or Rules

The above fig 5.1 shows the Decision tree regression model patterns or Rules on which test data are predict the target value.

## 6.2 Random Forest

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 80% data of data set and test contains 20% data and contains 12 variable where 12th variable is the target variable.

In R

```
#############Random Forest Model##########################
RF_model = randomForest(cnt ~ ., train, importance = TRUE, ntree = 200)
predictions_RF = predict(RF_model, test[,-12])
plot(RF_model)
```

In Python

```
#random forest
RFmodel = RandomForestRegressor(n_estimators = 200).fit(train.iloc[:,0:11], train.iloc[:,11])
RF_Predictions = RFmodel.predict(test.iloc[:,0:11])
```

Before performing any type of modeling we need to assess the importance of each predictor variable in our analysis. There is a possibility that many variables in our analysis are not important at all to the problem of class prediction. There are several methods of doing that. Below we have used Random Forests to perform features selection.

```
> varimp=importance(RF_model)
> varimp
             %IncMSE IncNodePurity
dteday     -3.041277     188367265
season     17.741681     271036743
yr         59.278176     547823699
mnth       16.136815     295674267
holiday     1.334991       4874040
weekday     3.008456      54587840
workingday  4.784316       8899118
weathersit 12.207384      78916620
temp       24.745749     539171232
hum        15.109335     110064899
windspeed   5.877273      76962028
```

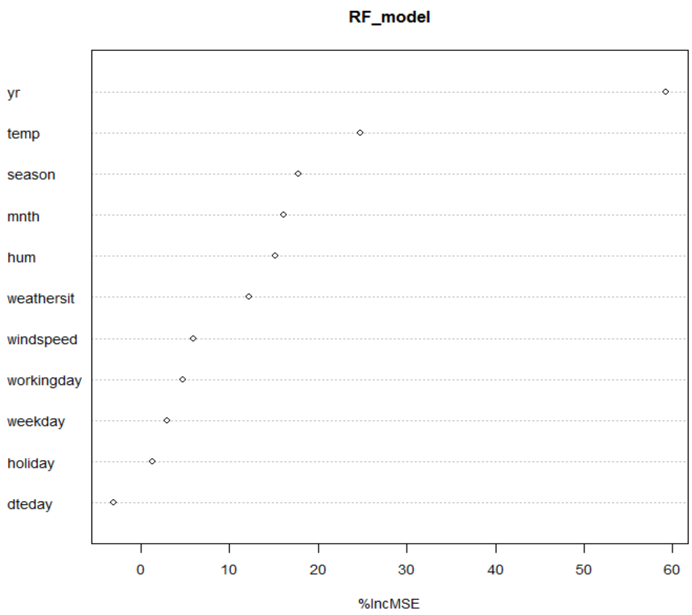



fig 6.2 variable importance plot

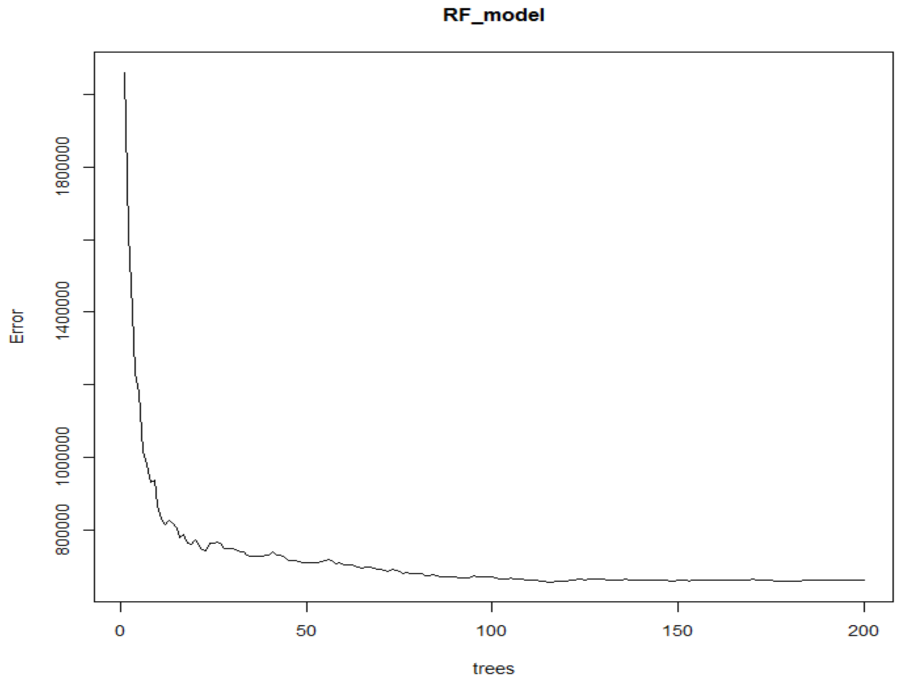



Fig 6.2 error rate vs no. of trees

The above fig6.3 represents the curve of error rate as the number of trees increases. After 200 trees the error rate reaches to be constant. In this model we are using 200 trees to predict the target variable.

## 6.3 Linear Regression

In this  linear regression model we have divided the categorical variable which have more than 2 classes into dummy variable. So that all categorical variable should be in binary classes form. On creating dummy variable there are 64 variable in both R and Python. Where 64th is the target variable. The data further divided into train and test with 80 % train data and 20 % test data using random sampling.

In R

```
#Linear regression model making
lm_model = lm(cnt ~., data = train_lr)
predictions_LR = predict(lm_model,test_lr[,-64])
```

```
lm_model = lm(cnt ~., data = train_lr)
> summary(lm_model)

Call:
lm(formula = cnt ~ ., data = train_lr)

Residuals:
    Min      1Q  Median      3Q     Max 
-3729.3  -370.2    41.4   487.0  2309.1 

Coefficients: (6 not defined because of singularities)
             Estimate Std. Error t value Pr(>|t|)    
(Intercept)   1909.40     486.15   3.928 9.72e-05 ***
dteday_01     -531.15     284.81  -1.865 0.062749 .  
dteday_02     -159.53     289.61  -0.551 0.581970    
dteday_03     -112.85     284.88  -0.396 0.692157    
dteday_04      -40.57     291.86  -0.139 0.889507    
dteday_05     -170.80     286.98  -0.595 0.551986    
dteday_06     -115.26     282.06  -0.409 0.682970    
dteday_07     -197.45     295.10  -0.669 0.503744    
dteday_08     -235.96     293.43  -0.804 0.421677    
dteday_09      -88.05     287.58  -0.306 0.759590    
dteday_10      -99.86     286.18  -0.349 0.727256    

……
dteday_27     -417.70     284.38  -1.469 0.142490    
dteday_28     -316.95     290.60  -1.091 0.275906    
dteday_29     -546.51     287.41  -1.901 0.057784 .  
dteday_30     -248.41     294.83  -0.843 0.399873    
dteday_31          NA         NA      NA       NA    
season_1     -1553.04     204.33  -7.601 1.36e-13 ***
season_2      -688.98     240.68  -2.863 0.004368 ** 
season_3      -793.68     219.12  -3.622 0.000320 ***
season_4           NA         NA      NA       NA    
mnth_1         107.02     201.66   0.531 0.595844    
mnth_2         220.43     204.50   1.078 0.281586    
mnth_3         644.20     204.74   3.146 0.001747 ** 
mnth_4         498.70     270.96   1.840 0.066265 .  
mnth_5         661.96     289.75   2.285 0.022732 *  
mnth_6         374.92     297.88   1.259 0.208732    
mnth_7          81.62     317.82   0.257 0.797417    
mnth_8         291.33     304.08   0.958 0.338471    
mnth_9         918.64     248.42   3.698 0.000240 ***
mnth_10        487.70     186.25   2.618 0.009087 ** 
mnth_11        -72.56     182.23  -0.398 0.690645    
mnth_12            NA         NA      NA       NA    
weekday_6      -27.41     120.23  -0.228 0.819718    
weekday_0     -407.33     118.92  -3.425 0.000662 ***
weekday_1     -305.39     124.99  -2.443 0.014884 *  
weekday_2     -200.40     121.26  -1.653 0.099021 .  
weekday_3      -13.60     119.96  -0.113 0.909753    
weekday_4      -72.54     123.52  -0.587 0.557258    
weekday_5          NA         NA      NA       NA    
weathersit_2  1487.19     228.94   6.496 1.91e-10 ***
weathersit_1  1885.39     245.81   7.670 8.37e-14 ***
weathersit_3       NA         NA      NA       NA    
yr1           1999.93      66.74  29.968  < 2e-16 ***
holiday1      -671.00     199.23  -3.368 0.000813 ***
workingday1        NA         NA      NA       NA    
temp          4902.99     477.12  10.276  < 2e-16 ***
hum          -1741.58     352.24  -4.944 1.03e-06 ***
windspeed    -3439.22     470.85  -7.304 1.04e-12 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 773 on 526 degrees of freedom
Multiple R-squared:  0.8569,	Adjusted R-squared:  0.8414 
F-statistic: 55.28 on 57 and 526 DF,  p-value: < 2.2e-16
```

In Python

```python
# Train the model using the training sets
model = sm.OLS(trainlr.iloc[:,63], trainlr.iloc[:,0:63]).fit()
```

```python
#Lr model summary
model.summary()
```

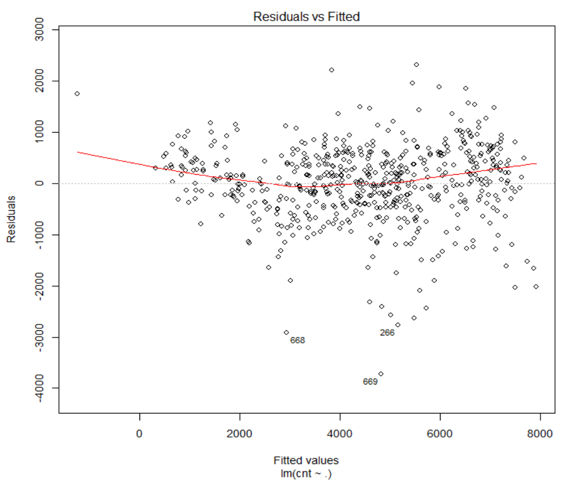
OLS Regression Results

| | | | |
|---|---|---|---|
| Dep. Variable: | cnt | R-squared: | 0.866 |
| Model: | OLS | Adj. R-squared: | 0.852 |
| Method: | Least Squares | F-statistic: | 59.68 |
| Date: | Thu, 04 Jul 2019 | Prob (F-statistic): | 1.16e-193 |
| Time: | 11:20:59 | Log-Likelihood: | -4660.8 |
| No. Observations: | 584 | AIC: | 9438. |
| Df Residuals: | 526 | BIC: | 9691. |
| Df Model: | 57 | | |
| Covariance Type: | nonrobust | | |

| | coef | std err | t | P>\|t\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| yr | 2003.6668 | 63.755 | 31.428 | 0.000 | 1878.422 | 2128.912 |
| holiday | 245.3282 | 163.229 | 1.503 | 0.133 | -75.332 | 565.988 |
| workingday | 653.9589 | 87.781 | 7.450 | 0.000 | 481.514 | 826.404 |
| temp | 4530.9358 | 453.079 | 10.000 | 0.000 | 3640.868 | 5421.003 |
| hum | -1700.7606 | 325.289 | -5.228 | 0.000 | -2339.785 | -1061.736 |
| windspeed | -3297.2871 | 451.755 | -7.299 | 0.000 | -4184.752 | -2409.822 |
| season_1 | -260.7711 | 144.934 | -1.799 | 0.073 | -545.491 | 23.949 |
| season_2 | 643.0092 | 150.047 | 4.285 | 0.000 | 348.244 | 937.774 |
| season_3 | 537.4436 | 160.208 | 3.355 | 0.001 | 222.717 | 852.170 |
| season_4 | 1388.2739 | 158.659 | 8.750 | 0.000 | 1076.590 | 1699.958 |
| dteday_01 | -128.9776 | 175.617 | -0.734 | 0.463 | -473.973 | 216.018 |
| dteday_02 | -86.5842 | 169.587 | -0.511 | 0.610 | -419.736 | 246.567 |
| dteday_03 | 163.6718 | 174.257 | 0.939 | 0.348 | -178.653 | 505.997 |
| dteday_04 | 238.3008 | 166.207 | 1.434 | 0.152 | -88.211 | 564.812 |
| dteday_05 | 102.0293 | 165.571 | 0.616 | 0.538 | -223.232 | 427.291 |
| dteday_06 | 162.3196 | 175.271 | 0.926 | 0.355 | -181.997 | 506.636 |
| dteday_07 | 153.5658 | 179.435 | 0.856 | 0.392 | -198.932 | 506.063 |
| dteday_08 | -66.7758 | 161.508 | -0.413 | 0.679 | -384.056 | 250.505 |
| dteday_09 | 87.0568 | 205.182 | 0.424 | 0.672 | -316.020 | 490.134 |
| dteday_30 | -301.9516 | 165.362 | -1.826 | 0.068 | -626.803 | 22.900 |
| dteday_31 | 195.0464 | 224.059 | 0.871 | 0.384 | -245.113 | 635.206 |
| weathersit_1 | 1547.8670 | 103.922 | 14.894 | 0.000 | 1343.713 | 1752.021 |
| weathersit_2 | 1119.8457 | 126.401 | 8.859 | 0.000 | 871.533 | 1368.158 |
| weathersit_3 | -359.7571 | 218.603 | -1.646 | 0.100 | -789.199 | 69.685 |
| mnth_1 | -143.1319 | 186.492 | -0.767 | 0.443 | -509.493 | 223.229 |
| mnth_2 | -75.8649 | 177.877 | -0.427 | 0.670 | -425.301 | 273.571 |
| mnth_3 | 495.3026 | 134.579 | 3.680 | 0.000 | 230.924 | 759.681 |
| mnth_4 | 318.6716 | 162.349 | 1.963 | 0.050 | -0.261 | 637.604 |
| mnth_5 | 520.4344 | 175.469 | 2.966 | 0.003 | 175.728 | 865.141 |
| mnth_6 | 293.7670 | 170.651 | 1.721 | 0.086 | -41.474 | 629.008 |
| mnth_7 | -67.8776 | 202.220 | -0.336 | 0.737 | -465.136 | 329.380 |
| mnth_8 | 270.6482 | 192.774 | 1.404 | 0.161 | -108.054 | 649.350 |
| mnth_9 | 918.4145 | 153.335 | 5.990 | 0.000 | 617.191 | 1219.638 |
| mnth_10 | 306.4019 | 172.831 | 1.773 | 0.077 | -33.123 | 645.926 |
| mnth_11 | -190.9326 | 180.377 | -1.059 | 0.290 | -545.281 | 163.416 |
| mnth_12 | -337.8776 | 154.894 | -2.181 | 0.030 | -642.164 | -33.591 |
| weekday_0 | 472.9576 | 117.077 | 4.040 | 0.000 | 242.962 | 702.953 |
| weekday_1 | 20.2915 | 81.106 | 0.250 | 0.803 | -139.040 | 179.623 |
| weekday_2 | 165.5724 | 86.802 | 1.907 | 0.057 | -4.949 | 336.094 |
| weekday_3 | 208.8218 | 85.360 | 2.446 | 0.015 | 41.134 | 376.509 |
| weekday_4 | 258.8638 | 82.402 | 3.141 | 0.002 | 96.987 | 420.741 |
| weekday_5 | 245.7376 | 82.036 | 2.995 | 0.003 | 84.578 | 406.897 |
| weekday_6 | 935.7109 | 118.397 | 7.903 | 0.000 | 703.123 | 1168.299 |

| | | | |
|---|---|---|---|
| Omnibus: | 65.518 | Durbin-Watson: | 1.893 |
| Prob(Omnibus): | 0.000 | Jarque-Bera (JB): | 141.399 |
| Skew: | -0.635 | Prob(JB): | 1.98e-31 |
| Kurtosis: | 5.049 | Cond. No. | 2.04e+16 |

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is corr
[2] The smallest eigenvalue is 3.52e-30. This might indicate that there are strong multicollinearity problems or that the design matrix is singular.



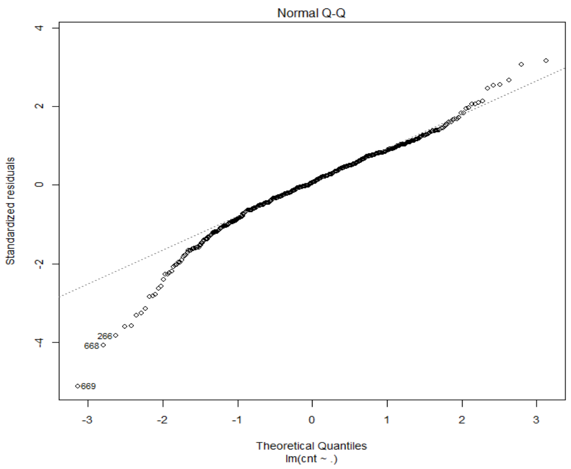



Fig6.3(a) Fig6.3(b)

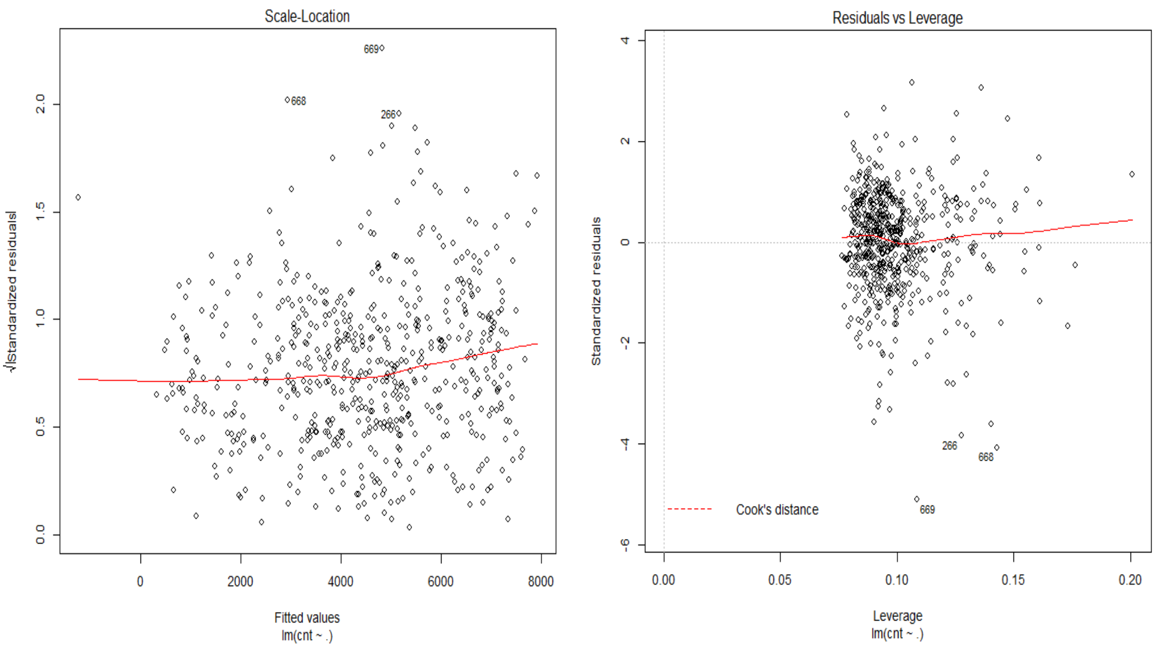



Fig6.3(c) Fig6.3(d)

A residual is the difference between the observed value of the dependent variable and the predicted value.

Fig 6.3(a): This plot tests the assumptions of whether the relationship between your variables is linear (i.e. linearity) and the whether there is equal variance along the regression line. In above figure red line represent the predicted values and small circle are actual values.

Fig 6.3(b): Normal Q-Q plot, It shows the normal distribution of dependent variable. From the plot we may assume that normality holds here.

Fig 6.3(c): There is some non-linearity here, but what we can also see is that the spread of magnitudes seems to be lowest in the fitted values close to 0, highest in the fitted values between 2500-6000, and medium around 7000. This suggests heteroskedasticity. ( variability)

Fig 6.3(d): A leverage point is defined as an observation that has a value that is far away from the mean. Plots helps to identify influential data points on the model. Any observation for which the Cook's distance larger, requires investigation.

# Chapter 7

# Conclusion

## 7.1 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 any of the following criteria:

1. Predictive Performance
2. Interpretability
3. Computational Efficiency

In our case of Bike Renting, the latter two, Interpretability and Computation Efficiency, do not hold much significance. Therefore we will use 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.

### 7.1.1 Mean Absolute Percentage Error (MAPE)

MAPE is one of the error measures used to calculate the predictive performance of the model. It measures the size of the error in percentage terms. We will apply this measure to our models that we have generated in the previous sections.

```
#defining MAPE function
def MAPE(y_true, y_pred):
 mape = np.mean(np.abs((y_true - y_pred) / y_true))*100
 return mape
```

In Python:

```
#MAPE
print('MAPE:')
#MAPE for decision tree regression
DT_M=MAPE(test.iloc[:,11], predictions_DT)
print('For DT:', round(DT_M, 2), '%.')
#MAPE for random forest regression
RF_M=MAPE(test.iloc[:,11],RF_Predictions)
print('For RF:', round(RF_M, 2), '%.')
#MAPE for linear regression
LR_M=MAPE(testlr.iloc[:,63], predictions_LR)
print('For LR:', round(LR_M, 2), '%.')
```

```
MAPE:
For DT: 30.7 %.
For RF: 17.85 %.
For LR: 17.66 %.
```

In R:

```
> #MAPE for Decision tree regression
> MAPE(test[,12], predictions_DT)
[1] 22.97977
> #MAPE for Random Forest Model
> MAPE(test[,12], predictions_RF)
[1] 22.11563
> #MAPE for Linear Regression
> MAPE(test[,12], predictions_LR)
[1] 20.51824
```

### 7.1.2 Root Mean Square Error(RMSE)

The root-mean-square deviation or root-mean-square error is a frequently used measure of the differences between values predicted by a model and the values observed.

```
#RMSE
def rmse(predictions, targets):
    return np.sqrt(((predictions - targets) ** 2).mean())
```

In Python:

```
#RMSE
print('RMSE:')
DT_rmse=rmse(test.iloc[:,11], predictions_DT)
print('For DT:', round(DT_rmse, 2))
RF_rmse=rmse(test.iloc[:,11],RF_Predictions)
print('For RF:', round(RF_rmse, 2))
LR_rmse=rmse(testlr.iloc[:,63], predictions_LR)
print('For LR:', round(LR_rmse, 2))
```

```
RMSE:
For DT: 1089.52
For RF: 662.16
For LR: 705.36
```

In R:

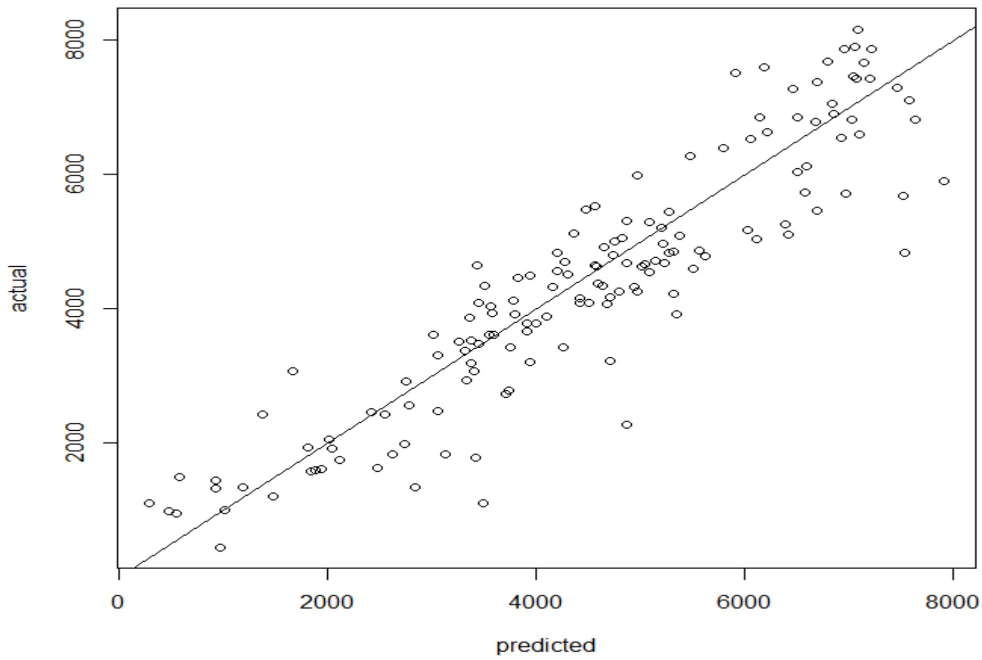
```
> #RMSE
> #RMSE for Decision tree regression
> RMSE(test[,12], predictions_DT)
[1] 1038.35
> #RMSE for Random Forest Model
> RMSE(test[,12], predictions_RF)
[1] 799.5338
> #RMSE for Linear Regression
> RMSE(test[,12], predictions_LR)
[1] 875.0181
```

## 7.2 Model Selection

| Model | Prog-Code | MAPE | ACCURACY | RMSE |
|---|---|---|---|---|
| Decision tree | PY | 30.7 | 69.3 | 1089.52 |
| | R | 22.97 | 77.03 | 1038.35 |
| Random forest | PY | 17.85 | 82.15 | 662.16 |
| | R | 22.11 | 77.89 | 799.54 |
| Linear Regression | PY | 17.66 | 82.34 | 705.36 |
| | R | 20.51 | 79.49 | 876.01 |

As we can see here we have a time-series dataset, Model having less RMSE and more ACCURACY will be our preferred model. Looking at the model Performance both Random Forest & Linear Regression Model has comparatively same Accuracy. but Linear Regression Model has bit more RMSE Value compared to Random Forest. So random forest model is selected with 82% accuracy in Python and with 78% accuracy in R.

As we can see, the model has more RMSE value. We don't have perfect predictions. Actually any model can't have perfect prediction for such type of dataset, where customer taking bike on rent would have some randomness. Assume for two days all situations are same except date and they are nearby dates. Then also there would not be same counting of bike renting even with same situation. So, there would be some randomness in dataset which is natural. So, our model is predicting quite well.




As we can see from above fig 7.1. Deviation for most of prediction from original value is low.

# Appendix A - R Code / Python code

Sample data(1-16 columns) (fig 1.1): R-code

```
View(head(br))
```

Fig 2.3 Numerical variable distribution (histograms): R code

```
par(mfrow=c(4,2))
par(mar = rep(2, 4))
hist(br$season)
hist(br$weathersit)
hist(br$hum)
hist(br$holiday)
hist(br$workingday)
hist(br$temp)
hist(br$atemp)
hist(br$windspeed)
hist(br$cnt)
```

2.4 and 2.5 Fig 2.4 probability density functions Distribution of variables: python code

```
#Check whether target variable is normal or not
sns.distplot(br['cnt']);
#descriptive statistics summary
br['cnt'].describe()
#Distribution independent numeric variables
#Check whether variable 'temp'is normal or not
sns.distplot(br['temp']);
#Check whether variable 'atemp'is normal or not
sns.distplot(br['atemp']);
#Check whether variable 'hum'is normal or not
sns.distplot(br['hum']);
#Check whether variable 'windspeed'is normal or not
sns.distplot(br['windspeed']);
#Check whether variable 'casual'is normal or not
sns.distplot(br['casual']);
#Check whether variable 'registered'is normal or not
sns.distplot(br['registered']);
```

Fig 2.6 Summary of the numerical variables.: Python code

```
cols = ['temp', 'atemp', 'hum', 'windspeed', 'casual', 'registered','cnt']
br[cols].describe()
```

Fig 2.7 Pair plot: R-Code

```
#check the relationship between all numeric variable using pair plot
library(GGally)
ggpairs(br[,c('atemp','temp','hum','windspeed','casual','registered','cnt')])
```

fig 2.8: R-code

```
#check the relationship between 'temp' and 'atemp' variable
ggplot(br, aes(x= temp,y=atemp)) +
  geom_point()+
  geom_smooth()
```

2.9(a) and 2.9(b) 2.9(c) 2.9(d): R-code

```
# Visualize categorical Variable 'yr' with target variable 'cnt'
ggplot(br, aes(x=as.factor(yr), y=cnt)) +
  stat_summary(fun.y="mean", geom="bar",fill="blue")+xlab("yr")

# Visualize categorical Variable 'mnth' with target variable 'cnt'
ggplot(br, aes(x=as.factor(mnth), y=cnt)) +
  stat_summary(fun.y="mean", geom="bar",fill="blue")+xlab("mnth")

# Visualize categorical Variable 'season' with target variable 'cnt'
ggplot(br, aes(x=as.factor(season), y=cnt)) +
  stat_summary(fun.y="mean", geom="bar",fill="blue")+xlab("season")

# Visualize categorical Variable 'weathersit' with target variable 'cnt'
ggplot(br, aes(x=as.factor(weathersit), y=cnt)) +
  stat_summary(fun.y="mean", geom="bar",fill="blue")+xlab("weathersit")
```

Fig 3.1 output shows there are no missing values

```
Calculating the null values in the dataframe
missing_value = pd.DataFrame(br.isnull().sum())
missing_value.reset_index()
missing_value = missing_value.rename(columns = {'index': 'Variables', 0: 'Missing_Val'})
missing_value
##There is no missing value in the data
```

Fig 4.1 Boxplot to visualize outliers

```
#.BoxPlots - Distribution and Outlier Check
numeric_index = sapply(br,is.numeric)     #selecting only numeric
numeric_data = br[,numeric_index]
cnames = colnames(numeric_data)

for (i in 1:length(cnames))
{
assign(paste0("gn",i),ggplot(data = br, aes_string(x = "cnt", y =cnames[i])) +
  stat_boxplot(geom = "errorbar", width = 0.5)+
        geom_boxplot(outlier.colour="red",fill="blue"))
}

gridExtra::grid.arrange(gn1,gn2,ncol=2)
gridExtra::grid.arrange(gn3,gn4,ncol=2)
gridExtra::grid.arrange(gn5,gn6,ncol=2)
```

Fig 5.1 and 5.2   Correlation Plot

```
## Correlation Plot
corrgram(br[,numeric_index], order = F,
 upper.panel=panel.pie, text.panel=panel.txt, main = "Correlation Plot")
cor(br[cnames])

library(RColorBrewer)
library(corrplot)
corrplot(cor(br[cnames]), method="number")
```

Fig 6.2 : Python code

```
# check Variable Importance
varimp=importance(RF_model)
# sort variable
sort_var <- names(sort(varimp[,1],decreasing =T))
# draw varimp plot
varImpPlot(RF_model,type = 1)
```

Fig6.3

```
#############Random Forest Model##########################
RF_model = randomForest(cnt ~ ., train, importance = TRUE, ntree = 200)
predictions_RF = predict(RF_model, test[,-12])
plot(RF_model)
```

Fig 7.1 Predicted vs actual cnt :R code

```
par(mfrow=c(1,1))
par(mar=c(5,4,4,4))
plot(predictions_RF,test[,12], xlab="predicted",ylab="actual")
abline(a=0,b=1)
```

## COMPLETE R-CODE:

```
########################BIKE#RENTING################################################################
######
#Remove all the objects stored
rm(list=ls())
#Set#check#current working director
setwd("F:/Ed_project")
getwd()
#Install required packages

#Load Libraries
x = c("ggplot2", "corrgram", "DMwR", "caret", "randomForest", "unbalanced", "C50", "dummies",
"e1071", "Information","MASS", "rpart", "gbm",
"ROSE", 'sampling', 'DataCombine',"data.table",'inTrees',"reshape","dplyr","plyr")

#install.packages(x)
lapply(x, require, character.only = TRUE)
rm(x)

#Read Bike Renting data in to R
br=read.csv("day.csv",header=T)

############## Analyze variables by visualize####################
#univariate distribution of numeric variables
#Numerical variable distribution
par(mfrow=c(4,2))
par(mar=rep(2,4))
hist(br$season)
hist(br$weathersit)
hist(br$hum)
hist(br$holiday)
hist(br$workingday)
hist(br$temp)
hist(br$atemp)
hist(br$windspeed)
hist(br$cnt)
par(mfrow=c(1,1))

# analyze the distribution of target variable 'cnt'
ggplot(br,aes(x=br$cnt,y=..density..,stat_bins=30))+
geom_histogram(fill= "DarkSeaGreen")+xlab("cnt")+
geom_density(colour="red")

# analyse the distrubution of independence variable 'temp'
ggplot(br,aes(x=br$temp,y=..density..,stat_bins=30))+
geom_histogram(fill= "DarkSeaGreen")+xlab("temp")+
geom_density(colour="red")

# analyse the distrubution of independence variable 'atemp'
ggplot(br,aes(x=br$atemp,y=..density..))+
geom_histogram(fill= "DarkSeaGreen")+xlab("atemp")+
geom_density(colour="red")

# analyse the distrubution of independence variable 'hum'
ggplot(br,aes(x=br$hum,y=..density..))+
geom_histogram(fill= "DarkSeaGreen")+xlab("hum")+
geom_density(colour="red")

# analyse the distrubution of independence variable 'windspeed'
ggplot(br,aes(x=br$windspeed,y=..density..,stat_bins=30))+
geom_histogram(fill= "DarkSeaGreen")+xlab("windspeed")+
geom_density(colour="red")
```

```
# analyse the distrubution of independence variable 'casual'
ggplot(br,aes(x=br$casual,y=..density..,stat_bins=30))+
geom_histogram(fill= "DarkSeaGreen")+xlab("casual")+
geom_density(colour="red")

##Bi-variate Analysis#
# Visualize categorical Variable 'yr' with target variable 'cnt'
ggplot(br, aes(x=as.factor(yr), y=cnt)) +
stat_summary(fun.y="mean", geom="bar",fill="blue")+xlab("yr")

# Visualize categorical Variable 'mnth' with target variable 'cnt'
ggplot(br, aes(x=as.factor(mnth), y=cnt)) +
stat_summary(fun.y="mean", geom="bar",fill="blue")+xlab("mnth")

# Visualize categorical Variable 'season' with target variable 'cnt'
ggplot(br, aes(x=as.factor(season), y=cnt)) +
stat_summary(fun.y="mean", geom="bar",fill="blue")+xlab("season")

# Visualize categorical Variable 'weathersit' with target variable 'cnt'
ggplot(br, aes(x=as.factor(weathersit), y=cnt)) +
stat_summary(fun.y="mean", geom="bar",fill="blue")+xlab("weathersit")

# Visualize categorical Variable 'holiday'
ggplot(br) +
geom_bar(aes(x=holiday),fill="blue")
# it is showing that almost all the cycle rentals are happening on holidays

# Visualize categorical Variable 'weekday'
ggplot(br) +
geom_bar(aes(x=weekday),fill="blue")
# it is showing counts are all most same on all weekdays

# Visualize categorical Variable 'weathersit'
ggplot(br) +
geom_bar(aes(x=weathersit),fill="blue")
# count is more when whether is " Clear, Few clouds, Partly cloudy, Partly cloudy"

#check the relationship between 'temp' and 'atemp' variable
ggplot(br, aes(x= temp,y=atemp)) +
geom_point()+
geom_smooth()
#This graph is saying that very strong relationship between 'temp' and 'atemp' which leads to collinearity

#check the relationship between 'temp' and 'hum' variable
ggplot(br, aes(x= temp,y=hum)) +
geom_point()+
geom_smooth()
# here it is showing Humidity is increses till temparature is 0.7 and it is decreasing gradually

#check the relationship between 'temp' and 'windspeed' variable
ggplot(br, aes(x= temp,y=windspeed)) +
geom_point()+
geom_smooth()
# it is showing that very less negative correlation between temp and windspeed

#check the relationship between all numeric variable using pair plot
library(GGally)
ggpairs(br[,c('atemp','temp','hum','windspeed','casual','registered','cnt')])
# that above plot stating that less nagative relationship between'cnt'-'hum' and cnt-windspeed
# and there is strong positive relationship between temp and atemp

#Relationship between target variables
ggplot(br, aes(x= casual,y=cnt)) +
geom_point()+
geom_smooth()
```

```
ggplot(br, aes(x= registered,y=cnt)) +
geom_point()+
geom_smooth()

ggplot(br, aes(x= casual+registered,y=cnt)) +
geom_point()+
geom_smooth()

##############################DATA#EXPLORATION##############################################
str(br)

##Variable Identification
br$instant=NULL #Removing thr Record Index variable
br$dteday=format(as.Date(br$dteday,format="%Y-%m-%d"), "%d")
br$dteday=as.factor(br$dteday)
br$season=as.factor(br$season)
br$yr=as.factor(br$yr)
br$mnth=as.factor(br$mnth)
br$holiday=as.factor(br$holiday)
br$weekday=as.factor(br$weekday)
br$workingday=as.factor(br$workingday)
br$weathersit=as.factor(br$weathersit)
#br=subset(br,select = -c(casual,registered))

###################Missing#Value#Analysis##################################
missing_value=data.frame(apply(br,2,function(x){sum(is.na(x))}))
missing_value$column=row.names(missing_value)
names(missing_value)[1]="missing Val"
row.names(missing_value)=NULL
missing_value=missing_value[,c(2,1)]
print(missing_value) #There are no missing values

##################Outlier#Analysis########################################
#Already all numeric variable are in normalize form so , no need to analysing Outliers
#here the six numerics variables are present out of six four variables are in normalize form.
# temp,atem,hum,windspread are in normalize form no need for outlier treatment.
# BoxPlots - Distribution and Outlier Check
numeric_index = sapply(br,is.numeric) #selecting only numeric
numeric_data = br[,numeric_index]
cnames = colnames(numeric_data)

for (i in 1:length(cnames))
{a
ssign(paste0("gn",i),ggplot(data = br, aes_string(x = "cnt", y =cnames[i])) +
stat_boxplot(geom = "errorbar", width = 0.5)+
geom_boxplot(outlier.colour="red",fill="blue"))
}
gridExtra::grid.arrange(gn1,gn2,ncol=2)
gridExtra::grid.arrange(gn3,gn4,ncol=2)
gridExtra::grid.arrange(gn5,gn6,ncol=2)
# detect outliers in 'hum','windspeed' and 'casual' variables
#no need for outlier treatment

#################Feature#Selection#or#dimension#reduction##########
## Correlation Plot
corrgram(br[,numeric_index], order = F,
upper.panel=panel.pie, text.panel=panel.txt, main = "Correlation Plot")
cor(br[cnames])
library(RColorBrewer)
library(corrplot)
corrplot(cor(br[cnames]), method="number")
## Dimension Reduction####
br = subset(br,select = -c(atemp)) #there is high coleration b/w 'temp' and 'atemp'.
br=subset(br,select = -c(casual,registered)) #collinearity with 'cnt'
```

```
##############################MODEL DEVELOPMENT##########################################
rmExcept("br")
set.seed(5426)
train_index = sample(1:nrow(br), 0.8 * nrow(br))
train = br[train_index,]
test = br[-train_index,]

###########Decision tree regression #################
fit = rpart(cnt ~ ., data = train, method = "anova")
predictions_DT = predict(fit, test[,-12])
print(fit)
par(cex= .9)
plot(fit)
text(fit)

#############Random Forest Model##########################
RF_model = randomForest(cnt ~ ., train, importance = TRUE, ntree = 200)
predictions_RF = predict(RF_model, test[,-12])
plot(RF_model)

#Extract rules metrics
treeList = RF2List(RF_model)
exec = extractRules(treeList, train[,-12])
readableRules = presentRules(exec, colnames(train))
readableRules[1:2,]
ruleMetric = getRuleMetric(exec, train[,-12], train$cnt)
ruleMetric[1:3,]

##check Variable Importance
varimp=importance(RF_model)
varimp
# sort variable
sort_var <- names(sort(varimp[,1],decreasing =T))
# draw varimp plot
varImpPlot(RF_model,type = 1)

################Linear Regression###################################
#converting multilevel categorical variable into binary dummy variable
cnames= c("dteday","season","mnth","weekday","weathersit")
data_lr=br[,cnames]
cnt=data.frame(br$cnt)
names(cnt)[1]="cnt"
library(fastDummies)
data_lr <- fastDummies::dummy_cols(data_lr)
data_lr= subset(data_lr,select = -c(dteday,season,mnth,weekday,weathersit))
d3 = cbind(data_lr,br)
d3= subset(d3,select = -c(dteday,season,mnth,weekday,weathersit,cnt))
data_lr=cbind(d3,cnt)

##dividind data into test and train
#train_index = sample(1:nrow(data_lr), 0.8 * nrow(data_lr))
train_lr = data_lr[train_index,]
test_lr = data_lr[-train_index,]

##Linear regression model making
lm_model = lm(cnt ~., data = train_lr)
predictions_LR = predict(lm_model,test_lr[,-64])

summary(lm_model)
#plot(lm_model)

#################Model Evaluation###############
#defining MAPE function
MAPE = function(y, yhat){
mean(abs((y - yhat)*100/y))
}
```

```
#MAPE for Decision tree regression
MAPE(test[,12], predictions_DT)
#MAPE for Random Forest Model
MAPE(test[,12], predictions_RF)
#MAPE for Linear Regression
MAPE(test[,12], predictions_LR)

#RMSE
#RMSE for Decision tree regression
RMSE(test[,12], predictions_DT)
#RMSE for Random Forest Model
RMSE(test[,12], predictions_RF)
#RMSE for Linear Regression
RMSE(test[,12], predictions_LR)

##########extacting predicted values output of all models######################
results=test
results$DT_predic_cnt=predictions_DT
results$RF_predic_cnt=predictions_RF
results$LR_predic_cnt=predictions_LR
write.csv(results, file = 'output_R .csv', row.names = FALSE, quote=FALSE)
#########################Thank#You#for#reading##################################
```

# Complete Python Code:

# BIKE RENTING

## Univariate Analysis

```
#Load libraries
import os
import datetime
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from random import randrange, uniform
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor
import statsmodels.api as sm
from sklearn.ensemble import RandomForestRegressor

#Changing the directoy
os.chdir("F:\Ed_project_py")
os.getcwd()

#Loading the csv
br = pd.read_csv("day.csv")
br.info()
br.describe()

#Check whether target variable is normal or not
sns.distplot(br['cnt']);
#Distribution of independent numeric variables
#Check whether variable 'temp'is normal or not
sns.distplot(br['temp']);
#Check whether variable 'atemp'is normal or not
sns.distplot(br['atemp']);
#Check whether variable 'hum'is normal or not
sns.distplot(br['hum']);
#Check whether variable 'windspeed'is normal or not
sns.distplot(br['windspeed']);
#Check whether variable 'casual'is normal or not
sns.distplot(br['casual']);
#Check whether variable 'registered'is normal or not
sns.distplot(br['registered']);

# it is clearly showing that chances of outliers present in 'casual' varible
#Numeric variable summary
cols = ['temp', 'atemp', 'hum', 'windspeed', 'casual', 'registered','cnt']
br[cols].describe()

############################ Bivariate Relationship ############################
#relation between Numerical Variable 'temp' and target variable 'cnt'
sns.regplot(x="temp", y="cnt", data=br,marker='o', color='g')
# It is showing there is good relation between 'temp' and 'cnt'

#relation between Numerical Variable 'atemp' and target variable 'cnt'
sns.regplot(x="atemp", y="cnt", data=br,marker='*', color='r')
# It is showing there is good relation between 'atemp' and 'cnt'
```

```
#relation between Numerical Variable 'hum' and target variable 'cnt'
sns.regplot(x="hum", y="cnt", data=br,marker='o', color='blue')

#relation between Numerical Variable 'windspeed' and target variable'cnt'
sns.regplot(x="windspeed", y="cnt", data=br,marker='o', color='black')
# It is showing there is negative relation between 'windspeed' and'cnt'

#relation between variables 'registered' and 'cnt'
sns.regplot(x="registered", y="cnt", data=br)

#relation between variables 'casual' and 'cnt'
sns.regplot(x="casual", y="cnt", data=br)

#box plot 'weekdays' with 'CNT'
var_weekdays = 'weekday'
data = pd.concat([br['cnt'], br[var_weekdays]], axis=1)
f, ax = plt.subplots(figsize=(8, 6))
fig = sns.boxplot(x=var_weekdays, y="cnt", data=data)
fig.axis(ymin=0, ymax=9000);
#below Boxplot is saying that median high on holidays when compare to weekdays

#box plot 'holiday'with 'CNT'
var_holiday = 'holiday'
data = pd.concat([br['cnt'], br[var_holiday]], axis=1)
f, ax = plt.subplots(figsize=(5, 6))
fig = sns.boxplot(x=var_holiday, y="cnt", data=data)
fig.axis(ymin=0, ymax=9000);
#below Boxplot is saying that median high on holidays when compare to weekdays

# check relationship with scatter plots
sns.set()
cols = ['temp', 'atemp', 'hum', 'windspeed', 'casual', 'registered','cnt']
sns.pairplot(br[cols], size = 2.5,kind="reg")
plt.show();
#As per scatter plots and above Correlation graph there is strong relation between Independ
# so dropping one of two variables for feature selection is necessary

print("Skewness: %f" % br['cnt'].skew())
print("Kurtosis: %f" % br['cnt'].kurt())
#Here Skewness is very less so target variable is normal distribution
```

# Exploratory Data Analysis[EDA]

```
#Exploratory Data Analysis[EDA]
br = br.drop('instant', axis=1)
br['season']= br['season'].astype('category')
br['yr']=br['yr'].astype('int')
br['mnth']=br['mnth'].astype('category')
br['holiday']=br['holiday'].astype('int')
br['weekday']=br['weekday'].astype('category')
br['workingday']=br['workingday'].astype('int')
br['weathersit']=br['weathersit'].astype('category')
d1=br['dteday'].copy()
for i in range (0,d1.shape[0]):
d1[i]=datetime.datetime.strptime(d1[i], '%Y-%m-%d').strftime('%d')
br['dteday']=d1
```

```
br['dteday']=br['dteday'].astype('category')
#br = br.drop(['instant','casual', 'registered'], axis=1)

br.head()
br.info()
```

# Missing Values

```
#Calculating the null values in the dataframe
missing_value = pd.DataFrame(br.isnull().sum())
missing_value.reset_index()
missing_value = missing_value.rename(columns = {'index': 'Variables', 0:
'Missing_Val'})

missing_value
```

# Outlier Analysis

```
##There is no missing value in the data
# Already all numeric variable are in normalize form.
# temp,atem,hum,windspread are in normalize form no need for outlier treatment
backup=br.copy()

#saving numeric values#
cnames=["temp","atemp","hum","windspeed","casual","registered","cnt"]

#ploting boxplotto visualize outliers#
plt.subplot(2,2,1)
sns.boxplot(x=br["temp"],orient ='h')
plt.subplot(2,2,2)
sns.boxplot(x=br["atemp"],orient ='h')
plt.subplot(2,2,3)
sns.boxplot(x=br["hum"],orient ='h')
plt.subplot(2,2,4)
sns.boxplot(x=br["windspeed"],orient ='h')
plt.subplot(2,2,1)
sns.boxplot(x=br["casual"],orient ='h')
plt.subplot(2,2,2)
sns.boxplot(x=br["registered"],orient ='h')
#"hum","windspeed","casual" shows outliers, but data is normalized, no need for outlier treatment
df_corr = br[cnames]

#Set the width and hieght of the plot
f, ax = plt.subplots(figsize=(7, 5))
#Generate correlation matrix
corr = df_corr.corr()
#Plot using seaborn library
sns.heatmap(corr, mask=np.zeros_like(corr, dtype=np.bool), 10,
as_cmap=True),square=True, ax=ax)cmap=sns.diverging_palette(220, 10,
as_cmap=True),square=True, ax=ax)
corr
br['atemp'].corr(br['cnt'])
#droping corelated variable
br = br.drop(['atemp','casual','registered'], axis=1)
br.shape
```

# Model Development

```
#dividing data into train and test
train, test = train_test_split(br, test_size=0.2)

################Decision tree#########################
#Decision tree for regression
fit_DT =
DecisionTreeRegressor(max_depth=2).fit(train.iloc[:,0:11],train.iloc[:,11])
predictions_DT = fit_DT.predict(test.iloc[:,0:11])

#################random forest################################
#random forest
RFmodel = RandomForestRegressor(n_estimators = 200).fit(train.iloc[:,0:11],
train.iloc[:,11]
RF_Predictions = RFmodel.predict(test.iloc[:,0:11])

################linear regression###############################

#linear regression
#creating dummy variable
data_lr=br.copy()
cat_names = ["season", "dteday", "weathersit", "mnth","weekday"]
for i in cat_names:
temp = pd.get_dummies(data_lr[i], prefix = i)
data_lr = data_lr.join(temp)
fields_to_drop = ['dteday', 'season', 'weathersit', 'weekday', 'mnth','cnt']
data_lr = data_lr.drop(fields_to_drop, axis=1)
data_lr=data_lr.join(br['cnt'])

#Divide data into train and test
#trainlr, testlr = train_test_split(data_lr, test_size=0.2)
#train data
index_train=train.index.values
trainlr=data_lr.iloc[index_train,:]
#test data
index_test=test.index.values
testlr=data_lr.iloc[index_test,:]

# Train the model using the training sets
model = sm.OLS(trainlr.iloc[:,63], trainlr.iloc[:,0:63]).fit()


# make the predictions by the model
predictions_LR = model.predict(testlr.iloc[:,0:63])

#lr model summary
model.summary()

######Model Evaluation##############
#defining MAPE function
def MAPE(y_true, y_pred):
mape = np.mean(np.abs((y_true - y_pred) / y_true))*100
return mape
```

```
#MAPE
print('MAPE:')
#MAPE for decision tree regression
DT_M=MAPE(test.iloc[:,11], predictions_DT)
print('For DT:', round(DT_M, 2), '%.')
#MAPE for random forest regression
RF_M=MAPE(test.iloc[:,11],RF_Predictions)
print('For RF:', round(RF_M, 2), '%.')
#MAPE for linear regression
LR_M=MAPE(testlr.iloc[:,63], predictions_LR)
print('For LR:', round(LR_M, 2), '%.')

# Calculate and display Accuracy
print('Accuracy:')
print('For DT:', round(100-DT_M, 2), '%.')
print('For RF:', round(100-RF_M, 2), '%.')
print('For LR:', round(100-LR_M, 2), '%.')

#Defining RMSE function
def rmse(predictions, targets):
return np.sqrt(((predictions - targets) ** 2).mean())

#RMSE
print('RMSE:')
DT_rmse=rmse(test.iloc[:,11], predictions_DT)
print('For DT:', round(DT_rmse, 2))
RF_rmse=rmse(test.iloc[:,11],RF_Predictions)
print('For RF:', round(RF_rmse, 2))
LR_rmse=rmse(testlr.iloc[:,63], predictions_LR)
print('For LR:', round(LR_rmse, 2))

result=pd.DataFrame(test.iloc[:,0:12])
result['DT_pred_cnt'] = (predictions_DT)
result['RF_pred_cnt'] = (RF_Predictions)
result['LR_pred_cnt'] = (predictions_LR)
result.head()


result.to_csv("output_python.csv",index=False)
```

**Regards**

**SWAROOP H**