Contents

1. Introduction

1.1 Problem Statement

Bike rentals is popular concept that has gained a lot of attention in the past few years. The process has become so simplified through the task of automation that everyone can easily book a rental bike within a few clicks.

The number of bikes issued over the various days in a year is influenced by the environmental and seasonal conditions. Analyzing that in which seasons and under which environmental conditions, the count of bikes issued is increased or decreased is a tedious process if left to human beings alone but using statistical models and data mining techniques we can make this process very easy and efficient thus the goal of our project is to automate this task.

2. Data

We will make our progress by predicting the count of bikes issued each day based on the environmental and seasonal factors. Given below is the snapshot of a sample of the dataset that we would be using to predict the bike rental count.

instant	dteday	season	yr	mnth	holiday	weekday	workingda	weathersit	temp	atemp	hum	windspeed	casual	registered
1	########	1	0	1	0	6	0	2	0.344167	0.363625	0.805833	0.160446	331	654
2	########	1	0	1	0	0	0	2	0.363478	0.353739	0.696087	0.248539	131	670
3	########	1	0	1	0	1	1	1	0.196364	0.189405	0.437273	0.248309	120	1229
4	########	1	0	1	0	2	1	1	0.2	0.212122	0.590435	0.160296	108	1454
5	########	1	0	1	0	3	1	1	0.226957	0.22927	0.436957	0.1869	82	1518
6	########	1	0	1	0	4	1	1	0.204348	0.233209	0.518261	0.089565	88	1518
7	########	1	0	1	0	5	1	2	0.196522	0.208839	0.498696	0.168726	148	1362
8	########	1	0	1	0	6	0	2	0.165	0.162254	0.535833	0.266804	68	891
9	########	1	0	1	0	0	0	1	0.138333	0.116175	0.434167	0.36195	54	768
10	########	1	0	1	0	1	1	1	0.150833	0.150888	0.482917	0.223267	41	1280

2.1.2 Dimensions of Dataset

```
> #Checking dimensions of our dataset
> dim(data)
[1] 731  16
```

2.1.3 Features of the Dataset

```
> names(data)
 [1] "instant"    "dteday"     "season"     "yr"         "mnth"       "holiday"    "weekday"    "workingday"
 [9] "weathersit" "temp"       "atemp"      "hum"        "windspeed"  "casual"     "registered" "cnt"
```

As we can see from the features above, we have a total of 15 predictors that can help us to predict the count ("cnt") of the bike rentals. They are listed below:

S.No	Predictor
1.	Instant
2.	Dteday
3.	season
4.	Yr
5.	Mnth
6.	Holiday
7.	Weekday
8.	Workingday
9.	Weathersit
10.	Temp
11.	Atemp
12.	hum
13.	windspeed
14.	casual
15.	registered

2.1.3 Structure & Summary of dataset

Lets take a look at the data types of the different attributes and also check an overall summary of them.

Fig 1.2 Structure of dataset

```
> str(data)
'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 8 9 10 ...
 $ 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 1.2 Summary of the variables of dataset

```
> summary(data)
    instant             dteday        season            yr              mnth          holiday           weekday     
 Min.   :  1.0   2011-01-01:  1   Min.   :1.000   Min.   :0.0000   Min.   : 1.00   Min.   :0.00000   Min.   :0.000  
 1st Qu.:183.5   2011-01-02:  1   1st Qu.:2.000   1st Qu.:0.0000   1st Qu.: 4.00   1st Qu.:0.00000   1st Qu.:1.000  
 Median :366.0   2011-01-03:  1   Median :3.000   Median :1.0000   Median : 7.00   Median :0.00000   Median :3.000  
 Mean   :366.0   2011-01-04:  1   Mean   :2.497   Mean   :0.5007   Mean   : 6.52   Mean   :0.02873   Mean   :2.997  
 3rd Qu.:548.5   2011-01-05:  1   3rd Qu.:3.000   3rd Qu.:1.0000   3rd Qu.:10.00   3rd Qu.:0.00000   3rd Qu.:5.000  
 Max.   :731.0   2011-01-06:  1   Max.   :4.000   Max.   :1.0000   Max.   :12.00   Max.   :1.00000   Max.   :6.000  
                 (Other)   :725                                                                                     
   workingday      weathersit         temp             atemp              hum           windspeed      
 Min.   :0.000   Min.   :1.000   Min.   :0.05913   Min.   :0.07907   Min.   :0.0000   Min.   :0.02239  
 1st Qu.:0.000   1st Qu.:1.000   1st Qu.:0.33708   1st Qu.:0.33784   1st Qu.:0.5200   1st Qu.:0.13495  
 Median :1.000   Median :1.000   Median :0.49833   Median :0.48673   Median :0.6267   Median :0.18097  
 Mean   :0.684   Mean   :1.395   Mean   :0.49538   Mean   :0.47435   Mean   :0.6279   Mean   :0.19049  
 3rd Qu.:1.000   3rd Qu.:2.000   3rd Qu.:0.65542   3rd Qu.:0.60860   3rd Qu.:0.7302   3rd Qu.:0.23321  
 Max.   :1.000   Max.   :3.000   Max.   :0.86167   Max.   :0.84090   Max.   :0.9725   Max.   :0.50746  

     casual         registered        cnt      
 Min.   :   2.0   Min.   :  20   Min.   :  22  
 1st Qu.: 315.5   1st Qu.:2497   1st Qu.:3152  
 Median : 713.0   Median :3662   Median :4548  
 Mean   : 848.2   Mean   :3656   Mean   :4504  
 3rd Qu.:1096.0   3rd Qu.:4776   3rd Qu.:5956  
 Max.   :3410.0   Max.   :6946   Max.   :8714  
```

2.1.3 Conversion of the Normalized attributes

We need to convert the normalized features like temp, actual temp, humidity and windspeed into raw denormalized values since the normalized values were very low and factorized the categorical attributes. We would create a function to denormalize the values and add the updated values as separate columns in our dataset.

```
denorm_temp <- function(x) x*(47) - 8
denorm_atemp <- function(x) x*(66) - 16
denorm_hum <- function(x) x * 100
denorm_wind <- function(x) x * 67

bikeData$denorm_temp = unlist(lapply(bikeData$temp, denorm_temp))
bikeData$denorm_atemp = unlist(lapply(bikeData$atemp, denorm_atemp))
bikeData$denorm_hum = unlist(lapply(bikeData$hum, denorm_hum))
bikeData$denorm_wind = unlist(lapply(bikeData$windspeed, denorm_wind))
```

3. Methodology

3.1 Exploratory Data Analysis

3.1.1 Outlier Analysis

Using boxplots, we were able to analyze the effect of various variables on the count of bikes. The plots are shown below:

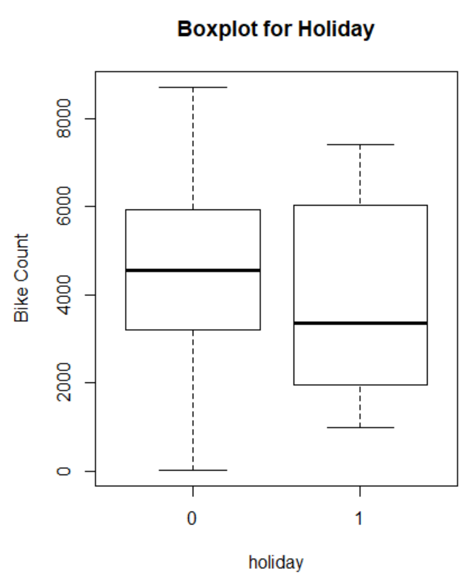


Fig 2.1 Count v/s Holiday

The max count of bikes issued when there was no Holiday is greater than 8000 as compared to a Holiday where the max no. of bikes issued was almost 6000. The median of bikes issues for no holiday is higher.

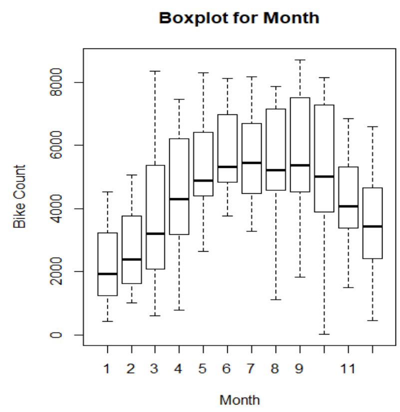


Fig 2.2 Month v/s Count

It can be seen that the highest median value is for the month of July whereas the max count of bikes issued was in September.

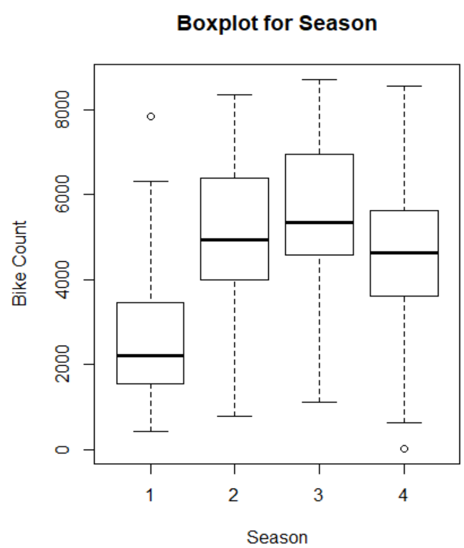


Fig 2.3 Season v/s Count

The Highest median value is for Season 3 i.e. fall season and the max. count of bikes issued in any season is close to 7000 and this value also lies in the fall season.

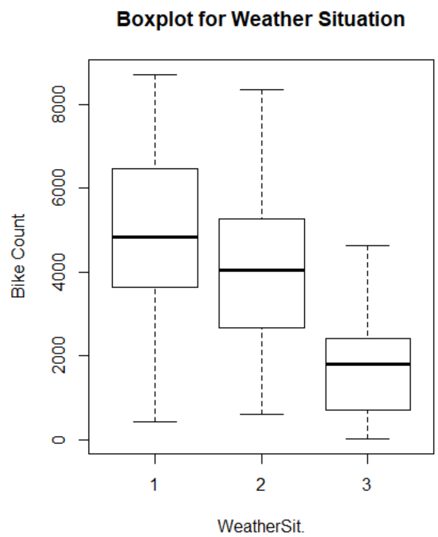


Fig 2.4 Weather Situation v/s Count

The median value for bikes issued during adverse weather conditions(3) remains fairly low whereas the median value of bikes issued during clear weather(1) is the highest.

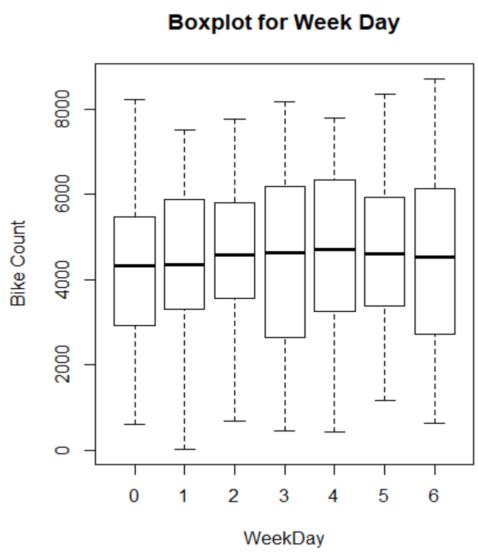


Fig 2.5 Count v/s Week Day

It can be observed tht during weekdays the median value for count bikes issued remains almost the same.

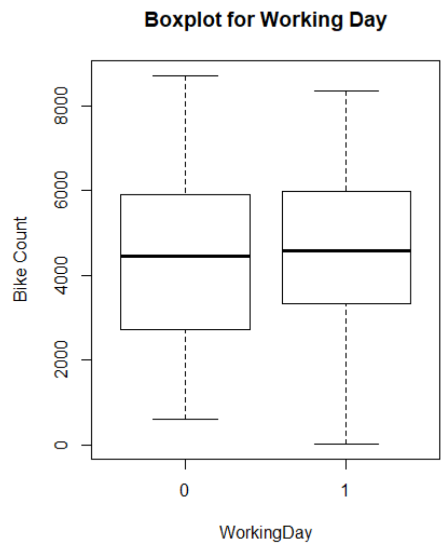


Fig 2.6 Count v/s Working Day

This boxplot tells us that on normal working-days (1) the minimum number of bikes issued lies around 2700 whereas on a weekend or holiday (0) the minimum number of bikes issued lies around 3400.

It can be noted from the plots above that the bulk of the values of the dependent variable lie in a particular range for each of the independent variables. Only a few outliers exist.

3.1.2 Distribution Analysis with Density Plots(Categorical Variables)

We will also plot density plots of all the categorical variables for a better representation of the categorical predictors.

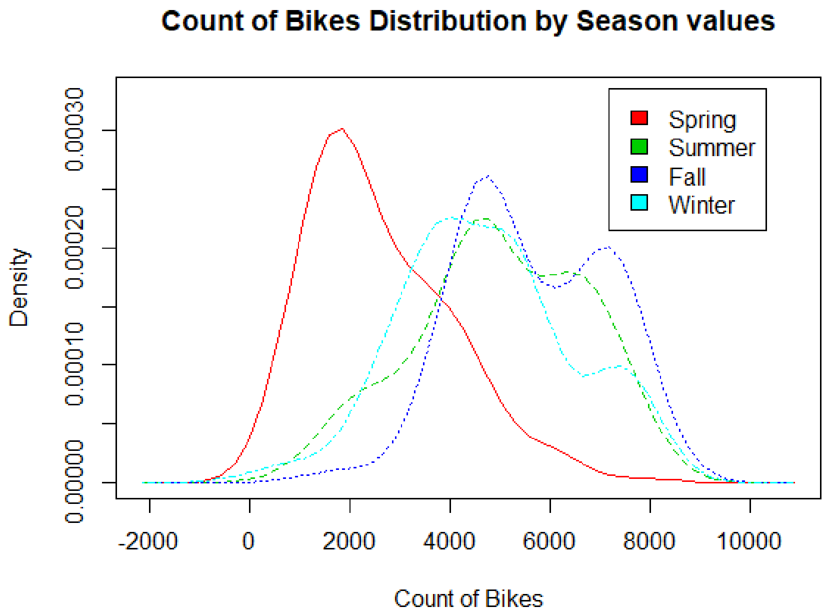


Fig 2.7 Count v/s Season

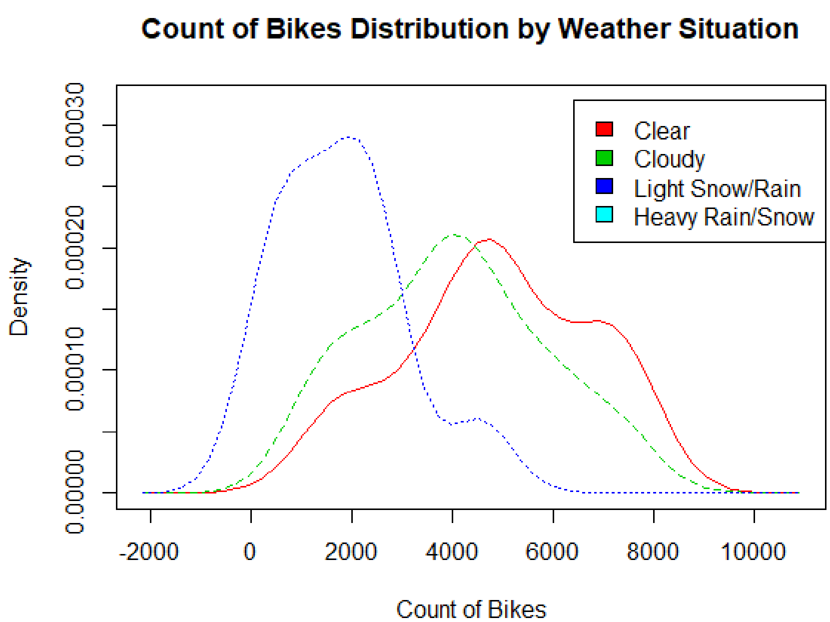


Fig 2.8 Count v/s Weather Situation

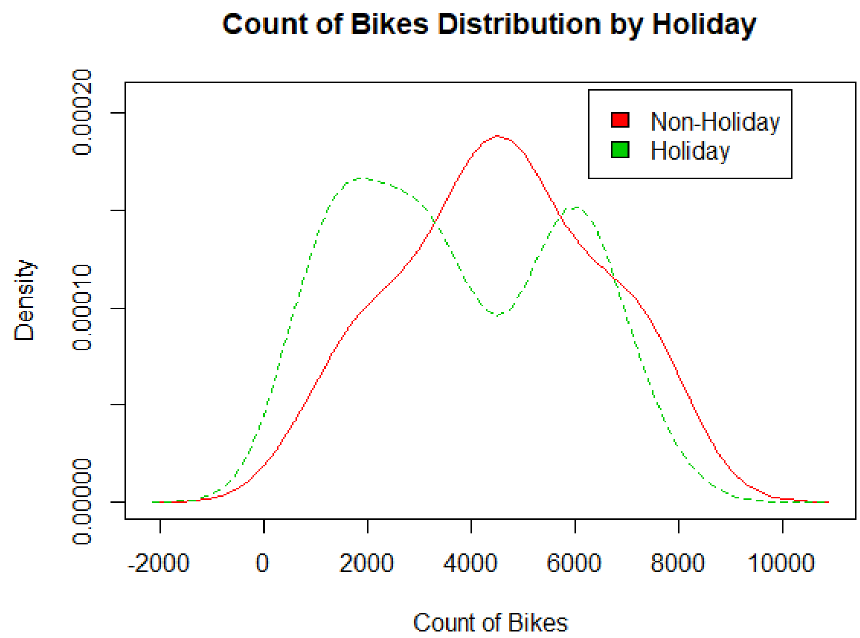


Fig 2.6 Count v/s Holiday

From above density plots, we can conclude there is no extreme uneven distribution in our categorical variables. They are by far, evenly distributed.

3.1.3 Distribution of Numerical variables

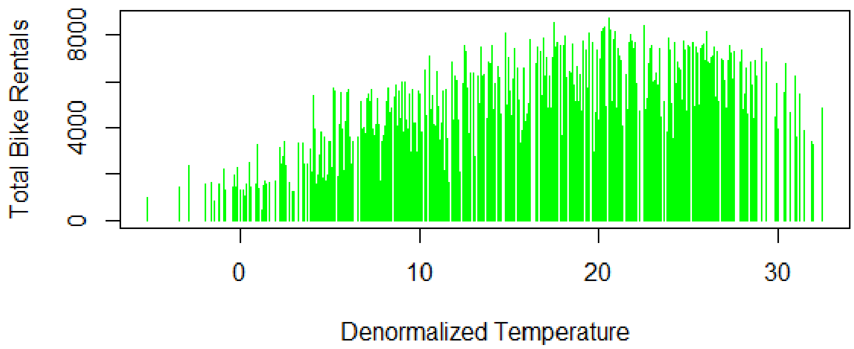


Fig 2.7 Count v/s Temperature

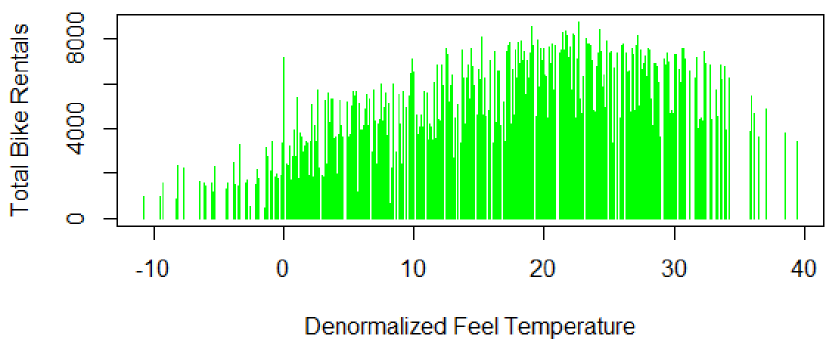


Fig 2.8 Count v/s Feel Temperature

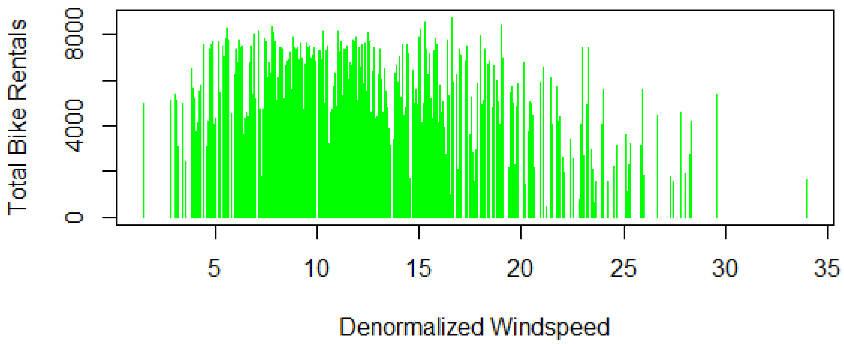


Fig 2.9 Count v/s Wind speed

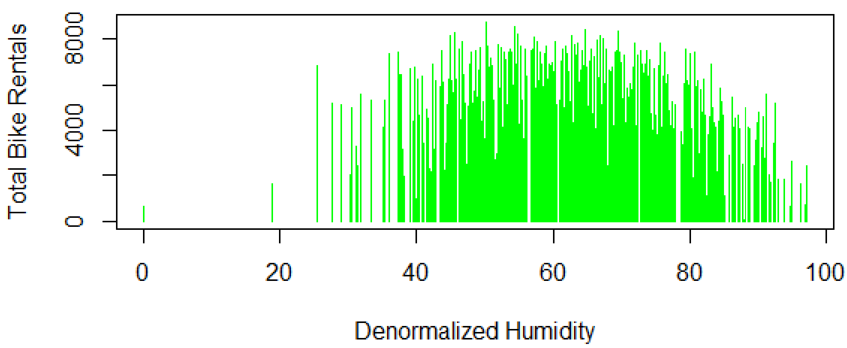


Fig 2.10 Count v/s Humidity

We can see from the above plots that the numerical values are distributed quite evenly.

3.2 Feature Selection

Since we are dealing with prediction of bike rental count which will have a numerical value, our task is to build suitable regression models depending on various factors such as humidity, temperature, season, time of the day(hour), just to name a few,

It can be noted at a glance that there are certain features that can be discarded immediately such as the date/time because other features exist that contribute to the count of bikes much better than a single date such as month, working day, holiday, year and weekday therefore we will discard this variable in our experiment.

From this point forward we will denote the count of bikes as the dependent variable and all the attributes as independent variables.

Now it's clear that all independent variables have a consistent effect on the dependent variable, we will take a look at how much the dependent variable is affected with a change in each of the independent variables. We use the random forests approach for feature selection in this case. Given below, are the % increase in MSE values with respect to each independent variable.

```
> predictor_importance <- randomForest(cnt ~ season + yr + mnth + holiday + weekday + workingday + temp + atemp + hum + w
indspeed, data = training_set,
+                                       ntree = 100, keep.forest = FALSE, importance = TRUE)
> importance(predictor_importance, type = 1)
             %IncMSE
season     10.400770
yr         45.957396
mnth        9.388992
holiday     2.796891
weekday     5.166776
workingday  3.809533
temp       11.787669
atemp      13.087685
hum        15.462086
windspeed   6.154748
```

Fig 3.1 Predictor Importance

It can be seen at a glance that 'year' contributes the most to the count of bikes issued and a quick look at the dataset explains this reason because the count of bikes issued in 2012(Total Count = 2049576). are far more than those issued in 2011(Total Count = 1243103). This is normal and can be assumed that the company made growth and sales increased after the initial year. Furthermore, the variables 'humidity' and 'temperature' contribute a significant amount to the dependent variable as well.

Another way to do feature selection is to look at the relationship between the variables themselves. This is known as correlation. We used the 'dplyr' library in R to find the correlation between various variables. Our dataset consists of attributes whose correlation can be determined at a glance, for example, the variables 'month' and 'season' are closely related because we already know that a season exists for a specific set of months. The attributes ‘temp’ and ‘atemp’ are highly correlated because they are almost similar values, ‘temp’ denotes the normalized temperature in Celsius and ‘atemp’ denotes the normalized feeling temperature in Celsius. All the correlations can be observed from the following figure:

```
> truncated = select(bikeData,season,mnth,weathersit,temp,atemp,hum,windspeed,cnt)
> symnum(cor(truncated))
           s m wt t a h wn c
season     1
mnth       + 1
weathersit     1
temp       .      1
atemp      .      B 1
hum            .      1
windspeed               1
cnt        .      , ,      1
attr(,"legend")
[1] 0 ‘ ’ 0.3 ‘.’ 0.6 ‘,’ 0.8 ‘+’ 0.9 ‘*’ 0.95 ‘B’ 1
```

Fig 3.3 Correlation between Variables

The correlation between month and season can be seen clearly above denoted by the legend ‘+’. Similarly the correlation between temp and atemp is denoted by ‘B’.

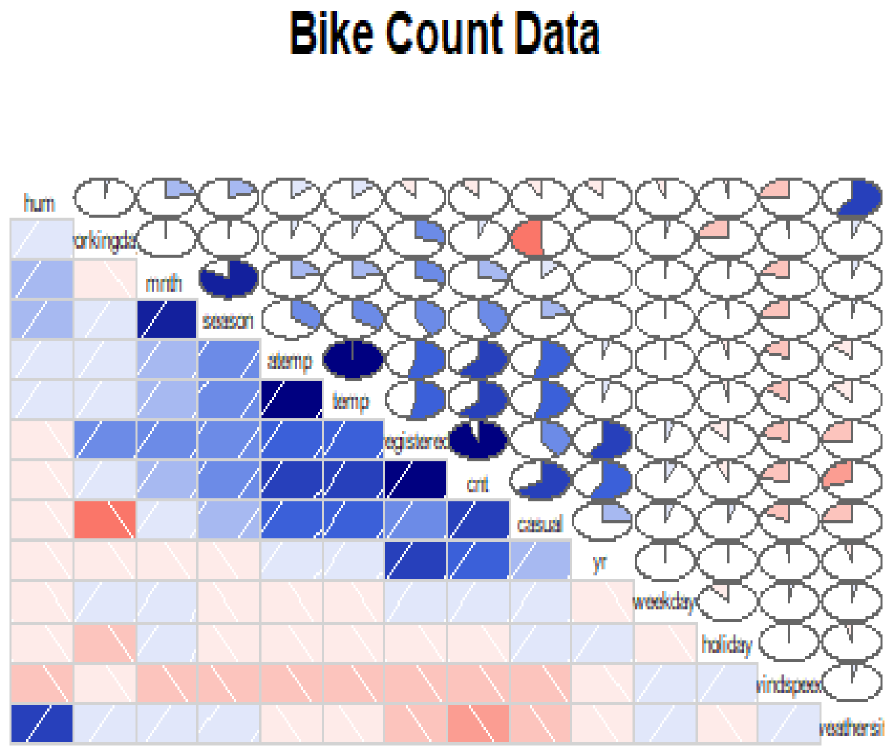


Fig 3.3 Correlation plot

4. Modeling

4.1 Choosing a model

There are various models that can be used to predict the dependent variable. Model selection depends on the type of dependent variable. In our case the dependent variable i.e. ‘count of bikes’ can be treated as an continuous interval therefore the only method that we can come up with is the regression analysis.

First we will split the data into training set on which the model will be trained and the test set with which the model predictions will be compared to check the accuracy of our model. There are multiple independent variables that affect the dependent variable in our case; however we will use only one out of the highly correlated predictors like season (season & month) and denorm_temp (denorm_temp & denorm_atemp) to avoid multi-collinearity. Therefore we will be using the multiple linear regression model and the regression tree model.

4.2 Multiple Linear Regression

```
> linearModel <- lm(cnt~season+weathersit+denorm_temp+denorm_hum+denorm_wind+yr, data=training_set)
> summary(linearModel)

Call:
lm(formula = cnt ~ season + weathersit + denorm_temp + denorm_hum +
    denorm_wind + yr, data = training_set)

Residuals:
    Min      1Q  Median      3Q     Max
-3929.1  -422.0    78.2   526.2  3006.9

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 3171.607    267.175  11.871  < 2e-16 ***
season       338.822     39.416   8.596  < 2e-16 ***
weathersit  -491.389    101.158  -4.858 1.61e-06 ***
denorm_temp  118.024      5.087  23.203  < 2e-16 ***
denorm_hum   -16.281      4.072  -3.998 7.39e-05 ***
denorm_wind  -47.612      8.318  -5.724 1.84e-08 ***
yr          1977.133     81.978  24.118  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 891.3 on 480 degrees of freedom
Multiple R-squared:  0.778,     Adjusted R-squared:  0.7752
F-statistic: 280.3 on 6 and 480 DF,  p-value: < 2.2e-16
```

Fig 4.1 Multiple Linear Regression Summary

It can be seen clearly from the statistics above that we were able to explain almost 78% (Adjusted R-squared value) of our data using multiple linear regression. Since we were able to able to identify the type of our dependent variable, we can say that this model proved to be quite accurate in predicting it. The high F-statistic value is also proof of the fact that our target variable depends on most of our predictor variables.

We also created a subset of the data for prediction purposes. All predictions can be seen in the appendix.

Similar results were obtained using Python as shown below. Linearity between variables and all statistics can be seen in the appendix.

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    cnt   R-squared:                       0.789
Model:                            OLS   Adj. R-squared:                  0.787
Method:                 Least Squares   F-statistic:                     450.6
Date:                Sat, 15 Sep 2018   Prob (F-statistic):         1.52e-240
Time:                        18:03:35   Log-Likelihood:                -6001.4
No. Observations:                 731   AIC:                         1.202e+04
Df Residuals:                     724   BIC:                         1.205e+04
Df Model:                           6
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const       3128.6541    214.187     14.607      0.000    2708.153    3549.155
season       408.4879     32.537     12.555      0.000     344.611     472.365
weathersit  -554.6193     79.721     -6.957      0.000    -711.131    -398.108
atemp         90.4915      3.385     26.732      0.000      83.846      97.137
hum          -12.7254      3.182     -4.000      0.000     -18.972      -6.479
windspeed    -37.1277      6.882     -5.395      0.000     -50.639     -23.616
yr          2032.0919     66.777     30.431      0.000    1900.992    2163.191
==============================================================================
Omnibus:                       95.432   Durbin-Watson:                   0.948
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              205.810
Skew:                          -0.742   Prob(JB):                     2.04e-45
Kurtosis:                       5.134   Cond. No.                         439.
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

Fig 4.2 OLS summary in Python

4.3 Regression Tree

The use of a regression tree was not normal in this case therefore we have added all the code and related figures in the appendix.

5. Conclusion

We were able to explain our data very efficiently using the multiple linear regression model that can be seen from the results explained above. For training our model purposes, we used subset of the data (training_set). We can conclude that by identifying the variables in our data, we can make a better choice about the type of model we want for our data which in our case proved to be the multiple regression model.

Also we were able to identify the dominant predictors that influenced the bike rental count and were able to derive the following conclusions that would benefit the bike rental company to forecast the count of bike rentals for any given factors such as season, weather situation, temperature, wind speed, humidity, whether working day or holiday. The conclusion are as follows:

Total bike rental count changes depending on season. We see higher number of rentals for summer and fall seasons, while the lesser for winter and spring.

There is a strong correlation between actual air temperature and the total number of bike rentals,

Weather condition and total number of bike rentals also seemed to be significantly correlated. The two popular weather conditions for bike rentals are Clear and Cloudy weather.

There exist a significant correlation between number of total bike rentals and type of day. For days which were not holiday, the number of rentals were higher compared to days which were holidays.

As conclusion, we can say that the amount of bike rentals depends mainly on the weather and on the temperature.

Appendix A - Details of Variables

instant: Record index

dteday: Date

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

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

mnth: Month (1 to 12)

holiday: weather day is holiday or not (extracted from Holiday Schedule)

weekday: Day of the week

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

weathersit: (extracted 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

Appendix B – Regression Tree Code and Figures

R Code

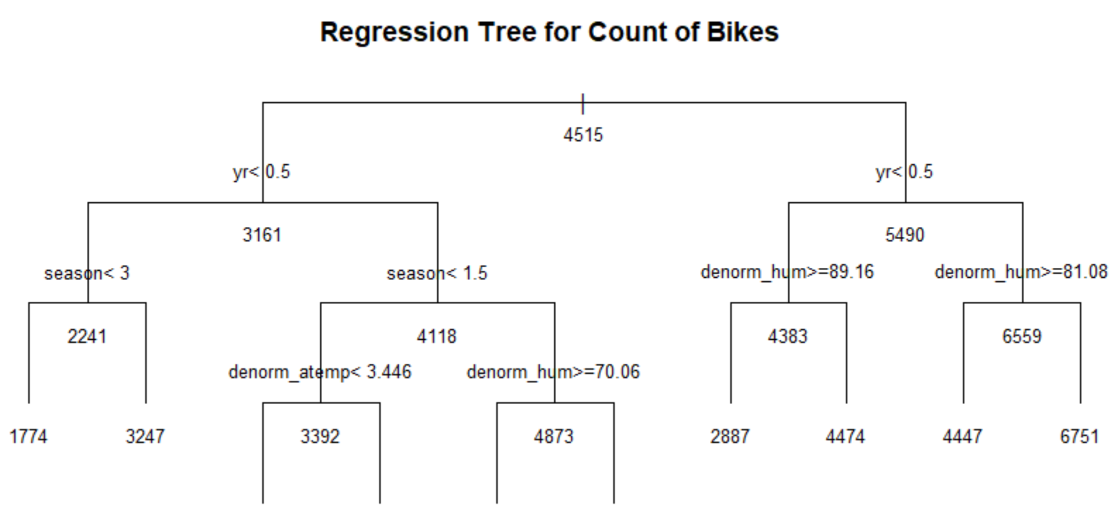
```
fit <- rpart(cnt~season+mnth+weathersit+denorm_atemp+denorm_hum+denorm_wind+yr,
             method="anova", data=training_set)

predictions_DT = predict(fit, test_set)
head(predictions_DT)


printcp(fit) # display the results
plotcp(fit) # visualize cross-validation results
summary(fit) # detailed summary of splits

plot(fit, uniform=TRUE,
     main="Regression Tree for Count of Bikes ")
text(fit, use.n=FALSE, all=TRUE, cex=.8)
```

Figures



Appendix C – Predictions

R Code

```
predictions <- predict(linearModel,prediction_data)
```

Output

```
> predictions <- predict(linearModel,test_set)
> predictions
         2          4          5          8         11         13         16         20         21         22 
1673.60741 1711.66650 2026.20443  775.26220 1014.38683 1264.06679 1971.25944 1533.58742 1188.46214 1203.04405 
        24         26         31         32         33         34         37         50         53         58 
1309.43741  -43.01213 1006.47976 1129.14368  920.14496 1513.08628 2282.77860 2364.32874 1520.77131 2473.54337 
        59         65         67         68         69         71         73         82         84         87 
1491.78339 1033.16384 2627.36832 1259.77936 2415.01806 2228.49061 2632.99890 1732.74182 2337.13566 2711.15352 
        88         89         97        104        106        107        111        114        115        118 
2856.07562 1982.09458 3341.17839 3773.98828 1287.34097 3197.91728 3260.39835 3215.73961 3923.25699 3182.93714 
       126        130        132        133        134        137        138        139        145        150 
3383.01031 4202.13187 3559.37118 2786.30412 2879.03354 2788.98136 2914.03025 3171.07015 4453.85524 4947.73314 
       151        167        173        174        175        179        181        183        189        190 
5321.02245 3628.43448 5119.01558 4394.40574 5127.85795 5387.75661 5318.99685 5757.26340 4242.84697 5292.98385 
       193        195        202        206        214        216        219        220        222        223 
5608.66942 4986.06740 4948.36816 5353.60060 5641.29334 4337.39978 5002.69672 5419.05932 5675.89841 5516.96573 
       224        229        230        238        240        246        248        249        250        256 
5605.32918 5369.80119 4890.56541 5128.28101 4788.18821 4706.48327 4030.82074 2223.96557 3290.80857 4750.39859 
       260        261        262        263        264        271        275        276        277        281 
3213.89055 3866.79721 3699.98449 3511.51144 3788.86177 4266.40933 2581.30021 3226.39475 3964.45360 4698.55199 
       290        294        295        296        297        300        301        303        304        313 
4551.35163 3821.67945 4115.04779 3906.11137 4025.53833 3252.73403 2749.28870 3282.86395 3493.01363 3877.15016 
       316        317        320        321        324        327        330        333        334        337 
3493.13161 3891.78660 2691.14713 2583.59968 3462.36996 2740.06568 3848.05162 2966.39763 3030.59354 3447.41808 
       340        347        352        356        357        360        363        366        373        376 
2386.16554 3249.46833 2917.33298 2547.67992 2153.67309 2270.32550 2136.68113 4363.97423 4556.06939 3283.05669 
       377        380        382        384        386        391        393        394        400        401 
3797.88715 3492.56165 3350.35137 3593.67129 2458.54028 3968.80369 4246.82359 4346.55049 3367.90498 3353.80471 
       403        407        410        412        417        425        430        431        434        443 
4738.52674 2197.71798 4052.41564 3799.74550 4024.71721 3589.11676 3847.51761 4100.71437 3853.56580 4458.12518 
       445        446        447        450        456        457        458        461        468        470 
5460.80201 5225.90428 5736.49111 4187.96162 4250.80381 4611.05688 4980.94058 5494.33877 4910.23381 5708.82639 
       472        474        480        482        484        485        488        490        491        494 
6251.98891 5007.30382 5148.67022 4868.39799 4777.14313 5604.49195 5782.20586 6154.55042 5627.93660 5065.08462 
       496        499        500        509        513        515        518        520        526        527 
5296.03699 6132.12519 5115.78939 5761.43306 6397.03220 6341.73359 5540.47134 6342.53345 7162.01065 7118.70864 
       529        531        532        534        536        538        541        543        545        546 

5482.41769 6250.46442 6413.05875 6240.43126 6613.53827 7896.14947 7608.64982 6512.14772 7649.92998 8033.46020 
       548        549        554        562        563        568        570        572        575        576 
7872.99018 7714.85070 8186.52282 7166.37590 7390.07486 4962.63106 7271.20204 7488.33180 7421.83717 7173.73779 
       581        582        583        585        589        593        596        599        602        606 
6752.00359 7310.50936 7007.53927 6801.31177 6227.72069 7099.30877 6942.76615 7001.59319 6903.69114 7151.33253 
       608        614        616        617        618        619        621        622        623        628 
7442.36037 7010.10359 6884.89285 5696.90543 6505.86698 6320.01687 6693.16611 6826.15215 6816.61079 6234.27802 
       631        632        634        637        638        639        642        643        647        651 
6377.81464 6531.17082 6437.50830 6364.56382 6467.80480 6609.71880 6719.74126 6672.37054 4943.13557 5867.57505 
       652        655        656        657        661        662        664        667        677        681 
5978.91138 6176.06499 6145.52738 5533.83311 6587.45995 6740.73754 5928.09554 4825.34207 4356.61337 5922.97326 
       682        683        684        685        701        704        714        719        722        723 
5997.80651 4311.17396 5136.86515 4863.28152 4728.19184 5957.89305 5165.56235 5307.39716 3508.77404 4151.98375 
       724        726        727        729 
3309.01082 2068.78517 2790.57604 3343.28765 
```

Appendix D – R Code

Importing and Sub-setting Data

```
#IMPORTING DATASET

bikeData_Original = read.csv("day.csv")

#CREATING A DUPLICATE DATASET FROM THE ORIGINAL DATASET

bikeData = bikeData_Original

# CREATING TRAINING AND TEST SETS AND CHECK BOTH THE SETS

split = sample.split(bikeData$cnt, SplitRatio = 2/3)
training_set = subset(bikeData, split == TRUE)
test_set = subset(bikeData, split == FALSE)
head(training_set)
head(test_set)
```

Creating Boxplots

```
boxplot(cnt~season, data=bikeData, main="Boxplot for Season", xlab="Season", ylab="Bike Count")
boxplot(cnt~holiday, data=bikeData, main="Boxplot for Holiday", xlab="Holiday", ylab="Bike Count")
boxplot(cnt~mnth, data=bikeData, main="Boxplot for Month", xlab="Month", ylab="Bike Count")
boxplot(cnt~weathersit, data=bikeData, main="Boxplot for Weather Situation", xlab="WeatherSit.", ylab="Bike Count")
boxplot(cnt~weekday, data=bikeData, main="Boxplot for Weekday", xlab="WeekDay", ylab="Bike Count")
boxplot(cnt~workingday, data=bikeData, main="Boxplot for Working Day", xlab="WorkingDay", ylab="Bike Count")
```

Predictor Importance using Random Forest

```
predictor_importance <- randomForest(cnt ~ season + yr + mnth + holiday + weekday + workingday + temp + atemp + hum + windspeed, data = training_set,
                                     ntree = 100, keep.forest = FALSE, importance = TRUE)
importance(predictor_importance, type = 1)
```

Correlation between Variables

```
truncated = select(bikeData,season,mnth,weathersit,temp,atemp,hum,windspeed,cnt)

symnum(cor(truncated))

corrgram(bikeData[3:16], order=TRUE, lower.panel=panel.shade,
         upper.panel=panel.pie, text.panel=panel.txt,
         main="Bike Count Data")
```

Multiple Linear Regression Model

```
#BUILDING LINEAR REGRESSION MODEL ON THE TRAINING SET USING MOST IMPORTANT PREDICTORS

linearModel <- lm(cnt~season+weathersit+denorm_temp+denorm_hum+denorm_wind+yr, data=training_set)
summary(linearModel)

#USING THE LINEAR REGRESSION MODEL FOR PREDICTIONS ON TEST SET
predictions <- predict(linearModel,test_set)
predictions
```

Regression Tree

```
#USING RANDOM FOREST REGRESSION MODEL ON TRAINING SET

fit <- rpart(cnt~season+weathersit+denorm_atemp+denorm_hum+denorm_wind+yr,
             method="anova", data=training_set)

#USING RANDOM FOREST FOR PREDICTIONS ON TEST SET

predictions_DT = predict(fit, test_set)
head(predictions_DT)

# VISULALIZATION OF THE RESULTS

printcp(fit) # display the results
plotcp(fit) # visualize cross-validation results
summary(fit) # detailed summary of splits

plot(fit, uniform=TRUE,
     main="Regression Tree for Count of Bikes ")
text(fit, use.n=FALSE, all=TRUE, cex=.8)
```

Mean Absolute Error Percentage (MAE):

```
> mape = function(y, yhat){
+             mean(abs((y-yhat)/y))*100
+ }
> mape(test_set[,16], predictions)
[1] 23.02673
```

Appendix E – Python Code

Importing Data

```
os.chdir('E:\Project Data')
```

```
bikeData = pd.read_csv('day.csv')
```

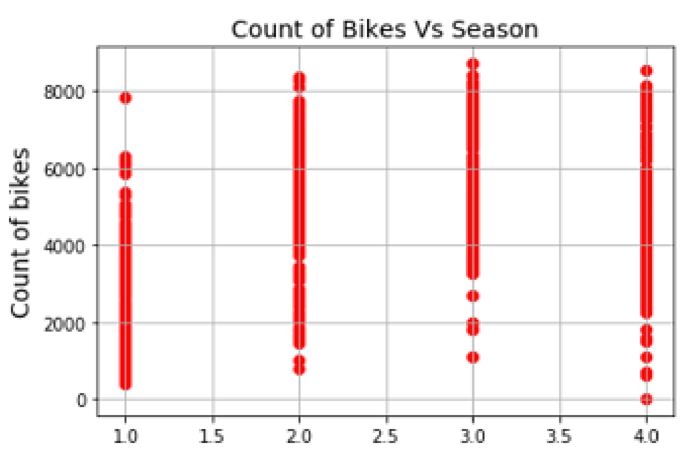
```
df = DataFrame(bikeData,columns=['instant','dteday','season','yr','mnth','holiday','weekday',
                                 'workingday','weathersit','temp','atemp','hum','windspeed',
                                 'casual','registered','cnt'])
```

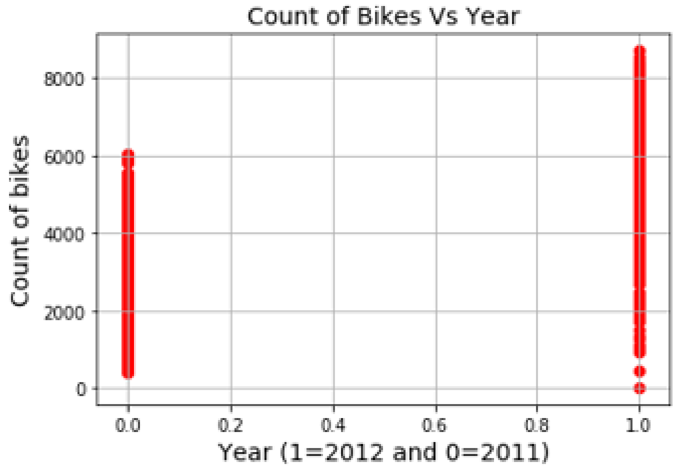
Linear Relationship between Variables

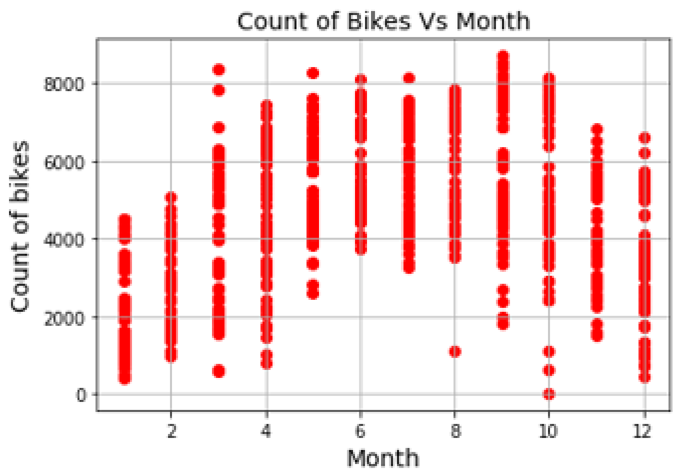
```
#LINEARITY CHECK FOR VARIABLE INTER-RELATIONSHIP
plt.scatter(df['season'], df['cnt'], color='red')
plt.title('Count of Bikes Vs Season', fontsize=14)
plt.xlabel('Season', fontsize=14)
plt.ylabel('Count of bikes', fontsize=14)
plt.grid(True)
plt.show()

plt.scatter(df['yr'], df['cnt'], color='red')
plt.title('Count of Bikes Vs Year', fontsize=14)
plt.xlabel('Year (1=2012 and 0=2011)', fontsize=14)
plt.ylabel('Count of bikes', fontsize=14)
plt.grid(True)
plt.show()

plt.scatter(df['mnth'], df['cnt'], color='red')
plt.title('Count of Bikes Vs Month', fontsize=14)
plt.xlabel('Month', fontsize=14)
plt.ylabel('Count of bikes', fontsize=14)
plt.grid(True)
plt.show()
```




Count of Bikes Vs Year
Count of bikes
8000
6000
4000
2000
0
0.0
0.2
0.4
0.6
0.8
1.0
Year (1=2012 and 0=2011)


Count of Bikes Vs Month
Count of bikes
8000
6000
4000
2000
0
2
4
6
8
10
12
Month

Multiple Linear Regression

Using STATSMODEL API(Ordinary Least Squares Method):

```
#Using Statsmodels API (OLS)

X = df2[['season','weathersit','atemp','hum','windspeed','yr']]
Y = df2['cnt']

X = sm.add_constant(X)

model = sm.OLS(Y, X).fit()
predictions = model.predict(X)

print_model = model.summary()
print(print_model)
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    cnt   R-squared:                       0.789
Model:                            OLS   Adj. R-squared:                  0.787
Method:                 Least Squares   F-statistic:                     450.6
Date:                Sat, 15 Sep 2018   Prob (F-statistic):          1.52e-240
Time:                        18:03:35   Log-Likelihood:                -6001.4
No. Observations:                 731   AIC:                         1.202e+04
Df Residuals:                     724   BIC:                         1.205e+04
Df Model:                           6
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const       3128.6541    214.187     14.607      0.000    2708.153    3549.155
season       408.4879     32.537     12.555      0.000     344.611     472.365
weathersit  -554.6193     79.721     -6.957      0.000    -711.131    -398.108
atemp         90.4915      3.385     26.732      0.000      83.846      97.137
hum          -12.7254      3.182     -4.000      0.000     -18.972      -6.479
windspeed    -37.1277      6.882     -5.395      0.000     -50.639     -23.616
yr          2032.0919     66.777     30.431      0.000    1900.992    2163.191
==============================================================================
Omnibus:                       95.432   Durbin-Watson:                   0.948
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              205.810
Skew:                          -0.742   Prob(JB):                     2.04e-45
Kurtosis:                       5.134   Cond. No.                         439.
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

Using SciKit Learn for Multiple Linear Regression:

```
#USING SCIKIT LEARN TO RUN MULITPLE LINEAR REGRESSION
regr = linear_model.LinearRegression()
regr.fit(X, Y)

print('\n\n\n**** Coefficients as Predicted by sklearn regression Model **** \n')
print('Coefficients: \n', regr.coef_)
```

```
**** Coefficients as Predicted by sklearn regression Model ****

Coefficients:
 [   0.          408.48794301 -554.61929394   90.49147588  -12.72535654
  -37.12768214 2032.09186733]
```