**Bike Rental Count Prediction**

Eeswara Chaitanya Vutukuru

24 October 2018

**1. Introduction**

**1.1 Problem Statement**

The objective of this Case is to Predication of bike rental count on daily based on the environmental and seasonal settings.

**1.2 Data**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **instant** | **dteday** | **season** | **yr** | **mnth** | **holiday** | **weekday** | **workingday** |
| 1 | 01-01-2011 | 1 | 0 | 1 | 0 | 6 | 0 |
| 2 | 02-01-2011 | 1 | 0 | 1 | 0 | 0 | 0 |
| 3 | 03-01-2011 | 1 | 0 | 1 | 0 | 1 | 1 |
| 4 | 04-01-2011 | 1 | 0 | 1 | 0 | 2 | 1 |
| 5 | 05-01-2011 | 1 | 0 | 1 | 0 | 3 | 1 |
| 6 | 06-01-2011 | 1 | 0 | 1 | 0 | 4 | 1 |
| 7 | 07-01-2011 | 1 | 0 | 1 | 0 | 5 | 1 |
| 8 | 08-01-2011 | 1 | 0 | 1 | 0 | 6 | 0 |
| 9 | 09-01-2011 | 1 | 0 | 1 | 0 | 0 | 0 |
| 10 | 10-01-2011 | 1 | 0 | 1 | 0 | 1 | 1 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **weathersit** | **temp** | **atemp** | **Hum** | **windspeed** | **casual** | **registered** | **cnt** |
| 2 | 0.344167 | 0.363625 | 0.805833 | 0.160446 | 331 | 654 | 985 |
| 2 | 0.363478 | 0.353739 | 0.696087 | 0.248539 | 131 | 670 | 801 |
| 1 | 0.196364 | 0.189405 | 0.437273 | 0.248309 | 120 | 1229 | 1349 |
| 1 | 0.2 | 0.212122 | 0.590435 | 0.160296 | 108 | 1454 | 1562 |
| 1 | 0.226957 | 0.22927 | 0.436957 | 0.1869 | 82 | 1518 | 1600 |
| 1 | 0.204348 | 0.233209 | 0.518261 | 0.089565 | 88 | 1518 | 1606 |
| 2 | 0.196522 | 0.208839 | 0.498696 | 0.168726 | 148 | 1362 | 1510 |
| 2 | 0.165 | 0.162254 | 0.535833 | 0.266804 | 68 | 891 | 959 |
| 1 | 0.138333 | 0.116175 | 0.434167 | 0.36195 | 54 | 768 | 822 |
| 1 | 0.150833 | 0.150888 | 0.482917 | 0.223267 | 41 | 1280 | 1321 |

Different variables present in the data set are:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Numeric Variables** | **Categorical Variables** |  |  |  |
| temp | instant |  |  |  |
| atemp | dteday |  |  |  |
| hum | season |  |  |  |
| windspeed | yr |  |  |  |
| casual | mnth |  | **Target Variable** | |
| registered | holiday |  | cnt (Numeric Variable) | |
|  | weekday |  |
|  | workingday |  |  |  |
|  | weathersit |  |  |  |

1. **Methodology**

**2.1 Pre-Processing**

Data pre-processing is a data mining technique that involves transforming raw data into an understandable format. Real-world data is often incomplete, inconsistent, and/or lacking in certain behaviours or trends, and is likely to contain many errors. Data pre-processing is a proven method of resolving such issues. Data pre-processing prepares raw data for further processing.

Below are several Data Pre-processing techniques used in this project.

* + 1. **Identifying the missing values:**

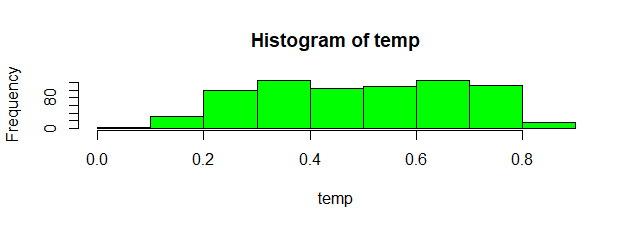
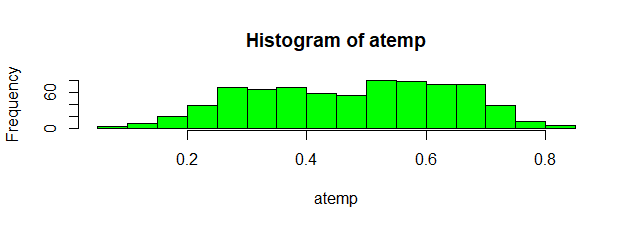
First the given datasets are evaluated for the missing values. Since there are no missing values in the given dataset, the dataset is ready for the next level pre-processing techniques.

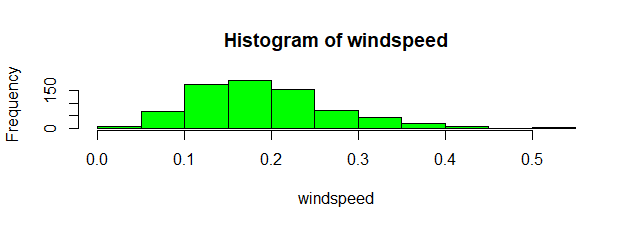
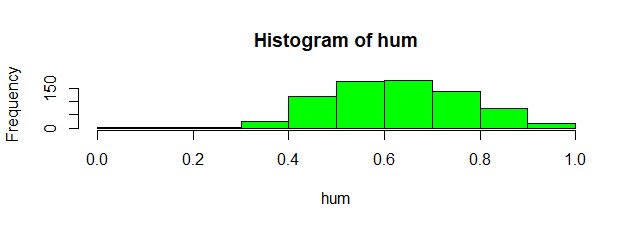
* + 1. **Conversion of variables into the required form:**

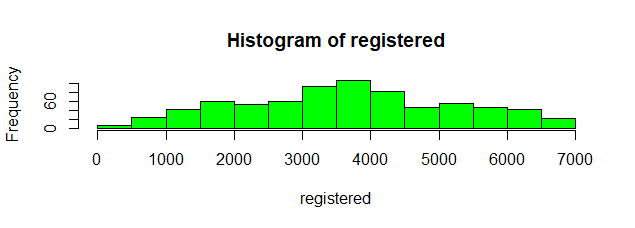
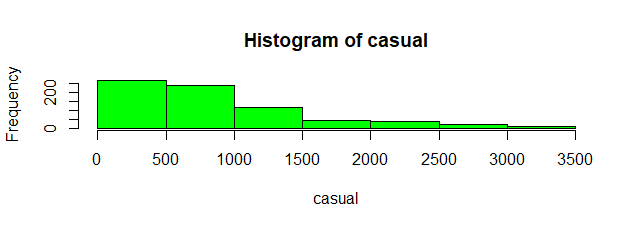
It is very important to identify the type of each variable in each dataset and converting it to the required format. If we observe the variables “instant”, “dteday”, “season”, “yr”, “mnth”, “holiday”, “weekday”, “workingday”, “weathersit” in the dataset, they appear to be a numeric variables (since they have numeric values). But, it doesn’t make any sense if we take median / mean or any kind of numeric operations on these variables. Hence, these variables need to be converted to a categorical type (i.e, factor in R).

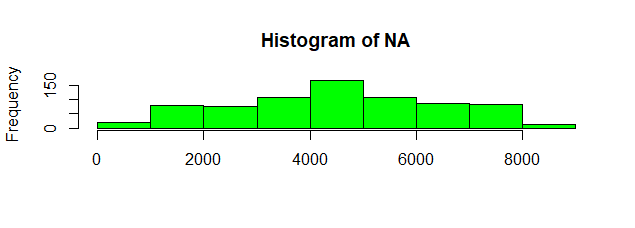
**2.1.3 Distribution of the numeric variables:**

Next part of the EDA is histogram plots. Histogram plot represents the distribution of the numeric data. It also helps us to understand the skewness of the data. Below are the figures which shows the histogram plots of each numeric variable in the Train\_data.







The above histogram plots show that most of the numeric variables are normally distributed except the variable “casual” which is left skewed.

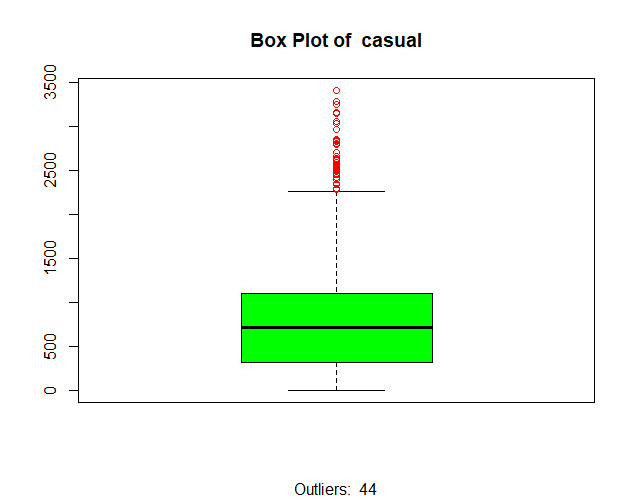
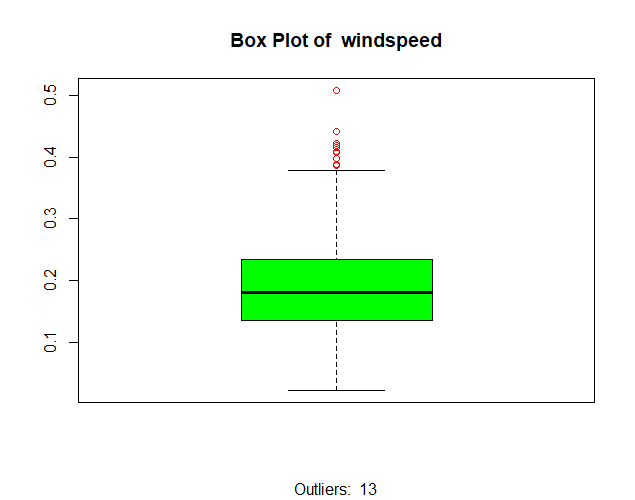
* + 1. **Outlier Analysis and Removal**

An outlier is an observation that lies an abnormal distance from other values in a random sample from a population. Outliers can be detected using boxplots.

A box plot is constructed by drawing a box between the upper and lower quartiles with a solid line drawn across the box to locate the median. The following quantities (called fences) are needed for identifying extreme values in the tails of the distribution:

1. lower inner fence: Q1 - 1.5\*IQ
2. upper inner fence: Q3 + 1.5\*IQ
3. lower outer fence: Q1 - 3\*IQ
4. upper outer fence: Q3 + 3\*IQ

Since there are 6 numeric variables, and only two variables out of them seem to be having outliers, let us plot the outlier for the two variables.

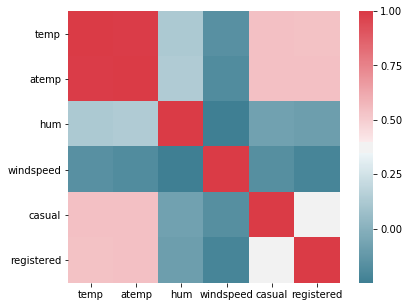


These outlier values are replaced with NAs which are in turn replaced with Imputed values from knn Imputation in R and median in python.

**2.1.5** **Feature Selection:**

The most important part of building a machine learning model is Feature Selection. For carrying out feature selection, the dataset is divided into numeric and categorical data.

A correlation plot which shows the correlation between each numerical variable with each other numerical variable is plotted as shown below.



The above figure shows the correlation between the numerical variables on a scale of 0 to 1 using a heatmap with squares.

Thus, the plot clearly depicts that the variable ‘atemp’ is highly correlated with ‘temp’ (Since correlation > 0.8). It is obvious since the temperature and feeling temperature vary similarly or will have a fixed difference all the time. Hence as a part of feature selection, the variable atemp is removed from the dataset since it carries same information as the variable ‘temp’.

The variable instant which is the record index is also removed from the dataset since it doesn’t have any impact on the target variable cnt. The variable dteday is also removed from the dataset since the date variable cannot directly be given as input to the machine learning model. Instead the categorical variables showing whether the bike rental day is weekday/weekend or if it is a public holiday or not and also variables like yr, mnth, hr , weather sit makes much sense for building a regression model.

* + 1. **Feature Scaling:**

Feature scaling is a method used to standardize the range of independent variables or features of data. In data processing, it is also known as data normalization and is generally performed during the data pre-processing step.

In this project, the data is normalized that is converted to values which ranges from 0 to 1. This becomes the last step of data pre-processing after which the data is divided into X\_train and Y\_train as input predictor and target variables.

Since most of the data given is already normalized except the two numeric variables casual and registered, these two variables are normalized and given to the model.

1. **Model Building and Selection**

After the preparation or pre-processing of the data, the dataset is split into Test and Train dataset using random sampling method. 80 percent of the dataset goes to Train for fitting/training the machine learning regression model and the remaining 20 percent goes to test dataset for evaluating purpose.

After feature selection, i.e., removing the unnecessary variables from the dataset, we got a total 13 variables in the dataset including the target variable ‘cnt’. Now the dataset containing the predictor variables (independent variables) and target variable (dependent variable) ‘cnt’ in the train dataset are used to train the machine learning model which learns the data pattern, finds out the best fit or the best model to predict the target variable effectively. The test dataset is used to evaluate the model performance to make the model robust enough to predict the output correctly while working on future unknown data.

According to the problem statement, we can clearly understand that the target variable which is the count of bike rentals on a period is a numerical variable. Hence this problem comes under Regression type since we must predict a numeric dependent variable.

Some of the available regression models are

* LinearRegression which comes under sklearn.linear\_model library,
* The “sm” model which is under statsmodels.api library in python and
* “lm” model which stands for Linear Model in R.

Now that we have the above models which are available for building a regression model, let us see what are the different evaluation metrics that are to be looked upon for evaluating whether the model that is built is good or bad or a great one.

1. **Evaluation Metrics:**

For a regression problem there are several model evaluation metrics and below are the ones that are more significant.

* 1. **R-Squared measure:**

R-squared is a statistical measure of how close the data are to the fitted regression line. Or it can also be explained as the percentage of the response variable variation that is explained by a linear model. It is also known as the coefficient of determination, or the coefficient of multiple determination for multiple regression.

R-squared = Explained variation / Total variation

R-squared is always between 0 and 100%:

* 0% indicates that the model explains none of the variability of the response data around its mean.
* 100% indicates that the model explains all the variability of the response data around its mean.
* In general, the higher the R-squared, the better the model fits your data. However, its hard to say whether the obtained R-squared value is a good or bad. Because the threshold for a good R-squared value depends widely on the domain. Therefore, it's most useful as a tool for comparing different models.
  1. **ANOVA Hypothesis Test**

Analysis of variance (ANOVA) can determine whether the means of three or more groups are different.

The hypotheses of interest in an ANOVA are as follows:

* H0: μ1 = μ2 = μ3 ... = μk
* H1: Means are not all equal.

where k = the number of independent comparison groups.

* + 1. **F-Statistic**

ANOVA uses F-tests to statistically test the equality of means.

The test statistic for testing H0: μ1 = μ2 = ... =   μk is:

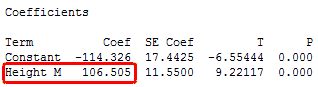
http://sphweb.bumc.bu.edu/otlt/MPH-Modules/BS/BS704_HypothesisTesting-ANOVA/lessonimages/equation_image35.gif

In the test statistic, nj = the sample size in the jth group (e.g., j =1, 2, 3, and 4 when there are 4 comparison groups), http://sphweb.bumc.bu.edu/otlt/MPH-Modules/BS/BS704_HypothesisTesting-ANOVA/lessonimages/equation_image36.gif is the sample mean in the jth group, and http://sphweb.bumc.bu.edu/otlt/MPH-Modules/BS/BS704_HypothesisTesting-ANOVA/lessonimages/equation_image37.gif is the overall mean.  k represents the number of independent groups (in this example, k=4), and N represents the total number of observations in the analysis. Note that N does not refer to a population size, but instead to the total sample size in the analysis (the sum of the sample sizes in the comparison groups, e.g., N=n1+n2+n3+n4). The test statistic is complicated because it incorporates all the sample data. While it is not easy to see the extension, the F statistic shown above is a generalization of the test statistic used for testing the equality of exactly two means.

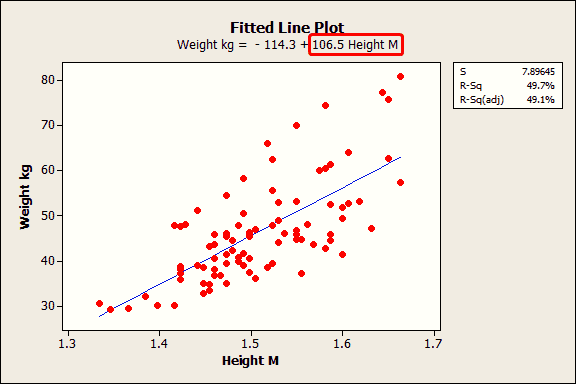
* + 1. **p-value:**
* The p-value for each term tests the null hypothesis that the coefficient is equal to zero (no effect). A low p-value (< 0.05) indicates that you can reject the null hypothesis. In other words, a predictor that has a low p-value is likely to be a meaningful addition to your model because changes in the predictor's value are related to changes in the response variable.
* Conversely, a larger (insignificant) p-value suggests that changes in the predictor are not associated with changes in the response.
  1. **Regression Coefficients:**

Regression coefficients represent the mean change in the response variable for one unit of change in the predictor variable while holding other predictors in the model constant. This [statistical control](https://blog.minitab.com/blog/adventures-in-statistics/a-tribute-to-regression-analysis) that regression provides is important because it isolates the role of one variable from all of the others in the model.

The key to understanding the coefficients is to think of them as slopes, and they’re often called slope coefficients. This is illustrated with an example where a person’s height is used to model their weight as shown below.



The fitted line plot shows the same regression results graphically.



The equation shows that the coefficient for height in meters is 106.5 kilograms. The coefficient indicates that for every additional meter in height you can expect weight to increase by an average of 106.5 kilograms.

The blue fitted line graphically shows the same information. If you move left or right along the x-axis by an amount that represents a one-meter change in height, the fitted line rises or falls by 106.5 kilograms. However, these heights are from middle-school aged girls and range from 1.3 m to 1.7 m. The relationship is only valid within this data range, so we would not actually shift up or down the line by a full meter in this case.

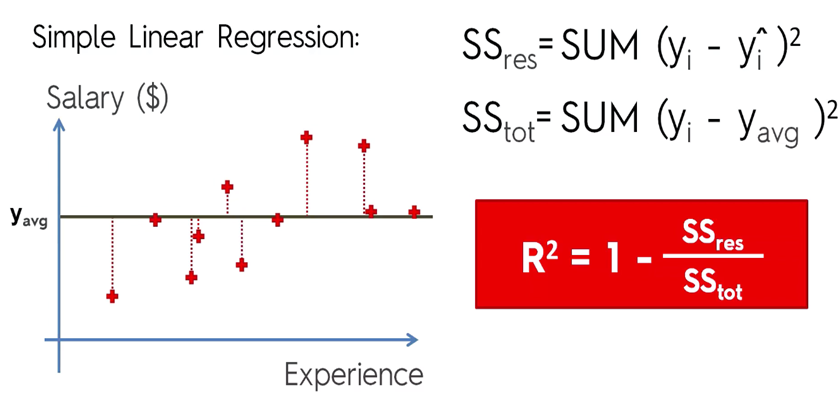
If the fitted line was flat (a slope coefficient of zero), the expected value for weight would not change no matter how far up and down the line you go. So, a low p-value suggests that the slope is not zero, which in turn suggests that changes in the predictor variable are associated with changes in the response variable.

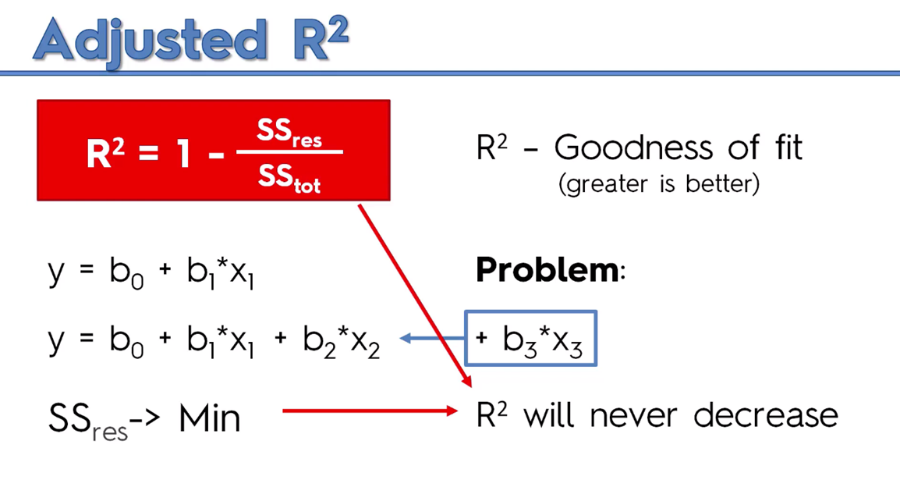
Since there is only one predictor variable in this example, this is explained using a simple 2-D graph using simple linear regression. But as our problem statement has multiple predictor variables, one would need an extra spatial dimension for each additional predictor to plot the results. However, the concepts hold true for multiple linear regression as well.

* 1. **Adjusted R-squared measure:**

This is another measure which is like R-squared which overcomes the problems associated with R-squared measure.

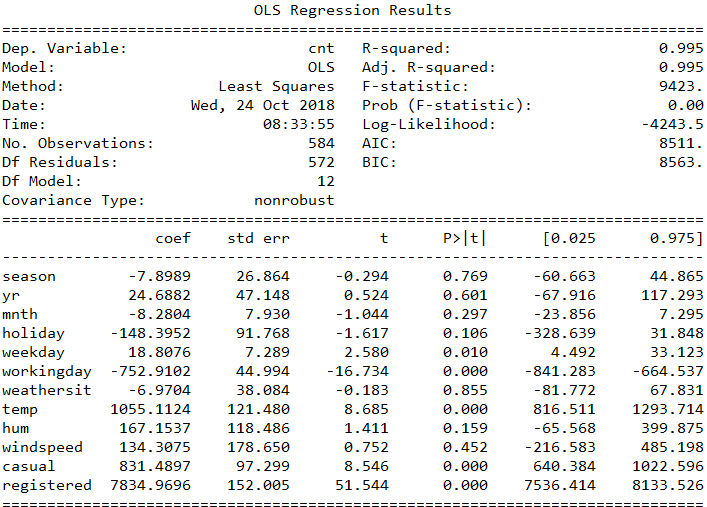
* the problem with R-squared is that it will either stay the same or increase with addition of more variables, even if they do not have any relationship with the output variables. This is where “Adjusted R square” comes to help. Adjusted R-square penalizes you for adding variables which do not improve your existing model.
* Hence, if you are building Linear regression on multiple variable, it is always suggested that you use Adjusted R-squared to judge goodness of model. In case you only have one input variable, R-square and Adjusted R squared would be exactly same.
* Typically, the more non-significant variables you add into the model, the gap in R-squared and Adjusted R-squared increases.





1. **Model Evaluation**

**5.1 Statsmodels(sm) in python summary:**



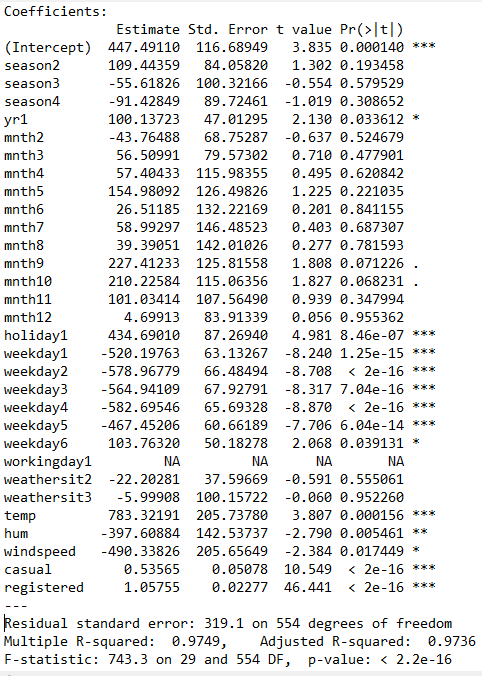
Below is the analysis of summary from the above model:

* As per the summary, the R-squared value stood at 0.995 which clearly shows that 99.5 percent variance in the target variable “cnt” is explained by the predictor variables. Its relatively a high R-squared value which shows the model or regression fit is perfect.
* There is not much difference between R-squared and adjusted R-squared values which shows there is not much noise added into the model due to the addition of all the predictor variables in the model.
* The p-values for each variable as shown in the summary indicate that the variables weekday, working day, temp, casual and registered are much significant for the model as the p-value for all these variables is below 0.05. This is no surprise because the count of bike rentals depends on mood, availability of the customers which varies with the day depending on their working days or day offs and also on weather conditions such as outside temperature.
* Overall it can be said, as per the given data and with the best fit of the regression model that is built, the most accurate predictions can be made.

**5.2 Linear Regression from sklearn in python summary:**

* Sklearn is a library in python which provides a method/model for performing linear regression.
* The method is “LinearRegression()” which has inbuilt functions such as fit for fitting the train data into the model for machine learning and predict for predicting the output.
* Though this model doesn’t have a summary function like statsmodel, we can check the R-squared value using the function sklearn.metrics.r2\_score().
* This model gives R-squared value of 0.995 which is same as that of statsmodel.

**5.3 Linear model (lm) in R Summary:**



Below is the analysis of summary from the above model:

* Based on the above summary for the linear model (lm), it can be observed that the R-squared value is 0.9749 which means that 97.49 percent variance in the target variable “cnt” is explained by the predictor variables. Though it is less when compared with statsmodels in python, it is also a good score.
* The rows with star marks in the above summary shows the variables which are more significant for the model. More the number of stars, higher is the significance for that variable. Hence holiday, weekday temperature, casual and registered form the list of highly significant predictor variables which is similar to that of the summary displayed in statsmodel of python.
* Also, there is not much difference (minute) between R-squared and adjusted R-square values which is the expected behaviour of a good model.

1. **Code**
   1. **Python Code:**

#Reading the dataset

df = pd.read\_csv('day.csv')

#Exploratory data analysis

df.dtypes

for i in range(0,9):

df.iloc[:,i] = df.iloc[:,i].astype(object)

df.dtypes

#Checking for missing values

df.isnull().sum()

df.columns.values

list(df)

#Saving numeric columns

cnames = df.select\_dtypes(exclude = ['object'])

numeric\_variables = list(cnames.columns.values)

numeric\_variables = numeric\_variables[0:-1]

#Saving Categoical columns

cat\_data = df.select\_dtypes(include = ['object'])

categorical\_variables = list(cat\_data.columns.values)

#Plotting the variables using scatterplot

for i in numeric\_variables :

plt.scatter(df[i], df['cnt'], color='red')

plt.title('Plot of cnt Vs '+i, fontsize=14)

plt.xlabel(i, fontsize=14)

plt.ylabel('cnt', fontsize=14)

plt.grid(True)

plt.show()

#Box plot

plt.boxplot(df['temp'])

plt.boxplot(df['atemp'])

plt.boxplot(df['hum'])

plt.boxplot(df['windspeed'])

plt.boxplot(df['casual'])

plt.boxplot(df['registered'])

for i in numeric\_variables:

print(i)

q75,q25 = np.percentile(df.loc[:,i],[75,25])

iqr = q75-q25

min = q25 - (iqr\*1.5)

max = q75 + (iqr\*1.5)

df.loc[df.loc[:,i]<min ,i] = np.nan

df.loc[df.loc[:,i]>max ,i] = np.nan

#Checking the number of missing values aka outliers

df.isnull().sum()

#Replacing the outliers with median value

df = df.fillna(df.median(),inplace = True)

#Feature Selection

f,ax = plt.subplots(figsize =(7,5))

#Generate correlation matrix

corr = cnames.drop(['cnt'],axis =1).corr()

#Plot using seaborne library

sns.heatmap(corr, mask =np.zeros\_like(corr, dtype = np.bool),

cmap =sns.diverging\_palette(220, 10, as\_cmap =True),

square = True, ax = ax)

#Feature selection

df = df.drop(['atemp'],axis =1)

df = df.drop(['instant'],axis =1)

df = df.drop(['dteday'],axis =1)

#Normalizing the data

a = df["casual"].max()

b = df["casual"].min()

df["casual"] = (df["casual"]- b)/(a-b)

a = df["registered"].max()

b = df["registered"].min()

df["registered"] = (df["registered"]- b)/(a-b)

#Dividing the dataset into Train and Test data

X = df.iloc[:,:-1]

Y = df.iloc[:,12]

X\_Train, X\_Test, Y\_Train, Y\_Test = train\_test\_split(X, Y, test\_size = 0.2, random\_state = 0)

#Building model using LinearRegression() method

regressor = LinearRegression()

regressor.fit(X\_Train, Y\_Train)

# Predicting the Test set results

Y\_Pred = regressor.predict(X\_Test)

print ('Variance score: %.2f' % regressor.score(X\_Test, Y\_Pred))

print("R2 score : %.2f" % sklearn.metrics.r2\_score(Y\_Test,Y\_Pred))

#Building regression model using statsmodels

X = sm.add\_constant(X) # adding a constant

model = sm.OLS(Y\_Train, X\_Train).fit()

predictions = model.predict(X\_Test)

print\_model = model.summary()

print(print\_model)

##############################################END##################################

**6.2 R- Code:**

#Clear workspace

rm(list = ls())

#Set the working directory

setwd('C:/Users/chait/Desktop/Project2')

#Load Libraries

x = c('ggplot2', 'corrgram', 'DMwR', 'caret', 'randomForest', 'unbalanced', 'C50', 'dummies',

'e1071', 'Information','MASS','rpart', 'gbm', 'ROSE')

#install.packages(x)

lapply(x, require, character.only = TRUE)

rm(x)

#Read the dataset

df = read.csv('day.csv')

#Data preprocessing

sum(is.na(df))

head(df)

str(df)

summary(df)

colnames(df)

#Converting the categorical variables to factor datatype

df[,1:9] =as.data.frame(lapply(df[,1:9],as.factor))

#Seperating the numeric data

numeric\_index = sapply(df,is.numeric)

numeric\_data = df[,numeric\_index]

cnames = colnames(numeric\_data)

cnames = cnames[1:6] #Removing the dependent variable

#Exploratory data analysis

for(i in 1:ncol(numeric\_data))

{

hist(numeric\_data[,i],main = paste("Histogram of" , cnames[i]),col='green',border = 'black',xlab = cnames[i])

}

#Scatterplot of numeric variables Vs Dependent variable

#Exploratory data analysis

for(i in 1:length(cnames))

{

x = df[,cnames[i]]

plot(x , y = df$cnt, main = paste("cnt vs ", cnames[i]),col = "blue")

}

#Smooth Scatter plot describing the relation ship between dependent and numeric variables

for(i in 1:(length(cnames)))

{

scatter.smooth(x=df[,cnames[i]],y=df$cnt,xlab = cnames[i],ylab = "cnt",col = "blue", main=paste("Smooth Scatter plot of cnt Vs ",cnames[i]))

}

#Outlier Analysis

#Boxplot of numeirc variables

for(i in 1:(length(cnames)))

{

boxplot(df[,cnames[i]],main = paste("Box Plot of ",cnames[i]),col = "green", outlier.color = "red", outcol ="red",sub = paste("Outliers: ",length(boxplot.stats(df[,cnames[i]])$out)))

}

for( i in cnames)

{

print(i)

val = df[,i][df[,i] %in% boxplot.stats(df[,i])$out]

df[,i][df[,i] %in% val] = NA

}

sum(is.na(df))

#Imputing the missing values with knnImputation

df = knnImputation(df, k=80)

#Feature Selection

library(corrplot)

library(corrgram)

M= cor(numeric\_data[,-7])

corrplot(M,method = "circle")

corrplot(M, type="upper")

#VIF

library(usdm)

#vif(df[,-16])

vifcor(numeric\_data[,-10],th = 0.9)

df1 = subset(df, select = -c(atemp,instant,dteday))

#Feature scaling

df[,i] = (df[,"casual"]- min(df[,"casual"]))/

(max(df[,"casual"] - min(df[,"casual"])))

df[,i] = (df[,"registered"]- min(df[,"registered"]))/

(max(df[,"registered"] - min(df[,"registered"])))

#Dividing Test and train datasets

train\_index <- sample(1:nrow(df1), 0.8 \* nrow(df1))

test\_index <- setdiff(1:nrow(df1), train\_index)

# Build X\_train, y\_train, X\_test, y\_test

X\_train <- df1[train\_index, -13]

y\_train <- df1[train\_index, "cnt"]

X\_test <- df1[test\_index, -13]

y\_test <- df1[test\_index, "cnt"]

df\_train = data.frame(X\_train,y\_train)

colnames(df\_test)[13] = "cnt"

#Linear Model

LinearModel = lm(cnt~., data = df\_train)

summary(LinearModel)

confint(LinearModel)

Y\_pred = predict(LinearModel,newdata = X\_test)

predict(LinearModel,type = "terms")

##############################END####################################

**References:**

* + - 1. [www.learning.edwisor.com](http://www.learning.edwisor.com)
      2. [www.analyticsvidhya.com](http://www.analyticsvidhya.com)