Predication of Daily Bike Rental Count based on

Environmental and Seasonal Settings

*Yatin Vij*

*4th November 2018*

**Contents**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| [**1**](#page3) | [**Introduction**](#page3) | | | **2** |
|  | [1.1](#page3) | [Problem Statement](#page3) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | | 2 |
|  | [1.2](#page3) | [Data](#page3) . | . . . . . . . . . . . . . . . . . . . . . . . . . .. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 2 |
| [**2**](#page5) | [**Methodology**](#page5) | | | **4** |
|  | [2.1](#page5) | [Exploratory Data Analysis](#page5). . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | | 4 |
|  |  | [2.1.1](#page5) | [Outlier Analysis](#page5) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 5 |
|  |  | [2.1.2](#page11) | [Feature Selection](#page11) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 9 |
|  |  | [2.1.](#page11)3 | [Feature Scaling.](#page11) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 10 |
|  |  |  |  |  |
|  | [2.2](#page13) | [Modeling](#page13) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | | 11 |
|  |  | [2.2.1](#page13) | [Model Selection](#page13) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 11 |
|  |  | [2.2.2](#page13) | [Multiple Linear Regression](#page13) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 11 |
|  |  | [2.2.3](#page15) | [Decision Tree Regression](#page15) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 12 |
|  |  | [2.2.4](#page15) | [Random](#page15) Forest Regression. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 12 |
|  |  | [2.2.](#page15)5 | K Nearest Neighbor Regression. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 12 |
| [**3**](#page16) | [**Conclusion**](#page16) | |  | **13** |
|  | [3.1](#page16) | [Model Evaluation](#page16) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .. . . . . . . . | | 13 |
|  | [3.2](#page17) | [Model Selection](#page17) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | | 14 |
| [**Appendix A - Extra**](#page24) **Figures** | | | | **15** |
| [**Appendix B - Python Code**](#page24) | | | | **15** |
| [**Appendix A - R Code**](#page24) | | | | **18** |
| [**References**](#page30) | | |  | **29** |

**Chapter 1**

**Introduction**

**1.1 Problem Statement**

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

**1.2 Data**

Our task is to build a prediction models which will predict the bike rental count based on the environmental and seasonal settings. Given below is a sample of the data set that we are using for building the prediction model:

Table 1.1: Sample Data (Columns: 1-8)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| S. No. | instant | dteday | season | yr | mnth | holiday | weekday | workingday |
| 1 | 1 | 1/1/2011 | 1 | 0 | 1 | 0 | 6 | 0 |
| 2 | 2 | 1/2/2011 | 1 | 0 | 1 | 0 | 0 | 0 |
| 3 | 3 | 1/3/2011 | 1 | 0 | 1 | 0 | 1 | 1 |
| 4 | 4 | 1/4/2011 | 1 | 0 | 1 | 0 | 2 | 1 |

Table 1.2: Sample Data (Columns: 8-16)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| S. No | Weathersit | temp | atemp | hum | windspeed | casual | registered | cnt |
| 1 | 2 | 0.344167 | 0.363625 | 0.805833 | 0.160446 | 331 | 654 | 985 |
| 2 | 2 | 0.363478 | 0.353739 | 0.696087 | 0.248539 | 131 | 670 | 801 |
| 3 | 1 | 0.196364 | 0.189405 | 0.437273 | 0.248309 | 120 | 1229 | 1349 |
| 4 | 1 | 0.2 | 0.212122 | 0.590435 | 0.160296 | 108 | 1454 | 1562 |

As you can see in the table below we have the following 15 variables, using which we can use to correctly predict the count of daily bike rental:

Table 1.3: Predictor Variables

S.No. Predictor

* instant

2 dteday

3 season

4 yr

5 mnth

* holiday

7 weekday

8 workingday

9 weathersit

1. temp
2. atemp
3. hum
4. windspeed
5. casual
6. registered

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

* instant: Record index
* dteday: Date
* season: Season (1: spring, 2: summer, 3: fall, 4: winter)
* yr: Year (0: 2011, 1: 2012)
* mnth: Month (1 to 12)
* hr: Hour (0 to 23)
* 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.
* atemp: Normalized feeling temperature in Celsius.
* 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

**Chapter 2**

**Methodology**

**2.1** **Exploratory Data Analysis:**

This phase is perhaps the most time consuming of all the phases as it deals with the following tasks:

* Exploring the data and all the variables that are present.
* Cleaning the data
* Visualizing the data

We have already explored the data we are dealing with in the above section where we listed all the variables we have and what they mean in the dataset. Let’s move forward with visualizing all the variables present in the data. As we have a mixture Numerical as well as Categorical data, we will be using different methods to visualize each. As we have to predict the count of daily bike rent, therefore the Target Variable will be Count.

Moreover, the variables Date and Instant are just being used to keep account of Date of the Day and the Record Number respectively which will not be contributing to the analysis so we remove them as well.

Now we can move forward with the visualization of the variables.

To start, let’s look at Fig 2.1 probability distribution functions of the numerical variables i.e. Temp, Actual Temp, Humidity, WindSpeed, Casual, Registered, and Count.

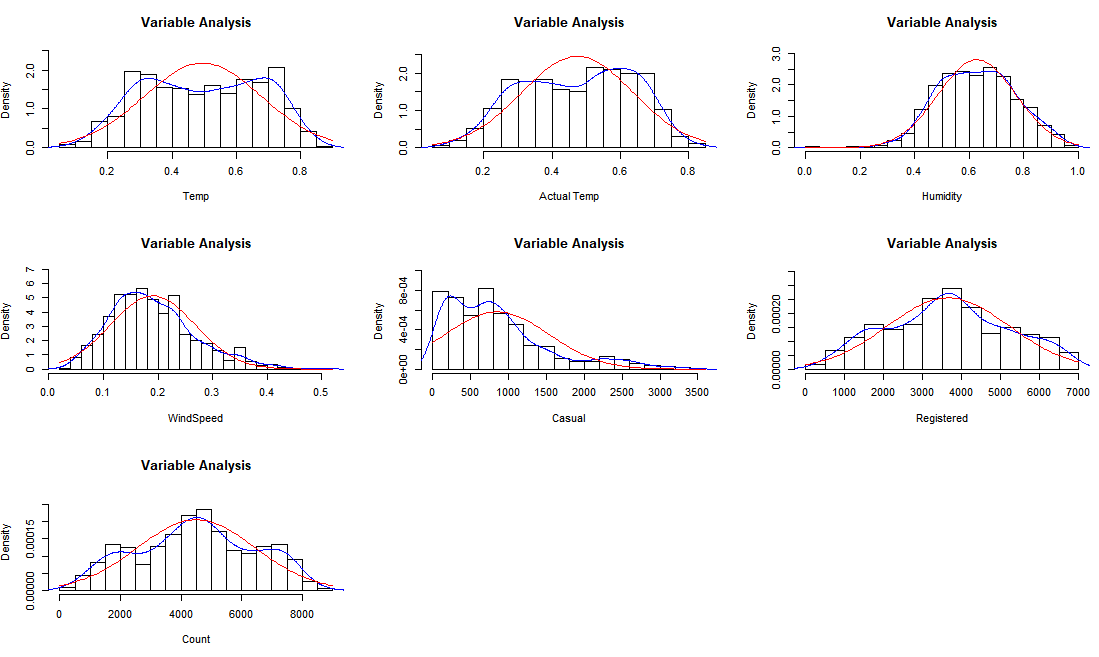
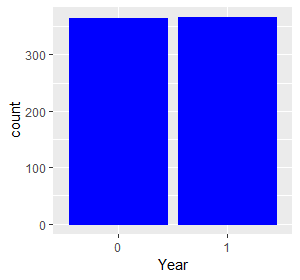
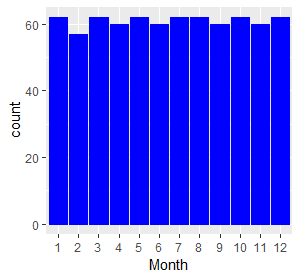
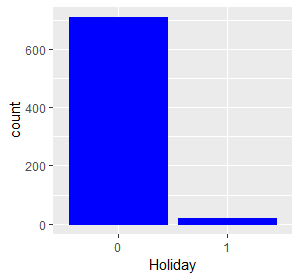
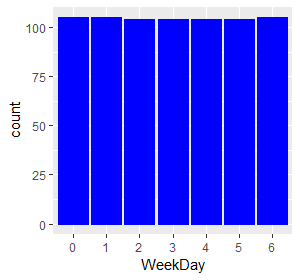
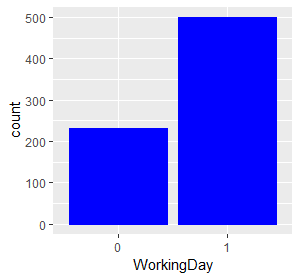
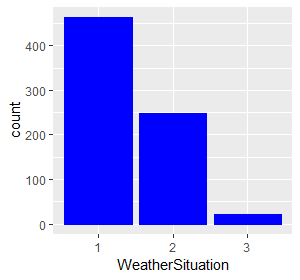


Fig 2.1 Probability Distribution Plot for all the numeric Variables

Now let’s visualize the categorical variables using a bar plot of each variable w.r.t. frequency of each class of the variable.



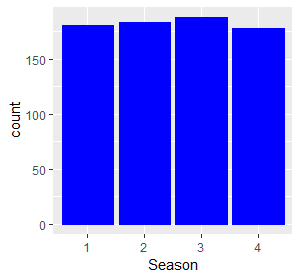


Fig 2.2 Bar Plot of Frequency Count for all classes of the Categorical Variables

**2.1.1 Outlier Analysis**

As we can see, the variables Humidity, Casual, and WindSpeed are skewed. This is most likely due to the presence of Outliers in the variables. To visualize the effect of skew, let’s look at Fig 2.3 the Histogram, Median, and Mean plot of these variables.

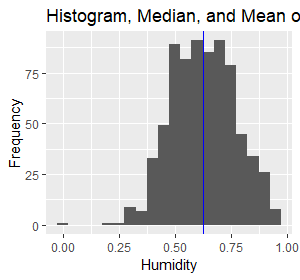
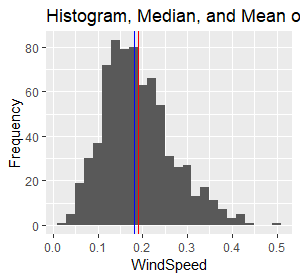
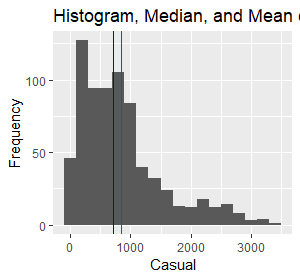
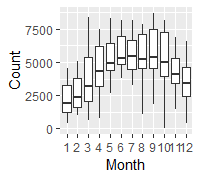
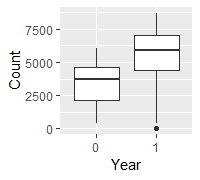
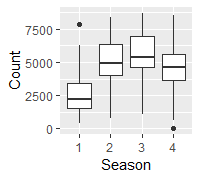


Fig 2.3 Histogram, Median and Mean of Predictor Variables

In Fig 2.3, the red line represents the mean and the blue line represents the median. As we can see, the mean is slightly displaced from the position of the median due to the presence of outliers in these variables.

To visualize the outliers, let’s construct the boxplots of all the variables:

For the categorical variables (Fig 2.3.1), the boxplot is constructed with categorical variable on the x-axis and target variable, Count, on the y-axis.



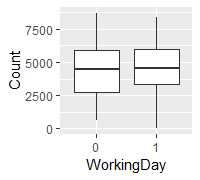
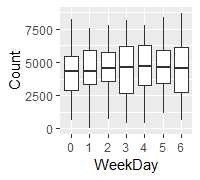
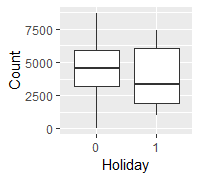
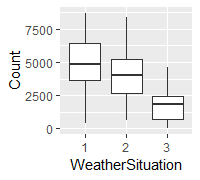


Fig 2.3.1 Boxplots for Categorical Variables w.r.t. Target Variable



As we can see from fig 2.3.1, there are outliers present only in Season and Year. The outlier count is really less as we can see there are only 2 outliers for Count w.r.t. Season and only 1 w.r.t. Year. So we will be replacing these values for Count.

For the numerical variables (Fig 2.3.2), the boxplot is constructed for all the variables with only the variable itself on the y-axis.

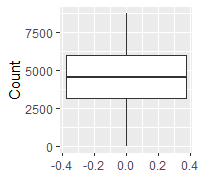
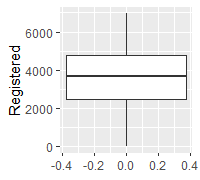
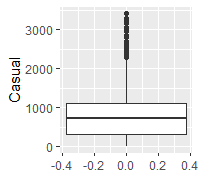
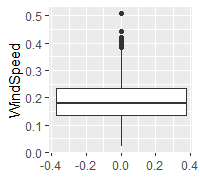
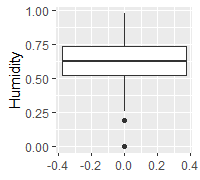
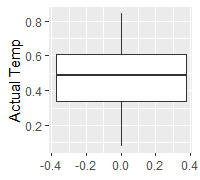
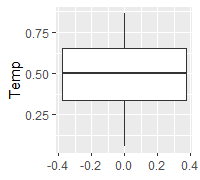
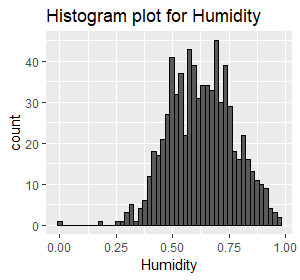
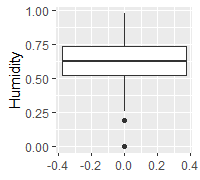
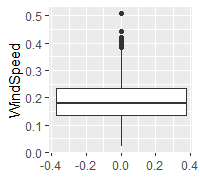
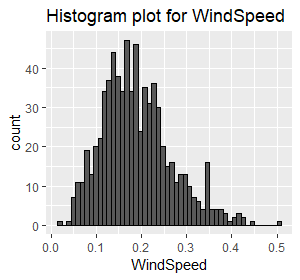


Fig 2.3.1 Boxplots for Numerical Variables

As we can see from Fig 2.3.2, the outliers are present in Humidity, Windspeed and Casual only. So we will be treating these outliers. In Fig 2.4.1, we can see the outliers and the histograms of these variables before imputing the outliers.



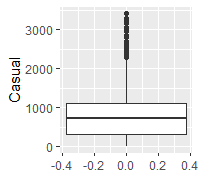
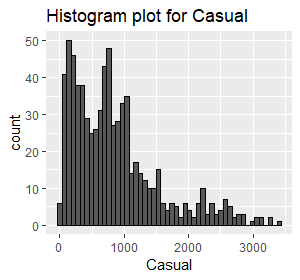
 

Fig 2.4.1 Boxplot & Histogram for Numerical Variables with outliers

After removing the outlier from the data, the histograms now look like shown in Fig 2.4.2. We get a much more normal distribution for the variables. Now we can move forward with the next step, Feature Selection.

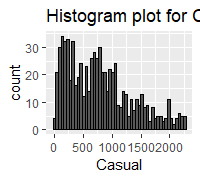
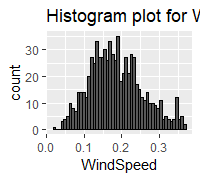
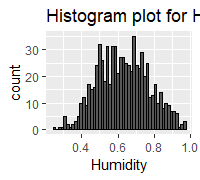


Fig 2.4.2 Histogram for Numerical Variables after removing outliers

**2.1.2 Feature Selection**

Now that we have cleaned the data, before stepping into modelling we will first identify the variables that are the most important for the analysis. We are dealing with two types variables i.e. Categorical and Numerical.

For the numerical variables, we conduct the correlation analysis to find the highly correlated variables. As shown in Fig 2.5, the variables that are highly correlated are Registered and Actual Temp. So we will be removing them from our dataframe as they individually do not add any more information for the analysis when their correlated counterparts are present.

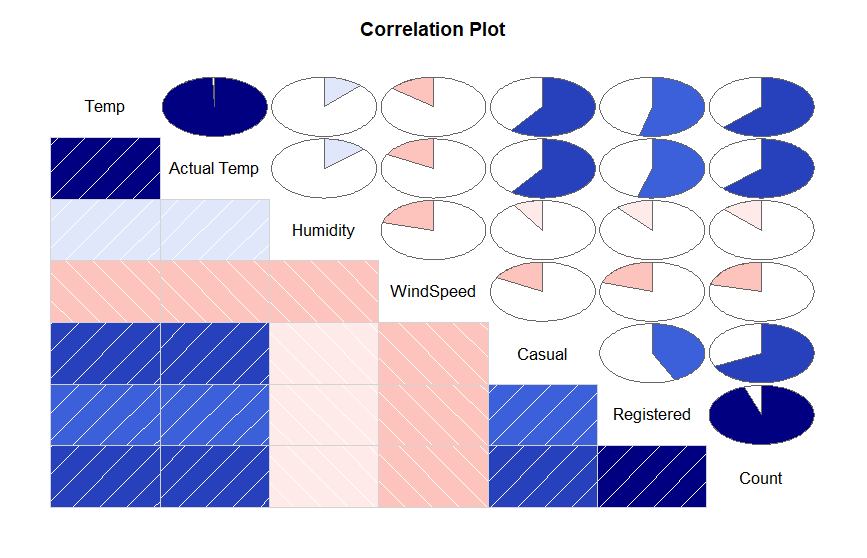


Fig 2.5 Correlation Plot for the Numerical Variables

For the categorical variables, we would conduct chi-square test but as the target variable is continuous we don’t go for the chi-square test. Instead we use randomForest’s importance estimator to find the importance of the variables. Fig 2.6 shows the result of running random forest on the data.

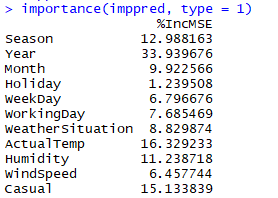


Fig 2.6 Variable Importance Estimated by Random Forest

So from the above figure we observe that Holiday category has the least importance so we can remove it from the data as well. After removing all the variables with no or low importance, now we can move to the next step i.e. Feature Scaling.

**2.1.3 Feature Scaling**

The last part of Exploratory Data Analysis is Feature Scaling. This is the most important step before pushing the data into the model for training. This is because the variables have different scales i.e. Count has values between 0-9000 whereas Temp has values between 0 and 1. So in order to make them comparable, we need to scale the numerical variables. Some models scaling built-in so we do not need to do it explicitly. Here we will be doing it, using the Standard Scalar.

The formula for Standardization is:

X=(x-u)/sd

After standardizing the numerical variables, we are now ready to step into Model Development.

**2.2 Modeling**

**2.2.1 Model Selection**

After completing the Exploratory Data Analysis and pre-processing the data, we now understand the data much better and can use this understanding to develop prediction models. In our dataset, the dependent variable, Count, is a numerical variable so we will be using the Regression analysis.

For Regression Analysis, we will be using the Multiple Linear Regression, Decision Tree Regression, and Random Forest Regression algorithms to build the model. From these models, we evaluate each using the error metrics and find the one that best fits our problem statement. Let’s start building model from the simplest to more complex. Therefore, we start with Multiple Linear Regression.

**2.2.2 Multiple Linear Regression**

As you can see in Fig 2.7, the Adjusted R-squared value, we can explain only about 90% of the data using our multiple linear regression model. After looking at the F-statistic and combined p-value we can reject the null hypothesis that target variable does not depend on any of the predictor variables.

Looking at the significance values of the predictors, we can see that each predictor is contributing at 10% significance level.

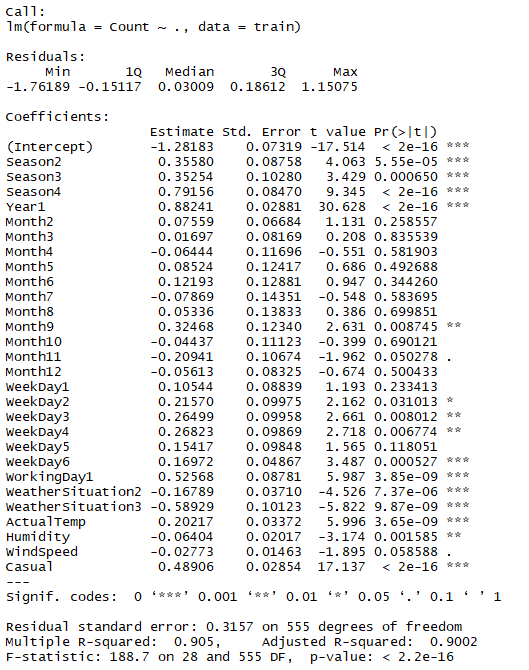
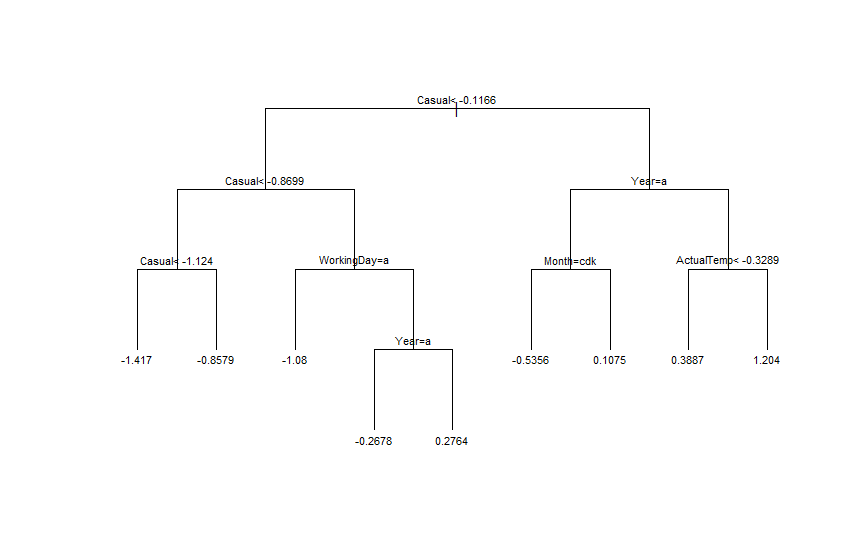


Fig 2.7 Linear Regression Model Summary

**2.2.3** **Decision Tree Regression**

Now we will try and use a Decision Tree Regression model to predict our *Count* target variable. The Fig 2.8 shows the tree formed by the Decision Tree Algorithm.

Fig 2.7 Decision Tree made using ANOVA



**2.2.4** **Random Forest Regression**

The random forest model is a type of additive model that makes predictions by combining decisions from a sequence of base models. All the base models are Decision Trees and are constructed independently using a different subsample of the data. Our Random Forest Model uses 50 decision tree estimators to create the final model that can be used for predictions.

**2.2.5** **K Nearest Neighbor Regression**

This deals with predicting the target variable’s value based on its neighbors. Our KNN Model, considers 5 neighbors for the prediction of the target variable.

**Chapter 3**

**Conclusion**

**3.1 Model Evaluation**

Now that we have a few models for predicting the target variable, we need to evaluate and compare the models. We can compare the models using any of the following criteria:

1. Predictive Performance

2. Interpretability

3. Computational Eﬃciency

Based on Interpretability, the models are ranked from most interpretable to the least as follows:

* Linear Regression
* Decision Trees
* K Nearest Neighbors
* Random Forest

Based on Computational Efficiency, the models are ranked from most efficient to the least as follows:

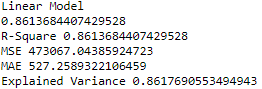
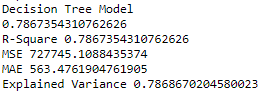
* Linear Regression
* K Nearest neighbors
* Decision Trees
* Random Forest

Predictive performance can be measured by comparing Predictions of the models with real values of the target variables, and calculating the error metrics for Regression.

The error metrics that we’ll be taking into consideration are:

* R-Square
  + R-Squared = 1 - (Explained Variation / Total Variation)
* Mean Square Error
  + the average squared difference between the estimated values and what is estimated.
* Mean Absolute Error
  + measures the average magnitude of the errors in a set of predictions, without considering their direction
* Explained Variance
  + measures the proportion to which a mathematical model accounts for the variation (dispersion) of a given data set.

The below figure shows the models and their respective error metrics.

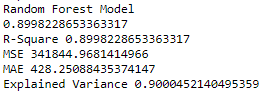
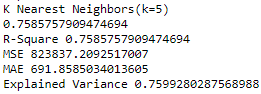
 

Fig 3.1 Models with their Error Metrics

**3.2 Model Selection**

We can see that both Linear Regression and Random Forest models perform quite well in case of R-squared values, but when it comes to MAE, MSE, and Explained Variance, the Random Forest Model outperforms the Linear Model.

Moreover, Random Forest is a bit computationally expensive when compared with Linear Model and therefore, we can should select Random Forest as it has the least error metric score and hence better accuracy.

**Appendix A - Extra Figures**

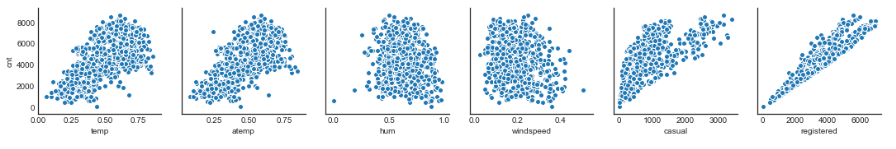


Fig 4.1 Scatter Plots for all the Numerical Variables w.r.t. Count

**Appendix B - Python Code**

import os

import pandas as pd

import numpy as np

import fancyimpute

import seaborn as sns

from sklearn.preprocessing import StandardScaler

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LinearRegression

from sklearn.tree import DecisionTreeRegressor

from sklearn.ensemble import RandomForestRegressor

from sklearn.neighbors import KNeighborsRegressor

from sklearn.metrics import (explained\_variance\_score, mean\_absolute\_error, mean\_squared\_error)

from sklearn.metrics import r2\_score

def load\_data():

# define path of csv

raw\_data\_path=os.path.join(os.path.pardir,'day.csv')

# import the dataset

df=pd.read\_csv(raw\_data\_path)

df.columns=['Instant', 'Date', 'Season', 'Year', 'Month', 'Holiday', 'Weekday',

'WorkingDay', 'WeatherSituation', 'Temp', 'ActualTemp', 'Humidity', 'WindSpeed',

'Casual', 'Registered', 'Count']

# Convert the categorical columns to object

cat\_cols=['Season', 'Year', 'Month', 'Holiday', 'Weekday',

'WorkingDay', 'WeatherSituation']

for i in cat\_cols:

df[i] = df[i].astype(object)

num\_cols=[]

for i in df.columns:

if(df[i].dtype==np.dtype('int64') or df[i].dtype==np.dtype('float64')):

num\_cols.append(i)

df[i] = df[i].astype(np.float64)

# viewing the dataframe's info

df.info()

return df

def outlier\_imputer(df\_o):

# Outlier Analysis

while True:

for i in num\_cols:

median=np.median(df\_o[i])

std=np.std(df\_o[i])

min=(df\_o[i].quantile(0.25)-1.5\*(df\_o[i].quantile(0.75)-df\_o[i].quantile(0.25)))

max=(df\_o[i].quantile(0.75)+1.5\*(df\_o[i].quantile(0.75)-df\_o[i].quantile(0.25)))

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

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

missing\_val = df\_o.isnull().sum()

print(missing\_val)

if(missing\_val.sum()>0):

df\_o\_knn=pd.DataFrame(fancyimpute.KNN(k = 3).complete(df\_o[num\_cols]), columns = num\_cols)

df\_o.iloc[:,9:]=df\_o\_knn.iloc[:,1:]

else:

break

return df\_o

def feature\_selection(df):

#Set the width and hieght of the plot

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

#Generate correlation matrix

corr = df.iloc[:,9:].corr()

#Plot using seaborn 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)

return df.drop(['Instant','Date','Holiday','Temp','Registered'],axis=1,inplace=False)

def split\_dataset(df):

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

y=df.iloc[:,-1].values

return train\_test\_split(X,y,test\_size=0.2,random\_state=123)

def feature\_scaling(X\_train,X\_test):

standardScaler=StandardScaler()

X\_train[:,6:]=standardScaler.fit\_transform(X\_train[:,6:])

X\_test[:,6:]=standardScaler.transform(X\_test[:,6:])

return X\_train,X\_test,standardScaler

def train\_lm(X\_train,y\_train):

lr\_model=LinearRegression()

lr\_model.fit(X\_train,y\_train)

return lr\_model

def train\_dt(X\_train,y\_train):

dtr\_model=DecisionTreeRegressor(random\_state=123)

dtr\_model.fit(X\_train,y\_train)

return dtr\_model

def train\_rf(X\_train,y\_train):

rf\_model=RandomForestRegressor(n\_estimators=50,random\_state=123)

rf\_model.fit(X\_train,y\_train)

return rf\_model

def train\_knn(X\_train,y\_train):

rf\_model=KNeighborsRegressor(n\_neighbors=5)

rf\_model.fit(X\_train,y\_train)

return rf\_model

def predict\_vals(model,X\_test,y\_test):

print(model.score(X\_test,y\_test))

preds=model.predict(X\_test)

return preds

def evaluate\_model(y\_test,y\_pred):

print('R-Square',r2\_score(y\_test,y\_pred))

print('MSE',mean\_squared\_error(y\_test,y\_pred))

print('MAE',mean\_absolute\_error(y\_test,y\_pred))

print('Explained Variance',explained\_variance\_score(y\_test,y\_pred))

def predict(model,X):

df=X

df.columns=['Instant', 'Date', 'Season', 'Year', 'Month', 'Holiday', 'Weekday',

'WorkingDay', 'WeatherSituation', 'Temp', 'ActualTemp', 'Humidity', 'WindSpeed',

'Casual', 'Registered']

# Convert the categorical columns to object

cat\_cols=['Season', 'Year', 'Month', 'Holiday', 'Weekday',

'WorkingDay', 'WeatherSituation']

for i in cat\_cols:

df[i] = df[i].astype(object)

num\_cols=[]

for i in df.columns:

if(df[i].dtype==np.dtype('int64') or df[i].dtype==np.dtype('float64')):

num\_cols.append(i)

df[i] = df[i].astype(np.float64)

X.drop(['Instant','Date','Holiday','Temp','Registered'],axis=1,inplace=True)

X=X.values

X[:,6:]=standardScaler.transform(X[:,6:])

return model.predict(X)

df=load\_data()

df=outlier\_imputer(df)

df=feature\_selection(df)

X\_train,X\_test,y\_train,y\_test=split\_dataset(df)

X\_train\_scaled,X\_test\_scaled,standardScaler=feature\_scaling(X\_train,X\_test)

lr\_model=train\_lm(X\_train,y\_train)

dt\_model=train\_dt(X\_train,y\_train)

rf\_model=train\_rf(X\_train,y\_train)

knn\_model=train\_knn(X\_train,y\_train)

print('Linear Model')

y\_pred=predict\_vals(lr\_model,X\_test,y\_test)

evaluate\_model(y\_test,y\_pred)

print('Decision Tree Model')

y\_pred=predict\_vals(dt\_model,X\_test,y\_test)

evaluate\_model(y\_test,y\_pred)

print('Random Forest Model')

y\_pred=predict\_vals(rf\_model,X\_test,y\_test)

evaluate\_model(y\_test,y\_pred)

print('K Nearest Neighbors(k=5)')

y\_pred=predict\_vals(knn\_model,X\_test,y\_test)

evaluate\_model(y\_test,y\_pred)

**Appendix C - R Code**

# Lets start with loading the data

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

# Lets see the structure of the data

str(df)

# Convert the Variables that are factors but were taken as numeric by R

df$season=as.factor(df$season)

df$yr=as.factor(df$yr)

df$mnth=as.factor(df$mnth)

df$holiday=as.factor(df$holiday)

df$weekday=as.factor(df$weekday)

df$workingday=as.factor(df$workingday)

df$weathersit=as.factor(df$weathersit)

df$dteday=as.character.Date(df$dteday)

# Giving the columns meaningful Names

colnames(df)=c('Instant','Date','Season','Year','Month','Holiday','WeekDay','WorkingDay','WeatherSituation','Temp','ActualTemp','Humidity','WindSpeed','Casual','Registered','Count')

# Lets view the structure again

str(df)

# Lets get a quick summary of the variables in the DataFrame

summary(df)

# remove columns that are of no use

df$Instant=NULL

df$Date=NULL

cat\_cols=NULL

num\_cols=NULL

for(i in 1:ncol(df)){

if(is.factor(df[,i])){

cat\_cols=c(cat\_cols,i)

}

else{

num\_cols=c(num\_cols,i)

}

}

#df$Casual=NULL

#df$Registered=NULL

# Lets step into visualization

# For Numerical Variables, we will be using the multi.hist fucntion to visualize

# All variables in one go. The plots will contain each variable's histogram,

# KDE plot, and a line representing Normal Distribution for comparison.

# Import package from library

library(psych)

library(ggplot2)

# Plot the multi.hist for all numeric variables in DF

multi.hist(df[,8:ncol(df)],dcol =c('blue','red'), dlty = c('solid','solid'),main = 'Variable Analysis' )

# Now for the factors, lets plot bar graph to see the count of each class

init=ggplot(data=df)

for(i in cat\_cols){

plot=init+geom\_bar(aes(x=df[,colnames(df[,i,drop=F])]),fill='blue',colour='blue')+xlab(colnames(df[,i,drop=F]))+

ylab('Frequency')

print(plot)

}

# Now lets only see the distributions of the skewed variables with their mean and median.

lh <- 10:12

bw <- c(0.05, 0.02, 200)

for (i in 1:3) {

plot <- init +

geom\_histogram(aes(x = df[,lh[i]]), binwidth = bw[i])+

geom\_vline(aes(xintercept = mean(df[,lh[i]])), color = "red")+

geom\_vline(aes(xintercept = median(df[,lh[i]])),color='blue')+

xlab(colnames(df)[lh[i]])+

ylab("Frequency")+

ggtitle(paste("Histogram, Median, and Mean of ",colnames(df)[lh[i]], ""))

print(plot)

}

# Now let's plot the boxplots to identify outliers from the variables

for(i in num\_cols){

print( init+

geom\_boxplot(aes(y=df[,colnames(df[,i,drop=F])]))+

ylab(colnames(df[,i,drop=F]))+

ggtitle(paste("Box plot for",colnames(df[,i,drop=F]))))

}

for(i in c(10,11,12)){

print( init+

geom\_histogram(aes(x=df[,colnames(df[,i,drop=F])]),bins =50,color='black')+

xlab(colnames(df[,i,drop=F]))+

ggtitle(paste("Histogram plot for",colnames(df[,i,drop=F]))))

}

#install.packages('DMwR2')

library(DMwR2)

outlierImputer=function(df,num\_cols){

while (TRUE) {

tot\_miss=NULL

for(i in num\_cols){

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

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

tot\_miss=c(tot\_miss,length(val))

}

print(sum(tot\_miss))

if(sum(tot\_miss)>0){

df=knnImputation(df,k = 3)

}

else{

break

}

}

return(df)

}

df=outlierImputer(df,num\_cols)

# Lets step into feature selection

# Finding highly correlated numerical features with Count

# install.packages('corrgram')

library(corrgram)

numFeatureSel=function(df,num\_cols){

corr=cor(df[,num\_cols])

corrgram(df[,num\_cols],order = F,upper.panel = panel.pie,text.panel = panel.txt,main='Correlation Plot')

}

numFeatureSel(df,num\_cols)

# based on correlation plot we remove the highly correlated variables

df=subset(df,select = -c(Temp,Registered))

numFeatureSel(df,8:12)

library(randomForest)

imppred=randomForest(Count ~ ., data = df,ntree = 100, keep.forest = FALSE, importance = TRUE)

importance(imppred, type = 1)

df=subset(df,select = -c(Holiday))

# Feature Scaling

#Standardisation

#for(i in 7:10){

# df[,i] = (df[,i] - mean(df[,i]))/sd(df[,i])

#}

df\_scaled = scale(df[,7:10,drop=F])

df[,7:10]=df\_scaled

# Modeling the Data

# Divide data into train and test using stratified sampling method

library(caTools)

set.seed(101)

sample = sample.split(df$Count, SplitRatio = .80)

train = subset(df, sample == TRUE)

test = subset(df, sample == FALSE)

#Develop Model on training data

# Multiple Linear Regression Model

lr\_model=lm(Count~.,train)

y\_pred=predict(lr\_model,test[,1:10])

#Summary of Linear model

summary(lr\_model)

r=sum((y\_pred-test[,11])^2)/sum((test[,11]-mean(test[,11]))^2)

1-r #-> 0.792

# Decision Tree Regression Model

# install.packages('rpart')

library(rpart)

dt\_model=rpart(Count~.,train,method = 'anova')

y\_pred\_dt=predict(dt\_model,test[,-11])

r=sum((y\_pred\_dt-test[,11])^2)/sum((test[,11]-mean(test[,11]))^2)

1-r #-> 0.792

summary(dt\_model)

plot(dt\_model, uniform = T, branch = 1, margin = 0.05, cex = 0.9)

text(dt\_model, cex = 0.7)

# Random Forest Model

RF\_model = randomForest(Count ~ ., train, importance = TRUE, ntree = 100)

library(inTrees)

treeList = RF2List(RF\_model)

exec = extractRules(treeList, train[,-11])

exec[1:6,]

readableRules = presentRules(exec, colnames(train))

readableRules[1:10,]

ruleMetric = getRuleMetric(exec, train[,-11], train$Count) # get rule metrics

ruleMetric[1:2,]

y\_pred\_rf = predict(RF\_model, test[,-11])

rt\_sample=getTree(RF\_model,k = 2,labelVar = T)

# KNN Regression Model

library(FNN)

knn\_model=knn.reg(train = train[,7:10],test=test[,7:10],y=train[,11],k=3)

y\_pred\_knn=knn\_model$pred

# Error Metrics

library(DMwR)

#calculate MAPE

MAPE = function(y, yhat){

mean(abs((y - yhat)/y))

}

MAPE(test[,11],y\_pred\_dt)

MAPE(test[,11],y\_pred)

MAPE(test[,11],y\_pred\_rf)

regr.eval(trues = test[,11],preds = y\_pred,train.y = train[,11],stats = c('mae','mse','rmse','mape','nmse','nmae'))

regr.eval(trues = test[,11],preds = y\_pred\_rf,train.y = train[,11],stats = c('mae','mse','rmse','mape','nmse','nmae'))

regr.eval(trues = test[,11],preds = y\_pred\_dt,train.y = train[,11],stats = c('mae','mse','rmse','mape','nmse','nmae'))

regr.eval(trues = test[,11],preds = y\_pred\_knn,train.y = train[,11],stats = c('mae','mse','rmse','mape','nmse','nmae'))

predict\_y=function(model,df){

# Convert the Variables that are factors but were taken as numeric by R

df$season=as.factor(df$season)

df$yr=as.factor(df$yr)

df$mnth=as.factor(df$mnth)

df$holiday=as.factor(df$holiday)

df$weekday=as.factor(df$weekday)

df$workingday=as.factor(df$workingday)

df$weathersit=as.factor(df$weathersit)

df$dteday=as.character.Date(df$dteday)

# Giving the columns meaningful Names

colnames(df)=c('Instant','Date','Season','Year','Month','Holiday','WeekDay','WorkingDay','WeatherSituation','Temp','ActualTemp','Humidity','WindSpeed','Casual','Registered')

X=subset(df,select = -c(Instant,Date,Registered,Temp,Holiday))

X[,7:10] = scale(X[,7:10], center=attr(df\_scaled, "scaled:center"),

scale=attr(df\_scaled, "scaled:scale"))

return(predict(model, X))

}

**References**

Mohd Zubair 2016, ‘Predicting Wine Quality Hamza’

<https://www.statmethods.net/>

<https://www.rdocumentation.org/>

<https://www.kaggle.com/>