Bike Rental Prediction

**Contents:**

1. **Introduction**
   1. **Problem Statement**
   2. **Data**
2. **Methodology**
   1. **Pre Processing**
      1. **Missing Value**
      2. **Plots for all the statistical parameters**
   2. **Modelling**
      1. **Decision Tree**
      2. **Linear Regression**
      3. **Random Forests**

**3 Conclusion**

**3.1 Model Evaluation**

**3.1.1 Accuracy/MAPE**

**Appendix A – Python code**

**Appendix B – R Code**

**Chapter 1**

**Introduction**

**1.1 Problem Statement**

In this project , we are going to forecast the bike rental count daily based on the environmental and seasonal settings.

**1.2 Data**

We are going to predict the bike renta count using the following data . This data will be divided into trainand test later when we are on the verge of developing a model.

**Train Data**

|  | **season** | **yr** | **mnth** | **holiday** | **weekday** | **workingday** | **weathersit** | **temp** | **atemp** | **hum** | **windspeed** | **casual** | **registered** | **cnt** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **count** | 731.000000 | 731.000000 | 731.000000 | 731.000000 | 731.000000 | 731.000000 | 731.000000 | 731.000000 | 731.000000 | 731.000000 | 731.000000 | 731.000000 | 731.000000 | 731.000000 |
| **mean** | 2.496580 | 0.500684 | 6.519836 | 0.028728 | 2.997264 | 0.683995 | 1.395349 | 0.495385 | 0.474354 | 0.627894 | 0.190486 | 848.176471 | 3656.172367 | 4504.348837 |
| **std** | 1.110807 | 0.500342 | 3.451913 | 0.167155 | 2.004787 | 0.465233 | 0.544894 | 0.183051 | 0.162961 | 0.142429 | 0.077498 | 686.622488 | 1560.256377 | 1937.211452 |
| **min** | 1.000000 | 0.000000 | 1.000000 | 0.000000 | 0.000000 | 0.000000 | 1.000000 | 0.059130 | 0.079070 | 0.000000 | 0.022392 | 2.000000 | 20.000000 | 22.000000 |
| **25%** | 2.000000 | 0.000000 | 4.000000 | 0.000000 | 1.000000 | 0.000000 | 1.000000 | 0.337083 | 0.337842 | 0.520000 | 0.134950 | 315.500000 | 2497.000000 | 3152.000000 |
| **50%** | 3.000000 | 1.000000 | 7.000000 | 0.000000 | 3.000000 | 1.000000 | 1.000000 | 0.498333 | 0.486733 | 0.626667 | 0.180975 | 713.000000 | 3662.000000 | 4548.000000 |
| **75%** | 3.000000 | 1.000000 | 10.000000 | 0.000000 | 5.000000 | 1.000000 | 2.000000 | 0.655417 | 0.608602 | 0.730209 | 0.233214 | 1096.000000 | 4776.500000 | 5956.000000 |
| **max** | 4.000000 | 1.000000 | 12.000000 | 1.000000 | 6.000000 | 1.000000 | 3.000000 | 0.861667 | 0.840896 | 0.972500 | 0.507463 | 3410.000000 | 6946.000000 | 8714.000000 |

**Chapter 2 :**

**2.1 Pre processing:**

Any predictive modeling requires that we look at the data before we start modeling. However, in data mining terms *looking at data* refers to so much more than just looking. Looking at data refers to exploring the data, cleaning the data as well as visualizing the data through graphs and plots. This is often called as **Exploratory Data Analysis**.

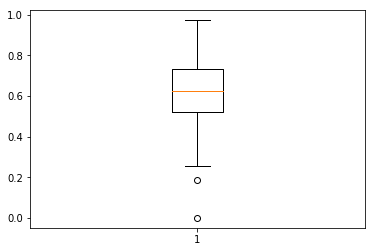
1. We check for null values in the data

In the above data theer were no missing values found.

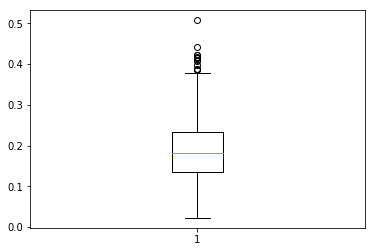
1. Do the outlier analysis for numerical data .

In the above date we found that , hum has a negative outlier and windspeed has a possitive outlier . We did the analysis using the box plot and we have set the quartile for the outlier t be removed from the data.

**Outlier for hum**

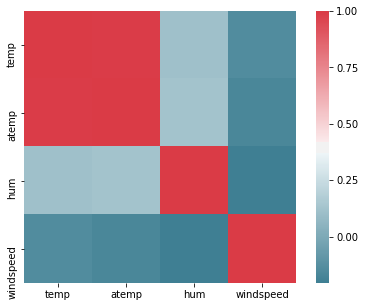


**Outlier for windspeed:**

****

**Feature Engineering:**

**After converting the data into the required datatype, we check the collinearity of the data using a correlation matrix.**

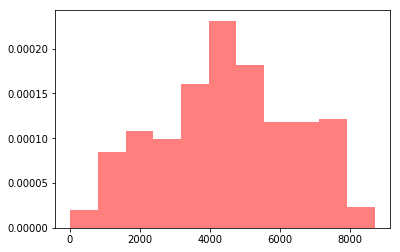
****

**From the above matrix , we know that atemp ishighly correlated . Hence we ignore this variable .**

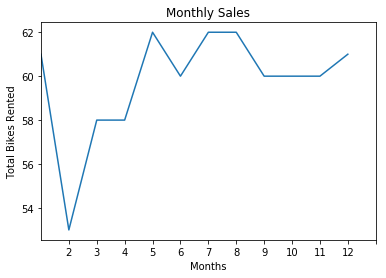
**We also ignore , eekday and holiday as they don’t contribute much on the data,**

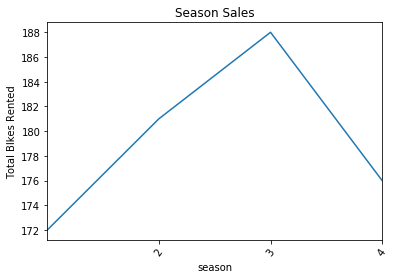
**Removing casual and registered which is our target variable.**

**Checking the distribution of target variable**

****

**Checking the bike rentals monthly sales and season sales.**

****

****

**2.2.1> Model Selection:**

As this is a prediction problem as the output is to predict the bike rental count based on weather and season .We will be using Linear regression model and test the mapr and accuracy of the model.

In this since we don’t have a separate test dat a, we split the data into train and test using a simple rule that 80% of the data wil be in training and 20% will be the test data.

We are using Linear regression , DT and Randon forest in python and DT and Random forest in R.

Result of Linear regression ( python) : Accuracy: 81.26 % , RMSE: 907.79

Result of Decision Tree (python) : Accuracy: 81.06 % , RMSE: 1016.24

Result of RF (python) : Accuracy: 85.87 %. RMSE: 666.99

As we see that RF has performed well that the other models , we will go with RF however, we can still reduce the RMSE by adding up more parameters for analysis .

Result of DT ( R):

**Appendix A**

# Loading Libraries

import numpy as np

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

import warnings

warnings.filterwarnings('ignore')

import datetime

from random import randrange, uniform

from sklearn import tree

from sklearn.metrics import accuracy\_score

from sklearn.cross\_validation import train\_test\_split

from sklearn.linear\_model import LinearRegression

from sklearn import linear\_model

from sklearn.metrics import mean\_squared\_error, r2\_score

from math import sqrt

from sklearn.tree import DecisionTreeRegressor

from sklearn.ensemble import RandomForestRegressor

bike\_rental\_data=pd.read\_csv("C:/Users/AnushaSanthosh/Desktop/day.csv",index\_col = 0)

**#understanding of data**

bike\_rental\_data.shape

#It contains (731, 16)

bike\_rental\_data.describe()

**#df\_day.info()**

**#data consist of Integers , Float and Object(categorical) variables**

**#Calculating the null values in the dataframe**

missing\_value = pd.DataFrame(bike\_rental\_data.isnull().sum())

missing\_value = (missing\_value/len(bike\_rental\_data))\*100

missing\_value.reset\_index()

missing\_value = missing\_value.rename(columns = {'index': 'Variables', 0: 'Missing\_percentage'})

**#Arranging Missing Values in Decreasing Order**

missing\_value = missing\_value.sort\_values('Missing\_percentage', ascending = False)

**#save output results**

missing\_value.to\_csv("Missing\_perc.csv", index = False)

missing\_value

**##There is no missing value in the dataframe**

**# outlier for hum**

sns.set(style="whitegrid")

sns.boxplot(x=bike\_rental\_data["hum"])

**# outlier for windspeed**

sns.set(style="whitegrid")

sns.boxplot(x=bike\_rental\_data["windspeed"])

**# saving numerical and categorical variables**

cnames = ["dteday","yr","season","mnth","workingday","weekday","weathersit","temp","atemp","hum","windspeed"]

pnames = ["temp","hum","windspeed"]

**#Detect & Delete Outliers**

for i in pnames :

print (i)

q75,q25 = np.percentile(bike\_rental\_data.loc[:,i],[75,25])

iqr = q75-q25

min = q25 - (iqr\*1.5)

max = q75 + (iqr\*1.5)

print (min)

print (max)

bike\_rental\_data = bike\_rental\_data.drop(bike\_rental\_data[bike\_rental\_data.loc[:,i] < min].index)

bike\_rental\_data = bike\_rental\_data.drop(bike\_rental\_data[bike\_rental\_data.loc[:,i] > max].index)

**#Converting respective variables to required data format**

bike\_rental\_data['dteday'] = pd.to\_datetime(bike\_rental\_data['dteday'],yearfirst=True)

bike\_rental\_data['season'] = bike\_rental\_data['season'].astype('category')

bike\_rental\_data['yr'] = bike\_rental\_data['yr'].astype('category')

bike\_rental\_data['mnth'] = bike\_rental\_data['mnth'].astype('category')

bike\_rental\_data['holiday'] = bike\_rental\_data['holiday'].astype('category')

bike\_rental\_data['weekday'] = bike\_rental\_data['weekday'].astype('category')

bike\_rental\_data['workingday'] = bike\_rental\_data['workingday'].astype('category')

bike\_rental\_data['weathersit'] = bike\_rental\_data['weathersit'].astype('category')

bike\_rental\_data['temp'] = bike\_rental\_data['temp'].astype('float')

bike\_rental\_data['atemp'] = bike\_rental\_data['atemp'].astype('float')

bike\_rental\_data['hum'] = bike\_rental\_data['hum'].astype('float')

bike\_rental\_data['windspeed'] = bike\_rental\_data['windspeed'].astype('float')

bike\_rental\_data['casual'] = bike\_rental\_data['casual'].astype('float')

bike\_rental\_data['registered'] = bike\_rental\_data['registered'].astype('float')

bike\_rental\_data['cnt'] = bike\_rental\_data['cnt'].astype('float')

**##Feature selection o the basis of various features like correlation, multicollinearity.**

**#Correlation Plot**

df\_corr = bike\_rental\_data.loc[:,cnames]

#Set the width and hieght of the plot

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

**#Generate correlation matrix**

corr = df\_corr.corr()

**#Plot using seaborn library**

sns.heatmap(corr, mask=np.zeros\_like(corr, dtype=np.bool), cmap=sns.diverging\_palette(220, 10, as\_cmap=True),

square=True, ax=ax)

**#Chi Square Test of Independence**

**#Saving Categorical Numbers**

cat\_names = ["season","yr","mnth","holiday","weekday","workingday","weathersit"]

from scipy.stats import chi2\_contingency

for i in cat\_names:

print(i)

chi2, p, dof, ex = chi2\_contingency(pd.crosstab(bike\_rental\_data['cnt'], bike\_rental\_data[i]))

print(dof)#Removing variables atemp beacuse it is highly correlated with temp,

**#Removing weekday,holiday because they don;t contribute much to the independent cariable**

**#Removing Causal and registered becuase that's what we need to predict.**

bike\_rental\_data = bike\_rental\_data.drop(['atemp','holiday','workingday','casual','registered'] ,axis=1)

**#Distribution of cnt**

**#%matplotlib inline**

num\_bins = 11

plt.hist(bike\_rental\_data['cnt'], num\_bins, normed=1, facecolor='red', alpha=0.5)

plt.show()

**#Bike Rentals Monthly**

sales\_by\_month = bike\_rental\_data.groupby('mnth').size()

print(sales\_by\_month)

**#Plotting the Graph**

plot\_by\_month = sales\_by\_month.plot(title='Monthly Sales',xticks=(1,2,3,4,5,6,7,8,9,10,11,12))

plot\_by\_month.set\_xlabel('Months')

plot\_by\_month.set\_ylabel('Total Bikes Rented')

**#Sales by Season**

sales\_by\_weekday = bike\_rental\_data.groupby('season').size()

plot\_by\_day = sales\_by\_weekday.plot(title='Season Sales',xticks=(range(1,4)),rot=55)

plot\_by\_day.set\_xlabel('season')

plot\_by\_day.set\_ylabel('Total BIkes Rented')

**# Model Building**

#Divide data into train and test

X = bike\_rental\_data.values[:,1:9]

Y = bike\_rental\_data.values[:,9]

X\_train,y\_train,X\_test,y\_test = train\_test\_split( X, Y, test\_size = 0.2)

lr\_model = linear\_model.LinearRegression()

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

y\_pred = lr\_model.predict(y\_train)

errors=abs(y\_pred-y\_test)

errors=abs(y\_pred-y\_test)

**# Calculate mean absolute percentage error (MAPE)**

mape = 100 \* (errors / y\_test)

**# Calculate and display accuracy**

accuracy = 100 - np.mean(mape)

print('Accuracy:', round(accuracy, 2), '%.')

print('RMSE: %.2f' % sqrt(mean\_squared\_error(y\_test, y\_pred)))

tree = DecisionTreeRegressor().fit(X\_train,X\_test)

prediction=tree.predict(y\_train)

errors=abs(prediction-y\_test)

**# Calculate mean absolute percentage error (MAPE)**

mape = 100 \* (errors / y\_test)

**# Calculate and display accuracy**

accuracy = 100 - np.mean(mape)

print('Accuracy:', round(accuracy, 2), '%.')

print('RMSE: %.2f' % sqrt(mean\_squared\_error(y\_test, prediction)))

**#RF\_model = RandomForestRegressor(n\_estimators = 100).fit(X\_train, y\_train)**

RF\_model = RandomForestRegressor(n\_estimators = 1000, random\_state = 1337)

**# Train the model on training data**

RF\_model.fit(X\_train, X\_test);

**# Use the forest's predict method on the test data**

predictions = RF\_model.predict(y\_train)

**# Calculate the absolute errors**

errors = abs(predictions - y\_test)

**# Calculate mean absolute percentage error (MAPE)**

mape = 100 \* (errors / y\_test)

**# Calculate and display accuracy**

accuracy = 100 - np.mean(mape)

print('Accuracy:', round(accuracy, 2), '%.')

print('RMSE: %.2f' % sqrt(mean\_squared\_error(y\_test, predictions)))

result = RF\_model.predict(y\_train)

**Appendix B:**

**Code in R :**

rm(list=ls(all=T)) **# clearing the R environment**

x = c("ggplot2", "corrgram", "DMwR", "caret", "randomForest", "unbalanced", "C50", "dummies", "e1071", "Information",

"MASS", "rpart", "gbm", "ROSE", 'sampling', 'DataCombine', 'inTrees','fastDummies') **# loading the required libraries**

install.packages("fastDummies" , repos="http://cran.rstudio.com/")

install.packages("corrgram" , repos="http://cran.rstudio.com/")

install.packages("inTrees" ,repos="http://cran.rstudio.com/")

install.packages("DataCombine",repos="http://cran.rstudio.com/")

install.packages("sampling",repos="http://cran.rstudio.com/")

install.packages("Information",repos="http://cran.rstudio.com/")

install.packages("dummies",repos="http://cran.rstudio.com/")

install.packages("C50",repos="http://cran.rstudio.com/")

install.packages("unbalanced",repos="http://cran.rstudio.com/")

install.packages("randomForest",repos="http://cran.rstudio.com/")

install.packages("corrgram" , repos="http://cran.rstudio.com/")

install.packages("ggplot2",repos="http://cran.rstudio.com/")

install.packages("DMwR",repos="http://cran.rstudio.com/")

install.packages("caret",repos="http://cran.rstudio.com/")

install.packages("dummies",repos="http://cran.rstudio.com/")

install.packages("inTrees",repos="http://cran.rstudio.com/")

install.packages("sampling",repos="http://cran.rstudio.com/")

install.packages("rpart" ,repos="http://cran.rstudio.com/")

install.packages("MASS",repos="http://cran.rstudio.com/")

rm(x)

**# loading dataset**

bike = read.csv("C:/Users/AnushaSanthosh/Desktop/day.csv", header = T, na.strings = c(" ", "", "NA"))

bike\_train=bike

**# plotting histogram of variables.**

bike\_train$season=as.numeric(bike\_train$season)

bike\_train$mnth=as.numeric(bike\_train$mnth)

bike\_train$yr=as.numeric(bike\_train$yr)

bike\_train$holiday=as.numeric(bike\_train$holiday)

bike\_train$weekday=as.numeric(bike\_train$weekday)

bike\_train$workingday=as.numeric(bike\_train$workingday)

bike\_train$weathersit=as.numeric(bike\_train$weathersit)

bike\_train$windspeed=as.numeric(bike\_train$windspeed)

par(mfrow=c(4,2))

par(mar = rep(2, 4))

hist(bike\_train$season)

hist(bike\_train$weather)

hist(bike\_train$holiday)

hist(bike\_train$workingday)

hist(bike\_train$temp)

hist(bike\_train$atemp)

hist(bike\_train$windspeed)

**# converting the variables into required data types**

bike\_train$season=as.numeric(bike\_train$season)

bike\_train$mnth=as.numeric(bike\_train$mnth)

bike\_train$yr=as.factor(bike\_train$yr)

bike\_train$holiday=as.factor(bike\_train$holiday)

bike\_train$weekday=as.factor(bike\_train$weekday)

bike\_train$workingday=as.factor(bike\_train$workingday)

bike\_train$weathersit=as.factor(bike\_train$weathersit)

bike\_train$windspeed=as.factor(bike\_train$windspeed)

bike\_train=subset(bike\_train,select = -c(instant,casual,registered))

d1=unique(bike\_train$dteday)

df=data.frame(d1)

bike\_train$dteday=as.Date(df$d1,format="%Y-%m-%d") # extracting date

df$d1=as.Date(df$d1,format="%Y-%m-%d")

bike\_train$dteday=format(as.Date(df$d1,format="%Y-%m-%d"), "%d")

bike\_train$dteday=as.factor(bike\_train$dteday)

str(bike\_train)

missing\_val = data.frame(apply(bike\_train,2,function(x){sum(is.na(x))})) # checking for missing values

numeric\_index = sapply(bike\_train,is.numeric) #selecting only numeric

numeric\_data = bike\_train[,numeric\_index]

cnames = colnames(numeric\_data)

**# checking for outliers using boxplot**

for (i in 1:length(cnames))

{

assign(paste0("gn",i), ggplot(aes\_string(y = (cnames[i]), x = "cnt"), data = subset(bike\_train))+

stat\_boxplot(geom = "errorbar", width = 0.5) +

geom\_boxplot(outlier.colour="red", fill = "blue" ,outlier.shape=18,

outlier.size=1, notch=FALSE) +

theme(legend.position="bottom")+

labs(y=cnames[i],x="cnt")+

ggtitle(paste("Box plot of count for",cnames[i])))

}

gridExtra::grid.arrange(gn1,gn2,ncol=3)

gridExtra::grid.arrange(gn3,gn4,ncol=2)

**# plotting correlation plot to check for multicollinearity**

corrgram(bike\_train[,numeric\_index], order = F,

upper.panel=panel.pie, text.panel=panel.txt, main = "Correlation Plot")

bike\_train = subset(bike\_train,select = -c(atemp))

rmExcept("bike\_train")

**# model selection**

**# splitting the data into traina nd test**

train\_index = sample(1:nrow(bike\_train), 0.8 \* nrow(bike\_train))

train = bike\_train[train\_index,]

test = bike\_train[-train\_index,]

train

fit = rpart(cnt ~ ., data = train)

**# using the DT model**

predictions\_DT = predict(fit, test[,-12])

**#cnames= c("dteday","season","mnth","weekday","weathersit")**

which(sapply(train, function(y) nlevels(y) > 53))

str(train)

train$windspeed<-as.numeric(train$windspeed)

test$windspeed<-as.numeric(test$windspeed)

str(train)

**# using the RF model**

RF\_model = randomForest(cnt ~ ., train, importance = TRUE, ntree = 200)

predictions\_RF = predict(RF\_model, test[,-12])

plot(RF\_model)

str(train)

**# checking the accuracy of the models**

MAPE = function(y, yhat){

mean(abs((y - yhat)/y))\*100

}

MAPE(test[,12], predictions\_DT)

MAPE(test[,12], predictions\_RF)

results <- data.frame(test, pred\_cnt = predictions\_RF)

**# submitting the most accurate model**

write.csv(results, file = 'RF output R .csv', row.names = FALSE, quote=FALSE)