Bike rental prediction

Date : 04-08-2019

Author : Pritam Sonawane

Introduction

Problem Statement:

The objective of this Case is to Predication of bike rental count on daily based on the environmental and seasonal settings. A bike rental system is a service in which users can rent/use bikes available for shared use on a short term basis. Our goal is to develop and optimize Machine Learning models that effectively predict the bike rental count on a daily basis.

Data-set :

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

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)

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. 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

The dataset consists of 731 rows and 16 column

5]:

	instant	dteday	season	yr	mnth	holiday	weekday	workingday	weathersit	temp	atemp	hum	windspeed	casual	registered	cnt
0	1	2011-01-01	1	0	1	0	6	0	2	0.344167	0.363625	0.805833	0.160446	331	654	985
1	2	2011-01-02	1	0	1	0	0	0	2	0.363478	0.353739	0.696087	0.248539	131	670	801
2	3	2011-01-03	1	0	1	0	1	1	1	0.196364	0.189405	0.437273	0.248309	120	1229	1349
3	4	2011-01-04	1	0	1	0	2	1	1	0.200000	0.212122	0.590435	0.160296	108	1454	1562
4	5	2011-01-05	1	0	1	0	3	1	1	0.226957	0.229270	0.436957	0.186900	82	1518	1600

As we can see from dataset the target variable contains continuous values so our task here is to build a regression model which will predict total count (cnt) which is bike rental count.

Methodology

Pre Processing

Data preprocessing is a data mining technique which is used to transform the raw data in a useful and efficient format.
Steps Involved in Data Preprocessing:

1. Data Cleaning:

This step is important because in most situations data provided by the customer has a bad quality or just cannot be directly fed to some kind of ML model. It includes data type conversion, data validation, handling dates, handling nominal and categorical variables. But in this case dataset variable data type is as follows:

```
'data.frame':   731 obs. of  16 variables:
 $ instant   : int  1 2 3 4 5 6 7 8 9 10 ...
 $ dteday    : chr  "2011-01-01" "2011-01-02" "2011-01-03" "2011-01-04" ...
 $ season    : int  1 1 1 1 1 1 1 1 1 1 ...
```

2

```
 $ 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 ...
```

- Here there is no need to work on data types ,but variable names are not understandable so let's replace column name with understandable name ,

'Instant':'id', 'dteday':'datetime' , 'yr':'year' , 'mnth':'month' , 'weathersit':'weather_condition' , 'hum':'humidity' , 'cnt':'total_count'

- Also I have added actual season , actual_holiday, act_weather_condition and actual_weekday column for better for visualisation of categorical variables.

2. Missing value analysis:

The concept of missing values is important to understand in order to successfully manage data. If the missing values are not handled properly then we may end up drawing an inaccurate inference about the data.

We can impute missing values by mean ,median of variable or for categorical variable we use mode.

Here there is no missing value found in data set :

```
df.isnull().sum()
```

[10]:

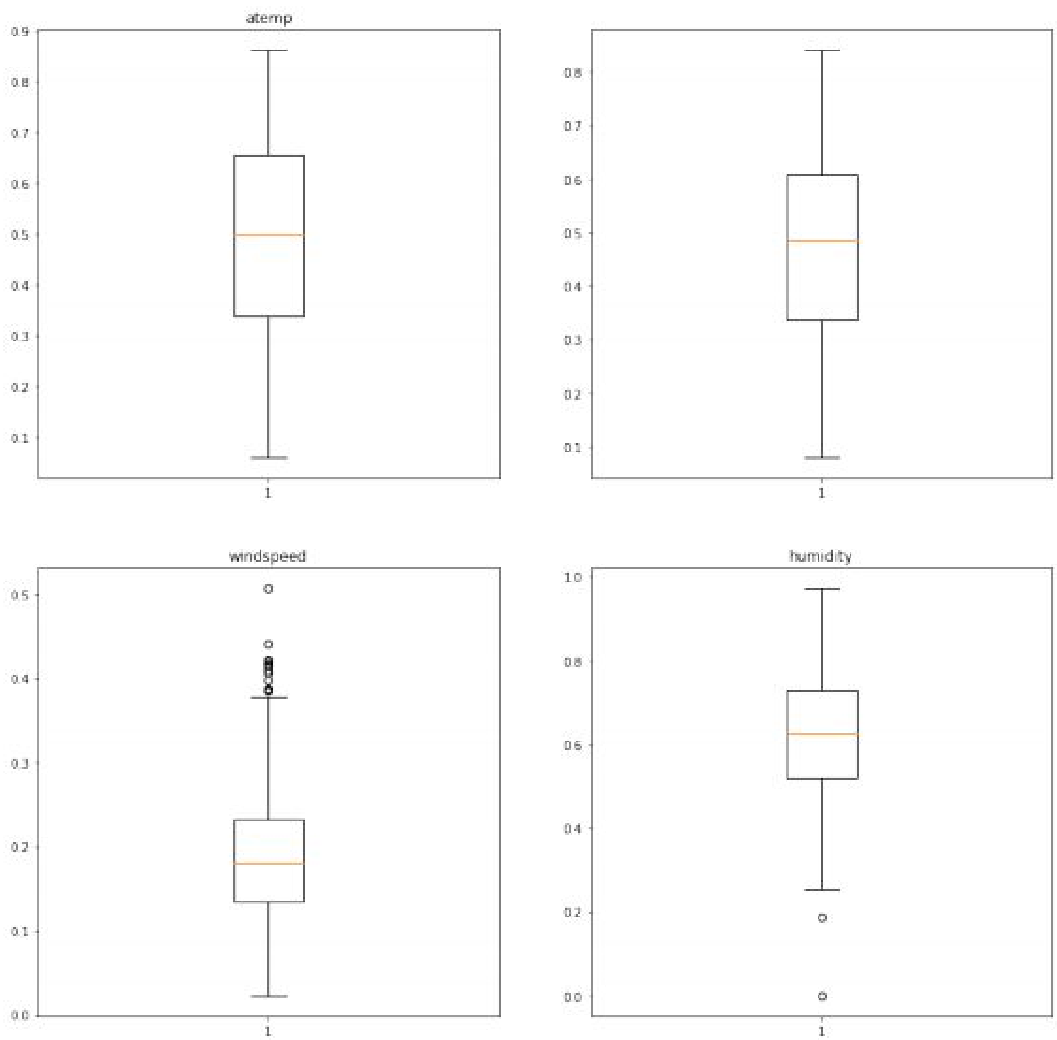
```
id                       0
datetime                 0
season                   0
year                     0
month                    0
holiday                  0
weekday                  0
workingday               0
weather_condition        0
temp                     0
atemp                    0
humidity                 0
windspeed                0
casual                   0
registered               0
total_count              0
actual_season            0
actual_holiday           0
act_weather_condition    0
actual_weekday           0
dtype: int64
```

There is no missing value present in dataset

3. Outlier analysis:

An outlier is an element of a data set that distinctly stands out from the rest of the data. It can affect the overall observation made from the data series.The easiest way to detect outliers is to create a plots such as Box plots

Boxplot for temp, total_count, wind speed , humidity are as follows:

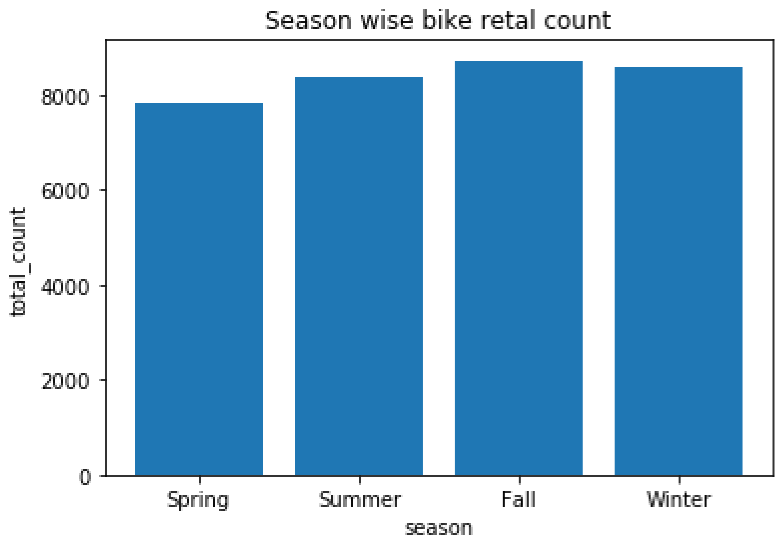


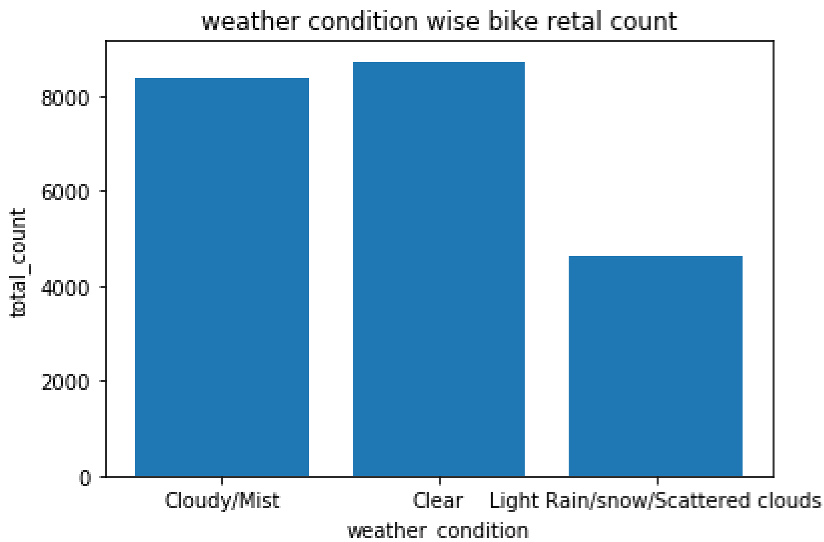
From the box plot, we can observe that no outliers are present in temp,total_count and registered variables but few outliers are present in wind-speed,and humidity variable.

The outliers are replaced with nan values and then as per null value imputation technique it is replaced with mean value.

Relationship between independent variables with dependant /target variable

(Bar graphs)





1. From the first season wise rental count bar graph we can say that bike rental count is slightly reduced in spring.
2. In the second bar graph we can say people uses rental bike more on working days than holiday.
3. From third bar graph we can say that in weather condition like Light Snow, Light Rain and Thunderstorm and Scattered clouds the bike rental demand is reduced

   Also there is no data available for fourth category of weather condition

4. Correlation analysis :

A scatterplot is used to graphically represent the relationship between two variables. Explore the relationship between scatterplots and correlations

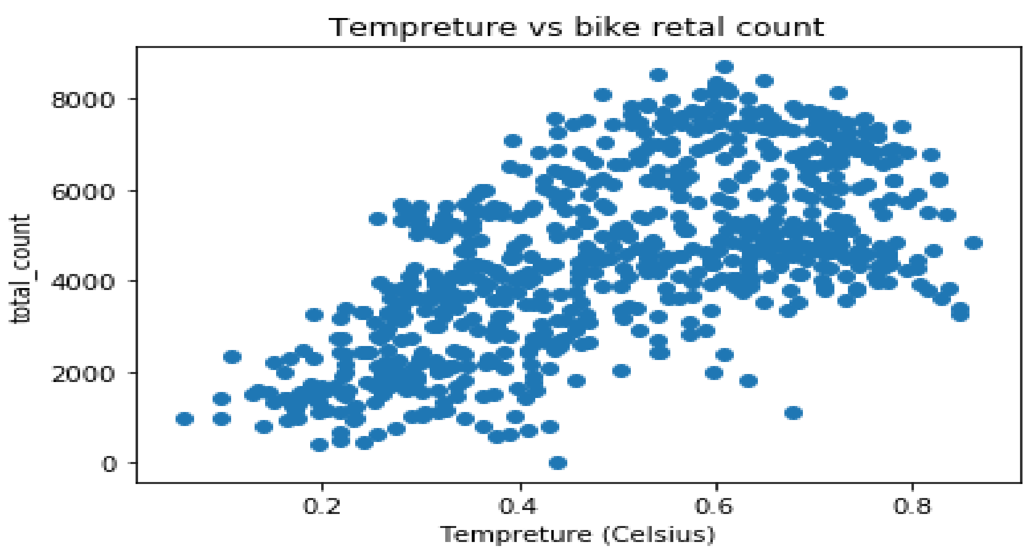
correlations have two properties: strength and direction. The **strength** of a correlation is determined by its numerical value. The **direction** of the correlation is determined by whether the correlation is positive or negative.

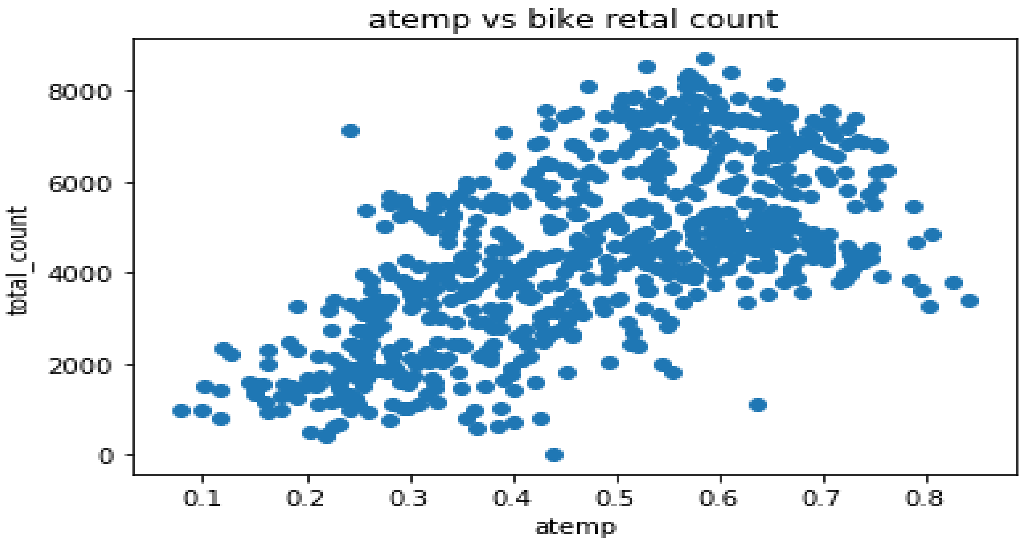
Positive correlation: Both variables move in the same direction. In other words, as one variable increases, the other variable also increases. As one variable decreases, the other variable also decreases.

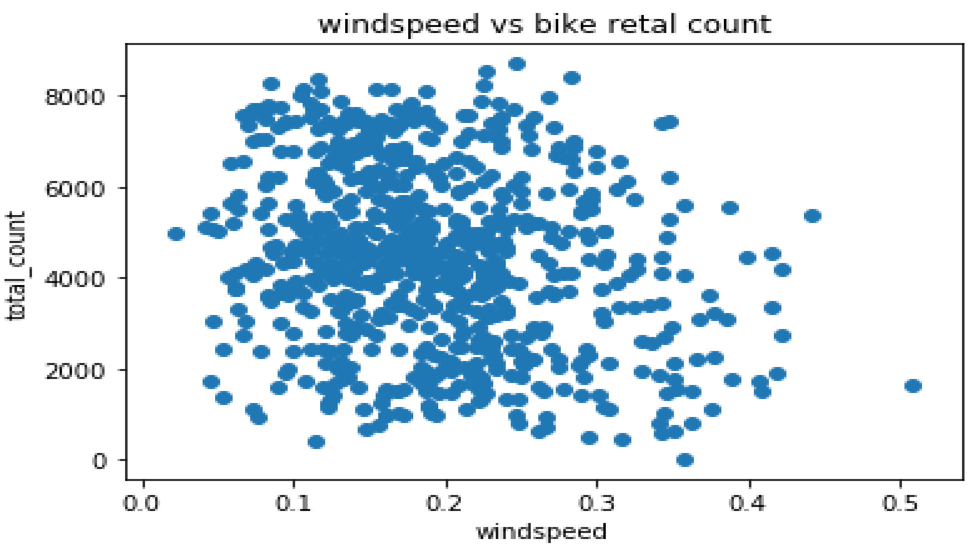
Negative correlation: The variables move in opposite directions. As one variable increases, the other variable decreases. As one variable decreases, the other variable increases.

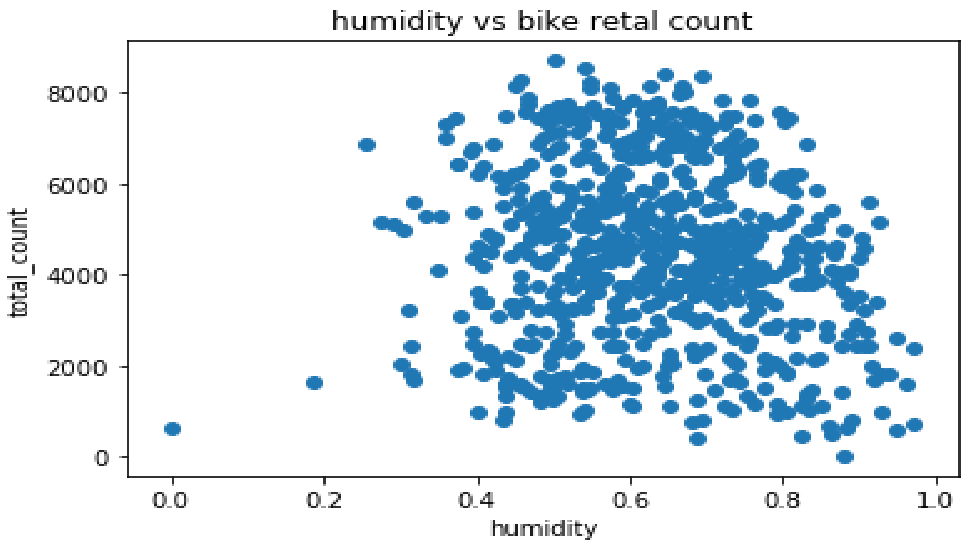
No Correlation: It means that there is no apparent relationship between the two variables.

Let's see the scatterplots to analyse correlation ,



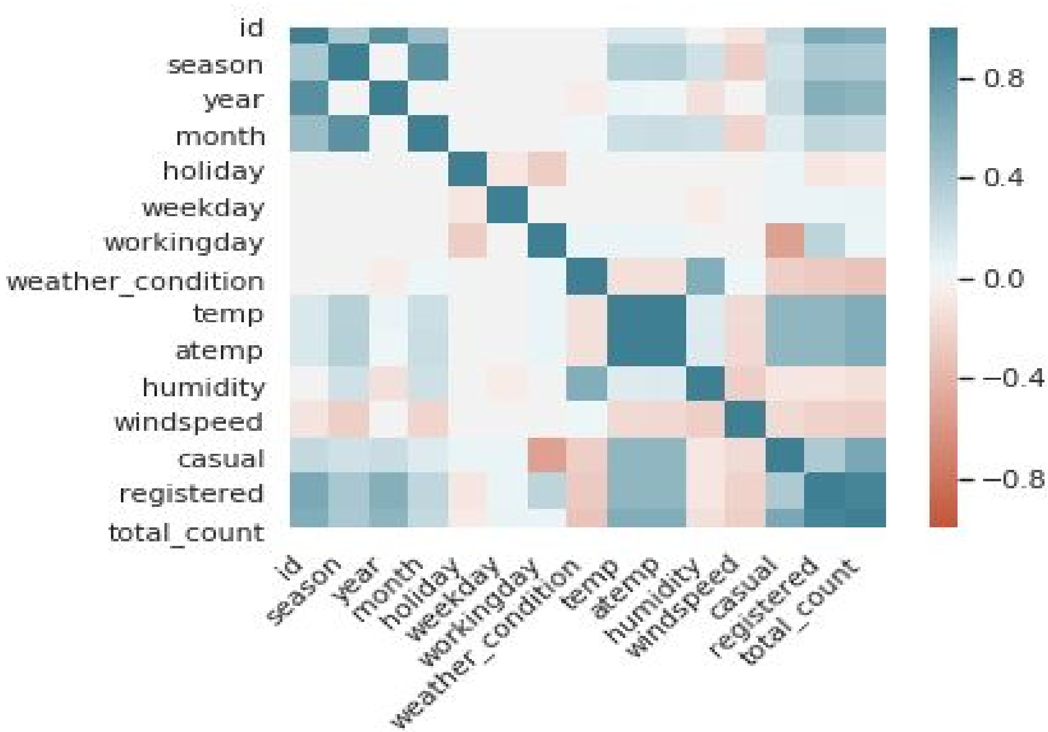

atemp vs bike retal count
8000
6000
4000
2000
0
total_count
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
atemp


windspeed vs bike retal count
8000
6000
4000
2000
0
total_count
0.0
0.1
0.2
0.3
0.4
0.5
windspeed



4. From Temperature vs bike rental scatterplots (temp vs total_count and atemp vs total_count) we can say that there is a **positive linear relationship** with temp and total count.
5. From windspeed vs bike count plot we can see there is **no correlation.**
6. From humidity vs bike count plot we can see there is a **slight negative correlation**

Lets see the strength of correlation between all variables



From above heatmap we can see the correlation strength between variables

- temp and atemp are strongly correlated as their strength as around 0.8
- If both features are included in the model, this will cause the issue of Multicollinearity. Hence we will take only one temperature feature into the model.
- The features casual and registered are removed because that is what we are going to predict.

Model Development

Model Selection :

From the earlier analysis on dataset we know that our target variable contains continuous values ,so will use regression machine learning model.

Divided data into train and test

Divided the data into 80% training and 20% testing dataset.

Training data contains 587 observations and 11 variables.

```
In [22]: ###################################Model Development#######################################

         #Divide data into train and test
         set.seed(1234)
         train.index = createDataPartition(final_bkr_data$total_count, p = .80, list = FALSE)
         train = final_bkr_data[ train.index,]
         test  = final_bkr_data[-train.index,]
         str(train)

         'data.frame':   587 obs. of  11 variables:
          $ season           : int  1 1 1 1 1 1 1 1 1 1 ...
          $ year             : int  0 0 0 0 0 0 0 0 0 0 ...
          $ month            : 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  0 1 2 4 5 6 0 1 2 5 ...
          $ workingday       : int  0 1 1 1 1 0 0 1 1 1 ...
          $ weather_condition: int  2 1 1 1 2 2 1 1 2 1 ...
          $ temp             : num  0.363 0.196 0.2 0.204 0.197 ...
          $ humidity         : num  0.696 0.437 0.59 0.518 0.499 ...
          $ windspeed        : num  0.2485 0.2483 0.1603 0.0896 0.1687 ...
          $ total_count      : int  801 1349 1562 1606 1510 959 822 1321 1263 1421 ...
```

- Divided the data into 80% training and 20% testing data
- training data consist of 587 observations with 11 variables.

Decision tree for regression

- Here our target variable having continuous values hence we have to use regression model in which by variance we decide best splits
- lower values of variance clearly leading to more pure node and high value of variance lead to impure node
- We will use variance reduction method for node splitting:
  In the anova method
  the splitting criteria is
  SST − (SSL + SSR), where SST = P(yi − y¯)2 is the sum of squares for the node, and SSR, SSL are the sums of squares for the right and left son, respectively.
  This is equivalent to choosing the split to maximize the between-groups sum-of-squares in a simple analysis of variance. This rule is identical to the regression option for tree
- rpart for regression
  rpart(formula, data=, method=,control=) where

formula : is in the format

outcome ~ predictor1+predictor2+predictor3+ect.

data : training dataset

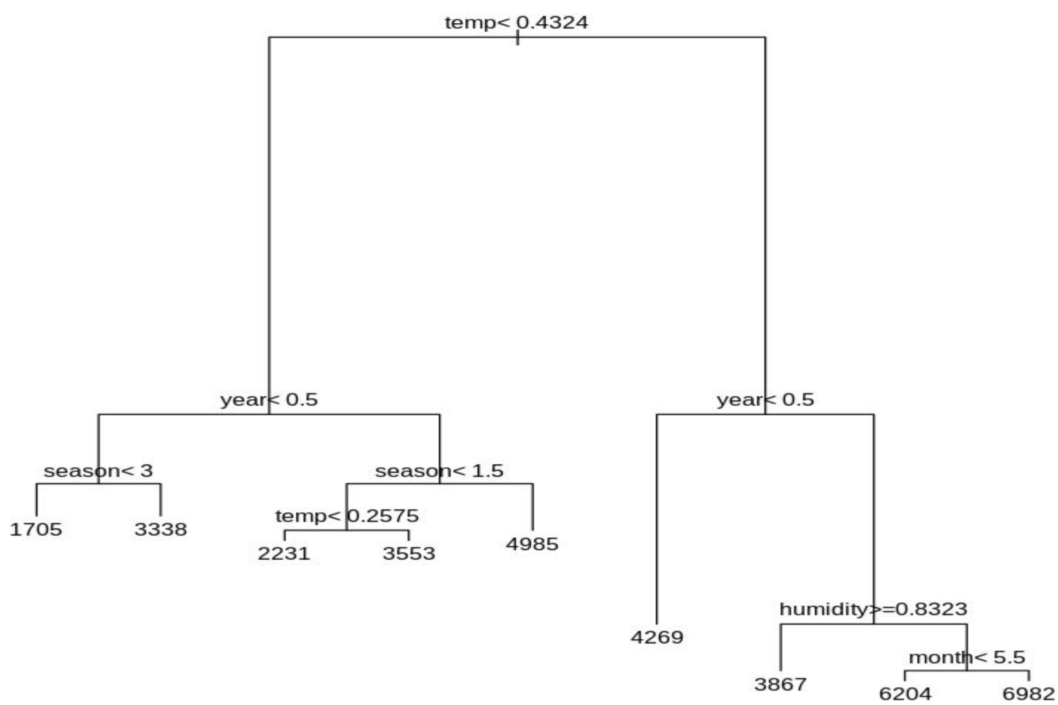
method :

"class" for a classification tree

"anova" for a regression tree

control : optional parameters for controlling tree growth.

Decision tree model visualisation



Random Forest for regression

A random forest allows us to determine the most important predictors across the explanatory variables by generating many decision trees and then ranking the variables by importance.

```
In [32]: # Number of variables randomly sampled as candidates at each split. Note that the default values are different for
         # where p is number of variables in x) and regression (p/3)
         ##############################Random Forest model##################################

         rf_2=randomForest(total_count ~ . , data = train,mtry =4,ntree=100 ,nodesize =10 ,importance =TRUE)
```

```
In [33]: predictions_RF2 = predict(rf_2, test[,-11])
```

Number of variables randomly sampled as candidates at each split.

From above

mtry :

Number of variables randomly sampled as candidates at each split. Note that the default values are different for classification (sqrt(p) where p is number of variables in x) and regression (p/3)

ntree : Number of trees to grow.

nodesize : Minimum size of terminal nodes. Setting this number larger causes smaller trees to be grown (and thus take less time). Note that the default values are different for classification (1) and regression (5).

Linear Regression

Regression is a parametric technique used to predict continuous (dependent) variable given a set of independent variables. Mathematically, regression uses a linear function to approximate (predict) the dependent variable given as: Y = βo + β1X + ∈ where, Y - Dependent variable X - Independent variable βo - Intercept β1 - Slope ∈ - Error

- βo and β1 are known as coefficients. This is the equation of simple linear regression.
- Error is an inevitable part of the prediction-making process. No matter how powerful the algorithm we choose, there will always remain an (∈) irreducible error The formula to calculate coefficients goes like this: β1 = Σ(xi - xmean)(yi-ymean)/ Σ (xi - xmean)^2 where i= 1 to n (no. of obs.)

  βo = ymean - β1(xmean)

In [86]:
```
##################################################Regression Analysis############################################
#the base function lm is used for regression.
regmodel <- lm(total_count ~ ., data = train)
summary(regmodel)
```

```
Call:
lm(formula = total_count ~ ., data = train)

Residuals:
    Min      1Q  Median      3Q     Max 
-4148.0  -454.5    41.6   542.4  3104.0 

Coefficients:
                  Estimate Std. Error t value Pr(>|t|)    
(Intercept)        1593.50     253.71   6.281 6.65e-10 ***
season              547.82      59.75   9.169  < 2e-16 ***
year               2066.45      72.30  28.581  < 2e-16 ***
month               -36.49      18.79  -1.942  0.05258 .  
holiday            -655.13     233.84  -2.802  0.00526 ** 
weekday              70.51      17.84   3.952 8.73e-05 ***
workingday          132.48      78.94   1.678  0.09383 .  
weather_condition  -637.24      88.03  -7.239 1.46e-12 ***
temp               5068.96     212.95  23.803  < 2e-16 ***
humidity          -1143.12     353.56  -3.233  0.00129 ** 
windspeed         -2322.85     536.55  -4.329 1.76e-05 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 864.8 on 576 degrees of freedom
Multiple R-squared:  0.8074,    Adjusted R-squared:  0.804 
F-statistic: 241.4 on 10 and 576 DF,  p-value: < 2.2e-16
```

- Intercept - This is the βo value. It's the prediction made by model when all the independent variables are set to zero.
- Estimate - This represents regression coefficients for respective variables.
- Std. Error - This determines the level of variability associated with the estimates.
- t value - t statistic is generally used to determine variable significance, i.e.
  if a variable is significantly adding information to the model.
- t value > 2 suggests the variable is significant.
- p value - It's the probability value of respective variables determining their significance in the model.
  p value < 0.05 is always desirable.

  The adjusted R^2 implies that our model explains ~80.4% total variance in the data.

Model Evaluation :

Mean Absolute Error (MAE) and Root mean squared error (RMSE) are two of the most common metrics used to measure accuracy for continuous variables.

Mean Absolute Error (MAE): MAE measures the average magnitude of the errors in a set of predictions, without considering their direction. It's the average over the test sample of the absolute differences between prediction and actual observation where all individual differences have equal weight.

MAE = 1/n $\sum_{i-0}^{n} yi - yj$

Root mean squared error (RMSE): RMSE is a quadratic scoring rule that also measures the average magnitude of the error. It's the square root of the average of squared differences between prediction and actual observation.

RMSE = $\sqrt{1/n \sum_{i=0}^{n}(yi - yj)}$

Multiple Linear Regression model

- mean absolute percentage error = 0.186550951376743
- accuracy = 81.3449048623257 %
- Root mean square error = 950.226605469929

Random Forest model

- mean absolute percentage error = 0.116119353398914
- accuracy = 88.3880646601086 %
- Root mean square error = 661.487858223226

Decision tree model

- mean absolute percentage error = 0.169504694884855
- accuracy = 83.0495305115145 %
- Root mean square error = 910.098855669325

By comparing Decision tree, Random Forest and Multiple Linear Regression models we can say that Random Forest model performing very well on this dataset

R-Code :

jupyter bike_rental_project (1) (1).r 6 hours ago

File Edit View Language

```r

# Load all the packages required for the analysis
library(tidyverse)
library(ggplot2) # Visualisation
install.packages("corrgram")
install.packages("caret")
library(caret)
library(corrgram)
install.packages("Metrics")
library(Metrics)
library(rpart)

#############################################Explore the data###########################################
# print dimention of the datast
dim(bike_pr_day)
# column names
names(bike_pr_day)
# datatypes of variable values
str(bike_pr_day)

## Read the data
bike_pr_day <- read.csv("./day.csv", stringsAsFactors=FALSE)

head(bike_pr_day)

##################################Missing Values Analysis##############################################
#calculate missing values present in dataset
missing_val = data.frame(apply(bike_pr_day,2,function(x){sum(is.na(x))}))
missing_val
```

```r

# rename columns of the dataset
names(bike_pr_day)<-
c('id','datetime','season','year','month','holiday','weekday','workingday','weather_condition','temp','atemp','humidity','windspeed','casual','registered','total_count')
head(bike_pr_day)

#add act_season , act_holiday, act_weathersit columns for better visualisation
bike_pr_day$act_season = factor(x = bike_pr_day$season, levels = c(1,2,3,4), labels = c("Spring","Summer","Fall","Winter"))
bike_pr_day$act_holiday = factor(x = bike_pr_day$holiday, levels = c(0,1), labels = c("Working day","Holiday"))

# 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
# we will take 1=Clear, 2=Cloudy/Mist ,3=Light Rain/snow/Scattered clouds, 4=Heavy Rain/Snow/Fog
bike_pr_day$act_weathersit = factor(x = bike_pr_day$weather_condition, levels = c(1,2,3,4),
                              labels = c("Clear","Cloudy/Mist","Light Rain/snow/Scattered clouds","Heavy Rain/Snow/Fog"))


bike_pr_day$act_weakday = factor(x = bike_pr_day$weekday, levels = c(0,1,2,3,4,5,6),
                              labels = c("Monday","Tuesday", "Wednesday","Thursday","Friday","Saturday","Sunday"))



# Bar graph of Season wise monthly distribution of counts
ggplot(bike_pr_day,aes(x=month,y=total_count,fill=act_season))+theme_bw()+geom_col()+
labs(x='Month',y='Total_Count',title='Season wise monthly distribution of counts')

```

```
# Bar graph of Season wise monthly distribution of counts
ggplot(bike_pr_day,aes(x=month,y=total_count,fill=act_season))+theme_bw()+geom_col()+
labs(x='Month',y='Total_Count',title='Season wise monthly distribution of counts')

#column plot for weekday wise monthly distribution of counts
ggplot(bike_pr_day,aes(x=month,y=total_count,fill=act_weakday))+theme_bw()+geom_col()+
labs(x='Month',y='Total_Count',title='Weekday wise monthly distribution of counts')

# Bar graph of Holiday wise monthly distribution of counts
ggplot(bike_pr_day,aes(x=month,y=total_count,fill=act_holiday))+theme_bw()+geom_col()+
labs(x='Month',y='Total_Count',title='Holiday wise monthly distribution of counts')

# Scatterplot of Distribution of Temperature
ggplot(data = bike_pr_day, aes(x =temp, y = total_count)) + ggtitle("Distribution of Temperature") + geom_point() +
xlab("Temperature") + ylab("Bike COunt")

#Scatterplot of Distribution of Humidity
ggplot(data = bike_pr_day, aes(x =humidity, y = total_count)) + ggtitle("Distribution of Humidity") + geom_point(color="red") +
xlab("Humidity") + ylab("Bike COunt")

Scatterplot of Distribution of Windspeed
ggplot(data = bike_pr_day, aes(x =windspeed, y = total_count)) + ggtitle("Distribution of Windspeed") + geom_point(color="red")
+ xlab("Windspeed") + ylab("Bike COunt")
###############################################Outlier analysis##########################################
#boxplot for total_count_outliers
par(mfrow=c(1, 1))#divide graph area in 1 columns and 1 rows
boxplot(bike_pr_day$windspeed,main='Total_count',sub=paste(boxplot.stats(bike_pr_day$windspeed)$out))

boxplot(bike_pr_day$temp,main='Total_count',sub=paste(boxplot.stats(bike_pr_day$temp)$out))

boxplot(bike_pr_day$humidity,main='Total_count',sub=paste(boxplot.stats(bike_pr_day$humidity)$out))

```

```
boxplot(bike_pr_day$total_count,main='Total_count',sub=paste(boxplot.stats(bike_pr_day$total_count)$out))

###############################################Outlier value imputation##################################################

#create subset for windspeed and humidity variable
wind_hum<-subset(bike_pr_day,select=c('windspeed','humidity'))

cnames<-colnames(wind_hum)
for(i in cnames){
    val=wind_hum[,i][wind_hum[,i] %in% boxplot.stats(wind_hum[,i])$out] #outlier values
    wind_hum[,i][wind_hum[,i] %in% val]= NA  # Replace outliers with NA
}
#Imputating the outlier values using mean imputation method
wind_hum$windspeed[is.na(wind_hum$windspeed)]<-mean(wind_hum$windspeed,na.rm=T)
wind_hum$humidity[is.na(wind_hum$humidity)]<-mean(wind_hum$humidity,na.rm=T)
new_df<-subset(bike_pr_day,select=-c(windspeed,humidity))

bike_rent_df<-cbind(new_df,wind_hum)
head(bike_rent_df)

####################################Feature Selection##############################################################
numeric_index= sapply(bike_rent_df,is.numeric) #selecting only numeric
## Correlation Plot
corrgram(bike_rent_df[,numeric_index], order = F,
         upper.panel=panel.pie, text.panel=panel.txt, main = "Correlation Plot")


#Create a new subset for training model
final_bkr_data<-subset(bike_rent_df,select=c('season','year','month','holiday',
'weekday','workingday','weather_condition','temp','humidity','windspeed','total_count'))

```

```
###################################Model Development#######################################


#Divide data into train and test
set.seed(1234)
train.index = createDataPartition(final_bkr_data$total_count, p = .80, list = FALSE)
train = final_bkr_data[ train.index,]
test  = final_bkr_data[-train.index,]
str(train)

###################################Decision tree for regression###########################################
#
# lets develope decision rules for predicting a continuous (regression tree) outcome.

# ##rpart for regression


fit = rpart(total_count ~ ., data = train, method = "anova")


# decision tree model visualisation
par(cex= 0.8)
plot(fit)
text(fit)

#Predict for new test cases

predictions_DT = predict(fit, test[,-11])


```

```
###################Mean Absolute Percentage Error###########################################

error <- mape(predictions_DT,test$total_count)
error
accuracy <- (1-error)*100
accuracy


###################Root Mean square Error###########################################
rmse(actual = test$total_count,predicted = predictions_DT)



# Accuracy of decision tree is 83.0495305115145 %
###############################################################################################
# save the model to disk
saveRDS(fit, "./DT_model.rds")

# load the model
model <- readRDS("./DT_model.rds")

# make a predictions on new data using saved model
final_predictions <- predict(model, test[,-11])




# Actual value vs predicted value plot tells about variance between actual target value and predicted target value
plot(test$total_count,final_predictions,xlab='Actual value',ylab='predicted value',main='Actual value vs predicted value plot')
abline(0,0)
```

```r
##############################################Random Forest##########################################################

library(randomForest)

rf <- randomForest(total_count ~ ., data = train, ntree=20)
predictions_RF = predict(rf, test[,-11])

###################Mean Absolute Percentage Error############################################

error <- mape(predictions_RF,test$total_count)
error
accuracy <- (1-error)*100
accuracy


###################Root Mean square Error############################################
rmse(actual = test$total_count,predicted = predictions_RF)



# Number of variables randomly sampled as candidates at each split. Note that the default values are different for classification (sqrt(p)
# where p is number of variables in x) and regression (p/3)

rf_2=randomForest(total_count ~ . , data = train,mtry =4,ntree=100 ,nodesize =10 ,importance =TRUE)

predictions_RF2 = predict(rf_2, test[,-11])

###################Mean Absolute Percentage Error############################################

error <- mape(predictions_RF2,test$total_count)
error
```

```r
error
accuracy <- (1-error)*100
accuracy


###################Root Mean square Error############################################
rmse(actual = test$total_count,predicted = predictions_RF2)



##################################################Regression Analysis##############################################
#the base function lm is used for regression.
regmodel <- lm(total_count ~ ., data = train)
summary(regmodel)


#check the residual plots, understand the pattern and derive actionable insights (if any):

par(mfrow=c(2,2))
#create residual plots
plot (regmodel)

# lets test model
regpred <- predict(regmodel, test[-11])

error <- mape(regpred,test$total_count)
error
accuracy <- (1-error)*100
accuracy
rmse(actual = test$total_count,predicted = regpred)
```

Original R code and python code is attached with this document