By Madhurima biswas

**Bike Renting - R code**

> #To clear the R environment of any predefined objects

> rm(list=ls())

> #To set working directory

> setwd("F:/DS/edWisor/Project 2")

> getwd()

[1] "F:/DS/edWisor/Project 2"

>

> #To load required libraries

> library(ggplot2) # used for ploting

Warning message:

package ‘ggplot2’ was built under R version 3.4.4

> library(dplyr) # used for data manipulation and joining

Attaching package: ‘dplyr’

The following objects are masked from ‘package:stats’:

filter, lag

The following objects are masked from ‘package:base’:

intersect, setdiff, setequal, union

Warning message:

package ‘dplyr’ was built under R version 3.4.4

> library(scales) # used for "pretty\_brakes() function"

Warning message:

package ‘scales’ was built under R version 3.4.4

> library(DMwR) # used for KNN Imputation

Loading required package: lattice

Loading required package: grid

Warning messages:

1: package ‘DMwR’ was built under R version 3.4.4

2: package ‘lattice’ was built under R version 3.4.4

> library(outliers) # used for outlier detection & modification

Warning message:

package ‘outliers’ was built under R version 3.4.4

> library(corrgram) # used for plotting correlation amongst variables

Attaching package: ‘corrgram’

The following object is masked from ‘package:lattice’:

panel.fill

Warning message:

package ‘corrgram’ was built under R version 3.4.4

> library(corrplot) # used for plotting correlation amongst variables

corrplot 0.84 loaded

Warning message:

package ‘corrplot’ was built under R version 3.4.4

> library(caret) # used for various model training

Warning message:

package ‘caret’ was built under R version 3.4.4

> library(lubridate) # used for handling date format data

Attaching package: ‘lubridate’

The following object is masked from ‘package:base’:

date

Warning message:

package ‘lubridate’ was built under R version 3.4.4

> library(FNN) # used for KNN modeling

Warning message:

package ‘FNN’ was built under R version 3.4.4

> library(randomForest) # used for Random Forest implementation

randomForest 4.6-14

Type rfNews() to see new features/changes/bug fixes.

Attaching package: ‘randomForest’

The following object is masked from ‘package:outliers’:

outlier

The following object is masked from ‘package:dplyr’:

combine

The following object is masked from ‘package:ggplot2’:

margin

Warning message:

package ‘randomForest’ was built under R version 3.4.4

> library(rpart) # used for Decision Tree algorithm implementation

Warning message:

package ‘rpart’ was built under R version 3.4.4

>

> #To load the data

> data = read.csv("day.csv",header = T, na.strings = c(""," ","NA",NA))

>

> ####################Data Exploration######################

> str(data) #"data.frame"

'data.frame': 731 obs. of 16 variables:

$ instant : int 1 2 3 4 5 6 7 8 9 10 ...

$ dteday : Factor w/ 731 levels "2011-01-01","2011-01-02",..: 1 2 3 4 5 6 7 8 9 10 ...

$ season : int 1 1 1 1 1 1 1 1 1 1 ...

$ yr : int 0 0 0 0 0 0 0 0 0 0 ...

$ mnth : int 1 1 1 1 1 1 1 1 1 1 ...

$ holiday : int 0 0 0 0 0 0 0 0 0 0 ...

$ weekday : int 6 0 1 2 3 4 5 6 0 1 ...

$ workingday: int 0 0 1 1 1 1 1 0 0 1 ...

$ weathersit: int 2 2 1 1 1 1 2 2 1 1 ...

$ temp : num 0.344 0.363 0.196 0.2 0.227 ...

$ atemp : num 0.364 0.354 0.189 0.212 0.229 ...

$ hum : num 0.806 0.696 0.437 0.59 0.437 ...

$ windspeed : num 0.16 0.249 0.248 0.16 0.187 ...

$ casual : int 331 131 120 108 82 88 148 68 54 41 ...

$ registered: int 654 670 1229 1454 1518 1518 1362 891 768 1280 ...

$ cnt : int 985 801 1349 1562 1600 1606 1510 959 822 1321 ...

> dim(data) # 731 x 16

[1] 731 16

>

> ###Univariate Analysis###

> #col = names(data)

> #To find the unique values in each column

> #for (i in col) {

> # print(i)

> # print(length(unique(data[,i])))

> #}

> #Data has 7 categorical variables, 8 numeric variables & one date type variable.

> #Target variable is integer type in nature.

>

> ###Data Consolidation###

> #Convert into Proper data types

> #-->ignoring "instant" as it is just like serial number.

> data = data[,-1]

> #dim(data) #731 x 15

>

> #\_\_\_\_\_Data type conversion\_\_\_\_\_#

> catnames = c("season","yr","mnth","holiday","weekday","workingday","weathersit") #categorical variables

> for (i in catnames) {

+ data[,i] = as.factor(data[,i])

+ }

>

> numnames = c("temp","atemp","hum","windspeed","casual","registered","cnt") #numerical variables

> for (i in numnames) {

+ data[,i] = as.numeric(data[,i])

+ }

>

> data$dteday = as.Date(data$dteday) #It changed date "02-04-11" to "2011-04-02".

>

>

> str(data)

'data.frame': 731 obs. of 15 variables:

$ dteday : Date, format: "2011-01-01" "2011-01-02" "2011-01-03" "2011-01-04" ...

$ season : Factor w/ 4 levels "1","2","3","4": 1 1 1 1 1 1 1 1 1 1 ...

$ yr : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...

$ mnth : Factor w/ 12 levels "1","2","3","4",..: 1 1 1 1 1 1 1 1 1 1 ...

$ holiday : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...

$ weekday : Factor w/ 7 levels "0","1","2","3",..: 7 1 2 3 4 5 6 7 1 2 ...

$ workingday: Factor w/ 2 levels "0","1": 1 1 2 2 2 2 2 1 1 2 ...

$ weathersit: Factor w/ 3 levels "1","2","3": 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 : num 331 131 120 108 82 88 148 68 54 41 ...

$ registered: num 654 670 1229 1454 1518 ...

$ cnt : num 985 801 1349 1562 1600 ...

> ###\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Graphical analysis\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_###

> #And so on. The graphs are plotted and recorded in the project report.

>

>

> ###To extract days from "dteday" and make a new variable

> data$day = day(data$dteday)

> #As we already have information about the year and month, we have the whole date information & can remove the "dteday" date type variable as it may not be suitable for modeling.

> data[,1] = data[,16]

> data[,16] = NULL #dim = 731 x 15

>

> col = names(data)

>

> ##################\_\_\_\_\_\_\_\_\_\_\_\_\_\_Missing Value Analysis\_\_\_\_\_\_\_\_\_\_\_\_\_\_##################

> sum(is.na(data))

[1] 0

>

> #There are no missing values for this data set.

> ####################\_\_\_\_\_\_\_\_\_\_\_\_Outlier Analysis\_\_\_\_\_\_\_\_\_\_\_\_########################

> ####Box Plot distribution & outlier check####

> str(data)

'data.frame': 731 obs. of 15 variables:

$ dteday : int 1 2 3 4 5 6 7 8 9 10 ...

$ season : Factor w/ 4 levels "1","2","3","4": 1 1 1 1 1 1 1 1 1 1 ...

$ yr : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...

$ mnth : Factor w/ 12 levels "1","2","3","4",..: 1 1 1 1 1 1 1 1 1 1 ...

$ holiday : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...

$ weekday : Factor w/ 7 levels "0","1","2","3",..: 7 1 2 3 4 5 6 7 1 2 ...

$ workingday: Factor w/ 2 levels "0","1": 1 1 2 2 2 2 2 1 1 2 ...

$ weathersit: Factor w/ 3 levels "1","2","3": 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 : num 331 131 120 108 82 88 148 68 54 41 ...

$ registered: num 654 670 1229 1454 1518 ...

$ cnt : num 985 801 1349 1562 1600 ...

> for(i in 1:length(numnames)){

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

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

+ geom\_boxplot(outlier.colour="red", fill = "light blue",outlier.shape=18,outlier.size=3, notch=FALSE) +

+ theme(legend.position="bottom")+

+ labs(y=numnames[i],x="Bike Rental Count")+

+ ggtitle(paste("Box plot for",numnames[i])))

+ }

>

> #Plotting plots together

> gridExtra::grid.arrange(gn1,gn2,gn3,gn4,ncol=4)

Warning messages:

1: Continuous x aesthetic -- did you forget aes(group=...)?

2: Continuous x aesthetic -- did you forget aes(group=...)?

3: Continuous x aesthetic -- did you forget aes(group=...)?

4: Continuous x aesthetic -- did you forget aes(group=...)?

5: Continuous x aesthetic -- did you forget aes(group=...)?

6: Continuous x aesthetic -- did you forget aes(group=...)?

7: Continuous x aesthetic -- did you forget aes(group=...)?

8: Continuous x aesthetic -- did you forget aes(group=...)?

> gridExtra::grid.arrange(gn5,gn6,gn7,ncol=3)

Warning messages:

1: Continuous x aesthetic -- did you forget aes(group=...)?

2: Continuous x aesthetic -- did you forget aes(group=...)?

3: Continuous x aesthetic -- did you forget aes(group=...)?

4: Continuous x aesthetic -- did you forget aes(group=...)?

5: Continuous x aesthetic -- did you forget aes(group=...)?

6: Continuous x aesthetic -- did you forget aes(group=...)?

>

> #To check number of outliers in data (ignoring categorical variables, checked earlier)

> out = 0.0

> for(i in numnames){

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

+ out = out + length(val)

+ print(i)

+ print(length(val))

+ }

[1] "temp"

[1] 0

[1] "atemp"

[1] 0

[1] "hum"

[1] 2

[1] "windspeed"

[1] 13

[1] "casual"

[1] 44

[1] "registered"

[1] 0

[1] "cnt"

[1] 0

> out #= 59. Total Outliers in the data set is 59.

[1] 59

> #(59/731)\*100 = 8.07% of data.

> ##To test for the best method to find missing values for this dataset

> #data[12,12] #data[12,12] = 0.304627 (actual)

> #data[12,12]= NA

> #By median method:

> #data$windspeed[is.na(data$windspeed)]=median(data$windspeed, na.rm = T)

> #data[12,12] #data[12,12] = 0.180971 (median)

>

> #reupload data

> #data[12,12] #data[12,12] = 0.304627 (actual)

> #data[12,12]= NA

> #by mean method:

> #data$windspeed[is.na(data$windspeed)]=mean(data$windspeed, na.rm = T)

> #data[12,12] #data[12,12] = 0.1903299 (mean)

>

> #reupload data

> #data[12,12] #data[12,12] = 0.304627 (actual)

> #data[12,12]= NA

> #By KNN imputation method:

> #(KNN takes only numeric inputs)

> #for (i in col) {

> # data[,i] = as.numeric(data[,i])

> #}

> #data= knnImputation(data, k=3) #For k=5,7,9, the difference was even more than k=3.

> #data[12,12] #data[12,12] = 0.2324425 (KNN)

> #We freeze NA imputation by MEDIAN method as it is closest to actual value.

>

> #reupload data

> #Converting outliers to NAs

> #Select variables with outliers

> Out\_Var = c('hum','windspeed','casual') #Variables with outliers

>

> for(i in Out\_Var){

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

+ data[,i][data[,i] %in% val] = NA

+ }

> sum(is.na(data)) #To verify

[1] 59

>

> data= knnImputation(data, k=3)

>

> sum(is.na(data)) #To verify

[1] 0

> #Confirm again if any outlier exists

> out = 0.0

> for(i in numnames){

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

+ out= out + length(val)

+ print(i)

+ print(length(val))

+ }

[1] "temp"

[1] 0

[1] "atemp"

[1] 0

[1] "hum"

[1] 0

[1] "windspeed"

[1] 2

[1] "casual"

[1] 1

[1] "registered"

[1] 0

[1] "cnt"

[1] 0

> out #= 3. Windspeed has 2 outliers & Casual has 1 outlier.

[1] 3

> for(i in Out\_Var){

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

+ data[,i][data[,i] %in% val] = NA

+ }

> sum(is.na(data)) #To verify

[1] 3

>

> data= knnImputation(data, k=3)

>

> sum(is.na(data)) #To verify

[1] 0

> #Confirm again if any outlier exists

> out = 0.0

> for(i in numnames){

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

+ out= out + length(val)

+ print(i)

+ print(length(val))

+ }

[1] "temp"

[1] 0

[1] "atemp"

[1] 0

[1] "hum"

[1] 0

[1] "windspeed"

[1] 1

[1] "casual"

[1] 0

[1] "registered"

[1] 0

[1] "cnt"

[1] 0

> out

[1] 1

> for(i in Out\_Var){

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

+ data[,i][data[,i] %in% val] = NA

+ }

> sum(is.na(data)) #To verify

[1] 1

>

> data= knnImputation(data, k=3)

>

> sum(is.na(data)) #To verify

[1] 0

> #Confirm again if any outlier exists

> out = 0.0

> for(i in numnames){

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

+ out= out + length(val)

+ print(i)

+ print(length(val))

+ }

[1] "temp"

[1] 0

[1] "atemp"

[1] 0

[1] "hum"

[1] 0

[1] "windspeed"

[1] 0

[1] "casual"

[1] 0

[1] "registered"

[1] 0

[1] "cnt"

[1] 0

> out

[1] 0

> write.csv(data, 'data\_without Outliers.csv', row.names = F)

|  |
| --- |
| > #To load the data  > #data = read.csv("data\_without Outliers.csv",header = T)  >  > ############################Feature Selection#############################  > #Correlation Plot  > corrgram(data, order = F,  + upper.panel=panel.pie, text.panel=panel.txt, main = "Correlation Plot", font.labels = 1)  > #cor(x), x must be numeric  > #Convert all columns to numeric type  > #for (i in col) {  > # data[,i] = as.numeric(data[,i])  > #} #NOTE: This changes all zero factor levels to numeric 1. so, "0" --> 1.  > #mat = cor(data)  > #corrplot(as.matrix(mat),method= 'pie',type = "lower", tl.col = "black", tl.cex = 0.7)  >  > #If |r|>0.8, those two variables are redundant variables.  > #Output: "mnth"-"season", "temp"-"atemp" & "cnt"-"registered" are highly positively correlated.  >  > #redo data conversion to proper types  > catnames = c("season","yr","mnth","holiday","weekday","workingday","weathersit") #categorical variables  > for (i in catnames) {  + data[,i] = as.factor(data[,i])  + }  >  > numnames = c("dteday","temp","atemp","hum","windspeed","casual","registered","cnt") #numerical variables  > for (i in numnames) {  + data[,i] = as.numeric(data[,i])  + }  >  > #######Chi-square Test of Independence (within Categorical Variables)  > for(i in catnames){  + for(j in catnames){  + if(i!=j){  + print(names(data[i]))  + print(paste0(" Vs ", names(data[j])))  + print(chisq.test(table(data[,j],data[,i])))  + }  + }  + }  [1] "season"  [1] " Vs yr"  Pearson's Chi-squared test  data: table(data[, j], data[, i])  X-squared = 0.0041569, df = 3, p-value = 0.9999  [1] "season"  [1] " Vs mnth"  Pearson's Chi-squared test  data: table(data[, j], data[, i])  X-squared = 1765.1, df = 33, p-value < 2.2e-16  [1] "season"  [1] " Vs holiday"  Pearson's Chi-squared test  data: table(data[, j], data[, i])  X-squared = 1.4961, df = 3, p-value = 0.6832  [1] "season"  [1] " Vs weekday"  Pearson's Chi-squared test  data: table(data[, j], data[, i])  X-squared = 0.39925, df = 18, p-value = 1  [1] "season"  [1] " Vs workingday"  Pearson's Chi-squared test  data: table(data[, j], data[, i])  X-squared = 0.64285, df = 3, p-value = 0.8866  [1] "season"  [1] " Vs weathersit"  Pearson's Chi-squared test  data: table(data[, j], data[, i])  X-squared = 14.884, df = 6, p-value = 0.02118  [1] "yr"  [1] " Vs season"  Pearson's Chi-squared test  data: table(data[, j], data[, i])  X-squared = 0.0041569, df = 3, p-value = 0.9999  [1] "yr"  [1] " Vs mnth"  Pearson's Chi-squared test  data: table(data[, j], data[, i])  X-squared = 0.016176, df = 11, p-value = 1  [1] "yr"  [1] " Vs holiday"  Pearson's Chi-squared test with Yates' continuity correction  data: table(data[, j], data[, i])  X-squared = 9.6166e-30, df = 1, p-value = 1  [1] "yr"  [1] " Vs weekday"  Pearson's Chi-squared test  data: table(data[, j], data[, i])  X-squared = 0.027203, df = 6, p-value = 1  [1] "yr"  [1] " Vs workingday"  Pearson's Chi-squared test with Yates' continuity correction  data: table(data[, j], data[, i])  X-squared = 1.6156e-30, df = 1, p-value = 1  [1] "yr"  [1] " Vs weathersit"  Pearson's Chi-squared test  data: table(data[, j], data[, i])  X-squared = 4.1212, df = 2, p-value = 0.1274  [1] "mnth"  [1] " Vs season"  Pearson's Chi-squared test  data: table(data[, j], data[, i])  X-squared = 1765.1, df = 33, p-value < 2.2e-16  [1] "mnth"  [1] " Vs yr"  Pearson's Chi-squared test  data: table(data[, j], data[, i])  X-squared = 0.016176, df = 11, p-value = 1  [1] "mnth"  [1] " Vs holiday"  Pearson's Chi-squared test  data: table(data[, j], data[, i])  X-squared = 9.6808, df = 11, p-value = 0.5593  [1] "mnth"  [1] " Vs weekday"  Pearson's Chi-squared test  data: table(data[, j], data[, i])  X-squared = 3.372, df = 66, p-value = 1  [1] "mnth"  [1] " Vs workingday"  Pearson's Chi-squared test  data: table(data[, j], data[, i])  X-squared = 2.7777, df = 11, p-value = 0.9933  [1] "mnth"  [1] " Vs weathersit"  Pearson's Chi-squared test  data: table(data[, j], data[, i])  X-squared = 38.861, df = 22, p-value = 0.01464  [1] "holiday"  [1] " Vs season"  Pearson's Chi-squared test  data: table(data[, j], data[, i])  X-squared = 1.4961, df = 3, p-value = 0.6832  [1] "holiday"  [1] " Vs yr"  Pearson's Chi-squared test with Yates' continuity correction  data: table(data[, j], data[, i])  X-squared = 9.6166e-30, df = 1, p-value = 1  [1] "holiday"  [1] " Vs mnth"  Pearson's Chi-squared test  data: table(data[, j], data[, i])  X-squared = 9.6808, df = 11, p-value = 0.5593  [1] "holiday"  [1] " Vs weekday"  Pearson's Chi-squared test  data: table(data[, j], data[, i])  X-squared = 58.623, df = 6, p-value = 8.567e-11  [1] "holiday"  [1] " Vs workingday"  Pearson's Chi-squared test with Yates' continuity correction  data: table(data[, j], data[, i])  X-squared = 43.598, df = 1, p-value = 4.033e-11  [1] "holiday"  [1] " Vs weathersit"  Pearson's Chi-squared test  data: table(data[, j], data[, i])  X-squared = 1.0188, df = 2, p-value = 0.6009  [1] "weekday"  [1] " Vs season"  Pearson's Chi-squared test  data: table(data[, j], data[, i])  X-squared = 0.39925, df = 18, p-value = 1  [1] "weekday"  [1] " Vs yr"  Pearson's Chi-squared test  data: table(data[, j], data[, i])  X-squared = 0.027203, df = 6, p-value = 1  [1] "weekday"  [1] " Vs mnth"  Pearson's Chi-squared test  data: table(data[, j], data[, i])  X-squared = 3.372, df = 66, p-value = 1  [1] "weekday"  [1] " Vs holiday"  Pearson's Chi-squared test  data: table(data[, j], data[, i])  X-squared = 58.623, df = 6, p-value = 8.567e-11  [1] "weekday"  [1] " Vs workingday"  Pearson's Chi-squared test  data: table(data[, j], data[, i])  X-squared = 644.2, df = 6, p-value < 2.2e-16  [1] "weekday"  [1] " Vs weathersit"  Pearson's Chi-squared test  data: table(data[, j], data[, i])  X-squared = 14.358, df = 12, p-value = 0.2785  [1] "workingday"  [1] " Vs season"  Pearson's Chi-squared test  data: table(data[, j], data[, i])  X-squared = 0.64285, df = 3, p-value = 0.8866  [1] "workingday"  [1] " Vs yr"  Pearson's Chi-squared test with Yates' continuity correction  data: table(data[, j], data[, i])  X-squared = 1.6156e-30, df = 1, p-value = 1  [1] "workingday"  [1] " Vs mnth"  Pearson's Chi-squared test  data: table(data[, j], data[, i])  X-squared = 2.7777, df = 11, p-value = 0.9933  [1] "workingday"  [1] " Vs holiday"  Pearson's Chi-squared test with Yates' continuity correction  data: table(data[, j], data[, i])  X-squared = 43.598, df = 1, p-value = 4.033e-11  [1] "workingday"  [1] " Vs weekday"  Pearson's Chi-squared test  data: table(data[, j], data[, i])  X-squared = 644.2, df = 6, p-value < 2.2e-16  [1] "workingday"  [1] " Vs weathersit"  Pearson's Chi-squared test  data: table(data[, j], data[, i])  X-squared = 2.7427, df = 2, p-value = 0.2538  [1] "weathersit"  [1] " Vs season"  Pearson's Chi-squared test  data: table(data[, j], data[, i])  X-squared = 14.884, df = 6, p-value = 0.02118  [1] "weathersit"  [1] " Vs yr"  Pearson's Chi-squared test  data: table(data[, j], data[, i])  X-squared = 4.1212, df = 2, p-value = 0.1274  [1] "weathersit"  [1] " Vs mnth"  Pearson's Chi-squared test  data: table(data[, j], data[, i])  X-squared = 38.861, df = 22, p-value = 0.01464  [1] "weathersit"  [1] " Vs holiday"  Pearson's Chi-squared test  data: table(data[, j], data[, i])  X-squared = 1.0188, df = 2, p-value = 0.6009  [1] "weathersit"  [1] " Vs weekday"  Pearson's Chi-squared test  data: table(data[, j], data[, i])  X-squared = 14.358, df = 12, p-value = 0.2785  [1] "weathersit"  [1] " Vs workingday"  Pearson's Chi-squared test  data: table(data[, j], data[, i])  X-squared = 2.7427, df = 2, p-value = 0.2538  Warning messages:  1: In chisq.test(table(data[, j], data[, i])) :  Chi-squared approximation may be incorrect  2: In chisq.test(table(data[, j], data[, i])) :  Chi-squared approximation may be incorrect  3: In chisq.test(table(data[, j], data[, i])) :  Chi-squared approximation may be incorrect  4: In chisq.test(table(data[, j], data[, i])) :  Chi-squared approximation may be incorrect  5: In chisq.test(table(data[, j], data[, i])) :  Chi-squared approximation may be incorrect  6: In chisq.test(table(data[, j], data[, i])) :  Chi-squared approximation may be incorrect  7: In chisq.test(table(data[, j], data[, i])) :  Chi-squared approximation may be incorrect  8: In chisq.test(table(data[, j], data[, i])) :  Chi-squared approximation may be incorrect  9: In chisq.test(table(data[, j], data[, i])) :  Chi-squared approximation may be incorrect  10: In chisq.test(table(data[, j], data[, i])) :  Chi-squared approximation may be incorrect  > #If p-value<0.05 (Reject Null Hypothesis) => variable A depends on variable B.  > #If p-value>0.05 (Do Not Reject Null Hypothesis) => Variable A & variable B are independent of each other.  > #Output: "workingday"-"holiday","weekday"-"workingday","weekday"-"holiday" & "mnth"-"season depend on each other significantly.  >  > #######Using Random Forest Algorithm:  > data.rf=randomForest(data$cnt~.,data = data, ntree=1000, keep.forest= F, importance= T)  > importance(data.rf,type = 1)  %IncMSE  dteday 4.673530  season 21.268255  yr 40.189180  mnth 21.522989  holiday 1.985289  weekday 24.293412  workingday 22.960790  weathersit 12.793259  temp 21.295601  atemp 24.730151  hum 17.895417  windspeed 7.856791  casual 41.353669  registered 69.804438  > #"holiday" has the least importance.  > varImpPlot(data.rf,type = 1)  >  > #######ANOVA test (comparision of Target Vs categorical variables)  > anovacat = aov(cnt ~ season + yr + mnth + holiday + workingday + weekday + weathersit , data = data)  > summary(anovacat)  Df Sum Sq Mean Sq F value Pr(>F)  season 3 950595868 316865289 436.234 < 2e-16 \*\*\*  yr 1 884008263 884008263 1217.030 < 2e-16 \*\*\*  mnth 11 187311622 17028329 23.443 < 2e-16 \*\*\*  holiday 1 3306975 3306975 4.553 0.03321 \*  workingday 1 3209216 3209216 4.418 0.03591 \*  weekday 5 12629845 2525969 3.478 0.00411 \*\*  weathersit 2 185659616 92829808 127.800 < 2e-16 \*\*\*  Residuals 706 512813988 726365  ---  Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1  > #If p-value<0.05 (Reject Null Hypothesis) => Population means are significantly different.  > #If p-value>0.05 (Do Not Reject Null Hypothesis) => Population means are not significantly different or are same. |
|  |
| |  | | --- | |  | |

> ###################\_\_\_\_\_\_\_\_\_\_\_Feature Engineering\_\_\_\_\_\_\_\_\_\_\_\_######################

> #From Chi-square test, we notice that "working day", "holiday" & "weekday" depend on each other and intuitively there is a logical connection within them.

> #We make a new variable using this connection between the three varibles

> #Denote: 1-->weekend, 2--> working day, 3--> holiday

>

> data$day = NA

> for (i in 1:nrow(data)){

+ if ((data[i,7]=="0") && (data[i,5]=="0")){data[i,16] = 1} #weekend

+ else if ((data[i,7]=="1") && (data[i,5]=="0")){data[i,16] = 2} #working day

+ else if ((data[i,7]=="0") && (data[i,5]=="1")){data[i,16] = 3} #holiday

+ else data[i,16] =NA

+ }

> sum(is.na(data$day)) #= 0, so no anomaly data case where it is working day & holiday both.

[1] 0

>

> ###################Dimensional Reduction######################

> #Won't remove "dteday" variable as the user count is tracked on each day.

> #As we added "day" new variable using "workingday" & "holiday", we can remove them both as "day" holds the information of both.

> data$holiday = data$day

> data$day = NULL

> colnames(data)[5] = "day"

> data$day = as.factor(data$day) # New variable "day": Factor w/ 3 levels "1","2","3"

> #"Season" has multicollinearity problem as well and it is related to "mnth", so we can remove it.

> data= subset(data, select= -c(season,workingday,temp,casual,registered))

> factor\_data = subset(data, select= c(yr,mnth,day,weekday,weathersit)) #5 factor variables

> num\_data = subset(data, select= c(dteday,atemp,hum,windspeed,cnt)) #5 numerical variables, contains target variable

> dim(data) # 731 obs. x 10 variables

[1] 731 10

> str(data)

'data.frame': 731 obs. of 10 variables:

$ dteday : num 1 2 3 4 5 6 7 8 9 10 ...

$ yr : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...

$ mnth : Factor w/ 12 levels "1","2","3","4",..: 1 1 1 1 1 1 1 1 1 1 ...

$ day : Factor w/ 3 levels "1","2","3": 1 1 2 2 2 2 2 1 1 2 ...

$ weekday : Factor w/ 7 levels "0","1","2","3",..: 7 1 2 3 4 5 6 7 1 2 ...

$ weathersit: Factor w/ 3 levels "1","2","3": 2 2 1 1 1 1 2 2 1 1 ...

$ 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 ...

$ cnt : num 985 801 1349 1562 1600 ...

> ################################Feature Scaling##################################

> #All continuous variables are already normalised in this data set.

>

>

> rm(list= ls()[!(ls() %in% c('data','factor\_data','num\_data'))])

>

> ##############################Sampling#################################

> set.seed(777)

>

> sample.index = sample(nrow(data), 0.8\*nrow(data), replace = F) #80% data -->Train set, 20%--> Test set

> train = data[sample.index,]

> test = data[-sample.index,]

> dim(train) # 584 x 11

[1] 584 10

> dim(test) # 147 x 11

[1] 147 10

> ##################################Model Development###################################

> #As the target variable is of numeric type, this is a regression problem.

> ######1.Decision Tree######

> #Decision trees can handle both categorical and numerical variables at the same time as features.

> dt=rpart(cnt~.,data = train,method= "anova")

> summary(dt)

Call:

rpart(formula = cnt ~ ., data = train, method = "anova")

n= 584

CP nsplit rel error xerror xstd

1 0.37445616 0 1.0000000 1.0051616 0.04580505

2 0.22311915 1 0.6255438 0.6603832 0.03348656

3 0.09060873 2 0.4024247 0.4239814 0.03179904

4 0.02962425 3 0.3118160 0.3290237 0.02734505

5 0.02934392 4 0.2821917 0.3120647 0.02819117

6 0.02895436 5 0.2528478 0.3120647 0.02819117

7 0.01189898 6 0.2238934 0.2660670 0.02168208

8 0.01131214 7 0.2119945 0.2668795 0.02194647

9 0.01000000 8 0.2006823 0.2633306 0.02187781

Variable importance

atemp mnth yr hum windspeed weathersit weekday

34 27 25 8 4 1 1

Node number 1: 584 observations, complexity param=0.3744562

mean=4565.748, MSE=3745566

left son=2 (234 obs) right son=3 (350 obs)

Primary splits:

atemp < 0.4308565 to the left, improve=0.37445620, (0 missing)

yr splits as LR, improve=0.35623910, (0 missing)

mnth splits as LLLRRRRRRRLL, improve=0.30009300, (0 missing)

weathersit splits as RLL, improve=0.07434951, (0 missing)

hum < 0.824394 to the right, improve=0.06695468, (0 missing)

Surrogate splits:

mnth splits as LLLRRRRRRRLL, agree=0.894, adj=0.735, (0 split)

hum < 0.5464585 to the left, agree=0.625, adj=0.064, (0 split)

windspeed < 0.06282915 to the left, agree=0.616, adj=0.043, (0 split)

dteday < 29.5 to the right, agree=0.601, adj=0.004, (0 split)

Node number 2: 234 observations, complexity param=0.09060873

mean=3117.359, MSE=2302852

left son=4 (126 obs) right son=5 (108 obs)

Primary splits:

yr splits as LR, improve=0.36780560, (0 missing)

atemp < 0.2607295 to the left, improve=0.23258030, (0 missing)

mnth splits as LLLRL--R-RRR, improve=0.19311160, (0 missing)

hum < 0.678777 to the right, improve=0.06662897, (0 missing)

weathersit splits as RLL, improve=0.06151398, (0 missing)

Surrogate splits:

hum < 0.5725 to the right, agree=0.577, adj=0.083, (0 split)

atemp < 0.332973 to the left, agree=0.573, adj=0.074, (0 split)

windspeed < 0.1871895 to the right, agree=0.568, adj=0.065, (0 split)

mnth splits as LRLRL--R-LRL, agree=0.564, adj=0.056, (0 split)

weekday splits as LLLLLRL, agree=0.543, adj=0.009, (0 split)

Node number 3: 350 observations, complexity param=0.2231192

mean=5534.1, MSE=2369868

left son=6 (164 obs) right son=7 (186 obs)

Primary splits:

yr splits as LR, improve=0.58840310, (0 missing)

hum < 0.834375 to the right, improve=0.15010660, (0 missing)

weathersit splits as RRL, improve=0.09686697, (0 missing)

atemp < 0.5018855 to the left, improve=0.06263038, (0 missing)

mnth splits as -LRLRRRRRRLR, improve=0.05588727, (0 missing)

Surrogate splits:

hum < 0.6947915 to the right, agree=0.580, adj=0.104, (0 split)

mnth splits as -RRLRLRRRRLR, agree=0.569, adj=0.079, (0 split)

atemp < 0.5296815 to the left, agree=0.549, adj=0.037, (0 split)

weekday splits as RLLRRRR, agree=0.546, adj=0.030, (0 split)

windspeed < 0.1741335 to the right, agree=0.543, adj=0.024, (0 split)

Node number 4: 126 observations, complexity param=0.02962425

mean=2265.302, MSE=1057926

left son=8 (75 obs) right son=9 (51 obs)

Primary splits:

mnth splits as LLLLR----RRR, improve=0.48612910, (0 missing)

atemp < 0.251738 to the left, improve=0.30669750, (0 missing)

windspeed < 0.112571 to the right, improve=0.24712020, (0 missing)

hum < 0.86 to the right, improve=0.11724950, (0 missing)

weathersit splits as RLL, improve=0.07345125, (0 missing)

Surrogate splits:

windspeed < 0.120031 to the right, agree=0.746, adj=0.373, (0 split)

atemp < 0.298832 to the left, agree=0.714, adj=0.294, (0 split)

hum < 0.611667 to the left, agree=0.611, adj=0.039, (0 split)

dteday < 22.5 to the left, agree=0.603, adj=0.020, (0 split)

day splits as LLR, agree=0.603, adj=0.020, (0 split)

Node number 5: 108 observations, complexity param=0.02895436

mean=4111.426, MSE=1920095

left son=10 (31 obs) right son=11 (77 obs)

Primary splits:

atemp < 0.279985 to the left, improve=0.30542030, (0 missing)

mnth splits as LLLR---R-LRL, improve=0.28345620, (0 missing)

hum < 0.697292 to the right, improve=0.16823620, (0 missing)

weathersit splits as RLL, improve=0.09756212, (0 missing)

weekday splits as LLLRRRL, improve=0.07717721, (0 missing)

Surrogate splits:

hum < 0.4647915 to the left, agree=0.741, adj=0.097, (0 split)

windspeed < 0.349942 to the right, agree=0.731, adj=0.065, (0 split)

mnth splits as RRRR---L-RRR, agree=0.722, adj=0.032, (0 split)

weathersit splits as RRL, agree=0.722, adj=0.032, (0 split)

Node number 6: 164 observations, complexity param=0.01131214

mean=4276.524, MSE=648554.7

left son=12 (29 obs) right son=13 (135 obs)

Primary splits:

mnth splits as -LLLRRRRRRLL, improve=0.23264010, (0 missing)

hum < 0.849375 to the right, improve=0.23168870, (0 missing)

weathersit splits as RLL, improve=0.18122010, (0 missing)

atemp < 0.5805125 to the left, improve=0.17080540, (0 missing)

windspeed < 0.1265645 to the right, improve=0.07228776, (0 missing)

Surrogate splits:

atemp < 0.456723 to the left, agree=0.872, adj=0.276, (0 split)

windspeed < 0.299444 to the right, agree=0.854, adj=0.172, (0 split)

hum < 0.908125 to the right, agree=0.829, adj=0.034, (0 split)

Node number 7: 186 observations, complexity param=0.02934392

mean=6642.93, MSE=1263643

left son=14 (9 obs) right son=15 (177 obs)

Primary splits:

hum < 0.8322915 to the right, improve=0.27309330, (0 missing)

weathersit splits as RLL, improve=0.13018900, (0 missing)

atemp < 0.4927355 to the left, improve=0.12328470, (0 missing)

mnth splits as -LLLRRRRRR-L, improve=0.07749548, (0 missing)

windspeed < 0.287627 to the right, improve=0.06415826, (0 missing)

Surrogate splits:

weathersit splits as RRL, agree=0.968, adj=0.333, (0 split)

windspeed < 0.3526145 to the right, agree=0.957, adj=0.111, (0 split)

Node number 8: 75 observations

mean=1673.933, MSE=304991.8

Node number 9: 51 observations

mean=3134.961, MSE=894587.3

Node number 10: 31 observations

mean=2904.516, MSE=1394240

Node number 11: 77 observations, complexity param=0.01189898

mean=4597.325, MSE=1309269

left son=22 (18 obs) right son=23 (59 obs)

Primary splits:

hum < 0.700625 to the right, improve=0.25817860, (0 missing)

mnth splits as LLLR-----LRL, improve=0.23626120, (0 missing)

weathersit splits as RL-, improve=0.15559330, (0 missing)

atemp < 0.3134065 to the left, improve=0.08333220, (0 missing)

dteday < 19.5 to the right, improve=0.07422147, (0 missing)

Surrogate splits:

weathersit splits as RL-, agree=0.792, adj=0.111, (0 split)

mnth splits as RRRR-----LRR, agree=0.779, adj=0.056, (0 split)

Node number 12: 29 observations

mean=3438.448, MSE=473523.1

Node number 13: 135 observations

mean=4456.556, MSE=502863

Node number 14: 9 observations

mean=4037.778, MSE=2317994

Node number 15: 177 observations

mean=6775.395, MSE=847392.4

Node number 22: 18 observations

mean=3544.722, MSE=1303907

Node number 23: 59 observations

mean=4918.458, MSE=869753.4

>

> #Predict for new test cases

> predict.dt=predict(dt,test[,-10])

>

> #Error metric:

> postResample(predict.dt,test[,10])

RMSE Rsquared MAE

1036.8218286 0.7105788 768.8217306

> #Output:

> #RMSE Rsquared MAE

> #1036.8218286 0.7105788 768.8217306

>

> #calculate MAPE

> mape = function(y,yi)

+ {mean(abs((y-yi)/y))\*100

+ }

> mape.dt = mape(test[,10],predict.dt) #30.79%

>

> library(mltools)

Warning message:

package ‘mltools’ was built under R version 3.4.4

> rmsle(predict.dt,test[,10]) #0.3665

[1] 0.3665201

>

> #######2.Random Forest Algorithm#######

> rf = randomForest(cnt~., train, importance = TRUE, ntree = 500)

> summary(rf)

Length Class Mode

call 5 -none- call

type 1 -none- character

predicted 584 -none- numeric

mse 500 -none- numeric

rsq 500 -none- numeric

oob.times 584 -none- numeric

importance 18 -none- numeric

importanceSD 9 -none- numeric

localImportance 0 -none- NULL

proximity 0 -none- NULL

ntree 1 -none- numeric

mtry 1 -none- numeric

forest 11 -none- list

coefs 0 -none- NULL

y 584 -none- numeric

test 0 -none- NULL

inbag 0 -none- NULL

terms 3 terms call

> #Predict for test case:

> predict.rf <- data.frame(predict(rf, subset(test, select = -c(cnt))))

> #Error metric:

> postResample(predict.rf,test[,10])

RMSE Rsquared MAE

770.2988607 0.8533753 571.7846934

> #Output:

> #RMSE Rsquared MAE

> #778.4675527 0.8507608 576.6110231

>

> mape.rf = mape(test[,10],predict.rf$predict.rf..subset.test..select....c.cnt...) # 24.9%

> ########3.Multiple Linear Regression########

>

> #creating dummy variables for categorical data

> library(dummies)

dummies-1.5.6 provided by Decision Patterns

Warning message:

package ‘dummies’ was built under R version 3.4.4

> factor\_new = dummy.data.frame(factor\_data, sep = ".") #731 x 27

>

> #sampling#

> df = cbind(factor\_new, num\_data)

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

> # df[,i] = as.numeric(df[,i])

> #}

> str(df) # 731 X 32

'data.frame': 731 obs. of 32 variables:

$ yr.0 : int 1 1 1 1 1 1 1 1 1 1 ...

$ yr.1 : int 0 0 0 0 0 0 0 0 0 0 ...

$ mnth.1 : int 1 1 1 1 1 1 1 1 1 1 ...

$ mnth.2 : int 0 0 0 0 0 0 0 0 0 0 ...

$ mnth.3 : int 0 0 0 0 0 0 0 0 0 0 ...

$ mnth.4 : int 0 0 0 0 0 0 0 0 0 0 ...

$ mnth.5 : int 0 0 0 0 0 0 0 0 0 0 ...

$ mnth.6 : int 0 0 0 0 0 0 0 0 0 0 ...

$ mnth.7 : int 0 0 0 0 0 0 0 0 0 0 ...

$ mnth.8 : int 0 0 0 0 0 0 0 0 0 0 ...

$ mnth.9 : int 0 0 0 0 0 0 0 0 0 0 ...

$ mnth.10 : int 0 0 0 0 0 0 0 0 0 0 ...

$ mnth.11 : int 0 0 0 0 0 0 0 0 0 0 ...

$ mnth.12 : int 0 0 0 0 0 0 0 0 0 0 ...

$ day.1 : int 1 1 0 0 0 0 0 1 1 0 ...

$ day.2 : int 0 0 1 1 1 1 1 0 0 1 ...

$ day.3 : int 0 0 0 0 0 0 0 0 0 0 ...

$ weekday.0 : int 0 1 0 0 0 0 0 0 1 0 ...

$ weekday.1 : int 0 0 1 0 0 0 0 0 0 1 ...

$ weekday.2 : int 0 0 0 1 0 0 0 0 0 0 ...

$ weekday.3 : int 0 0 0 0 1 0 0 0 0 0 ...

$ weekday.4 : int 0 0 0 0 0 1 0 0 0 0 ...

$ weekday.5 : int 0 0 0 0 0 0 1 0 0 0 ...

$ weekday.6 : int 1 0 0 0 0 0 0 1 0 0 ...

$ weathersit.1: int 0 0 1 1 1 1 0 0 1 1 ...

$ weathersit.2: int 1 1 0 0 0 0 1 1 0 0 ...

$ weathersit.3: int 0 0 0 0 0 0 0 0 0 0 ...

$ dteday : num 1 2 3 4 5 6 7 8 9 10 ...

$ 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 ...

$ cnt : num 985 801 1349 1562 1600 ...

>

> set.seed(123)

> train\_index = sample(1:nrow(df), 0.8\*nrow(df))

> train.df = df[train\_index,] #584 x 32

> test.df = df[-train\_index,] #147 x 32

>

> #Check Multicollinearity

> library(usdm)

Loading required package: sp

Loading required package: raster

Attaching package: ‘raster’

The following object is masked from ‘package:dplyr’:

select

Warning messages:

1: package ‘usdm’ was built under R version 3.4.4

2: package ‘sp’ was built under R version 3.4.4

3: package ‘raster’ was built under R version 3.4.4

> vif(df[,-32])

Variables VIF

1 yr.0 Inf

2 yr.1 Inf

3 mnth.1 Inf

4 mnth.2 Inf

5 mnth.3 Inf

6 mnth.4 Inf

7 mnth.5 Inf

8 mnth.6 Inf

9 mnth.7 Inf

10 mnth.8 Inf

11 mnth.9 Inf

12 mnth.10 Inf

13 mnth.11 Inf

14 mnth.12 Inf

15 day.1 Inf

16 day.2 Inf

17 day.3 Inf

18 weekday.0 Inf

19 weekday.1 Inf

20 weekday.2 Inf

21 weekday.3 Inf

22 weekday.4 Inf

23 weekday.5 Inf

24 weekday.6 Inf

25 weathersit.1 Inf

26 weathersit.2 Inf

27 weathersit.3 Inf

28 dteday 1.010204

29 atemp 6.049203

30 hum 2.294781

31 windspeed 1.207595

> vifcor(df[,-32], th = 0.8)

3 variables from the 31 input variables have collinearity problem:

yr.1 weathersit.2 day.2

After excluding the collinear variables, the linear correlation coefficients ranges between:

min correlation ( windspeed ~ weekday.3 ): -0.0001206042

max correlation ( weekday.0 ~ day.1 ): 0.6450846

---------- VIFs of the remained variables --------

Variables VIF

1 yr.0 1.049547

2 mnth.1 Inf

3 mnth.2 Inf

4 mnth.3 Inf

5 mnth.4 Inf

6 mnth.5 Inf

7 mnth.6 Inf

8 mnth.7 Inf

9 mnth.8 Inf

10 mnth.9 Inf

11 mnth.10 Inf

12 mnth.11 Inf

13 mnth.12 Inf

14 day.1 Inf

15 day.3 1.106961

16 weekday.0 Inf

17 weekday.1 Inf

18 weekday.2 Inf

19 weekday.3 Inf

20 weekday.4 Inf

21 weekday.5 Inf

22 weekday.6 Inf

23 weathersit.1 1.779943

24 weathersit.3 1.222714

25 dteday 1.010204

26 atemp 6.049203

27 hum 2.294781

28 windspeed 1.207595

> #Output:

> #3 variables from the 31 input variables have collinearity problem: yr.1, weathersit.2, day.2

> #removing multicollinear variables and redo check:

> df = subset(df, select= -c(yr.1, weathersit.2, day.2))

> train.df = subset(train.df, select= -c(yr.1, weathersit.2, day.2)) #584 x 29

> test.df = subset(test.df, select= -c(yr.1, weathersit.2, day.2)) #147 x 29

> dim(df) #731 x 29

[1] 731 29

> #Recheck VIFCORR: No variable from the 29 input variables has collinearity problem.

>

> #run regression model

> lr = lm(cnt~., data = train.df)

> #summary of the model

> summary(lr)

Call:

lm(formula = cnt ~ ., data = train.df)

Residuals:

Min 1Q Median 3Q Max

-3876.2 -387.8 50.8 509.4 2771.2

Coefficients: (3 not defined because of singularities)

Estimate Std. Error t value Pr(>|t|)

(Intercept) 4430.566 344.461 12.862 < 2e-16 \*\*\*

yr.0 -2113.166 71.121 -29.712 < 2e-16 \*\*\*

mnth.1 -825.824 175.718 -4.700 3.29e-06 \*\*\*

mnth.2 -716.510 179.767 -3.986 7.62e-05 \*\*\*

mnth.3 138.034 174.608 0.791 0.429552

mnth.4 632.261 191.820 3.296 0.001043 \*\*

mnth.5 957.277 209.921 4.560 6.29e-06 \*\*\*

mnth.6 673.222 240.603 2.798 0.005319 \*\*

mnth.7 362.334 258.956 1.399 0.162305

mnth.8 644.409 241.596 2.667 0.007868 \*\*

mnth.9 1396.680 213.404 6.545 1.35e-10 \*\*\*

mnth.10 1391.067 187.618 7.414 4.56e-13 \*\*\*

mnth.11 785.587 172.682 4.549 6.61e-06 \*\*\*

mnth.12 NA NA NA NA

day.1 8.810 129.991 0.068 0.945990

day.3 -813.416 212.812 -3.822 0.000147 \*\*\*

weekday.0 -424.315 129.802 -3.269 0.001146 \*\*

weekday.1 -165.593 133.805 -1.238 0.216395

weekday.2 -151.960 130.711 -1.163 0.245504

weekday.3 -23.876 130.518 -0.183 0.854920

weekday.4 -54.480 133.389 -0.408 0.683114

weekday.5 NA NA NA NA

weekday.6 NA NA NA NA

weathersit.1 448.480 95.588 4.692 3.41e-06 \*\*\*

weathersit.3 -1468.420 232.217 -6.323 5.24e-10 \*\*\*

dteday -10.119 3.989 -2.537 0.011455 \*

atemp 4592.019 519.614 8.837 < 2e-16 \*\*\*

hum -1522.767 365.632 -4.165 3.61e-05 \*\*\*

windspeed -2629.300 543.404 -4.839 1.69e-06 \*\*\*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 837.3 on 558 degrees of freedom

Multiple R-squared: 0.8237, Adjusted R-squared: 0.8158

F-statistic: 104.3 on 25 and 558 DF, p-value: < 2.2e-16

>

> #Predict for test case:

> predict.lr= predict(lr, test.df[,-29])

Warning message:

In predict.lm(lr, test.df[, -29]) :

prediction from a rank-deficient fit may be misleading

> #Error metric:

> postResample(predict.lr,test.df[,29])

RMSE Rsquared MAE

800.2783046 0.8303233 581.4298996

> #Output:

> #RMSE Rsquared MAE

> #800.2783046 0.8303233 581.4298996

>

> mape.lr = mape(test.df[,29],predict.lr) #17.5%

> ##############4.KNN Implementation##############

> #To check for best k value:

> model <- train(cnt~., data = train, method = "knn",

+ trControl = trainControl("cv", number = 10),

+ tuneLength = 15)

> model$bestTune

k

3 9

> #k = 3 , 9

> plot(model)

>

> #K=3:

> predict.knn = knn.reg(train = train.df[,-29],test = test.df[,-29],train.df$cnt, k= 3)

> print(predict.knn)

Prediction:

[1] 2065.6667 2536.3333 1932.3333 3527.6667 2452.6667 2715.0000 3432.6667 3249.0000 2683.3333 2535.3333 5425.0000

[12] 4446.0000 3188.3333 2695.3333 2765.3333 2717.6667 4299.0000 2918.6667 5065.3333 4840.6667 5374.0000 4964.6667

[23] 3303.3333 2663.6667 4450.0000 4485.0000 4465.6667 4904.6667 4851.6667 5427.0000 4686.3333 4353.0000 5078.0000

[34] 3303.6667 5599.3333 3440.0000 5282.0000 5925.6667 4382.3333 5548.3333 5256.0000 5491.6667 5779.6667 4732.6667

[45] 4389.6667 5922.6667 3241.6667 4966.3333 5510.0000 4561.6667 3120.3333 1752.3333 4949.6667 4383.3333 4426.6667

[56] 3805.6667 3521.3333 3140.6667 5116.0000 5404.3333 820.3333 3466.0000 2473.6667 3146.6667 4362.6667 2408.6667

[67] 3552.0000 4550.0000 4181.0000 3621.3333 3556.3333 3526.0000 2959.6667 6236.0000 4624.3333 4371.6667 3165.6667

[78] 2672.6667 3727.3333 4956.0000 5488.0000 5411.0000 4105.0000 3885.3333 4096.3333 4571.0000 5457.6667 5490.6667

[89] 5447.0000 4543.6667 4662.0000 6788.0000 6100.6667 2542.0000 6042.3333 5576.3333 3296.0000 5726.0000 5794.0000

[100] 4209.6667 5906.3333 2978.6667 4869.3333 6659.0000 6323.6667 6317.0000 4854.0000 6379.6667 4798.0000 5062.3333

[111] 4478.0000 4873.3333 5184.3333 6636.6667 5805.0000 6186.0000 4622.6667 5748.3333 6751.6667 5806.3333 5742.3333

[122] 6693.0000 6282.6667 6944.3333 4264.6667 4596.6667 5916.3333 5431.0000 6396.6667 5515.0000 6063.3333 4261.6667

[133] 3808.3333 4501.3333 6322.3333 5464.6667 2055.6667 4677.0000 4798.3333 3778.3333 6085.3333 5459.3333 3677.6667

[144] 6222.3333 6298.6667 5863.0000 4790.0000

>

> #Error metric:

> postResample(predict.knn$pred,test.df[,29])

RMSE Rsquared MAE

1392.7631351 0.4544424 1110.0045351

> #Output:

> #RMSE Rsquared MAE

> #1392.7631351 0.4544424 1110.0045351

>

> mape.knn = mape(test.df[,29],predict.knn$pred) #38.98%

>

> #K=5:

> #predict.knn = knn.reg(train = train.df[,-29],test = test.df[,-29],train.df$cnt, k= 5)

> #print(predict.knn)

> #Error metric:

> #mape(test.df[,29],predict.knn$pred)

> #Output:

> #mape

> #45.26592 %

> #postResample(predict.knn$pred,test.df[,29])

> #RMSE Rsquared MAE

> #1450.9419952 0.4484269 1169.3782313

>

> #K=7:

> #predict.knn = knn.reg(train = train.df[,-29],test = test.df[,-29],train.df$cnt, k= 7)

> #print(predict.knn)

> #Error metric:

> #mape(test.df[,29],predict.knn$pred)

> #Output:

> #mape

> #47.63637 %

> #postResample(predict.knn$pred,test.df[,29])

> #RMSE Rsquared MAE

> #1456.0507716 0.4983456 1171.8736638

> ######And so on, done upto k = 11.

>

> #A new dataframe to store results

> algorithm <- c('Decision Tree','Random Forest','Linear Regression','KNN')

> MAPE\_val <- c(mape.dt,mape.rf,mape.lr,mape.knn)

> results <- data.frame(algorithm, MAPE\_val)

> print(results)

algorithm MAPE\_val

1 Decision Tree 30.79662

2 Random Forest 24.98612

3 Linear Regression 17.55068

4 KNN 38.98097

> barplot(results$MAPE\_val, width = 1, names.arg = results$algorithm,

+ ylab="MAPE value", xlab = "Algorithm",col='pink')

>

> ##Thus, we find the "Multiple Linear Regression Algorithm" gives us the best result with the least MAPE for this dataset.

**Bike Renting – Python code**

*#Set working directory*

**import** **os**

os.chdir("F:/DS/edWisor/Project 2")

os.getcwd()

Out[1]:

'F:\\DS\\edWisor\\Project 2'

**Load libraries**

In [2]:

**import** **pandas** **as** **pd**

**import** **matplotlib.pyplot** **as** **plt**

**import** **numpy** **as** **np**

**import** **seaborn** **as** **sns**

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

**from** **random** **import** randrange, uniform

**from** **scipy.stats** **import** chi2\_contingency

**from** **ggplot** **import** \*

C:\Users\sir\Anaconda3\lib\site-packages\ggplot\utils.py:81: FutureWarning: pandas.tslib is deprecated and will be removed in a future version.

You can access Timestamp as pandas.Timestamp

pd.tslib.Timestamp,

In [3]:

**from** **fancyimpute** **import** KNN

C:\Users\sir\Anaconda3\lib\site-packages\h5py\\_\_init\_\_.py:36: FutureWarning: Conversion of the second argument of issubdtype from `float` to `np.floating` is deprecated. In future, it will be treated as `np.float64 == np.dtype(float).type`.

from .\_conv import register\_converters as \_register\_converters

Using TensorFlow backend.

In [4]:

**import** **datetime** **as** **dt**

In [5]:

*#Load the data*

data = pd.read\_csv("day.csv")

**Data exploration**

In [6]:

data.shape

Out[6]:

(731, 16)

In [7]:

In [8]:

type(data)

Out[8]:

pandas.core.frame.DataFrame

In [9]:

data.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 731 entries, 0 to 730

Data columns (total 16 columns):

instant 731 non-null int64

dteday 731 non-null object

season 731 non-null int64

yr 731 non-null int64

mnth 731 non-null int64

holiday 731 non-null int64

weekday 731 non-null int64

workingday 731 non-null int64

weathersit 731 non-null int64

temp 731 non-null float64

atemp 731 non-null float64

hum 731 non-null float64

windspeed 731 non-null float64

casual 731 non-null int64

registered 731 non-null int64

cnt 731 non-null int64

dtypes: float64(4), int64(11), object(1)

memory usage: 91.5+ KB

In [10]:

*#Missing Value Analysis*

*#Check for missing value*

data.isnull().sum()

*#No missing values in the dataset*

Out[10]:

instant 0

dteday 0

season 0

yr 0

mnth 0

holiday 0

weekday 0

workingday 0

weathersit 0

temp 0

atemp 0

hum 0

windspeed 0

casual 0

registered 0

cnt 0

dtype: int64

In [11]:

*#remove "instant" variable as it is just like serial number & doesn't predict*

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

In [12]:

data.shape

Out[12]:

(731, 15)

In [13]:

*#extracting day number from 'dteday' variable*

data['dteday'].apply(str)

data['dteday'] = pd.to\_datetime(data['dteday'])

data['dteday'] = pd.DatetimeIndex(data['dteday']).day

*#removing 'dteday' variable*

In [14]:

In [15]:

*#save numeric & categorical names*

numnames = ["dteday","temp","atemp","hum","windspeed","casual","registered","cnt"]

catnames = ["season","yr","mnth","holiday","weekday","workingday","weathersit"]

data.shape

Out[15]:

(731, 15)

In [16]:

**for** i **in** catnames:

data[i] = data[i].astype('object')

**for** i **in** numnames:

data[i] = data[i].astype('float')

In [17]:

data.dtypes

Out[17]:

dteday float64

season object

yr object

mnth object

holiday object

weekday object

workingday object

weathersit object

temp float64

atemp float64

hum float64

windspeed float64

casual float64

registered float64

cnt float64

dtype: object

**Outlier analysis**

In [18]:

*#Plot boxplot to visualize Outliers*

%**matplotlib** inline

plt.boxplot(data['windspeed'])

Out[18]:

{'whiskers': [<matplotlib.lines.Line2D at 0x1f10645b978>,

<matplotlib.lines.Line2D at 0x1f10645be10>],

'caps': [<matplotlib.lines.Line2D at 0x1f106471278>,

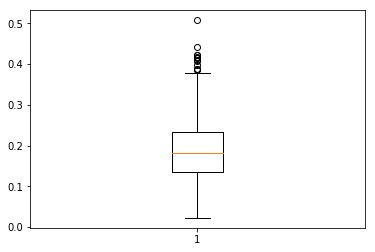
<matplotlib.lines.Line2D at 0x1f1064716a0>],

'boxes': [<matplotlib.lines.Line2D at 0x1f10645b828>],

'medians': [<matplotlib.lines.Line2D at 0x1f106471ac8>],

'fliers': [<matplotlib.lines.Line2D at 0x1f106471ef0>],

'means': []}



In [19]:

*#Detect and delete outliers from data*

**for** i **in** numnames:

print(i)

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

iqr = q75 - q25

min = q25 - (iqr\*1.5)

max = q75 + (iqr\*1.5)

print(min)

print(max)

*#Remove the outliers*

data = data.drop(data[data.loc[:,i] < min].index)

data = data.drop(data[data.loc[:,i] > max].index)

*#data.loc[data[i] < min,:i] = np.nan*

*#data.loc[data[i] > max,:i] = np.nan*

*#Calculate missing value*

*#missing\_val = pd.DataFrame(data.isnull().sum())*

*#Impute with KNN*

*#data = pd.DataFrame(KNN(21).fit\_transform(data), columns = data.columns)*

dteday

-14.5

45.5

temp

-0.14041600000000015

1.1329160000000003

atemp

-0.06829675000000018

1.0147412500000002

hum

0.20468725

1.0455212500000002

windspeed

-0.012431000000000025

0.380585

casual

-885.0

2323.0

registered

-840.0

8018.0

cnt

-788.125

9500.875

In [20]:

data.shape *#55 rows deleted*

Out[20]:

(676, 15)

In [21]:

data.isnull().sum()

Out[21]:

dteday 0

season 0

yr 0

mnth 0

holiday 0

weekday 0

workingday 0

weathersit 0

temp 0

atemp 0

hum 0

windspeed 0

casual 0

registered 0

cnt 0

dtype: int64

**Feature Selection**

In [22]:

*##Correlation analysis*

*#Correlation plot*

df\_corr = data.loc[:,numnames]

*#Set the width and height of the plot*

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

*#Generate correlation matrix*

corr = df\_corr.corr()

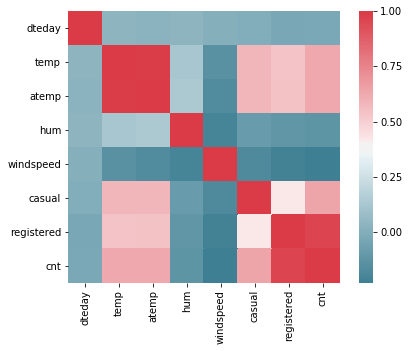
*#Plot using seaborn library*

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

square=**True**, ax=ax)

Out[22]:

<matplotlib.axes.\_subplots.AxesSubplot at 0x1f1064c17b8>



In [23]:

*#Chisquare test of independence*

*#loop for chi square values*

**for** i **in** catnames:

print(i)

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

print(p)

season

0.5306886312713439

yr

0.41642366315035007

mnth

0.4756091821561145

holiday

0.7870836122582522

weekday

0.43936502670720573

workingday

0.504633411642988

weathersit

0.5464467453059881

In [24]:

*#New Categorical Variable containing the data of "workingday" & "holiday"*

*#Denote: 1-->weekend, 2--> working day, 3--> holiday*

data.loc[(data['workingday'] == 0) & (data['holiday'] == 0), 'day'] = '1'

data.loc[(data['workingday'] == 1) & (data['holiday'] == 0), 'day'] = '2'

data.loc[(data['workingday'] == 0) & (data['holiday'] == 1), 'day'] = '3'

In [25]:

data = data.drop(["workingday","holiday","temp","casual","registered"], axis=1)

In [26]:

In [27]:

df = data[['dteday','mnth','yr','season','weekday','day','weathersit','atemp','hum','windspeed','cnt']]

In [28]:

In [29]:

*################################Feature Scaling##################################*

*#All continuous variables are already normalised in this data set.*

numnames = ["dteday","atemp","hum","windspeed"] *#not including "cnt" target variable*

catnames = ["mnth","yr","season","weekday","day","weathersit"]

**Model Development**

In [30]:

*#Data Sampling*

nrow= len(df.index)

train, test = train\_test\_split(df, test\_size = 0.2)

In [31]:

train.shape *#540 x 11*

test.shape *#136 x 11*

Out[31]:

(136, 11)

In [32]:

*#####Decision Tree Algortithm*

**from** **sklearn.tree** **import** DecisionTreeRegressor

fit\_dt= DecisionTreeRegressor(max\_depth=2).fit(train.iloc[:,0:10],train.iloc[:,10])

In [33]:

fit\_dt

Out[33]:

DecisionTreeRegressor(criterion='mse', max\_depth=2, max\_features=None,

max\_leaf\_nodes=None, min\_impurity\_decrease=0.0,

min\_impurity\_split=None, min\_samples\_leaf=1,

min\_samples\_split=2, min\_weight\_fraction\_leaf=0.0,

presort=False, random\_state=None, splitter='best')

In [34]:

predict\_dt= fit\_dt.predict(test.iloc[:,0:10])

In [35]:

*#Calculate RMSE*

**def** RMSE(actual, pred):

**return** np.sqrt(((pred - actual) \*\* 2).mean())

RMSE(test.iloc[:,10],predict\_dt)

*#output = 1162.84440171958*

Out[35]:

971.553404406351

In [36]:

*######Random Forest Algorithm*

**from** **sklearn.ensemble** **import** RandomForestRegressor

fit\_rf = RandomForestRegressor(n\_estimators = 100, random\_state = 99).fit(train.iloc[:,0:10],train.iloc[:,10])

In [37]:

fit\_rf

Out[37]:

RandomForestRegressor(bootstrap=True, criterion='mse', max\_depth=None,

max\_features='auto', max\_leaf\_nodes=None,

min\_impurity\_decrease=0.0, min\_impurity\_split=None,

min\_samples\_leaf=1, min\_samples\_split=2,

min\_weight\_fraction\_leaf=0.0, n\_estimators=100, n\_jobs=1,

oob\_score=False, random\_state=99, verbose=0, warm\_start=False)

In [38]:

predict\_rf= fit\_rf.predict(test.iloc[:,0:10])

In [39]:

RMSE(test.iloc[:,10],predict\_rf)

*#output = 765.0407919968172*

Out[39]:

640.3578319299065

In [40]:

*######Multiple Linear Regression*

**import** **statsmodels.api** **as** **sm**

*#Creat dataframe with all numerical variables*

df.lr = df[['cnt','dteday','atemp','hum','windspeed']]

*#create dummies for categorical variables*

**for** i **in** catnames:

temp = pd.get\_dummies(df[i],prefix = i)

df.lr = df.lr.join(temp)

C:\Users\sir\Anaconda3\lib\site-packages\ipykernel\_launcher.py:4: UserWarning: Pandas doesn't allow columns to be created via a new attribute name - see https://pandas.pydata.org/pandas-docs/stable/indexing.html#attribute-access

after removing the cwd from sys.path.

In [41]:

df.lr.shape *#676 x 36*

Out[41]:

(676, 36)

In [42]:

*#split data into train-test sets*

s = np.random.rand(len(df.lr))<0.8

train.lr = df.lr[s] *#80%*

test.lr = df.lr[~s] *#20%*

C:\Users\sir\Anaconda3\lib\site-packages\ipykernel\_launcher.py:3: UserWarning: Pandas doesn't allow columns to be created via a new attribute name - see https://pandas.pydata.org/pandas-docs/stable/indexing.html#attribute-access

This is separate from the ipykernel package so we can avoid doing imports until

C:\Users\sir\Anaconda3\lib\site-packages\ipykernel\_launcher.py:4: UserWarning: Pandas doesn't allow columns to be created via a new attribute name - see https://pandas.pydata.org/pandas-docs/stable/indexing.html#attribute-access

after removing the cwd from sys.path.

In [43]:

train.lr.shape *#564 x 36*

test.lr.shape *#112 x 36*

Out[43]:

(140, 36)

In [44]:

*#Build MLR model*

fit\_lr = sm.OLS(train.lr.iloc[:,0],train.lr.iloc[:,1:35]).fit()

fit\_lr.summary()

Out[44]:

|  |  |  |  |
| --- | --- | --- | --- |
| OLS Regression Results | | | |
| **Dep. Variable:** | cnt | **R-squared:** | 0.865 |
| **Model:** | OLS | **Adj. R-squared:** | 0.858 |
| **Method:** | Least Squares | **F-statistic:** | 116.5 |
| **Date:** | Tue, 12 Feb 2019 | **Prob (F-statistic):** | 9.03e-201 |
| **Time:** | 11:08:15 | **Log-Likelihood:** | -4251.2 |
| **No. Observations:** | 536 | **AIC:** | 8560. |
| **Df Residuals:** | 507 | **BIC:** | 8685. |
| **Df Model:** | 28 |  |  |
| **Covariance Type:** | nonrobust |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **coef** | **std err** | **t** | **P>|t|** | **[0.025** | **0.975]** |
| **dteday** | -4.8953 | 3.423 | -1.430 | 0.153 | -11.620 | 1.829 |
| **atemp** | 5004.3547 | 476.353 | 10.506 | 0.000 | 4068.486 | 5940.224 |
| **hum** | -1495.9409 | 327.803 | -4.564 | 0.000 | -2139.960 | -851.921 |
| **windspeed** | -2653.9405 | 457.089 | -5.806 | 0.000 | -3551.963 | -1755.918 |
| **mnth\_1** | -189.4452 | 180.020 | -1.052 | 0.293 | -543.122 | 164.231 |
| **mnth\_2** | -18.8093 | 169.510 | -0.111 | 0.912 | -351.837 | 314.218 |
| **mnth\_3** | 196.5986 | 135.886 | 1.447 | 0.149 | -70.371 | 463.568 |
| **mnth\_4** | 223.8030 | 162.762 | 1.375 | 0.170 | -95.968 | 543.574 |
| **mnth\_5** | 540.8962 | 176.178 | 3.070 | 0.002 | 194.768 | 887.025 |
| **mnth\_6** | 340.1530 | 166.083 | 2.048 | 0.041 | 13.857 | 666.449 |
| **mnth\_7** | -625.6951 | 205.824 | -3.040 | 0.002 | -1030.069 | -221.321 |
| **mnth\_8** | -22.3800 | 185.968 | -0.120 | 0.904 | -387.743 | 342.984 |
| **mnth\_9** | 605.9609 | 151.895 | 3.989 | 0.000 | 307.540 | 904.382 |
| **mnth\_10** | 433.9889 | 165.640 | 2.620 | 0.009 | 108.564 | 759.414 |
| **mnth\_11** | -133.4452 | 170.933 | -0.781 | 0.435 | -469.270 | 202.380 |
| **mnth\_12** | -193.9987 | 148.109 | -1.310 | 0.191 | -484.981 | 96.984 |
| **yr\_0** | -395.7701 | 159.882 | -2.475 | 0.014 | -709.882 | -81.658 |
| **yr\_1** | 1553.3972 | 156.655 | 9.916 | 0.000 | 1245.625 | 1861.170 |
| **season\_1** | -534.9101 | 144.198 | -3.710 | 0.000 | -818.210 | -251.611 |
| **season\_2** | 254.6098 | 146.252 | 1.741 | 0.082 | -32.724 | 541.944 |
| **season\_3** | 497.8568 | 155.222 | 3.207 | 0.001 | 192.899 | 802.815 |
| **season\_4** | 940.0706 | 154.690 | 6.077 | 0.000 | 636.158 | 1243.983 |
| **weekday\_0** | 70.4451 | 81.745 | 0.862 | 0.389 | -90.156 | 231.047 |
| **weekday\_1** | 3.9151 | 83.923 | 0.047 | 0.963 | -160.965 | 168.795 |
| **weekday\_2** | 165.8409 | 83.663 | 1.982 | 0.048 | 1.473 | 330.209 |
| **weekday\_3** | 155.4384 | 82.438 | 1.886 | 0.060 | -6.523 | 317.400 |
| **weekday\_4** | 228.3534 | 85.785 | 2.662 | 0.008 | 59.816 | 396.891 |
| **weekday\_5** | 248.7392 | 82.261 | 3.024 | 0.003 | 87.125 | 410.354 |
| **weekday\_6** | 284.8950 | 79.085 | 3.602 | 0.000 | 129.521 | 440.269 |
| **day\_1** | 355.3401 | 104.830 | 3.390 | 0.001 | 149.385 | 561.295 |
| **day\_2** | 642.0964 | 118.837 | 5.403 | 0.000 | 408.623 | 875.569 |
| **day\_3** | 160.1907 | 173.973 | 0.921 | 0.358 | -181.606 | 501.987 |
| **weathersit\_1** | 2139.8519 | 226.165 | 9.461 | 0.000 | 1695.516 | 2584.188 |
| **weathersit\_2** | 1700.9849 | 207.985 | 8.178 | 0.000 | 1292.366 | 2109.603 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Omnibus:** | 95.776 | **Durbin-Watson:** | 1.472 |
| **Prob(Omnibus):** | 0.000 | **Jarque-Bera (JB):** | 222.878 |
| **Skew:** | -0.934 | **Prob(JB):** | 4.01e-49 |
| **Kurtosis:** | 5.548 | **Cond. No.** | 1.00e+16 |

Warnings:  
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.  
[2] The smallest eigenvalue is 1.72e-27. This might indicate that there are  
strong multicollinearity problems or that the design matrix is singular.

In [45]:

predict\_lr = fit\_lr.predict(test.lr.iloc[:,1:35])

In [46]:

RMSE(test.lr.iloc[:,0],predict\_lr)

*#output = 713.1957640471251*

Out[46]:

892.5204419745069

In [47]:

*######KNN Implementation*

**from** **sklearn** **import** neighbors

rmse\_val = [] *#to store rmse values for different k*

**for** K **in** range(30):

K = K+1

fit\_knn = neighbors.KNeighborsRegressor(n\_neighbors = K)

fit\_knn.fit(train.iloc[:,0:10], train.iloc[:,10]) *#fit the model*

predict\_knn = fit\_knn.predict(test.iloc[:,0:10]) *#make prediction on test set*

error = RMSE(test.iloc[:,10] , predict\_knn) *#calculate rmse*

rmse\_val.append(error) *#store rmse values*

print('RMSE value for k= ' , K , 'is:', error)

RMSE value for k= 1 is: 1205.7335849768706

RMSE value for k= 2 is: 1259.7167267356765

RMSE value for k= 3 is: 1292.6514266360882

RMSE value for k= 4 is: 1428.151143569432

RMSE value for k= 5 is: 1479.955740196705

RMSE value for k= 6 is: 1493.1435739521633

RMSE value for k= 7 is: 1517.6304015518022

RMSE value for k= 8 is: 1533.6537566734078

RMSE value for k= 9 is: 1570.1710483274771

RMSE value for k= 10 is: 1546.1261414561611

RMSE value for k= 11 is: 1526.496100465862

RMSE value for k= 12 is: 1517.5136484571321

RMSE value for k= 13 is: 1486.9514980997221

RMSE value for k= 14 is: 1478.8068946139056

RMSE value for k= 15 is: 1463.907811840837

RMSE value for k= 16 is: 1459.5038055640289

RMSE value for k= 17 is: 1458.0110671935322

RMSE value for k= 18 is: 1458.9514319291206

RMSE value for k= 19 is: 1449.5929359139823

RMSE value for k= 20 is: 1434.4245678414761

RMSE value for k= 21 is: 1423.6093888400228

RMSE value for k= 22 is: 1415.6660108260676

RMSE value for k= 23 is: 1416.6971379552292

RMSE value for k= 24 is: 1409.898688944427

RMSE value for k= 25 is: 1425.570065466072

RMSE value for k= 26 is: 1432.9739021944652

RMSE value for k= 27 is: 1433.5920306962062

RMSE value for k= 28 is: 1450.2545484205818

RMSE value for k= 29 is: 1458.558312815522

RMSE value for k= 30 is: 1471.8719253246265

In [48]:

*#plotting the rmse values against k values*

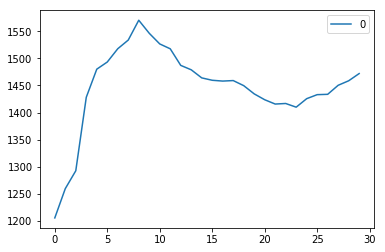
curve = pd.DataFrame(rmse\_val)

curve.plot()

*#K=2 is the value of neighbors for least RMSE.*

Out[48]:

<matplotlib.axes.\_subplots.AxesSubplot at 0x1f1069a9240>



In [49]:

*#For K=12:*

fit\_knn = neighbors.KNeighborsRegressor(n\_neighbors = 2)

fit\_knn.fit(train.iloc[:,0:10], train.iloc[:,10]) *#fit the model*

predict\_knn = fit\_knn.predict(test.iloc[:,0:10]) *#make prediction on test set*

RMSE(test.iloc[:,10] , predict\_knn)

*#output = 1209.595772142617*

Out[49]:

1259.7167267356765

In [50]:

*#Thus, we find the "Multiple Linear Regression Algorithm" gives us the best result with the least RMSE for this dataset.*