R version 4.3.1 (2023-06-16 ucrt) -- "Beagle Scouts"

Copyright (C) 2023 The R Foundation for Statistical Computing

Platform: x86\_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.

You are welcome to redistribute it under certain conditions.

Type 'license()' or 'licence()' for distribution details.

Natural language support but running in an English locale

R is a collaborative project with many contributors.

Type 'contributors()' for more information and

'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or

'help.start()' for an HTML browser interface to help.

Type 'q()' to quit R.

> # Group No: 2

> # Project name: Bike Sharing Demand

> # Phase: 3

> # Group Members: Vishnu Kariyattu, Harsha Bharti & Kritika Kaushik

>

> # Code Start

>

> # STEPS

> # LOADING LIBRARIES

>

> library(ggplot2) # Package for Data visualization

> library(caret) # Package for Data Splitting, Pre-Processing, Feature Selection, Model Training, Resampling,Performance Measurement, Ensembling etc.

Loading required package: lattice

> library(rpart) # Package for Decision Tree Building, Handling Missing Values, Pruning Trees, Cross-Validation

> library(rpart.plot) # Package for Plotting Decision Trees

> library(corrplot) # For Heat map

corrplot 0.92 loaded

Warning message:

package ‘corrplot’ was built under R version 4.3.2

> library(randomForest) # For Random Forest Model

randomForest 4.7-1.1

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

Attaching package: ‘randomForest’

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

margin

> library(nnet) # For Neural Network Model

>

> #LOAD

> source("F:/2. Babson/1. Curriculam + Studies/3. Sem 3/1. Course Work/Machine Learning/Complete data Set/Data/BabsonAnalytics.R")

> df = read.csv("F:/2. Babson/1. Curriculam + Studies/3. Sem 3/1. Course Work/Machine Learning/Complete data Set/Data/train.csv")

>

> # Data is loaded from the source for further analysis

>

> # DATA VISUALISATION

> # View (df) # This function can be used to understand the how the data is and what is the nature of the data.

>

> # RESULT

> # 10866 observations of 12 variables

>

> # Understanding Data Variable type

>

> str(df) # Shows the data type of the 12 variables. This helps in feature Engineering

'data.frame': 10886 obs. of 12 variables:

$ datetime : chr "1/1/2011 00:00" "1/1/2011 01:00" "1/1/2011 02:00" "1/1/2011 03:00" ...

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

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

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

$ weather : int 1 1 1 1 1 2 1 1 1 1 ...

$ temp : num 9.84 9.02 9.02 9.84 9.84 ...

$ atemp : num 14.4 13.6 13.6 14.4 14.4 ...

$ humidity : int 81 80 80 75 75 75 80 86 75 76 ...

$ windspeed : num 0 0 0 0 0 ...

$ casual : int 3 8 5 3 0 0 2 1 1 8 ...

$ registered: int 13 32 27 10 1 1 0 2 7 6 ...

$ count : int 16 40 32 13 1 1 2 3 8 14 ...

>

> # MANAGE THE DATA [Feature Engineering](Count is our target)

> # The columns "season","holiday","workingday" and "weather" are not "int" data types and should be converted to "categorical" data type.

>

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

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

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

> df$weather = as.factor(df$weather)

>

> # From the "datetime' coloumn, we need to create "date,"hour","weekDay", and "month"

> # Since 'datetime' is a character, we need to convert it to POSIXct to extract components

>

> df$datetime = as.POSIXct(df$datetime, format = "%m/%d/%Y %H:%M")

>

> # Now we can extract the components from the 'datetime' column

>

> df$date = as.Date(df$datetime) # Extracts date

> df$hour = format(df$datetime, "%H") # Extracts hour as a character string

> df$weekday = weekdays(df$date) # Extracts the day of the week

> df$month = months(df$date) # Extracts the name of the month

>

> # The columns "hour", "weekday", and "month" are created as character data types by default.

> # We need to convert "hour" to numeric, and "weekday" and "month" to factor (categorical) data types.

>

> df$hour = as.numeric(df$hour)

> df$weekday = factor(df$weekday)

> df$month = factor(df$month)

>

> # After extracting the relevant data, let us remove the "datetime" column

>

> df$datetime = NULL

>

> # View (df) # This function can be used to check whether the data is accurate and to our liking

>

> # CLEANING THE DATA

>

> # Missing Values Analysis

> total\_missing\_values = sum(is.na(df)) # To check missing values for the entire data set

>

> #RESULT

> # Zero missing values observed

>

> # OUTLIER ANALYSIS

> # Coded with the help of Chatgpt

>

> # For all numerical variables in the dataframe

> # boxplot(df[,sapply(df, is.numeric)], main = "Boxplots for all numeric variables")

>

> # Box Plot 1: Count

> p1 <- ggplot(df, aes(y = count)) + geom\_boxplot() + labs(title = "Box Plot On Count", y = "Count")

> # plot(p1)

>

> # Box Plot 2: Count across Season

> p2 <- ggplot(df, aes(x = season, y = count)) + geom\_boxplot() + labs(title = "Box Plot On Count Across Season", x = "Season", y = "Count")

> # plot(p2)

> # Spring season has the lowest dip in count.

>

> # Box Plot 3: Count across Hour of the Day

> p3 <- ggplot(df, aes(x = factor(hour), y = count)) + geom\_boxplot() + labs(title = "Box Plot On Count Across Hour Of The Day", x = "Hour Of The Day", y = "Count")

> # plot(p3)

> # The median value are relatively higher from 7AM - 8AM and from 5PM - 6PM, and this can be attributed to regular school and office users at that time.

>

> # Box Plot 4: Count across Working Day

> p4 <- ggplot(df, aes(x = factor(workingday), y = count)) + geom\_boxplot() + labs(title = "Box Plot On Count Across Working Day", x = "Working Day", y = "Count")

> # plot(p4)

>

> # Most of the outlier points are contributed from "Working Day" than from "Non Working Day".

>

> # CORRELATION ANALYSIS

> # GENERATE THE HEATMAP TO CALCULATE THE CORRELATION MATRIX

>

> corrMatt <- cor(df[, c("temp", "atemp", "casual", "registered", "humidity", "windspeed", "count")], use = "complete.obs")

>

> # Create a heatmap using corrplot

> corrplot(corrMatt, method = "color", type = "upper", order = "hclust", tl.col = "black", tl.srt = 45, addCoef.col = "black", diag = FALSE)

>

> #RESULT

> # "humidity" has negative correlation with "count"

> # "temp" and "atemp" have positive correlation with "count"

> # "windspeed" is not going be of much use as its correlation count denotes.

>

> # At this point we the value of windspeed for 26 obs is 0. any value below 6 is inputted as 0.

> # There is an option of building a Random Forest Model to predict the 0's in "Windspeed", however, we are ignoring this step as the impact of "windspeed" on count, as stated above, is very minimal.

>

>

> # PARTITION THE DATA

> set.seed(1234)

> N = nrow(df) #counting the total number of rows

> trainingSize = round(N\*0.6) # decide how many rows go to training, here 60%

> trainingCases = sample(N, trainingSize) # simple random sample the rows for training

> training = df[trainingCases, ] #slice training from df

> test = df[-trainingCases, ] #slice test from df

>

> #BUILD THE MODEL

> # MODEL 1 - LINEAR REGRESSION MODEL

> lm\_model = lm(count ~ ., data = training)

> lm\_model = step(lm\_model) # Performing stepwise feature selection to refine the model

Start: AIC=-368502.2

count ~ season + holiday + workingday + weather + temp + atemp +

humidity + windspeed + casual + registered + date + hour +

weekday + month

Step: AIC=-362411.3

count ~ holiday + workingday + weather + temp + atemp + humidity +

windspeed + casual + registered + date + hour + weekday +

month

Warning message:

attempting model selection on an essentially perfect fit is nonsense

> summary(lm\_model) # Displaying a summary of the final linear regression model

Call:

lm(formula = count ~ holiday + workingday + weather + temp +

atemp + humidity + windspeed + casual + registered + date +

hour + weekday + month, data = training)

Residuals:

Min 1Q Median 3Q Max

-6.157e-11 -4.200e-14 -2.000e-15 4.300e-14 3.267e-11

Coefficients:

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

(Intercept) 1.972e-13 9.769e-13 2.020e-01 0.8401

holiday1 -7.719e-14 8.386e-14 -9.200e-01 0.3574

workingday1 -5.036e-13 5.446e-14 -9.248e+00 < 2e-16 \*\*\*

weather2 1.204e-14 2.721e-14 4.420e-01 0.6583

weather3 3.041e-13 4.592e-14 6.624e+00 3.78e-11 \*\*\*

weather4 5.044e-14 8.956e-13 5.600e-02 0.9551

temp 1.846e-15 1.035e-14 1.780e-01 0.8585

atemp 6.967e-15 8.768e-15 7.950e-01 0.4269

humidity -7.544e-16 7.780e-16 -9.700e-01 0.3322

windspeed -1.816e-15 1.504e-15 -1.207e+00 0.2273

casual 1.000e+00 3.428e-16 2.917e+15 < 2e-16 \*\*\*

registered 1.000e+00 9.858e-17 1.014e+16 < 2e-16 \*\*\*

date -2.138e-17 6.384e-17 -3.350e-01 0.7377

hour 1.964e-15 1.850e-15 1.062e+00 0.2884

weekdayMonday 8.017e-14 4.339e-14 1.848e+00 0.0647 .

weekdaySaturday 7.001e-14 6.050e-14 1.157e+00 0.2472

weekdaySunday 7.503e-14 6.762e-14 1.110e+00 0.2672

weekdayThursday 1.020e-13 4.192e-14 2.434e+00 0.0150 \*

weekdayTuesday 6.313e-14 4.196e-14 1.504e+00 0.1325

weekdayWednesday 4.717e-14 4.236e-14 1.113e+00 0.2656

monthAugust -9.993e-16 6.762e-14 -1.500e-02 0.9882

monthDecember -6.043e-15 5.913e-14 -1.020e-01 0.9186

monthFebruary -2.400e-14 5.869e-14 -4.090e-01 0.6826

monthJanuary -1.154e-13 6.172e-14 -1.870e+00 0.0616 .

monthJuly 1.706e-14 6.882e-14 2.480e-01 0.8042

monthJune 2.705e-15 6.166e-14 4.400e-02 0.9650

monthMarch 9.664e-16 5.493e-14 1.800e-02 0.9860

monthMay 4.791e-14 5.697e-14 8.410e-01 0.4003

monthNovember -6.288e-16 5.683e-14 -1.100e-02 0.9912

monthOctober -1.840e-14 5.657e-14 -3.250e-01 0.7450

monthSeptember -1.704e-14 6.129e-14 -2.780e-01 0.7810

---

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

Residual standard error: 8.935e-13 on 6501 degrees of freedom

Multiple R-squared: 1, Adjusted R-squared: 1

F-statistic: 8.921e+30 on 30 and 6501 DF, p-value: < 2.2e-16

>

> ### PREDICTIONS ###

> predictions = predict(lm\_model, test)

>

> # EVALUATE

> observations = test$count #to compare with the true prices

> errors\_lm = observations - predictions # negative is overestimating

>

> ### PERFORMANCE EVALUATION ###

>

> #RMSE - root mean squared error

> rmse\_lm = sqrt(mean(errors\_lm^2))

>

> #MAPE - mean average percentage error

> mape\_lm = mean(abs(errors\_lm/observations))

>

> # BENCHMARKING AGAINST LINEAR REGRESSION MODEL

>

> predictions\_bench = mean(training$count)

> errors\_bench = observations - predictions\_bench

> rmse\_bench = sqrt(mean(errors\_bench^2))

> mape\_bench = mean(abs(errors\_bench/observations))

>

>

> # MODEL 2 - RANDOM FOREST MODEL

> ### BAGGING ###

> # Multiple decision trees are built and their results are averaged.

>

> rf\_model = randomForest(count ~., data = training, ntree = 500)

> pred\_rf = predict(rf\_model, test)

>

> #EVALUATE

> errors\_rf = test$count - pred\_rf

>

> ### PERFORMANCE EVALUATION ###

>

> #RMSE - root mean squared error

> rmse\_rf = sqrt(mean(errors\_rf^2))

>

> #MAPE - mean average percentage error

> mape\_rf = mean(abs(errors\_rf/test$count))

>

>

> # MODEL 3 - NEURAL NETWORK MODEL

> ### BOOSTING IN R ###

>

> #STANDARDIZING FOR NEURAL NETWORK MODE

> standardizer = preProcess(training, method = c("center", "scale"))

> training\_scaled = predict(standardizer, training)

> test\_scaled = predict(standardizer, test)

>

> # BUILDING THE MODEL

> set.seed(1234)

> nn\_model = nnet(count ~ ., data = training\_scaled, size = 10)

# weights: 351

initial value 9696.792538

final value 6531.000000

converged

> predictions\_nn = predict(nn\_model, test\_scaled, type = "raw")

>

> # Evaluate

> errors\_nn = test$count - predictions\_nn

>

> ### PERFORMANCE EVALUATION ###

>

> rmse\_nn = sqrt(mean(errors\_nn^2))

> mape\_nn = mean(abs(errors\_nn/test$count))

>

>

> # MANAGER MODEL

>

> # Predictions on training data for stacking

>

> pred\_lm\_train = predict(lm\_model, df)

> pred\_rf\_train = predict(rf\_model, df)

> pred\_nn\_train = predict(nn\_model, df, type = "raw")

>

> df\_stack = cbind(df, pred\_lm\_train, pred\_rf\_train, pred\_nn\_train)

> df\_stack = as.data.frame(df\_stack)

> training\_stack = df\_stack[trainingCases, ]

> test\_stack = df\_stack[-trainingCases, ]

>

> stack = lm(count ~., data = training\_stack)

> pred\_stack = predict(stack, test\_stack)

>

> # Evaluate

> error\_stack = test$count[-trainingCases] - pred\_stack # error stack

Warning message:

In test$count[-trainingCases] - pred\_stack :

longer object length is not a multiple of shorter object length

>

> ### PERFORMANCE EVALUATION ###

>

> rmse\_stack = sqrt(mean(error\_stack^2))

> mape\_stack = mean(abs(error\_stack/test$count))

> save.image("F:/2. Babson/1. Curriculam + Studies/3. Sem 3/1. Course Work/Machine Learning/Clarification phase 3/Workspace Phase 3.RData")