Data Mining- Group Project – Weather Data Prediction

Janakiram Kommaraju, Kalyani Jajula, Sai Sree Mithra Sripathi - Group 8

3/20/2021

**Background:**

Weather forecasting is the application of science and technology to predict the conditions of the atmosphere for a given location and time. Data Mining is an unsupervised learning approach i.e. it doesn’t require any training. The most commonly used techniques of data mining in weather forecasting are, Artificial Neural Networks, Genetic Algorithms, Clustering and Memory-Based Reasoning, Decision Trees, Logistic Regression, Discriminant Analysis and Decision Trees. Weather Forecasting is done based on certain Weather parameters like maximum temperature, minimum temperature, extent of rainfall, cloud conditions, wind streams and their directions, projected using images taken by the meteorological satellites. Data Mining helps in identifying or assesses future trends.

**Introduction:**

Weather condition patterns for any one region or for the whole planet can be charted on a weather map containing information about factors like- air temperature, pressure, humidity, cloud cover, precipitation and wind.

In this project, we took a data set – Weather.csv , which contains 23 columns related to weather forecasting. We filtered 7 columns of that which have more impact on prediction and applied data mining techniques –

1. Logistic Regression,
2. Random Forest and
3. KNN

To brief about these techniques:

**Logistic Regression with glm:**

The Logistic Regression is a regression model in which the response variable (dependent variable) has categorical values such as True/False or 0/1. It actually measures the probability of a binary response as the value of response variable based on the mathematical equation relating it with the predictor variables.

The steps followed for logistic regression are:

1. Import the data
2. Check for class bias
3. Create training and test samples
4. Compute information value to find out important variables
5. Build logit models and predict on test data
6. Do model diagnostics

**Random Forest:**

The random forest algorithm works by aggregating the predictions made by multiple decision trees of varying depth. Every decision tree in the forest is trained on a subset of the dataset called the bootstrapped dataset

We used the caTools package to split our data into training and tests sets as well as the random forest classifier provided by the randomForest package.

**KNN:**

KNN which stand for K Nearest Neighbor is a Supervised Machine Learning algorithm that classifies a new data point into the target class, depending on the features of its neighboring data points.

The KNN algorithm has the following features:

1. KNN is a Supervised Learning algorithm that uses labeled input data set to predict the output of the data points.
2. It is mainly based on feature similarity. KNN checks how similar a data point is to its neighbor and classifies the data point into the class it is most similar to.
3. Unlike most algorithms, KNN is a non-parametric model which means that it does not make any assumptions about the data set. This makes the algorithm more effective since it can handle realistic data.
4. KNN is a lazy algorithm, this means that it memorizes the training data set instead of learning a discriminative function from the training data.
5. KNN can be used for solving both classification and regression problems.

**Challenges in implementing:**

* The major problem was with the size of data. The initial dataset was very huge and running any functions on it would block the system. So we had to reduce it to 45000+ rows of data and also picked specific columns to implement the models.
* Including the necessary packages while running models is also a choice to be made.
* The data was noisy and some columns contained many NA values, all that was cleaned before running the models to get accurate weather prediction.
* Learning to run the models, their confusion matrices and accuracy was also a challenge.
* There is a lot which can be analysed from such data sets and we had to specific on what we wanted to derive.

**Implementation and Results:**

The packages needed for running the models on weather data are:

#list.files(path = "../input")  
#install.packages("GGally")  
#install.packages("InformationValue")  
#install.packages("scatterplot3d")  
#install.packages("randomForest")  
  
library(tidyverse)

library(gridExtra)

library(MASS)

require(scatterplot3d)

library(InformationValue)  
library(ggplot2)  
require(GGally)

library(dplyr)  
library(randomForest)

Read the data from weather.csv using ‘read.csv’ and load it in data variable.

Perform operations like head, str, summary to know about the columns in the data and values. Separating 7 required columns – MinTemp, MaxTemp, Humidity9am, Humidity3pm, Pressure9am, Pressure3pm, RainTomorrow to make the data easier to analyze.

#read data  
data <- read.csv("weather.csv")  
  
#information about the data  
head(data)

## Date Location MinTemp MaxTemp Rainfall Evaporation Sunshine WindGustDir  
## 1 12/1/2008 Albury 13.4 22.9 0.6 NA NA W  
## 2 12/2/2008 Albury 7.4 25.1 0.0 NA NA WNW  
## 3 12/3/2008 Albury 12.9 25.7 0.0 NA NA WSW  
## 4 12/4/2008 Albury 9.2 28.0 0.0 NA NA NE  
## 5 12/5/2008 Albury 17.5 32.3 1.0 NA NA W  
## 6 12/6/2008 Albury 14.6 29.7 0.2 NA NA WNW  
## WindGustSpeed WindDir9am WindDir3pm WindSpeed9am WindSpeed3pm Humidity9am  
## 1 44 W WNW 20 24 71  
## 2 44 NNW WSW 4 22 44  
## 3 46 W WSW 19 26 38  
## 4 24 SE E 11 9 45  
## 5 41 ENE NW 7 20 82  
## 6 56 W W 19 24 55  
## Humidity3pm Pressure9am Pressure3pm Cloud9am Cloud3pm Temp9am Temp3pm  
## 1 22 1007.7 1007.1 8 NA 16.9 21.8  
## 2 25 1010.6 1007.8 NA NA 17.2 24.3  
## 3 30 1007.6 1008.7 NA 2 21.0 23.2  
## 4 16 1017.6 1012.8 NA NA 18.1 26.5  
## 5 33 1010.8 1006.0 7 8 17.8 29.7  
## 6 23 1009.2 1005.4 NA NA 20.6 28.9  
## RainToday RainTomorrow  
## 1 No No  
## 2 No No  
## 3 No No  
## 4 No No  
## 5 No No  
## 6 No No

data <- data[,c(3,4,14,15,16,17,23)]  
head(data)

## MinTemp MaxTemp Humidity9am Humidity3pm Pressure9am Pressure3pm RainTomorrow  
## 1 13.4 22.9 71 22 1007.7 1007.1 No  
## 2 7.4 25.1 44 25 1010.6 1007.8 No  
## 3 12.9 25.7 38 30 1007.6 1008.7 No  
## 4 9.2 28.0 45 16 1017.6 1012.8 No  
## 5 17.5 32.3 82 33 1010.8 1006.0 No  
## 6 14.6 29.7 55 23 1009.2 1005.4 No

nrow(data)

## [1] 45584

ncol(data)

## [1] 7

str(data)

## 'data.frame': 45584 obs. of 7 variables:  
## $ MinTemp : num 13.4 7.4 12.9 9.2 17.5 14.6 14.3 7.7 9.7 13.1 ...  
## $ MaxTemp : num 22.9 25.1 25.7 28 32.3 29.7 25 26.7 31.9 30.1 ...  
## $ Humidity9am : int 71 44 38 45 82 55 49 48 42 58 ...  
## $ Humidity3pm : int 22 25 30 16 33 23 19 19 9 27 ...  
## $ Pressure9am : num 1008 1011 1008 1018 1011 ...  
## $ Pressure3pm : num 1007 1008 1009 1013 1006 ...  
## $ RainTomorrow: chr "No" "No" "No" "No" ...

summary(data)

## MinTemp MaxTemp Humidity9am Humidity3pm   
## Min. :-4.80 Min. : 6.30 Min. : 3.00 Min. : 1.00   
## 1st Qu.: 9.10 1st Qu.:19.30 1st Qu.: 58.00 1st Qu.: 37.00   
## Median :13.80 Median :23.20 Median : 71.00 Median : 53.00   
## Mean :13.22 Mean :23.73 Mean : 69.85 Mean : 52.79   
## 3rd Qu.:17.80 3rd Qu.:27.30 3rd Qu.: 83.00 3rd Qu.: 67.00   
## Max. :29.70 Max. :47.30 Max. :100.00 Max. :100.00   
## NA's :517 NA's :383 NA's :707 NA's :1370   
## Pressure9am Pressure3pm RainTomorrow   
## Min. : 980.5 Min. : 979 Length:45584   
## 1st Qu.:1013.8 1st Qu.:1011 Class :character   
## Median :1018.4 Median :1016 Mode :character   
## Mean :1018.3 Mean :1016   
## 3rd Qu.:1022.8 3rd Qu.:1020   
## Max. :1039.9 Max. :1038   
## NA's :6754 NA's :6713

The data contains ‘NA’ values in all the columns and the count of NA’s in each column can be seen in the plot attached in the end.

To remove all this noisy value, replacing all NA values with median values of than column.

#missing values in each column counts  
colSums(is.na(data))

## MinTemp MaxTemp Humidity9am Humidity3pm Pressure9am Pressure3pm   
## 517 383 707 1370 6754 6713   
## RainTomorrow   
## 1236

#Cleaning data  
data$MinTemp <- data$MinTemp %>%   
 replace\_na(median(data$MinTemp, na.rm=TRUE))   
data$MaxTemp <- data$MaxTemp %>%   
 replace\_na(median(data$MaxTemp, na.rm=TRUE))   
data$Humidity9am <- data$Humidity9am %>%   
 replace\_na(median(data$Humidity9am, na.rm=TRUE))   
data$Humidity3pm <- data$Humidity3pm %>%   
 replace\_na(median(data$Humidity3pm, na.rm=TRUE))  
data$Pressure9am <- data$Pressure9am %>%   
 replace\_na(median(data$Pressure9am, na.rm=TRUE))   
data$Pressure3pm <- data$Pressure3pm %>%   
 replace\_na(median(data$Pressure3pm, na.rm=TRUE))

The data is now cleaned and the summary makes it clear on which values are present in each column.  
#after cleaning  
summary(data)

## MinTemp MaxTemp Humidity9am Humidity3pm   
## Min. :-4.80 Min. : 6.30 Min. : 3.00 Min. : 1.0   
## 1st Qu.: 9.20 1st Qu.:19.30 1st Qu.: 58.00 1st Qu.: 38.0   
## Median :13.80 Median :23.20 Median : 71.00 Median : 53.0   
## Mean :13.23 Mean :23.73 Mean : 69.86 Mean : 52.8   
## 3rd Qu.:17.80 3rd Qu.:27.30 3rd Qu.: 83.00 3rd Qu.: 67.0   
## Max. :29.70 Max. :47.30 Max. :100.00 Max. :100.0   
## Pressure9am Pressure3pm RainTomorrow   
## Min. : 980.5 Min. : 979 Length:45584   
## 1st Qu.:1014.7 1st Qu.:1012 Class :character   
## Median :1018.4 Median :1016 Mode :character   
## Mean :1018.3 Mean :1016   
## 3rd Qu.:1022.0 3rd Qu.:1019   
## Max. :1039.9 Max. :1038

colSums(is.na(data))

## MinTemp MaxTemp Humidity9am Humidity3pm Pressure9am Pressure3pm   
## 0 0 0 0 0 0   
## RainTomorrow   
## 1236

#plot the frequencies after cleaning

par(mfrow=c(2,3))

data$MinTemp %>% hist(main='MinTemp')

data$MaxTemp %>% hist(main='MaxTemp')

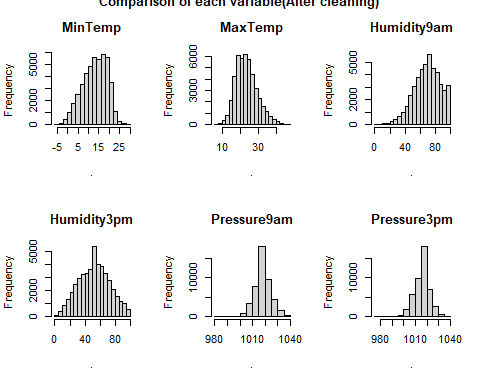
data$Humidity9am %>% hist(main='Humidity9am')

data$Humidity3pm %>% hist(main='Humidity3pm')

data$Pressure9am %>% hist(main='Pressure9am')

data$Pressure3pm %>% hist(main='Pressure3pm')

title(main = 'Comparison of each variable(After cleaning)', outer=TRUE)



#plots considering multiple components

data %>%

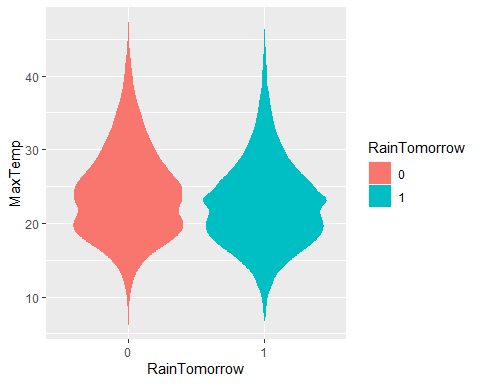
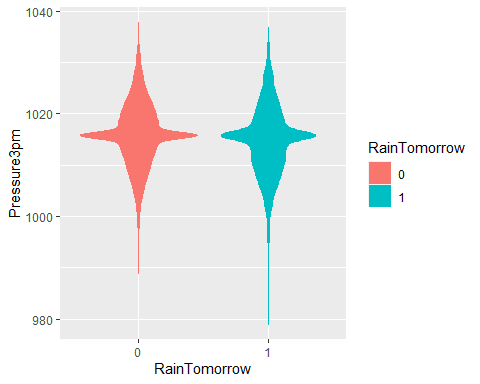
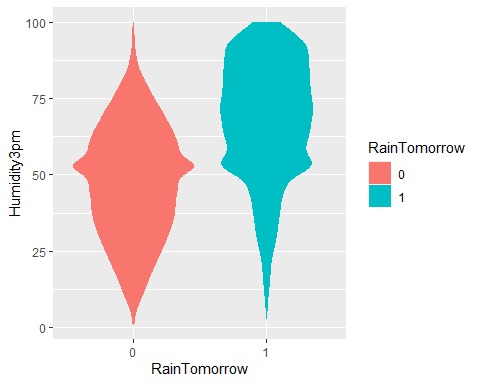
ggplot(aes(x=RainTomorrow, y=Humidity3pm, colour = RainTomorrow, fill= RainTomorrow)) + geom\_violin()

data %>%

ggplot(aes(x=RainTomorrow, y=Pressure3pm, colour = RainTomorrow, fill= RainTomorrow)) + geom\_violin()

data %>%

ggplot(aes(x=RainTomorrow, y=MaxTemp, colour = RainTomorrow, fill= RainTomorrow)) + geom\_violin()



The target column RainTomorrow contains yes/no values. But to make it numeric, changing the value of yes to ‘1’ and no to ‘0’. Also replaced all the ‘NA’ values as’1’ , presuming rain prediction of yes is better for data which is not available.

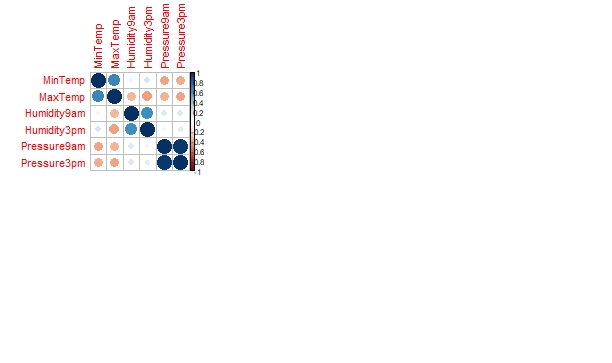
#changing yes/no in 'RainTomorrow' to 0/1  
data$RainTomorrow<-replace(data$RainTomorrow,data$RainTomorrow=='No',0)  
data$RainTomorrow<-replace(data$RainTomorrow,data$RainTomorrow=='Yes',1)  
data$RainTomorrow <- data$RainTomorrow %>%   
 replace\_na(1)

#corplot for considered columns

library(corrplot)

correlations <- cor(data[,1:6])

corrplot(correlations, method="circle")



Now the major task of splitting the data into train and test sets is done. The ratio of train and test sets are 80:20. After splitting the train set contains 36468 rows and test set contains 9116 rows.  
  
#Splitting dataset into training and testset.  
library(caTools)  
set.seed(123)  
split = sample.split(data$RainTomorrow, SplitRatio = 0.80)  
training\_set = subset(data, split == TRUE)  
test\_set = subset(data, split == FALSE)  
nrow(training\_set)

## [1] 36468

nrow(test\_set)

## [1] 9116

names(test\_set)

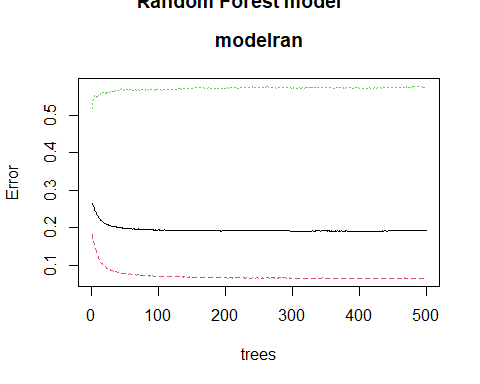
## [1] "MinTemp" "MaxTemp" "Humidity9am" "Humidity3pm" "Pressure9am"   
## [6] "Pressure3pm" "RainTomorrow"

Random Forest:

#Random Forest  
modelran <- randomForest(factor(RainTomorrow) ~ ., data = training\_set,  
 ntree = 500, mtry = 6,importance = TRUE)  
modelran

##   
## Call:  
## randomForest(formula = factor(RainTomorrow) ~ ., data = training\_set, ntree = 500, mtry = 6, importance = TRUE)   
## Type of random forest: classification  
## Number of trees: 500  
## No. of variables tried at each split: 6  
##   
## OOB estimate of error rate: 19.16%  
## Confusion matrix:  
## 0 1 class.error  
## 0 25654 1784 0.06501932  
## 1 5204 3826 0.57630122

plot(modelran)  
title(main = 'Random Forest model', outer=TRUE)



#prediction using the model  
predictions <- predict(modelran,test\_set,type="class")  
#Confusion matrix  
confusionMatrix(predictions,test\_set$RainTomorrow)

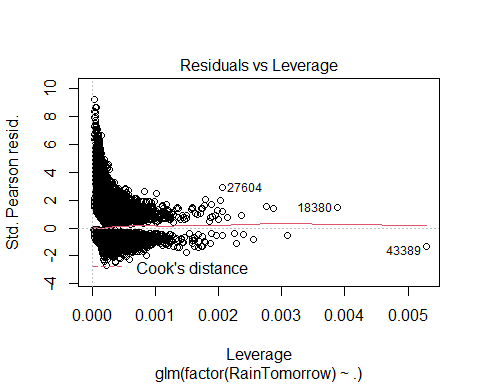
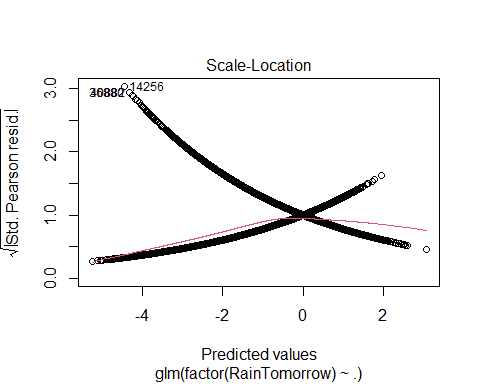
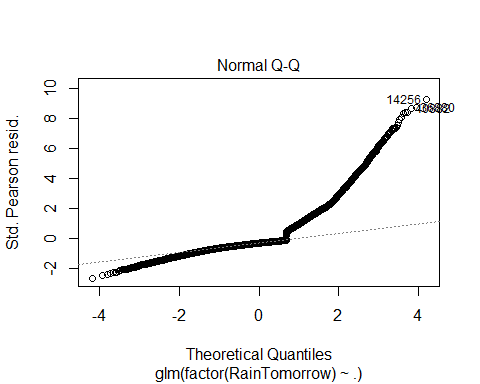
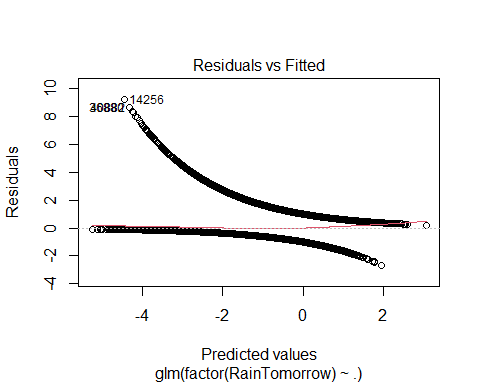
## 0 1  
## 0 6396 463  
## 1 1275 982

#calculating accuracy of RF model  
CM\_forest<-table(predictions,test\_set$RainTomorrow)  
accuracy\_forest<-(sum(diag(CM\_forest)))/sum(CM\_forest)  
accuracy\_forest

## [1] 0.8093462

**Logistic Regression:**

#Logisitic regression  
classifier = glm(factor(RainTomorrow) ~ .,  
 family = binomial,  
 data = training\_set)  
plot(classifier)



#predicton using the model  
prob\_pred = predict(classifier, type = 'response', newdata = test\_set[,1:ncol(training\_set)-1])  
logi\_pred = ifelse(prob\_pred > 0.5, 1, 0)  
#Confusion matrix   
confusionMatrix(logi\_pred, test\_set$RainTomorrow)

## 0 1  
## 0 6437 422  
## 1 1367 890

#calculating accuracy of logit  
table(logi\_pred>0.5,test\_set$RainTomorrow)

##   
## 0 1  
## FALSE 6437 1367  
## TRUE 422 890

CM\_logit<-table(logi\_pred>0.25,test\_set$RainTomorrow)  
CM\_logit

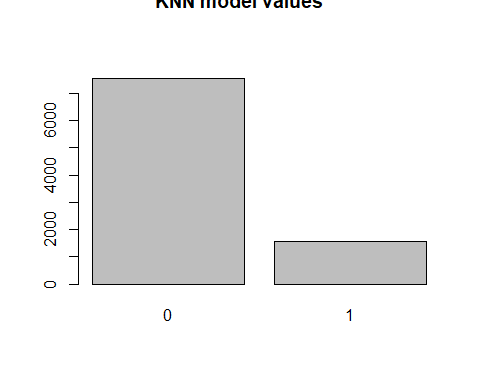
##   
## 0 1  
## FALSE 6437 1367  
## TRUE 422 890

accuracy\_logit<-(sum(diag(CM\_logit)))/sum(CM\_logit)  
accuracy\_logit

## [1] 0.8037516

**KNN:**

#knn  
library(class)  
knn1 <- knn(train = training\_set[,-23],   
 test = test\_set[,-23],   
 cl = training\_set$RainTomorrow,  
 k = 5)  
plot(knn1)  
title(main = 'KNN model values', outer=TRUE)



#Confusion Matrix  
confusionMatrix(knn1, test\_set$RainTomorrow)

## 0 1  
## 0 6394 465  
## 1 1147 1110

knnAccuracy <- 100 \* sum(test\_set$RainTomorrow == knn1)/NROW(test\_set$RainTomorrow)  
knnAccuracy

## [1] 82.31681

#comparing accuracies  
print("Random Forest Accuracy :")

## [1] "Random Forest Accuracy :"

100\*accuracy\_forest

## [1] 80.93462

print("Logistic Regression Accuracy :")

## [1] "Logistic Regression Accuracy :"

100\*accuracy\_logit

## [1] 80.37516

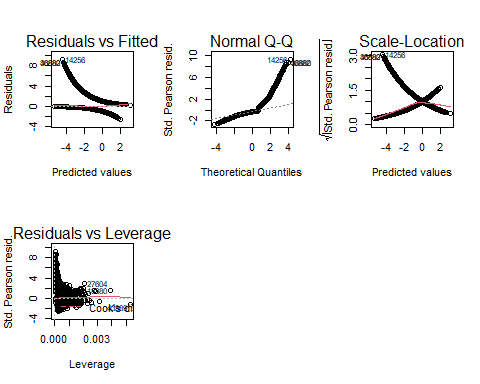
print("knn Accuracy :")

## [1] "knn Accuracy :"

knnAccuracy

## [1] 82.31681

write.csv(training\_set, file = 'Weather\_train.csv')  
write.csv(test\_set, file = 'Weather\_test.csv')



**Group Tasks:**

1. **Janakiram Kommaraju –** Reducing data size, data cleaning and analysis, data visualizations.
2. **Kalyani Jajula –** implementing modelsLogistic Regression, Random Forest and visualizations
3. **Sai Sree Mithra Sripathi -** document preparation, implementing model knn and calculating model accuracies.

**References:**

<https://www.tutorialspoint.com/r/r_logistic_regression.htm>

<https://www.edureka.co/blog/knn-algorithm-in-r/>

<https://rstudio-pubs-static.s3.amazonaws.com/316172_a857ca788d1441f8be1bcd1e31f0e875.html>

<https://www.datacamp.com/community/tutorials/logistic-regression-R>

<https://www.r-bloggers.com/2018/01/how-to-implement-random-forests-in-r/>

<https://www.datacamp.com/community/tutorials/make-histogram-basic->r

<https://www.analytics-tuts.com/how-to-split-train-and-test-data-in-r/>