Machine Learning Portfolio

By Sajid Ahmed

Introduction

This repository contains my portfolio of Machine learning projects, completed for self-learning and hobby purposes. It demonstrates my knowledge and enthusiasm to learn about the different fundamentals of Machine Learning, collated in a single file. If you have further enquiries, please feel free to contact me through my linkedin: https://www.linkedin.com/in/sajid-ahmed-b110a014a/

Content

Principal Component Analysis (PCA)

PCA is a very popular method to quickly capture the essence of the dataset through a few principal components. The most popular method is 'Dimensionality Reduction', used to reduce the dimension of the data.

Firstly, I decided to use the dataset 'airquality', which is a pre-loaded dataset. It has 153 observations of daily air quality measurements in New York from May to September 1973.

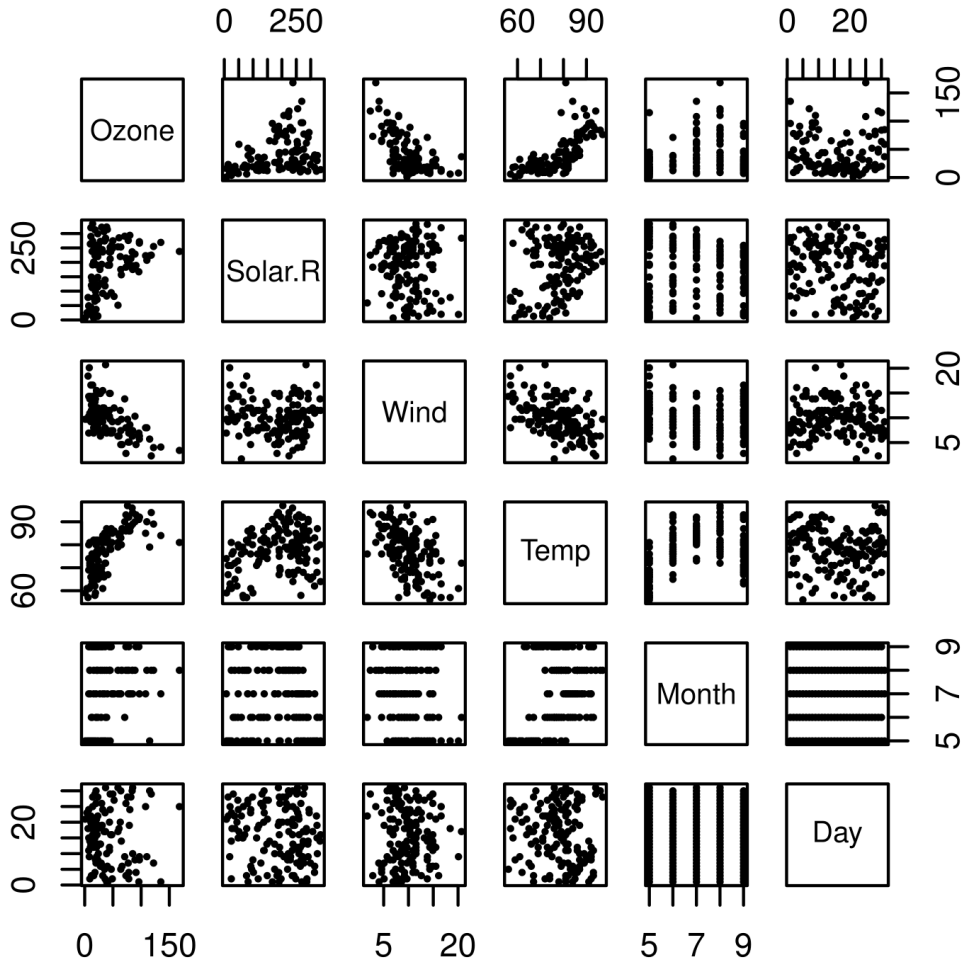
```
data(airquality); summary(airquality)

##      Ozone           Solar.R          Wind             Temp      
##  Min.   :  1.00   Min.   :  7.0   Min.   : 1.700   Min.   :56.00  
##  1st Qu.: 18.00   1st Qu.:115.8   1st Qu.: 7.400   1st Qu.:72.00  
##  Median : 31.50   Median :205.0   Median : 9.700   Median :79.00  
##  Mean   : 42.13   Mean   :185.9   Mean   : 9.958   Mean   :77.88  
##  3rd Qu.: 63.25   3rd Qu.:258.8   3rd Qu.:11.500   3rd Qu.:85.00  
##  Max.   :168.00   Max.   :334.0   Max.   :20.700   Max.   :97.00  
##  NA's   :37       NA's   :7                                       
##      Month            Day      
##  Min.   :5.000   Min.   : 1.0  
##  1st Qu.:6.000   1st Qu.: 8.0  
##  Median :7.000   Median :16.0  
##  Mean   :6.993   Mean   :15.8  
##  3rd Qu.:8.000   3rd Qu.:23.0  
##  Max.   :9.000   Max.   :31.0  
##                                
```

Table 1. Summary of Airquality dataset (includes all statistical measures)

Using the summary command, the initial dataset had a few missing entries.

```
pairs(airquality,pch=16,cex=0.5);   var(airquality)
```




```
##             Ozone Solar.R        Wind        Temp       Month         Day
## Ozone          NA      NA          NA          NA          NA          NA
## Solar.R        NA      NA          NA          NA          NA          NA
## Wind           NA      NA  12.4115385 -15.272136 -0.8897532   0.8488519
## Temp           NA      NA -15.2721362  89.591331  5.6439628 -10.9574303
## Month          NA      NA  -0.8897532   5.643963  2.0065359  -0.0999742
## Day            NA      NA   0.8488519 -10.957430 -0.0999742  78.5797214
```

Figure 1. Pairs plot of Airquality

The missing entries proved a problem when computing the variance-covariance matrix. I decided to apply some **data cleaning** to remove the empty entries. The two variables that will not be affected for cleaning are Month and Day.

```
sum(complete.cases(airquality))/nrow(airquality)*100

## [1] 72.54902

X<-airquality[complete.cases(airquality),]
X<-X[,-c(5:6)];  X<-scale(x=X,center=TRUE,scale=FALSE)
```

The command 'complete.cases' determines which are the values to be removed and create a new variable. 72% of the data was complete before the removal of missing entries. Another important step of **data cleaning** is to center/feature scale the data, as it helps normalise the data to a particular range. It also helps speed up calculations in an algorithm. **Data cleaning** is essential to prepare for Principal Component Analysis.

```
VX<-var(X);   E<-eigen(VX);    E$values/sum(E$values)*100

## [1] 88.98078028 10.47105443  0.46819497  0.07997032
```

Table 2. Proportion of total variability

Computing the proportional contribution of each eigenvalue to the total variability, we can see that almost 90% of the total variability in the dataset is accounted for from the first eigenvalue.

```
E$vectors

##              [,1]        [,2]         [,3]         [,4]
## [1,] -0.143014512  0.96704882 -0.202729051  0.057134580
## [2,] -0.989119298 -0.14698335 -0.004561518 -0.004254707
## [3,]  0.006117191 -0.06845434 -0.050046167  0.996379428
## [4,] -0.033947672  0.19628163  0.977944531  0.062813790
```

Table 3. Eigenvectors of Airquality

Looking at the coefficients of each component, we can see the first component (Ozone) is almost entirely due to the second variable (Solar.R) and vice versa.

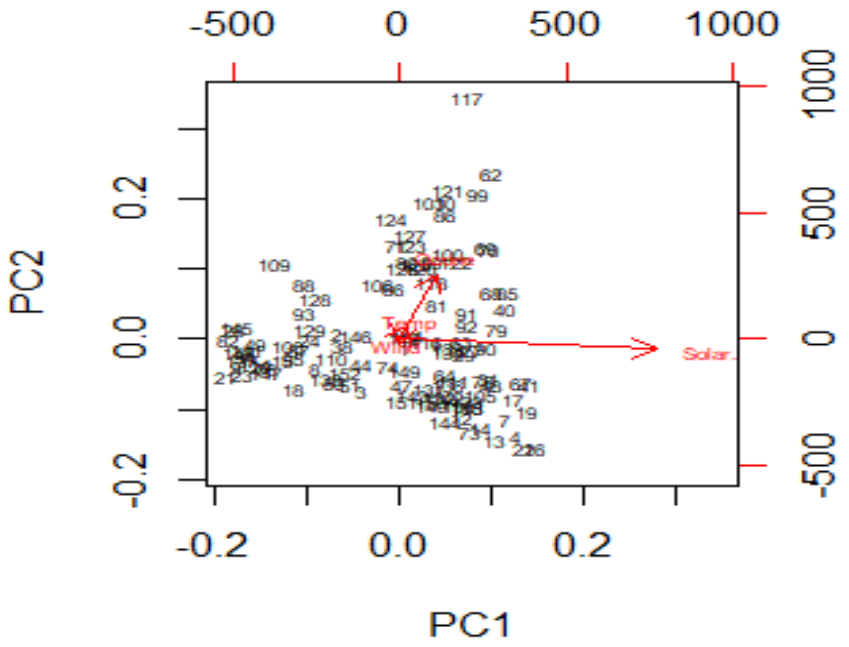


Figure 3. Biplot of Airquality

Data visualization is a key process which takes raw information and places it into a visual context, making data easier to understand and detect patterns/trends. Using the biplot, we can see two key variables in the dataset that hold the highest weight (Solar.R and Ozone). This means Solar.R and Ozone capture the essence of the data in a few principle components, which convey the most variation in the dataset.

Agglomerative Clustering

Clustering is a process of grouping sets of objects that are in a similar manner of each other than those in other groups. This is a sequential procedure to cluster n individuals, which can be summarized in a dendogram.

1. Initially, each individual is its own cluster
2. At every step, an updated table of distances between clusters is considered and minimised to merge an individual to a cluster
3. At the end of the procedure, a single cluster is achieved

To begin, I used the 'Ruspini' dataset which can be found in the 'Cluster' library package in R Studio. This dataset comprises 75 observations on 2 variables (x and y).

```
A<-agnes(x=ruspini,method = "single",metric="euclidean")
plot(A,which.plot=2,cex=0.5)
```

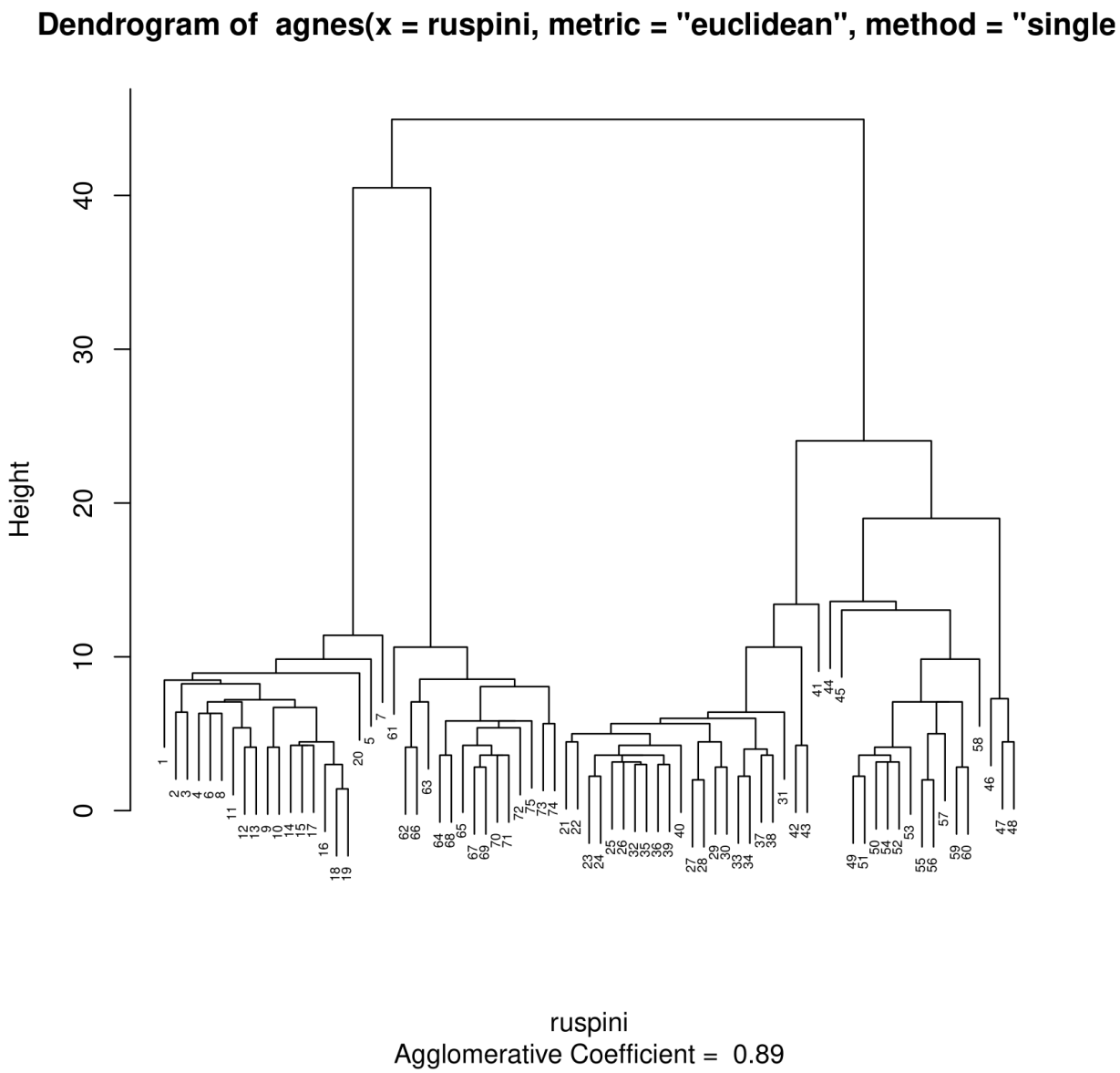


Figure 4. Dendrogram of Ruspini dataset using euclidean distance and single linkage

A dendrogram is a great **data visualisation** tool to understand the groups of clusters that can be formed. Looking at this figure, we want to choose the number of clusters by being

as close to the bottom and maintaining as many clusters as possible. From the look of the dendrogram, I would choose around 17. We would have 5 clusters.

Predicting customer default probability

This is a practical example of using Machine Learning regressors to help predict if a certain customer would default on their credit or not. The dataset is taken from the R Studio package 'ISLR'. The dataset comprises 10000 observations with 4 variables (default, student, balance, income).

```
library(cvTools)
set.seed(0)
n<-10000; K<-5
cvFolds(n=n,K=K)->CV
```

```
CV$subsets[CV$which!=5]->Train
CV$subsets[CV$which==5]->Test
```

We will first create a set of indices for an 80/20 data split of data that will be used for later modelling. Now we create the folds by running the function cvFolds for a split of n = 10000 values in K = 5 folds. We split the data into 5 folds, 4 for training the models and 1 to test the models.

The variable CV has two elements that can be used for the split. CV$subsets contain a random permutation of the data indices 1-10000, and the second element is CV$which which indexes the five folds.

```
library(GGally)
ggpairs(Default,ggplot2::aes(colour=Default$default))
```

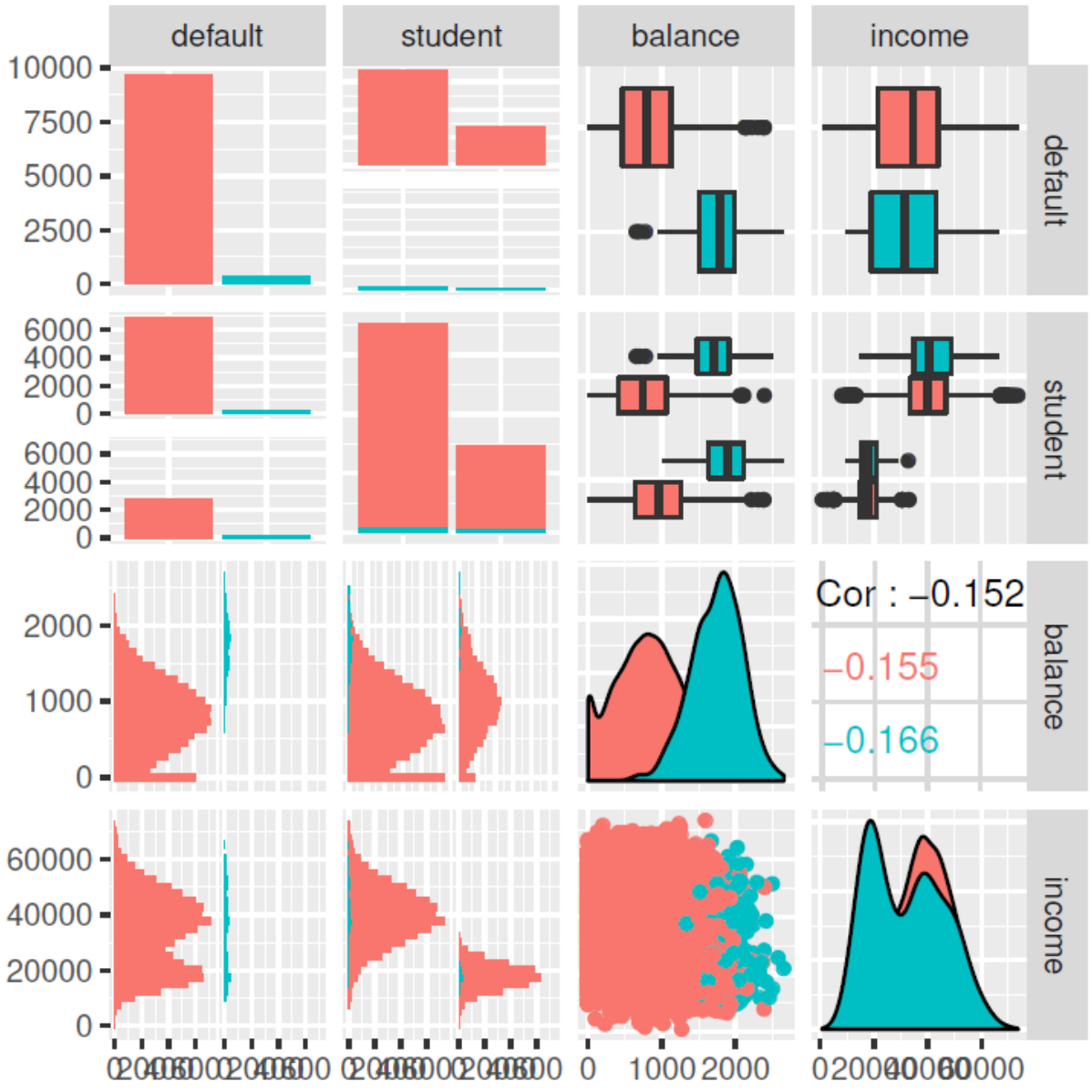


Figure 4. Ggplot of default dataset

As a self-learner, I researched on different **data visualisation** techniques that can be used to describe data distributions. I discovered this blog [1] that uses ggplot from the library

[1] https://www.blopig.com/blog/2019/06/a-brief-introduction-to-ggpairs/

'GGally' to illustrate different distributions. This plot has a very clear wow factor but also shows some very interesting details about the default dataset. The response variable default and the variable student are categorical, which means that they can be classified as one or the other (default or no default/student or no student). We also see a small proportion of persons in the database that have defaulted. It may be that balance can predict default, but no clear indication that students are more likely to default than non-students.

```
M1<-glm(default~balance,family = "binomial",data = Default[Train,])
M2<-glm(default~balance+student,family = "binomial",data = Default[Train,])
M3<-glm(default~.,family = "binomial",data = Default[Train,])
```

```
predict.glm(object = M1, newdata =Default[Test,],type="response" ) -> P1
predict.glm(object = M2, newdata =Default[Test,],type="response" ) -> P2
predict.glm(object = M3, newdata =Default[Test,],type="response" ) -> P3
```

I decided to fit three models: M1 which predicts default as a function of balance, M2 which predicts defaults as a function of balance and student, M3 which predicts default as a function of all the variables. I used the glm command which stands for standardised linear model. The predict glm obtains predictions and optimally estimates standard errors of those predictions.

```
Ytrue<-Default[Test,]$default=="Yes"
Y1<-P1>0.5
Y2<-P2>0.5
Y3<-P3>0.5
table(Ytrue,Y1) ## For M1

##         Y1
## Ytrue   FALSE TRUE
##   FALSE  1933    7
##   TRUE     39   21
```

Data analytics is a very crucial process for information, as it takes raw data and turns it into context, perspective and useful insights. One major analytic for regression models is to build a confusion matrix. This matrix allows us to measure the performance of classifiers through: True positive rate (TPR) and False positive rate (FPR). The TPR and FPR for model M1 are 0.35 and 0.0036 respectively. This means that the model M1 is a good model as it's

proportion of negatives wrongly classified as positive is really low and the proportion of positives correctly classified as positives is okay.