Managing Big Data

First Assignment

Peter Pan

03 / 11 / 2022

Contents

1 Subject 2

The paper “A new method of identifying key industries: a principal component analysis” details the enforcement of the method of PCA to three benchmark input–output tables of the USA of the years 2002, 2007 , 2012 and 2019 in order to cut back on data and analyse them more efficient and easier and the dimensions of the tables are reduced to 70 $\times$ 70. The datasets on input–output tables are available from the website of the BEA. . The rationale for the application of this method is to skew distribution of the economic system’s eigenvalues and the wide gap of the second from the maximal because the first eigenvalue along with the second compress most of the information regulating the economic system’s motion.The number of PC to be selected depending on the so-called "elbow rule”,according to which the usually selected percentage explained by the PC should be around 70%. the first two eigenvalues are much higher than the rest and from the third eigenvalue onwards starts the decaying of eigenvalues. , we do not improve our overall explanation or variability, and certainly, we do not affect qualitatively our results.The application of the PC analysis in input–output data consists with four steps. The application of the PC analysis in input–output data consists with four steps.
Clustering is a very useful tool in datasets. Especially algorithms use statistical methods that are considered as probability and non-parametric model-based approaches. The first statistical method used in pca is a mixture likelihood approach to clustering at the first approach. The Expectation and Maximization (EM) algorithm is the most frequently used in model-based. next it seems to use k$-$means ,and Silhouette value. This method is used in order to approach in deriving the key industries of the US economy in 4 years quite apart from each other. This measure ranges from -1 to $+$ 1 with higher values better match to its own cluster, whereas low indicate poorly matched values to neighboring clusters.In order to categorized the 70 industries into three clusters they use a method of partitioning n observations into k clusters in which each industry is assigned to a particular cluster according to the nearest mean or cluster centroid around which industries are crowded. The property of k$-$means clustering is that it minimizes within$-$cluster variances or Euclidean distances.Furthermore PCA use two more methods BL and FL with the aim of grouping industries into particular clusters. They prefer to use the method of PCA cause it captures better than any other parsimonious method the variance associated with each of the industries, enables the classification of industries according to their positive distance from zero, and the ranking using the average of both BL and FL is quite close to that of the PCA. A salient feature of the PCA is that clustering enables the presentation of the industries in a dendrogram, which marks the last step in cluster analysis and displays a grouping of industries into distinct clusters in order to decide upon the suitable number of clusters .In our case its observed a hierarchical location of the industries ranked according to the height of the branches, the nodes that are higher indicate the importance of each cluster and within the same cluster the importance of the industry. Clusters are ,S$-$E which include industries with importance, the second in importance is N$-$E and all the others that haven’t much importance.
Eventually , the use of PCA seemed to be one-way, since PCA possesses distinct advantages compared to the standard BL and FL methods. PCA was able to separate the data and group the industries into three basic categories. In this

case , the top two PCs perpendicular to each other, meaning their correlation is zero, grouped the data into three particular clusters for each of the four distant years with the k-means and Silhouette procedures. It reveals understanding of the interconnections of industries and changes in the structure of the economies whish is extremely useful in the planning of an effective industrial policy.

2 Subject 3

2.1 Exercise in R

Initially, we install a library named “MASS”. There, we found the dataset named “Cars93”. By observing “Cars93” we realize that many of the variables were discrete variables (Cylinders column). In order to run the PCA method it is necessary to remove this column.
We compile the variable : pcaData = Cars93[,c(4,5,6,7,8,12,13,14,15,17,18,19,20,21,22,23,24,25)] ,which includes only numerical variables. Afterwards, we took a closer look to our data to see if we have NA values. For this reason we run the following command:

```
if (any(is.na(pcaData[,"Min.Price"]))) {
        sprintf("Min.Price␣has␣NA␣values")
} else
sprintf("Min.Price␣is␣ok!␣")
```

for every column in our dataset. We observe that two columns have NA values and so we remove them with the command:
pcaData = pcaData[,-(16:17)]
str(pcaData)

then we execute the PCA:
principalComponents<-princomp(pcaData, cor=TRUE, score=FALSE)
print(principalComponents)
plot(principalComponents)
Finally after the execution of the PCA method and with the following commands appear:
(a) All the eigenvalues of the eigenvectors resulting from running the PCA method:

```
Eigenvectors<-principalComponents$loadings[,1:16]
print(Eigenvectors)
plot(Eigenvectors)
```

(b) All eigenvectors resulting from running the PCA method:

```
Eigenvalues<-principalComponents$sdev^2
print(Eigenvalues)
plot(Eigenvalues)
```

(c) The percentage of variance explained by each principal component resulting from the PCA algorithm:

```
percent = proportions(Eigenvalues)*100
print(percent)

principalComponents$loadings
principalComponents$scores
```

2.2 Exercise in Python

Similarly for python for the "Wine Quality" dataset we have the following results:

```
import pandas as pd
import numpy as np
from sklearn import datasets
from sklearn import decomposition
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
import csv
from math import fabs

##### Read csv file in python #####
data= pd.read_csv('C:\\path....csv',sep=";", header=0)

print('Lets take a look at the data')
print(data)

print('Lets take a look at the types of the data')
print(data.dtypes)


print('We have to delete variable quality')
datapca = data.loc[:, :"alcohol"]
print('Data ready for PCA')
print(datapca)

variables = []
##### PCA #####
pca = decomposition.PCA(n_components=11)
pca.fit(datapca)
tdata= pca.transform(datapca)
print('transformedwines')
print(tdata)
cov_mat = np.cov(tdata)

##### Print Eigenvalues #####
print("Eigenvalues:")
for i in range(len(pca.explained_variance_)):
print("———+++++———")
print(f"\lamda{i+1}: {pca.explained_variance_[i]}")
```

```
##### Print Eigenvectors #####
print("Eigenvectors:")
eigen = []
z = 0
for eigenvector in pca.components_:
print("−−−−−−−−−−−−−−−−−−++++++++++−−−−−−−−−−−−−−−−−−−−−−−−")
z += 1
print(f"Eigenvector {z} : {eigenvector}")
eigen.append(eigenvector)

variables = []
for col in datapca.columns:
variables.append(col)
print(variables)
for i in range(len(eigen)):
print(f'New variable {i + 1}= ')
for j in range(len(variables)):
if eigen[i][j] >= 0:
print(f'+({eigen[i][j]} * {variables[j]})',)
else:
print(f'−({fabs(eigen[i][j])} * {variables[j]})',)
print('\n')



##### Print the variance ratio #####
for i in range(len(pca.explained_variance_ratio_)) :
print("−−−−−−−−−−−−−−−−−−++++++++++++++++−−−−−−−−−−−−−−−−−−−−")
print(f"The variance ratio for (\lamda){i+1} is {pca.explained_
```

3 Subject 4

3.1 Exercise 4 in R

Firstly on this task we have to create a function called 'euclideanDistance' for calculating the distance between two vectors. In mathematics, the Euclidean distance is the length of a line segment between the two points and the formula to calculate it, is:

$$d(x,y) = \sqrt{\sum_{k=1}^{n} (x_k - y_k)^2}$$

On this subject we calculate the 'euclideanDistance' distance with this function.

```
euclideanDistance <− function(a,b){
        if (length(a) != length(b))
        return( "error")
        else
```

```
                dist <- sqrt(sum(((a-b)^2)))
                return(dist)
        }
```

In the first subquestion we calculated the Euclidean distance for four pair of vectors.
a) x=(1,2,3,4,5,6), y= (1,2,3,4,5,6)
b) x=(-0.5, 1, 7.3, 7, 9.4, -8.2, 9, -6, -6.3), y=(0.5, -1, -7.3, -7, -9.4, 8.2, -9, 6, 6.3)
c) x=(-0.5, 1, 7.3, 7, 9.4, -8.2), y=(1.25, 9.02, -7.3, -7, 5, 1.3)
d) x=(0, 0, 0.2), y=(0.2, 0.2, 0)
For task A we used the next code and we calculated that the distance between the vectors is 0

```
x= c(1,2,3,4,5,6)
y= c(1,2,3,4,5,6)
distance.a = euclideanDistance(x , y)
print(paste("The Distance between first x and y is : ", distanc
```

For task B used the next code and we calculated that the distance between the vectors is 40.7838

```
x= c(-0.5, 1, 7.3, 7, 9.4, -8.2, 9, -6, -6.3)
y= c(0.5, -1, -7.3, -7, -9.4, 8.2, -9, 6, 6.3)
distance.b = euclideanDistance(x , y)
print(paste("The Distance between first x
and y is : ",distance.b ))
```

For task C used the next code and we calculated that the distance between the vectors is 24.21059

```
x= c(-0.5, 1, 7.3, 7, 9.4, -8.2)
y= c(1.25, 9.02, -7.3, -7, 5, 1.3)
distance.c = euclideanDistance(x , y)
print(paste("The Distance between first x
and y is : ", distance.c))
```

For task D used the next code and we calculated that the distance between the vectors is 0.3464

```
x= c(0, 0, 0.2)
y= c(0.2, 0.2, 0)
distance.d = euclideanDistance(x , y)
print(paste("The Distance between first x
and y is : ", distance.d))
```

AA User	Minutes	SMS	MB
1	25000	14	7
2	42000	17	9
3	55000	22	5
4	27000	13	11
5	58000	21	13

For the second subquestion we have a table with informations for five users of telecommunication company.

For the fisrt task we had to find, by using the 'euclideanDistance' function, the user that is most facsimile with user 5.
We used the next code to calculate the distance for all users with user number 5. The distance for user 1 is 33000, for user 2 is 16000,for user 3 is 3000 and for user 4 is 31000. The distance for user 5 is 0, but we calculated for the needs of the exercise.

```
distance.15 = euclideanDistance(user1, user5)
distance.25 = euclideanDistance(user2, user5)
distance.35 = euclideanDistance(user3, user5)
distance.45 = euclideanDistance(user4, user5)
distance.55 = euclideanDistance(user5, user5)
```

User 3 has the smaller distance so he is the most similar with user 5.

For the next task we made a data frame with four vectors distances, minutes, sms, mb.

```
d = c(distance.15, distance.25, distance.35, distance.45)
df = data.frame (distances = c(distance.15, distance.25, distan
minutes = c(25000, 42000, 55000, 27000, 58000),
sms = c(12, 17, 22, 13, 21),
mb = c(7, 9, 5, 11, 13))
```

The next step is to run a linear regression models to describe the relationship between the variable 'Distance' and the other variables. The Coefficient of determination (R^2) for the model is 1, so all the variability of "Distance" is explained by the model. The variable with the highest variance , in the model, is the one that effects the most the variable 'Distance', which in our chase is 'Minutes'. On the next code we made all the calculations that are needed to be done to come in this conclusion.

```
model = lm(distances ~ minutes + sms + mb, data = df)
summary(model)$r.squared
var(df$minutes)
var(df$sms)
var(df$mb)
```

3.2 Exercise 4 in Python

We completed the same tasks in Python with the same results. Fisrt we imported the libraries we needed.

```
from sklearn.linear_model import  LinearRegression
import numpy as np
import pandas
import statistics
```

Next we difinied the Euclidean distance function.

```
def euclideanDistance(vec1, vec2):
dist = np.linalg.norm(vec1 - vec2)
return dist
```

For task A we used the next code and we calculated that the distance between the vectors is 0

```
x = np.array([1,2,3,4,5,6])
y = np.array([1,2,3,4,5,6])
dist = euclideanDistance(x, y)
```

For task B used the next code and we calculated that the distance between the vectors is 40.7838

```
x = np.array([-0.5, 1, 7.3, 7, 9.4, -8.2, 9, -6, -6.3])
y = np.array([0.5, -1, -7.3, -7, -9.4, 8.2, -9, 6, 6.3])
dist = euclideanDistance(x, y)
```

For task C used the next code and we calculated that the distance between the vectors is 24.21059

```
x = np.array([-0.5, 1, 7.3, 7, 9.4, -8.2])
y = np.array([1.25, 9.02, -7.3, -7, 5, 1.3])
dist = euclideanDistance(x, y)
```

For task D used the next code and we calculated that the distance between the vectors is 0.3464

```
x = np.array([0, 0, 0.2])
y = np.array([0.2, 0.2, 0])
dist = euclideanDistance(x, y)
```

In the second task we added the vectors for each user and puted them in a list.

```
ser1 = np.array([25000, 14, 7])
user2 = np.array([42000, 17, 9])
user3 = np.array([55000, 22, 5])
user4 = np.array([27000, 13, 11])
user5 = np.array([58000, 21, 13])
userlist = [user1, user2, user3, user4,user5]
```

Next with the function "euclideanDistance" we calculated the distances and made a list with them.

```
distances = []
for i in range(len(userlist)-1 ) :
```

```
distance = euclideanDistance(userlist[i], user5)
print(f'The distance between user{i+1}
and user 5 is {distance}')
distances.append(distance)
distances = list(distances)
```

Then we compare the distances to found the smaller. In the end we and the distance for number 5, wich is zero, on the distances list.

```
distances = list(distances)
minimum = min(distances)
finder = 0
for i in range(len(distances)):
if distances[i] == minimum :
finder = i
break
print(f'User {finder + 1} looks like the most with user 5')
distances.append(0)
```

We made a data frame for users.

```
user1 = np.array([25000, 14, 7, distances[0]])
user2 = np.array([42000, 17, 9, distances[1]])
user3 = np.array([55000, 22, 5, distances[2]])
user4 = np.array([27000, 13, 11, distances[3]])
user5 = np.array([58000, 21, 13, distances[4]])
data = pandas.DataFrame([user1,user2,user3,user4,user5],
["User 1", "User 2", "User 3", "User 4", "User 5"],
['min','sms', 'mb', 'distance'])
```

Then we performed linear regression model for each variable, found the coefficient of determination and calculated the variances of the variables.

```
X = data[['min', "sms", "mb"]]
y = data[['distance']]
regr = LinearRegression().fit(X, y)
r_sq = regr.score(X, y)

print(f"R squared for minutes is {r_sq}.
          So all the proportion of total variation of distance
          is explained by the model")
varmin =statistics.variance(data['min'])
print(f'The Variance of variable min is {varmin}')
varsms =statistics.variance(data['sms'])
print(f'The Variance of variable sms is {varsms}')
varmb =statistics.variance(data['mb'])
print(f'The Variance of variable mb is {varmb}')
print(f'The variable with the highest variance
          has the most effect on the variation of distance,
          which in our case is Minutes')
```

4 Subject 5

In this topic we created a function named cosineSimilarity which measures the similarity between two vectors of an inner product space. The cosine similarity always belongs to the interval [-1, 1]. For example, two proportional vectors have a cosine similarity of 1, two orthogonal vectors have a similarity of 0, and two opposite vectors have a similarity of -1. The mathematical formula for cosineSimilarity is:

$$cosinesimilarity = S_C(A,B) := \cos(\theta) = \frac{\mathbf{A}\cdot\mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum\limits_{i=1}^{n} A_i B_i}{\sqrt{\sum\limits_{i=1}^{n} A_i^2}\sqrt{\sum\limits_{i=1}^{n} B_i^2}},$$

Similarly we define the following cosineSimilarity functions for both R and Python.

4.1 Exercise in R

In R:

```
cosineSimilarity<- function(A,B) { (sum(A*B))/sqrt((sum(A^2))*(sum(B^2))) }
```

Working in the environment of r we arrive at the following conclusions:
For question **(a)** with the given vectors:
x=(9.32, -8.3, 0.2)
y= (-5.3, 8.2, 7)
the cosine similarity of vectors is: -0.7739558

For question **(b)** with the given vectors:
x=(6.5, 1.3, 0.3, 16, 2.4, -5.2, 2, -6, -6.3)
y=(0.5, -1, -7.3, -7, -9.4, 8.2, -9, 6, 6.3)
the cosine similarity of vectors is: -0.6546319

For question **(c)** with the given vectors:
x=(-0.5, 1, 7.3, 7, 9.4, -8.2)
y=(1.25, 9.02, -7.3, -7, 15, 12.3)
the cosine similarity of vectors is: -0.140921

For question **(d)** with the given vectors:
x=(2, 8, 5.2)
y=(2, 8, 5.2)
the cosine similarity of vectors is: 1
Using the following formulas in python yielded the same results. Firstly we define the cosineSimilarity fuction as:

4.2 Exersice in Python

```
def cosineSimilarity(vec1, vec2):
cos = cosine_similarity(vec1.reshape(1,-1),vec2.reshape(1,-1))
cos = str(cos)
```

```
if len(cos) < 7 :
return cos[2]
elif cos[2] == "-" :
return cos[2:9]
else:
return cos[2:8]
```

for question **(a)** the formula is:
x = np.array([9.32, -8.3, 0.2])
y = np.array([-5.3, 8.2, 7])
cosdist = cosineSimilarity(x, y)
print(f'The cosine similarity for task A is cosdist')

for question **(b)** the formula is:
x = np.array([6.5, 1.3, 0.3, 16, 2.4, -5.2, 2, -6, -6.3])
y = np.array([0.5, -1, -7.3, -7, -9.4, 8.2, -9, 6, 6.3])
cosdist = cosineSimilarity(x, y)
print(f'The cosine similarity for task B is cosdist')

for question **(c)** the formula is:
x = np.array([-0.5, 1, 7.3, 7, 9.4, -8.2])
y = np.array([1.25, 9.02, -7.3, -7, 15, 12.3])
cosdist = cosineSimilarity(x, y)
print(f'The cosine similarity for task C is cosdist')

for question **(d)** the formula is:
x = np.array([2, 8, 5.2])
y = np.array([2, 8, 5.2])
cosdist = cosineSimilarity(x, y)
print(f'The cosine similarity for task D is cosdist')

5 Subject 6

In this subject first of all we have to create a function which will estimate the distance between two vectors (x,y) which concludes only nominal variables. The definition of the function will be performed in both, the r and python environments for each of the three pairs of vectors given to us. The code we used for r is the following:

5.1 Exercise in R

```
nominalDistance <- function(x,y){
        k=0
        d=0

        if (length(x) != length(y))
        return( "different  number of variables inside the vect

        else
for (i in 1:length(x) ) {

          if(x[i]==y[i])
          k=0

          else{
                    k=1
          }

          d=d+k

}
return(d)}
```

if the distance between two vectors with nominal values is equal to 0, it means that their values are exactly the same.
So, for question **(a)** where it is given the pair of vectors:

x=("Green", "Potato", "Ford")
y=("Tyrian purple", "Pasta", "Opel")

the calculated distance between them is 3 and the given vectors contain 3 values. So we infer that no value between the vectors is the same.
Similarly for question **(b)** where is given the pair of vectors:

x=("Eagle", "Ronaldo", "Real madrid", "Prussian blue", "Michael Bay")
y=("Eagle", "Ronaldo", "Real madrid", "Prussian blue", "Michael Bay")

where the calculated distance between them is 0, we infer that values between the vectors are exactly the same.
Finally for the last question **(c)** where is given the pair of vectors:

x=("Werner Herzog", "Aquirre, the wrath of God", "Audi", "Spanish red")
y=("Martin Scorsese", "Taxi driver", "Toyota", "Spanish red")

where the calculated distance between them is 3 and the given vectors contain 4 values, we infer that there is only 1 value similar at both of vectors.

5.2 Exercise in Python

Similarly in python we create the function:

```
# we import the numpy library
import numpy as np

# we define the function
def nominalDistance(x,y):
i=0
l=0
d=0
for s in x:
```

```
print(i)

if s==y[i]:
l +=1
i=i+1
else:
l +=0
i=i+1
d = (len(x) - l)
return(d)
```

Question a

x= ("Green", "Potato", "Ford") y = ("Tyrian purple", "Pasta", "Opel")

d=nominalDistance(x,y) print("The nominal Distance between x and y is equal to : ", d)

Question b

x = np.array(["Eagle", "Ronaldo", "Real madrid", "Prussian blue", "Michael Bay"]) y = np.array(["Eagle", "Ronaldo", "Real madrid", "Prussian blue", "Michael Bay"])

d=nominalDistance(x,y) print("The nominal Distance between x and y is equal to : ", d)

Question c

x = np.array(["Werner Herzog", "Aquirre, the wrath of God", "Audi", "Spanish red"]) y = np.array(["Martin Scorsese", "Taxi driver", "Toyota", "Spanish red"])

d=nominalDistance(x,y) print("The nominal Distance between x and y is equal to : ", d)

Where the same results as those of R are produced.