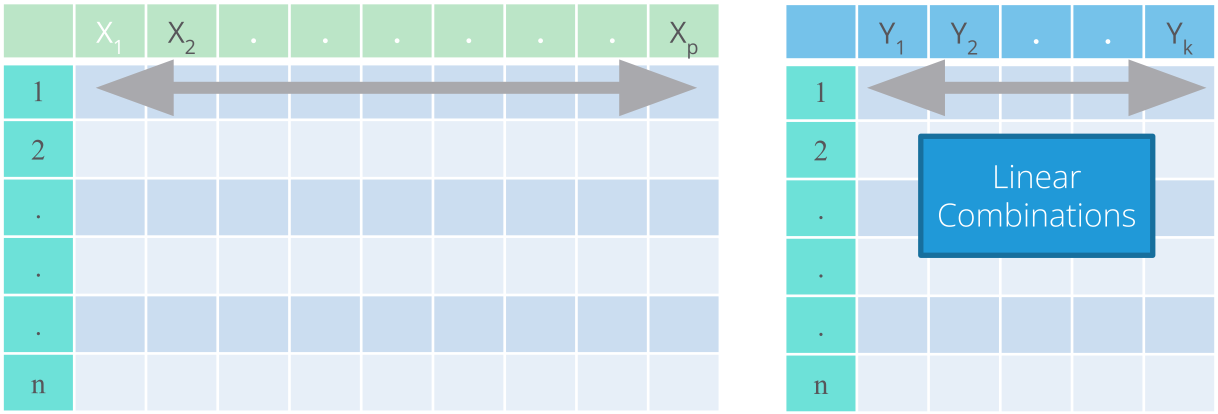
Principal Component Analysis

Learn How to Manage Data Dimensionality Without Losing Information

Contents

Data Reduction

- Summarization of data with p variables by a smaller set of (k) derived variables.
- These k derived variables are **linear combinations of original p variables.**



- In short, **n * p** matrix is **reduced to n * k** matrix.

Case Study – Athletics Records

Background

- Data on national athletics records for various countries is available.

Objective

- To achieve data reduction and obtain score for each country which can be used to rank countries based on athletics records.

Available Information

- Data Source: Applied Multivariate Statistical Analysis by Richard A. Johnson , Dean W. Wichern
- Sample size is 55 countries athletics.
- Records for 8 different athletics events – 100 meters to Marathon

Data Snapshot

Athleticsdata

Variables

Country	100m_s	200m_s	400m_s	800m_min	1500m_min	5000m_min	10000m_min	Marathon_min
Argentina	10.39	20.81	46.84	1.81	3.7	14.04	29.36	137.72
Australia	10.31	20.06	44.84	1.74	3.57	13.28	27.66	128.3

Observations

Column	Description	Type	Measurement	Possible Values
Country	Country Name	Categorical	-	-
100m_s	Time for 100 meter running	Continuous	Seconds	Positive Values
200m_s	Time for 200 meter running	Continuous	Seconds	Positive Values
400m_s	Time for 400 meter running	Continuous	Seconds	Positive Values
800m_min	Time for 800 meter running	Continuous	Minutes	Positive Values
1500m_min	Time for 1500 meter running	Continuous	Minutes	Positive Values
5000m_min	Time for 5000 meter running	Continuous	Minutes	Positive Values
10000m_min	Time for 10000 meter running	Continuous	Minutes	Positive Values
Marathon_min	Time for Marathon running	Continuous	Minutes	Positive Values

PCA in Python

```
#Importing data
import pandas as pd
import numpy as np

athletics=pd.read_csv("Athleticsdata.csv")
```

- **read_csv()** is used to import csv file. Our data is stored as an object named **athletics.**

```
# standardize all variables
athletics2=athletics.drop(['Country'], axis=1)


from sklearn.decomposition import PCA
from sklearn.preprocessing import scale

standardisedX = scale(athletics2)
X = pd.DataFrame(standardisedX, index=athletics2.index,
columns=athletics2.columns)
```

- **drop()** is used to remove the column named "Country" from the data.

- The data is standardised using the function **scale()** and stored as a dataframe object X

PCA in Python

```
#Running PCA and creating summary table
pca = PCA().fit(X)
names = ["PC"+str(i) for i in range(1,9)]

SD = list(np.std(pca.transform(X), axis=0))
VarProp = list(pca.explained_variance_ratio_)
CumProp = [np.sum(VarProp[:i]) for i in range(1,9)]

summary = pd.DataFrame(list(zip(SD, VarProp, CumProp)), index=names,
columns=['Standard Deviation','Proportion of Variance','Cumulative
Proportion'])
summary
```

- ❑ Create a vector of names as PC1,PC2 …. PC8.
- ❑ Extract Standard Deviation and Proportion of variance explained. Define Cumulative proportion for the summary table
- ❑ pca.transform(X) computes scores
- ❑ Create a dataframe of summary output

PCA in Python

```
# Output:
```

	Standard Deviation	Proportion of Variance	Cumulative Proportion
PC1	2.574068	0.828228	0.828228
PC2	0.935501	0.109395	0.937624
PC3	0.398207	0.019821	0.957445
PC4	0.352195	0.015505	0.972950
PC5	0.282863	0.010001	0.982951
PC6	0.260302	0.008470	0.991421
PC7	0.214848	0.005770	0.997191
PC8	0.149910	0.002809	1.000000

Interpretation:

- summary gives **std. deviation (sd), proportion of variance and cumulative proportion**. Variance is nothing but the Eigenvalue of correlation matrix.
- First Principal Component explains 83% of the variation. Note that 8 PC's are derived using 8 variables but first PC explains most of the variation.

PCA in Python - Loadings and Scores

#Component Loadings

```
rows = X.columns
col = ["Comp"+str(i) for i in range(1, len(X.columns)+1)]
L = pd.DataFrame(list(zip(pca.components_[0],pca.components_[1],
pca.components_[2],pca.components_[3],pca.components_[4],
pca.components_[5],pca.components_[6],pca.components_[7])),index=rows,
columns = col)
L
```

Output:

```
                Comp1     Comp2     Comp3  ...     Comp6     Comp7     Comp8
100m_s       0.318293 -0.564684  0.326323  ...  0.590449 -0.154303  0.113210
200m_s       0.336855 -0.462270  0.369020  ... -0.647587  0.128066 -0.101621
400m_s       0.355561 -0.249318 -0.561085  ... -0.158447  0.009292 -0.002585
800m_min     0.368626 -0.013405 -0.530948  ...  0.011856  0.237073 -0.040305
1500m_min    0.372682  0.140200 -0.154640  ...  0.143104 -0.608456  0.143305
5000m_min    0.364283  0.312458  0.189618  ...  0.155079  0.592691  0.543015
10000m_min   0.366702  0.307018  0.181817  ...  0.231701  0.165205 -0.796334
Marathon_min 0.341825  0.439947  0.260172  ... -0.329455 -0.393327  0.160236
```

Interpretation:

- First Principal Component can be interpreted as ‘general athletics skill’ since all variables have similar loadings.

PCA in Python - Loadings and Scores

```
#Scores Based on PCA
```

```
Score= PCA().fit_transform(X)
Score_df = pd.DataFrame(Score, index = athletics.index, columns = col)
athletics = athletics.assign(performance=Score_df.Comp1)
athletics.head()
```

```
# Output:
```

	Country	100m_s	200m_s	...	10000m_min	Marathon_min	performance
0	Argentina	10.39	20.81	...	29.36	137.72	0.265654
1	Australia	10.31	20.06	...	27.66	128.30	-2.466968
2	Austria	10.44	20.81	...	27.72	135.90	-0.813415
3	Belgium	10.34	20.68	...	27.45	129.95	-2.058239
4	Bermuda	10.28	20.58	...	30.55	146.62	0.747146

Interpretation:

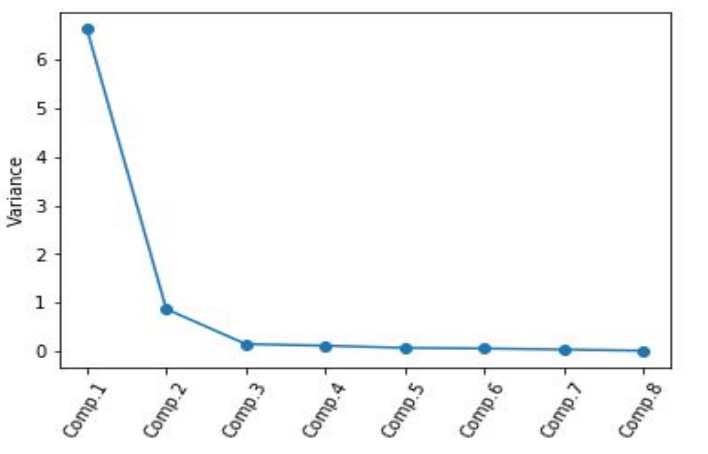
- New column ‘performance’ enumerates scores returned by the PCA summary.
- Lower score implies lesser time and hence better athletics performance.

PCA in Python – Scree Plot

```
#Scree Plot : plot the principal components vs. variances.
```

```
y = np.std(pca.transform(X), axis=0)**2
x = np.arange(len(y)) + 1
plt.plot(x, y, "o-")
plt.xticks(x, ["Comp."+str(i) for i in x], rotation=60)
plt.ylabel("Variance")
plt.show()
```

```
# Output:
```



Interpretation:

- First Principal Component is sufficient in explaining the maximum variation.

Exploratory Data Analysis Based on PCA

```
#Using Scores to Check Data Features
```

```
athletics.sort_values(by = 'performance').head(3)
athletics.sort_values(by = 'performance').tail(3)
```

- **sort_values()** sorts the data frame based on performance. Python takes **ascending = True** as default.
- **head()** and **tail()** extracts top n and bottom n countries in terms of performance measured by performance variable. Here n=3.

```
# Output :
```

```
                                  Country  100m_s  ...  Marathon_min  performance
52                                    USA    9.93  ...        128.22    -3.460450
20  Great Britain and Northern Ireland   10.11  ...        129.13    -3.050287
28                                  Italy   10.01  ...        131.08    -2.750446
```

```
           Country  100m_s  200m_s  ...  10000m_min  Marathon_min  performance
35       Mauritius   11.19   22.45  ...       31.77        152.23     4.299192
54  Western Samoa    10.82   21.86  ...       34.71        161.83     7.297965
11     Cook Isands   12.18   23.20  ...       35.38        164.70    10.653867
```

Interpretation:

- USA, Britain and Italy are the top three performing countries.
- Cook Islands, Western Samoa and Mauritius are the bottom three countries.

PCA in Python – Verification of PCA

```
#Verification that PC's are Uncorrelated
print(Score_df.corr().round())
```

❑ **corr()** and **round()** to calculate the rounded correlations for the principal components.

	Comp1	Comp2	Comp3	Comp4	Comp5	Comp6	Comp7	Comp8
Comp1	1.0	0.0	-0.0	-0.0	0.0	0.0	-0.0	-0.0
Comp2	0.0	1.0	0.0	-0.0	-0.0	0.0	0.0	-0.0
Comp3	-0.0	0.0	1.0	-0.0	-0.0	0.0	0.0	0.0
Comp4	-0.0	-0.0	-0.0	1.0	-0.0	0.0	-0.0	0.0
Comp5	0.0	-0.0	-0.0	-0.0	1.0	-0.0	-0.0	0.0
Comp6	0.0	0.0	0.0	0.0	-0.0	1.0	-0.0	0.0
Comp7	-0.0	0.0	0.0	-0.0	-0.0	-0.0	1.0	-0.0
Comp8	-0.0	-0.0	0.0	0.0	0.0	0.0	-0.0	1.0

Interpretation:

- The principal components are uncorrelated

Quick Recap

In this session we learnt about, Principal Component Regression :

Data Reduction and PCA	• Principal Component Analysis is a key data reduction method • Data Reduction is necessary while analyzing high dimensional data
PCA in Python	• **`PCA()`** function in Scikit Learn performs PCA • Scree Plot is used to decide number of components to be retained