Daniel Mocko

IT54-2015

Principal Component Analysis

Sadržaj

[1 Uvod 1](#_Toc105059)

[2 Pojam Principal Component Analysis 2](#_Toc105060)

[3 Osnovni statistički pojmovi 3](#_Toc105061)

[4 Matematički princip računanja PCA 4](#_Toc105062)

[4.1 Računanje srednje vrednosti 4](#_Toc105063)

[4.2 Računanje standradne devijacije 5](#_Toc105064)

[4.3 Računanje kovarijnse 5](#_Toc105065)

[4.4 Računanje koeficijenta korelacije 5](#_Toc105066)

[4.5 Računanje Ajgenvrednosti 6](#_Toc105067)

[4.6 Računanje Ajgenvektora 7](#_Toc105068)

[4.7 Sortiranje 8](#_Toc105069)

[4.8 Računanje PCA vrednosti 8](#_Toc105070)

[5 PCA u Python-u 9](#_Toc105071)

[5.1 PCA analize korak po korak 10](#_Toc105072)

[5.1.1 Učitavanje i prikaz podataka 10](#_Toc105073)

[5.1.2 Selektovanje i histogrami 10](#_Toc105074)

[5.1.3 Standardizacija 12](#_Toc105075)

[5.1.4 Eigenvectors i Eigenvalues 12](#_Toc105076)

[5.1.5 Sortiranje PCA 13](#_Toc105077)

[5.1.6 PCA vizuelizacija 14](#_Toc105078)

[5.2 Primer PCA analize korišćenjem sklearn biblioteke 17](#_Toc105079)

[6 Zaključak 17](#_Toc105080)

# Uvod

U današnje vreme softveri sve više koriste oblast iz IT struke koja se naziva **machine learning**. *Machine learning* je podoblast veštačke inteligencije koja ima za cilj da konstruiše algoritam koji će imati sposobnost da se prilagodi na nove i nepoznate situacije i da uči na osnovu prethodnog iskustva. Veliku primenu *machine learning* je našao i u *data sience*.

Jedan od problem koji se javlja u ove dve oblasti je vizualizacija podataka. Problem se javlja kada postoji n-dimenzionalni prostor nad kojim treba da se izvrši vizualizacija gde je jako teško razumeti šta se dešava sa tim podacima kada se nalaze u takvom prostoru. Ovaj problem se rešava primenom principa za smanjenje dimenzija. U temi je posvećena pažnja *Principal Component Analysis (PCA)* metodi.

U okviru ove teme se govori šta je PCA i kako se koristi. Takođe je dato obrazloženje u primerima. U okviru matematičkog modela prikazuje se PCA princip na osnovu nestandardizovanih podataka, dok u primeru gde je rađena analiza u Python jeziku se prikazuje PCA princip na osnovu standardizovanih podataka.

# Pojam Principal Component Analysis

Ovu tehniku je prvi put opisao Karl Pearson 1901. godine. Iako je vršio izračunavanja sa samo dve ili tri varijable Pearson je verovao da se analiza glavnih komponenti može upotrebiti i za rešavanje problema sa puno više promenljivih. Opis izračunavanja je dat mnogo kasnije od strane Htelling-a, 1933. godine. Međutim, i dalje su izračunavanja bila previše komplikovana i zamorna kada bi trebalo napraviti analizu sa većim brojem varijabli. Široka upotreba analize glavnih komponenti je usledila zapravo tek sa pojavom računara.

*Principal Component Analysis* (Analiza glavnih komponenti) koristi se kao alat u vizualizaciji podataka i pri pravljenju modela. Ovom statističkom procedurom se korišćenjem, početni skup varijabli transformiše u novi skup nekorelisanih varijabli putem linearnih transformacija. Pritom se indentifikuju one varijable koje su najbitnije za podatke, koji opisuju najviše varijabilnosti i oni koji nisu toliko bitni. Ti nebitni prediktori se mogu izbaciti i sprovodi se linearna regresija sa transformisanim prediktorima. Na ovaj način se PCA koristi za smanjivanje dimenzionalnosti.

PCA se koristi kada:

* Hoćemo da smanjimo dimenzionalnost
* Hoćemo da osiguramo da su prediktori nezavisni

Ideja je da se prediktori transformišu tako da se prvom glavnom komponentom opiše najveći deo disperzije podataka. Da to bude pravac, jedinični vektor, u *n*-dimenzionom prostoru po kom se najviše prostiru podaci. Svaki sledeći pravac se traži tako da bude normalan na prethodne i da ima drugu najveću moguću disperziju. Ovim se otkriva struktura u podacima kojom se najbolje opisuje varijabilnost. Kaže se i da su glavne komponente nekorelisane (ortogonalne- kovarijacija je upravo skalarni proizvod u prostoru slučajnih veličina).

Koraci PCA algoritma su:

1. Vrši se standardizacija originalnih podataka
2. Izračunava se matrica kovarijansi
3. Izračunavaju se ajgenvrednosti (*eigrnvalues*) i ajgenvektori (*eigenvectors*)
4. Komponente koje se u modelu odnose na malu proporciju varijacija podataka se eliminišu.

Ovde će biti prikazan PCA metod na dva načina:

* Putem matematičkog modela - gde će biti prikazano, korak po korak, kako se računa PCA vrednost za svaku varijablu i kako se donosi odluka koja varijabla se izbacuje, a koja dalje koristi u analizi podataka
* Putem koda - ovde će biti prikazan PCA princip u programskom jeziku pyhton zajedno sa vizuelizacijom podataka

# Osnovni statistički pojmovi

Aritmetička sredina predstavlja srenju vrednost nekog niza elementa

Standardna devijacija je kvadratna udaljenost svakog uzorka od njegove srednje vrednosti

Standardizacija podataka omogućava proporciju ili procenat slučajeva ispod ili iznad određenog rezultata.

Varijansa ili disperzija je mera rasipanja individualnih vrednosti oko srednje vrednosti, slična je standardnoj devijaciji, i izražena je formulom

Kovarijansa je mera koja se koristi za dobijanje vrednosti između dve ili više različitih dimenzija, odnosno ona meri jačinu veze između dve promenljive. Ako imamo 3D prostor na primer možemo da računamo kovarijansu između različitih dimenzija npr. i ili i .

Kovarijansa matrice

Koeficijent korelacije se izračunava tako što kocarijansu za x i y podelimo sa proizvodom standardnih devijacija promenljive x i y. Element koji je u korelaciji sa samim sobom, rezultat je uvek 1

# Matematički princip računanja PCA

U ovom primeru je dat matematički prikaz šta se radi sa podacima do koraka vitualizacije. Korak vizualizacije neće biti prikazan u ovom primeru već će to biti učinjeno u poglavlju 5 *PCA u Python-u*.

U ovom primeru je data tabela sa obeležjima x, y i z kao i 5 redova (torki).

|  |  |  |  |
| --- | --- | --- | --- |
| RB.R | x | y | z |
| 1 | 156 | 245 | 316 |
| 2 | 154 | 240 | 310 |
| 3 | 153 | 240 | 309 |
| 4 | 153 | 236 | 315 |
| 5 | 155 | 243 | 320 |

Tabela 1 - Prikaz podataka

Koraci kako će se sprovesti ovaj primer su sledeći:

1. Računanje srednje vrednosti
2. Računanje standradne devijacije
3. Računanje kovarijanse
4. Računanje koeficijenta korelaije
5. Računanje Ajgenvrednosti (*eigenvalue*)
6. Računanje Ajgenvektora (*eigenvector*)
7. Sortiranje
8. Računanje PCA vrednosti

Napomena: Iako je u poglavlju *2.* *Pojma Principal Component Analysis* rečeno da je prvi korak standardizacija podataka, ovde to neće biti učinjeno. Kada se standardizuju podaci i kada se izračuna kovarijnsa nema potrebe da se računa koeficijent korelacije, jer matrica kovarijansi za standardizovane vrednosti je korelaciona matrica. U koliko se ne uradi standardizacija podataka onda je potrebno za te podatke izračunati matricu kovarijanse gde se na osnovu nje dobija koeficijent korelacije. U ovom primeru će se raditi na nestandardizovanim podacima.

## Računanje srednje vrednosti

U ovom delu se vrši računanje srednje vrednosti za svaku dimenziju odnosno u ovom slučaju to su varijable x, y i z.

na isti način se računaju i srednje vrednosti za atribute y i z.

## Računanje standradne devijacije

Na isti način se vrši računanje i standardne devijacije za y i z

## Računanje kovarijnse

Ovde će biti prikazano računanje kovarijanse samo za primer između varijable xy. Postupak je isti i za računanje varijable xz, kao i yz.

## Računanje koeficijenta korelacije

Na osnovu prethodne kovarijanse ovde će biti prikazan račun za koeficijent korelacije između x i y. Koeficijent korelacije između xx,yy i zz je po *default* 1.

Na osnovu ovoga potrebno je izračunati i odnose između xz i yz. Pregled svih korelacija prikazan je u *Tabela 2*.

|  |  |  |  |
| --- | --- | --- | --- |
|  | *x* | *y* | *z* |
| *x* | 1.000 |  |  |
| *y* | 0.908 | 1.000 |  |
| *z* | 0.593 | 0.387 | 1.000 |

Tabela 2 - Izgled korelacione matrice

Naravno ove vrednosti korelacije mogu se predstaviti i putem matrice, u nastavku će se zvati matrica A.

A =

## Računanje Ajgenvrednosti

Ajgenvrednosti se dobijaju u nekoliko koraka:

1. Potrebno da se napravi matrica koja ima na dijagonali sve jedinice, tzv. jedinična matrica, nju obeležavamo simbolom I (veliko slovo i)
2. Matrica I se množi sa simbolom λ, λ\*Ι

λ\*Ι=

1. Nakon toga se od matrice korelacije A oduzima matrica λ\*Ι, A- λ\*Ι i to će se u nastavku zvati matrica B

B=A- λ\*Ι

1. Sledeći korak je da se izračuna determinanta na osnovu dobijene matrce B, gde rezultat predstavlja ajgenvrednosti matrice

na osnovu računanja determinante dobijaju se ajgenvrednosti, u ovom slučaju one iznose

## Računanje Ajgenvektora

U ovom delu je prikazan postupak za i dobijanje vektora .

Ajgenvektori se dobijaju tako što se od korelacione matrice A odzme vrednost (ajgenvrednost).

Zatim se dobijena matrica pomnoži sa vektorom .

Na osnovu ovoga postavlja se linearna jednačina.

Način na koji će se rešiti ovaj linearni sistem je taj što će se pretpostaviti da je , i na osnovu toga uzimaju se prve dve jednačine.

Rešavanjem ovog sistema dobija se rezultat , na osnovu toga potrebno je izračunati vektor . On se računa na sledeći način:

Ovakav postupak treba ponoviti i za slučaje i na osnovu toga dobiti vektore .

Sva rešenja za ajgenvektore i ajgenvrednosti su predstavljena u *Tabela 3*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | | Ajgenvrednosti | Koeficijenti ajgenvektora | | |
|  |  |  |
| Ajgenvektori | *x* | 2.285 | 0.64 | 0.59 | 0.47 |
| *y* | 0.655 | 0.18 | 0.49 | -0.85 |
| *z* | 0.061 | -0.75 | 0.63 | 0.21 |

Tabela 3 – Prikaz dobijenih ajgenvektora i ajgenvrednosti

## Sortiranje

Kada se dobiju svi vektori za svaku ajgenvrednost potrebno je izvršiti sortiranje podataka po ajgenvrednosti u opadajućem poretku. Na osnovu dobijenih podataka u tabeli podaci su slučajno sortirani.

## Računanje PCA vrednosti

Nakon sortiranja se gleda koliko komponeneti želimo da uzmemo da bi smo napravili vizualizaciju. Spram broja dimenzija biramo n najboljih vektora. Ako je to 2D onda biramo 2 vektora itd. Drugi način je da se izračuna ukupna varijansa po glavnoj komponenti.  Ako hoćemo da smanjimo dimenzionalnost tako da bude objašnjeno 80% početne disperzije (varijanse) uzima se onoliko prvih komponenti koliko je potrebno da procenat objašnjene disperzije bude minimum 80%. U ovom primeru bi izabrali prve dve varijable za prikaz ovih podataka u 2D prostoru jer je vrednost z=2.01%, što je zanemarljivo malo.

# PCA u Python-u

U ovom primeru će se pokazati kako se radi PCA analiza u programskom jeziku Python. Biće pokazano na dva načina. Prvi će biti dosta sličan načinu kao što je rađeno u matematičkom delu odnosno korak po korak samo za razliku od njega ovde će biti prikazana i vizuelizacija podataka kako bi se stvorila jasna slika o tome kako PCA zapravo radi. Drugi način je dosta kraći koji se radi putem python biblioteke **sklearn.**

Primer korišćenja PCA će se primeniti na Iris *dataset*-u, koji može da se nađe na linku <https://archive.ics.uci.edu/ml/datasets/Iris>

U ovom dataset-u nalaze se 3 vrste klasa, a to su:

1. Iris setosa
2. Iris versicolor
3. Iris virginica

Važno je napomenuti da dataset sadrži 150 redova i 4 atributa koji su:

1. Sepal length in cm (dužina listića čašice)
2. Sepal width in cm (širina listića čašice)
3. Petal length in cm (dužina latice)
4. Petal width in cm (širina latice)

## PCA analize korak po korak

### Učitavanje i prikaz podataka

Prvo je potrebno da se importuje pandas biblioteka kako bi mogli da učitamo Iris dataset sa satja. Nakon toga se vrši prikaz učitanih podataka.

import pandas as pd

*# Učitavanje Iris dataset-a sa sajta*

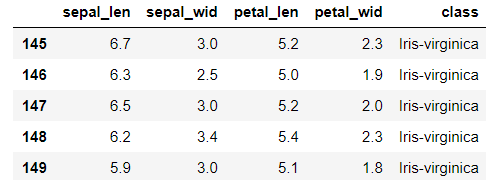
df = pd.read\_csv( filepath\_or\_buffer='https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data', header=None, sep=',')

# Imenovanje kolona za prikaz u tabeli

df.columns=['sepal\_len', 'sepal\_wid', 'petal\_len', 'petal\_wid', 'class']

*# Prikaz poslednjih 5 učitanih redova*

df.tail()



Slika 1 - Prikaz poslednjih 5 torki iz IRIS dataset-a

### Selektovanje i histogrami

*# Selektuju se atributi iz dataset-a koji će se koristiti za vizuelizaciju.*

*# Selektovanje prve 4 kolone iz dataset-a* *i to su NESTANDARDIZOVANI podaci*

X = df.iloc[:,0:4].values

*# Selektovanje poslednje 5te kolone iz dataset-a*

y = df.iloc[:,4].values

*# Kreiranje histograma za prikaz podataka dataset-a sa 4 dimenzije.*

*# Importovanje biblioteke za grafički prikaz podataka*

import plotly.plotly as py

data = []

legend = {0:False, 1:False, 2:False, 3:True}

*# Objekat koji sadrži za svaku klasu odgovarajuću boju*

colors = {'Iris-setosa': '#0D76BF', 'Iris-versicolor': '#00cc96', 'Iris-virginica': '#EF553B'}

*# Unutar ovih for petlji se kreiraju objekti putem rečnika dict() koji nose informaciju o # podacima koji će se prikazivati u histogramu i ti objekti se smeštaju u niz data.*

for col in range(4):

for key in colors:

trace = dict(

type='histogram',

x=list(X[y==key, col]),

opacity=0.75,

xaxis='x%s' %(col+1),

marker=dict(color=colors[key]),

name=key,

showlegend=legend[col]

)

data.append(trace)

*# Kreiranje objekta koji sadži opis histograma, odnosno njegove labele*

layout = dict(

barmode='overlay',

xaxis=dict(domain=[0, 0.25], title='sepal length (cm)'),

xaxis2=dict(domain=[0.3, 0.5], title='sepal width (cm)'),

xaxis3=dict(domain=[0.55, 0.75], title='petal length (cm)'),

xaxis4=dict(domain=[0.8, 1], title='petal width (cm)'),

yaxis=dict(title='count'),

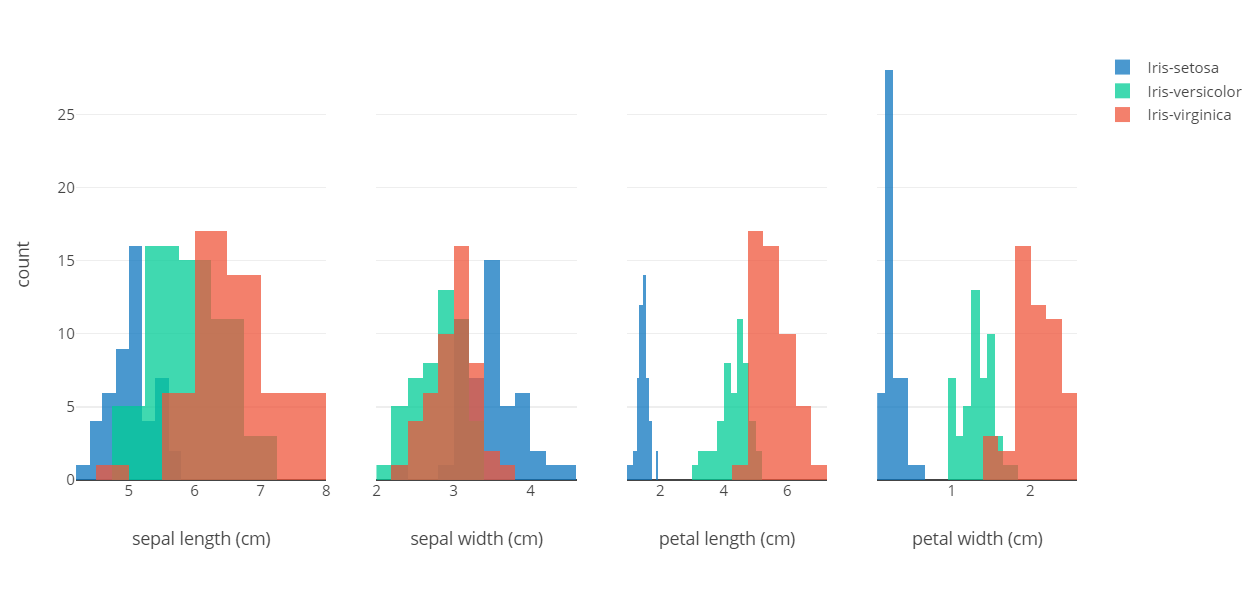
title='Distribution of the different Iris flower features'

)

*# Objekat koji nosi informacije o podacima koji će se prikazati i labelama histograma*

fig = dict(data=data, layout=layout)

*# Kreiranje histograma na osnovu podataka iz objekta fig i bira se vrsta histograma* py.iplot(fig, filename='exploratory-vis-histogram')



Slika 2 - Prikaz histograma za svaku dimenziju IRIS dataset-a

Ovde se vidi sa histograma da je problem da se svi podaci prikažu u 2D prostoru pošto dataset sadrži 4 atributa i 3 klase gde je gotovo nemoguće i besmisleno za sve klase odjednom napraviti neki 3D proctor u kome će biti prikazana sva 4 obeležja. Zato je ovde za svako obeležje prikazan histogram sa svoje 3 klase. Iz ovog razloga, u koliko želimo da prikažemo podatke u 3D ili 2D prostoru, je potrebno da se odaberu atributi koji će imati največi uticaj I time će stvoriti najbolju sliku o podacima sa što manje gubitaka. Zato se primenjuje PCA analiza, što znači da sve podatke nećemo morati prikazati u 4 grafika nego u jednom.

### Standardizacija

Prvo što moramo da uradimo kada se primenjuje PCA princip je da podatke standardizujemo kako bi se mogla napraviti matrica kovarijanse.

*#importovanje bilbioteke StandardScaler za standardizaciju podataka*  
from sklearn.preprocessing import StandardScaler

*#standardizacija podataka se vrsi putem fit\_transform(X)*

X\_std = StandardScaler().fit\_transform(X)

### Eigenvectors i Eigenvalues

U ovom primeru će se Eigenvectors i Eigenvalues dobijati putem matrice kovarinsi, ali je prikazano i kako se oni dobijaju putem korelacione matrice.

Matrica kovarijanse

Matricu kovarijanse možemo da dobijemo na dva načina.

*########### PRVI NAČIN ############*

*#učitavanje biblioteke numpy za kreiranje kovarijansne matrice*

import numpy as np

*#računanje srednjih vrednosti iz standardizovanih podataka*

mean\_vec = np.mean(X\_std, axis=0)

*#kreiranje kovarijansne matrice*

cov\_mat = (X\_std - mean\_vec).T.dot((X\_std - mean\_vec)) / (X\_std.shape[0]-1)

*#Ispis matrice*

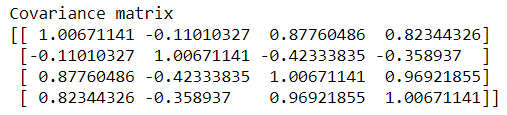
print('Covariance matrix \n%s' %cov\_mat)

*########## DRUGI NAČIN #############*

import numpy as np

print('Covariance matrix: **\n%s**' %**np**.cov(X\_std.T))

Oba ova načina daju isti rezultat koji je prikazan na *Slika 3*.



Slika 3 - Dobijen rezultat matrice kovarijnse

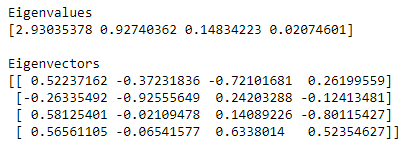
*#na osnovu kovarijnsne matrice racunaju se Eigenvalue i Eigenvector*

eig\_vals, eig\_vecs = np.linalg.eig(cov\_mat)

*#ispis Eigenvalues i Eigenvectors*

print('\nEigenvalues \n%s' %eig\_vals)

print('Eigenvectors \n%s' %eig\_vecs)



Slika 4 - Dobijeni Eigenvalues i Eigenvectors na osnovu matrice kovarijanse

Matrica korelacije

Pošto su podaci prethodno sandardizovani nema potrebe za ovim korakom, ali će se ipak demonstrirati kako bi se kreirala matrica korelacije na osnovu nestandardizovanih podataka.

*# Kreiranje korelacione matrice na osnovu NESTANDARDIZOVANIH podataka*

cor\_mat2 = np.corrcoef(X.T)

*# Dobijanje Eigenvalues i Eigenvector na osnovu korelacione matrice*

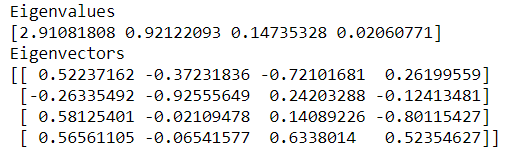
eig\_vals, eig\_vecs = np.linalg.eig(cor\_mat2)

*# Ispis Eigenvalues*

print('**\n**Eigenvalues **\n%s**' %**eig\_vals**)

*# Ispis Eigenvector*

print('Eigenvectors **\n%s**' %**eig\_vecs**)



Slika 5 - Dobijeni Eigenvalues i Eigenvectors na osnovu matrice korelacije

### Sortiranje PCA

*# Kreieanje liste koja sadrži parove (eigenvalue, eigenvector)*eig\_pairs = [(np.abs(eig\_vals[i]), eig\_vecs[:,i]) for i in range(len(eig\_vals))]

*# Sortiranje (eigenvalue, eigenvector) od viših ka nižim vrednostima, odnosno u # opadajućem poretku*

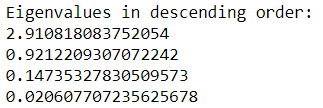
eig\_pairs.sort()

eig\_pairs.reverse()

*# Prikaz svih sortiranih Eigenvalues*

print('Eigenvalues in descending order:')

for i in eig\_pairs: print(i[0])

****

Slika 6 - Prikaz sortiranih Eigenvalues

### PCA vizuelizacija

*# Računanje koliko koji atribut nosi informacija (varijnsa) na osnovu kojih može da se # izvrši smanjenje dimenzionalnosti*

tot = sum(eig\_vals)

var\_exp = [(i / tot)\*100 **for** i **in** sorted(eig\_vals, reverse=True)]

cum\_var\_exp = np.cumsum(var\_exp)

*# Kreiranje objekta koji sadrži vrednosti informacija (varijansi) koju nose PCA atributi*

trace1 = dict(

type='bar',

x=['PC **%s**' %**i** for i in range(1,5)],

y=var\_exp,

name='Individual'

)

*# Kreiranje objekta koji sadrži kumulativnu vrednost svih informacija (varijansi) PCA*

*# atributa*

trace2 = dict(

type='scatter',

x=['PC **%s**' %**i** for i in range(1,5)],

y=cum\_var\_exp,

name='Cumulative'

)

*# Niz koji sadži vrednosti svakog PCA atributa i kumulativnu vrednost PCA atributa*

data = [trace1, trace2]

*# Kreiranje izgleda grafika za prikaz varijansi PCA atributa*

layout=dict(

title='Explained variance by different principal components',

yaxis=dict(

title='Explained variance in percent'

),

annotations=list([

dict(

x=1.16,

y=1.05,

xref='paper',

yref='paper',

text='Explained Variance',

showarrow=False,

)

])

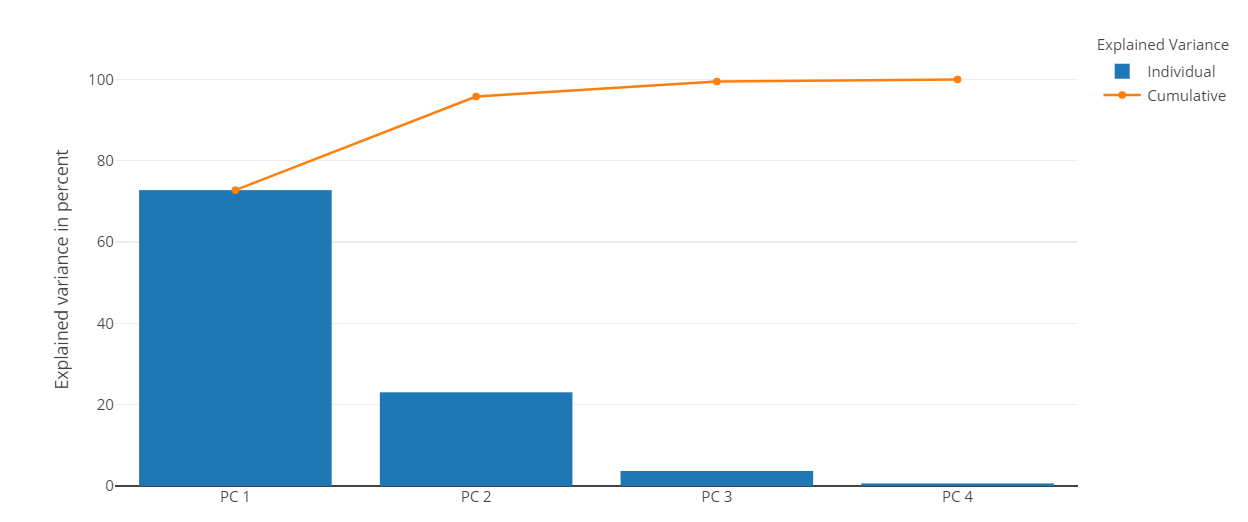
)

*# Kreiranje objekta koji će da prenese sve vrednosti iplot funkciji za iscrtavanje grafika*

fig = dict(data=data, layout=layout)

*# Prikaz PCA varijansi i njihovih ukupnih vrednosti*

py.iplot(fig, filename='selecting-principal-components')



Slika 7 - Dobijene vrednosti u procentima za svaki PCA atribut

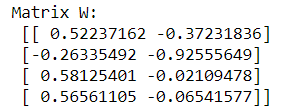
*# Kreiranje matrice koja sadrži vrednosti prve dve PCA komponente*

matrix\_w = np.hstack((eig\_pairs[0][1].reshape(4,1),

eig\_pairs[1][1].reshape(4,1)))

*# Prikaz matrice*

print('Matrix W:**\n**', matrix\_w)



*# Množnje standardiyovanih vrednosti sa matricom w kako bi se podaci projektovali u novi # podprostor, tj u 2D proctor.*

Y = X\_std.dot(matrix\_w)

data = []

*# Definisanje vrednosti za prikaz podataka*

for name, col in zip(('Iris-setosa', 'Iris-versicolor', 'Iris-virginica'), colors.values()):

trace = dict(

type='scatter',

x=Y[y==name,0],

y=Y[y==name,1],

mode='markers',

name=name,

marker=dict(

color=col,

size=12,

line=dict(

color='rgba(217, 217, 217, 0.14)',

width=0.5),

opacity=0.8)

)

data.append(trace)

*# Definisanje izglea grafika*

layout = dict(

showlegend=True,

scene=dict(

xaxis=dict(title='PC1'),

yaxis=dict(title='PC2')

)

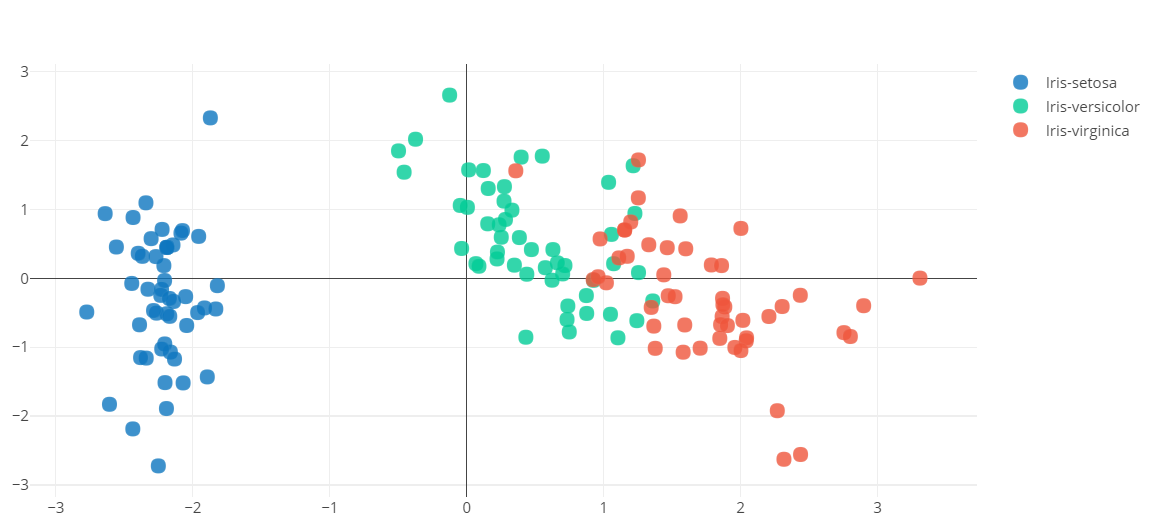
)

*# Objekat koji nosi informacije o podacima i izgledu grafika*

fig = dict(data=data, layout=layout)

*# Vizualizacija Iris dataset-a u 2D prostoru*

py.iplot(fig, filename='projection-matrix')



Slika 8 - Prikaz podataka IRIS dataset-a u 2D prostoru

## Primer PCA analize korišćenjem sklearn biblioteke

Ovo predstavlja istu analizu kao i u prethodnom primeru samo što se putem sklearn biblioteke mnogo brže primenjuje ova analiza.

*# Poziva se PCA funkcija importovanjem sklearn.decomposition biblioteke i vrši se # preimenovanje u sklearnPCA koja će se dalje koristiti. Podrazumeva se da su podaci # prethodno učitani i standardizovani*

**from** **sklearn.decomposition** **import** PCA **as** sklearnPCA

*# Definisanje dimenzija prostora. U ovom slučaju želimo da dobijemo 2D prostor*

sklearn\_pca = sklearnPCA(n\_components=2)

*# Vršenje PCA procesa nad standardizovanim podacima*

Y\_sklearn = sklearn\_pca.fit\_transform(X\_std)

data = []

*# Definisanje vrednosti za prikaz podataka*

**for** name, col **in** zip(('Iris-setosa', 'Iris-versicolor', 'Iris-virginica'), colors.values()):

trace = dict(

type='scatter',

x=Y\_sklearn[y==name,0],

y=Y\_sklearn[y==name,1],

mode='markers',

name=name,

marker=dict(

color=col,

size=12,

line=dict(

color='rgba(217, 217, 217, 0.14)',

width=0.5),

opacity=0.8)

)

data.append(trace)

*# Definisanje izgleda grafika*

layout = dict(

xaxis=dict(title='PC1', showline=False),

yaxis=dict(title='PC2', showline=False)

)

*# Objekat koji nosi informacije o podacima i izgledu grafika*

fig = dict(data=data, layout=layout)

*# Vizualizacija Iris dataset-a u 2D prostoru*

py.iplot(fig, filename='pca-scikitlearn')

Rezultat koji je dobijen u ovom primeru se nalazi na *Slika 8 .*

# Zaključak

Principal Component Analysis je izuzetno korisna i efikasna metoda za smanjenje dimenzionalnosti podataka. Takođe ona odlično ide i uz druge metode kao što su *Correspondence analysis*, *k-means clustering*, *Factor analysis* i *Non-negative matrix factorization*. Pored PCA analize koriste se još i *Independent Component Analysis* i *Network Component Analysis*. Naravno PCA analiza se ne koristi samo u Python programskom jeziku nego i u mnogim drugim, a neki od njih su *Analztica*, *MATLAB*, *Orange*, *R*, *Weka* i mnogi drugi.

**Slike**

[Slika 1 - Prikaz poslednjih 5 torki iz IRIS dataset-a 10](#_Toc99027)

[Slika 2 - Prikaz histograma za svaku dimenziju IRIS dataset-a 11](#_Toc99028)

[Slika 3 - Dobijen rezultat matrice kovarijnse 12](#_Toc99029)

[Slika 4 - Dobijeni Eigenvalues i Eigenvectors na osnovu matrice kovarijanse 13](#_Toc99030)

[Slika 5 - Dobijeni Eigenvalues i Eigenvectors na osnovu matrice kovarijanse 13](#_Toc99031)

[Slika 6 - Prikaz sortiranih Eigenvalues 13](#_Toc99032)

[Slika 7 - Dobijene vrednosti u procentima za svaki PCA atribut 15](#_Toc99033)

[Slika 8 - Prikaz podataka IRIS dataset-a u 2D prostoru 16](#_Toc99034)

**Tabele**

[Tabela 1 - Prikaz podataka 4](#_Toc99073)

[Tabela 2 - Izgled korelacione matrice 5](#_Toc99074)

[Tabela 3 – Prikaz dobijenih ajgenvektora i ajgenvrednosti 8](#_Toc99075)

**Literatura**

1. Multivarijaciona statisticka analiza, 08.02.2013, Principal Component Analysis
2. A tutorial on Principal Components Analysis, 26.02.2002, Lindsay I Smith
3. Metode za smanjenje dimenzionalnosti podataka i njihova primena u prirodnim naukama, 2013, U Novom Sadu, Vladimir Rančić
4. Matematički alati za redukciju dimenzionalnosti signala, 2016, Novi Sad, Srđan Lazendić
5. <https://plot.ly/ipython-notebooks/principal-component-analysis/>
6. <http://scindeks.ceon.rs/article.aspx?query=ISSID%26and%2612267&page=9&sort=8&styp> =0&backurl=%2Fissue.aspx%3Fissue%3D12267
7. https://www.nature.com/articles/nmeth.4346
8. <https://medium.com/@aptrishu/understanding-principle-component-analysis-e32be0253ef0>
9. <http://setosa.io/ev/principal-component-analysis/>
10. <https://www.statisticssolutions.com/principal-component-analysis-pca/>
11. <https://towardsdatascience.com/a-one-stop-shop-for-principal-component-analysis-5582fb7e0a9c>
12. <http://www.matf.bg.ac.rs/p/files/69-pca.html>