ANALIZA SKŁADOWYCH GŁÓWNYCH

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sb
from sklearn.decomposition import PCA
```

Załadowanie zbioru danych:

```
data=pd.read_csv('PCA_OlimpicData.csv')
data
```

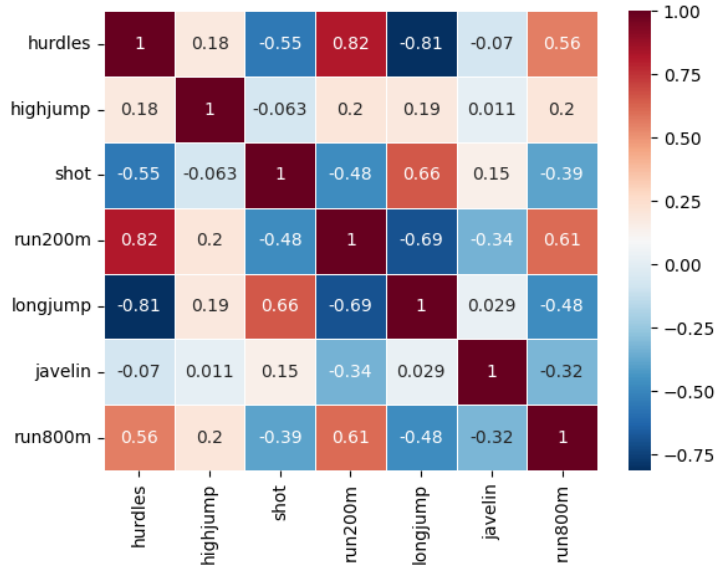
	name	hurdles	highjump	shot	run200m	longjump	javelin	run800m
0	JoynerKersee(USA)	12.69	1.86	15.80	22.56	7.27	45.66	128.51
1	John(GDR)	12.85	1.80	16.23	23.65	6.71	42.56	126.12
2	Behmer(GDR)	13.20	1.83	14.20	23.10	6.68	44.54	124.20
3	Sablovskaite(URS)	13.61	1.80	15.23	23.92	6.25	42.78	132.24
4	Choubenkova(URS)	13.51	1.74	14.76	23.93	6.32	47.46	127.90
5	Schulz(GDR)	13.75	1.83	13.50	24.65	6.33	42.82	125.79
6	Fleming(AUS)	13.38	1.80	12.88	23.59	6.37	40.28	132.54
7	Greiner(USA)	13.55	1.80	14.13	24.48	6.47	38.00	133.65
8	Lajbnerova(CZE)	13.63	1.83	14.28	24.86	6.11	42.20	136.05
9	Bouraga(URS)	13.25	1.77	12.62	23.59	6.28	39.06	134.74
10	Wijnsma(HOL)	13.75	1.86	13.01	25.03	6.34	37.86	131.49
11	Dimitrova(BUL)	13.24	1.80	12.88	23.59	6.37	40.28	132.54
12	Scheider(SWI)	13.85	1.86	11.58	24.87	6.05	47.50	134.93
13	Braun(FRG)	13.71	1.83	13.16	24.78	6.12	44.58	142.82
14	Ruotsalainen(FIN)	13.79	1.80	12.32	24.61	6.08	45.44	137.06
15	Yuping(CHN)	13.93	1.86	14.21	25.00	6.40	38.60	146.67
16	Hagger(GB)	13.47	1.80	12.75	25.47	6.34	35.76	138.48

```
corr=data.corr()
```

```
/var/folders/wy/7w1mlgcx4vjfgbpvhlk2y7m00000gn/T/ipykernel_53160/2057684327.py:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will
  corr=data.corr()
```

```
sb.heatmap(corr,xticklabels=corr.columns,yticklabels=corr.columns,
           cmap='RdBu_r',
           annot=True,
           linewidth=0.5)
```

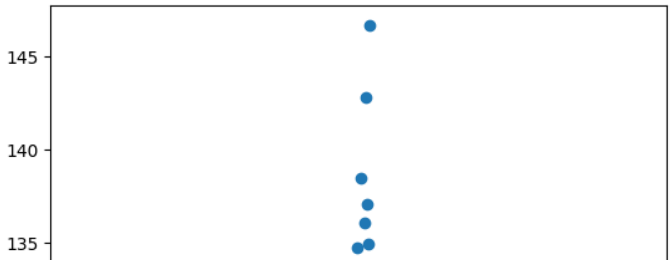
```
<AxesSubplot: >
```



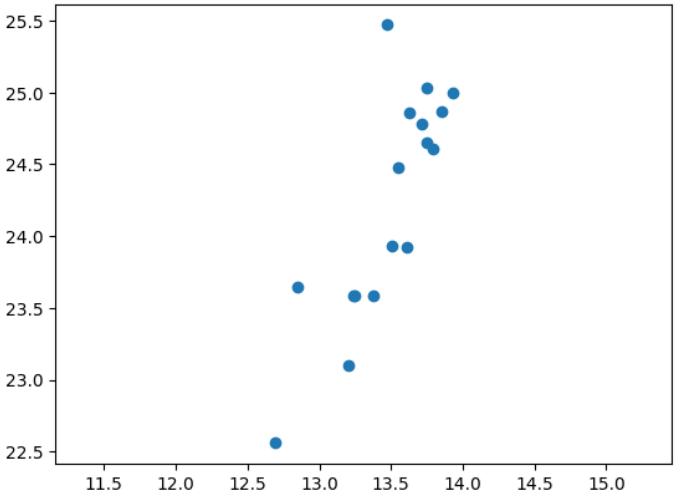

```
CorrMatrix = np.array(data.corr())
CorrMatrix
```

```
/var/folders/wy/7w1mlgcx4vjfgbpvhlk2y7m00000gn/T/ipykernel_53160/2744677357.py:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will
  CorrMatrix = np.array(data.corr())
array([[ 1.        ,  0.18388312, -0.54967946,  0.81674793, -0.81134111,
        -0.06951007,  0.55639401],
       [ 0.18388312,  1.        , -0.06284613,  0.2039109 ,  0.19034412,
         0.01057245,  0.20478232],
       [-0.54967946, -0.06284613,  1.        , -0.47714875,  0.65955868,
         0.15048167, -0.3930256 ],
       [ 0.81674793,  0.2039109 , -0.47714875,  1.        , -0.69418257,
        -0.336521  ,  0.60745396],
       [-0.81134111,  0.19034412,  0.65955868, -0.69418257,  1.        ,
         0.02881087, -0.4771387 ],
       [-0.06951007,  0.01057245,  0.15048167, -0.336521  ,  0.02881087,
         1.        , -0.3192694 ],
       [ 0.55639401,  0.20478232, -0.3930256 ,  0.60745396, -0.4771387 ,
        -0.3192694 ,  1.        ]])
```

```
plt.scatter(data.iloc[:,1], data.iloc[:,7])
plt.axis('equal');
plt.show()
```

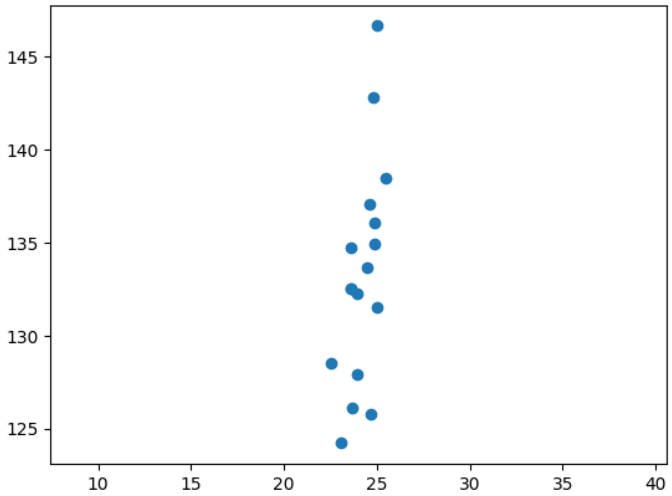



```
plt.scatter(data.iloc[:,1], data.iloc[:,4])
plt.axis('equal');
plt.show()
```




```
plt.scatter(data.iloc[:,3], data.iloc[:,5])
plt.axis('equal');
plt.show()
```


```
plt.scatter(data.iloc[:,4], data.iloc[:,7])
plt.axis('equal');
plt.show()
```




```
w, v = np.linalg.eig(CorrMatrix)
print(w)
print(v)
```

```
[3.52472173 1.18767668 1.03460734 0.08791372 0.13649556 0.57663232
 0.45195264]
[[-0.48130845 -0.07448868  0.21983478 -0.52962087  0.56235013  0.21656792
  -0.26613659]
 [-0.07990832  0.7113087   0.56818366  0.30279504  0.11656062 -0.2146112
  -0.11615824]
 [ 0.38626661  0.17993718 -0.03100608  0.21186781  0.15960862  0.84763563
  -0.16914578]
 [-0.47573064  0.13061435 -0.0250742  -0.06655574 -0.72846937  0.26441386
  -0.38856089]
 [ 0.45637306  0.41654755 -0.02368131 -0.75412328 -0.20548952 -0.05863538
   0.05731478]
 [ 0.15424231 -0.43316332  0.78471893 -0.09218235 -0.26741014  0.17369785
   0.25023655]
 [-0.39289723  0.2791264  -0.10442843 -0.03605205 -0.0173002   0.29159985
   0.81864852]]
```

```
v[:,0]
```

```
array([-0.48130845, -0.07990832,  0.38626661, -0.47573064,  0.45637306,
        0.15424231, -0.39289723])
```

```
v[:,1]
```

```
array([-0.07448868,  0.7113087 ,  0.17993718,  0.13061435,  0.41654755,
       -0.43316332,  0.2791264 ])
```

```
v[:,2]
```

```
array([ 0.21983478,  0.56818366, -0.03100608, -0.0250742 , -0.02368131,
        0.78471893, -0.10442843])
```

```
v[:,3]
```

```
array([-0.52962087,  0.30279504,  0.21186781, -0.06655574, -0.75412328,
       -0.09218235, -0.03605205])
```

```
v[:,4]
```

```
array([ 0.56235013,  0.11656062,  0.15960862, -0.72846937, -0.20548952,
       -0.26741014, -0.0173002 ])
```

```
v[:,5]
```

```
array([ 0.21656792, -0.2146112 ,  0.84763563,  0.26441386, -0.05863538,
        0.17369785,  0.29159985])
```

```
v[:,6]
```

```
array([-0.26613659, -0.11615824, -0.16914578, -0.38856089,  0.05731478,
        0.25023655,  0.81864852])
```

```
data2=data.drop("name", axis='columns')
```

Dwie składowe główne

Wyliczamy dwie składowe główne (**n_components=2**) czyli wektory bazowe nowego układu współrzędnych:

```
pca = PCA()
pca.fit(data2)
cumsum = np.cumsum(pca.explained_variance_ratio_)
d = np.argmax(cumsum >= 0.95) + 1
```

```
d
```

```
pca = PCA(n_components=2)
pca.fit(data2)
```

```
▾ PCA
PCA(n_components=2)
```

Wektory bazowe nowego układu wpółrzędnych:

```
print(pca.components_)
```

```
[[ 0.03058142  0.00104986 -0.08478133  0.08158174 -0.02226627 -0.25600056
   0.958743  ]
 [ 0.02043208  0.00111155 -0.01324563 -0.01541173 -0.01792898  0.96582466
   0.25696226]]
```

Wariancje dla nowych współrzędnych:

```
print(pca.explained_variance_)
```

```
[38.056815   10.59993405]
```

Dodajemy do wykresu wektory wyznaczające nowy układ współrzędnych. Ich długość określona jest przez wariancje:

```
def draw_vector(v0, v1, ax=None):
  ax = ax or plt.gca()
  arrowprops=dict(arrowstyle='->',
  linewidth=2,
  shrinkA=0, shrinkB=0, color='black')
  ax.annotate('', v1, v0, arrowprops=arrowprops)
```

```
pca.explained_variance_
```

```
array([38.056815  , 10.59993405])
```

```
#plt.scatter(data2.iloc[:, 0], data2.iloc[:, 4], alpha=0.2)
#for length, vector in zip(pca.explained_variance_, pca.components_):
 # v = vector * 3 * np.sqrt(length)
 # draw_vector(pca.mean_, pca.mean_ + v)
#plt.axis('equal');
```

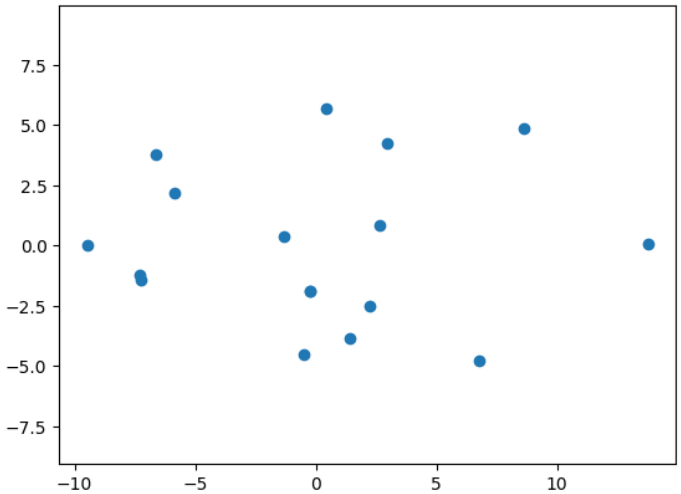
Współrzędne punktów w **nowym układzie współrzędnych**:

```
data_pca = pca.transform(data2)
data_pca
```

```
array([[-5.84157054,  2.19745394],
       [-7.2695973 , -1.41999405],
       [-9.47862585,  0.04205876],
       [-1.31811748,  0.3979566 ],
```

```
            [-6.64116148,  3.80550619],
            [-7.30349189, -1.20769311],
            [-0.22788469, -1.91015413],
            [ 1.38960463, -3.85559916],
            [ 2.64416319,  0.81785289],
            [ 2.21371063, -2.52077528],
            [-0.49654123, -4.53301057],
            [-0.23216609, -1.91301462],
            [ 0.45138884,  5.69013908],
            [ 8.61622437,  4.87367342],
            [ 2.93435729,  4.24024469],
            [13.76772074,  0.06955646],
            [ 6.79198686, -4.77420111]])
----> 6 ax.annotate('', v1, v0, arrowprops=arrowprops)
```

```
plt.scatter(data_pca[:,0], data_pca[:,1])
plt.axis('equal');
```



Jedna składowa główna

Wyliczamy jedną składową główną (**n_components=1**) - chcemy wyeliminować jeden wymiar danych.

```
pca = PCA(n_components=1)
pca.fit(data2)
```

```
PCA
PCA(n_components=1)
```

Współrzędne punktów w **nowym układzie współrzędnych**:

```
data_pca = pca.transform(data2)
```

Porównanie kształtów danych początkowych i po redukcji jednego wymiaru:

```
print("Początkowy shape: ", data2.shape)
print("Po transformacji shape:", data_pca.shape)
```

```
Początkowy shape:  (17, 7)
Po transformacji shape: (17, 1)
```

Dane początkowe i po redukcji wymiaru:

```
data_new = pca.inverse_transform(data_pca)
plt.scatter(data2.iloc[:, 0], data2.iloc[:, 1]),
plt.scatter(data_new[:, 0], data_new[:, 1]),
plt.axis('equal');
```