


```
In [1]: from sklearn.preprocessing import LabelEncoder
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import numpy as np
```

```
In [2]: df = pd.read_csv('diabetes.csv')
df.head()
```

```
Out[2]:
```

	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunct
0	148	72	35	0	33.6	0.
1	85	66	29	0	26.6	0.
2	183	64	0	0	23.3	0.
3	89	66	23	94	28.1	0.
4	137	40	35	168	43.1	2.



```
In [3]: df.isnull().sum()
```

```
Out[3]: Glucose      0
BloodPressure    0
SkinThickness    0
Insulin          0
BMI              0
DiabetesPedigreeFunction  0
Age              0
Outcome          0
dtype: int64
```

```
In [4]: df.duplicated().sum()
```

```
Out[4]: 0
```

```
In [5]: df.info()
```

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 8 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   Glucose                               768 non-null    int64
1   BloodPressure                         768 non-null    int64
2   SkinThickness                         768 non-null    int64
3   Insulin                              768 non-null    int64
4   BMI                                   768 non-null    float64
5   DiabetesPedigreeFunction              768 non-null    float64
6   Age                                   768 non-null    int64
7   Outcome                               768 non-null    int64
dtypes: float64(2), int64(6)
memory usage: 48.1 KB

```

In [7]: `df.describe()`

Out[7]:

	Glucose	BloodPressure	SkinThickness	Insulin	BMI	Diabe
count	768.000000	768.000000	768.000000	768.000000	768.000000	
mean	120.894531	69.105469	20.536458	79.799479	31.992578	
std	31.972618	19.355807	15.952218	115.244002	7.884160	
min	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	99.000000	62.000000	0.000000	0.000000	27.300000	
50%	117.000000	72.000000	23.000000	30.500000	32.000000	
75%	140.250000	80.000000	32.000000	127.250000	36.600000	
max	199.000000	122.000000	99.000000	846.000000	67.100000	

In [9]: `df.shape`

Out[9]: (768, 8)

In [8]:

```

for column in df.columns:
    distinct_values = df[column].unique()
    print(f"{column}: {len(distinct_values)} values")
    print(distinct_values)
    print("-" * 50)

```

Glucose: 136 values

```
[148 85 183 89 137 116 78 115 197 125 110 168 139 189 166 100 118 107
 103 126 99 196 119 143 147 97 145 117 109 158 88 92 122 138 102 90
 111 180 133 106 171 159 146 71 105 101 176 150 73 187 84 44 141 114
 95 129 79 0 62 131 112 113 74 83 136 80 123 81 134 142 144 93
 163 151 96 155 76 160 124 162 132 120 173 170 128 108 154 57 156 153
 188 152 104 87 75 179 130 194 181 135 184 140 177 164 91 165 86 193
 191 161 167 77 182 157 178 61 98 127 82 72 172 94 175 195 68 186
 198 121 67 174 199 56 169 149 65 190]
```

BloodPressure: 47 values

```
[ 72 66 64 40 74 50 0 70 96 92 80 60 84 30 88 90 94 76
 82 75 58 78 68 110 56 62 85 86 48 44 65 108 55 122 54 52
 98 104 95 46 102 100 61 24 38 106 114]
```

SkinThickness: 51 values

```
[35 29 0 23 32 45 19 47 38 30 41 33 26 15 36 11 31 37 42 25 18 24 39 27
 21 34 10 60 13 20 22 28 54 40 51 56 14 17 50 44 12 46 16 7 52 43 48 8
 49 63 99]
```

Insulin: 186 values

```
[ 0 94 168 88 543 846 175 230 83 96 235 146 115 140 110 245 54 192
 207 70 240 82 36 23 300 342 304 142 128 38 100 90 270 71 125 176
 48 64 228 76 220 40 152 18 135 495 37 51 99 145 225 49 50 92
 325 63 284 119 204 155 485 53 114 105 285 156 78 130 55 58 160 210
 318 44 190 280 87 271 129 120 478 56 32 744 370 45 194 680 402 258
 375 150 67 57 116 278 122 545 75 74 182 360 215 184 42 132 148 180
 205 85 231 29 68 52 255 171 73 108 43 167 249 293 66 465 89 158
 84 72 59 81 196 415 275 165 579 310 61 474 170 277 60 14 95 237
 191 328 250 480 265 193 79 86 326 188 106 65 166 274 77 126 330 600
 185 25 41 272 321 144 15 183 91 46 440 159 540 200 335 387 22 291
 392 178 127 510 16 112]
```

BMI: 248 values

```
[33.6 26.6 23.3 28.1 43.1 25.6 31. 35.3 30.5 0. 37.6 38. 27.1 30.1
 25.8 30. 45.8 29.6 43.3 34.6 39.3 35.4 39.8 29. 36.6 31.1 39.4 23.2
 22.2 34.1 36. 31.6 24.8 19.9 27.6 24. 33.2 32.9 38.2 37.1 34. 40.2
 22.7 45.4 27.4 42. 29.7 28. 39.1 19.4 24.2 24.4 33.7 34.7 23. 37.7
 46.8 40.5 41.5 25. 25.4 32.8 32.5 42.7 19.6 28.9 28.6 43.4 35.1 32.
 24.7 32.6 43.2 22.4 29.3 24.6 48.8 32.4 38.5 26.5 19.1 46.7 23.8 33.9
 20.4 28.7 49.7 39. 26.1 22.5 39.6 29.5 34.3 37.4 33.3 31.2 28.2 53.2
 34.2 26.8 55. 42.9 34.5 27.9 38.3 21.1 33.8 30.8 36.9 39.5 27.3 21.9
 40.6 47.9 50. 25.2 40.9 37.2 44.2 29.9 31.9 28.4 43.5 32.7 67.1 45.
 34.9 27.7 35.9 22.6 33.1 30.4 52.3 24.3 22.9 34.8 30.9 40.1 23.9 37.5
 35.5 42.8 42.6 41.8 35.8 37.8 28.8 23.6 35.7 36.7 45.2 44. 46.2 35.
 43.6 44.1 18.4 29.2 25.9 32.1 36.3 40. 25.1 27.5 45.6 27.8 24.9 25.3
 37.9 27. 26. 38.7 20.8 36.1 30.7 32.3 52.9 21. 39.7 25.5 26.2 19.3
 38.1 23.5 45.5 23.1 39.9 36.8 21.8 41. 42.2 34.4 27.2 36.5 29.8 39.2
 38.4 36.2 48.3 20. 22.3 45.7 23.7 22.1 42.1 42.4 18.2 26.4 45.3 37.
 24.5 32.2 59.4 21.2 26.7 30.2 46.1 41.3 38.8 35.2 42.3 40.7 46.5 33.5
 37.3 30.3 26.3 21.7 36.4 28.5 26.9 38.6 31.3 19.5 20.1 40.8 23.4 28.3
 38.9 57.3 35.6 49.6 44.6 24.1 44.5 41.2 49.3 46.3]
```

DiabetesPedigreeFunction: 517 values

```
[0.627 0.351 0.672 0.167 2.288 0.201 0.248 0.134 0.158 0.232 0.191 0.537
 1.441 0.398 0.587 0.484 0.551 0.254 0.183 0.529 0.704 0.388 0.451 0.263]
```

```
0.205 0.257 0.487 0.245 0.337 0.546 0.851 0.267 0.188 0.512 0.966 0.42
0.665 0.503 1.39 0.271 0.696 0.235 0.721 0.294 1.893 0.564 0.586 0.344
0.305 0.491 0.526 0.342 0.467 0.718 0.962 1.781 0.173 0.304 0.27 0.699
0.258 0.203 0.855 0.845 0.334 0.189 0.867 0.411 0.583 0.231 0.396 0.14
0.391 0.37 0.307 0.102 0.767 0.237 0.227 0.698 0.178 0.324 0.153 0.165
0.443 0.261 0.277 0.761 0.255 0.13 0.323 0.356 0.325 1.222 0.179 0.262
0.283 0.93 0.801 0.207 0.287 0.336 0.247 0.199 0.543 0.192 0.588 0.539
0.22 0.654 0.223 0.759 0.26 0.404 0.186 0.278 0.496 0.452 0.403 0.741
0.361 1.114 0.457 0.647 0.088 0.597 0.532 0.703 0.159 0.268 0.286 0.318
0.272 0.572 0.096 1.4 0.218 0.085 0.399 0.432 1.189 0.687 0.137 0.637
0.833 0.229 0.817 0.204 0.368 0.743 0.722 0.256 0.709 0.471 0.495 0.18
0.542 0.773 0.678 0.719 0.382 0.319 0.19 0.956 0.084 0.725 0.299 0.244
0.745 0.615 1.321 0.64 0.142 0.374 0.383 0.578 0.136 0.395 0.187 0.905
0.15 0.874 0.236 0.787 0.407 0.605 0.151 0.289 0.355 0.29 0.375 0.164
0.431 0.742 0.514 0.464 1.224 1.072 0.805 0.209 0.666 0.101 0.198 0.652
2.329 0.089 0.645 0.238 0.394 0.293 0.479 0.686 0.831 0.582 0.446 0.402
1.318 0.329 1.213 0.427 0.282 0.143 0.38 0.284 0.249 0.926 0.557 0.092
0.655 1.353 0.612 0.2 0.226 0.997 0.933 1.101 0.078 0.24 1.136 0.128
0.422 0.251 0.677 0.296 0.454 0.744 0.881 0.28 0.259 0.619 0.808 0.34
0.434 0.757 0.613 0.692 0.52 0.412 0.84 0.839 0.156 0.215 0.326 1.391
0.875 0.313 0.433 0.626 1.127 0.315 0.345 0.129 0.527 0.197 0.731 0.148
0.123 0.127 0.122 1.476 0.166 0.932 0.343 0.893 0.331 0.472 0.673 0.389
0.485 0.349 0.279 0.346 0.252 0.243 0.58 0.559 0.302 0.569 0.378 0.385
0.499 0.306 0.234 2.137 1.731 0.545 0.225 0.816 0.528 0.509 1.021 0.821
0.947 1.268 0.221 0.66 0.239 0.949 0.444 0.463 0.803 1.6 0.944 0.196
0.241 0.161 0.135 0.376 1.191 0.702 0.674 1.076 0.534 1.095 0.554 0.624
0.219 0.507 0.561 0.421 0.516 0.264 0.328 0.233 0.108 1.138 0.147 0.727
0.435 0.497 0.23 0.955 2.42 0.658 0.33 0.51 0.285 0.415 0.381 0.832
0.498 0.212 0.364 1.001 0.46 0.733 0.416 0.705 1.022 0.269 0.6 0.571
0.607 0.17 0.21 0.126 0.711 0.466 0.162 0.419 0.63 0.365 0.536 1.159
0.629 0.292 0.145 1.144 0.174 0.547 0.163 0.738 0.314 0.968 0.409 0.297
0.525 0.154 0.771 0.107 0.493 0.717 0.917 0.501 1.251 0.735 0.804 0.661
0.549 0.825 0.423 1.034 0.16 0.341 0.68 0.591 0.3 0.121 0.502 0.401
0.601 0.748 0.338 0.43 0.892 0.813 0.693 0.575 0.371 0.206 0.417 1.154
0.925 0.175 1.699 0.682 0.194 0.4 0.1 1.258 0.482 0.138 0.593 0.878
0.157 1.282 0.141 0.246 1.698 1.461 0.347 0.362 0.393 0.144 0.732 0.115
0.465 0.649 0.871 0.149 0.695 0.303 0.61 0.73 0.447 0.455 0.133 0.155
1.162 1.292 0.182 1.394 0.217 0.631 0.88 0.614 0.332 0.366 0.181 0.828
0.335 0.856 0.886 0.439 0.253 0.598 0.904 0.483 0.565 0.118 0.177 0.176
0.295 0.441 0.352 0.826 0.97 0.595 0.317 0.265 0.646 0.426 0.56 0.515
0.453 0.785 0.734 1.174 0.488 0.358 1.096 0.408 1.182 0.222 1.057 0.766
0.171]
```

Age: 52 values

```
[50 31 32 21 33 30 26 29 53 54 34 57 59 51 27 41 43 22 38 60 28 45 35 46
 56 37 48 40 25 24 58 42 44 39 36 23 61 69 62 55 65 47 52 66 49 63 67 72
 81 64 70 68]
```

Outcome: 2 values

```
[1 0]
```

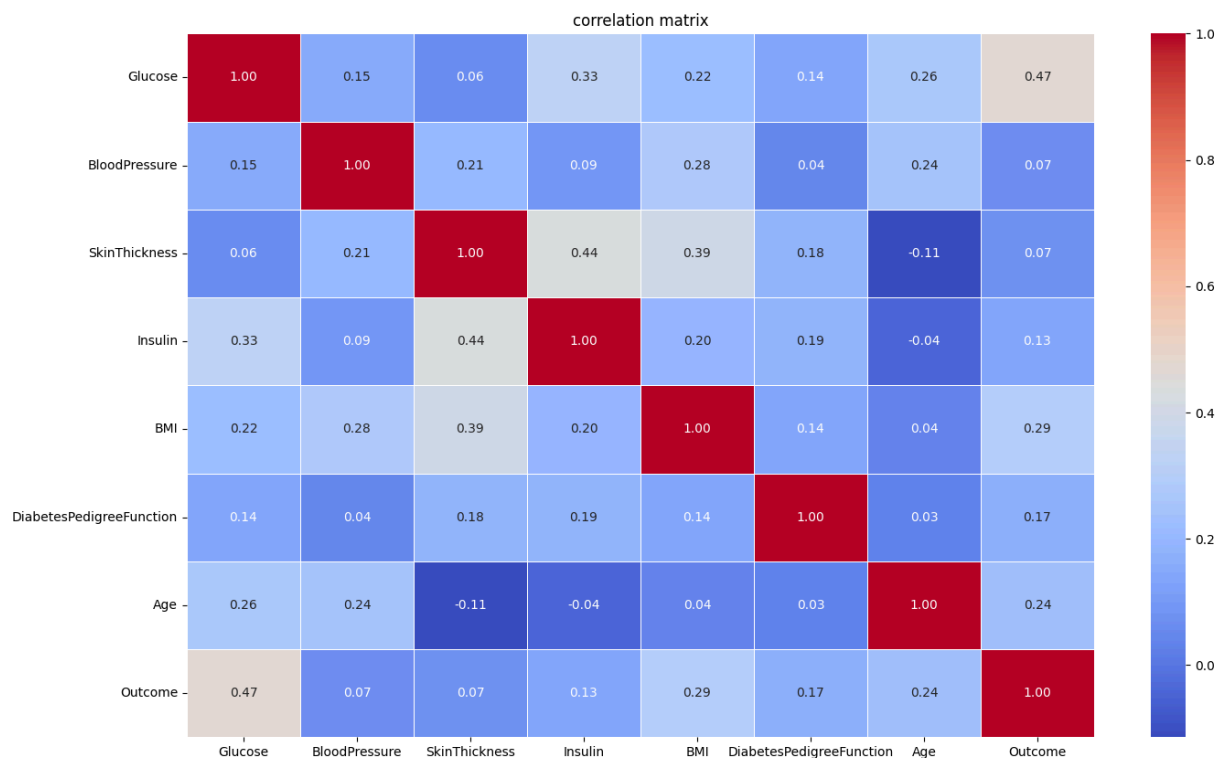
the correlation matrix

```
In [10]: correlation_matrix = df.corr()
correlation_matrix
```

```
Out[10]:
```

	Glucose	BloodPressure	SkinThickness	Insulin
Glucose	1.000000	0.152590	0.057328	0.331357
BloodPressure	0.152590	1.000000	0.207371	0.088933
SkinThickness	0.057328	0.207371	1.000000	0.436783
Insulin	0.331357	0.088933	0.436783	1.000000
BMI	0.221071	0.281805	0.392573	0.197859
DiabetesPedigreeFunction	0.137337	0.041265	0.183928	0.185071
Age	0.263514	0.239528	-0.113970	-0.042163
Outcome	0.466581	0.065068	0.074752	0.130548

```
In [11]: plt.figure(figsize=(16, 10))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f", line
plt.title("correlation matrix")
plt.show()
```



eigen values, percentage of inertia,
commulative percentage

```
In [12]: eigen_values = np.linalg.eigvals(correlation_matrix)
eigen_values_sorted = np.sort(eigen_values)[::-1]
inertia_percentage = (eigen_values_sorted / eigen_values_sorted.sum()) * 100
cumulative_percentage = np.cumsum(inertia_percentage)

result_df = pd.DataFrame({
    'eigen values': eigen_values_sorted,
    'percentage of inertia': inertia_percentage,
    'cumulative percentage': cumulative_percentage
}, index=range(1, len(eigen_values_sorted) + 1))

result_df.index.name = 'component'

result_df.round(2)
```

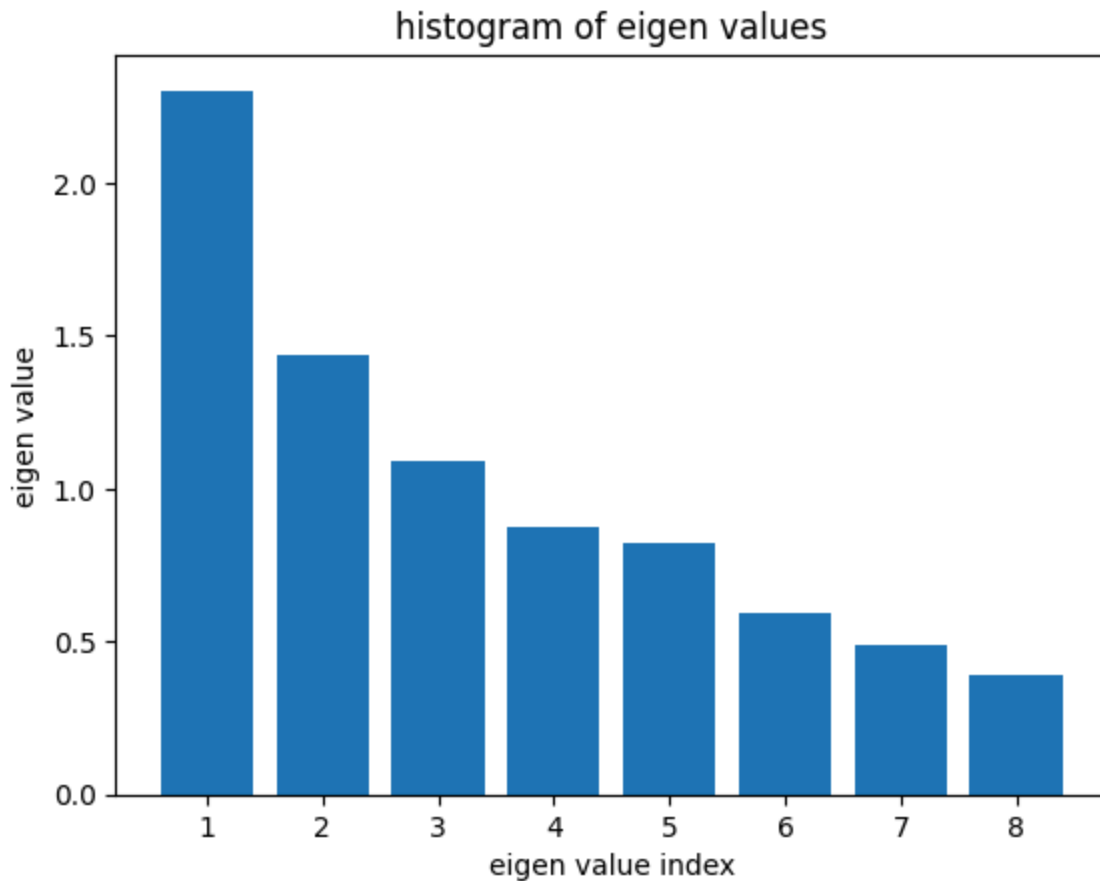
```
Out[12]:
```

	eigen values	percentage of inertia	cumulative percentage
component			

component			
1	2.30	28.78	28.78
2	1.43	17.93	46.70
3	1.09	13.66	60.36
4	0.87	10.93	71.29
5	0.83	10.32	81.61
6	0.59	7.42	89.03
7	0.49	6.08	95.11
8	0.39	4.89	100.00

histogram of eigen values

```
In [13]: plt.bar(range(1, len(eigen_values_sorted) + 1), eigen_values_sorted)
plt.xlabel('eigen value index')
plt.ylabel('eigen value')
plt.title('histogram of eigen values')
plt.show()
```



compute the principal components,
contributions, representational qualities of
individuals

```
In [14]: pca = PCA()
pca.fit(df)

principal_components = pd.DataFrame(pca.components_, columns=df.columns)
contributions = pd.DataFrame(np.abs(principal_components), columns=df.columns)
representational_qualities = np.square(contributions)
```

```
In [20]: print("principal components:")
print(principal_components)
```

```
principal components:
      Glucose  BloodPressure  SkinThickness  Insulin      BMI  \
0  0.097815      0.016095      0.060756  0.993112  0.014011
1  0.972553      0.141633      -0.057843 -0.094713  0.046982
2 -0.142387      0.922829      0.307662 -0.021181  0.132685
3  0.118136     -0.267279      0.887085 -0.065352  0.192929
4 -0.087851     -0.225609      0.249774  0.000118  0.018969
5 -0.050873     -0.075665     -0.221342  0.006133  0.970677
6 -0.006078      0.002419     -0.001054  0.000107 -0.013973
7  0.000509      0.000013     -0.002379 -0.000309  0.000380

      DiabetesPedigreeFunction      Age      Outcome
0          0.000537 -0.003561  0.000585
1          0.000818  0.139670  0.007007
2          0.000644  0.124107 -0.000317
3          0.002703 -0.293562  0.002711
4          0.001687  0.937335  0.005965
5          0.002039  0.016270  0.013175
6          0.217248 -0.006026  0.975975
7          0.976108  0.000303 -0.217270
```

```
In [21]: print("contributions:")
         print(contributions)
```

```
contributions:
      Glucose  BloodPressure  SkinThickness  Insulin      BMI  \
0  0.097815      0.016095      0.060756  0.993112  0.014011
1  0.972553      0.141633      0.057843  0.094713  0.046982
2  0.142387      0.922829      0.307662  0.021181  0.132685
3  0.118136      0.267279      0.887085  0.065352  0.192929
4  0.087851      0.225609      0.249774  0.000118  0.018969
5  0.050873      0.075665      0.221342  0.006133  0.970677
6  0.006078      0.002419      0.001054  0.000107  0.013973
7  0.000509      0.000013      0.002379  0.000309  0.000380

      DiabetesPedigreeFunction      Age      Outcome
0          0.000537  0.003561  0.000585
1          0.000818  0.139670  0.007007
2          0.000644  0.124107  0.000317
3          0.002703  0.293562  0.002711
4          0.001687  0.937335  0.005965
5          0.002039  0.016270  0.013175
6          0.217248  0.006026  0.975975
7          0.976108  0.000303  0.217270
```

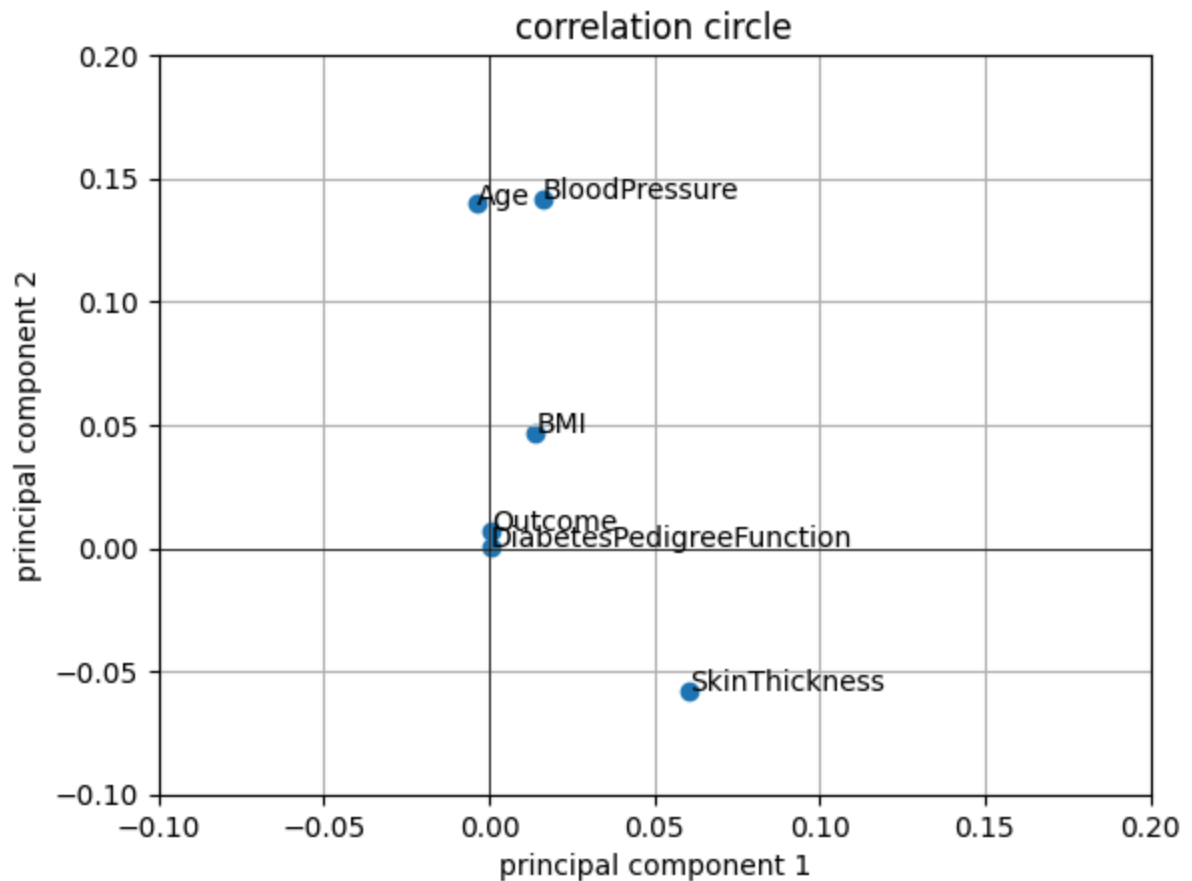
```
In [23]: print("representational qualities:")
         print(representational_qualities)
```


	representational	qualities:				
	Glucose	BloodPressure	SkinThickness	Insulin	BMI	\
0	9.567729e-03	2.590369e-04	0.003691	9.862723e-01	1.963087e-04	
1	9.458592e-01	2.005985e-02	0.003346	8.970552e-03	2.207333e-03	
2	2.027411e-02	8.516128e-01	0.094656	4.486158e-04	1.760537e-02	
3	1.395605e-02	7.143782e-02	0.786921	4.270824e-03	3.722147e-02	
4	7.717711e-03	5.089948e-02	0.062387	1.384397e-08	3.598251e-04	
5	2.588034e-03	5.725203e-03	0.048992	3.761441e-05	9.422143e-01	
6	3.694366e-05	5.853217e-06	0.000001	1.137727e-08	1.952416e-04	
7	2.588843e-07	1.722455e-10	0.000006	9.543914e-08	1.445423e-07	

	DiabetesPedigreeFunction	Age	Outcome
0	2.885481e-07	1.268357e-05	3.426749e-07
1	6.692062e-07	1.950759e-02	4.909116e-05
2	4.150814e-07	1.540261e-02	1.006460e-07
3	7.308077e-06	8.617860e-02	7.351372e-06
4	2.846191e-06	8.785974e-01	3.558533e-05
5	4.158663e-06	2.646999e-04	1.735693e-04
6	4.719683e-02	3.631234e-05	9.525277e-01
7	9.527875e-01	9.189848e-08	4.720626e-02

the individuals in the first factorial

```
In [27]: fig, ax = plt.subplots()
ax.scatter(pca.components_[0, :], pca.components_[1, :])
for i, txt in enumerate(df.columns):
    ax.annotate(txt, (pca.components_[0, i], pca.components_[1, i]))
ax.set_xlim(-0.1, 0.2)
ax.set_ylim(-0.1, 0.2)
ax.axhline(0, color='black', linewidth=0.5)
ax.axvline(0, color='black', linewidth=0.5)
plt.xlabel('principal component 1')
plt.ylabel('principal component 2')
plt.title('correlation circle')
plt.grid()
plt.show()
```



compute the principal components,
contributions, representational qualities of
the variables.

```
In [28]: pca_var = PCA()
pca_var.fit(df)
principal_components_var = pd.DataFrame(pca_var.components_, columns=df.columns)
contributions_var = pd.DataFrame(np.abs(principal_components_var), columns=df.columns)
representational_qualities_var = np.square(contributions_var)
```

```
In [29]: print("principal components:")
print(principal_components_var)
```

```
principal components:
      Glucose  BloodPressure  SkinThickness  Insulin      BMI  \
0  0.097815      0.016095      0.060756  0.993112  0.014011
1  0.972553      0.141633     -0.057843 -0.094713  0.046982
2 -0.142387      0.922829      0.307662 -0.021181  0.132685
3  0.118136     -0.267279      0.887085 -0.065352  0.192929
4 -0.087851     -0.225609      0.249774  0.000118  0.018969
5 -0.050873     -0.075665     -0.221342  0.006133  0.970677
6 -0.006078      0.002419     -0.001054  0.000107 -0.013973
7  0.000509      0.000013     -0.002379 -0.000309  0.000380

      DiabetesPedigreeFunction      Age      Outcome
0          0.000537 -0.003561  0.000585
1          0.000818  0.139670  0.007007
2          0.000644  0.124107 -0.000317
3          0.002703 -0.293562  0.002711
4          0.001687  0.937335  0.005965
5          0.002039  0.016270  0.013175
6          0.217248 -0.006026  0.975975
7          0.976108  0.000303 -0.217270
```

```
In [30]: print("contributions:")
         print(contributions_var)
```

```
contributions:
      Glucose  BloodPressure  SkinThickness  Insulin      BMI  \
0  0.097815      0.016095      0.060756  0.993112  0.014011
1  0.972553      0.141633      0.057843  0.094713  0.046982
2  0.142387      0.922829      0.307662  0.021181  0.132685
3  0.118136      0.267279      0.887085  0.065352  0.192929
4  0.087851      0.225609      0.249774  0.000118  0.018969
5  0.050873      0.075665      0.221342  0.006133  0.970677
6  0.006078      0.002419      0.001054  0.000107  0.013973
7  0.000509      0.000013      0.002379  0.000309  0.000380

      DiabetesPedigreeFunction      Age      Outcome
0          0.000537  0.003561  0.000585
1          0.000818  0.139670  0.007007
2          0.000644  0.124107  0.000317
3          0.002703  0.293562  0.002711
4          0.001687  0.937335  0.005965
5          0.002039  0.016270  0.013175
6          0.217248  0.006026  0.975975
7          0.976108  0.000303  0.217270
```

```
In [31]: print("representational Qualities:")
         print(representational_qualities_var)
```

representational Qualities:

	Glucose	BloodPressure	SkinThickness	Insulin	BMI \
0	9.567729e-03	2.590369e-04	0.003691	9.862723e-01	1.963087e-04
1	9.458592e-01	2.005985e-02	0.003346	8.970552e-03	2.207333e-03
2	2.027411e-02	8.516128e-01	0.094656	4.486158e-04	1.760537e-02
3	1.395605e-02	7.143782e-02	0.786921	4.270824e-03	3.722147e-02
4	7.717711e-03	5.089948e-02	0.062387	1.384397e-08	3.598251e-04
5	2.588034e-03	5.725203e-03	0.048992	3.761441e-05	9.422143e-01
6	3.694366e-05	5.853217e-06	0.000001	1.137727e-08	1.952416e-04
7	2.588843e-07	1.722455e-10	0.000006	9.543914e-08	1.445423e-07

	DiabetesPedigreeFunction	Age	Outcome
0	2.885481e-07	1.268357e-05	3.426749e-07
1	6.692062e-07	1.950759e-02	4.909116e-05
2	4.150814e-07	1.540261e-02	1.006460e-07
3	7.308077e-06	8.617860e-02	7.351372e-06
4	2.846191e-06	8.785974e-01	3.558533e-05
5	4.158663e-06	2.646999e-04	1.735693e-04
6	4.719683e-02	3.631234e-05	9.525277e-01
7	9.527875e-01	9.189848e-08	4.720626e-02

Question 8:

Plot the correlation circle.

```
In [32]: fig, ax = plt.subplots(figsize=(18, 18))
for i, var in enumerate(df.columns):
    ax.arrow(0, 0, principal_components[var][0], principal_components[var][1],
            color='r', alpha=0.5, head_width=0.05)
    ax.text(principal_components[var][0] * 1.1, principal_components[var][1],
            color='r', ha='center', va='center')

circle = plt.Circle((0, 0), radius=0.85, color='b', fill=False)
ax.add_patch(circle)

plt.xlabel('principal component 1')
plt.ylabel('principal component 2')
plt.title('correlation circle')
plt.axhline(0, color='black', lw=0.5)
plt.axvline(0, color='black', lw=0.5)
plt.grid(True, linestyle='--', alpha=0.7)
plt.show()
```

