PCA i zbalansowanie zbioru treningowego

Marcin Wierzbiński

Titanic data

X:

	Age	Fare
1	38.0	71.2833
3	35.0	53.1000
6	54.0	51.8625
10	4.0	16.7000
11	58.0	26.5500

y:

Survived
1
1
0
1
1

```
data['Survived'].value_counts()

1    123
0     60
Name: Survived, dtype: int64
```

Mnist is balanced

```
mnist_train_dataset = np.genfromtxt('/content/sample_data/mnist_train_small.csv', delimiter=',')
mnist_test_dataset = np.genfromtxt('/content/sample_data/mnist_test.csv', delimiter=',')
X_train = mnist_train_dataset[:, 1:]
y_train = mnist_train_dataset[:, 0].astype(int)
X_test = mnist_test_dataset[:, 1:]
y_test = mnist_test_dataset[:, 0].astype(int)

np.unique(y_test, return_counts=True)
```

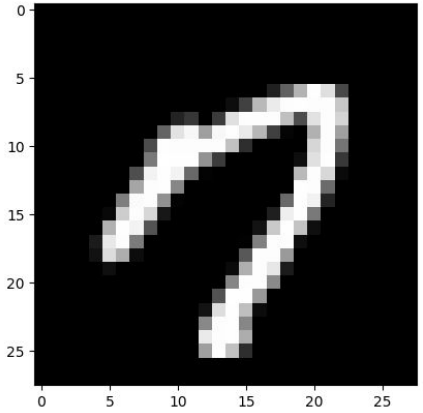
```
np.unique(y_train, return_counts=True)
```

```
(array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9]),
 array([1962, 2243, 1989, 2021, 1924, 1761, 2039, 2126, 1912, 2023]))
```

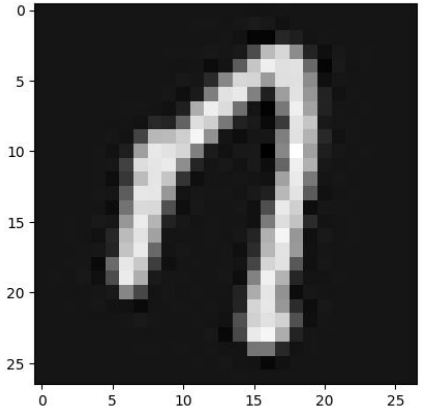
```
np.unique(y_test, return_counts=True)
```

```
(array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9]),
 array([ 980, 1135, 1032, 1010,  982,  892,  958, 1028,  974, 1009]))
```

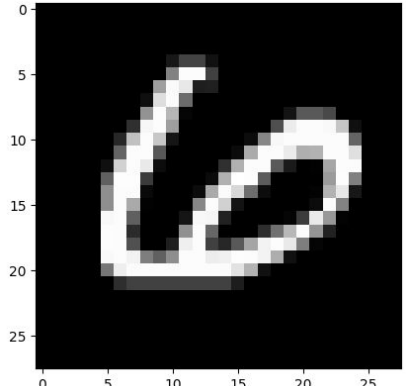
Augmentation of data



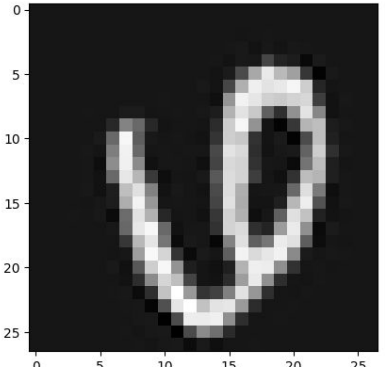
rotate by 20 degree



problem:



rotate by 45 degree



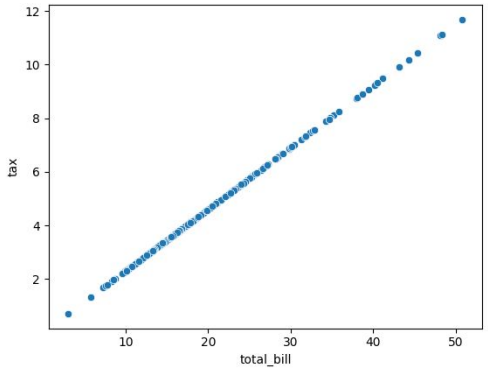
Correlation

tax = 0.23 * total_bill

	total_bill	tip	sex	smoker	day	time	size	tax
0	16.99	1.01	Female	No	Sun	Dinner	2	3.9077
1	10.34	1.66	Male	No	Sun	Dinner	3	2.3782
2	21.01	3.50	Male	No	Sun	Dinner	3	4.8323
3	23.68	3.31	Male	No	Sun	Dinner	2	5.4464
4	24.59	3.61	Female	No	Sun	Dinner	4	5.6557

correlation matrix

	total_bill	tip	size	tax
total_bill	1.000000	0.675734	0.598315	1.000000
tip	0.675734	1.000000	0.489299	0.675734
size	0.598315	0.489299	1.000000	0.598315
tax	1.000000	0.675734	0.598315	1.000000

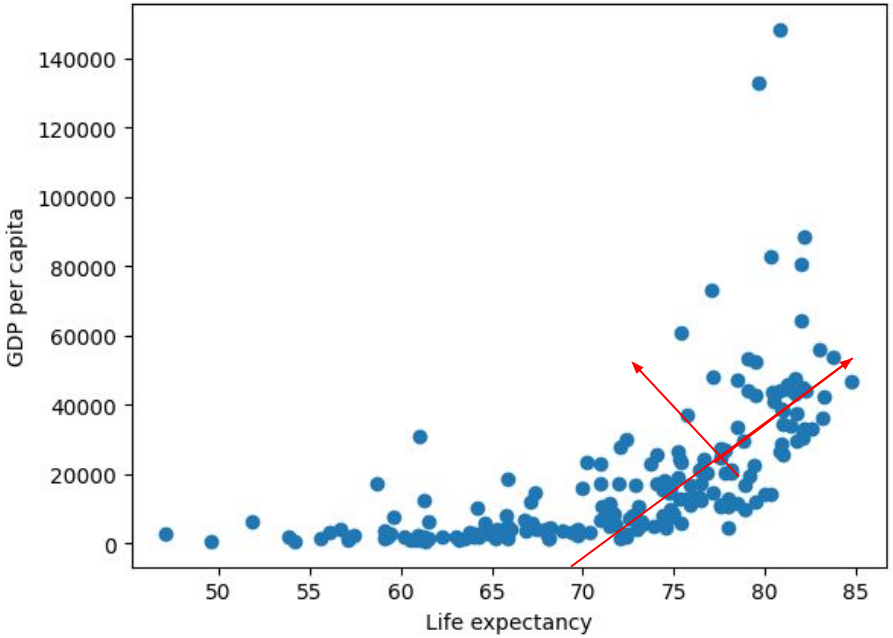


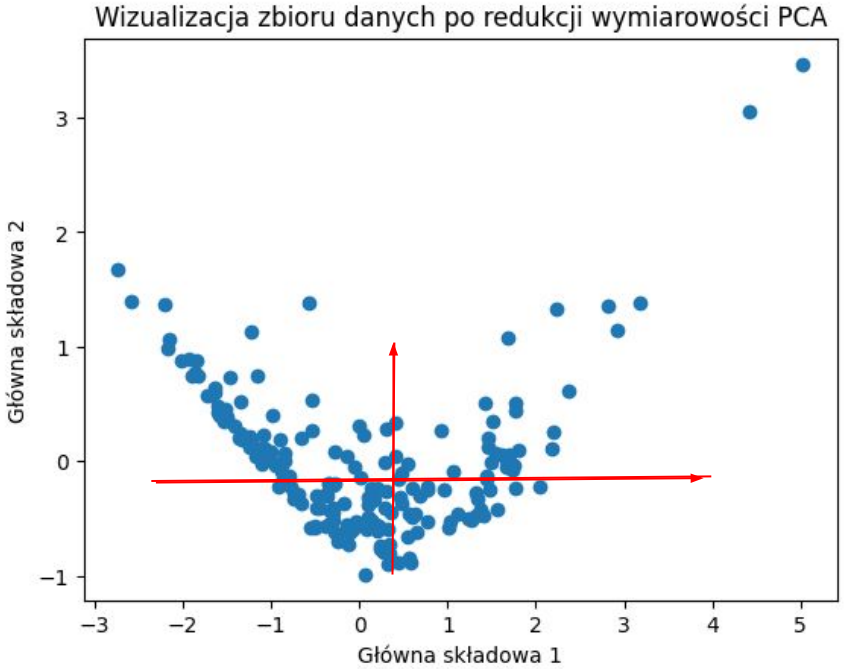
Principal Component Analysis (PCA)

- Technique used to reduce the dimensionality of dataset
- Goal: Transform of a large set of variables into a smaller set of variables, while retaining as much information as possible.
- Achievements: to identify patterns in data by fining the direction of maximum variance in high dimensional data
- The PCA first calculate the covariance matrix of dataset
- Then finding the directions of maximum variance in the data, and amount of variance explained.

Example:

Explained variance ratio: [0.8030581 0.1969419]



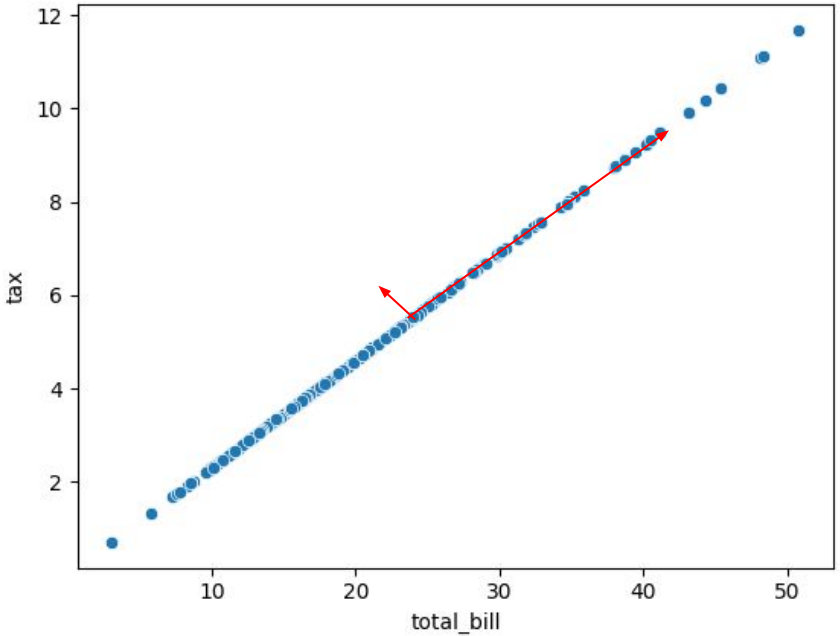


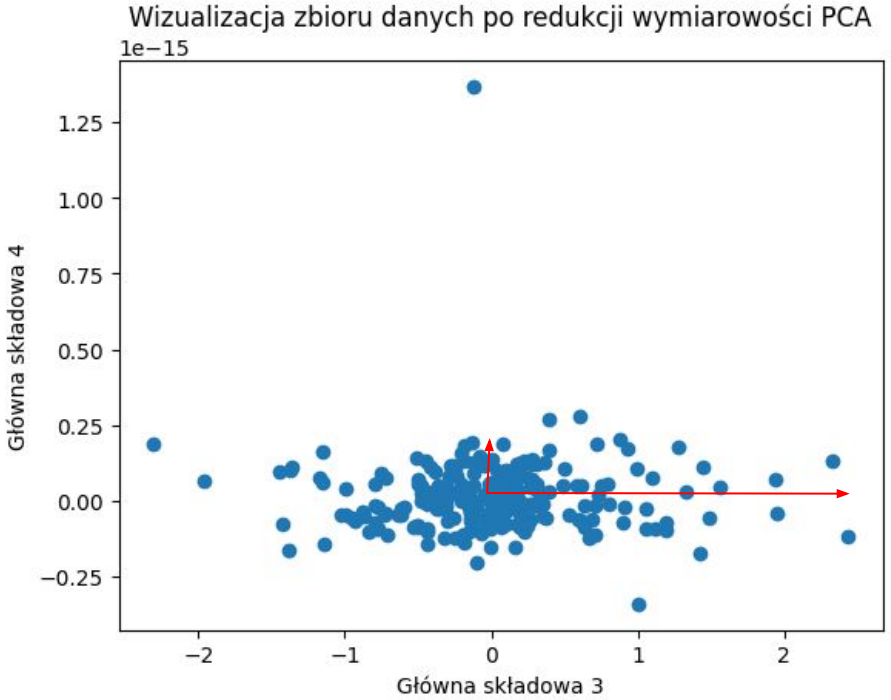
Implementation for tips

```
X = tips[['tip', 'total_bill', 'tax', 'size']]

X  = (X - X.mean())/ X.std()
pca =  PCA(n_components=4)
X_trans = pca.fit_transform(X)
```

Toy example:


12
10
8
6
4
2
tax
10
20
30
40
50
total_bill


Wizualizacja zbioru danych po redukcji wymiarowości PCA
1e−15
1.25
1.00
0.75
0.50
0.25
0.00
−0.25
Główna składowa 4
−2
−1
0
1
2
Główna składowa 3

PCA sklearn

```
from sklearn.decomposition import PCA
import numpy as np

# Załadowanie danych
data = pd.read_csv('income.csv', sep=';')
data = data[['GDP per capita', 'Life expectancy']]
# Wykonanie standaryzacji danych
mean = np.mean(data, axis=0)
std = np.std(data, axis=0)
data_standardized = (data - mean) / std
# Tworzenie instancji klasy PCA i wyznaczanie głównych składowych
pca = PCA(n_components=2)  # redukcja do dwóch wymiarów
principal_components = pca.fit_transform(data_standardized)
# Wyświetlenie wyjaśnionej wariancji dla każdej z głównych składowych
print("Wyjaśniona wariancja: ", pca.explained_variance_ratio_)
# Wykres punktowy nowych zmiennych
import matplotlib.pyplot as plt
plt.scatter(principal_components[:,0], principal_components[:,1])
plt.title("Wizualizacja zbioru danych po redukcji wymiarowości PCA")
plt.xlabel("Główna składowa 1")
plt.ylabel("Główna składowa 2")
plt.show()
```

PCA from scratch

```
def PCA_by(X , n_components):
    X_meaned = X - np.mean(X , axis = 0) #Step-1


    cov_mat = np.cov(X_meaned , rowvar = False) #Step-2

    eigen_values , eigen_vectors = np.linalg.eigh(cov_mat) #Step-3
    sorted_index = np.argsort(eigen_values)[::-1] #Step-4
    sorted_eigenvalue = eigen_values[sorted_index]
    sorted_eigenvectors = eigen_vectors[:,sorted_index]

    eigenvector_subset = sorted_eigenvectors[:,0:n_components]     #Step-5

    X_reduced = np.dot(eigenvector_subset.transpose() , X_meaned.transpose() ).transpose()  #Step-6

    return X_reduced, eigen_vectors, eigen_values
```

PCA for mnist