Практическое задание
Обучите любую модель классификации на датасете IRIS до применения PCA и после него. Сравните качество классификации по отложенной выборке.
Напишите свою реализацию метода главных компонент посредством сингулярного разложения с использованием функции numpy.linalg.svd().

In [1]:

```
import numpy as np
from sklearn import metrics
from sklearn import datasets
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split

import catboost as ctb

import seaborn as sns
```

In [2]:

```
# Загрузим игрушечный датасет из sklearn
iris = datasets.load_iris()
X = iris.data
X.shape
```

Out[2]:

```
(150, 4)
```

In [3]:

```
dataset = datasets.load_iris()
X = dataset.data
y = dataset.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30)
```

In [4]:

```
X_test.shape
```

Out[4]:

```
(45, 4)
```

In [5]:

```
model = ctb.CatBoostClassifier()
model.fit(X_train, y_train)
print(model)
```

```
Learning rate set to 0.070535
0:      learn: 1.0158560        total: 153ms    remaining: 2m 32s
1:      learn: 0.9559209        total: 156ms    remaining: 1m 17s
2:      learn: 0.9024653        total: 158ms    remaining: 52.5s
3:      learn: 0.8441637        total: 160ms    remaining: 40s
4:      learn: 0.7967177        total: 161ms    remaining: 32.1s
5:      learn: 0.7570304        total: 164ms    remaining: 27.1s
6:      learn: 0.7127928        total: 166ms    remaining: 23.5s
7:      learn: 0.6747743        total: 169ms    remaining: 21s
8:      learn: 0.6409302        total: 171ms    remaining: 18.8s
9:      learn: 0.6076929        total: 176ms    remaining: 17.4s
10:     learn: 0.5716956        total: 178ms    remaining: 16s
11:     learn: 0.5488025        total: 180ms    remaining: 14.8s
12:     learn: 0.5236103        total: 182ms    remaining: 13.8s
13:     learn: 0.4952281        total: 184ms    remaining: 12.9s
14:     learn: 0.4756991        total: 186ms    remaining: 12.2s
15:     learn: 0.4520682        total: 188ms    remaining: 11.5s
16:     learn: 0.4333506        total: 190ms    remaining: 11s
17:     learn: 0.4156313        total: 196ms    remaining: 10.7s
```

In [6]:

```
expected_y  = y_test
predicted_y = model.predict(X_test)
```

In [7]:

```
print(metrics.classification_report(expected_y, predicted_y))
print(metrics.confusion_matrix(expected_y, predicted_y))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        12
           1       1.00      0.83      0.91        18
           2       0.83      1.00      0.91        15

    accuracy                           0.93        45
   macro avg       0.94      0.94      0.94        45
weighted avg       0.94      0.93      0.93        45

[[12  0  0]
 [ 0 15  3]
 [ 0  0 15]]
```

PCA

In [8]:

```
from sklearn.decomposition import PCA
import pandas as pd
```

In [9]:

```
df = pd.DataFrame(data= X, columns = ['sepal length','sepal width','petal length','petal wi
```

In [10]:

```
df.head()
```

Out[10]:

	sepal length	sepal width	petal length	petal width
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

In [13]:

```
df_y = pd.DataFrame(y, columns = ['y'])
```

In [18]:

```
df_y.sample(5)
```

Out[18]:

	y
83	1
27	0
87	1
42	0
147	2

In [ ]:

```

```

In [25]:

```
pca = PCA(n_components=2)
X_train_pca = pca.fit_transform(X_train)
```

In [26]:

```
df_pca = pd.DataFrame(data = X_train_pca, columns = ['component_one', 'component_two'])
```

In [27]:

```
df_pca.head()
```

Out[27]:

	component_one	component_two
0	2.464194	0.482074
1	2.630030	0.084013
2	0.979917	-0.007575
3	1.263209	-0.119363
4	2.252007	-0.166438

In [28]:

```
df_r = pd.concat([df_pca, df_y[['y']]], axis = 1)
```

In [29]:

```
df_r.head()
```

Out[29]:

	component_one	component_two	y
0	2.464194	0.482074	0
1	2.630030	0.084013	0
2	0.979917	-0.007575	0
3	1.263209	-0.119363	0
4	2.252007	-0.166438	0

In [30]:

```
df_r.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 3 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   component_one  105 non-null    float64
 1   component_two  105 non-null    float64
 2   y              150 non-null    int32  
dtypes: float64(2), int32(1)
memory usage: 3.1 KB
```

In [31]:

```
X_train_pca, X_test_pca, y_train_pca, y_test_pca = train_test_split(df_r[['component_one',
```

In [32]:

```
X_train_pca.shape
```

Out[32]:

```
(105, 2)
```

In [33]:

```
model_pca = ctb.CatBoostClassifier()
model_pca.fit(X_train_pca, y_train_pca)
print(model)
```

```
Learning rate set to 0.070535
0:      learn: 1.0475367        total: 1.78ms   remaining: 1.78s
1:      learn: 1.0085589        total: 3.5ms    remaining: 1.75s
2:      learn: 0.9665162        total: 5.33ms   remaining: 1.77s
3:      learn: 0.9299865        total: 7.14ms   remaining: 1.78s
4:      learn: 0.8970879        total: 8.83ms   remaining: 1.76s
5:      learn: 0.8668591        total: 10.6ms   remaining: 1.76s
6:      learn: 0.8375440        total: 12.5ms   remaining: 1.78s
7:      learn: 0.8130169        total: 14.4ms   remaining: 1.79s
8:      learn: 0.7900714        total: 16.2ms   remaining: 1.78s
9:      learn: 0.7696709        total: 18.1ms   remaining: 1.79s
10:     learn: 0.7512185        total: 20.1ms   remaining: 1.8s
11:     learn: 0.7323344        total: 22ms     remaining: 1.81s
12:     learn: 0.7177287        total: 23.8ms   remaining: 1.81s
13:     learn: 0.7018925        total: 25.7ms   remaining: 1.81s
14:     learn: 0.6870390        total: 27.7ms   remaining: 1.82s
15:     learn: 0.6747483        total: 29.5ms   remaining: 1.81s
16:     learn: 0.6621506        total: 31.4ms   remaining: 1.82s
17:     learn: 0.6494650        total: 32.5ms   remaining: 1.77s
```

In [34]:

```
expected_y_pca  = y_test_pca
predicted_y_pca = model_pca.predict(X_test_pca)
```

In [35]:

```
print(metrics.classification_report(expected_y_pca, predicted_y_pca))
print(metrics.confusion_matrix(expected_y_pca, predicted_y_pca))
```

```
              precision    recall  f1-score   support

           0       0.43      0.32      0.36        19
           1       0.39      0.60      0.47        15
           2       1.00      0.73      0.84        11

    accuracy                           0.51        45
   macro avg       0.61      0.55      0.56        45
weighted avg       0.56      0.51      0.52        45

[[ 6 13  0]
 [ 6  9  0]
 [ 2  1  8]]
```

In [36]:

```
print(metrics.classification_report(expected_y, predicted_y))
print(metrics.confusion_matrix(expected_y, predicted_y))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        12
           1       1.00      0.83      0.91        18
           2       0.83      1.00      0.91        15

    accuracy                           0.93        45
   macro avg       0.94      0.94      0.94        45
weighted avg       0.94      0.93      0.93        45

[[12  0  0]
 [ 0 15  3]
 [ 0  0 15]]
```

С PCA явно качество стало хуже, но этот датасет не нуждается в понижении размерности

numpy.linalg.svd()

In [ ]:

```

```

In [68]:

```
X = np.random.normal(size=[150,4])
```

In [69]:

```
df = pd.DataFrame(X)
```

In [70]:

```
df.head()
```

Out[70]:

	0	1	2	3
0	-0.396983	-1.156423	0.248381	-1.310929
1	-0.765302	0.020662	0.238294	-2.446115
2	-0.274813	-0.706302	0.360957	0.202848
3	-1.332770	-1.154290	0.287957	-0.181508
4	-1.667763	-0.504344	-1.199451	0.166317

In [71]:

```
P, D, Q = np.linalg.svd(df, full_matrices=False)
```

In [72]:

```
df_P = pd.DataFrame(P)
```

In [73]:

```
# X_a = np.matmul(np.matmul(P, np.diag(D)), Q)

X_a = P @ np.diag(D) @ Q
```

In [74]:

```
print(np.std(X), np.std(X_a), np.std(X - X_a))
```

```
0.993145275718599 0.9931452757185996 6.951341192625555e-16
```

In [ ]:

```

```