Principal Component Analysis (PCA) - анализ главных компонент

Principal Component Analysis (PCA) - анализ главных компонент; метод анализа данных основанный на преобразовании взаимозависимых (коррелированных) компонент вектора характеристик событий в набор независимых (некоррелированных) переменных, называемых главными компонентами. Основная цель метода главных компонент заключается в выделении существенных (ответственных за основные изменения) в рассматриваемой системы характеристик и снижении размерности соответствующего пространства характеристик.

Изображение выглядит как мяч, сфера, Красочность

Автоматически созданное описание Изображение выглядит как сфера, мяч, Красочность

Автоматически созданное описание или Изображение выглядит как сфера, мяч, небо, Красочность

Автоматически созданное описание

Рассмотрим случай пространства характеристик размерности . Пусть  вектор в -мерном пространстве характеристик, описывающий некоторое событие  в системе. Компоненты  представляют характеристики отдельного события, нижний индекс  обозначает номер компоненты вектора характеристик, верхний индекс -номер события. Будем считать, что данные центрированы:



и нормированы:



Если это не так, то необходимо применить преобразование данных:

Корреляционная -матрица  имеет вид:



недиагональные элементы матрицы 

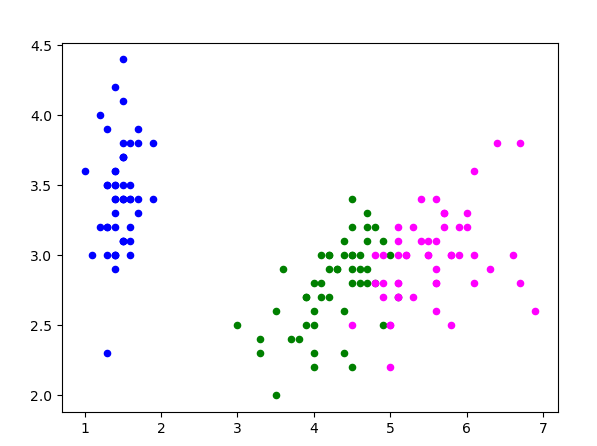
, 

определяет меру зависимости компонент вектора характеристик . Находим собственные значения  и собственные вектора  вариационной матрицы:  (заметим, что ). Будем считать, что собственные значения упорядочены в порядке убывания: . Если некоторые из собственных значений существенно меньше других: , то это означает, что среди  компонент  вектора характеристик  только  реально независимых. Поэтому при переходе к новым, независимым компонентам :

имеет смысл использовать только первые  из них. В этом случае размерность данных снижается, что может существенно упростить анализ состояния системы.

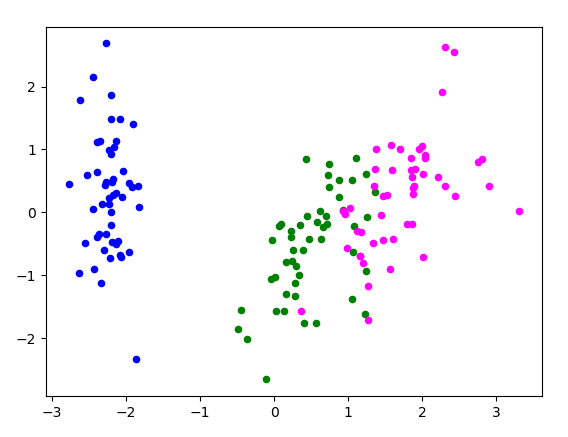
Практические примеры PCA

***Ирисы фишера:***



Исходные данные - 2 из 6 характеристик

Точность – 0.89



Редуцированные данные – 1 и 2 компоненты

Точнось – 0.83

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn import preprocessing

from sklearn.cluster import KMeans

from sklearn.decomposition import PCA

DIM\_N = 4

iris = pd.read\_csv('../iris.csv')

POINT\_N = iris.shape[0]

targ = np.array(iris)[:, DIM\_N]

targ = preprocessing.LabelEncoder().fit\_transform(targ)

data = np.array(iris)[:, 0:DIM\_N]

colors = ['blue', 'green', 'magenta', 'black']

for i in range(POINT\_N):

plt.scatter(data[i][2], data[i][1], c=colors[targ[i]], s=20)

plt.show()

kmeans = KMeans(n\_clusters=3, random\_state=2).fit(data)

res = kmeans.labels\_ == targ

print('\n', res.sum()/len(res))

# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* PCA \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

scaler = preprocessing.StandardScaler().fit(data)

for i in range(POINT\_N):

data[i] = (data[i]-scaler.mean\_)/scaler.scale\_

data\_r = PCA(n\_components=2).fit\_transform(data)

for i in range(POINT\_N):

plt.scatter(data\_r[i][0], data\_r[i][1], c=colors[targ[i]], s=20)

plt.show()

kmeans\_r = KMeans(n\_clusters=3, random\_state=0).fit(data\_r)

res = kmeans\_r.labels\_ == targ

print('\n', res.sum()/len(res))

***Качество вина:***

Вино имеет 11 характеристик и его качество оценивается по шкале 1-10.

Изображение выглядит как снимок экрана, Прямоугольник, диаграмма, текст

Контент, сгенерированный ИИ, может содержать ошибки.

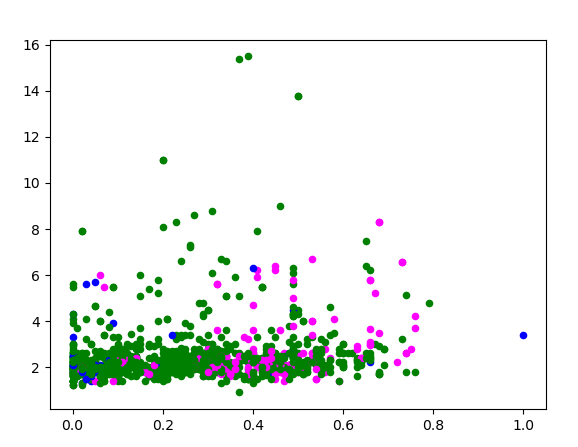
Распределение оценок экспертов

Изменим шкалу оценок:

Изображение выглядит как текст, снимок экрана, Прямоугольник, диаграмма

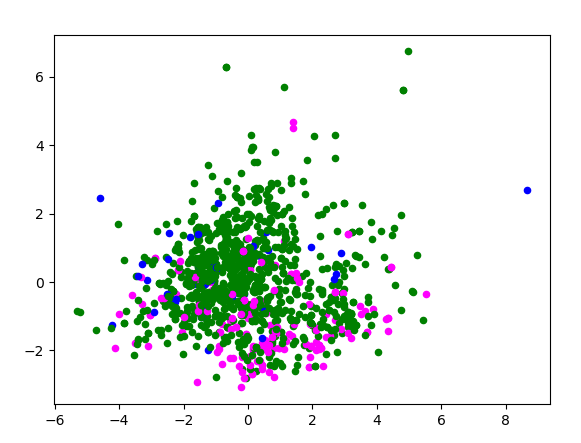
Контент, сгенерированный ИИ, может содержать ошибки.

Шкала оценок (0, 1, 2)



Данные на плоскости 2-й и 3-й характеристик (в качестве примера)

Результат классификации методом ближайших соседей (5 соседей) – 0,84



Редуцированные данные – 1 и 2 компоненты

Результат классификации редуцированных данных – 0,82

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn import preprocessing

from sklearn.cluster import KMeans

from sklearn.decomposition import PCA

from sklearn.model\_selection import train\_test\_split

from sklearn.neighbors import KNeighborsClassifier

DIM\_N = 11

wine = pd.read\_csv('../WineQT.csv')

POINT\_N = wine.shape[0]

targ = np.array(wine)[:, DIM\_N].astype(int)

data = np.array(wine)[:, 0:DIM\_N]

plt.hist(targ, color = 'blue', bins = 5)

plt.show()

for i in range(POINT\_N):

if targ[i]<5:

targ[i] = 0

else:

if targ[i]<7:

targ[i] = 1

else:

targ[i] = 2

plt.hist(targ, color = 'blue', bins = 3)

plt.show()

colors = ['blue', 'green', 'magenta', 'cyan', 'brown', 'orange', 'black']

for i in range(POINT\_N):

plt.scatter(data[i][2], data[i][3], c=colors[targ[i]], s=20)

plt.show()

scaler = preprocessing.StandardScaler().fit(data)

for i in range(POINT\_N):

data[i] = (data[i]-scaler.mean\_)/scaler.scale\_

data\_train, data\_test, targ\_train, targ\_test = train\_test\_split(data, targ, test\_size=0.5, random\_state=0)

neigh = KNeighborsClassifier(n\_neighbors=5)

neigh.fit(data\_train, targ\_train)

res\_test = neigh.predict(data\_test)

res = (res\_test==targ\_test)

print('\n', res.sum()/len(res))

# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* PCA \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

data\_r = PCA(n\_components=2).fit\_transform(data)

for i in range(POINT\_N):

plt.scatter(data\_r[i][0], data\_r[i][1], c=colors[targ[i]%6], s=20)

plt.show()

data\_r\_train, data\_r\_test, targ\_train, targ\_test = train\_test\_split(data\_r, targ, test\_size=0.5, random\_state=0)

neigh = KNeighborsClassifier(n\_neighbors=5)

neigh.fit(data\_r\_train, targ\_train)

res\_r\_test = neigh.predict(data\_r\_test)

res = (res\_r\_test==targ\_test)

print('\n', res.sum()/len(res))