Машинное обучение

НИЯУ МИФИ, КАФЕДРА ФИНАНСОВОГО МОНИТОРИНГА

КУРС ЛЕКЦИЙ

В.Ю. РАДЫГИН. Д.Ю. КУПРИЯНОВ

ЛЕКЦИЯ 6

Библиотеки

В данной лекции будут рассмотрены примеры с использованием следующих библиотек:

- NumPy – https://numpy.org/
- Pandas – https://pandas.pydata.org/
- scikit-learn – https://scikit-learn.org
- Matplotlib – https://matplotlib.org/

Часть 1

МЕТОД ГЛАВНЫХ КОМПОНЕНТ

Избыточные данные

Одной из проблем интеллектуального анализа данных является недостаток информации для решения задачи. Но не менее проблематичен и избыток данных. Слишком большое количество наблюдаемых параметров может приводить к высокой вычислительной сложности алгоритмов решения поставленной задачи. Поэтому, зачастую приходится решать вопрос уменьшения числа переменных, описывающих наблюдения (уменьшение размерности данных). При этом во многих случаях нельзя просто отбросить часть параметров.

Одним из наиболее популярных методов уменьшения размерности данных является метод главных компонент.

Пример 1

Рассмотрим информацию о посещаемости студентами-магистрами потока 2018 года набора дисциплины СТБД и оценки, заработанные ими после 7-ми занятий.

days	points
57,14	38
100	73
100	99,5
42,86	73
71,43	24
28,57	23
14,29	0
71,43	0
71,43	75
71,43	66,5
57,14	12,5
57,14	0
57,14	7,5
85,71	97,5
85,71	69,5
42,86	64,5
85,71	93
71,43	20
85,71	75
14,29	0
57,14	0
28,57	0
57,14	7,5
14,29	0
85,71	98,5
28,57	0
100	105
71,43	32
100	105
85,71	27,5
0	0
100	99,5
85,71	30,5
85,71	104
85,71	97
71,43	102
85,71	102
28,57	5

Построим точечную диаграмму

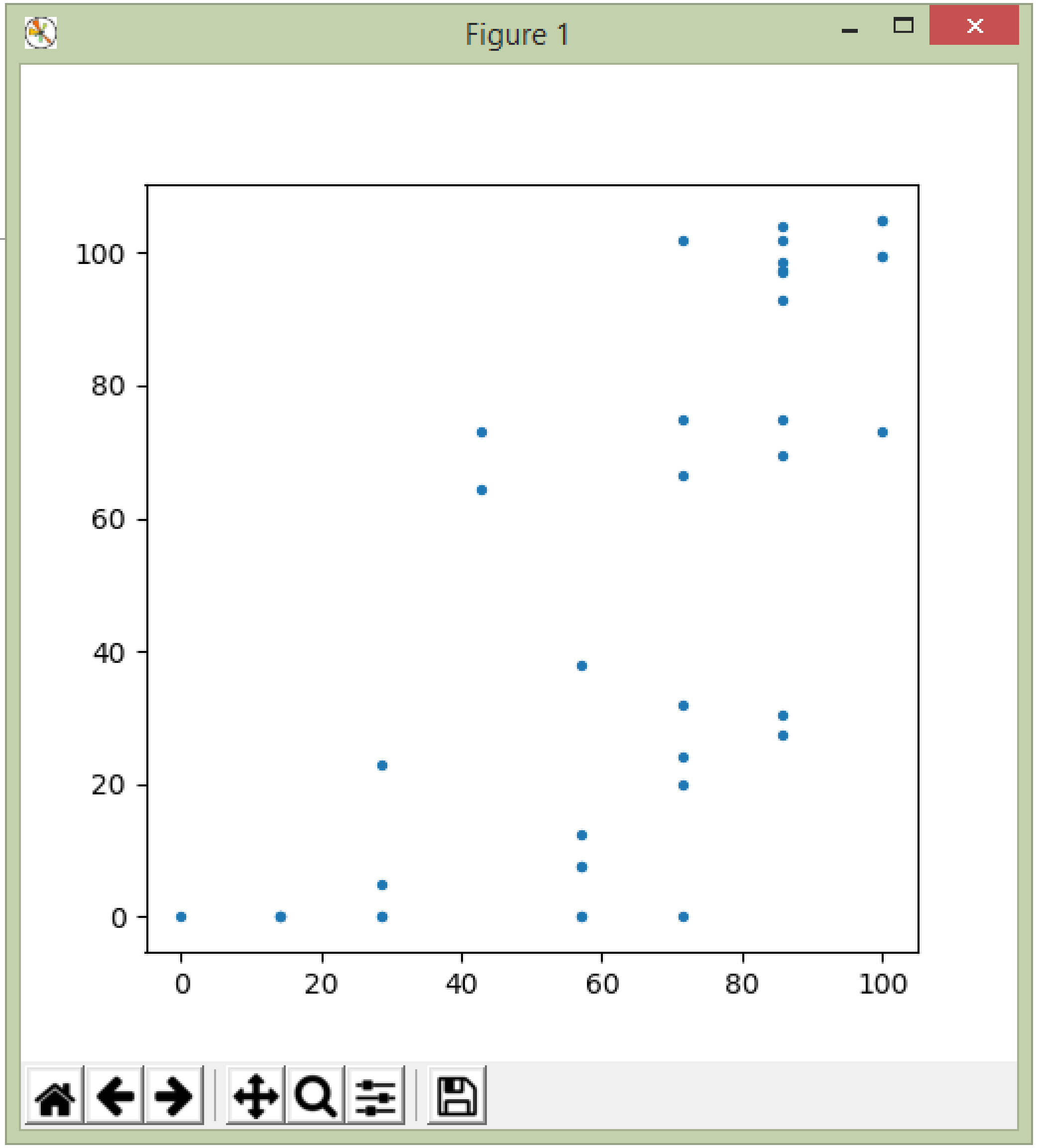
example6runner.py - E:\Works\Victor\Students\STDB\Lection6\example6runner.py (3.7.2)

File Edit Format Run Options Window Help

```python
import numpy as np
import pandas as pd
import math

import matplotlib.pyplot as plt
from matplotlib.patches import Ellipse
import matplotlib.transforms as transforms

pd.set_option('display.max_columns', 2000)
pd.set_option('display.width', 2000)

table = pd.read_excel("data.xlsx")

fig, ax = plt.subplots(figsize=(5, 5))
ax.scatter(table['days'], table['points'], s = 2)

plt.show()
```

Ln: 19 Col: 10

Точечная диаграмма



Предикативный или доверительный эллипс

Из построенной точечной диаграммы видно наличие некоторой корреляции между посещаемостью и оценкой студентов.

Для лучшего понимания наличия корреляции мы построим предикативный эллипс – фигуру, в которую должно попасть определённое число новых наблюдений, если считать, что исходный набор точек бы подвержен двумерному нормальному распределению.

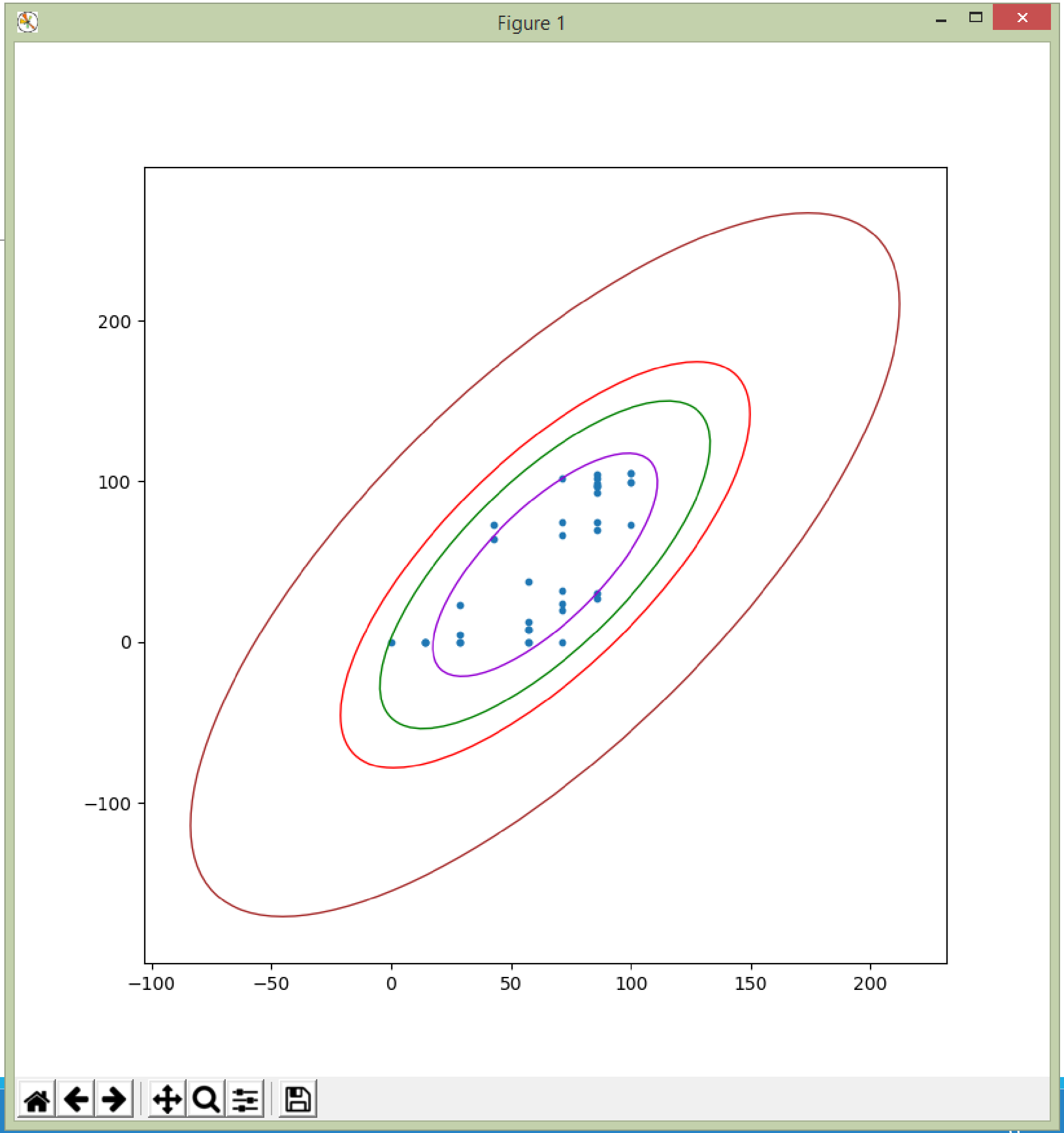
Например, 80%-предикативный эллипс – это фигура, в которую должно попасть 80% новых наблюдений.

Наиболее часто используют 95% предикативный эллипс.

К сожалению, в python это не так уж и просто! Но пока не будем об этом задумываться и рассмотрим только результат.

Результат

Фиолетовый цвет – 75%;
Зелёный цвет – 95%;
Красный цвет – 99%;
Коричневый цвет – 99,9999%.



О чём говорит эллипс

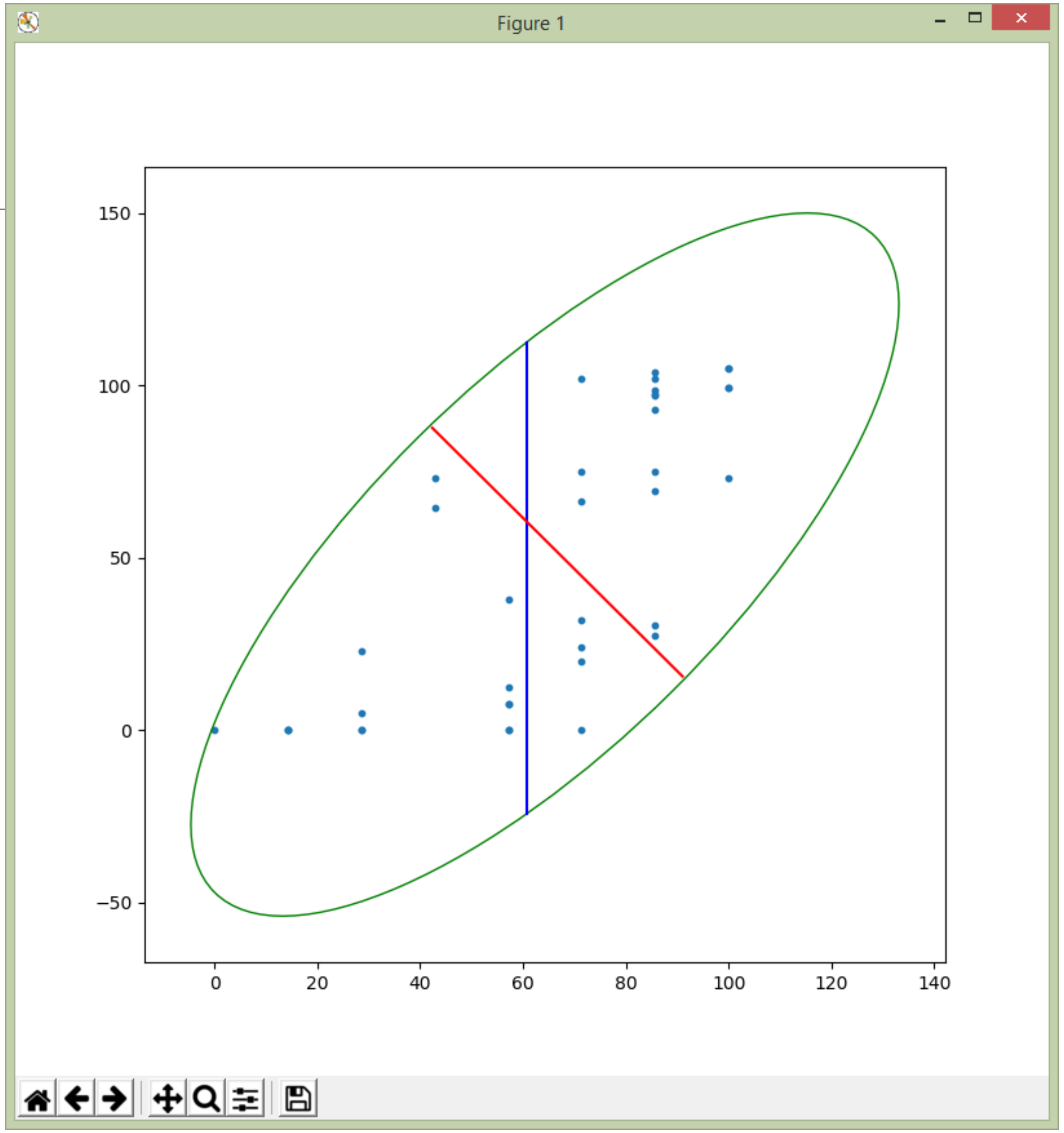
Так как у нас получился именно эллипс, а не круг, то можно говорить о наличии корреляции между посещаемостью и оценкой. Угол наклона эллипса меньше 90 градусов. Следовательно корреляция прямая. Чем более вытянут эллипс, тем ближе коэффициент корреляции к единице.

Кроме того, больший радиус предиктивного эллипса показывает направление, вдоль которого наблюдается наибольшая дисперсия наблюдений. Или, говоря другим языком, направление, вдоль которого в нашем наборе данных наибольшая изменчивость.

Предиктивный эллипс

Если провести в точке 60 горизонтальной координатной оси перпендикулярную прямую, то она пересечёт эллипс в двух точках, образуя отрезок (синий цвет). Данный отрезок будет показывать максимальный разброс значений оценок студента при 60% посещаемости.

При полной корреляции величине 60 посещаемости соответствует точка 60 оценки студента. Очевидно, что через данную точку можно провести более короткий отрезок (чем синий), вершины которого лежат по разные стороны эллипса. Достаточно провести отрезок, параллельный меньшему радиусу эллипса (красный).



Более компактная система координат

Так как в нашем примере угол наклона эллипса близок к 45 градусам (на картинке он действительно 45 градусов, но шкалы осей OX и OY различны, поэтому видимый наклон не совпадает с реальным), то мы можем сделать переход к новой системе координат. Одна ось будет получаться сложением значения посещаемости и значения оценки:

PC1 = посещаемость + оценка = (X + Y) / $\sqrt{2}$

Очевидно, что вторая ось, перпендикулярная первой будет получаться вычитанием координат:

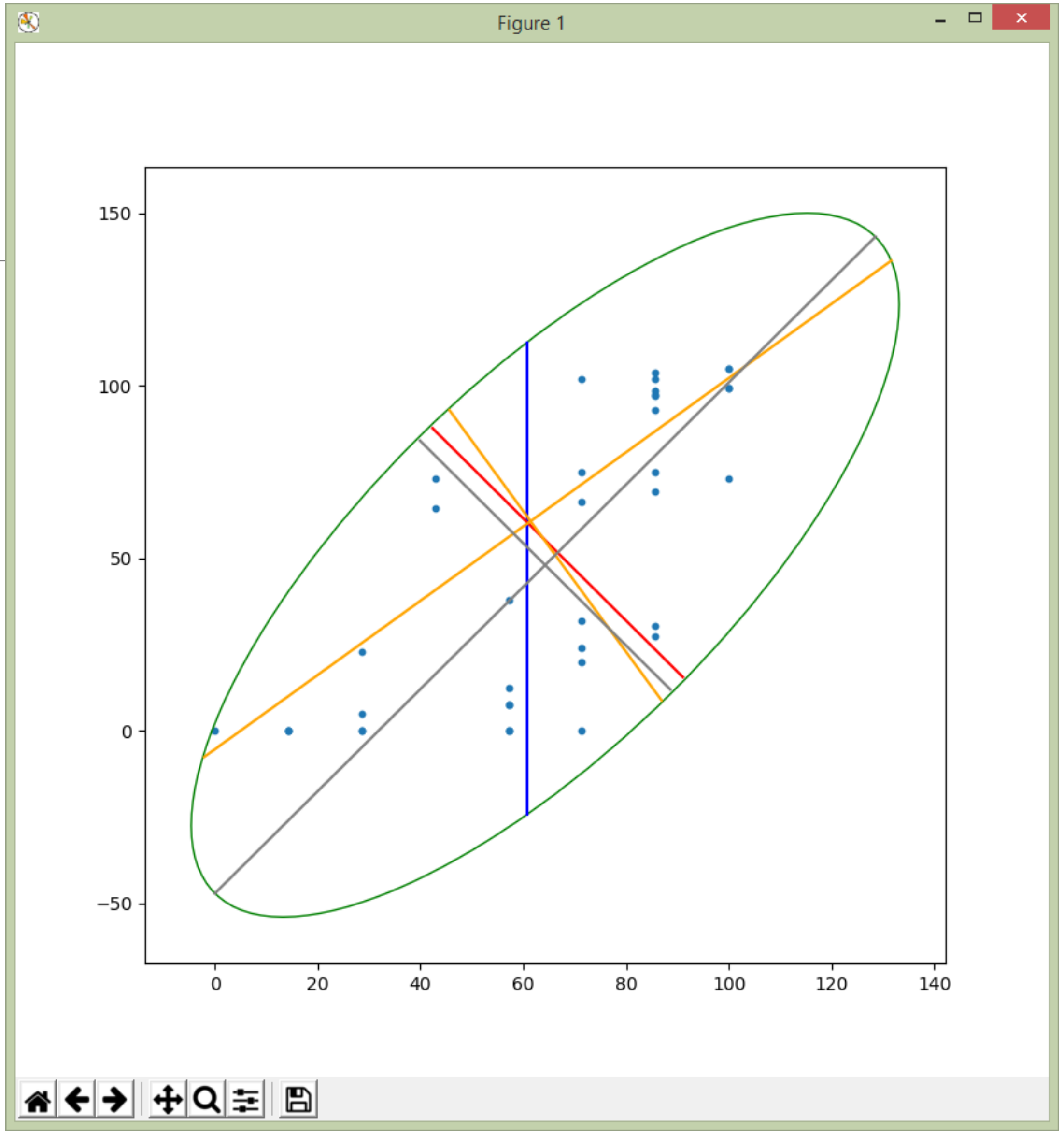
PC2 = оценка – посещаемость = (Y - X) / $\sqrt{2}$ = (-X + Y) / $\sqrt{2}$

Новые координаты

Новая система координат (оранжевый цвет) более эффективна, так как разброс координаты PC2 внутри эллипса меньше, чем разброс координаты Y.

Тем не менее, видно, что, если бы наклон осей координат совпадал бы с реальной картиной полностью (серый цвет), то разброс был бы ещё меньше.

Но как узнать правила перехода к наиболее эффективной системе координат?



Матрица поворота

В общем случае правила перехода к новой системе координат можно описать следующим образом:

PC1 = A11 * X + A12 * Y

PC2 = A21 * X + A22 * Y

Матрица $\begin{pmatrix} A11 & A12 \\ A21 & A22 \end{pmatrix}$ называется матрицей поворота.

Матрица ковариаций

Настоящая матрица ковариации для нашей задачи будет следующей:

[[1.02702703 0.76063983]

[0.76063983 1.02702703]]

С её помощью можно оценить значимость полученных компонент новой системы координат. Для этого нужно определить собственные значения и собственные вектора данной матрицы.

Собственные значения и вектора

Составим для матрицы ковариации матричное уравнение:

Ax = λx, где x – собственные вектора, а λ – собственные значения.

Если упростить данное выражение и перенести всё в одну сторону, то получим:

A - λE = 0

Собственные вектора позволяют сформировать матрицу поворота. А собственные значения позволяют оценить степень влияния каждой из новых координат на изменчивость исходной задачи.

Стандартизация данных

Предварительная нормировка нужна для обоснованного выбора метрики, в которой будет вычисляться наилучшая аппроксимация данных, или будут искаться направления наибольшего разброса (что эквивалентно). Обычно для нормировки используют стандартизацию.

Помимо масштабирования, стандартизация также центрирует наблюдения относительно математического ожидания.

Стандартизация данных

Стандартизация данных – это такое биективное отображение данных из пространства действенных чисел в пространство действительных чисел, при котором данные оказываются распределёнными вокруг 0 со стандартным отклонением 1:

$$x' = \frac{x - M_x}{\sigma_x},$$

где M_x – математическое ожидание (среднее арифметическое) величины x, а σ_x – стандартное отклонение величины x.

Стандартизация данных

```
from sklearn import preprocessing
scaler_std = preprocessing.StandardScaler()
x = scaler_std.fit_transform(table[['days']])
table['days'] = x[0:]
x = scaler_std.fit_transform(table[['points']])
table['points'] = x[0:]
```

Метод главных компонент

```
from sklearn.decomposition import PCA
pca = PCA(n_components = 2)

table_train = pca.fit_transform(table)
```

В table_train будут размещены данные после преобразования. Для получения остальной информации будем использовать дополнительные атрибуты и методы. Стоит отметить, что теперь нам доступен метод transform, выполняющий данное преобразование для новых данных.

Матрица ковариации

example6runner.py - C:\Users\User\Desktop\Students\STDBPY\Lection6\example6runner.py (3.7.2)

File Edit Format Run Options Window Help

```python
cov = pca.get_covariance()
print(cov)
```

Ln: 40 Col: 10

```
[[1.02702703 0.76063983]
 [0.76063983 1.02702703]]
>>>
```

Матрица преобразования

example6runner.py - E:\Works\Victor\Students\STDB\Lection6\example6runner.py (3.7.2)

File Edit Format Run Options Window Help

```
[en, ev] = np.linalg.eig(cov)
print(ev)
```

Ln: 47 Col: 10

```
[[ 0.70710678 -0.70710678]
 [ 0.70710678  0.70710678]]
>>>
```

Подумайте, как, используя эту матрицу, перейти от (x, y) к (PC1, PC2) и наоборот?

Матрица преобразования

Получить матрицу преобразования можно и другим способом. Метод components_ возвращает координаты нового базиса. Транспонировав данную матрицу можно получить матрицу поворота.

example6runner.py - C:\Users\User\Desktop\Students\STDBPY\Lection6\example6runner.py (3.7.2)

File Edit Format Run Options Window Help

```
comp = pca.components_
explained_variance = pca.explained_variance_
print(explained_variance)
print(comp)
print(comp.T)
```

Ln: 63 Col: 0

```
[1.78766686 0.2663872 ]
[[-0.70710678 -0.70710678]
 [ 0.70710678 -0.70710678]]
[[-0.70710678  0.70710678]
 [-0.70710678 -0.70710678]]
>>>
```

Почему матрицы не совпали?

На слайде 23 и 24 получились разные матрицы поворота. Почему? Разница только в знаке векторов. Но с точки зрения информативности нет никакой разницы в какой сторону направлен базис. Поэтому, можно считать, что матрицы эквивалентны по смыслу.

Не забывайте, что для получения правильной матрицы поворота для метода главных компонент собственные вектора должны быть отсортированы в порядке следования собственных значений!

В нашем примере так и получилось. 1-е собственное значение 1,788 > 2-го – 0,266. Но в общем случае это может быть не так и потребуется переставить столбцы матрицы ev.

Матрица преобразования

Для перехода от вектора a = (x, y) к вектору a’ = (PC1, PC2) с помощью матрицы преобразования A нужно вычислить:

$a' = a \cdot A$

Так как для матрицы преобразования верно, что $A^{-1} = A^T$, то

$a = a' \cdot A^T$

Оценка информативности координат

example6runner.py - E:\Works\Victor\Students\STDB\Lection6\example6runner.py (3.7.2)

File Edit Format Run Options Window Help

```
print(en)
en_r = en / en.sum()
print(en_r)

explained_variance = pca.explained_variance_
explained_variance_ratio = pca.explained_variance_ratio_
print(explained_variance)
print(explained_variance_ratio)
```

Ln: 54 Col: 0

Оценка информативности координат

```
[1.78766686 0.2663872 ]
[0.8703115 0.1296885]
[1.78766686 0.2663872 ]
[0.8703115 0.1296885]
>>>
```

Что это значит? Это значит, что 87% изменчивости данных – это координата PC1, а 13% – координата PC2. Таким образом, перейдя от двух координат x, y к одной – PC1 мы потеряем только 13% информации.

Пример 2. Ирисы Фишера на википедии

Ирисы Фишера

Длина чашелистика	Ширина чашелистика	Длина лепестка	Ширина лепестка	Вид ириса
5.1	3.5	1.4	0.2	*setosa*
4.9	3.0	1.4	0.2	*setosa*
4.7	3.2	1.3	0.2	*setosa*
4.6	3.1	1.5	0.2	*setosa*
5.0	3.6	1.4	0.2	*setosa*
5.4	3.9	1.7	0.4	*setosa*
4.6	3.4	1.4	0.3	*setosa*
5.0	3.4	1.5	0.2	*setosa*
4.4	2.9	1.4	0.2	*setosa*
4.9	3.1	1.5	0.1	*setosa*
5.4	3.7	1.5	0.2	*setosa*
4.8	3.4	1.6	0.2	*setosa*
4.8	3.0	1.4	0.1	*setosa*

Iris setosa

Iris virginica

Нахождение главных компонент

example6.py - E:\Works\Victor\Students\STDB\Lection6\example6.py (3.7.2)

File Edit Format Run Options Window Help

```python
table = pd.read_excel("irises.xlsx")

from sklearn import preprocessing
scaler_std = preprocessing.StandardScaler()
for name in ['sepal_length', 'sepal_width', 'petal_length', 'petal_width']:
    x = scaler_std.fit_transform(table[[name]])
    table[name] = x[0:]

from sklearn.decomposition import PCA
pca = PCA(n_components = 4)
table_train = pca.fit_transform(table[['sepal_length', 'sepal_width',
                                       'petal_length', 'petal_width']])

cov = pca.get_covariance()
[en, ev] = np.linalg.eig(cov)
print(ev)

explained_variance_ratio = pca.explained_variance_ratio_
print(explained_variance_ratio)
```

Ln: 168 Col: 31

Результат

```
[[ 0.52106591 -0.37741762 -0.71956635  0.26128628]
 [-0.26934744 -0.92329566  0.24438178 -0.12350962]
 [ 0.5804131  -0.02449161  0.14212637 -0.80144925]
 [ 0.56485654 -0.06694199  0.63427274  0.52359713]]
[0.72962445 0.22850762 0.03668922 0.00517871]
>>>
```

Что это значит? Это значит, что 96% изменчивости данных – это координаты PC1 и PC2, а 4% – координаты PC3, PC4. Таким образом, перейдя от 4 координат к 2 мы потеряем только 4% информации.

Часть 2

МЕТОД ОПОРНЫХ ВЕКТОРОВ

Метод опорных векторов

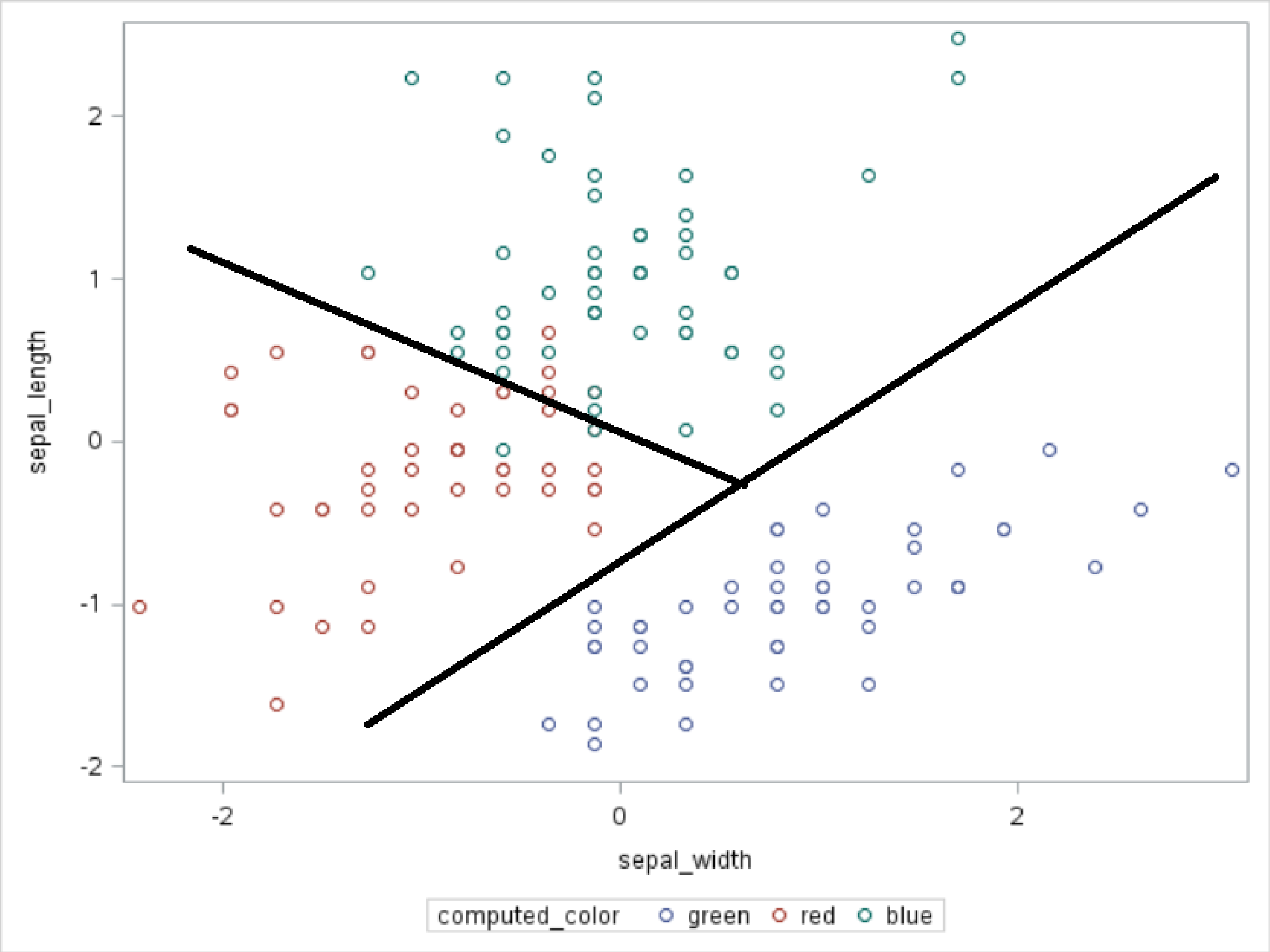
Метод опорных векторов (Support vector machines – SVM) – это метод классификации данных, использующий альтернативный по отношению к деревьям решений подход.

В основе идеи лежит два основных шага:

1. Переход в пространство большей размерности на основе некого правила перехода (ядра).
2. Разделение данных в новом пространстве при помощи проведения гиперплоскостей.

Метод SVM требует обучения. Фактически, сначала мы находим, как разделить уже классифицированные данные, а потом найденные гиперплоскости используем для разделения новых, ещё не классифицированных данных.

Пример с точками



Пример с точками

В рассмотренном примере множество точек двумерного пространства соответствует трём классам. Если провести прямую, то можно отделить один класс от двух других. Два оставшихся класса, в свою очередь, могут быть разделены ещё одной прямой.

Определив, прямые мы сможем построить классификатор. Если точка попадает в пространство ниже прямой, то она принадлежит одному классу. Если же точка попадает в пространство выше прямой, то другому.

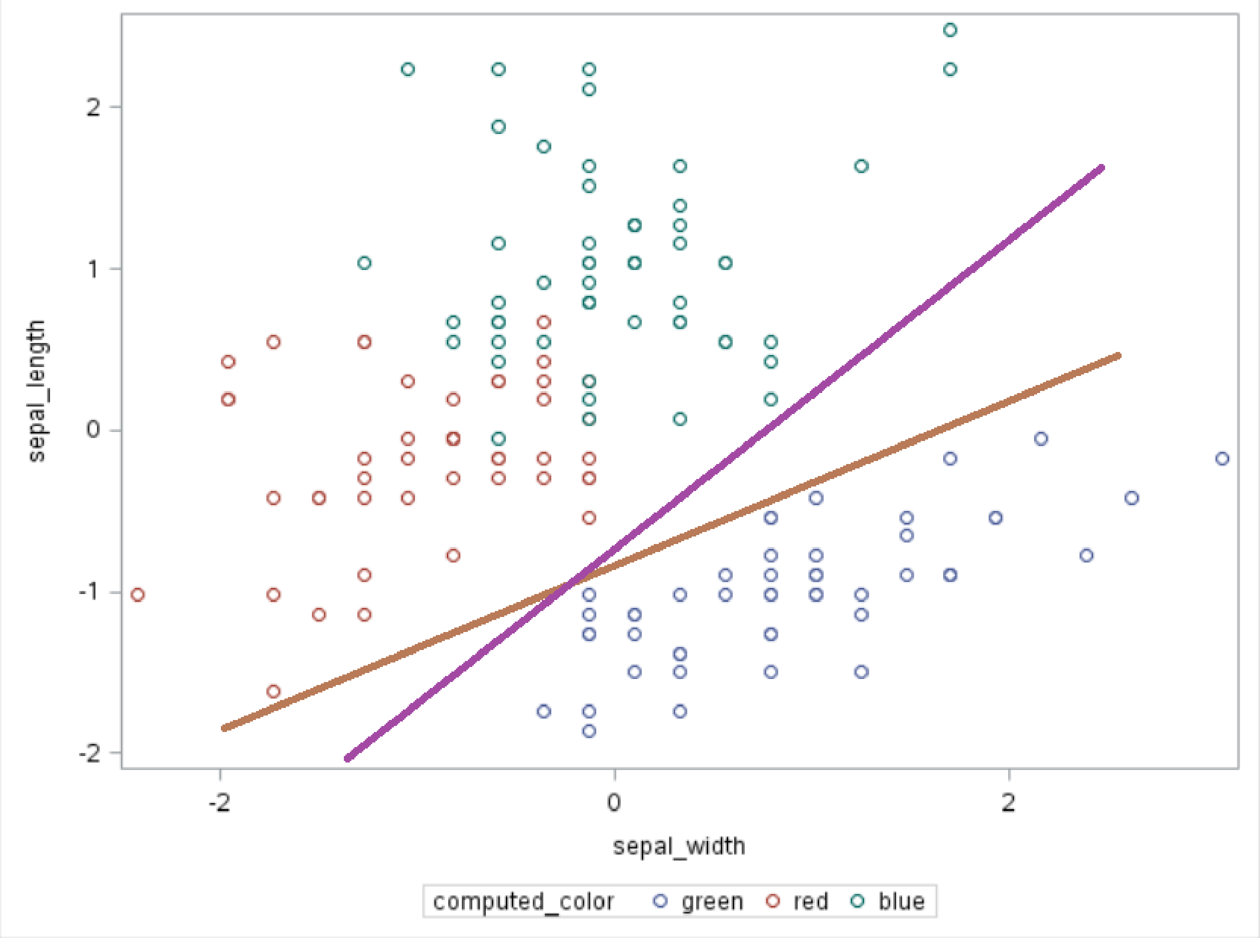
Собственно говоря, именно в этом и заключается идея метода опорных векторов. Только в пространстве большей размерности вместо точек возникают векторы, а вместо прямых гиперплоскости.

Отступ (Margin)

Разделяя два набора сущностей мы можем выбрать несколько вариантов гиперплоскости. Какая из них будет лучше? В методе SVM выбирают ту гиперплоскость, которая максимально равноудалена от всех точек двух разделяемых множеств. То есть, до ближайшей из точек каждого из разделяемых множеств должно быть одинаковое расстояние и оно должно быть максимально возможным среди всех вариантов.

Данной расстояние называется отступом. А точки двух множеств, от которых оно отсчитывалось – опорными векторами.

Таким образом, на самом деле у нас есть разделяющая полоса, а не прямая.



Как найти разделяющую гиперплоскость?

Пусть задана обучающая выборка $X: (\overrightarrow{x_1}, k_1), (\overrightarrow{x_2}, k_2), \cdots (\overrightarrow{x_n}, k_n)$, где $x_i \in \mathbb{R}^m$, а $k_i \in \{1, -1\}$.

Тогда разделяющая гиперплоскость будет задаваться как некоторая функция $\acute{F}(x) = \langle\vec{w}, \vec{x}\rangle + b$ где $\langle\ \rangle$ – скалярное произведение, b – некоторая константная величина, а $\vec{w}$ - вектор нормали к разделяющей гиперплоскости.

Функция классификации получается добавлением функции сигнума: $\mathrm{F}(x) = sign(\langle\vec{w}, \vec{x}\rangle + b)$.

Задача нахождения минимума

Задача может быть сведена к задачи оптимизации (квадратичного программирования) нахождения максимума:

$$\begin{cases} \dfrac{1}{\|\vec{\boldsymbol{w}}\|} \to max \\ k_i(\langle \vec{w}, \vec{x_i} \rangle + b) \geq 1 \end{cases}$$

Или эквивалентной ей задаче нахождения минимума:

$$\begin{cases} \|\vec{\boldsymbol{w}}\|^2 \to min \\ k_i(\langle \vec{w}, \vec{x_i} \rangle + b) \geq 1 \end{cases}$$

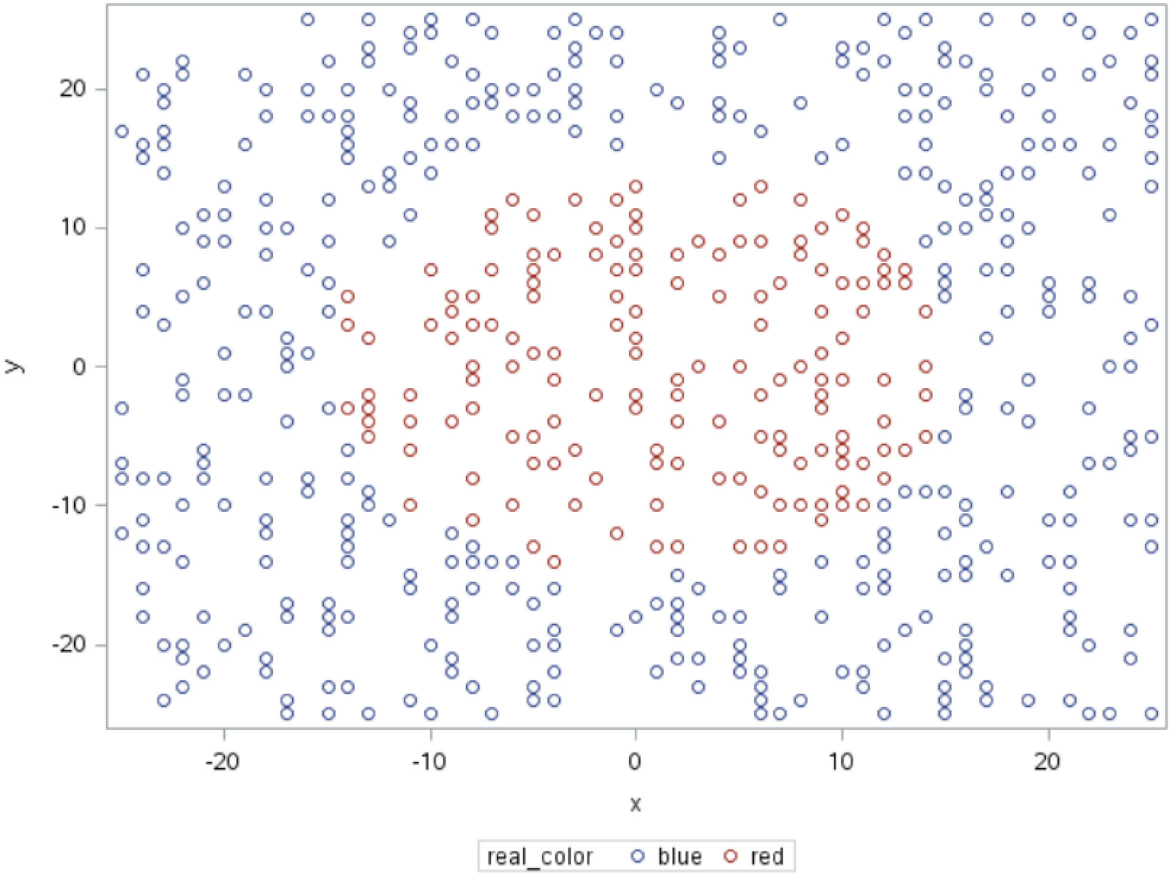
Обе задачи имеют решение, которое может быть получено с помощью множителей Лагранжа или других современных методов.

Линейно-разделимая задача

Если точки могут быть разделены на классы гиперплоскостями в текущем пространстве, то данная задача классификации является линейно-разделимой.

Иначе задача является линейно-неразделимой.

Линейно-неразделимая задача



Пространство большей размерности

Если задача линейно-неразделима в текущем пространстве, то это не значит, что её нельзя сделать линейно-разделимой в пространстве большей размерности.

В общем случае, если выборка непротиворечива, то всегда найдётся пространство большей размерности в котором она будет линейно разделимой.

Метод опорных вектор хорошо представляется при помощи следующей ассоциации. Пусть у нас есть разноцветные шары, лежащие на столе. Пока они лежат в плоскости стола, мы не можем их разделить палкой на две группы, так как они частично перемешаны.

Подбросим шары вверх, причём возможно так, что не все из них окажутся на одинаковой высоте. Теперь шары в трёхмерном пространстве и появилась вероятность того, что у нас получится отделить шары одного цвета от шаров другого при помощи листа бумаги.

Визуальный пример

Для понимания причин перехода в методе SVM в пространство большей размерности рассмотрим видео пример, доступный на сайте youtube.com:

https://www.youtube.com/watch?v=3liCbRZPrZA&feature=youtu.be

Ядро

Пусть мы ввели новое пространство H большей размерности, чем исходное пространство X и со скалярным произведением, определив переход:

$$\psi: X \to H$$

Для того, чтобы в новом пространстве мы могли действовать алгоритмически также как и в исходном необходимо, чтобы существовала некая функция (ядро):

$$K: X \times X \to \mathbb{R},$$

такая, что:

$$K(x, x') = \langle \psi(x), \psi(x') \rangle.$$

В этом случае функция классификации примет вид: $\mathrm{F}(x) = sign(\langle \vec{w}, \psi(\vec{x}) \rangle + b)$.

В общем случае ядром классификации может выступать любая положительно определённая функция двух переменных.

Основные типы ядер

- Линейное (без перехода к пространству большей размерности)
- Полиномиальное
- Радиальная базисная функция
- Сигмоид

Полиномиальное ядро

Функция K для полиномиального ядра определяется следующим образом:

$K(x_1, x_2) = (\gamma\langle x_1, x_2\rangle + r)^d$

γ указывается в scikit-learn SVM при помощи параметра gamma, r – при помощи параметра coef0, d – при помощи параметра degree.

Пример работы полиномиального ядра был показан в видео из слайда визуальный пример.

Радиальная базисная функция

Функция K для ядра на основе радиальной базисной функции (RBF) определяется следующим образом:

$$K(x_1, x_2) = e^{-\gamma \|x_1 - x_2\|^2}, \text{где } \gamma > 0.$$

В scikit-learn есть два специальных значения γ: 'auto' и 'scale'.

γ_{auto} = 1/N; γ_{scale} = 1/(N*D),

где N – число параметров, а D – дисперсия X.

Сигмоид

Функция K для ядра на основе сигмоида определяется следующим образом:

$K(x_1, x_2) = \mathrm{th}(\gamma <x_1, x_2> + r)$.

Вес класса

Важным параметром настройки является параметр С – class_weight. Это параметр позволяет управлять тем, что более значимо: принадлежность к классу или уникальность отдельного наблюдения.

Данный параметр начинает играть большую роль при решении несбалансированных задач. Значение параметра $c > 0$.

SVM в Scikit-learn

В библиотеке Scikit-learn предусмотрен целый набор методов для алгоритма SVM. Мы рассмотрим один из вариантов работы с ним. Нам понадобятся:

Метод SVC модуля svm (svn.SVC()).

Метод fit.

Метод predict.

Метод score.

Пример 1. Ирисы Фишера

Ирисы Фишера

Длина чашелистика	Ширина чашелистика	Длина лепестка	Ширина лепестка	Вид ириса
5.1	3.5	1.4	0.2	*setosa*
4.9	3.0	1.4	0.2	*setosa*
4.7	3.2	1.3	0.2	*setosa*
4.6	3.1	1.5	0.2	*setosa*
5.0	3.6	1.4	0.2	*setosa*
5.4	3.9	1.7	0.4	*setosa*
4.6	3.4	1.4	0.3	*setosa*
5.0	3.4	1.5	0.2	*setosa*
4.4	2.9	1.4	0.2	*setosa*
4.9	3.1	1.5	0.1	*setosa*
5.4	3.7	1.5	0.2	*setosa*
4.8	3.4	1.6	0.2	*setosa*
4.8	3.0	1.4	0.1	*setosa*

Iris setosa

Iris virginica

Пример 1

В качестве первого примера возьмём без обработки данные «Ирисов Фишера», поделив их на обучающую и тестовую выборки в соотношении 6 к 4.

Пример 1. Подготовка данных

example2-1runner.py - E:\Works\Victor\Students\STDB\Term2\example2-1runner.py (3.7.2)

File Edit Format Run Options Window Help

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split

table = pd.read_excel("irises.xlsx")
train_table, test_table = train_test_split(table, test_size = 0.4,
                                           random_state = 22222)
test_table = test_table.reset_index()
train_table = train_table.reset_index()
```

Ln: 5 Col: 23

Пример 1. Подготовка данных

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split

table = pd.read_excel("irises.xlsx")
train_table, test_table = train_test_split(table, test_size = 0.4, random_state = 22222)
test_table = test_table.reset_index()
train_table = train_table.reset_index()
```

Пример 1. Обучение

example2-1runner.py - E:\Works\Victor\Students\STDB\Term2\example2-1runner.py (3.7.2)

File Edit Format Run Options Window Help

```
from sklearn import svm

for krl in ['linear', 'poly', 'rbf', 'sigmoid']:
    clf = svm.SVC(kernel = krl)
    clf.fit(train_table[['sepal_length', 'sepal_width', 'petal_length',
                         'petal_width']], train_table['class_label'])
```

Ln: 10 Col: 0

Пример 1. Обучение

```
from sklearn import svm


for krl in ['linear', 'poly', 'rbf', 'sigmoid']:
    clf = svm.SVC(kernel = krl)
    clf.fit(train_table[['sepal_length', 'sepal_width', 'petal_length',
                         'petal_width']], train_table['class_label'])
```

Пример 1. Тестирование и первая метрика

example2-1runner.py - E:\Works\Victor\Students\STDB\Term2\example2-1runner.py (3.7.2)

File Edit Format Run Options Window Help

```
    res = clf.predict(test_table[['sepal_length', 'sepal_width', 'petal_length',
                                   'petal_width']])
    test_table2 = test_table.copy()
    test_table2['new_class'] = pd.Series(res)
    good = test_table2[test_table2['new_class'] == test_table2['class_label']]
    good_size = good['sepal_length'].count()
    all_size = test_table2['sepal_length'].count()
    print(krl)
    if krl == 'linear':
        print(clf.coef_)
    print(good_size/all_size * 100)
    print(clf.score(test_table[['sepal_length', 'sepal_width', 'petal_length',
                                'petal_width']], test_table[['class_label']]))
```

Ln: 30 Col: 77

Пример 1. Тестирование и первая метрика

```
res = clf.predict(test_table[['sepal_length',
'sepal_width', 'petal_length',
                    'petal_width']])
  test_table2 = test_table.copy()
  test_table2['new_class'] = pd.Series(res)
  good = test_table2[test_table2['new_class'] ==
                          test_table2['class_label']]
  good_size = good['sepal_length'].count()
  all_size = test_table2['sepal_length'].count()
```

```
print(krl)
  if krl == 'linear':
    print(clf.coef_)
  print(good_size/all_size * 100)
  print(clf.score(test_table[['sepal_length',
                              'sepal_width',
                              'petal_length',
                              'petal_width']],
                  test_table[['class_label']]))
```

Пример 1. Accuracy метрика и линейные коэффициенты

Python 3.7.2 Shell

File Edit Shell Debug Options Window Help

```
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 23:09:28) [MSC v.1916 64 bit
(AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
====== RESTART: E:\Works\Victor\Students\STDB\Term2\example2-1runner.py ======
linear
[[-0.04916513  0.48106903 -0.88736393 -0.56915569]
 [-0.17488073  0.15898249 -0.46104921 -0.22257548]
 [ 0.35665083  0.56979449 -1.89145644 -1.8209073 ]]
93.33333333333333
0.9333333333333333
poly
96.66666666666667
0.9666666666666667
rbf
91.66666666666666
0.9166666666666666
sigmoid
28.333333333333332
0.2833333333333333
>>>
```

Ln: 20 Col: 4

Пример 2. Применим стандартизацию

example2-1runner.py - E:\Works\Victor\Students\STDB\Term2\example2-1runner.py (3.7.2)

File Edit Format Run Options Window Help

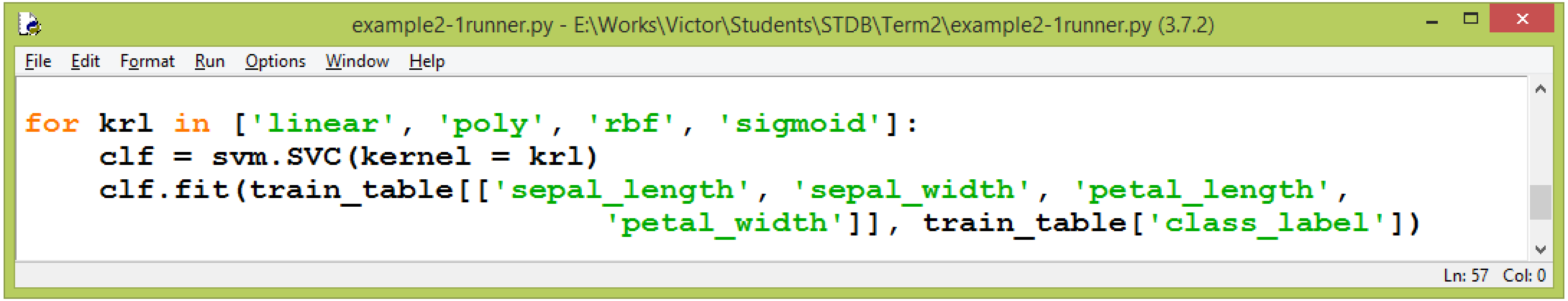
```
from sklearn import preprocessing
scaler_std = preprocessing.StandardScaler()

x = scaler_std.fit_transform(table[['sepal_length', 'sepal_width',
                                    'petal_length', 'petal_width']])
table[['sepal_length', 'sepal_width',
                                   'petal_length', 'petal_width']] = x

train_table, test_table = train_test_split(table, test_size = 0.4,
                                           random_state = 22222)
test_table = test_table.reset_index()
train_table = train_table.reset_index()
```

Ln: 38 Col: 43

Пример 2. Применим стандартизацию

```
from sklearn import preprocessing
scaler_std = preprocessing.StandardScaler()


x = scaler_std.fit_transform(table[['sepal_length', 'sepal_width',
                              'petal_length', 'petal_width']])
table[['sepal_length', 'sepal_width', 'petal_length', 'petal_width']] = x
train_table, test_table = train_test_split(table, test_size = 0.4,
                                           random_state = 22222)
test_table = test_table.reset_index()
train_table = train_table.reset_index()
```

Пример 2. Обучение




```
for krl in ['linear', 'poly', 'rbf', 'sigmoid']:
    clf = svm.SVC(kernel = krl)
    clf.fit(train_table[['sepal_length', 'sepal_width', 'petal_length',
                         'petal_width']], train_table['class_label'])
```

Пример 2. Обучение

```
for krl in ['linear', 'poly', 'rbf', 'sigmoid']:
    clf = svm.SVC(kernel = krl)
    clf.fit(train_table[['sepal_length', 'sepal_width', 'petal_length',
                          'petal_width']], train_table['class_label'])
```

Пример 2. Тестирование, метрика и линейные коэффициенты

example2-1runner.py - E:\Works\Victor\Students\STDB\Term2\example2-1runner.py (3.7.2)

File Edit Format Run Options Window Help

```python
    res = clf.predict(test_table[['sepal_length', 'sepal_width', 'petal_length',
                                  'petal_width']])
    print(krl)
    if krl == 'linear':
        print(clf.coef_)
    print(clf.score(test_table[['sepal_length', 'sepal_width', 'petal_length',
                                'petal_width']], test_table[['class_label']]))
```

Ln: 57 Col: 0

Пример 2. Тестирование, метрика и линейные коэффициенты

```
res = clf.predict(test_table[['sepal_length', 'sepal_width', 'petal_length',
                              'petal_width']])
  print(krl)
  if krl == 'linear':
      print(clf.coef_)
  print(clf.score(test_table[['sepal_length', 'sepal_width', 'petal_length',
                              'petal_width']], test_table[['class_label']]))
```

Пример 2. Accuracy метрика и линейные коэффициенты

Python 3.7.2 Shell

File Edit Shell Debug Options Window Help

```
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 23:09:28) [MSC v.1916 64 bit
(AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
====== RESTART: E:\Works\Victor\Students\STDB\Term2\example2-1runner.py ======
linear
[[-0.1815098   0.54326934 -0.54851961 -0.77340287]
 [-0.19016356  0.18970543 -0.4623333  -0.41469316]
 [ 0.07519968  0.43910277 -1.93064707 -1.95965963]]
0.9666666666666667
poly
0.85
rbf
0.95
sigmoid
0.9166666666666666
>>> 
```

Ln: 16 Col: 4

Сравним

Метод	Без стандартизации	Со стандартизацией
Linear	93,3%	96,7%
Poly	96,7%	85%
RBF	91,7%	95%
Sigmoid	28,3%	91,7%

Пример 3. Используем метод главных компонент

Вычислим, сколько нужно главных компонент.

example2-1runner.py - E:\Works\Victor\Students\STDB\Term2\example2-1runner.py (3.7.2)

File Edit Format Run Options Window Help

```
from sklearn.decomposition import PCA
pca = PCA()
table_pca = pca.fit_transform(table[['sepal_length', 'sepal_width',
                                      'petal_length', 'petal_width']])
explained_variance_ratio = pca.explained_variance_ratio_
print(explained_variance_ratio)
explained_variance = pca.explained_variance_
print(explained_variance)
```

Ln: 73 Col: 25

Пример 3. Используем метод главных компонент

Вычислим, сколько нужно главных компонент.

```
from sklearn.decomposition import PCA
pca = PCA()
table_pca = pca.fit_transform(table[['sepal_length', 'sepal_width',
                                     'petal_length', 'petal_width']])
explained_variance_ratio = pca.explained_variance_ratio_
print(explained_variance_ratio)
explained_variance = pca.explained_variance_
print(explained_variance)
```

Сколько нужно компонент?

```
[0.72962445 0.22850762 0.03668922 0.00517871]
[2.93808505 0.9201649  0.14774182 0.02085386]
```

По методу Кайзера нужна 1 главная компонента. По критерию 95% вариативности – 2 компоненты.

Пример 3. Подготовим набор для 2-х компонент

example2-1runner.py - E:\Works\Victor\Students\STDB\Term2\example2-1runner.py (3.7.2)

File Edit Format Run Options Window Help

```
pca = PCA(n_components = 2)
pca.fit_transform(table[['sepal_length', 'sepal_width',
                         'petal_length', 'petal_width']])

table3_p = pca.transform(table[['sepal_length', 'sepal_width',
                         'petal_length', 'petal_width']])
table3 = pd.DataFrame(table3_p, columns = ['pc1', 'pc2'])
table3['class_label'] = table['class_label']
```

Ln: 74 Col: 0

Пример 3. Подготовим набор для 2-х компонент

```
pca = PCA(n_components = 2)
pca.fit_transform(table[['sepal_length', 'sepal_width',
                         'petal_length', 'petal_width']])


table3_p = pca.transform(table[['sepal_length', 'sepal_width',
                         'petal_length', 'petal_width']])
table3 = pd.DataFrame(table3_p, columns = ['pc1', 'pc2'])
table3['class_label'] = table['class_label']
```

Пример 3. Подготовим набор для 1-й компоненты

example2-1runner.py - E:\Works\Victor\Students\STDB\Term2\example2-1runner.py (3.7.2)

File Edit Format Run Options Window Help

```
pca = PCA(n_components = 1)
pca.fit_transform(table[['sepal_length', 'sepal_width',
                                        'petal_length', 'petal_width']])

table4_p = pca.transform(table[['sepal_length', 'sepal_width',
                                        'petal_length', 'petal_width']])
table4 = pd.DataFrame(table4_p, columns = ['pc1'])
table4['class_label'] = table['class_label']
```

Ln: 82 Col: 44

Пример 3. Подготовим набор для 1-й компоненты

```
pca = PCA(n_components = 1)
pca.fit_transform(table[['sepal_length', 'sepal_width',
                         'petal_length', 'petal_width']])

table4_p = pca.transform(table[['sepal_length', 'sepal_width',
                         'petal_length', 'petal_width']])
table4 = pd.DataFrame(table4_p, columns = ['pc1'])
table4['class_label'] = table['class_label']
```

Пример 3. 2 компоненты. Обучение

example2-1runner.py - E:\Works\Victor\Students\STDB\Term2\example2-1runner.py (3.7.2)

File Edit Format Run Options Window Help

```
train_table3, test_table3 = train_test_split(table3, test_size = 0.4,
                                             random_state = 22222)
test_table = test_table3.reset_index()
train_table = train_table3.reset_index()

for krl in ['linear', 'poly', 'rbf', 'sigmoid']:
    clf = svm.SVC(kernel = krl)
    clf.fit(train_table[['pc1', 'pc2']], train_table['class_label'])
```

Ln: 98 Col: 0

Пример 3. 2 компоненты. Обучение

```
train_table3, test_table3 = train_test_split(table3, test_size = 0.4,
                                            random_state = 22222)
test_table = test_table3.reset_index()
train_table = train_table3.reset_index()


for krl in ['linear', 'poly', 'rbf', 'sigmoid']:
    clf = svm.SVC(kernel = krl)
    clf.fit(train_table[['pc1', 'pc2']], train_table['class_label'])
```

Пример 3. 2 компоненты. Тестирование

example2-1runner.py - E:\Works\Victor\Students\STDB\Term2\example2-1runner.py (3.7.2)

File Edit Format Run Options Window Help

```
    res = clf.predict(test_table[['pc1', 'pc2']])
    print(krl)
    if krl == 'linear':
        print(clf.coef_)
    print(clf.score(test_table[['pc1', 'pc2']], test_table[['class_label']]))
```

Ln: 109 Col: 0

Пример 3. 2 компоненты. Тестирование

```
res = clf.predict(test_table[['pc1', 'pc2']])
  print(krl)
  if krl == 'linear':
      print(clf.coef_)
  print(clf.score(test_table[['pc1', 'pc2']], test_table[['class_label']]))
```

Пример 3. 2 компоненты. Accuracy

Python 3.7.2 Shell

File Edit Shell Debug Options Window Help

```
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 23:09:28) [MSC v.1916 64 bit
(AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> 
====== RESTART: E:\Works\Victor\Students\STDB\Term2\example2-1runner.py ======
[0.72962445 0.22850762 0.03668922 0.00517871]
[2.93808505 0.9201649  0.14774182 0.02085386]
linear
[[-1.08647484  0.39891273]
 [-0.70330871  0.08047453]
 [-2.42249679  0.38776215]]
0.9
poly
0.8333333333333334
rbf
0.9166666666666666
sigmoid
0.9
>>> 
```

Ln: 18 Col: 4

Пример 3. 1 компонента. Обучение

example2-1runner.py - E:\Works\Victor\Students\STDB\Term2\example2-1runner.py (3.7.2)

File Edit Format Run Options Window Help

```
train_table4, test_table4 = train_test_split(table4, test_size = 0.4,
                                              random_state = 22222)
test_table = test_table4.reset_index()
train_table = train_table4.reset_index()

for krl in ['linear', 'poly', 'rbf', 'sigmoid']:
    clf = svm.SVC(kernel = krl)
    clf.fit(train_table[['pc1']], train_table['class_label'])
```

Ln: 119 Col: 0

Пример 3. 1 компонента. Обучение

```
train_table4, test_table4 = train_test_split(table4, test_size = 0.4,
                                          random_state = 22222)
test_table = test_table4.reset_index()
train_table = train_table4.reset_index()


for krl in ['linear', 'poly', 'rbf', 'sigmoid']:
    clf = svm.SVC(kernel = krl)
    clf.fit(train_table[['pc1']], train_table['class_label'])
```

Пример 3. 1 компонента. Тестирование

example2-1runner.py - E:\Works\Victor\Students\STDB\Term2\example2-1runner.py (3.7.2)

File Edit Format Run Options Window Help

```
    res = clf.predict(test_table[['pc1']])
    print(krl)
    if krl == 'linear':
        print(clf.coef_)
    print(clf.score(test_table[['pc1']], test_table[['class_label']]))
```

Ln: 128 Col: 4

Пример 3. 1 компонента. Тестирование

```
res = clf.predict(test_table[['pc1']])
  print(krl)
  if krl == 'linear':
      print(clf.coef_)
  print(clf.score(test_table[['pc1']], test_table[['class_label']]))
```

Пример 3. 1 компонента. Accuracy

Python 3.7.2 Shell

File Edit Shell Debug Options Window Help

```
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 23:09:28) [MSC v.1916 64 bit
(AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
====== RESTART: E:\Works\Victor\Students\STDB\Term2\example2-1runner.py ======
[0.72962445 0.22850762 0.03668922 0.00517871]
[2.93808505 0.9201649  0.14774182 0.02085386]
linear
[[-1.37844998]
 [-0.71626803]
 [-2.31214949]]
0.8833333333333333
poly
0.8666666666666667
rbf
0.8666666666666667
sigmoid
0.9166666666666666
>>> 
```

Ln: 18 Col: 4

Сравним

Метод	Без стандартизации	Со стандартизацией	PCA 2	PCA1
Linear	93,3%	96,7%	90%	88,3%
Poly	96,7%	85%	83,3%	86,7%
RBF	91,7%	95%	91,7%	86,7%
Sigmoid	28,3%	91,7%	90%	91,7%

Можно ли улучшить? Поиграем с параметрами.

Пример 4. Разные параметры

example2-1runner.py - E:\Works\Victor\Students\STDB\Term2\example2-1runner.py (3.7.2)

File Edit Format Run Options Window Help

```
train_table, test_table = train_test_split(table, test_size = 0.4,
                                           random_state = 22222)
test_table = test_table.reset_index()
train_table = train_table.reset_index()

gammas = ['scale', 'auto', 1, 10, 0.1, 20, 0.01]
coef0s = [0, 1, 2, 5, 10, 0.1, 0.01]
degrees = [2, 3, 4, 5]
Cs = [0.001, 0.01, 0.1, 0.5, 1, 2, 10, 20]
```

Ln: 134 Col: 39

Пример 4. Разные параметры

```
train_table, test_table = train_test_split(table, test_size = 0.4,
                                           random_state = 22222)
test_table = test_table.reset_index()
train_table = train_table.reset_index()


gammas = ['scale', 'auto', 1, 10, 0.1, 20, 0.01]
coef0s = [0, 1, 2, 5, 10, 0.1, 0.01]
degrees = [2, 3, 4, 5]
Cs = [0.001, 0.01, 0.1, 0.5, 1, 2, 10, 20]
```

Пример 4. Перебор ядер и параметров

example2-1runner.py - E:\Works\Victor\Students\STDB\Term2\example2-1runner.py (3.7.2)

File Edit Format Run Options Window Help

```
for krl in ['linear', 'poly', 'rbf', 'sigmoid']:

    best = [0]

    for gamma in gammas:
        for coef0 in coef0s:
            for degree in degrees:
                for C in Cs:
                    clf = svm.SVC(kernel = krl, gamma = gamma, coef0 = coef0,
                                  degree = degree, C = C)
```

Ln: 132 Col: 0

Пример 4. Перебор ядер и параметров

```
for krl in ['linear', 'poly', 'rbf', 'sigmoid']:

    best = [0]

    for gamma in gammas:
        for coef0 in coef0s:
            for degree in degrees:
                for C in Cs:
                    clf = svm.SVC(kernel = krl, gamma = gamma, coef0 = coef0,
                                  degree = degree, C = C)
```

Пример 4. Обучение и тестирование

example2-1runner.py - E:\Works\Victor\Students\STDB\Term2\example2-1runner.py (3.7.2)

File Edit Format Run Options Window Help

```python
                    clf.fit(train_table[['sepal_length', 'sepal_width',
                                         'petal_length', 'petal_width']],
                            train_table['class_label'])

                    res = clf.predict(test_table[['sepal_length', 'sepal_width',
                                                  'petal_length', 'petal_width']]
                    score = clf.score(test_table[['sepal_length', 'sepal_width',
                                                  'petal_length', 'petal_width']]
                                      test_table[['class_label']])
                    if score > best[0]:
                        best = [score, gamma, coef0, degree, C]
    print(krl, best)
```

Ln: 156 Col: 57

Пример 4. Обучение и тестирование

```
degree = degree, C = C)
            clf.fit(train_table[['sepal_length', 'sepal_width',
                         'petal_length', 'petal_width']], train_table['class_label'])
            res = clf.predict(test_table[['sepal_length', 'sepal_width',
                              'petal_length', 'petal_width']])
            score = clf.score(test_table[['sepal_length', 'sepal_width',
                              'petal_length', 'petal_width']],
                       test_table[['class_label']])
            if score > best[0]:
               best = [score, gamma, coef0, degree, C]
  print(krl, best)
```

Пример 4. Результаты

```
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 23:09:28) [MSC v.1916 64 bit
(AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
====== RESTART: E:\Works\Victor\Students\STDB\Term2\example2-1runner.py ======
linear [0.9666666666666667, 'scale', 0, 2, 1]
poly [0.9833333333333333, 'auto', 0.01, 2, 20]
rbf [0.9666666666666667, 'scale', 0, 2, 10]
sigmoid [0.95, 'scale', 0.1, 2, 1]
>>>
```

Сравним

Метод	Без стандартизации	Со стандартизацией	PCA 2	PCA1	После подбора параметров
Linear	93,3%	96,7%	90%	88,3%	96,7%
Poly	96,7%	85%	83,3%	86,7%	98,3%
RBF	91,7%	95%	91,7%	86,7%	96,7%
Sigmoid	28,3%	91,7%	90%	91,7%	95%

Интернет ресурсы и литература

1. https://scikit-learn.org/stable/modules/svm.html
2. https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html#sklearn.svm.SVC

Часть 3

ПОСТРОЕНИЕ ПРЕДИКТИВНОГО (ДОВЕРИТЕЛЬНОГО) ЭЛЛИПСА

Источники

Для составления программы использовались:

- https://matplotlib.org/devdocs/gallery/statistics/confidence_ellipse.html
- https://www.xarg.org/2018/04/how-to-plot-a-covariance-error-ellipse/

Программа. Часть 1

example6runner.py - E:\Works\Victor\Students\STDB\Lection6\example6runner.py (3.7.2)

File Edit Format Run Options Window Help

```
def confidence_ellipse(x, y, ax, p_value, facecolor = 'none', **kwargs):
    if x.size != y.size:
        raise ValueError("x and y должны быть одинакового размера")

    cov = np.cov(x, y)
    pearson = cov[0, 1] / np.sqrt(cov[0, 0] * cov[1, 1])
    # Получение собственных значений
    ell_radius_x = np.sqrt(1 + pearson)
    ell_radius_y = np.sqrt(1 - pearson)
    ellipse = Ellipse((0, 0),
        width = ell_radius_x * 2,
        height = ell_radius_y * 2,
        facecolor = facecolor,
        **kwargs)
```

Ln: 70 Col: 39

Программа. Часть 2

example6runner.py - E:\Works\Victor\Students\STDB\Lection6\example6runner.py (3.7.2)

File Edit Format Run Options Window Help

```python
# Масштабирование согласно стандартному отклонению и p-value
if p_value > 0 and p_value < 1:
    n_std = math.sqrt(-2 * math.log(p_value))
else:
    print("P-vale должно быть в интервале от 0 до 1")

# x
scale_x = np.sqrt(cov[0, 0]) * n_std
mean_x = np.mean(x)

# y
scale_y = np.sqrt(cov[1, 1]) * n_std
mean_y = np.mean(y)
```

Ln: 70 Col: 39

Программа. Часть 3

example6runner.py - E:\Works\Victor\Students\STDB\Lection6\example6runner.py (3.7.2)

File Edit Format Run Options Window Help

```
    # Преобразование координат, согласно полученным значениям

    transf = transforms.Affine2D() \
        .rotate_deg(45) \
        .scale(scale_x, scale_y) \
        .translate(mean_x, mean_y)

    ellipse.set_transform(transf + ax.transData)
    return ax.add_patch(ellipse)
```

Ln: 70 Col: 39

Запуск (на примере 1)

example6runner.py - E:\Works\Victor\Students\STDB\Lection6\example6runner.py (3.7.2)

File Edit Format Run Options Window Help

```
fig, ax = plt.subplots(figsize=(8, 8))
confidence_ellipse(table['days'], table['points'], ax, 0.05,
                   edgecolor = 'green')
ax.scatter(table['days'], table['points'], s = 10)
plt.show()
```

Ln: 70 Col: 39

Результат