Машинное обучение и интеллектуальный анализ данных

Семинар 3

Г.А. Ососков*, О.И. Стрельцова*, Д.И. Пряхина*, Д.В. Подгайный*, А.В. Стадник*, Ю.А. Бутенко*

Государственный университет «Дубна»

***Лаборатория информационных технологий, ОИЯИ**

Дубна, Россия

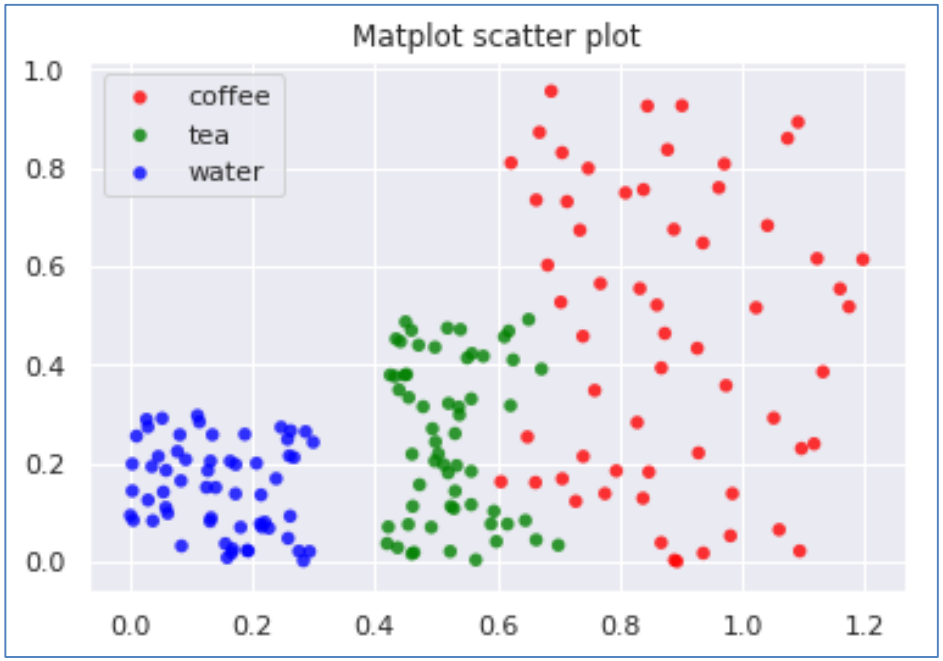
Задачи машинного обучения. Задачи классификации

Реализация средствами Python	Реализация средствами Python + NumPy	Реализация средствами Python + NumPy + Scikit-Learn

matplotlib – для визуализации данных, результатов и т.д.

seaborn: statistical data visualization

Data generation for classification tasks:




```
from sklearn.datasets.samples_generator import make_blobs
```

Библиотека scikit-learn. *Machine Learning in Python*: https://scikit-learn.org/dev/

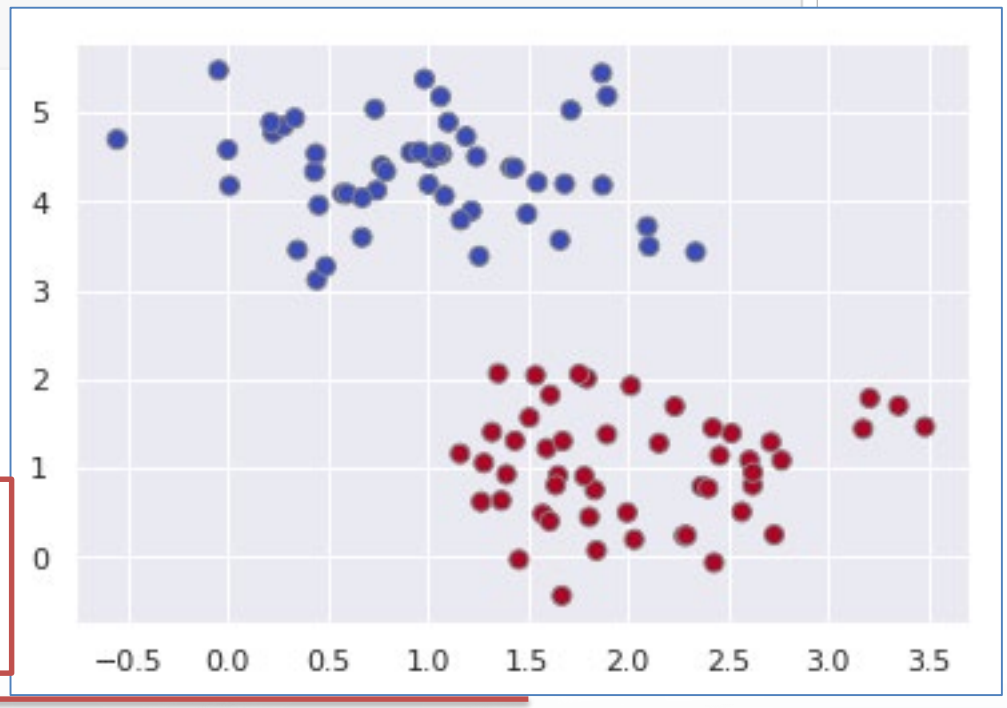
1. Генерация данных (модельных)

```
from sklearn.datasets import make_blobs
```

```
X, y = make_blobs(n_samples=100, centers=2, n_features=2,
                  random_state=0, cluster_std=0.60)
print(X.shape)
plt.scatter(X[:, 0], X[:, 1], c=y, edgecolor="grey", s=50, cmap='coolwarm');

print(y)
```

Смоделированные данные:
2 центра;
2 признака;
Параметр
`cluster_std= 0.6`



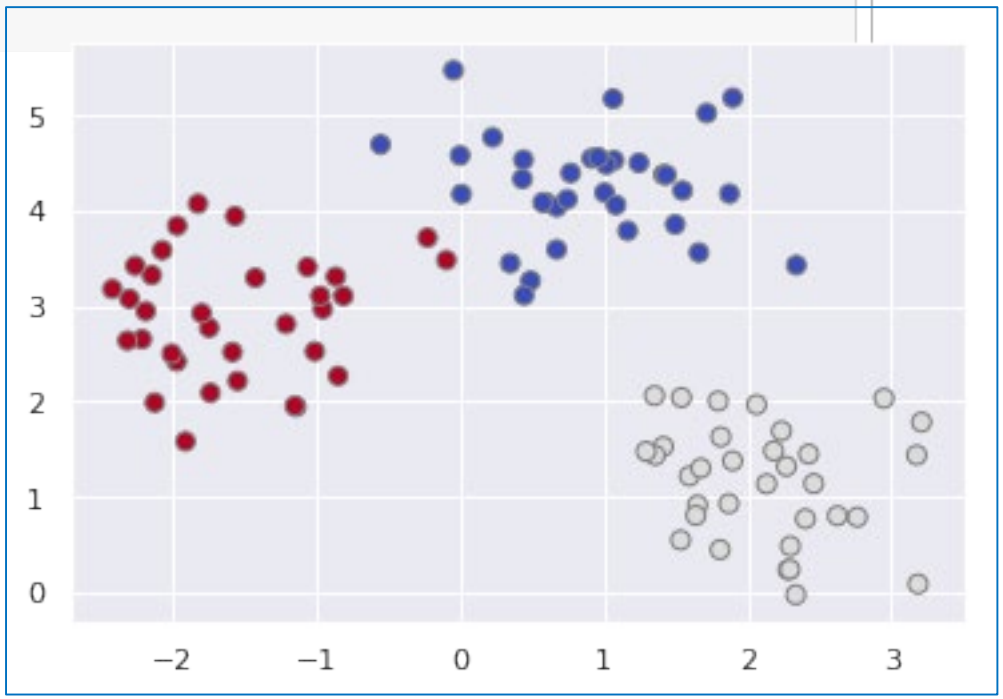
Изменяем параметр
(Стандартное отклонение кластеров):
`cluster_std (0.9;1.0,1.6,2.6)`

Библиотека scikit-learn. *Machine Learning in Python*:
Generate isotropic Gaussian blobs for clustering.
https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_blobs.html

Генерация данных (модельных)

Изменяем количество центров:
centers = **3** , cluster_std= **0.6**

```
1  X, y = make_blobs(n_samples=100, centers=3, n_features=2,
2                    random_state=0, cluster_std=0.6)
3  print(X.shape)
4  plt.scatter(X[:, 0], X[:, 1], c=y, edgecolor="grey", s=50, cmap='coolwarm');
5  print(y)
```



Цвет признака передается через
c = y

!

Обратите внимание какой массив сгенерирован:
`print(X.shape)`

Библиотека scikit-learn. *Machine Learning in Python*:
Generate isotropic Gaussian blobs for clustering.
https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_blobs.html

Метод k ближайших соседей: K-means

Mathematical formulation

The `KMeans` algorithm clusters data by trying to separate samples in n groups of equal variance, minimizing a criterion known as the *inertia* or within-cluster sum-of-squares (see below). This algorithm requires the number of clusters to be specified. It scales well to large number of samples and has been used across a large range of application areas in many different fields.

The k-means algorithm divides a set of N samples X into K disjoint clusters C, each described by the mean μ_j of the samples in the cluster. The means are commonly called the cluster "centroids"; note that they are not, in general, points from X, although they live in the same space.

The K-means algorithm aims to choose centroids that minimise the **inertia**, or **within-cluster sum-of-squares criterion**:

$$\sum_{i=0}^{n} \min_{\mu_j \in C} (||x_i - \mu_j||^2)$$

Библиотека scikit-learn. *Machine Learning in Python*:
https://scikit-learn.org/stable/modules/clustering.html#k-means

Метод k ближайших соседей: KNeighborsClassifier

Example

```
>>> X = [[0], [1], [2], [3]]
>>> y = [0, 0, 1, 1]
>>> from sklearn.neighbors import KNeighborsClassifier
>>> neigh = KNeighborsClassifier(n_neighbors=3)
>>> neigh.fit(X, y)
KNeighborsClassifier(...)
>>> print(neigh.predict([[1.1]]))
[0]
>>> print(neigh.predict_proba([[0.9]]))
[[0.66666667 0.33333333]]
```

Библиотека scikit-learn. *Machine Learning in Python*:
https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html