SCIKIT-LEARN

(самая известная библиотека Python для машинного обучения)

scikit-learn требует наличия пакетов NumPy и SciPy.
Для построения графиков и интерактивной работы необходимо также установить matplotlib, IPython и Jupyter Notebook

Установка свободного дистрибутива Python для научных вычислений, специально предназначенного для Windows, включающего:

NumPy, SciPy, matplotlib, pandas, IPython и scikit-learn

Установка из командной строки Windows cmd: **pip install numpy scipy matplotlib ipython scikit-learn pandas**
Запуск cmd: **ipython3 notebook**
Запустится в веб-браузере Jupyter notebook

[Удобнее работать в JupyterLab: инсталл. **pip install jupyter lab**,
запуск: **jupyter lab**
Запустится JupyterLab]

Jupyter Notebook, JupyterLab

Интерактивная среда для запуска программного кода в браузере.
Инструмент для анализа данных,
Позволяет легко интегрировать программный код, текст и изображения.

NumPy

Один из основных пакетов для научных вычислений в Python.
Содержит функциональные возможности для работы с многомерными массивами, высокоуровневыми математическими функциями (операции линейной алгебры, преобразование Фурье, генератор псевдослучайных чисел).
Задает структуру данных - массив «NumPy»

Класс ndarray, многомерный (n-мерный) массив

```
import numpy as np
x = np.array([[1, 2, 3], [4, 5, 6]])
print("x:\n{}".format(x))
x:
[[1 2 3]
 [4 5 6]]
```

SciPy

Библиотека для научных вычислений: матричные вычисления, процедуры линейной алгебры, оптимизация, обработка сигналов, статистика.

SCIKIT-LEARN использует набор функций SciPy для реализации своих алгоритмов.

Пакет scipy.sparse создает разреженные матрицы (sparse matrices), которые представляют собой еще один формат данных для SCIKIT-LEARN.

Разреженная матрица - это матрица с преимущественно нулевыми элементами.

Подробную информацию о разреженных матрицах SciPy можно найти в **SciPy Lecture Notes**

```
 # (Создаем 2D массив NumPy с единицами по главной диагонали и нулями в остальных ячейках)
from scipy import sparse
eye = np.eye(4)
#numpy.eye(R, C = None, k = 0, dtype = type <‘float’>) : Return a matrix having 1’s on the diagonal and 0’s elsewhere w.r.t. k
print("массив NumPy:\n{}".format(eye))
массив NumPy:
[[1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]
 [0. 0. 0. 1.]]
```

SciPy

```
# Массив NumPy преобразуем в разреженную матрицу SciPy в формате CSR
# Compressed Sparse Row Format (CSR), Compressed Sparse Column Format (CSC)
sparse_matrix = sparse.csr_matrix(eye)                    # единичная - по диагонали 1, ост.0
print("\nразреженная матрица SciPy в формате CSR:\n{}".format(sparse_matrix))
разреженная матрица SciPy в формате CSR:
(0, 0) 1.0
(1, 1) 1.0
(2, 2) 1.0
(3, 3) 1.0
```

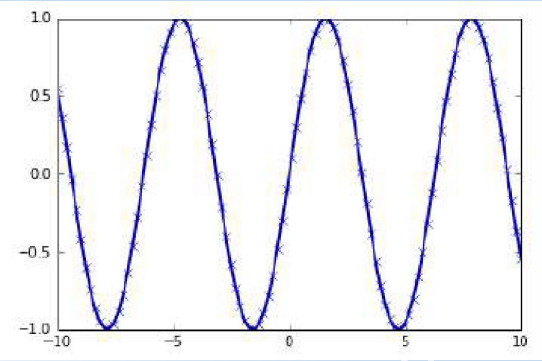
```
# Создание разреженной матрицы с использованием формата
# COO (coordinate format) – координатный формат, задаем только координаты ненулевые элементов матрицы
# (номера строк и столбцов)
data = np.ones(4)
row_indices = np.arange(4)
col_indices = np.arange(4)
eye_coo = sparse.coo_matrix((data, (row_indices, col_indices)))
print("формат COO:\n{}".format(eye_coo))
формат COO:
(0, 0) 1.0
(1, 1) 1.0
(2, 2) 1.0
(3, 3) 1.0
```

Задание: создать разреженную матрицу M, dim(M)=10×6, где $M_{2,4}=M_{6,4}=M_{2,5}=M_{6,6}=1$ с использованием обоих форматов. Вывести на печать, сравнить.

Matplotlib

Основная библиотека для построения графиков.
Включает функции для создания высококачественных визуализаций типа линейных диаграмм, гистограмм, диаграмм разброса и т.д.
При работе в Jupyter Notebook можно вывести рисунок прямо в браузере с помощью встроенных команд %matplotlib notebook и %matplotlib inline.

```
# Построение графика с использованием библ. Matplotlib
%matplotlib inline
import matplotlib.pyplot as plt
x = np.linspace(-10, 10, 100)          # переменная X из 100 чисел от -10 до 10 (ось абсцисс)
y = np.sin(x)                          # функция от X
plt.plot(x, y, marker="x")             # построение графика
```



Pandas

Библиотека для обработки и анализа данных.
Построена на основе структуры данных DataFrame (таблицы, похожие на таблицы Excel).
Имеет широкие возможности по работе с таблицами, в частности, позволяет выполнять SQL-подобные запросы.
В отличие от NumPy, который требует, чтобы все записи в массиве были одного и того же типа, в pandas каждый столбец может иметь отдельный тип (например, целые числа, даты, числа с плавающей точкой и строки).
Способна работать с различными форматами файлов и баз данных, например, с файлами SQL, Excel и CSV.

Подробная информация – в книге
McKinney W. Python for Data Analysis. Data Wrangling with Pandas, NumPy, and Ipython. O'Reilly, 2012

```
# Пример создания DataFrame таблицы
inlineimport pandas as pd
# набор данных с характеристиками пользователей
data = {'Name': ["John", "Anna", "Peter", "Linda"], 'Location' : ["New York", "Paris", "Berlin", "London"], 'Age' : [24, 13, 53, 33]}
data_pandas = pd.DataFrame(data)
display(data_pandas)                    # IPython.display позволяет "красиво напечатать"  таблицу
```

	Age	Location	Name
0	24	New York	John
1	13	Paris	Anna
2	53	Berlin	Peter
3	33	London	Linda

в Jupyter notebook

	Name	Location	Age
0	John	New York	24
1	Anna	Paris	13
2	Peter	Berlin	53
3	Linda	London	33

в JupyterLab

Современные методы анализа данных

Байесовский классификатор

- линейный дискриминант Фишера;
- метод парзеновского окна;
- разделение смеси вероятностных распределений, EM(empirical mode) алгоритмы;
- метод потенциальных функций или метод радиальных базисных функций;
- метод ближайших соседей.

Нейронная сеть

- персептрон;
- многослойный персептрон;
- сети векторного квантования, обучаемые с учителем (Learning Vector Quantization);
- гибридная сеть встречного распространения.

Индукция правил

- решающее дерево;
- решающий список;
- решающий лес;
- тестовый алгоритм;
- алгоритм вычисления оценок.

Линейный разделитель

- линейный дискриминант Фишера;
- однослойный персептрон;
- логистическая регрессия;
- машина опорных векторов.

Алгоритмическая композиция

- взвешенное голосование;
- бустинг;
- бэггинг;
- метод комитетов;
- смесь экспертов.

Сокращение размерности

- селекция признаков;
- метод главных компонент;
- метод независимых компонент;
- многомерное шкалирование.

Выбор модели

- минимизация эмпирического риска;
- структурная минимизация риска;
- минимум длины описания;
- скользящий контроль;
- извлечение признаков
- самоорганизация моделей;
- случайный поиск с адаптацией;
- генетический алгоритм.

2. Метод ближайших соседей в задаче классификации

Вспомнить:

***class* (target, цель)**

Есть ли на фото тигр?
Болен ли пациент таким-то заболеванием?
Продается ли этот товар нужными объемами?

классификация

Обучить классификатор на известных классах так, чтобы при предъявлении ему неизвестного класса, он отнес бы его к одному из известных.

Задача: классифицировать сорта цветков ириса

Исходные данные: *features*
-длина и ширина лепестков (см),
-длина и ширина чашелистиков (см).

Возможные сорта *classes*
-Setosa,
-Versicolor,
-Virginica
различаются на основе перечисленных характеристик (признаков, features)

Цель: построить классификатор (модель машинного обучения), который сможет обучиться на основе перечисленных характеристик цветков ириса, классифицированных по сортам, и затем предскажет сорт для любого далее предъявляемого ему цветка ириса. *labels*

? Это обучение с учителем или без?

Поскольку есть примеры классов, то решаемая задача является задачей *обучения с учителем*

2. Метод ближайших соседей в задаче классификации

2.1 Классификация с использованием библиотеки scikit-learn

Загрузить файл данных из модуля datasets библиотеки scikit-learn, вызвав функцию load_iris:

```
# загрузка файла данных  загрузка ф-цией load_iris
import numpy as np
from sklearn.datasets import load_iris                    # набор данных создан в 1936 году
iris_dataset = load_iris()
X = iris_dataset.data
y = iris_dataset.target
```

Объект iris - набор типа Bunch: содержит ключи и значения. Просмотр структуры

```
#объект Bunch - содержит информацию о наборе данных, а также данные
# Структура - ключи и значения
print("Ключи iris_dataset: \n{}".format(iris_dataset.keys()))
```

Ключи iris_dataset:
dict_keys(['target_names', 'feature_names', 'DESCR', 'data', 'target'])

```
# ключ DESCR – краткое описание набора данных/ Просмотр DESCR одним из способов:
print(iris_dataset.DESCR)
# print("Ключи iris_dataset: \n{}".format(iris_dataset.keys()))
# print("Ключи iris_dataset: {}".format(iris_dataset.DESCR))
```

```
# Сами данные записаны в массивах target и data. data – массив NumPy, который содержит количественные измерения длины
# чашелистиков, ширины чашелистиков, длины лепестков и ширины лепестков:
print("Тип массива data: {}".format(type(iris_dataset['data'])))
```

Тип массива data: <class 'numpy.ndarray'>

2. Метод ближайших соседей в задаче классификации

2.1 Классификация с использованием библиотеки scikit-learn

```
# Сами данные записаны в массивах target и data. data – массив NumPy, который содержит количественные измерения длины
# чашелистиков, ширины чашелистиков, длины лепестков и ширины лепестков:
print("Тип массива data: {}".format(type(iris_dataset['data'])))
```

Тип массива data: <class 'numpy.ndarray'>

```
# Строки в data соответствуют цветам ириса = примерам (samples), а столбцы - 4 характеристики (признака, feautures)
print("Форма (shape) массива data: {}".format(iris_dataset['data'].shape))
```

Форма (*shape*) массива data: (150, 4)

Задание1: вывести на печать первые 5 примеров (samples) массива data

2.1 Классификация с использованием библиотеки scikit-learn

```
# Массив target содержит сорта измеренных цветов, записанные в виде массива NumPy
# и представляет собой одномерный массив - по 1му элементу для каждого цветка
# Сорта (классы) кодируются  целыми числами от 0 до 2:
# 0 – setosa,1 – versicolor, 2 – virginica

print("Тип массива target: {}".format(type(iris_dataset['target'])))
print("Форма массива target: {}".format(iris_dataset['target'].shape))
print("Классы:\n{}".format(iris_dataset['target']))
```

```
Тип массива target: <class 'numpy.ndarray'>
Форма массива target: (150,)
Классы:
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
2 2]
```

Задание2: просмотреть остальные ключи

2.1 Классификация с использованием библиотеки scikit-learn

Для решения задачи классификации с учителем надо иметь 2 набора данных:

-обучающие данные (training data, training set).

-тестовые данные (test data, test set, hold-out set).

Функция train_test_split (библиотека scikit-learn) перемешивает исходный набор данных случайным образом и разбивает его на две части: обучающий набор = 75% samples, тестовый набор = 25% samples

```
# Чтобы в точности для отладки повторно воспроизвести случайное перемешивание, в генераторе псевдослучайных чисел зададим
# фиксированное стартовое значение random_state=0
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(iris_dataset['data'], iris_dataset['target'], random_state=0)

форма массива X_train: (112, 4)
форма массива y_train: (112,)
форма массива X_test: (38, 4)
форма массива y_test: (38,)
```

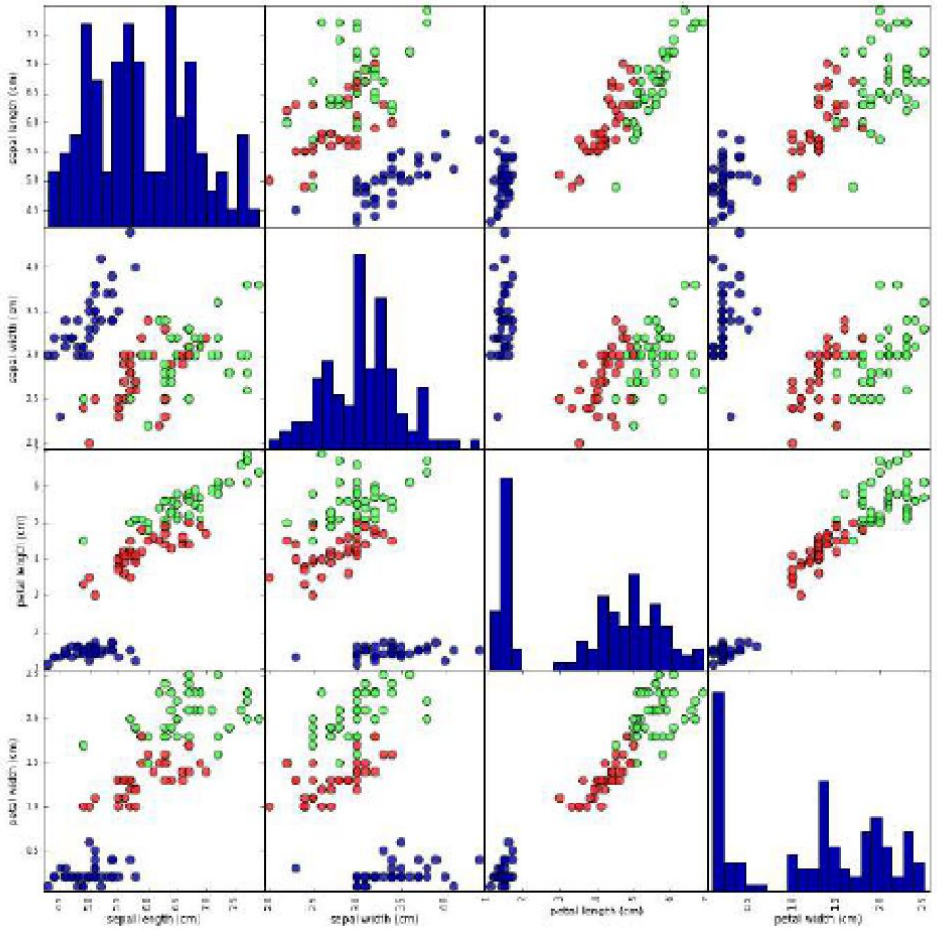
2.1 Классификация с использованием библиотеки scikit-learn

Качественный анализ данных: матрица диаграмм рассеяния

Для пары признаков – на плоскости (scatter plot). Если признаков больше, то строятся матрицы диаграмм (scatterplot matrix, pair plots) для всех возможных пар (в pandas функция scatter_matrix)

```
# матрица диаграмм рассеяния
# создаем таблицу (dataframe) из данных в массиве X_train
# маркируем столбцы, используя строки в iris_dataset.feature_names
# создаем матрицу рассеяния из dataframe, цвет точек атоматом, По диагонали - гистограммы каждого признака
import pandas as pd
from pandas import plotting
%matplotlib inline
iris_dataframe = pd.DataFrame(X_train, columns=iris_dataset.feature_names)
grr = plotting.scatter_matrix(iris_dataframe, c=y_train, figsize=(15, 15), marker='o',
hist_kwds={'bins': 20}, s=60, alpha=.8)
```

2.1 Классификация с использованием библиотеки scikit-learn



Задание3: сделать вывод по матрицам рассеяния

Признаки позволяют относительно хорошо разделить три класса
Модель машинного обучения, вероятно,
сможет научиться разделять их.

2.1 Классификация с использованием библиотеки scikit-learn

Для решения задачи классификации с учителем (построения классификатора) используем метод k ближайших соседей

Библиотека scikit-learn, где модели машинного обучения реализованы в собственных классах, называемых Estimators (support vector machines (SVM), random forests, neural networks).

Алгоритм классификации на основе метода k ближайших соседей реализован в классификаторе KNeighborsClassifier модуля neighbors.

Задание4: формализовать использование метода k ближайших соседей для решения рассматриваемой задачи классификации, пояснить особенности его использования

- Тренировка выполняется на обучающем наборе данных;
- В ходе классификации вводимой точки данных алгоритм находит точку в обучающем наборе, которая ближе всего находится к вводу;
- присвоение метки (классификация, отнесение к классу) введенной новой точки.

k означает: вместо того, чтобы использовать лишь «ближайшего соседа» к вводу, рассматривается любое фиксированное число k>1 соседей (например, три, или пять, или более соседей).

Т.о. классификация (прогноз, predict) для вводимой точки данных выполняется для того класса, которому принадлежит большинство из k соседей.

2.1 Классификация с использованием библиотеки scikit-learn

Создать объект-экземпляр класса, задав параметры модели: количество соседей k (установим k = 1)

```
# Создать объект-экземпляр класса, задав параметры модели: количество соседей k= 1
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=1)        # Объект knn включает алгоритм, который будет использоваться для
                                                 # обучения, а также алгоритм классификации
```

Для обучения вызывать метод fit объекта knn, который принимает в качестве аргументов массивы NumPy: X_train и y_train, содержащие обучающие и тестовые данные

```
knn.fit(X_train, y_train)
```

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski', metric_params=None, n_jobs=1, n_neighbors=1, p=2,
weights='uniform')
```

Основные функции алгоритмов библиотеки scikit-learn:

-fit(): выполняет алгоритм обучения. Входные параметры:
-массив (матрица) обучающих данных (samples по строкам, features по столбцам)
-классы, соответствующие строкам матрицы

-predict(): Классификация. Входные параметры:
-или новые входные данные, или тестовый набор данных - массивы (матрицы) обучающих данных (samples по строкам, features по столбцам)

2.1 Классификация с использованием библиотеки scikit-learn

Для обучения вызывать метод fit объекта knn, который принимает в качестве аргументов массивы NumPy: X_train и y_train, содержащие обучающие и тестовые данные

```
knn.fit(X_train, y_train)
```

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski', metric_params=None, n_jobs=1, n_neighbors=1, p=2, weights='uniform')
```

Почти все параметры классификатора KNeighborsClassifier имеют значения по умолчанию (параметр n_neighbor=1 задавали).

Большинство классификаторов в scikit-learn имеют массу параметров, но большая часть из них связана с оптимизацией скорости вычислений или предназначена для особых случаев использования.

Не стоит подробно останавливаться на всех параметрах, выводимых классификатором

Задание5: разобраться с полями algorithm, leaf_size, metric, metric_params, n_jobs, p, weights. Пояснить метрики (евклидова, манхеттена, и др.), какие из них в каких случаях целесообразно использовать.

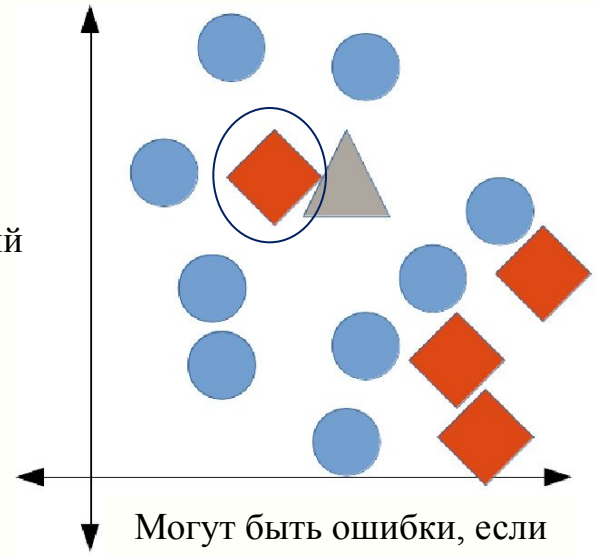
2.1 Классификация с использованием библиотеки scikit-learn

Выбор n_neighbor

правильная классификация – Круг.
Но т.к. ближайший сосед - Квадрат, то при n=1 ответ будет неверным

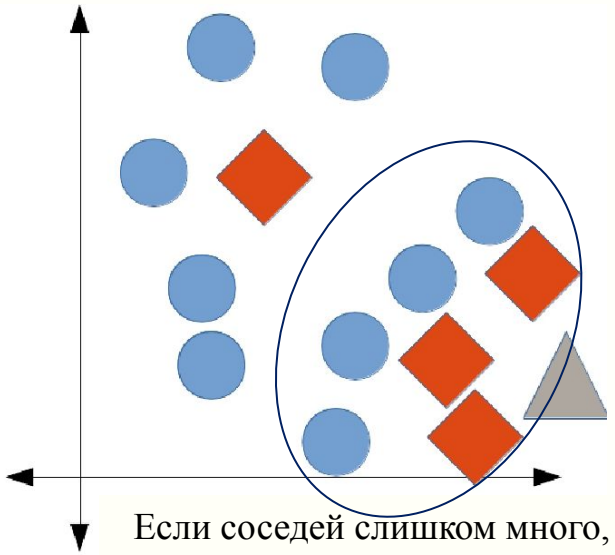
правильная классификация – Квадрат.
Но если n=7 ответ будет неверным

ближайший сосед



Могут быть ошибки, если соседей мало

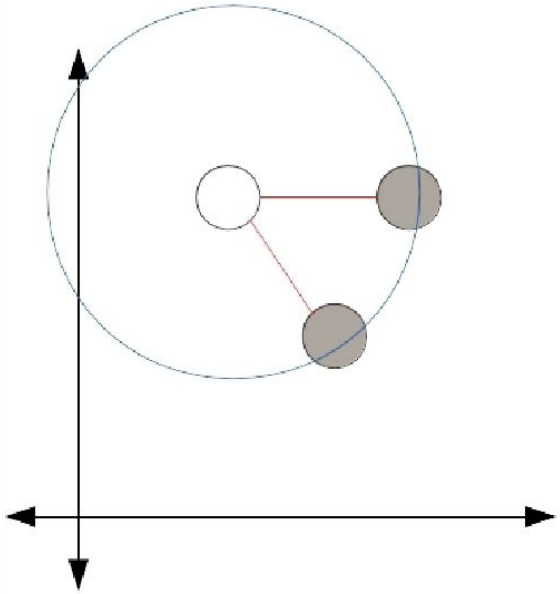
ближайшие соседи



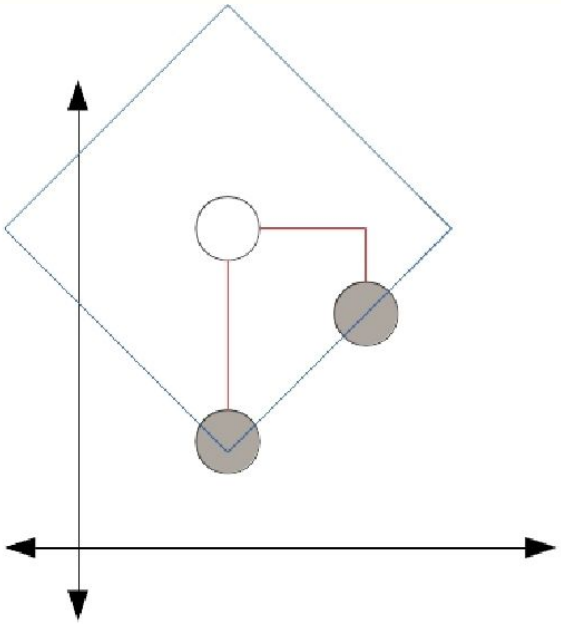
Если соседей слишком много, то может ухудшаться точность

2.1 Классификация с использованием библиотеки scikit-learn

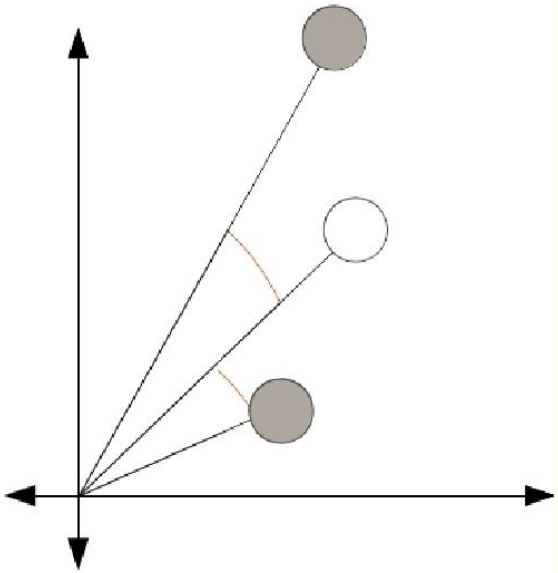
Особенности метрик



Евклидова метрика



Манхеттеновское расстояние



Косинусное сходство

Когда значительная разница в абсолютных значениях features и большая разреженность исходных данных

2.1 Классификация с использованием библиотеки scikit-learn

Оценка качества работы классификатора (валидация)

Оценка правильности (accuracy) выполняется на тестовых данных

```
# Валидация классификатора на тестовых данных
# проверка расчетом м.о. mean
y_test = knn.predict(X_test)                    # Для классификации на тестовых данных вызываем ф-цию predict объекта knn
print(«Классификация тестового набора:\n {}".format(y_test))
print("Правильность валидации (сравнение рез-тов классиф-ра с тестовым набором): {:.2f}".format(np.mean(y_valid == y_test)))

# или та же проверка с использ. метода score объекта knn
# может выполняться сразу по входным наборам (без разделения train_test_split)
print("Правильность методом score: {:.2f}".format(knn.score(X_test, y_test)))
```

Классификация тестового набора:
[2 1 0 2 0 2 0 1 1 1 2 1 1 1 1 0 1 1 0 0 2 1 0 0 2 0 0 1 1 0 2 1 0 2 2 1 0 2]
Правильность валидации (сравнение рез-тов классиф-ра с тестовым набором): 0.97
Правильность методом score: 0.97

```
# или валидация с использованием ф-ции cross_val_score
# может выполняться сразу по входным наборам (без разделения train_test_split)

from sklearn.model_selection import cross_val_score
scores = cross_val_score(knn, X_train, y_train, scoring='accuracy', cv=3)    #по обучающим массивам!!! Не нужны тестовые выборки
average_accuracy = np.mean(scores) * 100
print("Точность с исп. ф-ции cross_val_score на обучающем массиве ДРУГАЯ! {0:.1f}%".format(average_accuracy))

scores_All = cross_val_score(knn, X, y, scoring='accuracy', cv=3)
average_accuracy_All = np.mean(scores_All) * 100
print("Точность с исп. ф-ции cross_val_score на всем массиве {0:.1f}%".format(average_accuracy_All))
```

Точность с исп. ф-ции cross_val_score на обучающем массиве ДРУГАЯ! 94.7%
Точность с исп. ф-ции cross_val_score на всем массиве 96.7%

2.1 Классификация с использованием библиотеки scikit-learn

Задание6: пояснить, почему точности (cross_val_score) на обучающем массиве и на исходном массиве отличаются?

Задание7: выполнить отдельный пример валидации с использованием функции cross_val_score БЕЗ предварительного использования train_test_split

Пример работы классификатора

Установить: длина ч. 5 см, ширина ч. 2.9 см, длина л. 1 см, ширина л. 0.2 см.

```
# Создать массив NumPy кол-во строк=кол-во samples, кол-во столбцов=кол-во features
X_new = np.array([[5, 2.9, 1, 0.2]])                    # Создаем массив 1строка!!! X 4столбца
print("форма массива X_new: :\n {}".format(X_new.shape))
форма массива X_new:
(1, 4)

# Для классификации вызываем ф-цию predict объекта knn
klassifik = knn.predict(X_new)
print("Прогноз: {}".format(klassifik))
print("Классификация ввода (метка класса): {}".format(iris_dataset['target_names'][klassifik]))
Прогноз: [0]
Классификация ввода (метка класса): ['setosa']
```

2.1 Классификация с использованием библиотеки scikit-learn

Полный код классификатора

```
import numpy as np
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
iris_dataset = load_iris()
X_train, X_test, y_train, y_test = train_test_split(iris_dataset['data'], iris_dataset['target'], random_state=0)
knn = KNeighborsClassifier(n_neighbors=1)
knn.fit(X_train, y_train)
print("Правильность на тестовом наборе: {:.2f}".format(knn.score(X_test, y_test)))
```

Правильность на тестовом наборе: 0.97

2.1 Классификация с использованием библиотеки scikit-learn

Нормализация данных

Пример

```
# Пример для пояснения нормализации данных
X_broken = np.array(X)                          # создаем массив новый
X_broken[:,0] /= 100
X_broken[:,2] *= 100
X_broken[:,3] /= 1000
print(X_broken[:5])
```

```
[[5.1e-02 3.5e+00 1.4e+02 2.0e-04]
 [4.9e-02 3.0e+00 1.4e+02 2.0e-04]
 [4.7e-02 3.2e+00 1.3e+02 2.0e-04]
 [4.6e-02 3.1e+00 1.5e+02 2.0e-04]
 [5.0e-02 3.6e+00 1.4e+02 2.0e-04]]
```

```
# показываем, что точность ухудшилась
from sklearn.model_selection import cross_val_score
estimator = KNeighborsClassifier(n_neighbors=1)
broken_scores = cross_val_score(estimator, X_broken, y, scoring='accuracy', cv=3)  # исп. сразу cross_val_score
print(" Точность была 97%, а для 'перекошенных' данных УХУДШИЛАСЬ : {0:.1f}%".format(np.mean(broken_scores) * 100))
```

```
Точность была 97%, а для 'перекошенных' данных УХУДШИЛАСЬ: 92.8%
```

2.1 Классификация с использованием библиотеки scikit-learn

Нормализация данных

Нормализация выполняется функцией fit_transform (не меняет shape исходного массива) из модуля MinMaxScaler библиотеки sklearn.preprocessing

```
# нормализация [0;1] "перекошенных" данных (оч.удобно, когда и отрицат. и положит. числа)
from sklearn.preprocessing import MinMaxScaler
X_transformed = MinMaxScaler().fit_transform(X_broken) estimator = KNeighborsClassifier()
transformed_scores = cross_val_score(estimator, X_transformed, y, scoring='accuracy')
print(" Точность для нормализ данных: {0:.1f}%".format(np.mean(transformed_scores) * 100))
```

Точность для нормализ данных: 95.4%

Задание8: показать, что в новом массиве [0;1]

Задание9: выполнить отдельный пример, где задать массив чисел [-5;+10] с шагом 0.5. Выполнить нормализацию. Какое исходное число стало равно 0? Какое – 1?

Задание10: пояснить нормализацию с использованием функций

- sklearn.preprocessing.Normalizer
- sklearn.preprocessing.StandardScaler
- sklearn.preprocessing.Binarizer

2. Метод ближайших соседей в задаче классификации

2.2 OneR (One Rule) алгоритм классификации

Реализуется алгоритм:

-расчет м.о. для каждого признака (feature) ;

-для каждого признака используется пороговая функция переключения для сравнения его с м.о., результатом применения которой является бинаризация каждого признака в массиве;

Т.о. осуществляется переход от continuous features к categorical features

-для каждого нового ввода (предъявляемого для классификации цветка) классификатор укажет (predict) тот класс, для которого наибольшее количество признаков =1;

-т.к. классов четыре, может возникнуть случай, когда два класса =0, и два класса =1. В этом случае для каждого признака:

-находим тот класс, где такое же значение признака чаще всего встречается

-считаем ошибку подобного отнесения к данному классу по данному признаку;

-то же выполняем относительно инверсного значения признака;

-считаем суммарные ошибки классификации по прямым и по инверсным значениям признаков;

-выбираем ответ по минимальной суммарной ошибке.

Задание 1 по теме 2

Загрузить код OneR алгоритма из файла «ex2_iris_OneR_code.ipynb».

При необходимости выполнить коррекцию.

Выходные значения кода (для проверки выполнения) приведены ниже

Пояснить код в форме:

 # комментарий по каждой строке, по действиям, по переменным и т.п.

Пояснить выходные результаты

```
There are (112,) training samples
There are (38,) testing samples
The best model is based on variable 2 and has error 37.00
{'variable': 2, 'predictor': {0: 0, 1: 2}}

[0 0 0 2 2 2 0 2 0 2 2 0 2 2 0 2 0 2 2 2 0 0 0 2 0 2 0 2 2 0 0 0 2 0 2 0 2
 2]

The test accuracy is 65.8%

precision    recall  f1-score   support

           0       0.94      1.00      0.97        17
           1       0.00      0.00      0.00        13
           2       0.40      1.00      0.57         8

    accuracy                           0.66        38
   macro avg       0.45      0.67      0.51        38
weighted avg       0.51      0.66      0.55        38
```

Задание 2 по теме 2

Загрузить матрицу из файла «ex2.data», csv-формат, где:

кол-во строк = кол-ву samples;
первые 5 столбцов – continuous variables;
6й столбец - categorical variable, задающий классы A, B;

Реализовать OneR алгоритм.

Выполнить его кросс-валидацию.

Пояснить результаты тестирования.

Задание 3 по теме 2

Загрузить файлы «ionosphere.data», «ionosphere.names»

-выполнить анализ качества данных в Excel; пояснить различия (на графиках, вычислениями или т. п.); попытаться классифицировать данные «вручную»;

-вывести на печать типы и формы (shape) массивов X, y. Печать первых 2-х строк (X с округлением до 0.0);

-используя train_test_split разделить исходный массив данных на обучающие и тестовые выборки. Сколько записей в каждом наборе (вывести на печать .shape)?

для однообразия задать random_state=14

-используя KNeighborsClassifier() (вывести на печать параметры) выполнить обучение fit классификатора;

-выполнить валидацию predict классификатора, оценить его точность **A1** сравнением реальных значений тестовой выборки и предсказанных классификатором (вывести на печать в % с округл. до 0.0);

-используя модуль cross_val_score библиотеки scikit-learn рассчитать среднюю точность **A2** на тестовой выборке, вывести на печать, сравнить **A1** и **A2**;

-выполнить настройку классификатора – выбрать количество соседей n, обеспечивающее максимальную точность классификатора на тестовых данных;

-«перекосить данные» - разделить каждый 2й признак на 10. Подтвердить ухудшение точности на «перекошенных данных». Выполнить нормализацию, проверить улучшение точности.

Задание 3 по теме 2 help

В файле data представлены данные измерений от комплекса радаров Goose Bay, Labrador.
Комплекс включает 17 ФАР.
Наблюдение осуществляется за свободными электронами ионосферы.

В массиве данных 351 строка (samples) × 35 столбцов (features):

-первые 34 признака типа continuous; представляют собой значения измерений (количество пучков импульсов в единицу времени) от 17 антенн × 2 канала измерений от каждой антенны = 34 величины;

-35й признак типа categorical; имеет 2 значения - 'g' (хорошо) или 'b (плохо). "Good" измерения, когда регистрируются некоторые структуры в ионосфере, "Bad" – не регистрируются.

Формат массива данных *.CSV (Comma-Separated Values)

```
import numpy as np
import csv
X = np.zeros((351, 34), dtype='float')
y = np.zeros((351,), dtype='bool')
data_filename = (r"C:\Users\User\2019-20\dannye\ionosphere.data")
with open(data_filename, 'r') as csvfile:
    reader = csv.reader(csvfile)
    for i, row in enumerate(reader):                    # цикл по строкам ф-цией enumerate
        data = [float(datum) for datum in row[:-1]]     # переводим 34 переменные строки в формат float
        X[i] = data                                     # и сохраняем в матрице X
        y[i] = row[-1] == 'g'       # для последней переменной в строке (35й_categorical) - True (1) если 'g', F->'b', сохр. в y
```

Задание 3 по теме 2 help

Задать n (количество соседей) от 1 до 20, рассчитать точности (в т.ч. средние значения)

```
avg_scores = []
all_scores = []
parameter_values = list(range(1, 21))    #от 1 до 21 = всего 20
for n_neighbors in parameter_values:
    estimator = KNeighborsClassifier(n_neighbors=n_neighbors)
    scores = cross_val_score(estimator, X, y, scoring='accuracy', cv=3)
    avg_scores.append(np.mean(scores))
```

и построить график «средние точности (n)». Какая максимальная точность достижима за счет выбора n? - выбрать n.

Летучки по теме 2

Пояснить метрики k ближайших соседей

Пояснить, почему точности обучающей и тестовой выборок, рассчитанные функцией score, обычно отличаются.

Пояснить процедуры нормализации библиотеки sklearn