Лабораторная работа №4

Работа с классификацией данных при помощи skikit-learn. Построение модели машинного обучения методом k-средних.

Оглавление

[Ход работы: 2](#_Toc146484799)

[Постановка задачи 2](#_Toc146484800)

[Загружаем и изучаем данные 2](#_Toc146484801)

[Метрики эффективности: обучающий и тестовые наборы 5](#_Toc146484802)

[Исследование данных. Визуализация 7](#_Toc146484803)

[Построение модели методом k-ближайших средних 9](#_Toc146484804)

[Получение прогнозов 10](#_Toc146484805)

[Оценка качества модели 11](#_Toc146484806)

[Обобщение 12](#_Toc146484807)

[Задание: 13](#_Toc146484808)

[Постройте модели классификации по методу k-ближайших средних. 13](#_Toc146484809)

[ **load\_digits**(\*[, n\_class, return\_X\_y, as\_frame]) - Оптическое распознавание набора данных рукописных цифр 13](#_Toc146484810)

[ **load\_wine**(\*[, return\_X\_y, as\_frame]) - Набор данных по распознаванию вин 13](#_Toc146484811)

[ **load\_breast\_cancer**(\*[, return\_X\_y, as\_frame]) - Набор данных по раку груди висконсин (диагностический) 13](#_Toc146484812)

**Цель работы**: Применить библиотеку sсikit-learn для построения простой модели классификации по методу k-ближайших средних. Изучить полученные данные, научиться оценивать качество моделей.

# Ход работы:

## Постановка задачи[[1]](#footnote-1)

Рассмотрим пример построения модели классификации сортов ирисов.

Получены несколько характеристик: длина и ширина лепестков, а также длина и ширина чашелистиков.

Причем ранее есть измерения этих же характеристик ирисов, которые ранее позволили отнести их к сортам setosa, versicolor и virginica.

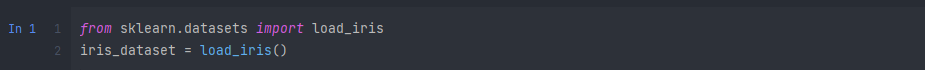
Наша цель заключается в построении модели машинного обучения, которая сможет обучиться на основе характеристик ирисов, уже классифицированных по сортам, и затем предскажет сорт для нового цветка ириса.

Поскольку у нас есть примеры, по которых мы уже знаем правильные сорта ириса, решаемая задача является задачей обучения с учителем. В этой задаче нам нужно спрогнозировать один из сортов ириса. Это пример задачи классификации (classification). Возможные ответы (различные сорта ириса) называются классами (classes). Каждый ирис в наборе данных принадлежит к одному из трех классов, таким образом решаемая задача является задачей трехклассовой классификации. Ответом для отдельной точки данных (ириса) является тот или иной сорт этого цветка. Cорт, к которому принадлежит цветок (конкретная точка данных), называется меткой (label).

## Загрузка и изучение данные

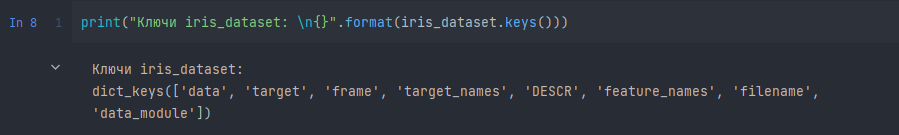
Данные, которые мы будем использовать для этого примера, – это набор данных Iris, классический набор данных в машинном обучении и статистике. Он уже включен в модуль datasets библиотеки scikit-learn.

Мы можем загрузить его, вызвав функцию load\_iris:



Объект iris, возвращаемый load\_iris, является объектом Bunch,

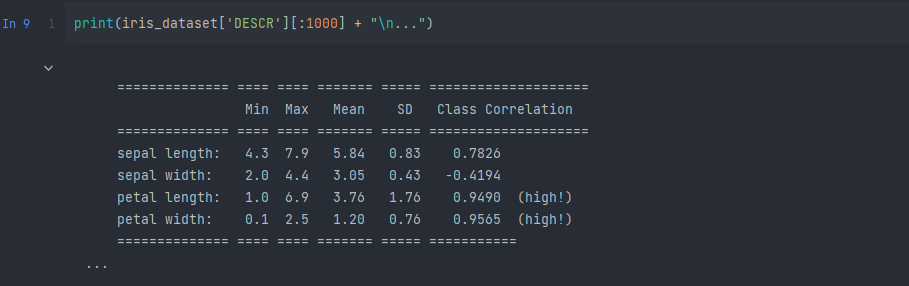
который очень похож на словарь. Он содержит ключи и значения:



Значение ключа DESCR – это краткое описание набора данных. Здесь

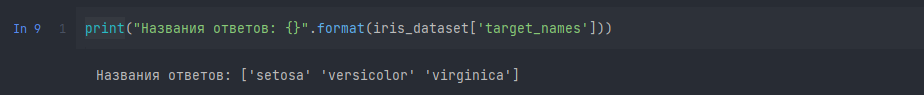
мы покажем начало описания (оставшуюся часть описания вы можете

посмотреть самостоятельно):

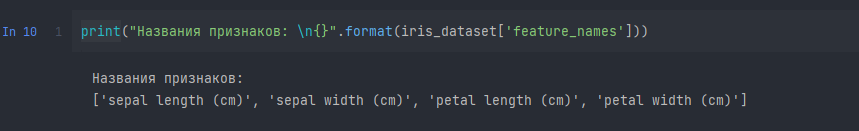


[: 1000] – указывает количество выводимых символов

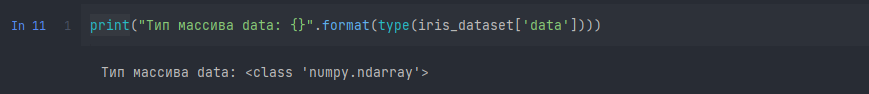
Значение ключа target\_names – это массив строк, содержащий сорта цветов, которые мы хотим предсказать:



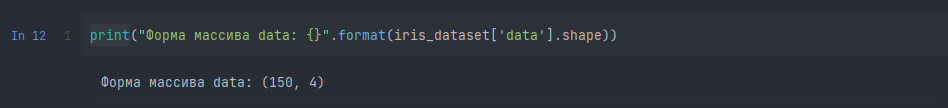
Значение feature\_names – это список строк с описанием каждого признака:



Сами данные записаны в массивах target и data. data – массив NumPy, который содержит количественные измерения длины чашелистиков, ширины чашелистиков, длины лепестков и ширины лепестков:

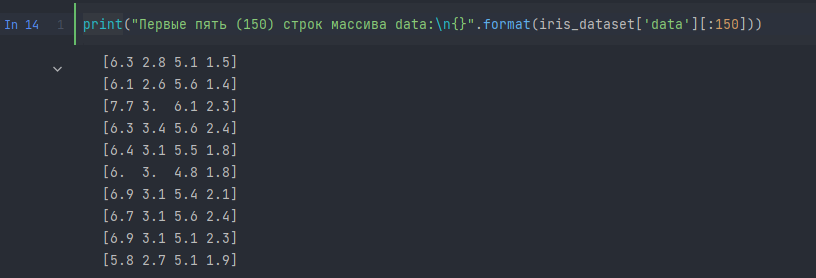


Строки в массиве data соответствуют цветам ириса, а столбцы представляют собой четыре признака, которые были измерены для каждого цветка:



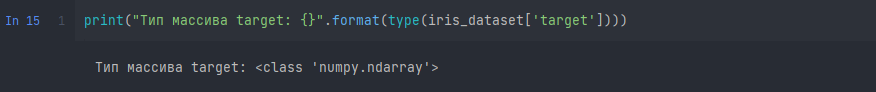
Мы видим, что массив содержит измерения для 150 различных цветов по 4 признакам. Вспомним, что в машинном обучении отдельные элементы называются примерами (samples), а их свойства – характеристиками или признаками (feature). Форма (shape) массива данных определяется количеством примеров, умноженным на количество признаков. Это является общепринятым соглашением в scikit-learn, и ваши данные всегда будут представлены в этой форме.

Ниже приведены значения признаков для первых пяти (150) примеров:

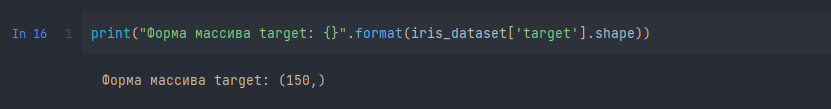


Взглянув на эти данные, мы видим, что все пять цветов имеют ширину лепестка 0.2 см и первый цветок имеет самую большую длину чашелистика, 5.1 см.

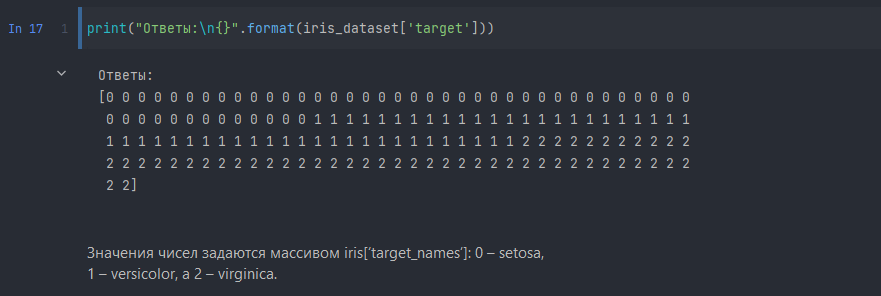
Массив target содержит сорта уже измеренных цветов, тоже записанные в виде массива NumPy:



target представляет собой одномерный массив, по одному элементу для каждого цветка:



Сорта кодируются как целые числа от 0 до 2:



## Метрики эффективности: обучающий и тестовые наборы

На основе этих данных нам нужно построить модель машинного обучения, которая предскажет сорта ириса для нового набора измерений.

Но прежде, чем мы применить нашу модель к новому набору, мы должны убедиться в том, что модель на самом деле работает и ее прогнозам можно доверять.

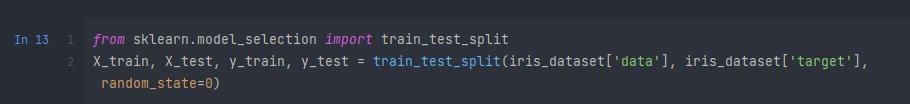
К сожалению, для оценки качества модели мы не можем использовать данные, которые были взяты нами для построения модели. Это обусловлено тем, что наша модель просто запомнит весь обучающий набор и поэтому она всегда будет предсказывать правильную метку для любой точки данных в обучающем наборе. Это «запоминание» ничего не говорит нам об обобщающей способности модели (другими словами, мы не знаем, будет ли эта модель так же хорошо работать на новых данных).

Для оценки эффективности модели, мы предъявляем ей новые размеченные данные (размеченные данные, которые она не видела раньше). Обычно это делается путем разбиения собранных размеченных данных (в данном случае 150 цветов) на две части. Одна часть данных используется для построения нашей модели машинного обучения и называется обучающими данными (training data) или обучающим набором (training set). Остальные данные будут использованы для оценки качества модели, их называют тестовыми данными (test data), тестовым набором (test set) или контрольным набором (hold-out set).

В библиотеке scikit-learn есть функция **train\_test\_split**, которая перемешивает набор данных и разбивает его на две части. **Эта функция отбирает в обучающий набор 75% строк данных с соответствующими метками. Оставшиеся 25% данных с метками объявляются тестовым набором.** Вопрос о том, сколько данных отбирать в обучающий и тестовый наборы, является дискуссионным, однако использование естового набора, содержащего 25% данных, является хорошим правилом.

В scikit-learn данные, как правило, обозначаются заглавной X, тогда как метки обозначаются строчной у. Это навеяно стандартной математической формулой f(х)=у, где х является аргументом функции, а у – выводом. В соответствии с некоторыми математическими соглашениями мы используем заглавную X, потому что данные представляют собой двумерный массив (матрицу) и строчную у, потому что целевая переменная – это одномерный массив (вектор).

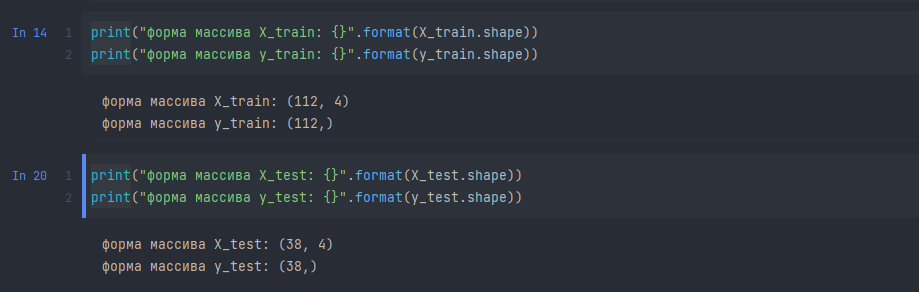
Давайте вызовем функцию train\_test\_split для наших данных и зададим обучающие данные, обучающие метки, тестовые данные, тестовые метки, используя вышеупомянутые буквы:\_\_



Перед разбиением функция train\_test\_split перемешивает набор данных с помощью генератора псевдослучайных чисел. Если мы просто возьмем последние 25% наблюдений в качестве тестового набора, все точки данных будет иметь метку 2, поскольку все точки данных отсортированы по меткам (смотрите вывод для iris['target'], показанный ранее). Используя тестовый набор, содержащий только один из трех классов, вы не сможете объективно судить об обобщающей способности модели, таким образом, мы перемешиваем наши данные, чтобы тестовые данные содержали все три класса.

Чтобы в точности повторно воспроизвести полученный результат, мы воспользуемся генератором псевдослучайных чисел с фиксированным стартовым значением, которое задается с помощью параметра random\_state. Это позволит сделать результат воспроизводим, поэтому вышеприведенный программный код будет генерировать один и тот же результат. Будем задавать random\_state при использовании рандомизированных процедур.

Выводом функции train\_test\_split являются X\_train, X\_test, y\_train и y\_test, которые все являются массивами Numpy. X\_train содержит 75% строк набора данных, а X\_test содержит оставшиеся 25%:



## Исследование данных. Визуализация

Перед тем как строить модель машинного обучения, неплохо было бы исследовать данные, чтобы понять, можно ли легко решить поставленную задачу без машинного обучения или содержится ли нужная информация

в данных.

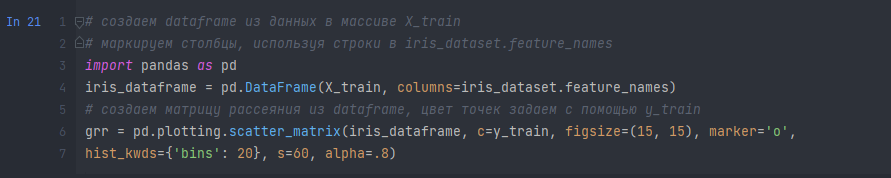
Кроме того, исследование данных – это хороший способ обнаружить аномалии и особенности. Например, вполне возможно, что некоторые из ваших ирисов измерены в дюймах, а не в сантиметрах. В реальном мире

нестыковки в данных и неожиданности очень распространены. Один из лучших способов исследовать данные – визуализировать их. Это можно сделать, используя диаграмму рассеяния (scatter plot). В диаграмме рассеяния один признак откладывается по оси х, а другой признак – по оси у, каждому наблюдению соответствует точка. К сожалению, экран компьютера имеют только два измерения, что позволяет разместить на графике только два (или, возможно, три) признака одновременно. Таким образом, трудно разместить на графике наборы данных с более чем тремя признаками. Один из способов решения этой проблемы – построить матрицу диаграмм рассеяния (scatterplot matrix) или парные диаграммы рассеяния (pair plots), на которых будут изображены все возможные пары признаков. Если у вас

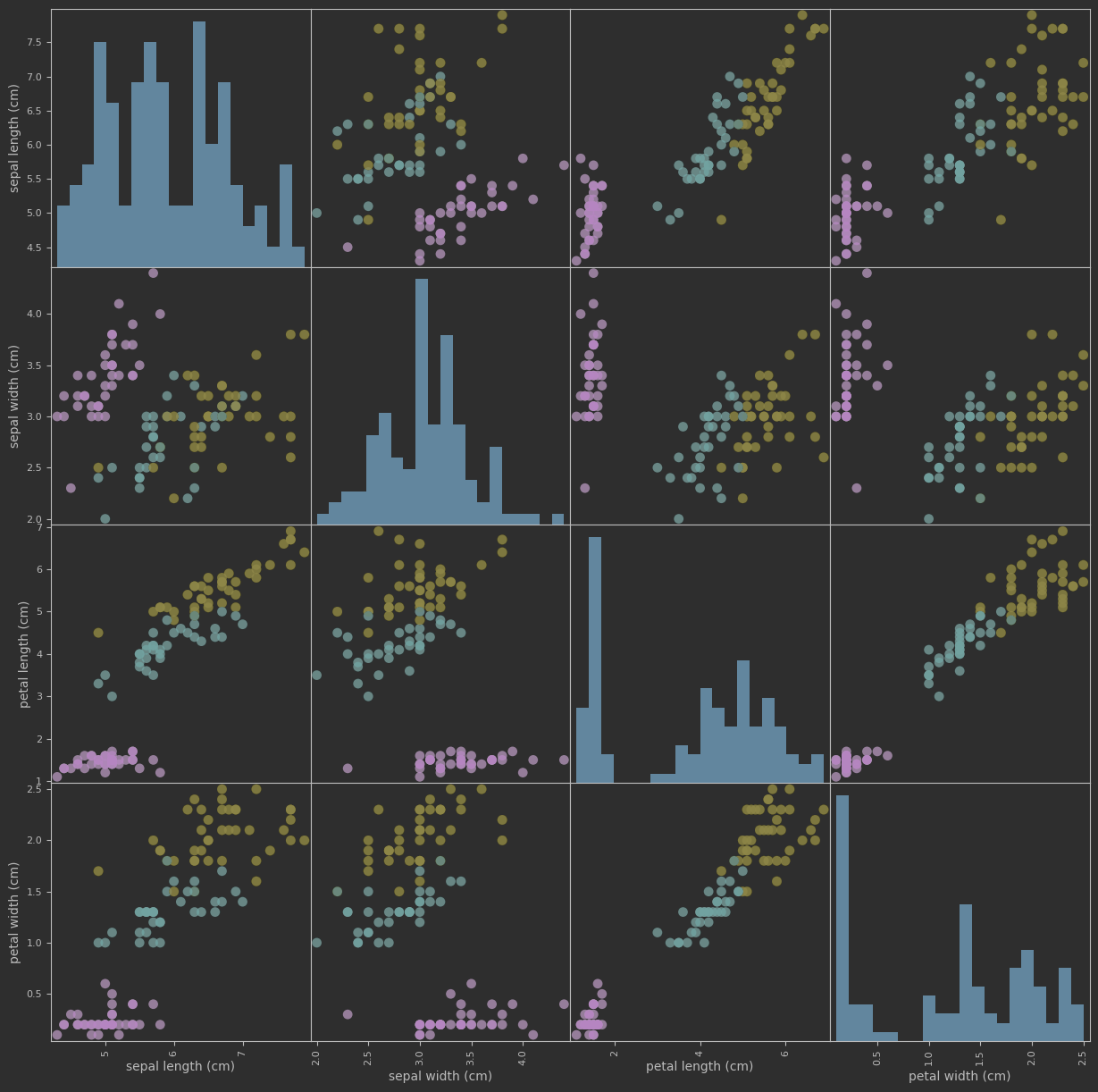
есть небольшое число признаков, например, четыре, как здесь, то использование матрицы диаграмм рассеяния будет вполне разумным. Однако, вы должны помнить, что матрица диаграмм рассеяния не показывает взаимодействие между всеми признаками сразу, поэтому некоторые интересные аспекты данных не будут выявлены с помощью этих графиков.

Рис. представляет собой матрицу диаграмм рассеяния для признаков обучающего набора. Точки данных окрашены в соответствии с сортами ириса, к которым они относятся. Чтобы построить диаграммы, мы сначала преобразовываем массив NumPy в DataFrame (основный тип данных в библиотеке pandas). В pandas есть функция для создания парных диаграмм рассеяния под названием scatter\_matrix. По

диагонали этой матрицы располагаются гистограммы каждого признака:



Результат построения:



Взглянув на график, мы можем увидеть, что, похоже, измерения чашелистиков и лепестков позволяют относительно хорошо разделить три класса. Это означает, что модель машинного обучения, вероятно, сможет научиться разделять их.

## Построение модели методом k-ближайших средних

В библиотеке scikit-learn имеется довольно много алгоритмов классификации, которые мы могли бы использовать для построения модели. В данном примере мы будем использовать классификатор на основе метода k ближайших соседей, который легко интерпретировать.

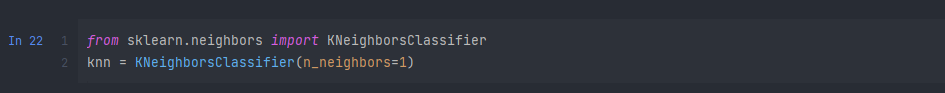
Построение этой модели заключается лишь в запоминании обучающего набора. Для того, чтобы сделать прогноз для новой точки данных, алгоритм находит точку в обучающем наборе, которая находится ближе всего к новой точке. Затем он присваивает метку, принадлежащую этой точке обучающего набора, новой точке данных.

k в методе k ближайших соседей означает, что вместо того, чтобы использовать лишь ближайшего соседа новой точки данных, мы в ходе обучения можем рассмотреть любое фиксированное число (k) соседей (например, рассмотреть ближайшие три или пять соседей). Тогда мы можем сделать прогноз для точки данных, используя класс, которому принадлежит большинство ее соседей. Подробнее мы поговорим об этом

в главе 2, а в данный момент мы будем использовать только одного соседа.

В scikit-learn все модели машинного обучения реализованы в собственных классах, называемых классами Estimator. Алгоритм классификации на основе метода k ближайших соседей реализован в классификаторе KNeighborsClassifier модуля neighbors. Прежде чем использовать эту модель, нам нужно создать объект-экземпляр класса.

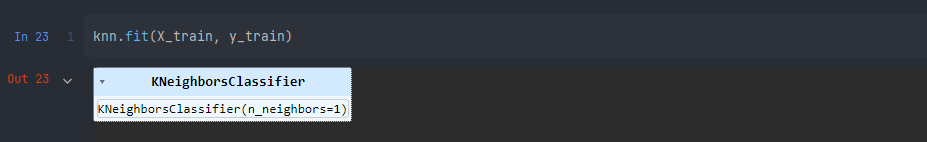
Это произойдет, когда мы зададим параметры модели. Самым важным параметром KNeighborsClassifier является количество соседей, которые установим равным 1:



Объект knn включает в себя алгоритм, который будет использоваться для построения модели на обучающих данных, а также алгоритм, который сгенерирует прогнозы для новых точек данных. Он также будет содержать информацию, которую алгоритм извлек из обучающих данных. В случае с KNeighborsClassifier он будет просто хранить

обучающий набор.

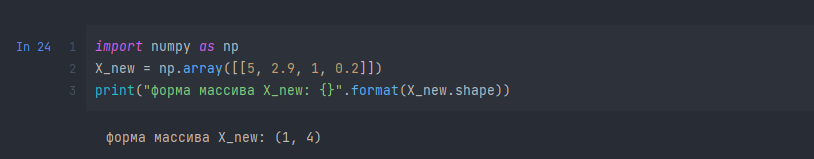
Для построения модели на обучающем наборе, мы вызываем метод fit объекта knn, который принимает в качестве аргументов массив NumPy X\_train, содержащий обучающие данные, и массив NumPy y\_train, соответствующий обучающим меткам:



Метод fit возвращает сам объект knn (и изменяет его), таким образом, мы получаем строковое представление нашего классификатора. Оно показывает нам, какие параметры были использованы при создании модели. Почти все параметры имеют значения по умолчанию, но вы также можете обнаружить параметр n\_neighbor=1, заданный нами.

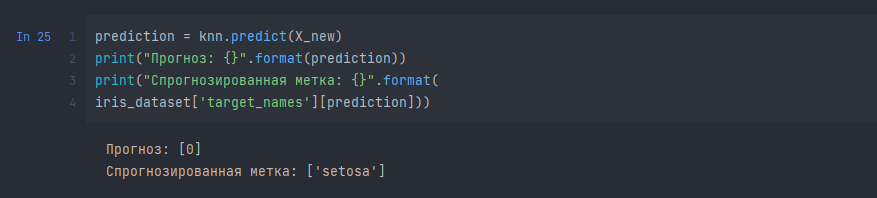
## Получение прогнозов

Теперь мы можем получить прогнозы, применив эту модель к новым данным, по которым мы еще не знаем правильные метки. Представьте, что мы нашли в дикой природе ирис с длиной чашелистика 5 см, шириной чашелистика 2.9 см, длиной лепестка 1 см и шириной лепестка 0.2 см. К какому сорту ириса нужно отнести этот цветок? Мы можем поместить эти данные в массив NumPy, снова вычисляя форму массива, т.е. количество примеров (1), умноженное на количество признаков (4):



Обратите внимание, что мы записали измерения по одному цветку в двумерный массив NumPy, поскольку scikit-learn работает с двумерными массивами данных.

Чтобы сделать прогноз, мы вызываем метод predict объекта knn:

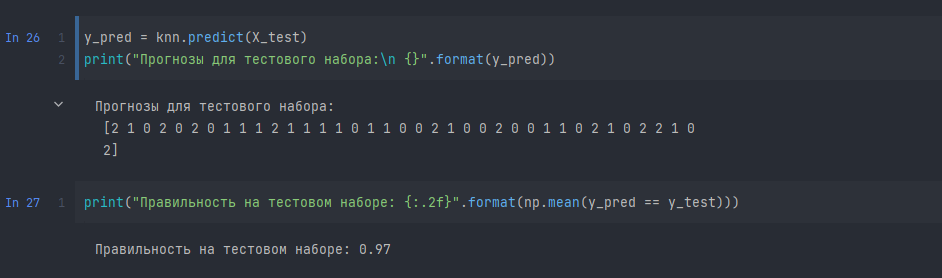


Наша модель предсказывает, что этот новый цветок ириса принадлежит к классу 0, что означает сорт setosa. Но как узнать, можем ли мы доверять нашей модели? Правильный сорт ириса для этого примера нам неизвестен, а ведь именно получение правильных прогнозов и является главной задачей построения модели!

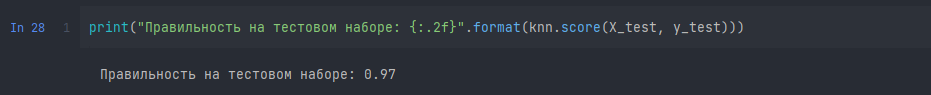
## Оценка качества модели

Теперь нам понадобится созданный ранее тестовый набор. Эти данные не использовались для построения модели, но мы знаем правильные сорта для каждого ириса в тестовом наборе.

Таким образом, мы можем сделать прогноз для каждого ириса в тестовом наборе и сравнить его с фактической меткой (уже известным сортом). Мы можем оценить качество модели, вычислив правильность (accuracy) – процент цветов, для которых модель правильно спрогнозировала сорта:



Кроме того, мы можем использовать метод score объекта knn, который вычисляет правильность модели для тестового набора:



Правильность этой модели для тестового набора составляет около 0.97, что означает, что мы дали правильный прогноз для 97% ирисов в тестовом наборе. При некоторых математических допущениях, это означает, что мы можем ожидать, что наша модель в 97% случаев даст правильный прогноз для новых ирисов.

## Обобщение

В результате была решена задача обучения с учителем.

Было три возможных сорта ирисов – setosa, versicolor и virginica, что делало нашу задачу задачей 3-классовой классификации. В задаче классификации возможные сорта ирисов называются классами (classes) а сами названия сортов – метками (labels).

Набор данных Iris состоит из двух массивов NumPy: один содержит данные и в scikit-learn обозначается как X, другой содержит правильные или нужные ответы и обозначается как y. Массив Х представляет собой двумерный массив признаков, в котором одна строка соответствует одной точке данных, а один столбец – одному признаку.

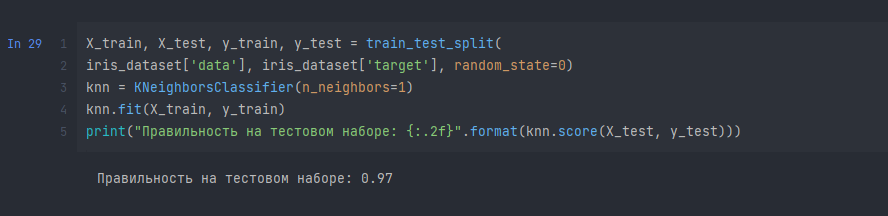
Массив у представляет собой одномерный массив, который для каждого примера содержит метку класса, целое число от 0 до 2.

Мы разделили наш набор данных на обучающий набор (training set), чтобы построить нашу модель, а также тестовый набор (test set), чтобы оценить, насколько хорошо наша модель будет классифицировать новые, ранее незнакомые ей данные.

Мы выбрали алгоритм классификации k ближайших соседей, который генерирует прогноз для новой точки данных, рассматривая ее ближайшего соседа(ей) в обучающем наборе. Все это реализовано в классе KNeighborsClassifier, который содержит алгоритм, строящий модель, а также алгоритм, который дает прогнозы, используя построенную модель. Мы создали объект-экземпляр класса, задав параметры. Затем мы построили модель, вызвав метод fit и передав обучающие данные (X\_train) и обучающие ответы (y\_train) в качестве параметров. Мы оценили качество модели с использованием метода score, который вычисляет правильность модели. Мы применили метод score к тестовым данным и тестовым ответам и обнаружили, что наша модель демонстрирует правильность около 97%. Это означает, что модель выдает правильные прогнозы для 97% наблюдений тестового набора.

Это убедило нас в том, что модель можно применить к новым данным (в нашем примере это измерения характеристик новых цветов), и мы надеемся, что эта модель даст правильные прогнозы в 97% случаев.

Ниже приводится краткое изложение программного кода, необходимого для всей процедуры обучения и оценки модели:



Этот фрагмент содержит базовый код, необходимый для применения любого алгоритма машинного обучения с помощью scikit-learn.

Методы **fit**, **predict** и **score** являются общими для моделей контролируемого обучения в scikit-learn и, используют принципы, которые можно применять для решения различных задач машинного обучения.

# Задание:

### Постройте модели классификации по методу k-ближайших средних.

* **Загрузите** [**датасеты**](https://scikit-learn.ru/7-1-toy-datasets/#:~:text=%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%20Bunch%20%E2%80%94%20%D1%8D%D1%82%D0%BE%20%D1%81%D0%BB%D0%BE%D0%B2%D0%B0%D1%80%D1%8C%2C%20%D0%BA%D0%BE%D1%82%D0%BE%D1%80%D1%8B%D0%B9,%D0%B4%D0%BB%D1%8F%20return_X_y%20%D0%BF%D0%B0%D1%80%D0%B0%D0%BC%D0%B5%D1%82%D1%80%D0%B0%20%D0%B7%D0%BD%D0%B0%D1%87%D0%B5%D0%BD%D0%B8%D0%B5%20True%20.) **классификации из библиотеки skikit-learn:**

## **[load\_digits](https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_digits.html" \l "sklearn.datasets.load_digits" \t "_blank)**(\*[, n\_class, return\_X\_y, as\_frame]) - Оптическое распознавание набора данных рукописных цифр

## **[load\_wine](https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_wine.html" \l "sklearn.datasets.load_wine" \t "_blank)**(\*[, return\_X\_y, as\_frame]) - Набор данных по распознаванию вин

## **[load\_breast\_cancer](https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_breast_cancer.html" \l "sklearn.datasets.load_breast_cancer" \t "_blank)**(\*[, return\_X\_y, as\_frame]) - Набор данных по раку груди висконсин (диагностический)

**Опираясь на приведенные выше примеры, выполните следующие задачи, указанными выше датасетами:**

* Изучите данные
* Разделите данные на обучающий и тестовые наборы
* Исследуйте данные с помощью визуализации. Подберите нужные параметры графиков
* Постройте модель по методу k-ближайших средних. Протестируйте на произвольном наборе данных
* Получите прогноз модели
* Оцените качество модели
* **Сделайте выводы по качеству моделей**

1. Andreas C. Mueller and Sarah Guido. Introduction to Machine Learning with Python. O’Reilly 2016 [↑](#footnote-ref-1)