

Интеллектуальные системы и технологии

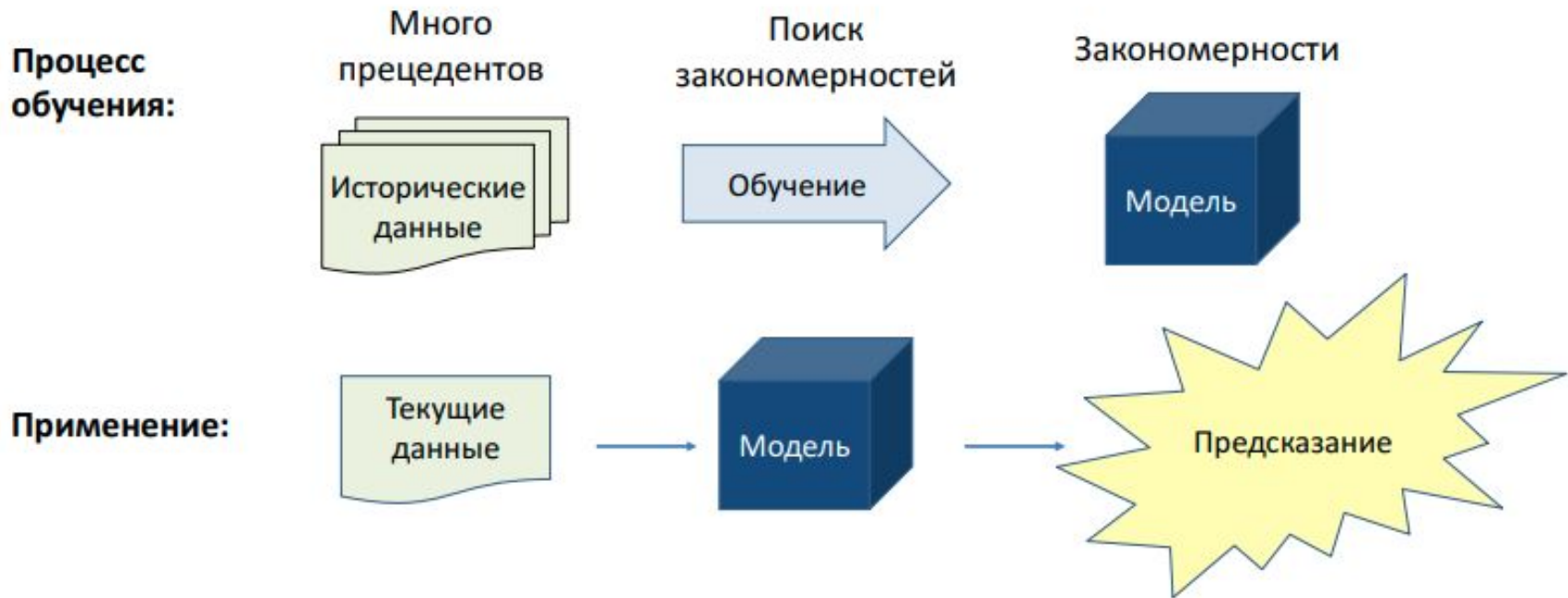


- Лекция 12.

KNN - классификатор

Машинное обучение

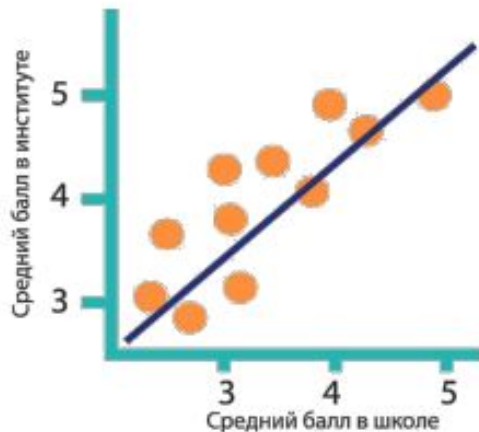
Машинное обучение – технологии автоматического обучения алгоритмов ИИ распознаванию и классификации на тестовых выборках объектов



Задачи машинного обучения

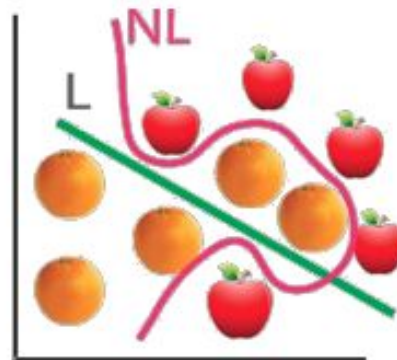
Регрессия

Установление аналитического выражения зависимости между исследуемыми признаками



Классификация

Построении моделей, выполняющих отнесение интересующего нас объекта к одному из нескольких известных классов



L – линейный классификатор
NL – нелинейный классификатор

Кластеризация

Разбиения всей исходной совокупности элементов на отдельные группы однородных объектов, сходных между собой, но имеющих отчетливые отличия этих групп друг от друга





Классификация

- В анализе данных — разбиение множества объектов или наблюдений на априорно заданные группы, называемые **классами**, внутри каждой из которых они предполагаются похожими друг на друга, имеющими примерно **одинаковые свойства и признаки**.



Класс

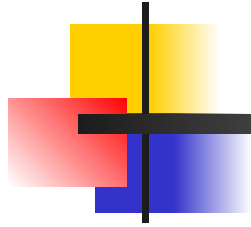
- В статистике и анализе данных классом называют группу объектов или явлений, обладающих общими свойствами.
- Например, среди заемщиков банка можно выделить классы добросовестных (которые не допускают просрочки) и недобросовестных (допускают). Также клиентов можно разбить на классы по уровню их активности (активный, пассивный) и т.д.



Виды классов

- **непересекающиеся** — одно наблюдение может одновременно принадлежать только одному классу
- **пересекающиеся** — одно и то же наблюдение может принадлежать нескольким классам одновременно
- **нечеткими** — наблюдение принадлежит к классу с некоторой степенью принадлежности (обычно степень принадлежности задается в интервале от 0 до 1).

Машинное обучение. Общая задача.



Дано:

- Набор наблюдений $X = \{x_i\}$
- Набор соответствующих им «результатов» $Y = \{y_i\}$

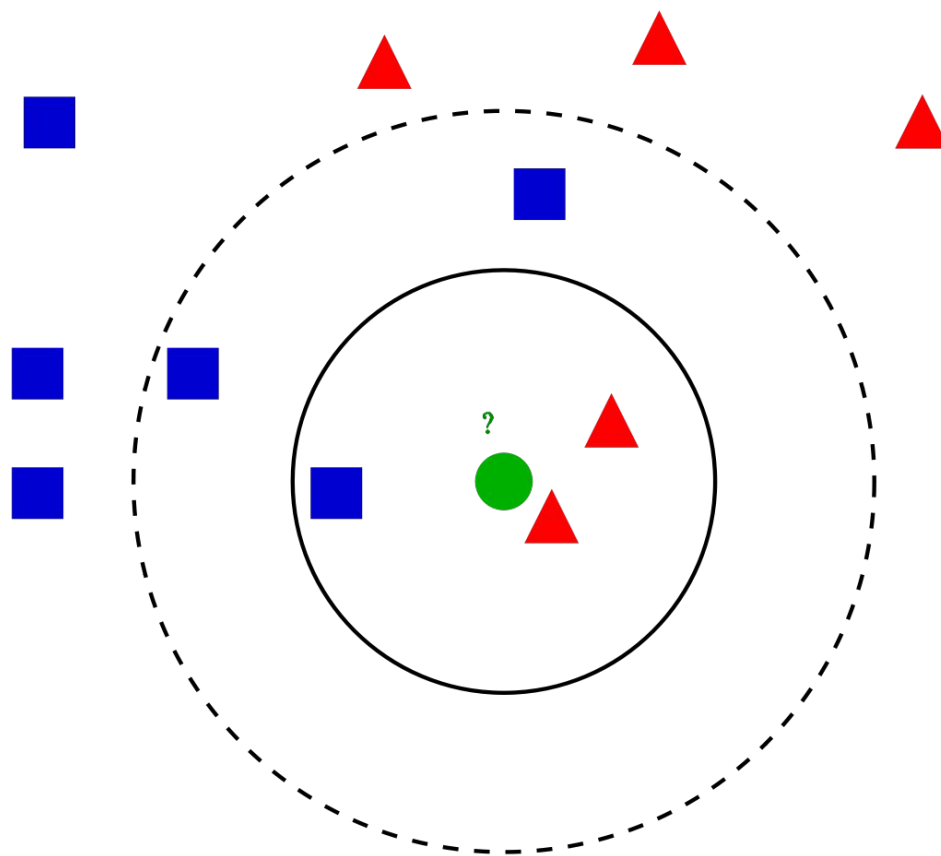
Надо:

- Автоматически найти функцию $f(x) = y$, «наилучшим» образом отражающую связь между X и Y

К ближайших соседей (k-Nearest Neighbors)

- Метод k-ближайших соседей используется для решения задачи классификации.
- Он относит объекты к классу, которому принадлежит большинство из k его ближайших соседей в многомерном пространстве признаков.
- Число k – это количество соседних объектов в пространстве признаков, которые сравниваются с классифицируемым объектом. Иными словами, если $k = 10$, то каждый объект сравнивается с 10-ю соседями.

К ближайших соседей





Алгоритм

- На первом шаге алгоритма следует задать число k – количество ближайших соседей.
- Если принять $k = 1$, то алгоритм потеряет обобщающую способность (то есть способность выдавать правильный результат для данных, не встречавшихся ранее в алгоритме) так как новой записи будет присвоен класс самой близкой к ней.



Алгоритм

- На втором шаге находятся k записей с минимальным расстоянием до вектора признаков нового объекта (поиск соседей).
- Для упорядоченных значений атрибутов находится Евклидово расстояние:

$$D_E = \sqrt{\sum_i^n (x_i - y_i)^2},$$

- где n – количество атрибутов.



Выделение значимых атрибутов

- При нахождении расстояния иногда учитывают значимость атрибутов. Она определяется экспертом или аналитиком субъективно, полагаясь на собственный опыт. В таком случае при нахождении расстояния каждый i -ый квадрат разности в сумме умножается на коэффициент Z_i . Например, если атрибут А в три раза важнее атрибута В ($Z_A = 3$, $Z_B = 1$), то расстояние будет находиться следующим образом:

$$D_E = \sqrt{3(x_A - y_A)^2 + (x_B - y_B)^2}.$$


- Подобный прием называют растяжением осей (stretching the axes), что позволяет снизить ошибку классификации.



Алгоритм

- На следующем шаге, когда найдены записи, наиболее похожие на новую, необходимо решить, как они влияют на класс новой записи. Для этого используется функция сочетания (combination function). Одним из основных вариантов такой функции является простое невзвешенное голосование (simple unweighted voting).

Простое невзвешенное голосование

- 
- Расстояние от каждой записи при голосовании здесь больше не играет роли.
 - Все имеют равные права в определении класса. Каждая имеющаяся запись голосует за класс, к которому принадлежит.
 - Новой записи присваивается класс, набравший наибольшее количество голосов.
 - Но что делать в случае, если несколько классов набрали равное количество голосов? Эту проблему снимает взвешенное голосование (weighted voting).



Взвешенное голосование

- Учитывается расстояние до новой записи. Чем меньше расстояние, тем более значимый вклад вносит голос. Голоса за класс находятся по следующей формуле:

$$votes(class) = \sum_{i=1}^n \frac{1}{d^2(X, Y_i)},$$

- где $d^2(X, Y_i)$ – квадрат расстояния от известной записи Y_i до новой X , n – количество известных записей класса, для которого рассчитываются голоса, $class$ - наименование класса.



Нормализация

- Разные атрибуты могут иметь разный диапазон представленных значений в выборке (например атрибут А представлен в диапазоне от 0.1 до 0,5, а атрибут Б представлен в диапазоне от 1000 до 5000), то значения дистанции могут сильно зависеть от атрибутов с большими диапазонами.
- **Нормализация** предполагает замену номинальных признаков так, чтобы каждый из них лежал в диапазоне от 0 до 1.
- Минимаксная нормализация:

$$X^* = \frac{X - \min(X)}{\max(X) - \min(X)},$$



Отбор признаков

- Важным при решении задачи является умение правильно отобрать и даже создать признаки.
- В англоязычной литературе это называется Feature Selection и Feature Engineering. В то время как Feature Engineering довольно творческий процесс и полагается больше на интуицию и экспертные знания, для Feature Selection есть уже большое количество готовых алгоритмов.



Данные

- Имеется набор данных, собранных Р. Фишером, о 150 цветках ириса трех классов: Iris Setosa, Iris Versicolour, Iris Virginica, по 50 записей для каждого. Для каждой записи известны:

- длина чашелистика
- ширина чашелистика
- длина лепестка
- ширина лепестка

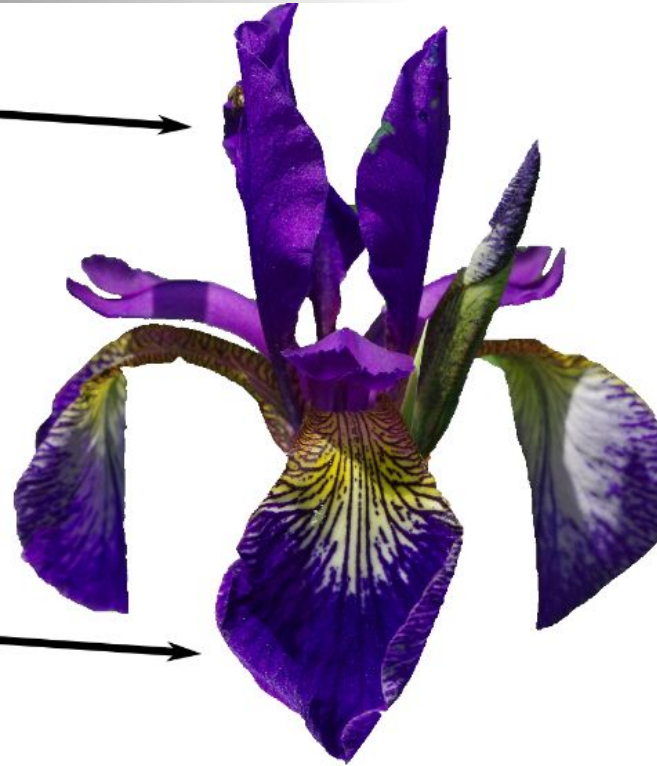
```
[[6.3, 2.7, 4.9, 1.8]  
[6.7, 3.3, 5.7, 2.1]  
[4.4, 3.2, 1.3, 0.2]  
[6, 3, 4.8, 1.8]  
[6.9, 3.2, 5.7, 2.3]  
[6.5, 2.8, 4.6, 1.5]  
[7.6, 3, 6.6, 2.1]  
[7.7, 2.6, 6.9, 2.3]  
[4.4, 2.9, 1.4, 0.2]  
[6.3 2.8 5.1 1.5]]
```

Ирис (Лепесток и Чашелистник)

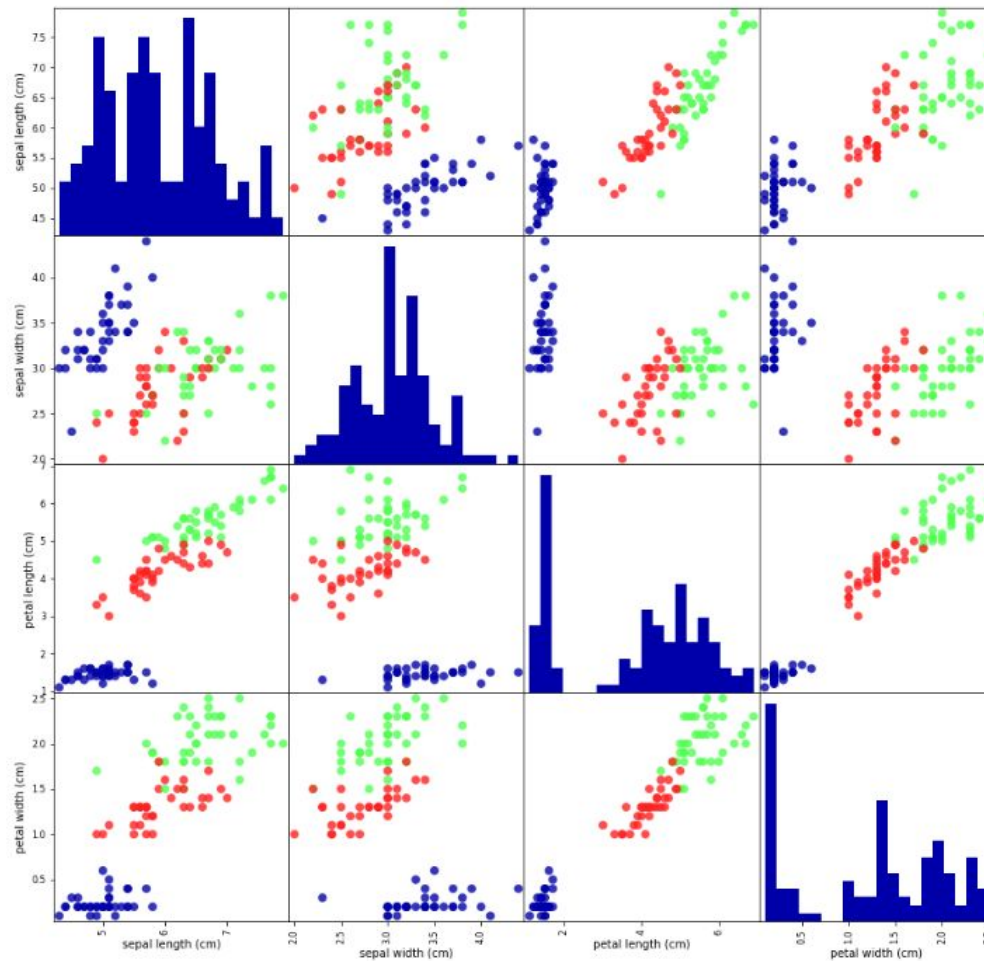
Petal



Sepal



Изучить данные





Цель

- Цель заключается в построении модели машинного обучения, которая сможет обучиться на основе характеристик ирисов, уже классифицированных по сортам, и затем предскажет сорт для любого нового цветка ириса.
- Поскольку у нас есть примеры, из которых мы уже знаем, как правильно определить сорт ириса, решаемая задача является задачей обучения с учителем.
- Нужно предсказать к какому сорту из трех относится конкретное растение ириса.

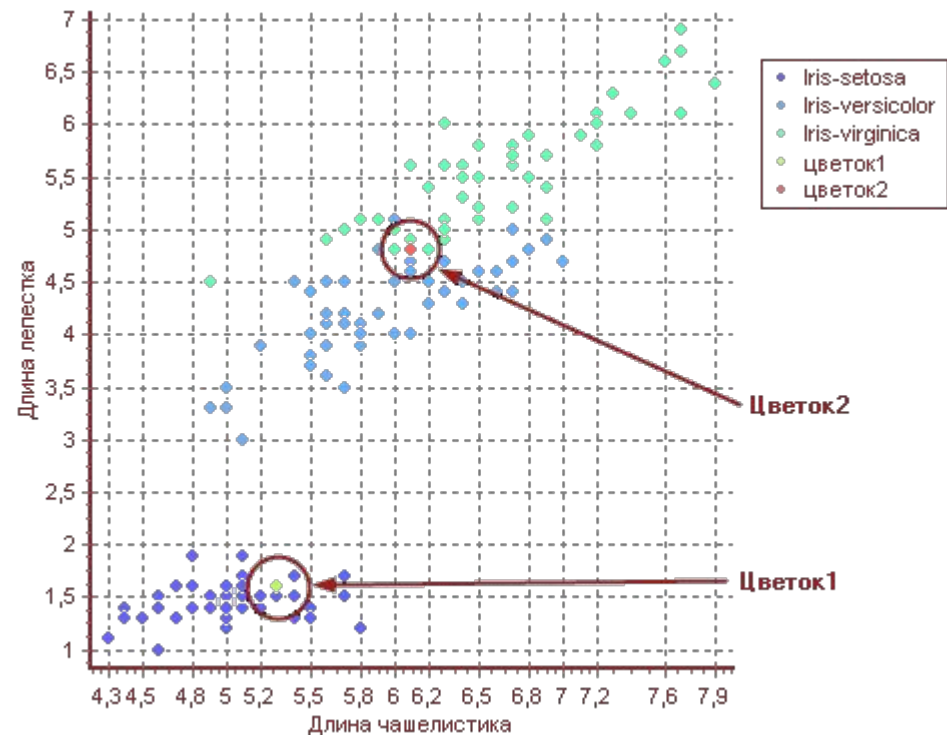


Задача классификации

- Это пример задачи классификации
- Возможные ответы (различные сорта риса) в машинном обучении называют 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 (растения ириса) является тот иной сорт этого цветка.
- Сорт, к которому принадлежит цветок, по терминологии МО называют меткой (label).

Пример применения KNN

- Для простоты рассмотрим только два входных поля: длина чашелистика и длина лепестка, а также выходное – класс цветка.
- Необходимо определить класс двух цветков со следующими значениями длины чашелистика и лепестка в сантиметрах: 5,3 и 1,6 (первый), 6,1 и 4,8 (второй).





Первый цветок

- Рассмотрим первый цветок. Установим значение $k = 3$. Таким образом, необходимо найти трех ближайших соседей. Для первого цветка это будут цветки со следующими значениями длины чашелистика и лепестка: $A(5,3; 1,5)$, $B(5,2; 1,5)$ и $C(5,2; 1,5)$. Покажем, чему равны расстояния до первого цветка (оба атрибута равнозначны):

$$d(\text{цветок1}, A) = \sqrt{(5,3 - 5,3)^2 + (1,6 - 1,5)^2} = \sqrt{0 + 0,01} = 0,1$$

$$d(\text{цветок1}, B) = \sqrt{(5,3 - 5,2)^2 + (1,6 - 1,5)^2} = \sqrt{0,01 + 0,01} = 0,141421356$$

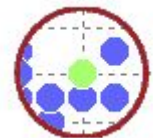
$$d(\text{цветок1}, C) = \sqrt{(5,3 - 5,2)^2 + (1,6 - 1,5)^2} = \sqrt{0,01 + 0,01} = 0,141421356$$

■

Первый цветок

Запись	Длина чашелистика	Длина лепестка	Расстояние	Класс
Цветок 1	5,3	1,6	–	–
A	5,3	1,5	0,1	Iris-Setosa
B	5,2	1,5	0,14	Iris-Setosa
C	5,2	1,5	0,14	Iris-Setosa

- Теперь определим класс нового цветка. Воспользуемся простым невзвешенным голосованием. Так как все три цветка A, B и C принадлежат к классу Iris-Setosa, то он получает 100% голосов, и первому цветку присваиваем этот класс.
- На диаграмме окрестность около первого цветка окружена точками синего цвета, соответствующие цветам класса Iris-Setosa.





Второй цветок

- Зададим $k = 3$ и предположим, что длина лепестка вдвое важнее длины чашелистика. Ближайшими тремя соседями будут следующие цветки:
 $A(6,1; 4,7)$, $B(6; 4,8)$, $C(6,2 4,8)$

$$d(\text{цветок2}, A) = \sqrt{(6,1 - 6,1)^2 + 2(4,8 - 4,7)^2} = \sqrt{0,02} = 0,141421356$$

$$d(\text{цветок2}, B) = \sqrt{(6,1 - 6)^2 + 2(4,8 - 4,8)^2} = \sqrt{0,01} = 0,1$$

$$d(\text{цветок2}, C) = \sqrt{(6,1 - 6,2)^2 + 2(4,8 - 4,8)^2} = \sqrt{0,01} = 0,1$$

- В и С – это *Iris Virginica*, а А – *Iris Versicolour*.



Второй цветок

Запись	Длина чашелистика	Длина лепестка	Расстояние	Класс
Цветок 2	6,1	4,8	–	–
A	6,1	4,7	0,14	Iris Versicolour
B	6	4,8	0,1	Iris Virginica
C	6,2	4,8	0,1	Iris Virginica

- Взвешенное голосование:

$$Votes(IrisVersicolour) = \frac{1}{\left(\sqrt{0,02}\right)^2} = \frac{1}{0,02} = 50$$

$$Votes(IrisVirginica) = \frac{1}{0,1^2} + \frac{1}{0,1^2} = \frac{2}{0,01} = 200$$

- Так как $200 > 50$, то второй цветок классифицируется как Iris Virginica.



Библиотека sklearn

- **Scikit-Learn** — это Python-библиотека, впервые разработанная David Cournapeau в 2007 году. В этой библиотеке находится большое количество алгоритмов для задач, связанных с классификацией и машинным обучением в целом. Scikit-Learn даёт доступ ко множеству различных алгоритмов классификации. Вот основные из них:
- Метод k-ближайших соседей (K-Nearest Neighbors);
 - Метод опорных векторов (Support Vector Machines);
 - Классификатор дерева решений (Decision Tree Classifier) / Случайный лес (Random Forests);
 - Наивный байесовский метод (Naive Bayes);
 - Линейный дискриминантный анализ (Linear Discriminant Analysis);
 - Логистическая регрессия (Logistic Regression).

<https://scikit-learn.org/stable/>



Реализация алгоритма KNN

```
knn.py x
1 from sklearn import metrics
2 from sklearn.model_selection import train_test_split
3 from sklearn.neighbors import KNeighborsClassifier
4 from sklearn.datasets import load_iris
5
6 iris_dataset = load_iris()
7
8 X_train, X_test, y_train, y_test = train_test_split(iris_dataset["data"], iris_dataset["target"])
9
10 model = KNeighborsClassifier(n_neighbors=4) # количество соседей
11 model.fit(X_train, y_train)
12
13 expected = y_test
14 predicted = model.predict(X_test)
15
16 print(metrics.classification_report(y_test, predicted))
17 print(metrics.confusion_matrix(y_test, predicted))
```



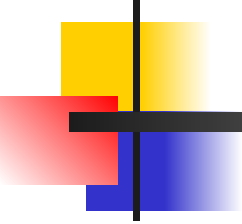
Матрица ошибок

- Допустим, что у нас есть два класса и алгоритм, предсказывающий принадлежность каждого объекта одному из классов, тогда матрица ошибок классификации будет выглядеть следующим образом:

	$y = 1$	$y = 0$
$\hat{y} = 1$	True Positive (TP)	False Positive (FP)
$\hat{y} = 0$	False Negative (FN)	True Negative (TN)

Здесь \hat{y} — это ответ алгоритма на объекте, а y — истинная метка класса на этом объекте. Таким образом, ошибки классификации бывают двух видов: False Negative (FN) и False Positive (FP).

Матрица ошибок классификации ириса



13	0	0
0	11	1
0	2	11

Доля правильных ответов модели (Accuracy)

- Интуитивно понятной, очевидной и почти неиспользуемой метрикой является accuracy — доля правильных ответов алгоритма:

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$



Ограничение Accuracy

- Метрика бесполезна в задачах с неравными классами.
- Допустим, мы хотим оценить работу спам-фильтра почты. У нас есть 100 не-спам писем, 90 из которых наш классификатор определил верно (True Negative = 90, False Positive = 10), и 10 спам-писем, 5 из которых классификатор также определил верно (True Positive = 5, False Negative = 5).
- Тогда accuracy:
$$accuracy = \frac{5 + 90}{5 + 90 + 10 + 5} = 86,4$$
- Однако если просто предсказывать все письма как не-спам, то получим более высокую accuracy:
$$accuracy = \frac{0 + 100}{0 + 100 + 0 + 10} = 90,9$$
- Преодолеть это нам поможет переход с общей для всех классов метрики к отдельным показателям качества классов.



Отчет классификации

	precision	recall	f1-score	support
0	1.00	1.00	1.00	13
1	0.85	0.92	0.88	12
2	0.92	0.85	0.88	13
accuracy			0.92	38
macro avg	0.92	0.92	0.92	38
weighted avg	0.92	0.92	0.92	38



Precision

- Precision можно интерпретировать как долю объектов, названных классификатором положительными и при этом действительно являющимися положительными.

$$precision = \frac{TP}{TP + FP}$$



Recall

- Recall показывает, какую долю объектов положительного класса из всех объектов положительного класса нашел алгоритм.

$$recall = \frac{TP}{TP + FN}$$

- Precision и recall не зависят, в отличие от accuracy, от соотношения классов и потому применимы в условиях несбалансированных выборок.



F-мера

- F-мера - среднее гармоническое precision и recall

$$F_{\beta} = (1 + \beta^2) \cdot \frac{precision \cdot recall}{(\beta^2 \cdot precision) + recall}$$



Преимущества алгоритма

- Алгоритм устойчив к аномальным выбросам, так как вероятность попадания такой записи в число k -ближайших соседей мала. Если же это произошло, то влияние на голосование (особенно взвешенное) (при $k > 2$) также, скорее всего, будет незначительным, и, следовательно, малым будет и влияние на итог классификации.
- Программная реализация алгоритма относительно проста.
- Результат работы алгоритма легко поддается интерпретации. Экспертам в различных областях вполне понятна логика работы алгоритма, основанная на нахождении схожих объектов.
- Возможность модификации алгоритма, путем использования наиболее подходящих функций сочетания и метрик позволяет подстроить алгоритм под конкретную задачу.



TF-IDF

- Показатель TF-IDF оценивает значимость слова в документе, на основе данных о всей коллекции документов.
- Аббревиатура расшифровывается так: TF - term frequency (частота слова), а IDF - inverse document frequency (обратная частота документа).
- Это простой и удобный способ оценить важность термина для какого-либо документа относительно всех остальных документов. Принцип такой — если слово встречается в каком-либо документе часто, при этом встречаясь редко во всех остальных документах — это слово имеет большую значимость для того самого документа.



Векторная модель

- Векторная модель (англ. vector space model) — в информационном поиске представление коллекции документов векторами из одного общего для всей коллекции векторного пространства.
- Мера TF-IDF часто используется для представления документов коллекции в виде числовых векторов, отражающих важность использования каждого слова из некоторого набора слов (количество слов набора определяет размерность вектора) в каждом документе. Подобная модель называется векторной моделью и даёт возможность сравнивать тексты, сравнивая представляющие их вектора.



Term Frequency

- **TF** — это частотность термина, которая измеряет, насколько часто термин встречается в документе.
- Логично предположить, что в длинных документах термин может встретиться в больших количествах, чем в коротких, поэтому абсолютные числа тут не катят. Поэтому применяют относительные — делят количество раз, когда нужный термин встретился в тексте, на общее количество слов в тексте.

$$\text{tf}(t, d) = \frac{n_t}{\sum_k n_k}$$



Inverse Document Frequency

- Показатель IDF равен логарифму отношения количества документов в коллекции к количеству документов в коллекции, в которых встречается заданное слово.

$$\text{idf}(t, D) = \log \frac{|D|}{|\{ d_i \in D \mid t \in d_i \}|}$$

- Учет IDF позволяет снизить вес слов, употребляемых часто (например, предлогов).

Term Frequency — Inverse Document Frequency

- Большой вес в TF-IDF получают слова с высокой частотой в пределах конкретного документа и с низкой частотой употреблений в других документах.

$$\text{tf-idf}(t, d, D) = \text{tf}(t, d) \times \text{idf}(t, D)$$



Пример расчета 1

- Для первого примера расчета TF-IDF возьмём документ из 10000 символов в котором слово “пропан” встречается 25 раз, а коллекция состоит из 2 миллионов документов, в 2000 из которых также встречается данное слово.
- $TF = 25 \div 10000 = 0,0025$
- $IDF = \lg(2000 \div 2000000) = \lg(1 \div 1000) = -3$ (\lg - логарифм с основанием 10)
- $TF-IDF = 0,0025 \times 3 = 0,0075$



Пример расчета 2

- Для второго примера расчета TF-IDF возьмём тот же документ из 10000 символов в котором союз “но” встречается 30 раз, а коллекция по-прежнему состоит из 2 миллионов документов, в 200000 из которых также встречается данное слово.
- $TF = 30 \div 10000 = 0,003$
- $IDF = \lg(200000 \div 2000000) = \lg(1 \div 10) = -1$ (\lg - логарифм с основанием 10)
- $TF-IDF = 0,003 \times 1 = 0,003$