"Нормальный" метод, или
Наивный баесовский классификатор

Привет, сегодня мы будем изучать такую вот... вещь. Мда уж, звучит не очень, но сегодня мы точно разберемся, почему его называют наивным, почему он баесовский, и почему вообще классификатор. Поехали!

1. Почему Наивный
   Дело в том, что для реализации алгоритма мы принимаем так называемое *допущение,* что каждый исследуемый нами параметр данных не зависит от других параметров, то есть они не оказывают никакое влияние друг на друга. К примеру, рост и вес человека имеют некое влияние друг на друга, ведь верно будет предположить, что более высокий человек в среднем будет весить больше других.

Пример

Давайте сейчас быстренько посмотрим на примере, как работает такого рода алгоритм. Для начала, давайте создадим набор данных Fruits из магазина какого-нибудь дяди Пети на рынке. В них мы представим данные о различных видах фруктов: бананах, апельсинах, и сливах. Сделаем для них несколько факторов, таких, как, к примеру:

- Кол-во Длинных (Long)
- Кол-во Сладких (Sweet)
- Кол-во Желтых (yellow)
- Количество в магазине (Total)
  Создадим набор таких данных

```
import numpy as np
import pandas as pd

data = np.array([[400, 350, 450, 500],
                 [0, 150, 300, 300],
                 [30, 180, 100, 200],
                 [430, 680, 850, 1000]])

idx = ['Banana', 'Orange', 'Plum', 'Total']
col = ['Long', 'Sweet', 'Yellow', 'Total']

fruits = pd.DataFrame(data, columns=col, index=idx)
fruits
```

	Long	Sweet	Yellow	Total
Banana	400	350	450	500
Orange	0	150	300	300
Plum	30	180	100	200
Total	430	680	850	1000

Данный метод на самом деле оперирует теорией вероятностей, значит, к примеру, если мы возьмем 1 банан среди этих 500-ста, то с вероятностью 80% он будет длинный, с вероятностью 70% он будет сладким, и 90% уйдет на то что он будет желтым.
Допустим, мы хотим узнать, к какому классу относится фрукт с параметрами {длинный, сладкий, желтый}. Вероятность того, что объект принадлежит какому-либо классу при данных параметрах обозначается как:

$$P(Class|Long, Sweet, Yellow)$$

Чтобы вычислить вероятность в случае дискретных (конечных) данных, мы делим количество соответствующих признаку объектов на общее число объектов. Например, если мы хотим узнать вероятность $P(Long|Banana)$, мы вычисляем $400/500 = 80\%$, как уже упоминалось выше.
Данный метод основан на теореме английского математика-статистика Томаса Байеса. По сути, она позволяет предсказать

класс на основании набора параметров, используя вероятность. Общая формула Байеса для класса с одним признаком выглядит так:

$$P(ClassA|Feature1) = \frac{P(Feature1|ClassA) \cdot P(ClassA)}{P(Feature1)}$$

$P(ClassA|Feature1)$ - вероятность того, что объект является классом A при том, что его признак соответствует $Feature1$.

Упрощенное уравнение для классификации при двух признаках выглядит так:

$$P(ClassA|Feature1, Feature2) = \frac{P(Feature1|ClassA) \cdot P(Feature2|ClassA) \cdot P(ClassA)}{P(Feature1) \cdot P(Feature2)}$$

И далее для большего количества признаков формула меняется соответствующим образом.

Напомню глупеньким (кек в), что вероятность - число, которое принимает значения от 0 до 1, при этом 0 - полное несоответствие класса признакам, а 1 - однозначно определенный класс. Соответственно, чем ближе значение вероятности определенного класса к 1, тем больше шанс того, что объект принадлежит именно этому классу.
Знаменатель можно проигнорировать, так как он будет одинаков для всех вычислений и никакой важной информации не даст.
Тогда получится следующее уравнение:

$$P(ClassA|Feature1, Feature2) = P(Feature1|ClassA) \cdot P(Feature2|ClassA) \cdot P(ClassA)$$

То есть для каждого возможного класса вычисляем только произведение вероятностей того, что каждый признак соответствует классу, и вероятности того, что объект принадлежит этому классу. Соответственно, наибольшее значение этого произведения, рассчитанного с признаками конкретного объекта, для какого-то из классов будет указывать на принадлежность объекта к этому классу.

Вернемся к нашему примеру с фруктами. Чтобы понять, к какому классу принадлежит объект с признаками {Long, Sweet, Yellow}, рассчитаем для каждого из классов формулу Байеса, используя для этого данные частотной таблицы. На выходе получаем вероятность того, что объект принадлежит классу. Класс с наибольшей вероятностью является ответом.

```
result = {}

print(fruits.values)

for i in range (fruits.values.shape[0] - 1):
    p = 1

    print(f"Сначала p={p}")

    for j in range (fruits.values.shape[1] - 1):
        p *= fruits.values[i, j] / fruits.values[i, -1]

        print(f"p = p * {fruits.values[i, j]}/{fruits.values[i, -1]} = {p}")

    p *= fruits.values[i, -1] / fruits.values[-1, -1]

    result[fruits.index[i]] = p

result
```

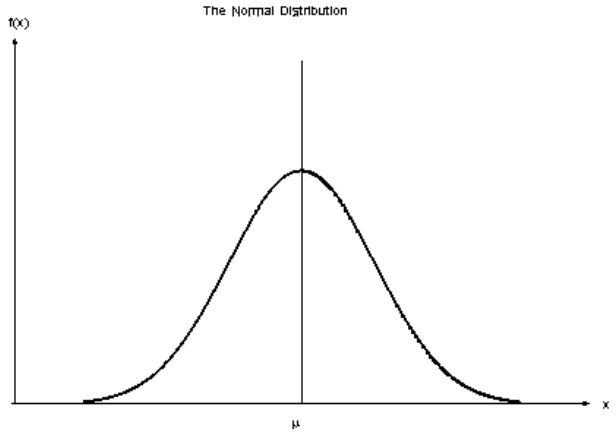
```
[[ 400  350  450  500]
 [   0  150  300  300]
 [  30  180  100  200]
 [ 430  680  850 1000]]
Сначала p=1
p = p * 400/500 = 0.8
p = p * 350/500 = 0.5599999999999999
p = p * 450/500 = 0.504
Сначала p=1
p = p * 0/300 = 0.0
p = p * 150/300 = 0.0
```

```
p = p * 300/300 = 0.0
Сначала p=1
p = p * 30/200 = 0.15
p = p * 180/200 = 0.135
p = p * 100/200 = 0.0675



{'Banana': 0.252, 'Orange': 0.0, 'Plum': 0.013500000000000002}
```

Как видите, мы перемножили вероятностей всех классов для каждого из фрукта, и запихнули в отдельный кортеж (вроде бы так называется).

Ну, и как вы можете заметить, вероятность того, что у фрукта одновременность будет и длина, и желтость, и сочность, будет у банана! 25%!

Но что делать, если у нас не частотная таблица, как в примере выше, а непрерывные данные? Например, мы не можем вычислять, сколько человек с конкретным ростом 1,81м, 1,67м и т.д. присутствует в выборке - нам это попросту ничего не даст, а лишь добавит громоздких вычислений. Поэтому обычно при непрерывных значениях параметров используется гауссовский наивный Байес, в котором сделано предположение о том, что значения параметров взяты из нормального распределения.



Сори, но дальше возьму уже тупо готовую теорию, потому что я это идельано знаю и помню.

На графике - плотность вероятности нормального распределения. По сути, где больше площадь под графиком, там и наиболее вероятные значения. Поскольку способ представления значений в наборе данных изменяется, то и формула условной вероятности изменяется на

$$p(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} exp(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2})$$

Здесь σ^2 - дисперсия (разброс) данных, а μ - математическое ожидание (среднее значение). При этом y - предполагаемый класс, а x_i - значение признака у того объекта, который нужно классифицировать.

По большому счету в нашей начальной формуле для вычисления вероятности того, что объект с данными признаками относится к конкретному классу, мы просто заменяем формулу вычисления вероятности как отношения количества соответствующих признаку объектов к общему числу объектов на данную, а дальше проводим идентичные вычисления.

Теперь попробуем использовать Гауссову кривую, построенную по нашим дискретным значениям, чтобы вывести вероятность на новый уровень, способную находить вероятность для любого события EVER!

Теперь импортируем модель GaussianNB из библиотеки sklearn и посмотрим, как она работает на уже известном нам датасете Iris.

```
from sklearn.naive_bayes import GaussianNB
```

```
nb = GaussianNB()

from sklearn.datasets import load_iris

iris_dataset=load_iris() # возвращает объект с несколькими полями

print(iris_dataset.keys())

iris_dataframe=pd.DataFrame(iris_dataset['data'], columns = iris_dataset.feature_names)
iris_dataframe.head()
```

```
dict_keys(['data', 'target', 'frame', 'target_names', 'DESCR', 'feature_names', 'filename', 'data_module'])
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

```
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(iris_dataset.data[:, 2:4],
                                                    iris_dataset['target'],
                                                    random_state=57) # random_state - для воспроизводимости

print(f'X_train shape: {x_train.shape}, y_train shape: {y_train.shape},\n'
      f'X_test shape: {x_test.shape}, y_test shape: {y_test.shape}')
```

```
X_train shape: (112, 2), y_train shape: (112,),
X_test shape: (38, 2), y_test shape: (38,)
```

```
nb_model = nb.fit(x_train, y_train) # Если возникают вопросы, что, зачем, и почему, возвращайтесь на предыдущую статью, вам здесь пока рано находиться.
```

```
nb_predictions=nb.predict(x_test)
nb_predictions
```

```
array([2, 2, 1, 1, 0, 1, 0, 2, 1, 0, 1, 0, 0, 0, 1, 2, 1, 0, 0, 2, 2, 0,
       2, 2, 2, 0, 2, 2, 2, 2, 0, 2, 0, 0, 1, 2, 0, 1])
```

Для определения точности предсказаний воспользуемся встроенной функцией *score*.

```
accuracy= nb.score(x_test, y_test)
print(f"Accuracy:{accuracy}")
```

```
Accuracy:1.0
```

В итоге мы реализовали "Нормальный" метод, будем его так теперь)