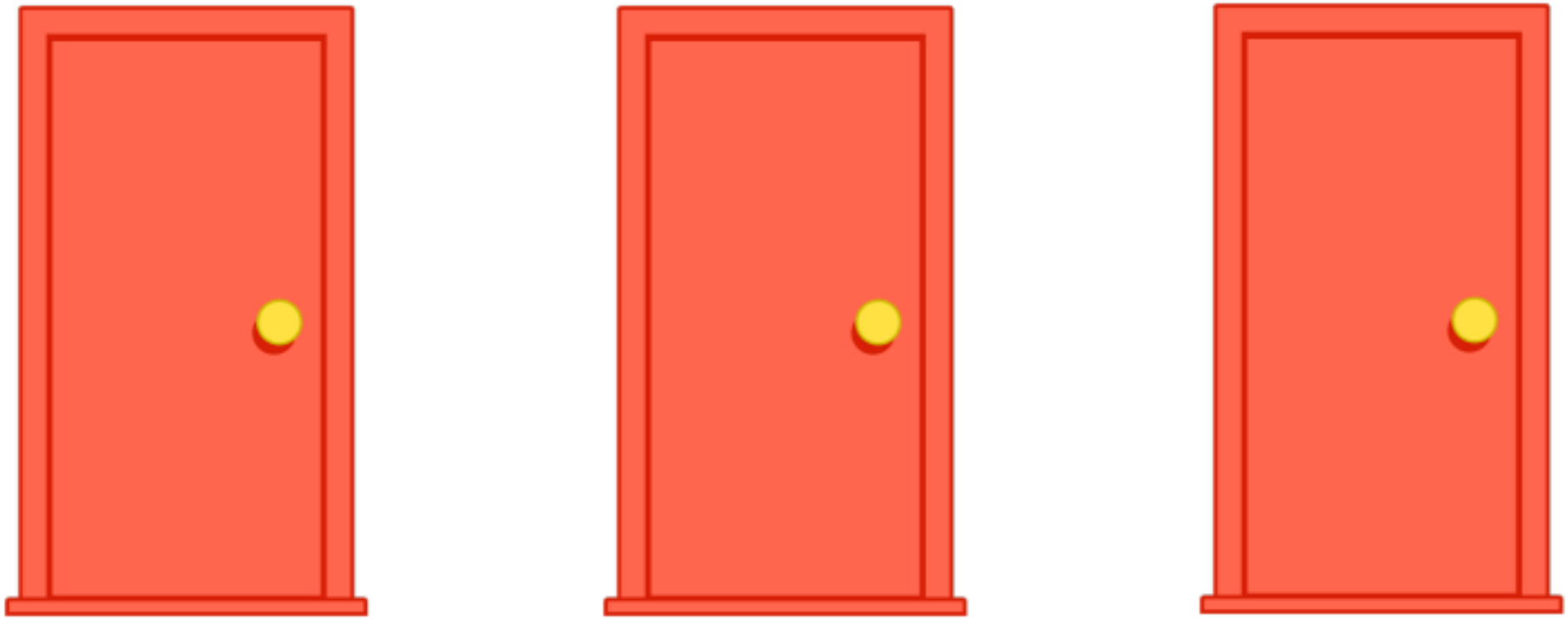
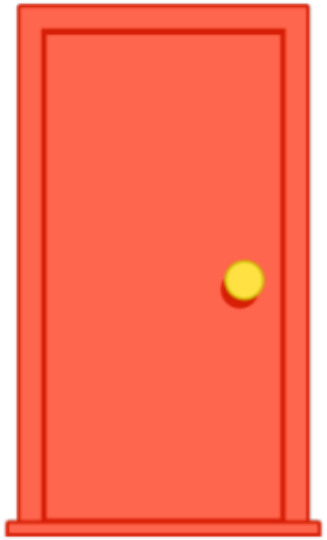
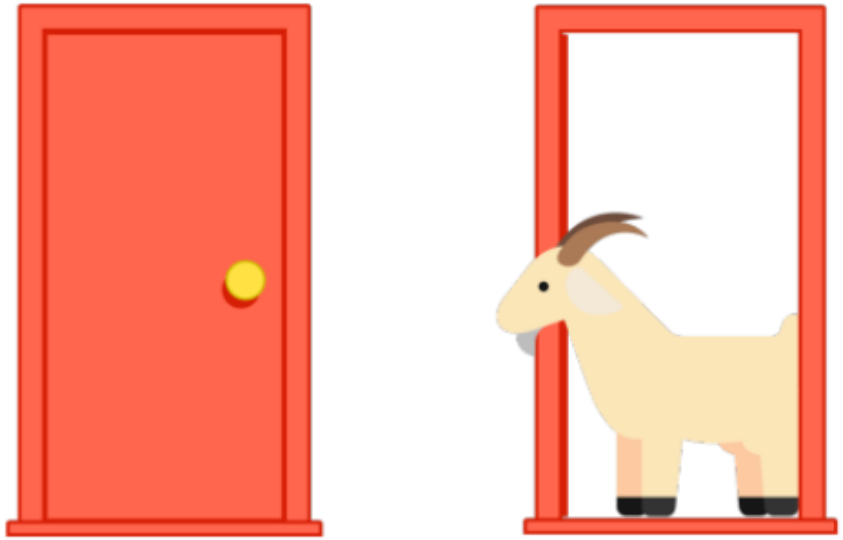
Наивный байесовский классификатор

Парадокс Монти-Холла

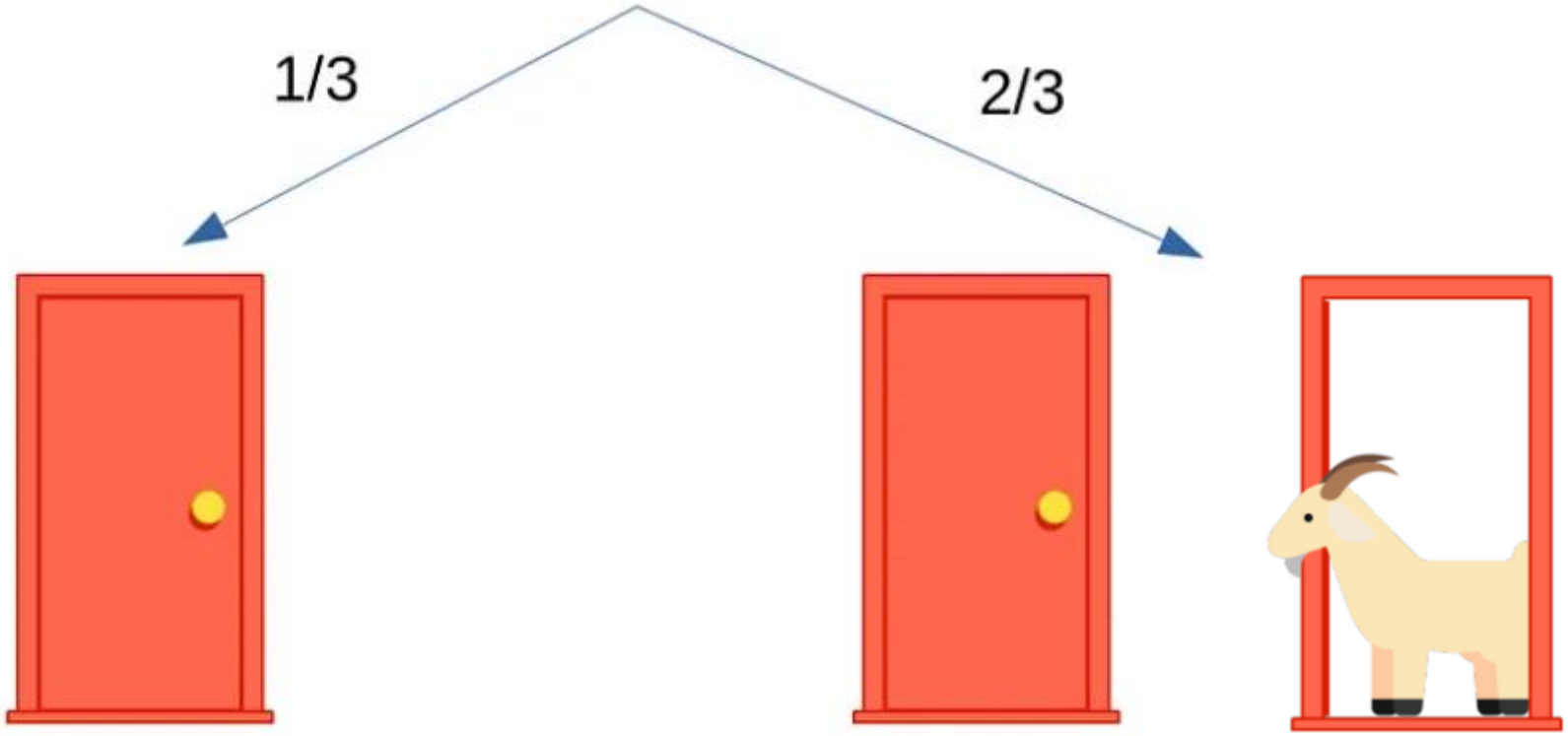


Парадокс Монти-Холла

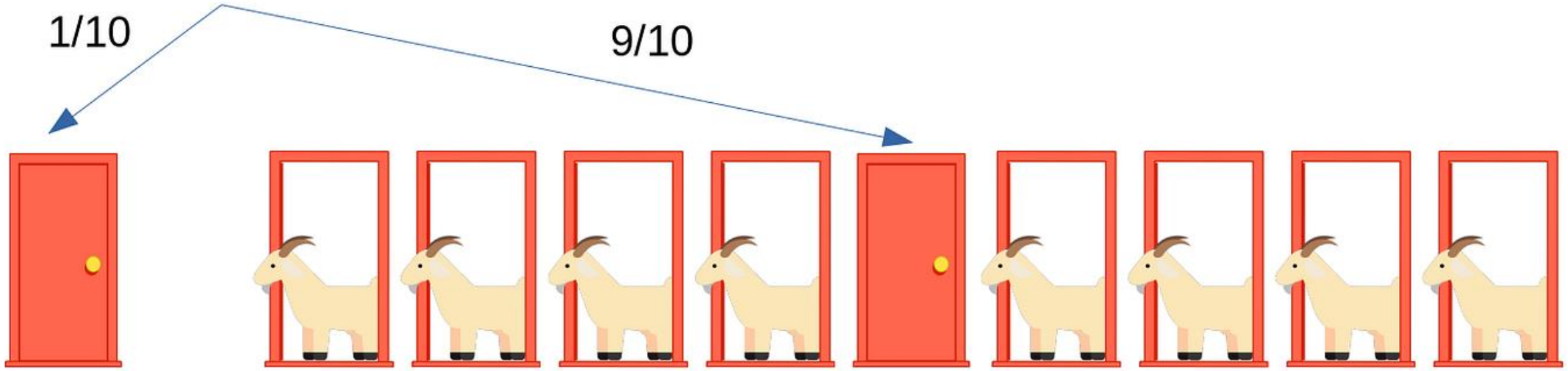




Парадокс Монти-Холла



Парадокс Монти-Холла



Наивный байесовский классификатор

- Постановка задачи и приемы работы с текстом в машинном обучении.
- Принципы работы Наивного байесовского классификатора.
- Построение и тренировка модели, измерение качества модели.

Какие задачи связанные с текстом можно решать с помощью методов МО?

1. Синтаксические задачи
    - Разметка по частям речи и по морфологическим признакам
    - Деление слов в тексте на морфемы (суффикс, приставка и пр.)
    - Поиск имен и названий в тексте ("распознавание именованных сущностей")
    - Разрешение смысла слов в заданном контексте (зАмок или замОк)
2. Задачи на понимание текста, в которых есть "учитель"
    - Машинный перевод
    - Диалоговые модели (чат-боты)
3. Другие задачи:
    - Описание изображения
    - Распознавание речи

Сложность работы с текстом 1

"Мама мыла раму, и теперь она блестит"
"Мама мыла раму, и теперь она сильно устала"

"Кубок не помещался в чемодан, потому что он был слишком велик. Что именно было слишком велико, чемодан или кубок?"

Сложность работы с текстом 2

Рассмотрим следующую ситуацию:

Есть набор из 1000 объектов недвижимости (дома). Каждый объект характеризуется одним и тем же набором признаков. Допустим, что это следующие признаки:

1. Площадь.
2. Возраст после ремонта.
3. Удаленность от ближайшей остановки.

Будем обозначать такой набор признаков как x. Например: $x = (150, 5, 600)$.

Мы хотим предсказать цену, за которую этот дом можно продать на рынке. Будем ее обозначать как y (в млн руб). Например, $y = 8$.

Запишем то же самое более формально:

Есть набор векторов $\{x_i\}_{i=1}^{N}$, всего $N = 1000$ штук.

Каждому объекту соответствует переменная, которую мы хотим научиться предсказывать $\{y_i\}_{i=1}^{N}$.

Как преобразовать входной текст в вектор из чисел?

1. Зафиксируем словарь размера K.
2. Каждое слово в тексте будем представлять в следующем виде:

$$(0, 0, 0, \ldots 0, 1, 0, \ldots 0)$$

Этот подход называется "1-hot-encoding". Слова в таком контексте называем "токенами".

Пример

Текст: "Кот увидел кота".

Игнорируем разницу в окончаниях и суффиксах. Получаем словарь размером 2, и итоговый текст представляем в виде:

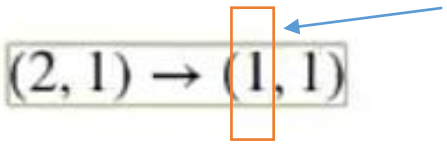
$$(1, 0)$$
$$(0, 1)$$
$$(1, 0)$$

Полученное представление будет зависеть от длины текста. Наивное и простое решение -- просуммировать по столбцам. Получаем:

$$(2, 1)$$

Получился просто счетчик слов. Этот подход будем называть **"bag-of-words"** и далее использовать.

Полученное представление будет зависеть от длины текста. Наивное и простое решение -- просуммировать по столбцам. Получаем:



1 – если слово встречалось,
0 - если не встречалось

Получился просто счетчик слов. Этот подход будем называть **"bag-of-words"** и далее использовать.

Итого:

Создаем словарь, делаем счетчики слов для каждого текста. Получаем вектор фиксированной длины для каждого текста.

Еще более простой подход: **boolean mask**.

Теорема Байеса:

Условная переменная значения A примет значение a

Условная вероятность события B при условии A

$$p(A = a|B = b) = \frac{p(B = b|A = a)p(A = a)}{p(B = b)}$$

Для нескольких переменных:

Какова вероятность принадлежности переменой y с признаками x1 и x2 классу c

Вероятность пронаблюдать признаки x1 и x2 в объекте класса c

Априорную вероятность наблюдения класса c

$$p(y = c|x_1, x_2) = \frac{p(x_1, x_2|y = c)p(y = c)}{p(x_1, x_2)}$$

Вероятность наблюдения признаков x1 и x2

Задача для решения:

Классифицировать электронные письма по категориям спам / не-спам (spam/ham). Это задача бинарной классификации ($C = 2$).

Какова вероятность принадлежности переменой y с признаками x1 и x2 классу c

Вероятность пронаблюдать признаки x1 и x2 в объекте класса c

Априорную вероятность наблюдения класса c

$$p(y = c|x_1, x_2) = \frac{p(x_1, x_2|y = c)p(y = c)}{p(x_1, x_2)}$$

Вероятность наблюдения признаков x1 и x2

X1=слово «деньги», X2=слово «приз»

Формулы и детали

Пусть $\mathbf{x}$ это boolean mask представление для объекта. Мы хотим найти следующую величину:

$$p(y|\mathbf{x})$$

По ТБ:

$$p(y = c|\mathbf{x}) = \frac{p(\mathbf{x}|y)p(y)}{p(\mathbf{x})} \sim p(\mathbf{x}|y)p(y)$$

Не зависит от метки класса

Задача: узнать спам/не спам. Чтобы это узнать надо записать выражения p(y=c|x) для каждого из случаев(спам (c=0) и не спам (c=1)), соответственно посмотреть чему равно выражение p (x|y)p(y). Для того C для которого значение p (x|y)p(y) окажется большим – тот вариант будет верным.

Итого, чтобы выполнить классификацию, нужно знать явный вид функций распределения $p(\mathbf{x}|y$ и $p(y)$. Последнее это просто категориальное распределение, а чтобы вычислить первое нужно ввести дополнительные допущения.

Допущения

Признаки (features) проявляются независимо друг от друга при условии фиксированной метки класса:

Boolean mask

$$\mathbf{x} = (x_1, \ldots x_K)$$

$$p(\mathbf{x}|y) = \prod_{j=1}^{K} p(x_j|y)$$

Вероятность пронаблюдать метку x при условии спам

Будем факторизовать по всем компонентам

Преимущества такой факторизации:

- Модель устойчива к переобучению, т.к. общее число параметров составляет $O(CD)$.

Недостатки:

- Сильное допущение, маловероятно что будет выполняться на практике.

Конкретный вид $p(x_j|y)$ зависит от того, с какими признаками мы имеем дело. Будем использовать самый простой вариант, распределение Бернулли:

$$p(x = True) = \theta$$
$$p(x = False) = 1 - \theta$$

Тогда каждый объект характеризуется K признаками (есть или нет j-ое слово в тексте).
Можем записать искомые вероятности в виде:

Какова будет вероятность, что мы пронаблюдаем в письме слово из словаря под номером j

Предположим, что к нам пришло письмо из категории спам:

Если это слово встречалось, то индикаторная ф-ция = 1(xj):

Если это слово не встречалось, то индикаторная ф-ция = 1 - xj :

$$p(\mathbf{x}|y = c) = \prod_{j=1}^{K} p(x_j|y = c) = \prod_{j=1}^{K} \theta_{jc}^{\mathbf{1}(x_j)}(1 - \theta_{jc})^{\mathbf{1}(1-x_j)}$$

$$p(y = c) = \pi_c$$

Вероятность того, что это слово встречалось будет равно Teta jc

Вероятность того, что это слово не встречалось будет равно 1-Teta jc

Модельный пример

Пускай дана следующая обучающая выборка ($x \to y$):

$$(1,0,0) \to 0$$
$$(0,1,0) \to 0$$
$$(0,0,1) \to 1$$
$$(0,0,1) \to 1$$

Как будет классифицирован вектор $(1,1,0)$?

Априорные вероятности для каждого класса равны 0.5 (=2/4). Прочие параметры (вероятности наблюдать 1 для заданного признака):

$$\theta_{10} = 1/2 = 0.5$$
$$\theta_{20} = 1/2 = 0.5$$
$$\theta_{30} = 0/2 = 0$$
$$\theta_{11} = 0/2 = 0$$
$$\theta_{21} = 0/2 = 0$$
$$\theta_{31} = 2/2 = 1$$

Априорная вероятность

$$p(y = c|\mathbf{x}) = \frac{p(\mathbf{x}|y)p(y)}{p(\mathbf{x})} \sim p(\mathbf{x}|y)p(y)$$

$$p(\mathbf{x}|y = c) = \prod_{j=1}^{K} p(x_j|y = c) = \prod_{j=1}^{K} \theta_{jc}^{\mathbf{1}(x_j)}(1 - \theta_{jc})^{\mathbf{1}(1-x_j)}$$

Априорные вероятности для каждого класса равны 0.5 (=2/4). Прочие параметры (вероятности наблюдать 1 для заданного признака):

$(1, 1, 0)$?

$\theta_{10} = 1/2 = 0.5$
$\theta_{20} = 1/2 = 0.5$
$\theta_{30} = 0/2 = 0$
$\theta_{11} = 0/2 = 0$
$\theta_{21} = 0/2 = 0$
$\theta_{31} = 2/2 = 1$

$$p(y = c|\mathbf{x}) = \frac{p(\mathbf{x}|y)p(y)}{p(\mathbf{x})} \sim p(\mathbf{x}|y)p(y)$$

Рассчитываем вероятности принадлежности к классам для нового объекта $\mathbf{x} = (1, 1, 0)$:

$$p(y = 0|\mathbf{x}) \sim p(y = 0)p(\mathbf{x}|y = 0) = 0.5 \cdot p(x_1 = 1|y = 0) \cdot p(x_2 = 1|y = 0)$$
$$\cdot p(x_3 = 0y = 0) = 0.5 \cdot 0.5 \cdot 0.5 \cdot (1 - 0) = 0.125$$

$$p(y = 1|\mathbf{x}) \sim p(y = 1)p(\mathbf{x}|y = 1) = 0.5 \cdot 0 \cdot 0 \cdot 0 = 0$$

Таким образом, наивный байесовский классификатор предскажет класс 0 для этого объекта.

Построение модели

In [1]:
```
import numpy as np
import pandas as pd

from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import BernoulliNB
from sklearn.metrics import accuracy_score
```

In [2]:
```
df = pd.read_csv('spam.csv', usecols=[0, 1], encoding='latin-1')
df.head()
```

Out[2]:

	v1	v2
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...

```
In [4]: df['v1'].value_counts()

Out[4]: ham     4825
        spam     747
        Name: v1, dtype: int64

In [5]: vectorizer = CountVectorizer(binary=True)

In [ ]: x = vectorizer.fit_transform(df['v2'])
        y = pd.get_dummies(df['v1'])['spam']

In [ ]: x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.33
```

Training

```
In [8]: clf = BernoulliNB()

In [ ]: clf.fit(x_train, y_train)
```

Evaluation

In [10]:
```
y_pred = clf.predict(x_train)
print('Train accuracy: {:.2f}%'.format(100*accuracy_score(y_train, y_pre
```

```
Train accuracy: 98.61%
```

In [11]:
```
y_pred = clf.predict(x_test)
print('Test accuracy: {:.2f}%'.format(100*accuracy_score(y_test, y_pred)
```

```
Test accuracy: 97.82%
```

Analyzing features

In [14]:
```
voc = {v:k for k, v in vectorizer.vocabulary_.items()}
```

In [15]:
```
features_ham = clf.feature_log_prob_[0]
features_spam = clf.feature_log_prob_[1]
```

In [16]:
```
top_ham = (-features_ham).argsort()[:30]
print([voc[x] for x in top_ham])
```

```
['you', 'to', 'the', 'in', 'me', 'and', 'is', 'my', 'it', 'that', 'fo
r', 'of', 'so', 'but', 'have', 'not', 'at', 'are', 'can', 'on', 'your',
'do', 'will', 'if', 'we', 'be', 'get', 'now', 'just', 'how']
```

In [17]:
```
top_spam = (-features_spam).argsort()[:30]
print([voc[x] for x in top_spam])
```

```
['to', 'call', 'you', 'your', 'now', 'or', 'free', 'for', 'the', 'txt',
'is', 'have', 'from', 'on', 'mobile', 'claim', 'text', 'with', 'and',
'stop', 'ur', 'www', 'get', 'this', 'reply', 'of', 'won', 'prize', 'onl
y', 'are']
```

Gaussian

- используется в случае непрерывных признаков;
- значения признаков имеют нормальное распределение

Multinomial

- используется в случае дискретных признаков;
- Например, в задаче классификации текстов признаки могут показывать, сколько раз каждое слово встречается в данном тексте