# Лекция 8

# Метод Опорных Векторов

- Метод Опорных Векторов
- Ядра
- Наивный байесовский классификатор

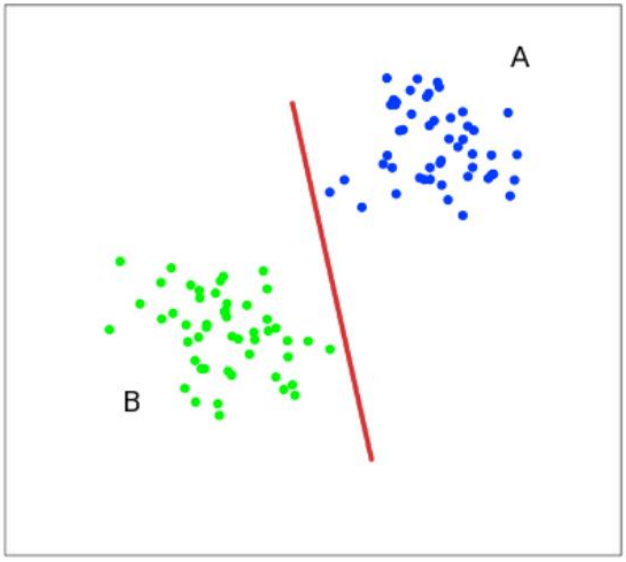
## Метод Опорных Векторов

https://habr.com/ru/post/428503/

**Метод Опорных Векторов** или **SVM** (от англ. Support Vector Machines) — это линейный алгоритм, используемый в задачах классификации и регрессии. Метод Опорных Векторов относится к методам обучения с учителем. Данный метод изначально относится к бинарным классификаторам, хотя существуют способы заставить его работать и для задач мультиклассификации. Данный алгоритм имеет широкое применение на практике. *Суть работы Метода Опорных Векторов: алгоритм создает линию или гиперплоскость, которая разделяет данные на классы.*

Рассмотрим решение задачи классификации, математическая формулировка которой такова: пусть **X** — пространство объектов (например, $R^n$), **Y** — наши классы (например, Y = {-1,1}). При обучении алгоритм должен построить функцию **F(x)=y**, которая принимает в себя аргумент **x** — объект из пространства $R^n$ и выдает метку класса **y**.

Идею метода удобно проиллюстрировать на следующем простом примере:




даны точки на плоскости, разбитые на два класса.

Проведем линию, разделяющую эти два класса (красная линия). Далее, все новые точки (не из обучающей выборки) автоматически классифицируются следующим образом:
точка правее прямой попадает в класс A,
точка левее прямой — в класс B.

Такую прямую назовем разделяющей прямой. Однако, в пространствах высоких размерностей прямая уже не будет разделять наши классы, так как понятие «правее прямой» или «левее прямой» теряет всякий смысл. Поэтому вместо прямых необходимо рассматривать гиперплоскости — пространства, размерность которых на единицу меньше, чем размерность исходного

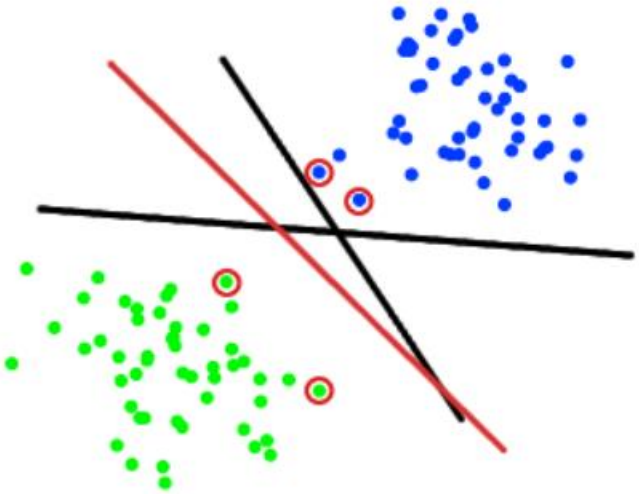
пространства. Например, в трехмерном пространстве гиперплоскость — это обычная двумерная плоскость.

Формальное определение гиперплоскости

Гиперплоскость — это n-1 мерная подплоскость в n-мерном Евклидовом пространстве, которая разделяет пространство на две отдельные части. *Например, представим, что наша линия представлена в виде одномерного Евклидова пространства (т.е. наш набор данных лежит на прямой). Выберите точку на этой линии. Эта точка разделит набор данных, в нашем случае линию, на две части. У линии есть одна мера, а у точки 0 мер. Следовательно, точка — это гиперплоскость линии.*

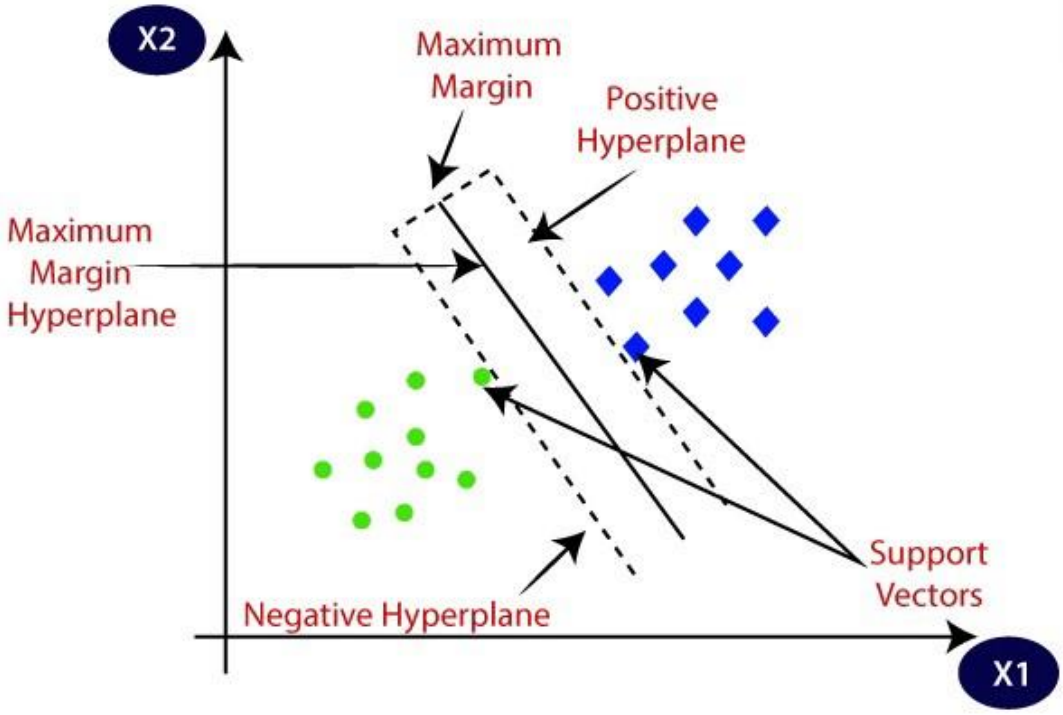
Для двумерного датасета, с которым мы познакомились ранее, разделяющая прямая была той самой гиперплоскостью. Проще говоря, для n-мерного пространства существует n-1 мерная гиперплоскость, разделяющая это пространство на две части.

Однако в нашем примере существует несколько прямых, разделяющих два класса:

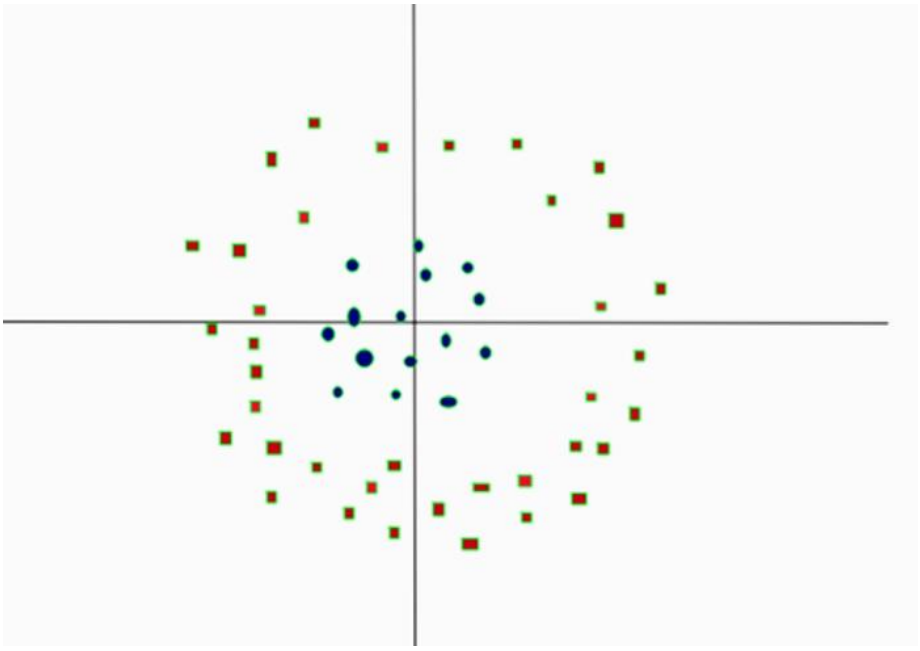


**Как SVM находит лучшую линию**

Алгоритм SVM устроен таким образом, что он ищет точки, которые расположены ближе всех к разделяющей гиперплоскости. Эти точки называются **опорными векторами (support vectors)** *(на рисунке обведены красным)*. Затем, алгоритм вычисляет расстояние между опорными векторами и разделяющей плоскостью. Это расстояние называется **зазором**. Основная цель алгоритма — максимизировать расстояние зазора. *Лучшей гиперплоскостью считается такая гиперплоскость, для которой этот зазор является максимально большим, поскольку, как правило, чем больше зазор, тем ниже ошибка обобщения классификатора.* Такая гиперплоскость, называется **оптимальной разделяющей гиперплоскостью**.




Рассмотрим пример, с более сложным датасетом, который нельзя разделить линейно.

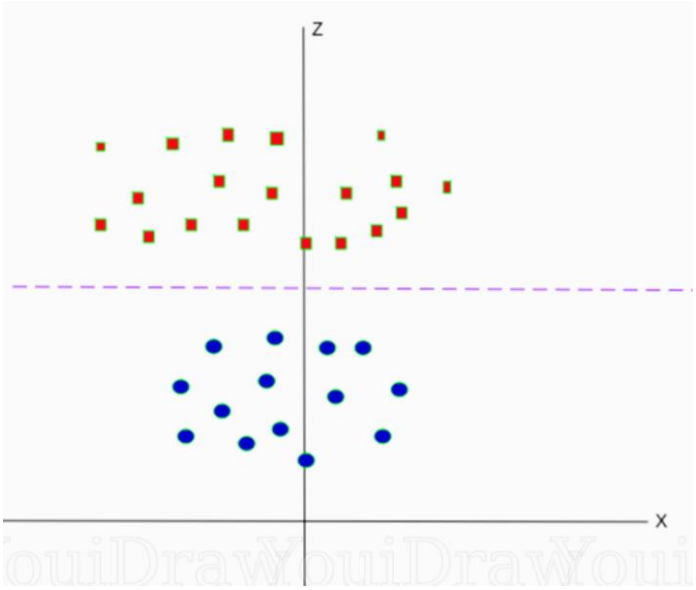


Здесь невозможно начертить прямую линию, которая бы классифицировала эти данные. Но этот датасет можно разделить линейно, добавив дополнительное измерение, которое мы назовем осью Z. Представим, что координаты на оси Z регулируются следующим ограничением:

$$z = x + y$$

Таким образом, ордината Z представлена из квадрата расстояния точки до начала оси.

Ниже приведена визуализация того же набора данных, на оси Z.



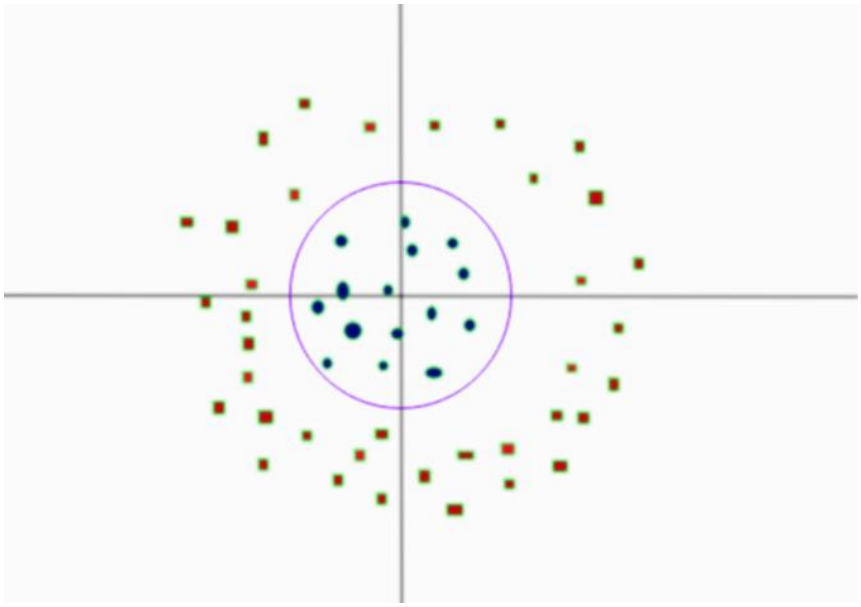

Теперь данные можно разделить линейно. Допустим линия разделяющая данные z=k, где k константа. Если

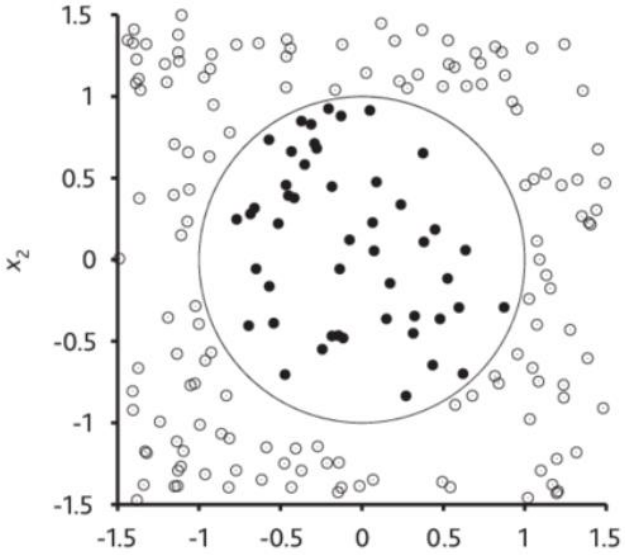
$$z = x^2+y^2$$

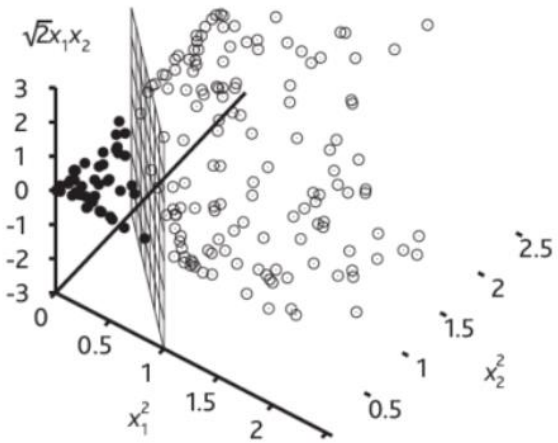
, то следовательно и

$$k = x^2+y^2$$

— формула окружности. Таким образом, можно спроецировать линейный разделитель, обратно к исходному количеству измерений выборки, используя эту трансформацию. В итоге, можно классифицировать нелинейный набор данных добавив к нему дополнительное измерение, а затем, привести обратно к исходному виду используя математическую трансформацию.






Итак, если выборка объектов с признаковым описанием из $X=R^n$ не является линейно разделимой, можно предположить, что существует некоторое пространство **H**, вероятно большей размерности, при переходе в которое выборка станет линейно разделимой. Пространство **H** называют **спрямляющим**.

Подобные подходы к изменению признакового пространства называют ядровыми методами, в которых используют повышение размерность пространства при помощи ядер.

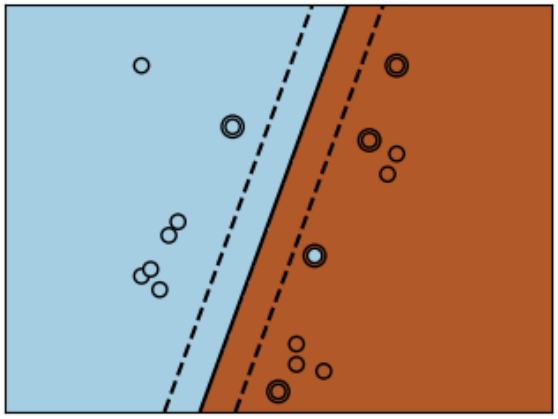
**Ядром** называют функцию K(x, z), представленную в виде скалярного произведения в некотором пространстве: $K(x,z)=\langle\varphi(x),\varphi(z)\rangle$,

где $где\ \varphi: X \to H$ — отображение из исходного признакового пространства в некоторое спрямляющее пространство.

**Примеры влияния разных ядер на вид оптимальной разделяющей гиперплоскости**:

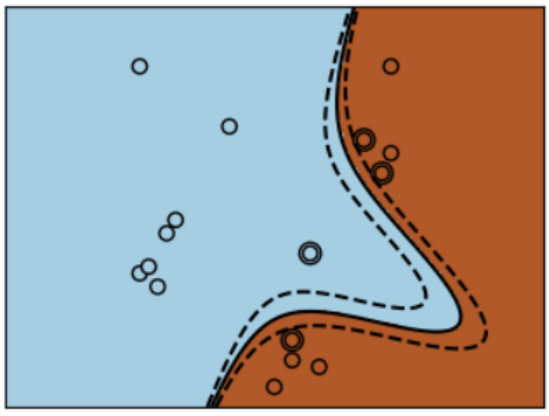
Linear kernel

```
>>> svc = svm.SVC(kernel='linear')
```



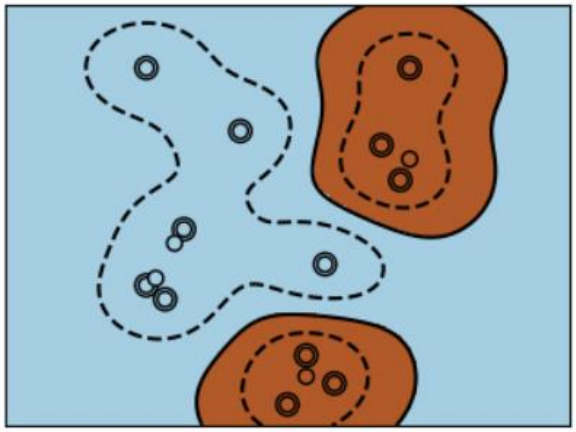
Полиномиальное ядро

```
>>> svc = svm.SVC(kernel='poly',
...               degree=3)
>>> # degree: polynomial degree
```



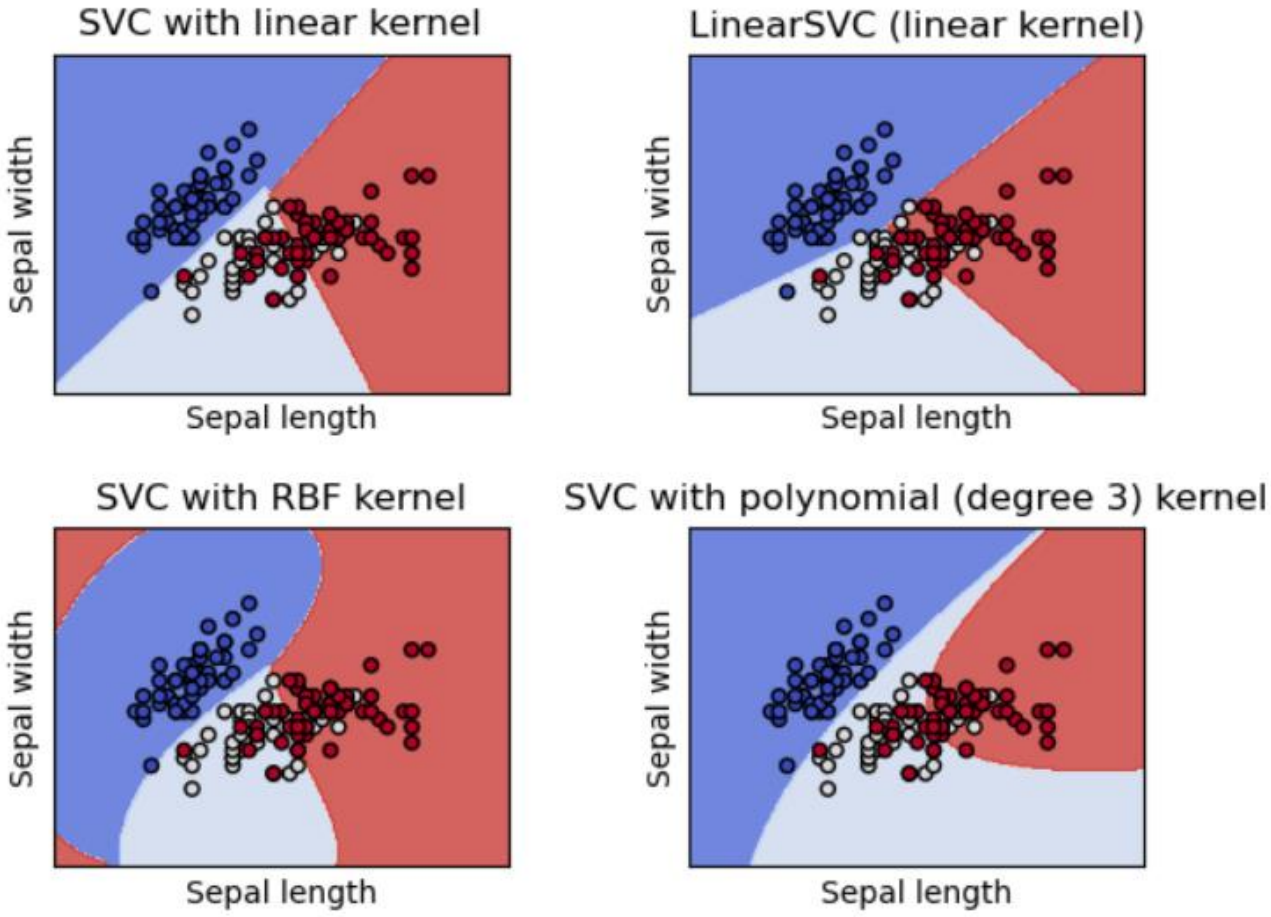
## Ядро RBF (радиальная базисная функция)

```
>>> svc = svm.SVC(kernel='rbf')
>>> # gamma: inverse of size of
>>> # radial kernel
```



https://scikit-learn.org/stable/auto_examples/svm/plot_svm_kernels.html#sphx-glr-auto-examples-svm-plot-svm-kernels-py

Для многоклассовой классификации результат будет таким:




https://scikit-learn.org/stable/modules/svm.html#tips-on-practical-use

Класс **LinearSVC** лучше работает с большими выборками в отличие от класса **SVC**, использует линейное ядро по умолчанию.

**Пример**: Рассчитаем модели SVC с разными ядрами на датасете «диабет», результаты неоднозначные:

```
# instantiate the model (using the default parameters)
model_SVC = SVC(kernel='poly',degree=3)

# fit the model with data
model_SVC.fit(X_train,y_train)

model_SVC.score(X_train, y_train)
```

```
0.7743055555555556
```

```
# calculate the predicted values
y_pred=model_SVC.predict(X_test)
```

```
# import the metrics class
from sklearn import metrics
```

**Вычислим confusion matrix (матрицу ошибок)**

```
cnf_matrix = metrics.confusion_matrix(y_test, y_pred)
cnf_matrix
```

```
array([[120,  10],
       [ 33,  29]], dtype=int64)
```

В таблице представлены результаты оценки моделей SVC с разными ядрами, (score и confusion matrix) :

| Линейное ядро | Полиномиальное степень 3 | RBFядро |
|---|---|---|
| 0.765625 | 0.7743 | 0.7638 |
| [117, 13]<br>[ 25, 37] | [120, 10]<br>[ 33, 29] | [119, 11]<br>[ 32, 30] |

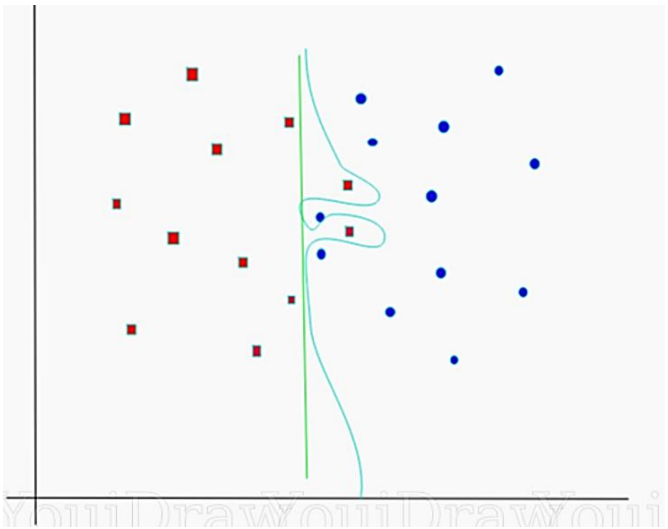
# Реализация в sklearn

*class* sklearn.svm.SVC(*, *C=1.0*, *kernel='rbf'*, *degree=3*, *gamma='scale'*, *coef0=0.0*, *shrinking=True*, *probability=False*, *tol=0.001*, *cache_size=200*, *class_weight=None*, *verbose=False*, *max_iter=- 1*, *decision_function_shape='ovr'*, *break_ties=False*, *random_state=None*)

**Настройка отельных параметров**

В данном примере есть несколько порогов принятия решений, которые можно определить для этой конкретной выборки. Например, если в качестве порога решений использовать прямую, то несколько объектов классифицируются неверно. Эти неверно классифицированные точки называются выбросами в данных.

Если настроить параметры таким образом, что в конечном итоге получим более изогнутую линию, то модель явно получается переученной (точно классифицирует все данные обучающей выборки, однако не сможет показать столь же хорошие результаты на новых данных).



## Параметр C

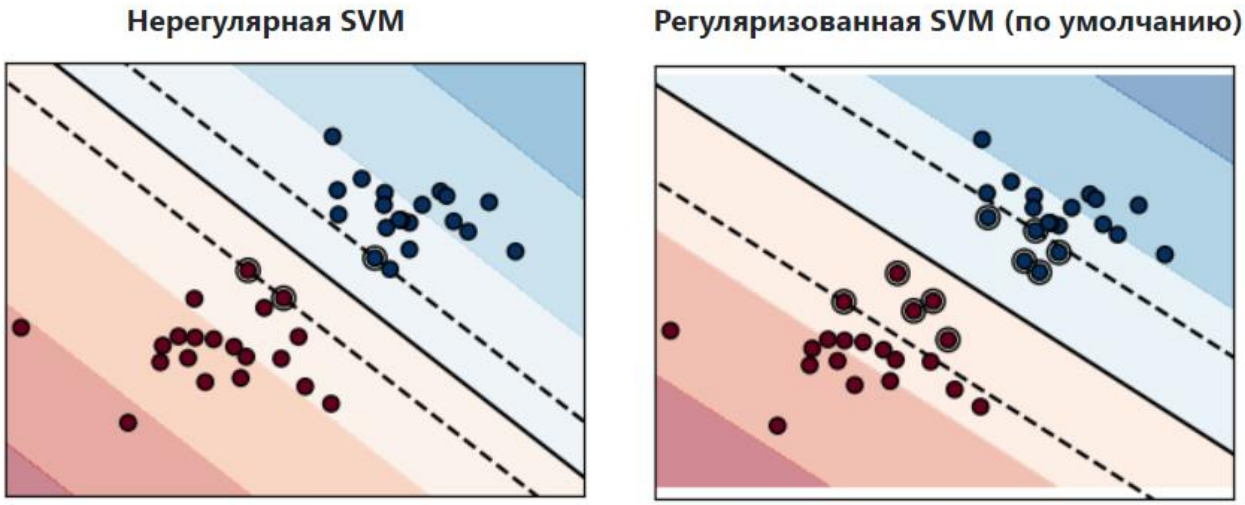
**C : *float, default=1.0***

Regularization parameter. The strength of the regularization is inversely proportional to C. Must be strictly positive. The penalty is a squared l2 penalty.

Параметр **C** помогает отрегулировать ту тонкую грань между гладкостью и точностью классификации объектов обучающей выборки. Чем больше значение C, тем больше объектов обучающей выборки будут правильно классифицированы.

Чем выше число **C** тем более запутанная гиперплоскость будет в вашей модели, но и выше число верно-классифицированных объектов обучающей выборки. Поэтому, важно “подкручивать” параметры модели под конкретный набор данных, чтобы избежать переобучения но, в то же время достигнуть высокой точности.

Иными словами, параметр C задает степень регуляризации: небольшое значение C означает, что зазор рассчитывается с использованием многих или всех наблюдений вокруг разделительной линии (большая регуляризация); большое значение для C означает, что запас рассчитывается для наблюдений, близких к разделительной линии (меньше регуляризации).




## Параметр Гамма

**gamma : *{'scale', 'auto'} or float, default='scale'***

Kernel coefficient for 'rbf', 'poly' and 'sigmoid'.

- if `gamma='scale'` (default) is passed then it uses 1 / (n_features * X.var()) as value of gamma,
- if 'auto', uses 1 / n_features.

Чем ниже гамма, тем больше элементов, даже тех, которые достаточно далеки от разделяющей гиперплоскости, принимают участие в процессе выбора идеальной разделяющей гиперплоскости. Если же, гамма высокая, тогда алгоритм будет “опираться” только на те элементы, которые наиболее близки к самой гиперплоскости.

Если задать уровень гаммы слишком высоким, тогда в процессе принятия решения о расположении линии будут участвовать только самые близкие к линии элементы. Это поможет игнорировать выбросы в данных. Алгоритм SVM устроен таким образом, что точки расположенные наиболее близко относительно друг друга имеют больший вес при принятии решения.

Однако при правильной настройке **C** и **gamma** можно добиться оптимального результата, который построит более линейную гиперплоскость, игнорирующую выбросы, и, следовательно, более обобщающую.

Рекомендуется использовать **GridSearchCV** с параметрами C и gamma для выбора хороших значений.

# Наивный байесовский классификатор

Наивный байесовский алгоритм (naive Bayes algorithm, НБА) – это алгоритм классификации, основанный на теореме Байеса с допущением о независимости признаков. Другими словами, НБА предполагает, что наличие какого-либо признака в классе не связано с наличием какого-либо другого признака. *Например, фрукт может считаться яблоком, если он красный, круглый и его диаметр составляет порядка 8 сантиметров. Даже если эти признаки зависят друг от друга или от других признаков, в любом случае они вносят независимый вклад в вероятность того, что этот фрукт является яблоком. В связи с таким допущением алгоритм называется «наивным».*

Модели на основе НБА достаточно просты и крайне полезны при работе с очень большими наборами данных. При своей простоте НБА способен превзойти даже некоторые сложные алгоритмы классификации.

Теорема Байеса позволяет рассчитать апостериорную вероятность P(c|x) на основе P(c), P(x) и P(x|c).

$$P(H_k \mid A) = \frac{P(H_k) * P(A \mid H_k)}{P(A)},$$

$P(H_k)$ - априорная вероятность события $H_k$
$P(H_k \mid A)$ - вероятность события $H_k$ при наступлении $A$
$P(A \mid H_k)$ - вероятность наступления $A$ при истинности $H_k$
$P(A)$ - полная вероятность события $A$

**Априорная вероятность** — распределение вероятностей, которое выражает предположения о вероятности события до учёта экспериментальных данных.

**Апостерио́рная вероя́тность** — условная вероятность случайного события при условии того, что известны апостериорные данные, то есть полученные после опыта.

# Пример работы наивного байесовского алгоритма.

http://datareview.info/article/6-prostyih-shagov-dlya-osvoeniya-naivnogo-bayesovskogo-algoritma-s-primerom-koda-na-python/

Обучающий набор данных, содержит один признак «Погодные условия» (weather) и целевую переменную «Игра» (play), которая обозначает возможность проведения матча. На основе погодных условий мы должны определить, состоится ли матч. Чтобы сделать это, необходимо выполнить следующие шаги.

Шаг 1. Преобразуем набор данных в частотную таблицу (frequency table).

| Weather | Play |
|---|---|
| Sunny | No |
| Overcast | Yes |
| Rainy | Yes |
| Sunny | Yes |
| Sunny | Yes |
| Overcast | Yes |
| Rainy | No |
| Rainy | No |
| Sunny | Yes |
| Rainy | Yes |
| Sunny | No |
| Overcast | Yes |
| Overcast | Yes |
| Rainy | No |

| Frequency Table | | |
|---|---|---|
| Weather | No | Yes |
| Overcast | | 4 |
| Rainy | 3 | 2 |
| Sunny | 2 | 3 |
| Grand Total | 5 | 9 |

Шаг 2. Создадим таблицу правдоподобия (likelihood table), рассчитав соответствующие вероятности. Например, вероятность облачной погоды (overcast) составляет 0,29, а вероятность того, что матч состоится (yes) – 0,64.

| Likelihood table | | | | |
|---|---|---|---|---|
| Weather | No | Yes | | |
| Overcast | | 4 | =4/14 | 0.29 |
| Rainy | 3 | 2 | =5/14 | 0.36 |
| Sunny | 2 | 3 | =5/14 | 0.36 |
| All | 5 | 9 | | |
| | =5/14 | =9/14 | | |
| | 0.36 | 0.64 | | |

Шаг 3. С помощью теоремы Байеса рассчитаем апостериорную вероятность для каждого класса при данных погодных условиях. Класс с наибольшей апостериорной вероятностью будет результатом прогноза.

**Задача**. Состоится ли матч при солнечной погоде (sunny)?

Мы можем решить эту задачу с помощью описанного выше подхода.

P(Yes | Sunny) = P(Sunny | Yes) * P(Yes) / P(Sunny)

Здесь мы имеем следующие значения:

P(Sunny | Yes) = 3 / 9 = 0,33

P(Sunny) = 5 / 14 = 0,36

P(Yes) = 9 / 14 = 0,64

Теперь рассчитаем P(Yes | Sunny):

P(Yes | Sunny) = 0,33 * 0,64 / 0,36 = 0,60

Значит, при солнечной погоде более вероятно, что матч состоится.

Аналогичным образом с помощью НБА можно прогнозировать несколько различных классов на основе множества признаков. Этот алгоритм в основном используется в области классификации текстов и при решении задач многоклассовой классификации.

## Достоинства НБА:

- ✓ Классификация, в том числе многоклассовая, выполняется легко и быстро.
- ✓ Когда допущение о независимости выполняется, НБА превосходит другие алгоритмы, такие как логистическая регрессия (logistic regression), и при этом требует меньший объем обучающих данных.
- ✓ НБА лучше работает с категорийными признаками, чем с непрерывными. Для непрерывных признаков предполагается нормальное распределение, что является достаточно сильным допущением.

## Недостатки НБА:

- ✓ Если в тестовом наборе данных присутствует некоторое значение категорийного признака, которое не встречалось в обучающем наборе данных, тогда модель присвоит нулевую вероятность этому значению и не сможет сделать прогноз. Это явление известно под названием «нулевая частота» (zero frequency). Данную проблему можно решить с помощью сглаживания, например сглаживание по Лапласу (Laplace smoothing).
- ✓ Хотя НБА является хорошим классификатором, значения спрогнозированных вероятностей не всегда являются достаточно точными. Поэтому не следует слишком полагаться на результаты, возвращенные методом predict_proba.
- ✓ Еще одним ограничением НБА является допущение о независимости признаков. В реальности наборы полностью независимых признаков встречаются крайне редко.

## Варианты применения наивного байесовского алгоритма

**Классификация в режиме реального времени**. НБА очень быстро обучается, поэтому его можно использовать для обработки данных в режиме реального времени.

**Многоклассовая классификация**. НБА обеспечивает возможность многоклассовой классификации. Это позволяет прогнозировать вероятности для множества значений целевой переменной.

**Классификация текстов, фильтрация спама, анализ тональности текста**. При решении задач, связанных с классификацией текстов, НБА превосходит многие другие алгоритмы. Благодаря этому, данный алгоритм находит широкое применение в области фильтрации спама (идентификация спама в электронных письмах) и анализа тональности текста (анализ социальных медиа, идентификация позитивных и негативных мнений клиентов).

**Рекомендательные системы**. Наивный байесовский классификатор в сочетании с коллаборативной фильтрацией (collaborative filtering) позволяет реализовать рекомендательную систему. В рамках такой системы с помощью методов машинного обучения и интеллектуального анализа данных новая для пользователя информация отфильтровывается на основании спрогнозированного мнения этого пользователя о ней.

В библиотеке scikit-learn реализованы следующие типы моделей на основе наивного байесовского алгоритма:

- ✓ **GaussianNB.** Основан на использовании распределения. Поскольку распределение Гаусса является непрерывным, этот классификатор поддерживает функции с непрерывными значениями.
- ✓ **MultinomialNB.** Использует функции полиномиального распределения. Полиномиальное (мультиномиальное) распределение - совместное распределение вероятностей случайных величин, каждая из которых есть число появлений одного из нескольких взаимно исключающих событий.

Используется в случае дискретных признаков. Например, в задаче классификации текстов признаки могут показывать, сколько раз каждое слово встречается в данном тексте.

- ✓ **BernoulliNB.** Основан на использовании распределения Бернулли. Используется в случае двоичных дискретных признаков (могут принимать только два значения: 0 и 1). Например, в задаче классификации текстов с применением подхода «мешок слов» (bag of words) бинарный признак определяет присутствие (1) или отсутствие (0) данного слова в тексте.

Случайная величина X имеет **распределение Бернулли**, если она принимает всего два значения: 1 и 0 с вероятностями p и q = 1-p соответственно.

✓ **CategoricalNB.** Реализует категориальный наивный алгоритм Байеса для категориально распределенных данных. Предполагается, что каждая функция, описываемая индексом, имеет свое категориальное распределение.

Наивные байесовские модели могут использоваться для решения крупномасштабных задач классификации, для которых полный обучающий набор может не поместиться в памяти. Чтобы справиться с этим делом, MultinomialNB, BernoulliNBи GaussianNB выставить partial_fitметод , который может быть использован пошагово.

Все наивные байесовские классификаторы поддерживают взвешивание выборки.

ВЗВЕШЕННАЯ ВЫБОРКА (WEIGHTED SAMPLE) — выборка, которая не является строго пропорциональной распределению классов в популяции. Взвешенная выборка откорректирована таким образом, что включает бОльшие по сравнению с другими доли некоторых классов всей популяции. Эти классы, получившие больший «вес», в противном случае не были бы достаточно представлены в выборке в численном выражении, для того чтобы привести к обобщаемым выводам. Другая причина для применения взвешенной выборки — некоторые классы считаются более важными, более интересными, более стоящими детального исследования и т.п.

**Пример:**

| d | Текст | Класс |
|---|---|---|
| 1 | котики такие мокрые | мимими |
| 2 | котики котики няшки | мимими |
| 3 | пушистые котики | мимими |
| 4 | мокрые морские свинки | не мимими |
| 5 | котики мокрые морские котики | ??? |

С помощью алгоритма MultinomialNB вычислить $p(\text{мимими}|d_5)$

$C_1$ – «мимими», $C_2$ – «не мимими»,

$P(C_1) = ¾$, $P(C_2) = ¼$

P(котики| $C_1$)= ½, P(котики| $C_2$)= 0,

P(мокрые| $C_1$)= 1/8, P(мокрые| $C_2$)= 1/3,

P(свинки| $C_1$)= 0, P(свинки| $C_2$)= 1/3,

P(морские| $C_1$)= 0, P(морские| $C_2$)= 1/3,

**$P(C_1|d_5)$** = ¾* ½ *1/8*1* ½ = **3/128**

**$P(C_2|d_5)$** = ¼ * 1 *1/8*1/8* 1 = **1/256**