Support Vector Machines

1. Support Vector Machines

1.1 Определение

Support Vector Machines (SVM) са мощен клас алгоритми за машинно обучение, използвани основно за задачи с **класификация**, но също така могат да се прилагат и за регресия. Основната идея зад SVM е да намери **оптимална хиперплоскост** (hyperlane), която разделя данните на различни класове с максимално възможно разстояние (наречено **margin**) между най-близките точки от всеки клас, наречени **поддържащи вектори** (support vectors).
Целта на SVM е да намери тази хиперплоскост така, че да минимизира грешките при класифицирането на нови, непознати данни.

1.2 Принцип на работа

- **Линейна класификация:** При линейно разделими данни SVM намира най-добрата линия (или хиперплоскост (hyperlane) в многомерно пространство), която разделя точките от различните класове с максимален margin.
- **Нелинейна класификация:** За по-сложни, нелинейно разделими данни, SVM използва **kernel trick** (ядрена функция), която трансформира данните в по-високоизмерно пространство, където те могат да бъдат линейно разделени.

1.3 Добри примери за използване на SVM:

1. Класификация на изображения
2. **Разпознаване на лица**

1.4 Основни предимства на SVM:

- **Висока производителност** при работата с високоизмерни пространства (когато броят на признаците е голям).
- **Ефективен при малки набори от данни** с голям брой признаци.
- **Гъвкавост чрез kernel trick**, позволяващ решаване на нелинейни проблеми.

1.5 Ограничения на SVM:

- **Нуждае се от големи ресурси** при работа с много големи набори от данни.

- Изборът на **подходяща ядрена функция** (kernel) може да бъде труден и критичен за успеха на модела.
- **Чувствителност към шума** – ако данните са прекалено шумни (т.е. имат грешни или объркани етикети), SVM може да не работи добре.

2. Kernel trick

2.1 Определение

Kernel trick е техника, използвана в Support Vector Machines (SVM) и други алгоритми за машинно обучение, която позволява работа с **нелинейно разделими** данни, като проектира данните в **по-високоизмерно пространство**, където те могат да бъдат линейно разделени.

SVM се основава на намирането на **линейна хиперплоскост**, която разделя различни класове данни. Когато данните са **линейно разделими**, намирането на тази хиперплоскост (hyperlane) е сравнително лесно.
Но ако данните не могат да бъдат разделени с права линия (или хиперплоскост (hyperlane) в по-високо измерение), тогава стандартният SVM няма да работи добре.
Пример за нелинейно разделими данни може да бъде концентрични кръгове или всякакъв друг сложен модел.

2.2 Как Kernel trick работи?

1. **Трансформация на данни в по-високоизмерно пространство**: Вместо експлицитно да се извършва трансформация на всеки елемент, Kernel trick използва **ядрена функция** (kernel function), за да изчисли вътрешния продукт между две точки директно в по-високоизмерното пространство, без да е необходимо да ги проектираме там.
2. **Вътрешен продукт (dot product)**: В много алгоритми, включително SVM, резултатът зависи само от вътрешните произведения между различни точки. Kernel trick позволява тези вътрешни произведения да бъдат изчислени ефективно, без действително да преминаваме през процеса на проектиране на данните в по-високоизмерно пространство.

2.3 Пример за ядра (kernels)

1. **Линейно ядро (Linear Kernel)**:
   Това е най-простото ядро, което се използва, когато данните са линейно разделими. То просто пресмята обичайния вътрешен продукт на две точки.

2. **Полиномиално ядро (Polynomial Kernel)**:
   Трансформира данните в полиномиално пространство и е полезно, когато връзката между признаците и изхода е полиномиална.
3. **Радиално-базисно ядро (Radial Basis Function, RBF) или Гаусово ядро**:
   Едно от най-популярните ядра за нелинейни задачи. То трансформира точките по начин, който дава възможност на модела да разделя данни, които не могат да бъдат разделени в първоначалното им пространство.
4. **Сигмоидно ядро (Sigmoid Kernel)**:
   Използва се в невронни мрежи и имитира активираща функция на невроните.

3. Hinge loss

3.1 Определение

Hinge loss е функция за загуба, която се използва основно в задачите за **класификация** с алгоритми като **Support Vector Machines (SVM)**. Тази функция измерва колко добре са класифицирани данните и се стреми да максимизира **margin**-а между класовете, като същевременно минимизира грешките.
Hinge loss се използва за обучение на модели, които трябва да разделят данните на два класа (обикновено обозначени като +1 и −1). Тя е базирана на идеята, че не е достатъчно просто да класифицираме точките правилно, а че правилно класифицираните точки трябва да бъдат **на достатъчно голямо разстояние** (margin) от хиперплоскостта (hyperlane) на разделяне.

3.2 Как работи hinge loss?

- Ако $yi \cdot f(xi) \geq 1$, това означава, че точката е **правилно класифицирана с достатъчен margin**, така че загубата е 0.
- Ако $yi \cdot f(xi) < 1$ това означава, че точката или е класифицирана **грешно**, или е **правилно класифицирана, но е твърде близо до границата (хиперплоскостта)**. В този случай има ненулева загуба.

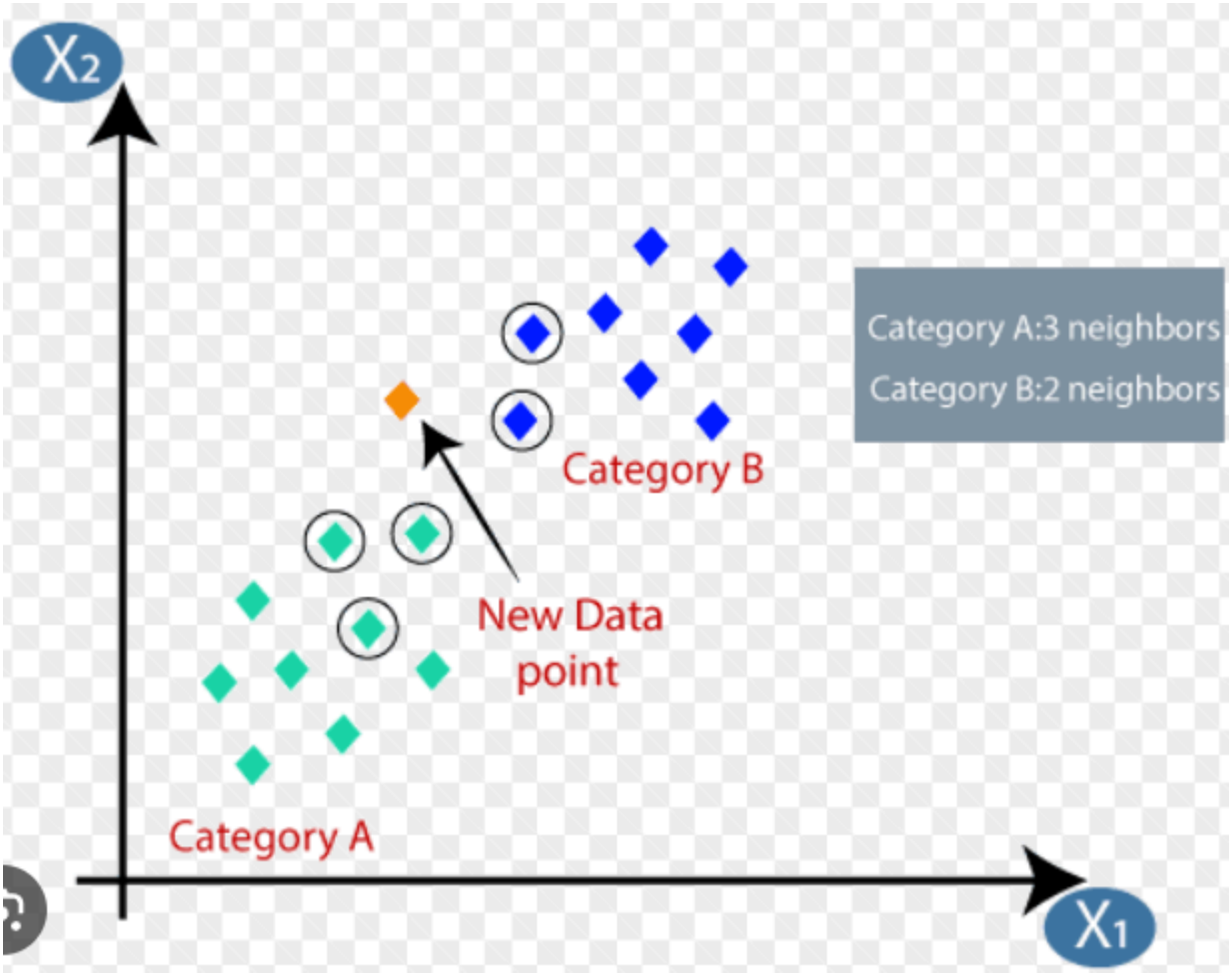
4. make_circles

Функцията `make_circles` от библиотеката `scikit-learn` в Python се използва за генериране на **двумерни данни**, които образуват два концентрични кръга. Тези данни са често използвани за тестове на алгоритми за класификация, особено когато се търси решение за **нелинейно разделими данни**.

5. SVC()

`SVC()` е клас от библиотеката `scikit-learn`, който имплементира **Support Vector Classification (SVC)**, т.е. **поддържащи вектори за класификация**. Този клас позволява изграждането на класификационен модел, който използва различни ядра за разделяне (KERNELS) на данните на различни класове.

6. k-Nearest neighbours



6.1 Определение

K-Nearest Neighbors (K-NN) е прост, но мощен класификационен и регресионен алгоритъм, базиран на сходството между примери от данните. Той не изисква обучение в традиционния смисъл, тъй като моделът запазва всички тренировъчни примери и прави предсказания на база на **най-близките съседи**.

anomaly detection
OneClassSVM
partial fit