

# دوره‌های AI

## جلسه دوم

محمد رضا بر جیان



## کلاسیفایر بیزین

$$P(A|B) = \frac{P(B|A) P(A)}{P(B)}$$

01

مزایا:

- سادگی و سرعت
- نیاز به داده های کوچک
- مدیریت داده های نویز و غیر کامل
- کاربرد در مسائل واقعی

02

معایب:

- فرض استقلال ویژگی
- پاسخ ضعیف به ویژگی های وابسته
- محدودیت در داده های پیوسته

03

فرمول:

احتمال اینکه کلاس های داده شده به شرط لیبل وجود داشته باشند  $\times$  احتمال لیبل مورد نظر / احتمال کلاس ها

04

# مثال

مجموعه داده‌ای داریم که شرایط آب و هوایی برای بازی گلف را شرح می‌دهد. با توجه به شرایط آب و هوایی شرایط برای بازی گلف مناسب (بله) یا نامناسب (خیر) طبقه‌بندی می‌شود.

آب و هوای (Outlook)  
(Rainy), (Sunny) (Overcast)  
دما (Temperature)  
گرم (Hot) یا خنک (Mild) (Cool)  
رطوبت (Humidity)  
نرمال (Normal) بالا (High)  
باد (Windy)  
(True) (False)

أب و هو

	Yes	No	P(Yes)	P(No)
Sunny	2	3	29	35
Overcast	4	0	49	05
Rainy	3	2	39	25
Total	9	5	100%	100%

الحرارة

	Yes	No	P(Yes)	P(No)
Hot	2	2	29	25
Mild	4	2	49	25
Cool	3	1	39	15
Total	9	5	100%	100%

الرطوبة

	Yes	No	P(Yes)	P(No)
High	3	4	39	45
Normal	6	1	69	15
Total	9	5	100%	100%

ال�ين

	Yes	No	P(Yes)	P(No)
False	6	2	69	25
True	3	3	39	35
Total	9	5	100%	100%

Play		P(Yes)/P(No)
Yes	No	
9	5	9/14
No	5	5/14
Total	14	100%



حال فرض کنید باید نمونه‌ی جدید X را طبقه‌بندی کنیم که اطلاعات آن به‌این شرح است:

**Outlook = sunny**

**Temperature = cool**

**Humidity = high**

**Wind = true**

یعنی هوا آفتابی، دما خنک، رطوبت زیاد و باد هم وجود دارد.

بنابراین احتمال بازی گلف به‌این شرح است:

برای شرایطی که این داده جدید دارد هم احتمال بازی‌کردن و هم احتمال بازی‌نکردن را از روی جدول بالا بار دیگر منویسیم:

$$P(\text{outlook} = \text{sunny} \mid \text{play} = \text{yes}) = 2/9$$

$$P(\text{temperature} = \text{cool} \mid \text{play} = \text{yes}) = 3/9$$

$$P(\text{Humidity} = \text{high} \mid \text{play} = \text{yes}) = 3/9$$

$$P(\text{wind} = \text{true} \mid \text{play} = \text{yes}) = 3/9$$

$$P(\text{play} = \text{yes}) = 9/14$$

$$P(\text{outlook} = \text{sunny} \mid \text{play} = \text{no}) = 3/5$$

$$P(\text{temperature} = \text{cool} \mid \text{play} = \text{no}) = 1/5$$

$$P(\text{Humidity} = \text{high} \mid \text{play} = \text{no}) = 4/5$$

$$P(\text{wind} = \text{true} \mid \text{play} = \text{no}) = 3/5$$

$$P(\text{play} = \text{no}) = 5/14$$

حال با توجه به فرمول بیز ساده، اول احتمال این را که بازی صورت گیرد یا نگیرد برای این داده‌ی جدید، یعنی  $X$ ، محاسبه می‌کنیم. به این صورت که یک بار احتمال هر یک از ویژگی‌ها (باد، رطوبت، دما، هوا) را برای صورت‌گرفتن بازی که در بالا هم بار دیگر از روی جدول برای راحتی کار نوشتیم و احتمال  $P(\text{Yes}) = 9/14$  در هم ضرب می‌کنیم؛ سپس بار دیگر بار احتمال هر یک از ویژگی‌ها (باد، رطوبت، دما، هوا) را برای صورت‌نگرفتن بازی و احتمال  $P(\text{No}) = 5/14$  در هم ضرب می‌کنیم

$$P(X \mid \text{play} = \text{yes}).P(\text{play} = \text{yes}) = 2/9 * 3/9 * 3/9 * 3/9 * 9/14 = 0.0053$$

$$0.0206 = 5/14 * 3/5 * 1/5 * 4/5 * 3/5 = (P(X \mid \text{play} = \text{no}).P(\text{play} = \text{no}))$$

حال که مقدار  $(c)$  را داریم، باید اول مقدار  $P(X)$  را به دست آوریم و درنهایت طبق فرمول بیز،  $P(c \mid x) P(x) / P(c)$  را بر  $P(c)$  تقسیم کنیم:

$$(P(X) = P(\text{Outlook} = \text{sunny}) * P(\text{Humidity} = \text{High}) * P(\text{Temperature} = \text{Cool}) * P(\text{Wind} = \text{True})$$

$$0.02186 = 6/14 * 7/14 * 4/14 * 5/14 =$$

$$P(\text{play} = \text{Yes} \mid X) = (P(X \mid \text{play} = \text{yes}).P(\text{play} = \text{yes})) / P(X) = 0.2424$$

$$P(\text{play} = \text{No} \mid X) = (P(X \mid \text{play} = \text{no}).P(\text{play} = \text{no})) / P(X) = 0.9421$$

این یعنی با احتمال **0.9421**، پیش‌بینی ما در مورد بازی گلف «نه» است؛ یعنی بازی صورت نمی‌گیرد

# KNN

## همایه نزدیک k



این الگوریتم بر اساس داده‌های نزدیک به نمونه جدید صورت می‌گیرد.

مزایا:

- سادگی و سرعت
- میتوانه برای رگرسن و کلاسیفیکیشن اجرا بشه

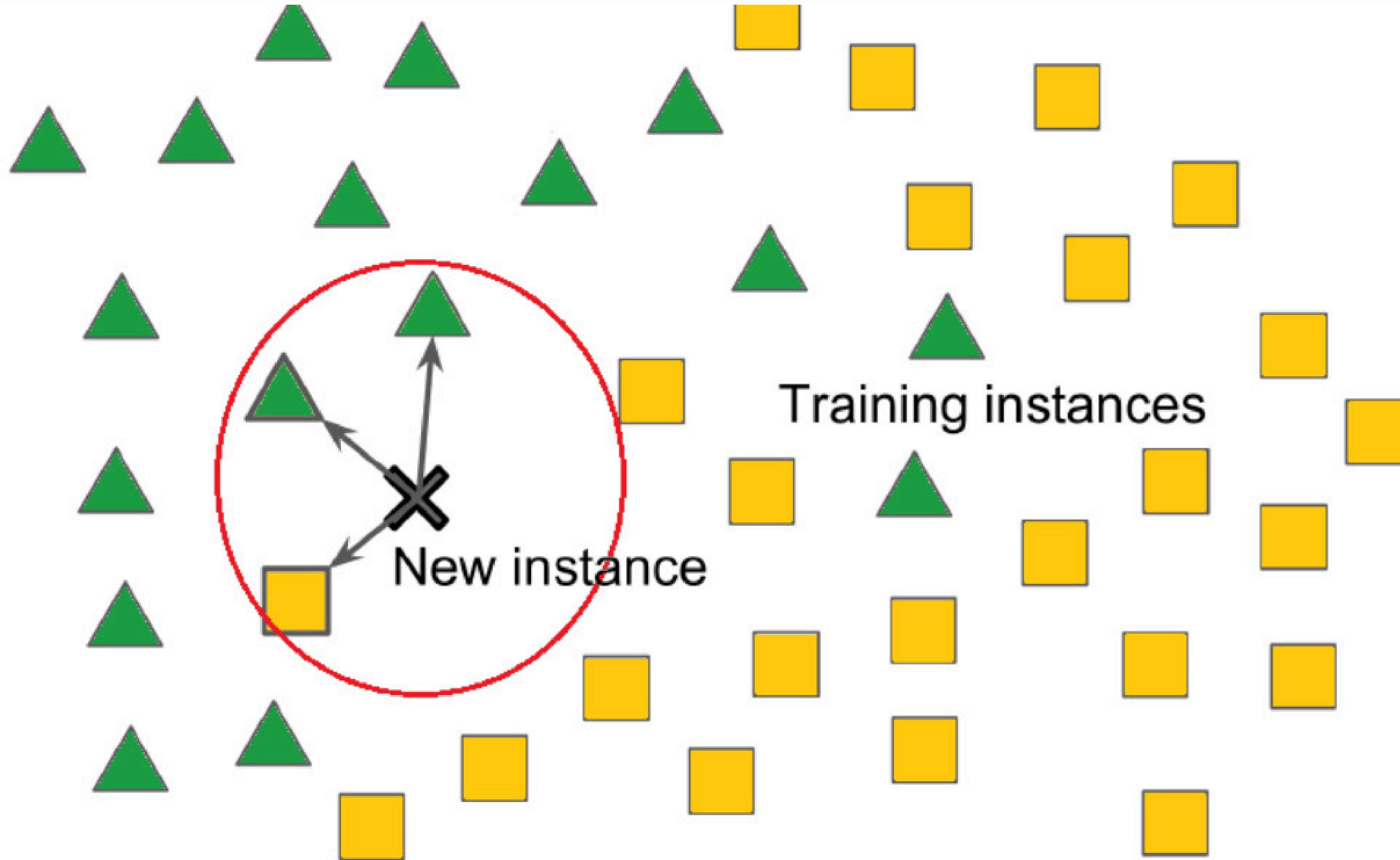
معایب:

در داده‌ها با فیچرهای زیاد به خوبی کار نمی‌کند  
به داده‌های پرت حساس است و نویز میتوانه اختلال  
ایجاد کنه.

برای داده‌های بزرگ، محاسبه تمام فاصله‌ها وقت  
گیر است

کاربردها:

KNN به طور گسترده‌ای در دسته‌بندی تصاویر،  
تشخیص نویسه‌ها، سیستم‌های توصیه‌گر، و  
شناسایی الگوها استفاده می‌شود.



## نحوه عملکرد

### k همسایه نزدیک

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

01

مرحله اول داده را مشخص میکنیم که کجاست.  
فاصله را از تک نقطه هارو به دست میاریم

02

فواصل را از کم به زیاد مشخص میکنیم. میگردد  
دنبال تعداد همسایه هایی که گفتیم بعد میبینه به  
کدام کلاس نزدیکه.

03

با زیاد شدن تعداد داده ها سرعت میاد پایین  
داده کافی نباشه با زیاد شدن K به مشکل میخوریم.

04

کاربردها:  
KNN به طور گسترده‌ای در دسته‌بندی تصاویر،  
تشخیص نوبیسه‌ها، سیستم‌های توصیه‌گر، و  
شناسایی الگوها استفاده می‌شود.

# درخت تصمیم (Decision Tree)

## درخت تصمیم

01

یک الگوریتم یادگیری ماشین است که برای حل مسائل دسته‌بندی و رگرسیون استفاده می‌شود. این الگوریتم با ایجاد یک ساختار درختی، به صورت سلسله مراتبی تصمیم‌گیری می‌کند و داده‌ها را به صورت گام به گام به کلاس‌ها یا مقادیر خروجی خاص تقسیم می‌کند.

02

- ساختار درختی به راحتی قابل درک است و حتی برای تصمیم‌گیری‌های تجاری کاربرد دارد.
- درخت‌های تصمیم معمولاً به پردازش اولیه زیادی نیاز ندارند و حتی با داده‌های طبقه‌بندی شده و بدون نیاز به نرمال‌سازی کار می‌کنند.

03

- مشکل Overfitting: درخت‌های تصمیم معمولاً روی داده‌های آموزشی بیش از حد تطبیق پیدا می‌کنند، که با روش‌هایی مثل هرس‌کردن (Pruning) یا استفاده از الگوریتم‌های جمعی Random Ensemble (مانند جنگل تصادفی (Random Forest) برطرف می‌شود.
- حساسیت به نویز: درخت‌های تصمیم به داده‌های پرت حساس هستند و ممکن است دقتشان کاهش یابد.

# ID3

یک الگوریتم یادگیری ماشین مبتنی بر درخت تصمیم است که برای دسته‌بندی



محاسبه آنتروپی: ابتدا آنتروپی داده‌ها را محاسبه می‌کند که معیاری برای اندازه‌گیری عدم اطمینان در داده‌ها است. آنتروپی بیشتر نشان‌دهنده توزیع یکنواخت داده‌ها بین کلاس‌ها و آنتروپی کمتر نشان‌دهنده خلوص بالای یک دسته است.



انتخاب ویژگی با بیشترین اطلاعات: برای هر ویژگی، میزان شاخص اطلاعات را محاسبه می‌کند که تعیین می‌کند کدام ویژگی بیشترین کاهش آنتروپی (بیشترین اطلاعات) را به وجود می‌آورد. این ویژگی برای تقسیم داده‌ها در گره فعلی انتخاب می‌شود.



تقسیم داده‌ها: داده‌ها بر اساس این ویژگی تقسیم می‌شوند و یک گره جدید به درخت اضافه می‌شود. این کار به صورت بازگشتی برای هر گره فرعی ادامه پیدا می‌کند تا در نهایت به گره‌های خالص (یا تقریباً خالص) برسد.



ساخت گره برگ: زمانی که داده‌های یک گره به طور کامل به یک کلاس خاص تعلق داشته باشند یا دیگر ویژگی قابل تفکیکی وجود نداشته باشد، گره برگ ساخته می‌شود و درخت کامل می‌شود.

Input attributes or Decision factors				result
	Sky	barometer	Wind	Rain
1	clear	rising	north	-
2	cloudy	rising	south	+
3	cloudy	steady	north	+
4	clear	falling	north	-
5	cloudy	falling	north	+
6	cloudy	rising	north	+
7	cloudy	falling	south	-
8	clear	rising	south	-

مثال: استفاده از الگوریتم ID3 برای پیش‌بینی بارش باران بر اساس متغیرهای ورودی (فاکتورهای) زیر:



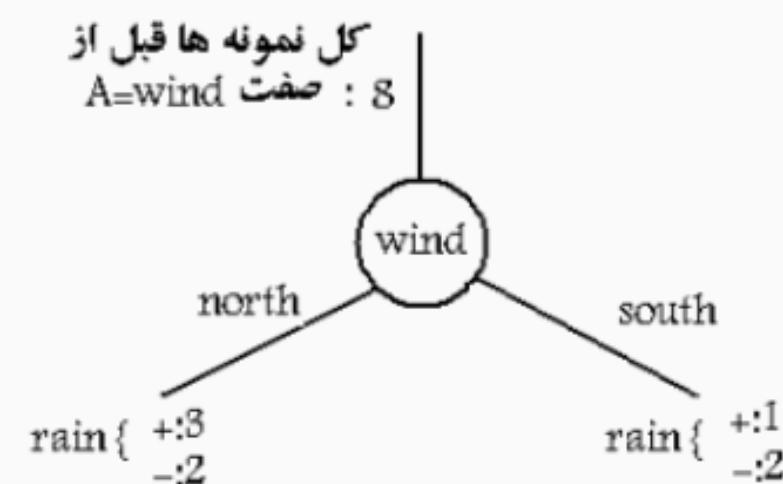
حال با صفات (فاکتورهای) wind , Sky, Barometer مواجه ایم. برای هر کدام از این صفات باید Gain را محاسبه کنیم.

عدم قطعیت در کل نمونه ها  
یا  
تعداد بیتها برای بیان عدم قطعیت

$$\begin{aligned}
 H(I) &= -\sum_{i=1}^2 \frac{\#classi}{\#I} \log_2 \frac{\#classi}{\#I} \\
 &= -\frac{4}{8} \log_2 \frac{4}{8} - \frac{4}{8} \log_2 \frac{4}{8} \\
 &= 1bit
 \end{aligned}$$

$$\begin{aligned}
 H(I, \text{wind}) &= \sum_{i=1}^2 \frac{\#classi}{\#I} H(classi) \\
 &= \frac{\#class(north)}{\#I} H(north) + \frac{\#class(south)}{\#I} H(south) \\
 &= \frac{5}{8} H(north) + \frac{3}{8} H(south) \quad \textcircled{1}
 \end{aligned}$$

حالت اول: A=wind :



# ID3

مفاهیم کد نویسی



ماکسیمم عمق: یعنی بعد از گره اصلی چند گره  
بیایم پایین



مینیمم سمپل پر لیف. درخت ما که تصمیم گیری  
میکنه حداقل چند نمونه باید تفکیک کنه؟ اگر کم  
بدیم آورفیت میشه



مین سمپل اسپلیت: تعداد نمونه ها برای  
اسپلیت کردن چندتا باشه؟ مثلا چون رسید به ۱۱



ساخت گره برگ: زمانی که داده های یک گره به طور  
کامل به یک کلاس خاص تعلق داشته باشند یا دیگر  
ویژگی قابل تفکیکی وجود نداشته باشد، گره برگ  
ساخته می شود و درخت کامل می شود.

به عنوان مثال، اگر  **$\text{min\_samples\_split} = 5$**  باشد، و **7** نمونه در یک گره داخلی وجود داشته باشد، تقسیم مجاز است. اما فرض کنید که تقسیم به دو برگ منجر می شود، یکی با **1** نمونه و دیگری با **6** نمونه. اگر  **$\text{min\_samples\_leaf} = 2$**  باشد، تقسیم مجاز نخواهد بود (حتی اگر گره داخلی **7** نمونه داشته باشد) زیرا یکی از برگ های به دست آمده کمتر از حداقل تعداد نمونه های مورد نیاز برای قرار گرفتن در یک گره برگ خواهد بود.

# Random Forest

مفاهیم کد نویسی

جنگل تصادفی به جای استفاده از یک درخت تصمیم، از مجموعه‌ای از درخت‌های تصمیم (معمولًاً صدها یا هزاران درخت) استفاده می‌کند و نتیجه نهایی را براساس نظر اکثیریت یا میانگین خروجی‌های درخت‌ها به دست می‌آورد.

نمونه‌برداری با جایگذاری (Bootstrap Sampling): برای ساخت هر درخت تصمیم، به جای استفاده از کل داده، یک نمونه تصادفی از داده‌ها (با جایگذاری) گرفته می‌شود. این روش که به نام Bagging نیز شناخته می‌شود، باعث افزایش تنوع و کاهش واریانس مدل می‌شود.

انتخاب تصادفی ویژگی‌ها: در هر گره از درخت، به جای استفاده از تمام ویژگی‌ها برای انتخاب شرط تقسیم‌بندی، یک زیرمجموعه تصادفی از ویژگی‌ها انتخاب می‌شود و درخت فقط از آن ویژگی‌ها استفاده می‌کند. این انتخاب تصادفی باعث افزایش تنوع در درخت‌ها و کاهش خطر Overfitting می‌شود.

مراحل بالا برای ساخت تعداد زیادی درخت تصمیم تکرار می‌شود. هر درخت تصمیم به صورت مستقل و با داده و ویژگی‌های متفاوت آموخته می‌بیند.



# Random Forest

مفاهیم کد نویسی

- برای دسته‌بندی: جنگل تصادفی با گرفتن رای اکثیریت درخت‌ها برای هر نمونه، کلاس نهایی را انتخاب می‌کند.
  - برای رگرسیون: خروجی‌ها با میانگین‌گیری از نتایج تمام درخت‌ها به دست می‌آیند.
- مزایا
- دقیق‌تر: با ترکیب نتایج چندین درخت، جنگل تصادفی می‌تواند مدل بسیار دقیقی ایجاد کند.
  - کاهش Overfitting: به دلیل استفاده از چندین درخت مستقل، احتمال Overfitting کاهش می‌یابد.
  - مقاومت در برابر نویز: این مدل به داده‌های پرت و نویز حساسیت کمتری نشان می‌دهد.
- زمان و منابع بیشتر: به دلیل ساخت تعداد زیادی درخت، این مدل به زمان و حافظه بیشتری نیاز دارد.
  - قابلیت تفسیر کم: تفسیر نتیجه جنگل تصادفی به دلیل استفاده از تعداد زیادی درخت تصمیم دشوارتر از یک درخت تصمیم واحد است.



# SVM

## ماشین بردار پشتیبان

- 
- 01
  - 02
  - 03
  - 04

SVM به دنبال پیدا کردن خطی است که بهترین جداسازی را بین کلاس‌های مختلف داده‌ها انجام دهد. این خط به گونه‌ای انتخاب می‌شود که فاصله آن از نزدیک‌ترین نقاط هر کلاس (به نام نقاط پشتیبان یا support vectors) حداقل باشد.

این نقاط، نزدیک‌ترین داده‌ها به خط هستند و نقش مهمی در تعیین موقعیت آن دارند. تنها این نقاط هستند که در تعیین خط تأثیرگذارند و داده‌های دیگر تأثیر کمتری دارند.

SVM می‌تواند به وسیله استفاده از توابع هسته (Kernel Functions) به ابعاد بالاتر منتقل شود تا داده‌های غیرقابل تفکیک خطی را جداسازی کند. این توابع به SVM اجازه می‌دهند تا شکل‌های پیچیده‌تری از مرزهای جداسازی را ایجاد کند.

- عملکرد بالا: SVM معمولاً در مواجهه با داده‌های با ابعاد بالا و تعداد کم نقاط داده، عملکرد خوبی دارد.
- قابلیت تعمیم: به خوبی می‌تواند روی داده‌های جدید پیش‌بینی کند.
- کنترل بر سرشت تعادل: SVM دارای یک پارامتر (C) است که می‌تواند به کنترل دقیق مدل در حین طبقه‌بندی کمک کند.

## **(Linear Kernel) ۱**

این کرنل برای داده‌هایی که به‌خوبی با یک خط (در دو بعد) یا یک صفحه (در ابعاد بالاتر) قابل تفکیک هستند، استفاده می‌شود.

## **(Polynomial Kernel) ۲**

این کرنل برای داده‌هایی که به‌صورت غیرخطی ولی با روابط چندجمله‌ای قابل تفکیک هستند، مناسب است.

## **(RBF Kernel) یا Gaussian Kernel ۳**

این کرنل یکی از محبوب‌ترین و پرکاربردترین کرنل‌ها است و برای داده‌های غیرخطی استفاده می‌شود.  $\sigma$  پارامتری است که تأثیر شعاعی بر روی داده‌ها دارد.

## **(Sigmoid Kernel) ۴**

این کرنل بر اساس تابع سیگموئید ساخته شده است و به عنوان تابع هسته‌ای در برخی از شبکه‌های عصبی نیز استفاده می‌شود.

## **(Laplacian Kernel) ۵**

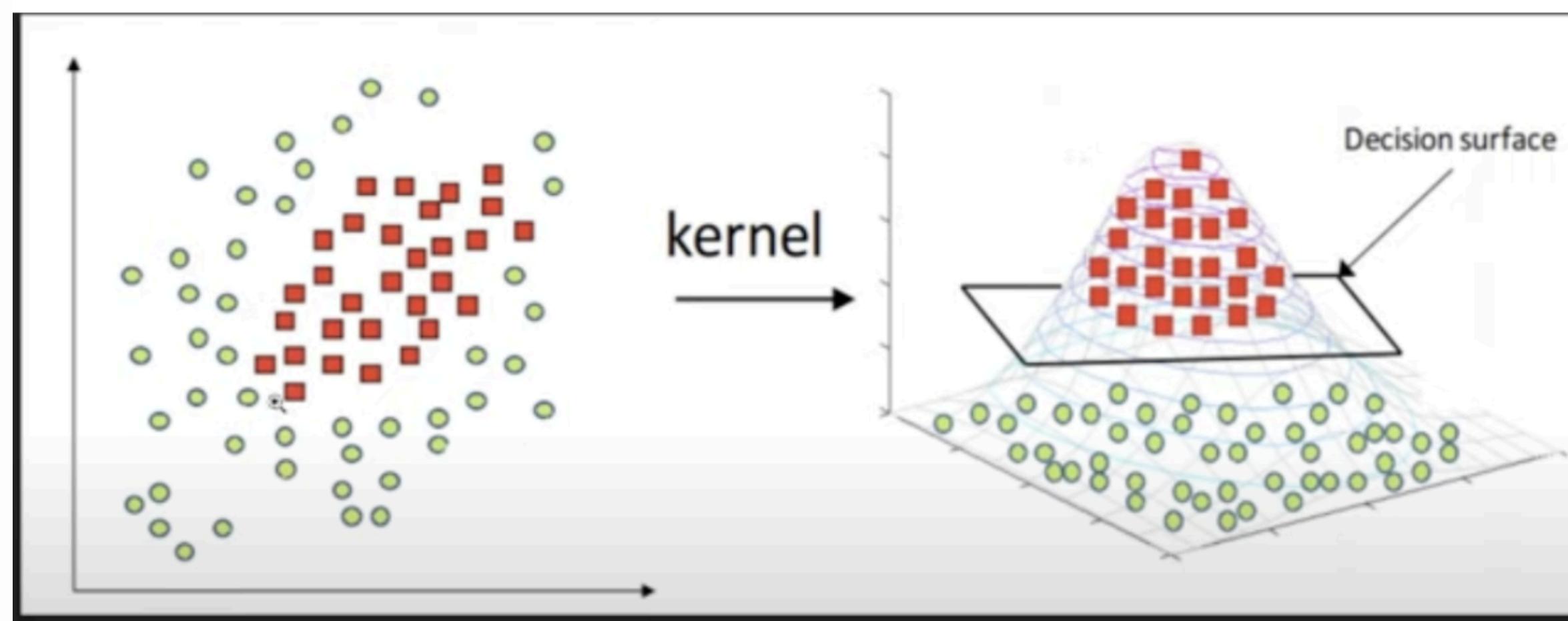
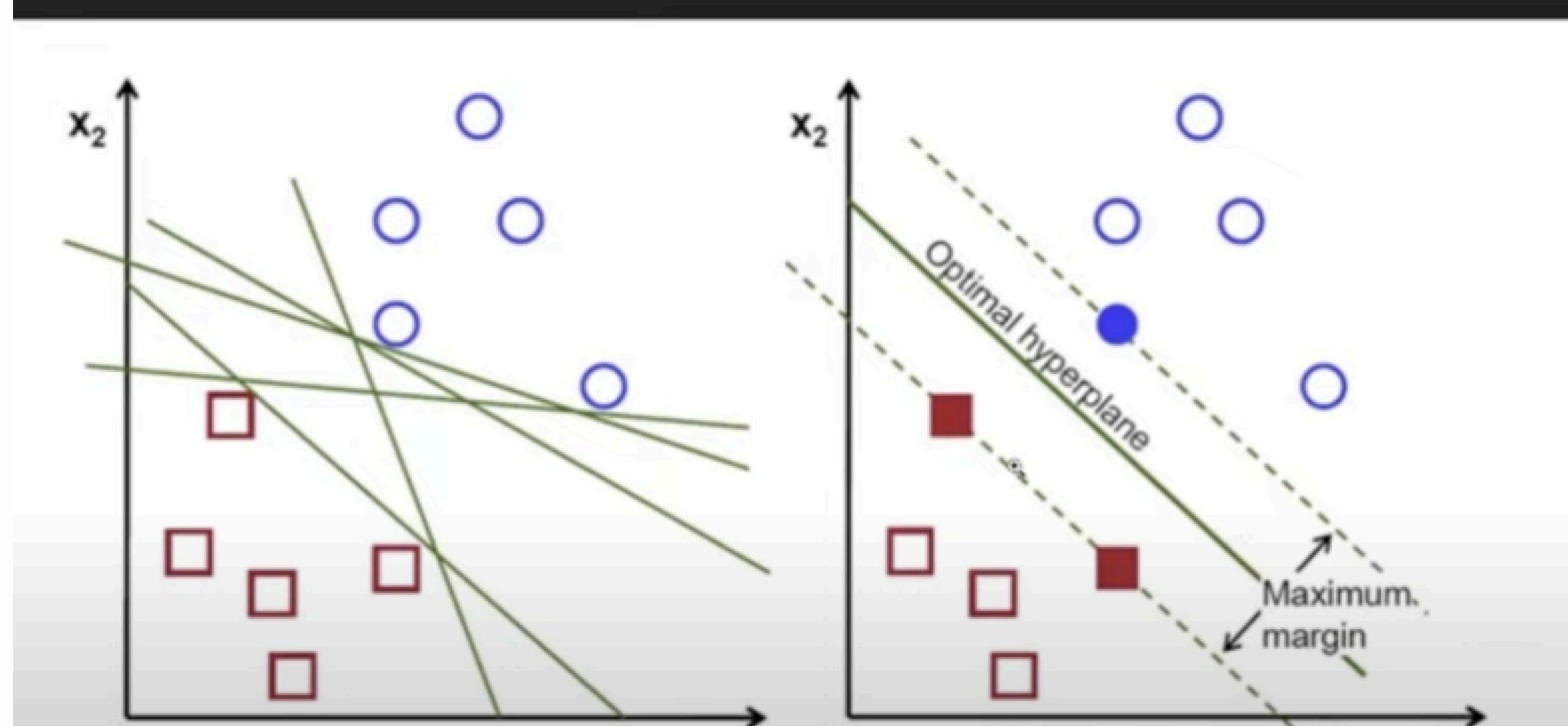
این کرنل مشابه کرنل گوس است، اما به جای فاصله‌گذاری مربعی، از فاصله لپلاس استفاده می‌کند.

## **(Chi-squared Kernel) ۶**

این کرنل به‌ویژه برای داده‌های متنی و دسته‌بندی تصویر استفاده می‌شود و بر مبنای تجزیه و تحلیل جفت‌سازی داده‌ها طراحی شده است.

## **(Cluster Kernel) ۷**

این نوع کرنل به‌طور خاص برای مسائل کلاسترینگ طراحی شده است و می‌تواند برای شناسایی الگوهای پیچیده در داده‌های چندکلاسی مفید باشد.



دقت (Accuracy): دقت، نسبت پیش‌بینی‌های صحیح به کل پیش‌بینی‌ها است

$$\frac{TP + TN}{TP + TN + FP + FN} = \text{Accuracy}$$

Recall نسبت نمونه‌های مثبت درست پیش‌بینی شده به کل نمونه‌های مثبت واقعی است. به عبارت دیگر، نشان می‌دهد که چه درصدی از مثبت‌های واقعی به درستی شناسایی شده‌اند.

$$\frac{TP}{TP + FN} = \text{Recall}$$

Precision: نسبت نمونه‌های مثبت درست پیش‌بینی شده به کل نمونه‌هایی است که به عنوان مثبت پیش‌بینی شده‌اند. به عبارت دیگر، نشان می‌دهد که از همه نمونه‌های پیش‌بینی شده به عنوان مثبت، چه درصدی واقعاً مثبت هستند.

$$\frac{TP}{TP + FP} = \text{Precision}$$



## متريکس

**TP (True Positive):** تعداد نمونه‌های مثبت درست پیش‌بینی شده.

**TN (True Negative):** تعداد نمونه‌های منفی درست پیش‌بینی شده.

**FP (False Positive):** تعداد نمونه‌های منفی که به اشتباه مثبت پیش‌بینی شده‌اند.

**FN (False Negative):** تعداد نمونه‌های مثبت که به اشتباه منفی پیش‌بینی شده.