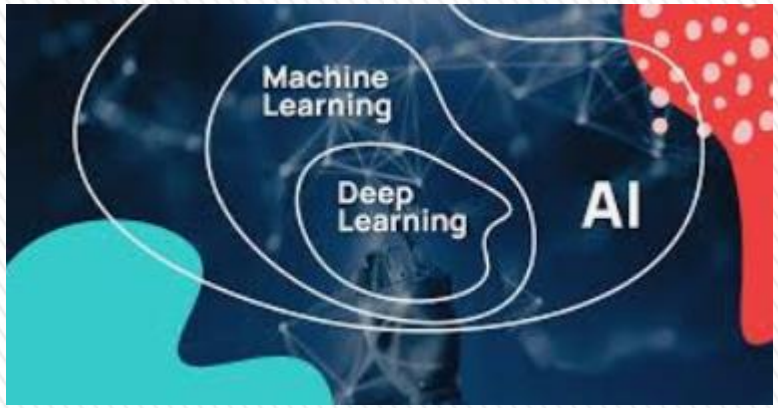


»Лекционен курс

»Въведение в машинното учене

Увод в машинното учене

Два основни вида

- Областта на машинното учене има два основни вида:
  - Контролирано учене.
  - Неконтролирано учене.
- И много подклонове, които свързват двете.

Видове модели за контролирано учене

- Моделите за контролирано учене се предлагат в две разновидности:
  - Регресионни модели - целта на регресионния модел е да предвиди числов резултат (като например цената, на която ще се продаде жилище, или възрастта на човек на снимка).
  - Класификационни модели - класификационните модели предвиждат клас или категория от краен набор от класове, дефинирани в данните за обучение.

Контролирано учене

- При контролирано учене ИИ агентът има достъп до етикети, които може да използва, за да подобри представянето си при решаване на дадена задача.
- Пример:
  - В проблема с имейл филтъра за спам имаме набор от данни от имейли с целия текст във всеки един имейл.
  - Ние също така знаем кои от тези имейли са спам и кои не са (етикетираме ги).
  - Тези етикети са много ценни, като помагат на ИИ за контролирано учене да отдели спам имейлите от останалите.

Неконтролирано учене

- При неконтролирано учене етикети не са налични.
- Следователно задачата на ИИ агента не е добре дефинирана и ефективността не може да бъде толкова ясно измерена.
- Пример:
  - Проблем с филтъра за нежелана поща – този път без етикети.
  - Сега ИИ агентът ще се опита да разбере основната структура на имейлите, разделяйки базата данни с имейли в различни групи, така че имейлите в една група да са подобни един на друг, но различни от имейлите в други групи.

Неконтролирано учене

- Този проблем с неконтролираното учене не е толкова ясно дефиниран от проблема с контролираното учене и е по-труден за разрешаване от ИИ агента.
- Но ако използва добре методите за неконтролирано учене, решението е по-добро.

Защо по-добро решение

- Неконтролираният учащ се ИИ може да намери няколко групи, които по-късно маркира като „спам“.
- Но може също да намери групи, които по-късно маркира като „важни“ или категоризира като „семейни“, „професионални“, „новини“ ”, „пазаруване” и т.н.
- С други думи, тъй като проблемът няма строго дефинирана задача, ИИ агентът може да намери интересни модели също извън това, което първоначално сме търсили.

Защо по-добро решение

- Нещо повече, тази неконтролирана система е по-добра от контролираната система в намирането на нови модели в бъдещи данни, което прави неконтролираното решение по-гъвкаво в бъдеще.
- Това е силата на неконтролираното учене.

Силни страни на контролираното учене

- Контролираното учене се отличава с оптимизиране на производителността при добре дефинирани задачи с много етикети.
- Пример:
  - Разглеждаме много голям набор от данни от изображения на обекти, където всяко изображение е етикетирано.
  - Ако наборът от данни е достатъчно голям и ние обучаваме с помощта на правилните алгоритми за МУ (напр. конволюционни невронни мрежи) и с достатъчно мощни компютри, можем да изградим много добра система за класифициране на изображения, базирана на контролирано учене.

Силни страни на контролираното учене

- Тъй като ИИ с контролирано учене се обучава върху данни, той ще може да измерва своята производителност (чрез функция на разходите), като сравнява етикета на предвиденото изображение с етикета на истинското изображение.
- ИИ явно ще се опита да минимизира тази функция на разходите, така че нейната грешка при никога невиждани изображения да е възможно най-ниска.

Силни страни на контролираното учене

- Ето защо етикетите са толкова мощни - те помагат да насочват ИИ агента, като му предоставят мярка за грешка.
- ИИ използва мярката за грешка, за да подобри своята производителност с течение на времето.
- Без такива етикети ИИ не знае колко е успешен в класифициране на изображения.

Слаби страни на контролираното учене

- Разходите за ръчно етикетиране на набор от данни за изображения обаче са високи.
- И дори най-добре подбраните набори от данни за изображения имат хиляди етикети.
- Това е проблем, тъй като системите за контролирано учене ще бъдат много добри в класифицирането на изображения на обекти, за които има етикети, но лоши в класифицирането на изображения на обекти, за които няма етикети.

Слаби страни на контролираното учене

- Колкото и мощни да са системите за контролирано учене, те също така са ограничени в обобщаването на знания извън етикетираните елементи, по които са се обучавали.
- Тъй като по-голямата част от данните в света са неетикетирани, с контролирано учене способността на ИИ да разшири своята производителност до невиждани досега случаи е доста ограничена.

Обобщение

- С други думи, контролираното учене е добро средство за решаване на тесни специфични проблеми.
- Но не е толкова добро при решаването на по-амбициозни, по-малко ясно дефинирани проблеми от типа на силния ИИ.

Силни страни на неконтролираното учене

- Разбира се, самият ИИ без контрол не може да етикетира тези групи като „столове“ или „кучета“.
- Но сега, когато подобни изображения са групирани заедно, хората имат много по-проста задача за етикетиране.
- Вместо да етикетират милиони изображения на ръка, хората могат ръчно да етикетират само отделни групи и етикетите ще се прилагат за всички членове във всяка група.

Силни страни на неконтролираното учене

- След първоначалното обучение, ако неконтролираният учещ се ИИ открие изображения, които не принадлежат към никоя от етикетираните групи, той ще създаде отделни групи за некласифицираните изображения, задействайки човек да етикетира новите групи, които все още не са етикетирани като изображения.

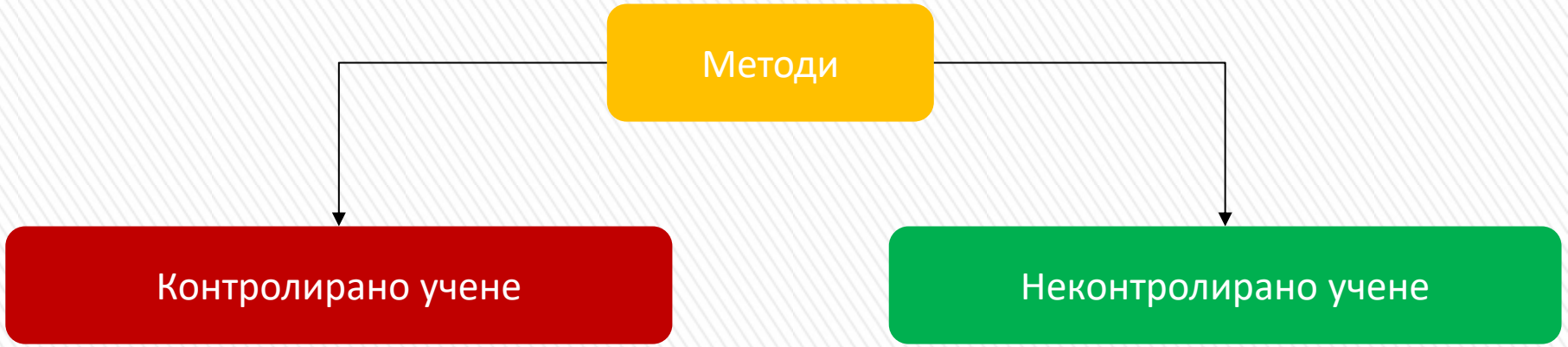
Силни страни на неконтролираното учене

- Неконтролираното учене прави по-разрешими нерешимите проблеми и е много по-гъвкаво в намирането на скрити модели както в историческите данни, които са достъпни за обучение, така и в бъдещи данни.
- Освен това сега разполагаме с ИИ подход за огромните количества немаркирани данни, които съществуват в света.

Обобщение

- Въпреки че ученето без контрол е по-малко умело от ученето с контрол при решаването на специфични, тясно дефинирани проблеми, то е по-добро при справяне с по-отворени проблеми от типа на силния ИИ и при обобщаване на това знание.
- Също толкова важно е, че ученето без контрол може да се справи с много от често срещаните проблеми, с които се сблъскват учените при изграждането на решения за машинно учене.

Общ преглед на методите



- Линейни методи
- Базирани на съседна близост
- Базирани на дървета
- Поддържащи векторни машини
- Невронни мрежи

- Редуциране на размерност
- Клъстериране
- Извличане на свойства
- Неконтролирано дълбоко учене
- Последователни данни

Линейни методи

- Най-основните алгоритми за контролирано учене моделират проста линейна връзка между входните характеристики и изходната променлива, която искаме да предвидим.
- Линейната регресия е може би най-простият алгоритъм за учене.
- Работи най-добре с данни, които са сравнително линейни, т.е. точки от данни, които попадат приблизително по една линия.

Линейно уравнение

$$Y = bX + c$$

Y: нарича се зависима променлива, изход за модела, т.е. това е, което се опитваме да предвидим.
b: нарича се коефициент. Показва връзката между зависимите и независимите променливи.
X: нарича се независимата променлива. Това са данните, които въвеждаме в уравнението, за да получим резултата.
c: нарича се константата. Това е стойност, която остава постоянна, независимо от независимата променлива.

Създаване модел

- За да създадем модел с помощта на регресионния анализ, ще е необходим набор от данни за независимите и зависимите променливи.
- Често се използва софтуер за изчисляване на стойностите за коефициента и константата.
- За целите на ИИ това може да бъде Python, Java, R, …

Пример

AI Expenditures	Value Created
$1 million	$1.2 million
$2 million	$2.5 million
$3 million	$7.1 million
$4 million	$7 million
$5 million	$8.9 million
$6 million	$8.9 million
$7 million	$7.8 million
$8 million	$10 million
$9 million	$10.10 million
$10 million	$12.9 million

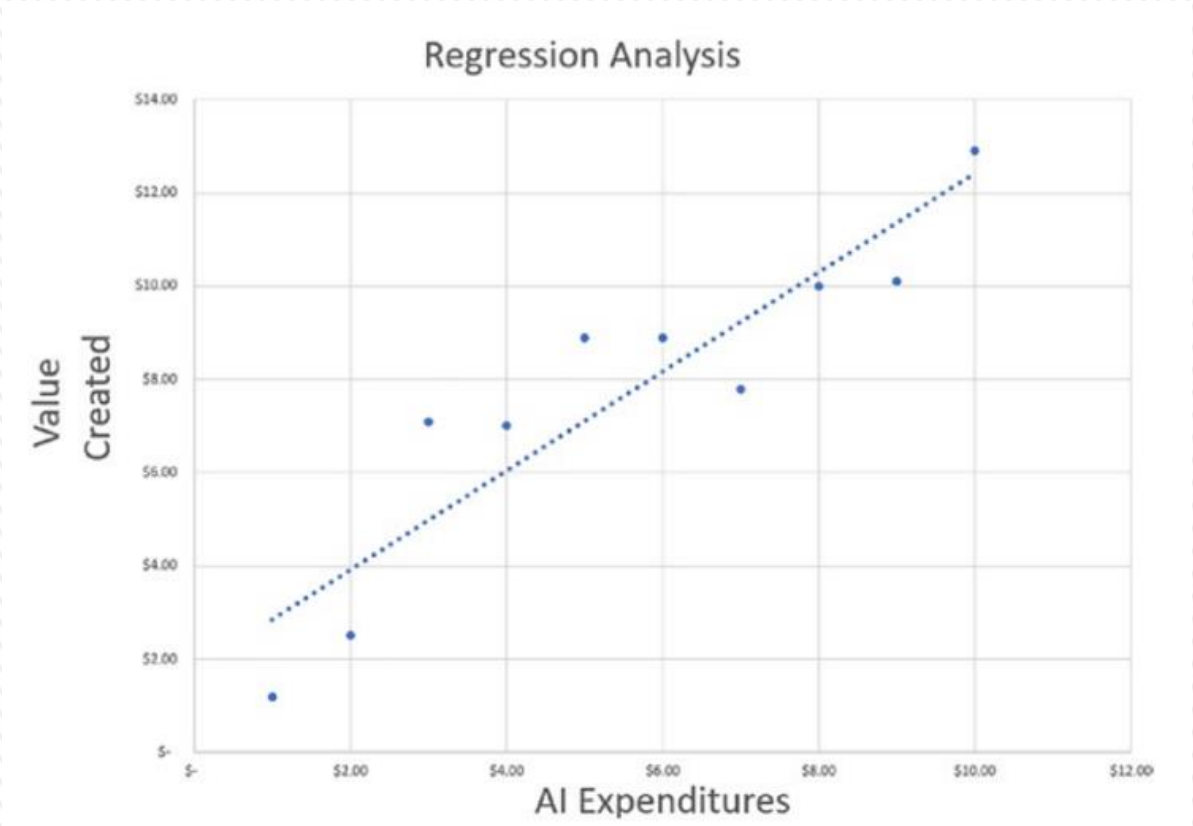
Да предположим, че съставяме модел за разходи за ИИ и получената от това стойност на една компания.

С тези данни изчисляваме следното регресионно уравнение:
Y = 1.063X + 1.7933

Пример: ако похарчи 10 милиона долара, тогава очакваният приход на компанията ще бъде 12,4 милиона долара

- ✓ Т.е. 1.063 X 10 + 1.7933.

Диаграма



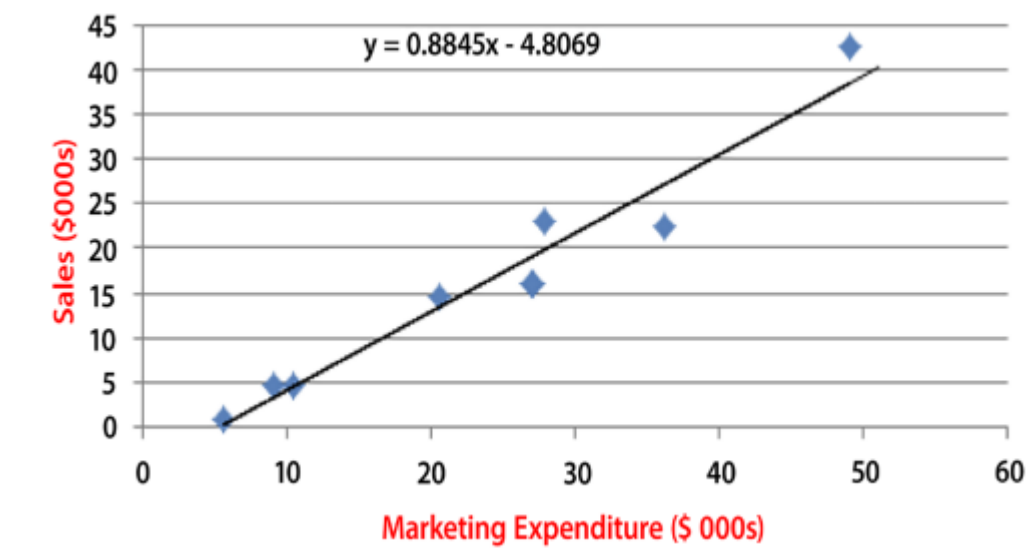
Можем да дадем числата като диаграма.

Това е наклонена нагоре крива - когато коефициентът е положителен.

Вярно, това е много опростено, но нито един модел не може да обясни всичко.

- ✓ Трябва да има обобщение и фокусиране върху онези фактори, които имат най-голямо въздействие.

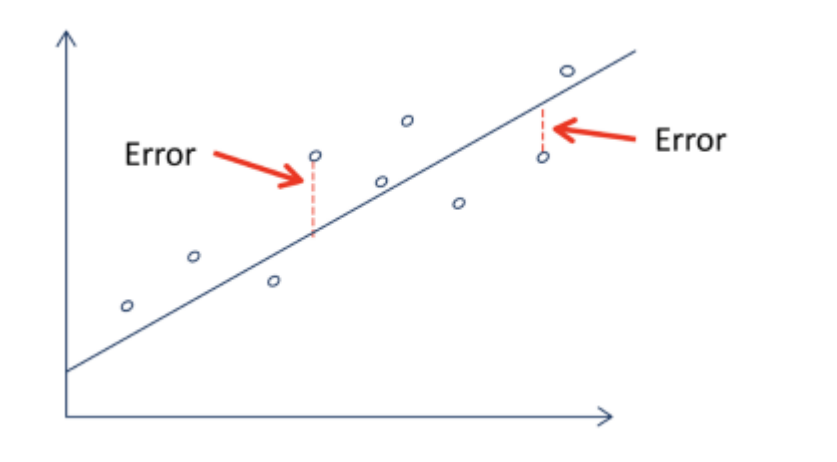
Друг пример



Да се опитаме да формулираме връзката между маркетинговите разходи и продажбите за конкретен продукт.
Установено е, че те са пряко свързани един с друг.
Маркетинговите разходи и продажбите са дадени на двуразмерна графика и са показани като сини ромбове.
Връзката може да се определи най-добре чрез начертаване на права линия, както е показано на графиката.

След като линията бъде начертана, можем да видим математическата връзка между маркетинговите разходи и продажбите.

Оценяване на регресорите



Линията е приближение на връзката между зависимите и независимите променливи.

Дори най-добрата линия ще има известно отклонение от действителните стойности, както е показано тук.

Средна квадратична грешка

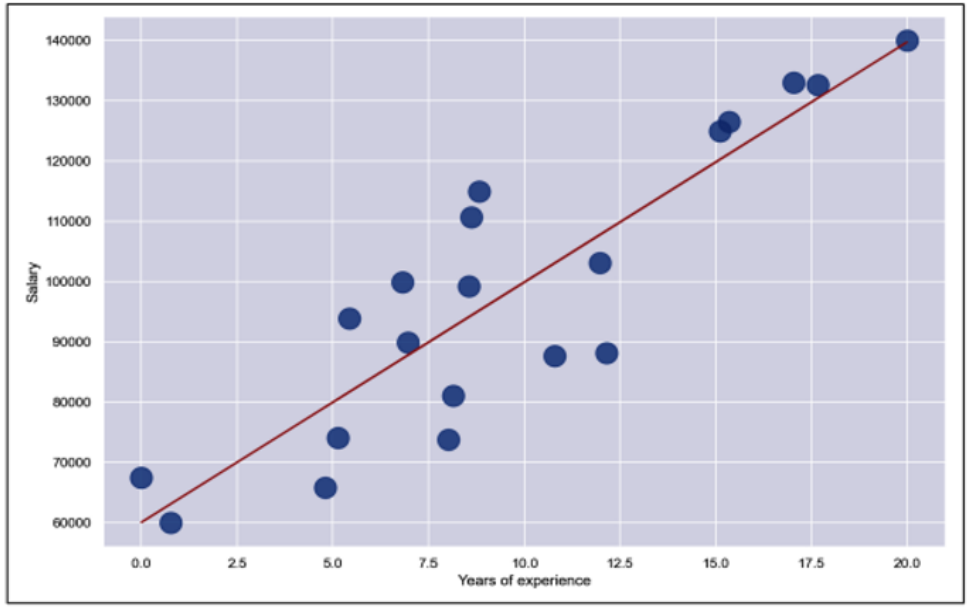
$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}^{(i)} - y^{i})^2}$$

Типичен начин за количествено определяне на производителността на моделите на линейна регресия е чрез използване на средна квадратична грешка (RMSE).
Това изчислява математически стандартното отклонение на грешките, направени от обучения модел.

Това води до следната функция на разходите, която минимизира загубата на всички примери в набора за обучение:

Пример

$$Y = bX + c$$



Даден набор от данни: заплата спрямо години стаж.

Фигурата показва напасване на регресионна линия към точките от данни.

Прогнозирането на дохода за програмист с 10-годишен стаж е толкова просто, колкото намирането на точката на правата, където x = 10

Уравнението на правата е y = 3,984x + 60 040. Заместваики x с 10 в това уравнение, прогнозираният доход е 99 880.
y=3.984·10+60040=99880

Цел

- Целта при учене с линеен регресионен модел е да се намерят стойности за b и c, които произвеждат най-точните прогнози.
- Това обикновено се прави с помощта на итеративен процес, който започва с накакви приети стойности за b и c и се повтаря, докато достигне до подходящи стойности.

Регресия на най-малките квадрати

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

- Най-разпространеният метод за напасване на линия към набор от точки е обикновената регресия на най-малките квадрати.
- Повдига на квадрат разстоянието в посока **y** между всяка точка и линията на регресия, сумира квадратите и разделя на броя на точките, за да изчисли средната квадратна грешка или MSE.
  - Поставяне на квадрат всеки разстояние не позволява на отрицателните разстояния да компенсират положителните разстояния.

Регресия на най-малките квадрати

- След това коригира b и c, за да намали MSE следващия път и повтаря, докато MSE стане достатъчно ниско.
- Няма да навлизам в подробности за това как се определя в коя посока да се коригират b и c (не е трудно, но включва малко смятане - по-специално, използване на частични производни на функцията MSE, за да се определи дали да се увеличи или намали b и c в следващата итерация), но OLS често може да напасне линия към набор от точки с дузина или по-малко итерации.

Класове за линейна регресия

```
model = LinearRegression()
model.fit(x, y)
```

Scikit-Learn има редица класове, които подпомагат изграждане на линейни регресионни модели, включително:
Клас LinearRegression, който въплъщава OLS
Клас Polynomial Features, който пасва на полиномна крива, а не на права линия към данните за обучение.

Обучението на линеен регресионен модел може да бъде толкова просто, колкото това, показано на слайта.

Линейна регресия

- Интерпретирането на линейната връзка също е много лесно.
- Интерпретируемостта е много важно съображение за приложното машинно учене, тъй като решенията трябва да бъдат разбрани и въведени както от технически, така и от нетехнически хора в индустрията.
- Без интерпретация решенията се превръщат в неразгадаеми черни кутии.

Силни страни

- Линейната регресия е проста, интерпретируема и трудна за претоварване, защото не може да моделира прекалено сложни връзки.
- Това е отличен избор, когато основната връзка между входните и изходните променливи е линейна.

Слабости

» Линейната регресия няма да отговаря на данните, когато връзката между входните и изходните променливи е нелинейна.

Пример за приложение

» Тъй като истинската основна връзка между човешкото тегло и човешкия ръст е линейна, линейната регресия е чудесна за прогнозиране на теглото, използвайки височината като входна променлива или обратно, за прогнозиране на височината, използвайки теглото като входна променлива.

Логистична регресия

$$P(y = 1|x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}}$$

- Логистичната регресия е линеен метод, но прогнозите се трансформират с помощта на логистична функция.
- Резултатите от тази трансформация са класови вероятности
  - С други думи, вероятностите, че екземплярът принадлежи към различните класове, където сумата от вероятностите за всеки екземпляр е 1.
- Всеки екземпляр се присвоява на класа, към който има най-голяма вероятност да принадлежи.

Силни страни

- Подобно на линейната регресия, логистичната регресия е проста и интерпретируема.
- Когато класовете, които се опитваме да предвидим, не се припокриват и са линейно разделими, логистичната регресия е отличен избор.

Слабости

» Когато класовете не са линейно разделими, логистичната регресия ще се провали.

Приложение

- Когато класовете в повечето случаи не се припокриват
- Например:
  - Ръстът на малките деца спрямо ръста на възрастните.
  - Логистичната регресия ще работи добре.

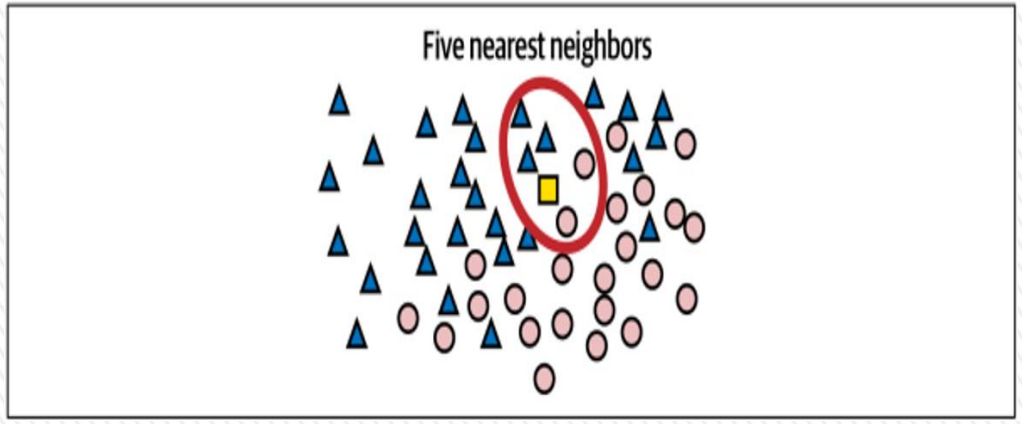
Базираните на съседство методи

- Базираните на съседство методи са „мързеливи“ обучаеми, тъй като те се научават как да етикетират нови точки въз основа на близостта на новите точки до съществуващите етикетирани точки.
- За разлика от линейните методи, моделите, базирани на съседство, не се обучават за да прогнозират етикети за нови точки.
- По-скоро тези модели предвиждат етикети за нови точки въз основа единствено на разстоянието на новите точки до съществуващите етикетирани точки.

k-най-близки съседи (KNN)

- KNN е най-често срещаният метод, базиран на съседство.
- За да обозначи всяка нова точка, KNN разглежда k на брой най-близките обозначени точки, които определят как да се обозначи новата точка.
- По подразбиране KNN използва евклидово разстояние, за да измери най-близкото.

Пример



Ако n = 5 и петте най-близки съседи включват три триъгълника и две елипси, тогава отговорът е триъгълник, както е показано на фигурата.

k-най-близки съседи

- Изборът на k е много важен.
- Ако k е с много ниска стойност, KNN става много гъвкав, очертавайки силно нюансирани граници и потенциално пренастройвайки данните.
- Ако k е с много висока стойност, KNN става негъвкав, очертавайки твърда граница и потенциално подхождайки на данните.

Силни страни

- За разлика от линейните методи, KNN е много гъвкав и способен да изучава по-сложни, нелинейни връзки.
- И все пак KNN остава прост и интерпретируем.

Слабости

- KNN се справя зле, когато броят на наблюденията и характеристиките расте.
  - KNN става изчислително неефективен в това гъсто населено пространство с големи размери, тъй като трябва да изчисли разстояния от новата точка до много близки точки с етикети, за да предвиди етикети.
- Не може да разчита на ефективен модел с намален брой параметри, за да направи необходимата прогноза.
- Освен това KNN е много чувствителен към избора на k.

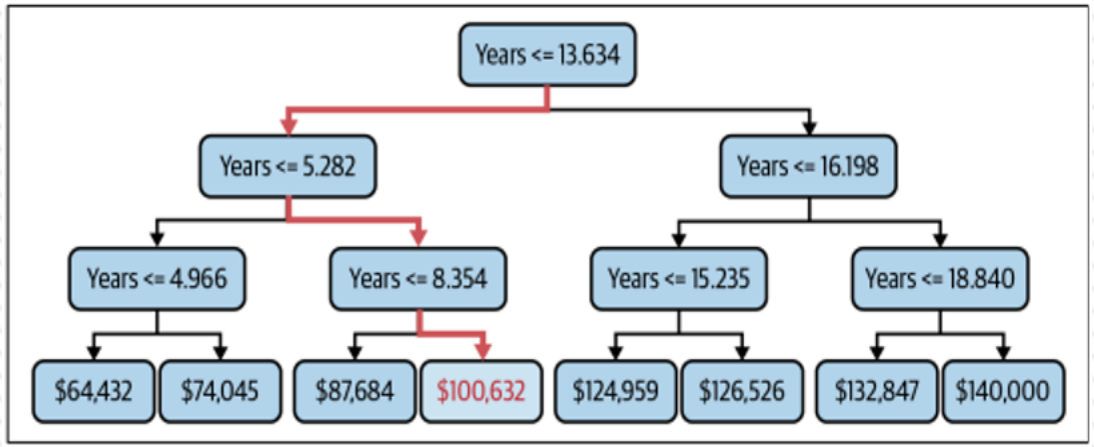
Приложения

- KNN се използва в системи за препоръки, като например на:
  - филми (Netflix)
  - музика (Spotify)
  - приятели (Facebook)
  - снимки (Instagram)
  - търсене (Google)
  - пазаруване (Amazon).

Базирани на дървета методи

- В МУ дървото на решения е дървовидна структура, която прогнозира резултат, като отговаря на поредица от въпроси.
- Повечето дървета на решения са двоични дървета, в който случай въпросите изискват прости отговори с да или не.

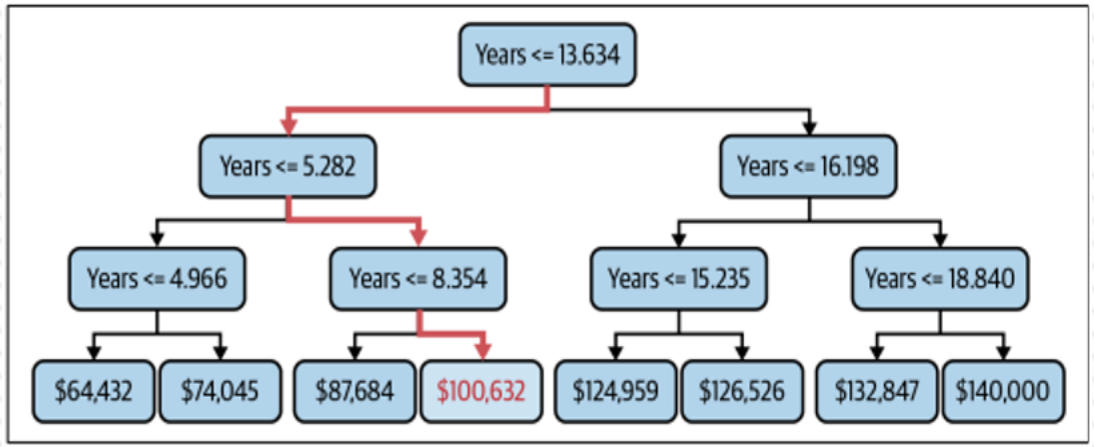
Пример



Фигурата показва дърво на решенията, изградено от Scikit от набора от данни доход срещу стаж (представен преди).

Дървото е просто, защото наборът от данни съдържа само една колона с характеристики (години стаж) и дълбочината на дървото е ограничена до 3 (но техниката се простира до дървета с неограничен размер и сложност).

Пример



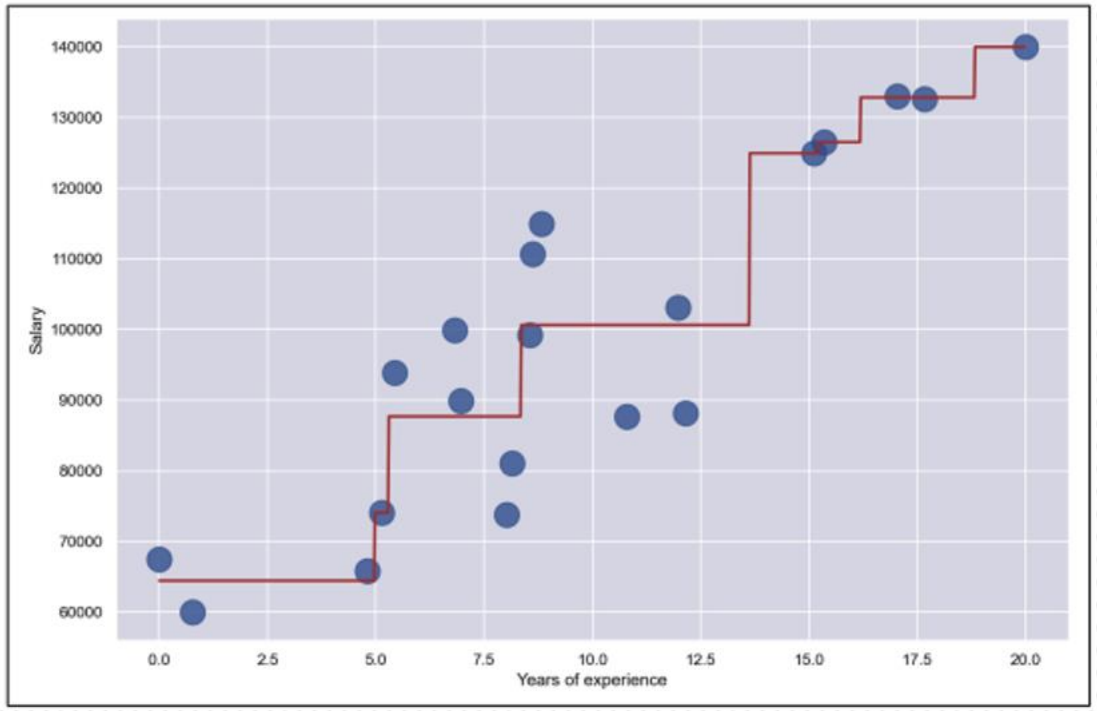
В този пример предвиждането на заплата за програмист с 10 години стаж изисква само три да/не решения, както е показано с червените стрелки.

Отговорът е около $100K, което е доста близо до това, което k-близките съседи и линейната регресия прогнозират, когато се прилагат към същия набор от данни.

Приложение

- Дърветата на решенията могат да се използват за регресия и класификация.
  - При регресор: листата (възли без наследници) представляват регресионни стойности.
  - При класификатор: листата представляват класове.

Пример



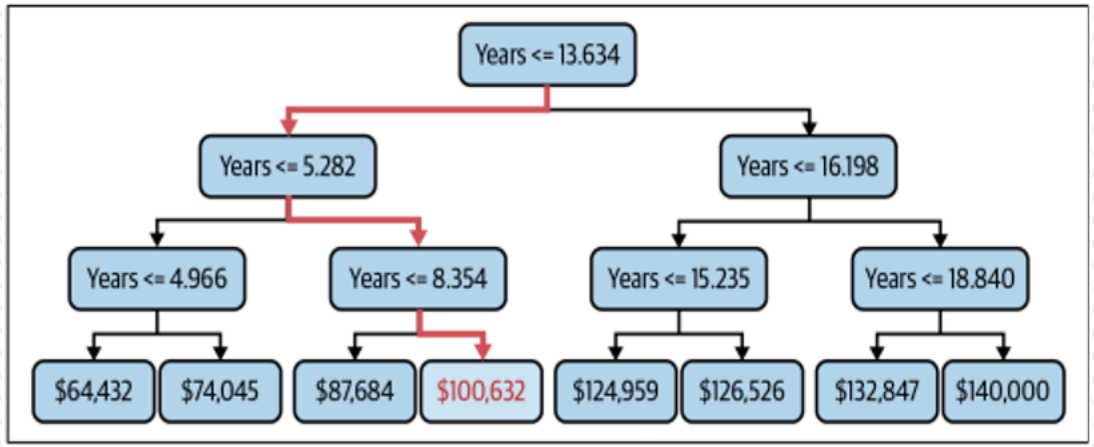
В предишния пример получаваме същия отговор, ако искаме дървото да предвиди заплата за някой с 10 години стаж и някой с 13 години.

При увеличение на годините стаж до 14, прогнозираната заплата скача до $125K (фигурата).

Ако позволим на дървото да расте по-дълбоко, отговорите стават по-прецизни.

- ✓ Но оставянето му да расте твърде дълбоко може да доведе до проблеми.

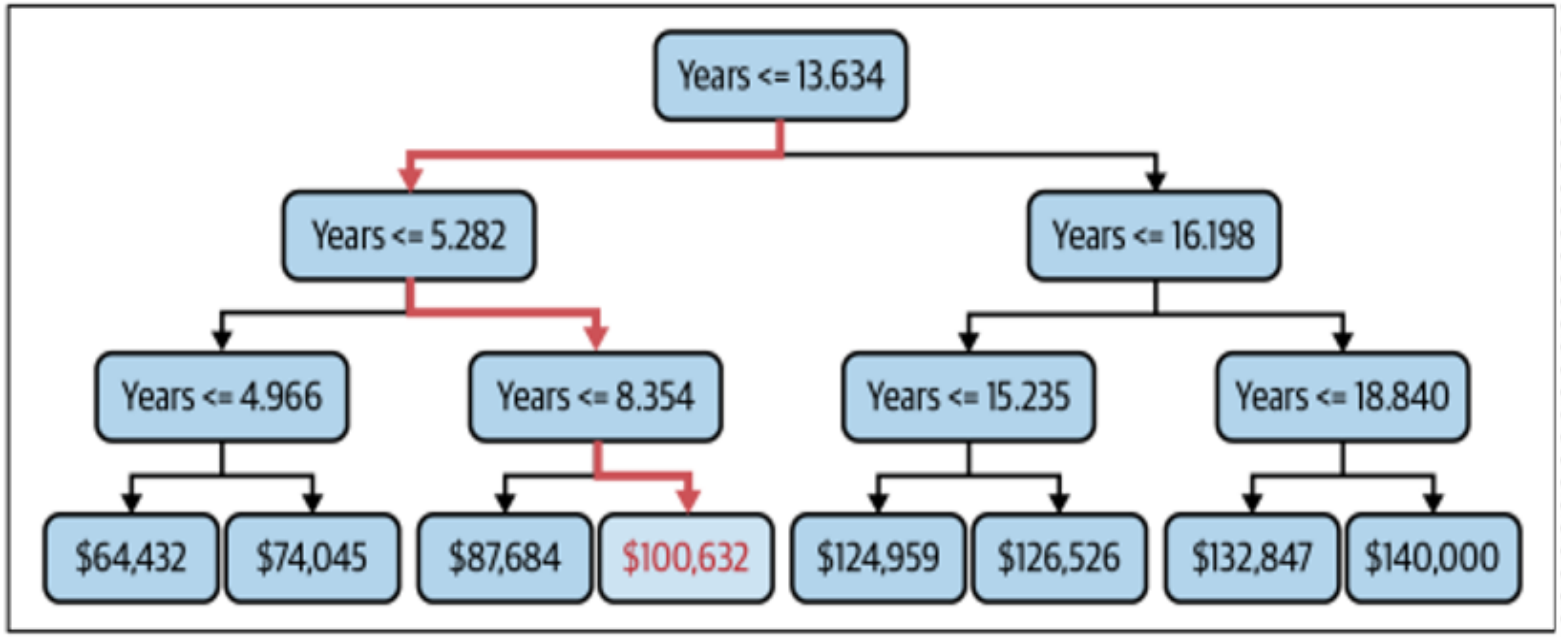
Пример



След като моделът на дървото на решенията е обучен – т.е. след като дървото е изградено – прогнозите се правят бързо.

Но как решаваме какви решения да вземем във всеки възел?

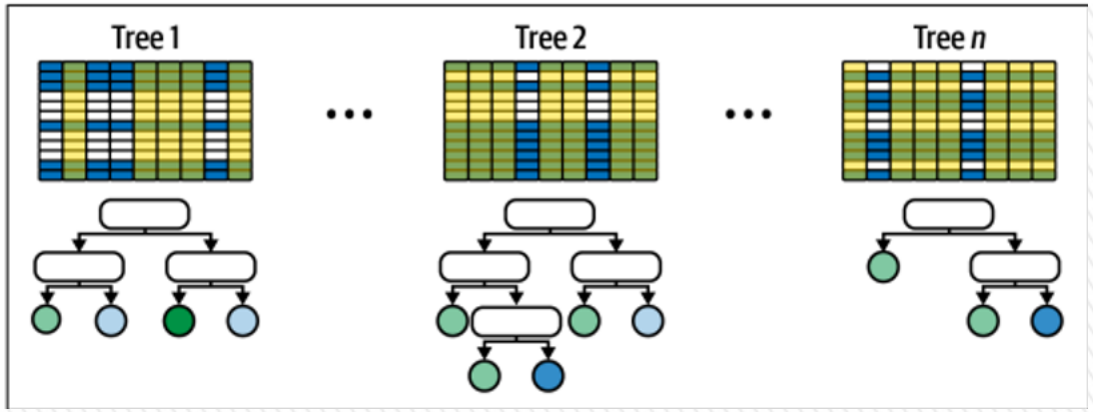
Пример



Например, защо броят години, представен от коренния възел на фигурата е равен на 13,634? Защо не 10 000 или 8 742 или някое друго число?

По този въпрос, ако наборът от данни има множество колони с характеристики, как решавате коя колона да прекъснете във всеки възел за решение?

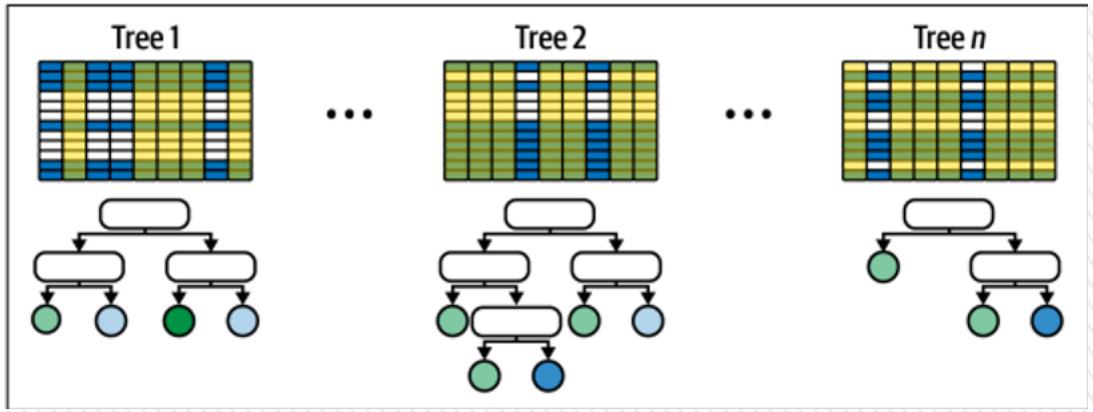
Случайни гори



Случайна гора е колекция от дървета на решения (често стотици от тях), всяко обучено по различен начин с един и същ набор данни.

Обикновено всяко дърво се обучава на случайно избрани редове в набора от данни, а разклоняването се основава на колони, които се избират на случаен принцип при всяко разделяне.

Случайни гори



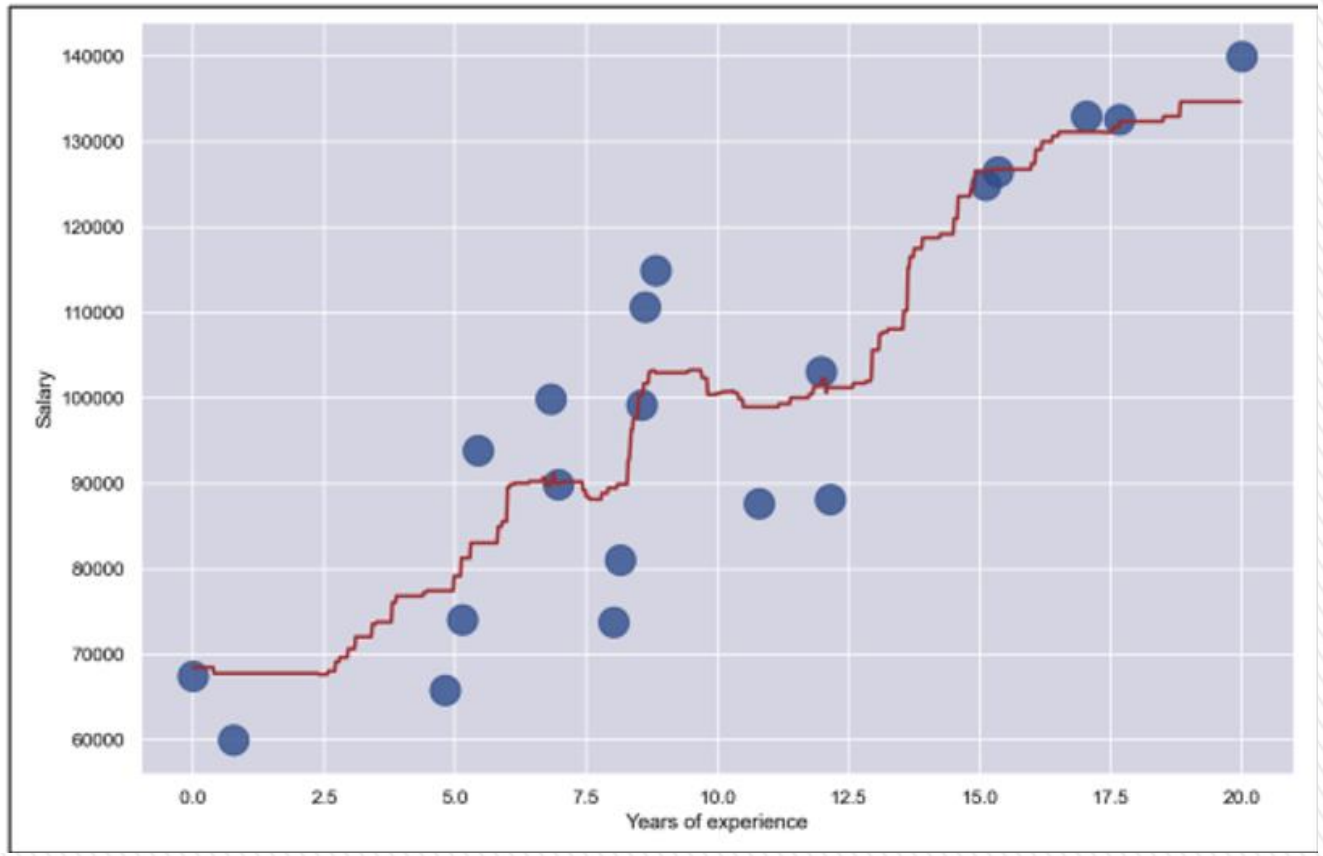
Моделът не може да се вмести твърде плътно в данните за обучение, тъй като всяко дърво се обучава на различно подмножество от данните.

Дърветата се изграждат независимо и когато моделът прави прогноза, той преминава през всички дървета на решенията и осреднява резултата.

Тъй като дърветата са конструирани независимо, обучението може да бъде паралелно на хардуер, който го поддържа.

- Това е проста концепция, която работи добре на практика.
- Случайните гори могат да се използват както за регресия, така и за класификация
- Scikit предоставя класове като Random ForestRegressor и RandomForestClassifier.
- Те разполагат с редица регулируеми параметри, включително:
  - n_estimators, който определя броя на дърветата в произволната гора (по подразбиране = 100);
  - max_depth, което ограничава дълбочината на всяко дърво;
  - max_samples, който определя частта от редовете в данните за обучение, използвани за изграждане на отделни дървета.

Пример



Фигурата показва как RandomForestRegressor се вписва в набора от данни доход срещу стаж с max_depth=3 и max_samples=0,5, което означава нито едно дърво не вижда повече от 50% от редовете в набора от данни.

И въпреки че фигурата показва как случайна гора пасва на линеен набор от данни, произволните гори са напълно способни да моделират и нелинейни набори от данни.

Поддържащи векторни машини

- Вместо да изграждаме дървета за разделяне на данни, можем да използваме алгоритми за създаване на хиперравнини в пространството, които разделят данните, направлявани от етикетите, които имаме.
- Подходът е известен като опорни векторни машини (SVM).

Невронни мрежи

- Можем да научим представяне на данните с помощта на невронни мрежи, които се състоят от:
  - Входен слой.
  - Няколко скрити слоя.
  - Изходен слой.
- Една невронна мрежа е в състояние да учи сложни концепции, изграждайки ги от по-прости.

Редуциране на размерността

- Проектират оригиналните високоразмерни входни данни в нискоразмерно пространство, като филтрират не толкова значимите параметри и запазват колкото се може повече от значимите.
- Намаляването на размерността позволява на неконтролиран обучаващ се ИИ по-ефективно да идентифицира модели и по-ефективно да решава мащабни, изчислително скъпи проблеми (често включващи изображения, видео, реч и текст).

Видове

- Има два основни типа:
  - Линейна проекция.
  - Нелинейна редукция на размерността.

Клъстериране

- След като сме намалили набора от оригинални характеристики до по-малък, по-управляем набор, можем да намерим интересни модели, като групираме подобни екземпляри на данни заедно.
- Това е известно като клъстериране и може да бъде постигнато с различни алгоритми за неконтролирано учене и да се използва за реални приложения.

k-средно

- За да клъстерираме добре, трябва да идентифицираме отделни групи, така че екземплярите в една група да са подобни един на друг, но различни от екземплярите в други групи.
- Един такъв алгоритъм е k-средно клъстериране.
- С този алгоритъм ние определяме броя на желаните клъстери k и алгоритъмът ще присвои всеки екземпляр на точно един от тези k клъстери.

Йерархично клъстеризиране

- Алтернативен подход за клъстериране – такъв, който не изисква предварително ангажиране с определен брой клъстери.
- Една версия на йерархично клъстериране, наречена агломеративно клъстериране, използва дървовиден метод за клъстериране и изгражда това, което се нарича дендрограма.
- Дендрограмата може да бъде изобразена графично като обърнато дърво, където листата са отдолу, а стволът е отгоре.

Извличане на свойства

- С неконтролирано учене можем да научим нови представяния на оригиналните характеристики на данните - известно като извличане на характеристики.
- Извличането на характеристики може да се използва за намаляване на броя на оригиналните параметри и стова намаляване на размерността.
- Може също да генерира нови представяния на параметри
  - Напр. енкодери,автокодери, ...

Неконтролирано дълбоко учене

- Неконтролираното учене изпълнява много важни функции в областта на дълбокото учене.
- Това поле е известно като неконтролирано дълбоко учене.
- До съвсем скоро обучението на дълбоки невронни мрежи беше изчислително неразрешим проблем.
- В тези невронни мрежи скритите слоеве научават вътрешни представяния, за да помогнат за решаването на проблема.
- Представянията се подобряват с течение на времето въз основа на това как невронната мрежа използва градиента на функцията за грешка във всяка итерация на обучение, за да актуализира теглата на различните възли.
- Тези актуализации са скъпи от изчислителна гледна точка.

Неконтролирано предварително учене

- За да се справят с тези трудности при обучението на много дълбоки, многопластови невронни мрежи, изследователите на машинното учене обучават невронни мрежи в последователни етапи, където всеки етап включва плитка невронна мрежа.
- След това изходът на една плитка мрежа се използва като вход на следващата невронна мрежа.
- Обикновено първата плитка невронна мрежа в този тръбопровод включва неконтролирана невронна мрежа, но по-късните мрежи са контролирани.

Неконтролирано предварително учене

- През 2006 г. Джефри Хинтън демонстрира успешното прилагане на неконтролирано предварително учене за инициализиране на обучението на по-дълбоки тръбопроводи невронни мрежи, поставяйки началото на настоящата революция в дълбокото учене.

Сравнение: Контролирано vs. Неконтролирано
Сравнение: Контролирано vs. Неконтролирано

Критерий	Контролирано	Неконтролирано
Данни	Етикетирани	Неетикетирани
Цел	Прогноза	Откриване на структура
Примери	Класификация, регресия	Клъстериране, PCA
Силни страни	По-точни прогнози, ясна цел	Автоматично откриване на скрити модели
Слаби страни	Зависи от етикети, възможно наднапасване	Трудно валидиране, неясна интерпретация

Пример за създаване на модел (регресия) Python

```python
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split

# Данни
X = [[1], [2], [3], [4], [5]]
y = [1.2, 1.9, 3.0, 4.1, 5.1]

# Разделяне
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

# Модел
model = LinearRegression()
model.fit(X_train, y_train)

# Прогноза и грешка
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
print("Средна квадратична грешка:", mse)
```

1

2

3

4

5

Регистрация

https://tinyurl.com/yo45mq78


Благодаря за вниманието!