Validació

lloc:	Institut d'Ensenyaments a Distància de les Illes Balears	Imprès per:	Carlos Sanchez Recio
Curs:	Sistemes d'aprenentatge automàtic	Data:	diumenge, 16 de març 2025, 22:02
Llibre:	Validació		

Taula de continguts

1. Presentació

Una vegada que hem presentat les aplicacions d'aprenentatge supervisat i no supervisat, i un primer lliurament de xarxes neuronals, estam en condicions d'aprofundir en el problema de la selecció i validació del nostre model i l'ajust dels seus hiperparàmetres.

Per a l'elaboració d'aquest capítol utilitzarem tres fonts principals següents, en ordre d'aparició.

- L'apartat 19.4 del llibre de Russell i Norvig, "Model Selection and Optimization". Això ens donarà la visió breu general del primer apartat.
- El capítol 6è del llibre de Raschka i Mirjalili, "Learning Best Practices for Model Evaluation and Hyperparameter Tuning". A partir del treball sobre el conjunt de dades Breast Cancer Wisconsin veurem l'aplicació pràctica dels conceptes exposats abans. Hi haurà un quadern Colab que recollirà una versió de tot el codi presentat al llibre.
- Finalment, l'apartat **Model Selection** de la documentació en línia de la llibreria scikit-learn ens permetrà aprofundir en els detalls de moltes funcions útils en aquest lliurament, així com cobrir també les aplicacions de **regressió** i **clusterització**, que els dos llibres anteriors deixen fora.

Som-hi!

2. Selecció del model i optimització

El nostre objectiu en aprenentatge automàtic és seleccionar una hipòtesi que s'ajustarà de forma òptima als exemples futurs.

Per precisar això, hem de definir què entenem per exemple futur i ajust òptim.

Primerament, farem la suposició que els exemples futurs seran com els del passat: aquesta és la suposició d'**estacionarietat**; sense això, no hi ha res a fer.

Suposam que cada exemple té la mateixa distribució de probabilitat i que és independent dels exemples anteriors.

Els exemples que compleixen aquestes dues condicions són **independents i idènticament distribuïts**, **iid**.

A continuació, hem de definir **ajust òptim**. De moment, direm que l'ajust òptim és la hipòtesi que minimitza la **taxa d'error**: la proporció de vegades que $h(x)y$/per a un exemple (x,y). Més endavant elaborarem aquesta definició per donar crèdit parcial a les respostes que són parcialment correctes. Podem estimar la taxa d'error d'una hipòtesi donant-li un test: mesurant la seva precisió en un **conjunt de test** d'exemples. Seria fer trampa, per a una hipòtesi, mirar les respostes del test abans de realitzar-lo. La forma més senzilla de garantir que això no passi és repartir els exemples de què disposem en dos conjunts: un **conjunt d'entrenament** per a crear la hipòtesi, i un **conjunt de test** diferent per avaluar-la.

Si només hem de crear una hipòtesi, aleshores aquest plantejament és suficient. Però sovint acabam creant hipòtesis múltiples: podem voler comparar dos models d'aprenentatge automàtic totalment diferents, o bé podem voler ajustar els diferents botons (*knobs*) d'un model. Per exemple, podem voler assajar diferents graus d'un polinomi de regressió. Anomenam aquests controls **hiperparàmetres**; són paràmetres de la classe de models, no del model individual.

Per poder fer aquesta operació de triar els hiperparàmetres sense utilitzar les dades de test, realment necessitam tres conjunts separats.

1. Un **conjunt d'entrenament** per entrenar els models candidats.
2. Un **conjunt de validació**, també conegut com a **conjunt de desenvolupament**, o **dev set**, per avaluar els models candidats i triar-ne el millor.
3. Un **conjunt de test**, per realitzar una avaluació final sense biaix del millor model.

I si no tenim prou dades per fer els tres conjunts de dades? Podem esprémer les dades usant una tècnica anomenada **validació creuada k-fold** (en anglès, *k-fold cross-validation*). La idea és que cada exemple fa les dues funcions (com a dada d'entrenament i dada d'avaluació) però no alhora. Primerament repartim les dades en k subconjunts iguals. En segon lloc, realitzam k voltes d'aprenentatge; en cada volta una fracció $\frac{1}{k}$ de les dades se separa com a conjunt de validació i la resta d'exemples s'usen com a conjunt d'entrenament. La puntuació mitjana de test de les k voltes hauria de ser una estimació millor que no una sola puntuació. Valors habituals de k són 5 o 10, suficient per donar una estimació estadística probablement precisa, a un cost d'unes 5 o 10 vegades el temps de computació. El cas extrem de validació creuada k-fold és k=n, també conegut com leave-one-out cross-validation o LOOCV. Fins i tot amb validació creuada, encara necessitam un conjunt de test separat.

A la figura següent veim una funció lineal infraajustada al conjunt de dades, i un polinomi de grau elevat sobreajustat a les dades. Podem pensar la tasca de trobar una bona hipòtesi com dues subtasques: la **selecció del model** tria un bon espai d'hipòtesis, i l'**optimització** (també anomenada **entrenament**) troba la millor hipòtesi dins aquest espai.

Tot i que el nom "seleccio del model" és d'ús habitual, hauria estat millor dir-ne "selecció de la classe del model". La paraula "model" s'usa per referir-se a tres nivells diferents d'especificitat: un espai d'hipòtesis ampli (per exemple, els polinomis), un espai d'hipòtesis amb els hiperparàmetres fixats (per exemple, polinomis de grau dos), i una hipòtesi específica amb tots els paràmetres determinats (per exemple, $5x^2 + 3x + 2$).

Una part de la selecció del model és qualitativa i subjectiva: podem triar polinomis en comptes d'arbres de decisió en base a qualque cosa que sabem sobre el problema. Una altra part és quantitativa i empírica: dins la classe dels polinomis, triam que el grau sigui 2, perquè aquest valor obté un resultat millor al conjunt de dades de validació.

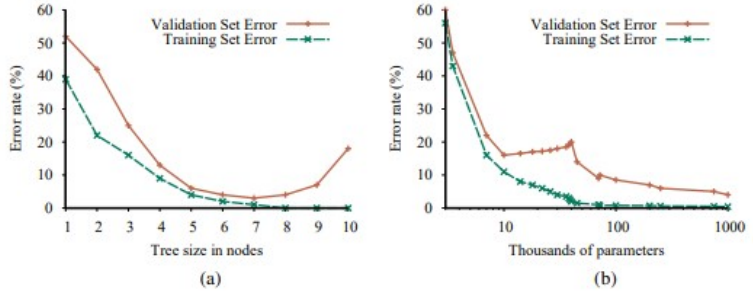
2.1. Selecció del model

La figura següent descriu un algorisme de selecció del model.

```
function MODEL-SELECTION(Learner, examples, k) returns a (hypothesis, error rate) pair

  err ← an array, indexed by size, storing validation-set error rates
  training_set, test_set ← a partition of examples into two sets
  for size = 1 to ∞ do
    err[size] ← CROSS-VALIDATION(Learner, size, training_set, k)
    if err is starting to increase significantly then
      best_size ← the value of size with minimum err[size]
      h ← Learner(best_size, training_set)
      return h, ERROR-RATE(h, test_set)

function CROSS-VALIDATION(Learner, size, examples, k) returns error rate

  N ← the number of examples
  errs ← 0
  for i = 1 to k do
    validation_set ← examples[(i − 1) × N/k:i × N/k]
    training_set ← examples − validation_set
    h ← Learner(size, training_set)
    errs ← errs + ERROR-RATE(h, validation_set)
  return errs / k        // average error rate on validation sets, across k-fold cross-validation
```

Pren com a argument un algorisme d'aprenentatge, *Learner*. *Learner* pren un hiperparàmetre, que s'anomena *size* a la figura. En arbres de decisió podria ser el nombre de nodes de l'arbre; en polinomis, *size* seria el grau. La funció MODEL-SELECTION comença amb el valor més petit de *size*, i això dona com a resultat un model simple, que probablement infraajusta a les dades, i va iterant cap a valors més grans de *size*, de forma que té en compte models cada vegada més complexos. Al final MODEL-SELECTION tria el model que té la taxa d'error mitjana més baixa sobre les dades de validació reservades per aquesta mesura.

A la figura següent veim dos patrons típics que es donen en selecció de models.



En els dos casos l'error sobre el conjunt d'entrenament decreix monotònicament (amb lleugeres fluctuacions aleatòries) a mesura que augmenta la complexitat del model. La complexitat correspon al nombre de nodes de l'arbre de decisió en el primer cas, i el nombre de paràmetres de la xarxa neuronal en el segon. En moltes classes de model, l'error d'entrenament arriba a zero així com la complexitat augmenta.

Els dos casos són marcadament diferents pel que fa a l'error del conjunt de validació. En el primer cas veim una corba amb forma d'U: l'error decreix momentàniament a mesura que la complexitat creix, però quan s'arriba a un punt en què el model comença a sobreajustar, l'error de validació augmenta. La funció MODEL-SELECTION triarà el valor a la vall de la corba d'error de validació: en aquest cas un arbre amb set nodes. Aquest és el punt d'equilibri entre infraajust i sobreajust. En el cas de les xarxes neuronals (b), veim una secció inicial amb forma d'U, però després l'error de validació comença a davallar una altra vegada; el mínim error de validació és al punt final del gràfic, amb un milió de paràmetres.

Per què hi ha corbes com la primera i corbes com la segona? Depèn de com els diferents models fan ús de l'excés de capacitat, i com de bé això encaixa amb el problema d'interès. A mesura que s'afegeix capacitat a una classe de model, sovint s'arriba al punt en què tots els exemples d'entrenament es poden representar perfectament dins el model. Per exemple, donat un conjunt d'entrenament amb n exemples diferents, sempre hi ha un arbre de decisió amb n nodes terminals que pot representar tots els exemples.

Es diu que un model que s'ajusta exactament a totes les dades d'entrenament ha **interpolat** les dades, o les ha memoritzat. Les classes de models típicament comencen a sobreajustar a mesura que la capacitat s'acosta al punt d'interpolació. Sembla que això és perquè la majoria de la capacitat del model es concentra en els exemples

d'entrenament, i la capacitat romanent se situa aleatòriament d'una forma que no és representativa dels patrons en el conjunt de validació. Algunes classes de models ja no es recuperen mai d'aquest sobreajust, com per exemple els arbres de decisió. Però en altres classes de models, afegir capacitat significa que hi ha més funcions candidates, i algunes d'elles estaran ben ajustades als patrons de les dades a la funció veritable $f(x)$. Com més gran és la capacitat, més representacions adequades com aquestes hi ha, i és més probable que el mecanisme d'optimització arribi a una d'elles.

Les xarxes neuronals profundes, així com les màquines de kernel, els *random forests* i els *boosted ensembles* tenen tots la propietat que el seu error al conjunt de validació decreix a mesura que la capacitat creix.

Es podria estendre l'algorisme de selecció de model de diverses formes: es podrien comparar diverses classes de models, com per exemple invocant MODEL-SELECTION amb paràmetres com DECISION-TREE-LEARNER primer i després POLYNOMIAL-LEARNER, i mirant quin ho fa més bé. Es podrien permetre múltiples hiperparàmetres, cosa que significa que necessitaríem un algorisme d'optimització més complex, com per exemple la cerca en graella (**grid search**) en lloc d'una cerca lineal.

2.2. De les taxes d'error a la pèrdua

Fins ara, hem mirat de minimitzar la taxa d'error. Això està clar que és millor que no maximitzar-la, però no és tot el que es pot fer. Per exemple, considerem el problema de classificar els correus entre *spam* i *no-spam*. És pitjor classificar un no-spam com a spam i perdre un missatge important que no a l'inrevés i simplement perdre uns segons de molèstia. Així, un classificador amb una taxa d'error de l'1%, en què gairebé totes les errades fossin classificar spam com a no-spam, seria millor que un classificador amb només una taxa del 0,5%, si la majoria d'aquests errors fossin classificar no-spam com a spam. Els sistemes de presa de decisions han de maximitzar la utilitat esperada, i aquesta utilitat és el que els sistemes d'aprenentatge han de maximitzar, també. No obstant això, en aprenentatge automàtic és habitual expressar-ho a la inversa: minimitzar una **funció de pèrdua** (*loss function*) més que no maximitzar una funció d'utilitat. La funció de pèrdua $L(x, y, \hat{y})$ es defineix com la quantitat d'utilitat perduda en predir $h(x) = \hat{y}$ quan la resposta correcta és $f(x) = y$:

$L(x, y, \hat{y}) = Utilitat$(resultat d'usar y donada una entrada x $)-Utilitat$(resultat d'usar $\hat{y}$ donada una entrada x $)$

Aquesta és la formulació més general de la funció de pèrdua. Sovint se n'usa una versió simplificada, $L(x, \hat{y})$, independent de x. Així, no podem valorar la diferència a la utilitat segons quin sigui l'emissor del correu, però sí, per exemple, valorar 10 vegades pitjor classificar no-spam com a spam que a l'inrevés.

$$L(spam, no-spam) = 1; L(no-spam, spam) = 10$$

Observem que $L(y, y)$ sempre és zero; per definició no hi ha cap pèrdua quan la classificació és correcta. A les funcions amb sortida discreta, es pot enumerar un valor de pèrdua per a cada possible errada de classificació, però no es poden enumerar totes les possibilitats per a les dades reals. En general, els errors petits són millors que no els grans; dues funcions que implementen aquesta idea són el **valor absolut de la diferència** (la pèrdua L_1), i el **quadrat de la diferència** (la pèrdua L_2). Per a sortides discretes, si n'hi ha prou amb minimitzar la taxa d'error, es pot usar la funció de pèrdua $L_{0/1}$, que té una pèrdua 1 si la resposta és incorrecta.

Pèrdua del valor absolut: $L_1(y, \hat{y}) = |y - \hat{y}|$

Pèrdua de l'error quadràtic: $L_2(y, \hat{y}) = (y - \hat{y})^2$

Pèrdua $0/1$ = $L_{0/1}(y, \hat{y}) = 0$ si $y = \hat{y}$; si no, 1

3. Cas pràctic: detecció de càncer de pit

A continuació vegem l'aplicació dels diferents conceptes de selecció de models i validació sobre el dataset Breast Cancer Wisconsin.

Tenim el quadern complet a https://colab.research.google.com/drive/1M0w--8_ibkxFAuFbEFa5URWCFhaAuXei?usp=sharing

3.1. Validació creuada k-fold estratificada

A la validació creuada k-fold, dividim el conjunt de dades d'entrenament aleatòriament en k plegaments (*folds*) sense reemplaçament; k–1 parts s'utilitzaran per entrenar i la restant per a l'avaluació de rendiment. El procediment es repeteix k vegades, de forma que s'obtenen k models i estimacions de precisió.

Una lleugera millora sobre el procediment estàndard de validació creuada s'aconsegueix amb la validació creuada k-fold estratificada. Aquesta variant aconsegueix millors estimacions del biaix i la variància, especialment en casos de proporcions de classe desiguals.

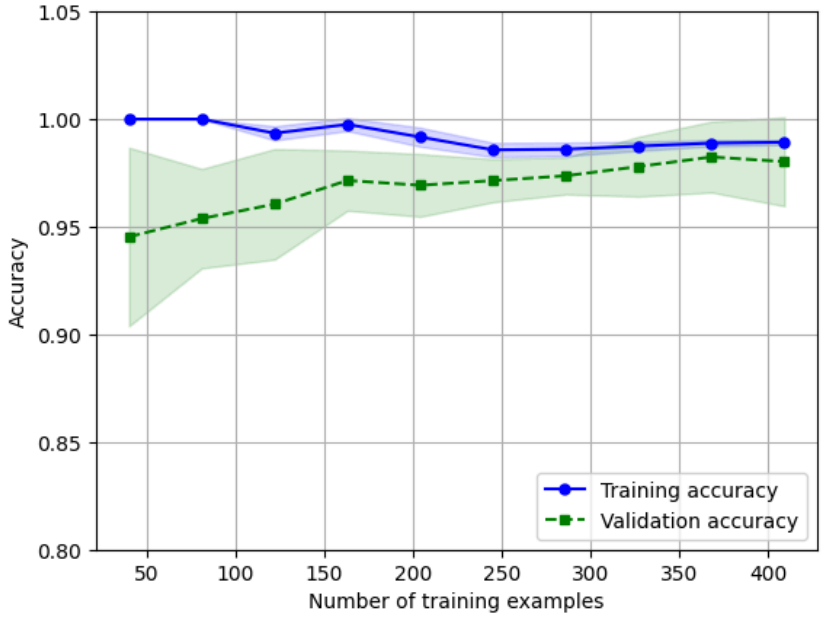
A la validació creuada k-fold estratificada, les proporcions d'etiqueta de classe es mantenen en cada plegament per assegurar que cada plegament sigui representatiu de les proporcions de classe del conjunt d'entrenament.

Ho il·lustram a continuació mitjançant l'ús de l'iterador **StratifiedKFold** de **scikit-learn**.

3.2. Corba d'aprenentatge

La corba d'aprenentatge representa l'exactitud (*accuracy*) en funció de la mida del conjunt d'entrenament.

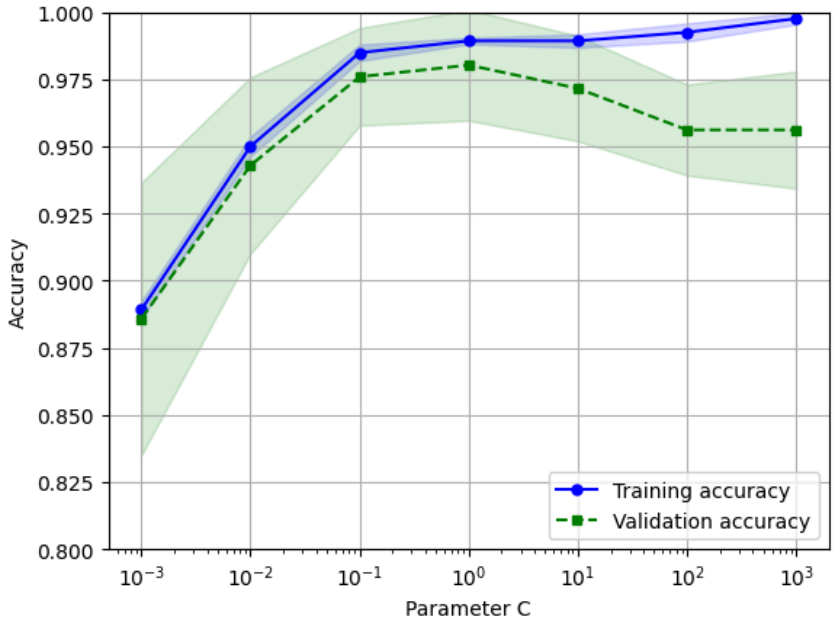
Observam que com més gran és el conjunt d'entrenament més bona és la capacitat de generalització. Per això l'exactitud en dades no observades (les del conjunt de validació) s'acosta a l'exactitud en dades observades (les del conjunt d'entrenament).



3.3. Corba de validació

La corba de validació representa l'exactitud en els conjunts d'entrenament i de validació respecte del valor de l'hiperparàmetre que volem optimitzar.

El valor òptim de l'hiperparàmetre és el que coincideix amb el màxim de l'exactitud de **validació**. A la gràfica de baix, C=1.



3.4. Matriu de confusió

La matriu de confusió és una matriu quadrada que mostra el recompte dels veritables positius (TP), veritables negatius (TN), falsos positius (FP) i falsos negatius (FN).

```
from sklearn.metrics import confusion_matrix

pipe_svc.fit(X_train, y_train)
y_pred = pipe_svc.predict(X_test)
confmat = confusion_matrix(y_true=y_test,
                           y_pred=y_pred)
print(confmat)
```

```
[[71  1]
 [ 2 40]]
```

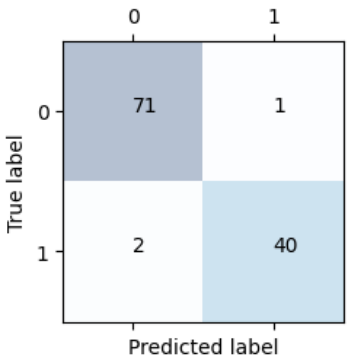
En aquest cas, els valors són

TP = 71

FN = 1

FP = 2

TN = 40



3.5. Precisió i reclam

Tant l'error de predicció (ERR) com l'exactitud o *accuracy* (ACC) donen informació general sobre quants d'exemples estan mal classificats. Les seves expressions són les següents.

$$ERR = \frac{FP + FN}{FP + FN + TP + TN}$$

$$ERR = \frac{TP + TN}{FP + FN + TP + TN} = 1 - ERR$$

Les taxes de falsos positius i veritables positius es defineixen així:

$$FPR = \frac{FP}{FP + TN}$$

$$TPR = \frac{TP}{FN + TP}$$

A partir dels recomptes de positius i negatius, es defineixen la precisió (*precision*) i el reclam (recall) de la forma següent.

$$PRE = \frac{TP}{TP + FP}$$

$$REC = \frac{TP}{FN + TP}$$

Per equilibrar la influència d'aquestes dues mesures, sovint s'utilitza la seva mitjana harmònica.

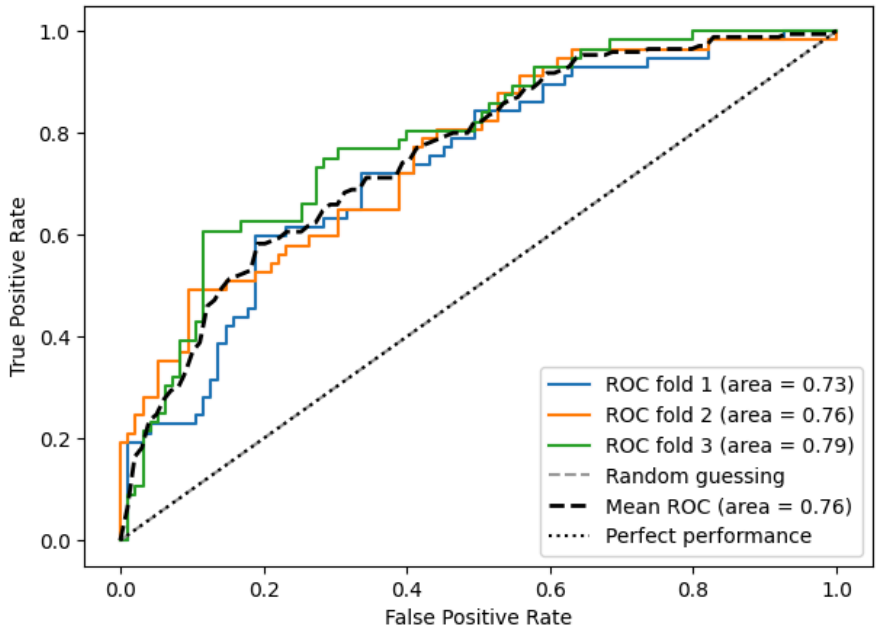
$$F_1 = 2\frac{PRE \times REC}{PRE + REC}$$

3.6. ROC i AUC

Els gràfics de Receiver Operating Characteristic (ROC) són eines útils per triar models de classificació segons les taxes FPR i TPR, que es calculen desplaçant el llindar de decisió del classificador.

La diagonal del gràfic ROC es pot interpretar com una decisió a l'atzar, i els models per baix de la diagonal són pitjors que decidir a l'atzar. Un classificador perfecte seria al racó de dalt a l'esquerra, amb TPR=1 i FPR=0.

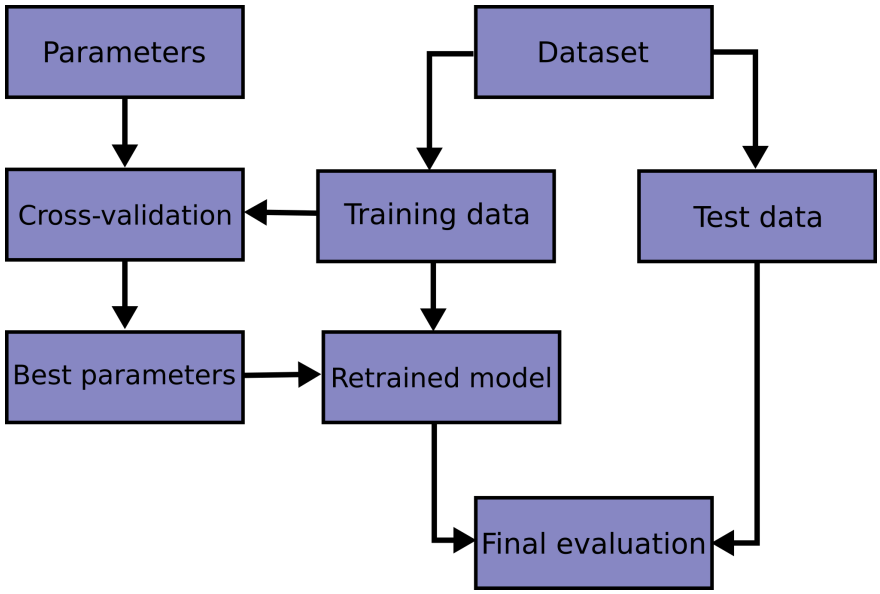
A partir de la corba ROC, es calcula l'àrea sota la corba (**AUC**, *Area Under the Curve*), que caracteritza el rendiment d'un model de classificació. El valor òptim ideal és AUC=1.



A partir d'aquí veurem detalls de les funcions implementades a la llibreria scikit-learn, així com mètriques per a aplicacions de regressió i clusterització.

4. Validació creuada: avaluació del rendiment

Utilitzar les mateixes dades per aprendre els paràmetres d'una funció de predicció i per testar-la és un error metodològic. Un model que simplement repetís les etiquetes dels exemples que ha vist tendria una puntuació perfecta però seria incapaç de predir res útil respecte de dades no vistes. Aquesta situació és el **sobreajust** (*overfitting*). Per evitar-la, quan es fa un experiment d'aprenentatge automàtic supervisat es reserva una part de les dades disponibles com a conjunt de test. (X_test, y_test). Al diagrama següent veim un flux de treball típic de validació creauda en entrenament de models. Els paràmetres òptims es poden determinar mitjançant tècniques de cerca de graella. Ho veurem a l'apartat d'afinació de paràmetres.



Imatge: *Flux de treball d'entrenament d'un model amb validació creuada*

Amb scikit-learn, podem usar la funció auxiliar train_test_split per obtenir un repartiment aleatori entre els conjunts d'entrenament i test. Carreguem les dades Iris i ajustem-hi una màquina de suport vectorial.

```
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn import datasets
from sklearn import svm

X, y = datasets.load_iris(return_X_y=True)
X.shape, y.shape
```

```
((150, 4), (150,))
```

Ara podem mostrejar un conjunt d'entrenament reservant un 40% de les dades per a avaluar (*test*) el classificador.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=0)
```

Comprovem la mida dels tensors.

```
X_train.shape, y_train.shape
```

```
((90, 4), (90,))
```

```
X_test.shape, y_test.shape
```

```
((60, 4), (60,))
```

Ajustam el model i calculam la taxa d'encert sobre el conjunt de test.

```
clf = svm.SVC(kernel='linear', C=1).fit(X_train, y_train)
clf.score(X_test, y_test)
```

```
0.96...
```

Quan s'avaluen diferents ajustos (hiperparàmetres) per als estimadors, com per exemple, el coeficient C que s'ha d'ajustar manualment en un SVM, encara hi ha el risc de sobreajustar sobre el conjunt de test perquè els paràmetres es poden ajustar fins que el model obté un resultat òptim. D'aquesta forma, el coneixement sobre el conjunt de test es pot filtrar dins el model i la mètrica d'avaluació ja no representaria la capacitat de generalització. Per evitar aquest problema, encara es pot reservar una altra part del conjunt de dades, l'anomenat **conjunt de validació**.

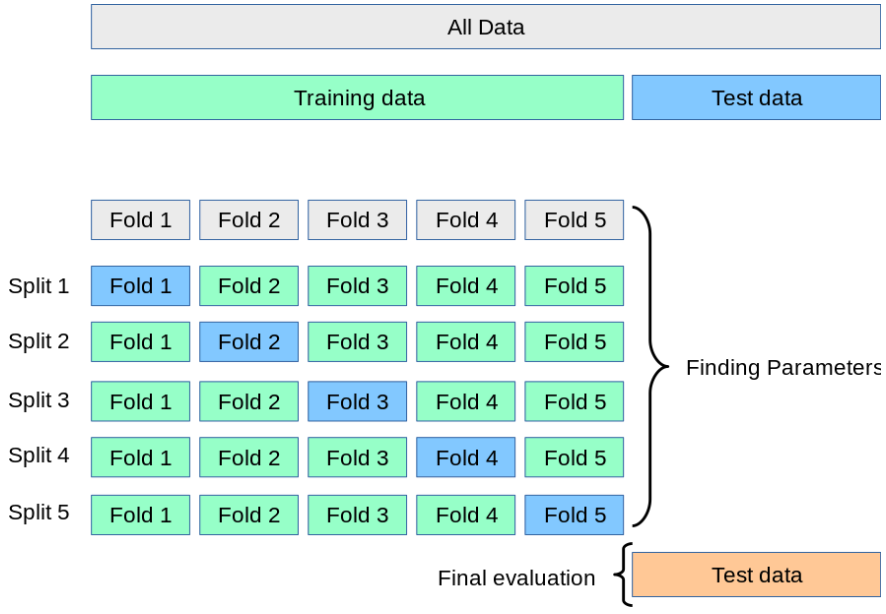
Es fa l'entrenament sobre el conjunt d'entrenament, l'avaluació dels diferents hiperparàmetres es fa sobre el conjunt de validació, i una vegada s'ha triat el millor model, l'avaluació final es fa sobre el conjunt de test.

El problema ara és que quan es divideixen les dades en tres conjunts, es redueix dràsticament el nombre de mostres que s'usen per a l'aprenentatge dedl model, i els resultats poden dependre d'una tria aleatòria del parell de conjunts d'entrenament i validació.

Una solució a aquest problema és l'anomenada **validació creuada** (*cross-validation*, **CV**). S'ha de reservar un conjunt de test per a l'avaluació final, però no cal el conjunt de validació quan es fa CV. En la versió bàsica, k-fold CV, el conjunt d'entrenament es reparteix en k conjunts més petits. Llavors se segueix el procediment següent per a cada un dels k *folds*.

- S'entrena un model amb cada un dels $k - 1$ plecs (*folds*) com a conjunt d'entrenament.
- El model resultant es valida sobre la part romanent de les dades.

La mesura de rendiment final de la validació creuada k-fold és finalment el promig dels valors calculats al bucle.



Imatge: *Validació creuada k-fold*

5. Cerca d'hiperparàmetres

Els hiperparàmetres són paràmetres que no s'aprenen directament dins els estimadors. Dins scikit-learn són arguments que es passen al constructor de les classes dels estimadors. Exemples típics d'hiperparàmetres són C, kernel i gamma en els classificadors de Suport Vectorial, alpha per a Lasso, l'ordre d'una aproximació polinòmica, el nombre de neurones en cada capa d'una xarxa...

Es recomana fer una cerca en l'espai d'hiperparàmetres per aconseguir el millor resultat de validació creuada.

Qualsevol paràmetre dels que es donen quan es construeix un estimador es pot optimitzar d'aquesta manera. Concretament, per trobar els noms i valors actuals de tots els paràmetres d'un estimador donat, podem usar la instrucció següent.

```
estimator.get_params()
```

Una cerca consisteix en:

- un estimador (un regressor o classificar com per exemple **sklearn.svm.SVC()**)
- un espai de paràmetres
- un mètode per cercar o mostrejar els candidats
- un esquema de validació creuada i
- una mètrica objectiva

Hi ha dues aproximacions a la cerca de paràmetres: per a uns valors concrets, GridSearchCV considera totes les combinacions de paràmetres exhaustivament, mentre que RandomizedSearchCV pot mostrejar un nombre determinat de candidats de l'espai de paràmetres amb una distribució especificada. Les dues eines tenen versions HalvingGridSearchCV i HalvingRandomSearchCV, que poden ser molt més ràpides a trobar una bona combinació de paràmetres.

5.1. Cerca exhaustiva de graella

La cerca que dona GridSearchCV genera tots els candidats de la graella de paràmetres especificada al paràmetre **param_grid**. Per exemple, podem indicar el següent.

```
param_grid = [
  {'C': [1, 10, 100, 1000], 'kernel': ['linear']},
  {'C': [1, 10, 100, 1000], 'gamma': [0.001, 0.0001], 'kernel': ['rbf']},
 ]
```

Així s'indica que cal explorar dues graelles, una amb un kernel lineal i valors de C iguals a 1,10,100 i 1000; l'altra amb kernel de funció de base radial, els mateixos valors de C i gamma igual a 0.001 i 0.0001.

Exhaustive Grid Search

5.2. Optimització de paràmetres randomitzada

Tot i que el mètode més emprat d'optimització d'hiperparàmetres és una graella dels valors, altres mètodes tenen propietats més favorables. La funció **RandomizedSearchCV** implementa una cerca aleatoritzada sobre els paràmetres, a partir de la seva funció de distribució. Això té dos avantatges damunt la cerca exhaustiva:

- Es pot triar un pressupost de computació independentment del nombre de paràmetres i dels possibles valors
- Afegir paràmetres que no afectin al rendiment del model no redueix l'eficiència de l'optimització.

Per especificar com s'han de mostrejar els paràmetres es fa amb un diccionari, molt semblant a com s'especifiquen els paràmetres amb GridSearchCV. A més, amb el paràmetre n_iter s'especifica un pressupost de computació. Per a cada paràmetre, es pot indicar un una distribució sobre els possibles valors o bé una llista d'opcions discretes, que es mostrejarien uniformement.

```
{'C': scipy.stats.expon(scale=100), 'gamma': scipy.stats.expon(scale=.1),
  'kernel': ['rbf'], 'class_weight':['balanced', None]}
```

Aquest codi usa el mòdul **scipy.stats**, que conté moltes distribucions útils per mostrejar paràmetres, com ara **expon**, **gamma**, **uniform**, **loguniform** o **randint**.

En els paràmetres continus, com la C de més amunt, és important especificar una distribució contínua per aprofitar l'aleatorització. D'aquesta forma, incrementar **n_iter** sempre durà a una cerca més fina.

Una variable aleatòria contínua de tipus log-uniform és la versió contínua d'un paràmetre espaiat logarítmicament. Per exemple, per especificar l'equivalent de C de dalt, es pot usar loguniform(1,100) en comptes de [1,10,100].

Refent l'exemple de l'apartat anterior de cerca en graella, podem especcificar una variable aleatòria contínua distribuïda log-uniformement entre 1e0 i 1e3 (entre 1 i 1000).

```
from sklearn.utils.fixes import loguniform
{'C': loguniform(1e0, 1e3),
 'gamma': loguniform(1e-4, 1e-3),
 'kernel': ['rbf'],
 'class_weight':['balanced', None]}
```

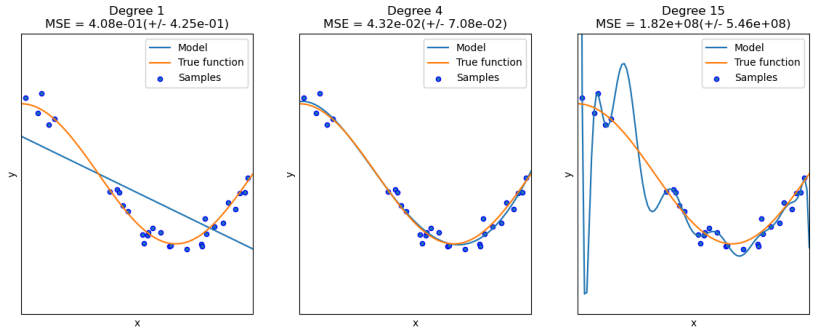
Randomized Parameter Optimization

6. Corbes de validació

Cada estimador té els seus avantatges i els seus inconvenients. El seu error de generalització es pot descomposar sen els termes de **biaix**, **variància** i **soroll**.

- El **biaix** d'un estimador és el seu error promig per a diferents conjunts d'entrenament.
- La **variància** d'un estimador indica com és de sensible a conjunts d'entrenament canviants.
- El **soroll** és una propietat de les dades.

Al gràfic següent, veim una funció /(f(x) = \cos(\frac{3}{2}\pix) /) i algunes mostres amb soroll d'aquesta funció. Utilitzam tres estimadors diferents per ajustar la funció: regressió lineal amb característiques polinòmiques de grau 1, 4 i 15. Observam que el primer estimador dona només un ajust pobre a les mostres i a la funció real perquè és massa simple (alt biaix, *high bias*); el segon estimador fa una aproximació gairebé perfecta; i el darrer estimador s'ajusta a les dades d'entrenament perfectament bé però no s'ajusta a la funció gaire bé: això significa que és molt sensible als canvis en les dades d'entrenament (alta variància, *high variance*).



El biaix i la variància són propietats inherents als estimadors i hem de triar els algorismes d'aprenentatge i els hiperparàmetres de forma que tant el biaix com la variància siguin tan baixos com sigui possible.

Una altra forma de reduir la variància d'un model és usar més dades d'entrenament. Tanmateix, només hauríem de recollir més dades d'entrenament si la veritable funció és massa complexa per aproximar-la amb un estimador de variància més petita.

En aquest problema simple unidimensional de l'exemple anterior, és senzill veure si l'estimador bé alt biaix o alta variància. No obstant això, en espais de dimensionalitat elevada, els models poden ser molt mals de visualitzar. Per això sovint és útil usar les eines que descrivim a continuació: la corba de validació i la corba d'aprenentatge.

https://scikit-learn.org/stable/modules/learning_curve.html#validation-curves-plotting-scores-to-evaluate-models

6.1. Corba de validació

Per validar un model hem de menester una funció de ***scoring***, per exemple l'**exactitud** (*accuracy*) en el cas dels classificadors.

La manera adequada de triar múltiples hiperparàmetres d'un estimador és la **cerca en graella** (*grid search*) o mètodes semblants, que seleccionaran els valors d'hiperparàmetres amb la màxima puntuació sobre un conjunt de validació o sobre diversos conjunts de validació.

Hem d'observar que si optimitzam els hiperparàmetres en base a una puntuació de validació aquesta puntuació de validació és esbiaixada i ja no és una bona mesura de la capacitat de generalització.

Per obtenir una bona mesura de la generalització del model hem de calcular la puntuació sobre un altre conjunt de dades de prova.

Tanmateix, de vegades és útil representar la influència d'un hiperparàmetre en la puntuació d'entrenament i la puntuació de validació, per determinar si l'estimador està infraajustat o sobreajustat per a alguns valors de l'hiperparàmetre.

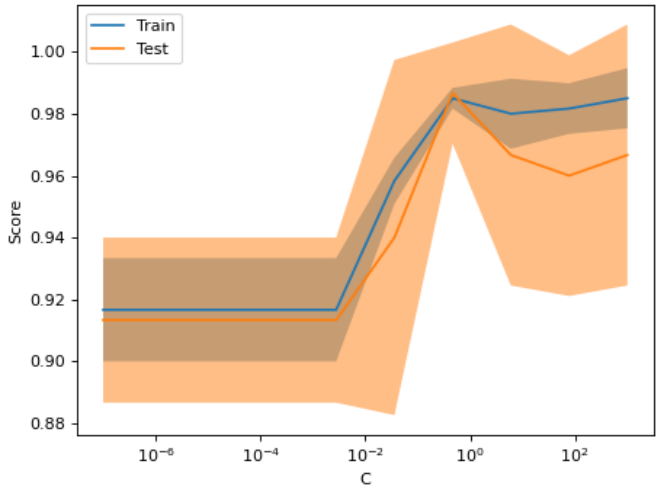
En aquest cas és útil la funció validation_curve.

```
import numpy as np
from sklearn.model_selection import validation_curve
from sklearn.datasets import load_iris
from sklearn.svm import SVC

np.random.seed(0)
X, y = load_iris(return_X_y=True)
indices = np.arange(y.shape[0])
np.random.shuffle(indices)
X, y = X[indices], y[indices]

train_scores, valid_scores = validation_curve(
    SVC(kernel="linear"), X, y, param_name="C", param_range=np.logspace(-7, 3, 3),
)
```

Si només tenim interès a representar les corbes de validació, la classe ValidationCurveDisplay és més directa que no usar matplotlib manualment a partir dels resultats de la crida a **validation_curve**.

Podem usar el mètode from_estimator de forma semblant a **validation_curve** per generar i mostrar la corba de validació.

```
from sklearn.datasets import load_iris
from sklearn.model_selection import ValidationCurveDisplay
from sklearn.svm import SVC
from sklearn.utils import shuffle
X, y = load_iris(return_X_y=True)
X, y = shuffle(X, y, random_state=0)
ValidationCurveDisplay.from_estimator(
   SVC(kernel="linear"), X, y, param_name="C", param_range=np.logspace(-7, 3, 10)
)
```



- Si la puntuació d'entrenament i la puntuació de validació són baixes totes dues, l'estimador està infraajustat. (C<1 a la gràfica)
- Si la puntuació d'entrenament és alta i la puntuació de validació és baixa, l'estimador està sobreajustat. (C>1 a la gràfica)
- Si les dues puntuacions són altes, l'estimador està funcionant molt bé. (C=1 a la gràfica)

Normalment és impossible que hi hagi una puntuació d'entrenament baixa i una puntuació de validació alta.

https://scikit-learn.org/stable/modules/learning_curve.html#validation-curve

6.2. Corba d'aprenentatge

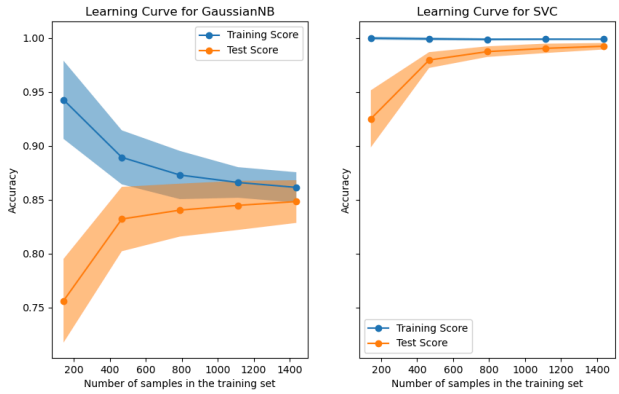
La **corba d'aprenentatge** (en anglès, *learning curve*) mostra les puntuacions de validació i entrenament d'un estimador per a diferents mides del conjunt d'entrenament.

És una eina que serveix per a saber com de molt ens beneficiam d'afegir més dades d'entrenament i si l'estimador té més error de variància o error de biaix.

Considerem l'exemple següent, en que mostram la corba d'aprenentatge d'un classificador naive Bayes i una màquina de suport vectorial.

Per al naive Bayes, tant la puntuació de validació com la puntuació d'entrenament convergeixen a un valor que és bastant baix, a mesura que creix la mida del conjunt d'entrenament. Per tant, probablement ja no treurem gaire profit de més dades d'entrenament.

En canvi, per a quantitats petites de dades, la puntuació d'entrenament del SVM és molt més gran que la puntuació de validació. Afegir més exemples d'entrenament probablement millorarà la generalització.

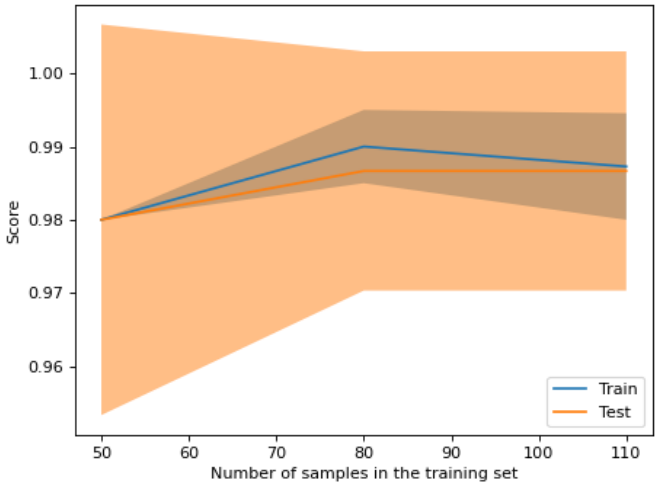


Podem usar la funció learning_curve per generar els valors necessaris per representar aquesta corba d'aprenentatge: nombre de mostres que s'han usat, puntuacions mitjanes ens els conjunts d'aprenentatge i puntuacions mitjanes ens els conjunts de validació.

```
from sklearn.model_selection import learning_curve
from sklearn.svm import SVC

train_sizes, train_scores, valid_scores = learning_curve(
    SVC(kernel='linear'), X, y, train_sizes=[50, 80, 110], cv=5)
train_sizes
train_scores
valid_scores
```

Si només tenim interès a la gràfica, la classe LearningCurveDisplay serà més simple de fer servir. Podem emprar el mètode from_estimator de forma semblant a learning_curve per crear i mostrar la corba d'aprenentatge.

```
from sklearn.datasets import load_iris
from sklearn.model_selection import LearningCurveDisplay
from sklearn.svm import SVC
from sklearn.utils import shuffle
X, y = load_iris(return_X_y=True)
X, y = shuffle(X, y, random_state=0)
LearningCurveDisplay.from_estimator(
   SVC(kernel="linear"), X, y, train_sizes=[50, 80, 110], cv=5)
```



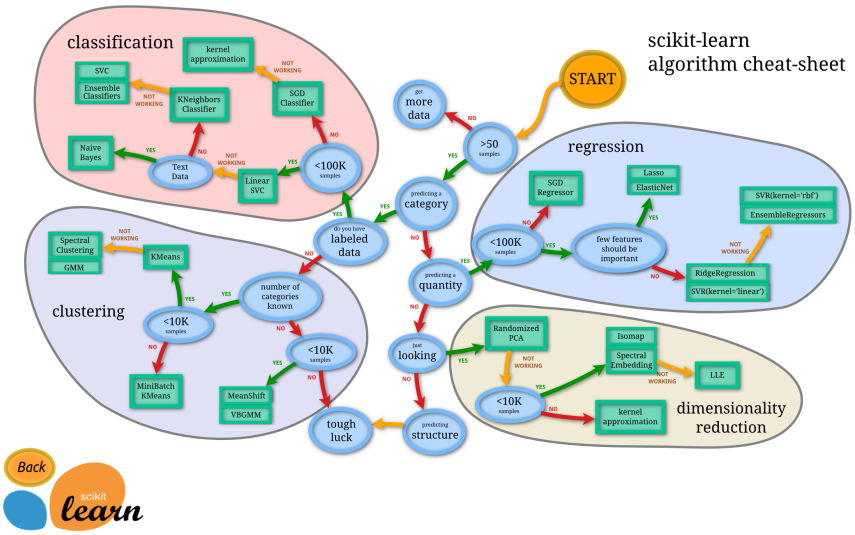
https://scikit-learn.org/stable/modules/learning_curve.html#learning-curve

7. Validació segons tipus de model

En aquestes alçades de curs ja hem vist una panoràmica de models d'aprenentatge automàtic, dividits en els dos grans grups d'**aprenentatge supervisat** i aprenentatge no supervisat.

Quan disposam dels valors de la variable predita, si aquests valors són discrets realitzam classificació, i si els valors són continus, regressió.

Quan no disposam dels valors de la variable predita, realitzam **agrupament** o clústering quan volem estudiar l'estructura de les dades o **reducció de la dimensionalitat** quan cercam una representació en un espai més simple. Aquesta representació pot tenir una funció simplement exploratòria, per visualitzar les dades, o pot ser la primera etapa d'una anàlisi posterior.



Imatge: *Visió general dels algorismes d'aprenentatge automàtic*

Segons quina sigui la tasca que realitzam (classificació, regressió o clústering) les mètriques d'avaluació dels models són diferents, d'acord amb les característiques del problema.

Així, veurem que la classificació es basa en mesures derivades de la **matriu de confusió**, primer per a classificació binària i després en classificació multiclasse. Per això, caldrà definir primer les taxes de falsos positius, falsos negatius, veritables positius i veritables negatius. A partir d'aquí, es defineixen les mesures de precisió (precision) i sensitivitat (recall), i la mesura F1-score que les agrupa en un sol valor. A continuació definirem la corba ROC i l'àrea sota aquesta corba (AUC, *area under the curve*), com a mesura compacta de resum de la corba.

Per a la regressió hi ha diferents mesures de la bondat d'ajust, entre les quals la més important és el **coeficient de determinació**.

En situacions de clústering la mesura més destacada és el **coeficient de Silhouette**.

Metrics and scoring: quantifying the quality of predictions

8. Validació en problemes de classificació

En aquest apartat veurem les mètriques següents per avaluar els models de classificació.

- Exactitud (*accuracy*)
- Matriu de confusió
- Precisió, reclam i mesura F
- Corba ROC

8.1. Exactitud (accuracy)

L'exactitud o *accuracy* representa la fracció de prediccions correctes.

A la classificació multi-classe, la funció retorna la precisió del subconjunt. Si tot el conjunt d'etiquetes predites per a una mostra coincideix estrictament amb el conjunt vertader d'etiquetes, aleshores la precisió del conjunt és 1; en cas contrari, és 0.

A la classificació multi-etiqueta, la funció retorna la precisió del subconjunt. Si tot el conjunt d'etiquetes predites per a una mostra coincideix estrictament amb el conjunt vertader d'etiquetes, aleshores la precisió del subconjunt és 1; en cas contrari, és 0.

$$accuracy(y, \hat{y}) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} si(\hat{y_i} = y_i)$$

Vegem-ho en un exemple de codi.

```
>>> import numpy as np
>>> from sklearn.metrics import accuracy_score
>>> y_pred = [0, 2, 1, 3]
>>> y_true = [0, 1, 2, 3]
>>> accuracy_score(y_true, y_pred)
0.5
>>> accuracy_score(y_true, y_pred, normalize=False)
2.0
```

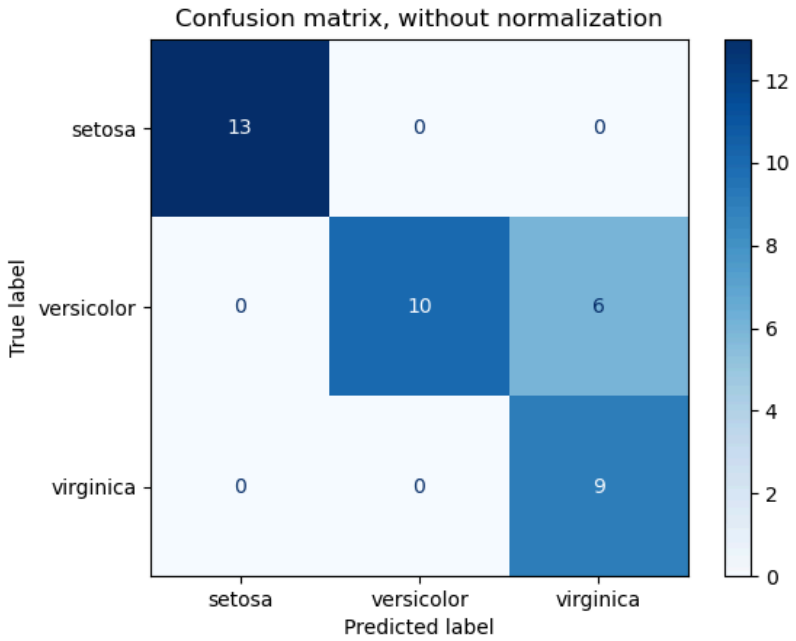
Referència: https://scikit-learn.org/stable/modules/model_evaluation.html#accuracy-score

8.2. Matriu de confusió

La matriu de confusió serveix per a avaluar l'exactitud (*accuracy*) de la classificació, en la qual cada filera correspon a la classe esperada. Per definició, l'entrada en una matriu de confusió i, j és el nombre d'observacions que realment són al grup i, però que es prediu que són al grup j. Aquí en tenim un exemple.

```
>>> from sklearn.metrics import confusion_matrix
>>> y_true = [2, 0, 2, 2, 0, 1]
>>> y_pred = [0, 0, 2, 2, 0, 2]
>>> confusion_matrix(y_true, y_pred)
array([[2, 0, 0],
       [0, 0, 1],
       [1, 0, 2]])
```

A continuació veim un exemple de representació de la matriu de confusió, usant la funció ConfusionMatrixDisplay.



Imatge: *Matriu de confusió en classificació de tres classes*

El paràmetre **normalize** permet calcular proporcions en lloc de recomptes. La matriu de confusió es pot normalitzar de tres formes diferents: 'pred', 'true' i 'all', que dividiran els recomptes per la suma de cada columna, filera o la matriu completa, respectivament. Cada tipus de normalització és adequada en diferents situacions.

```
y_true = [0, 0, 0, 1, 1, 1, 1, 1]
y_pred = [0, 1, 0, 1, 0, 1, 0, 1]
confusion_matrix(y_true, y_pred, normalize='all')
```

En problemes binaris (classificació en dues classes) podem obtenir recomptes de veritables negatius, falsos positius, falsos negatius i veritables positius de la forma següent.

```
y_true = [0, 0, 0, 1, 1, 1, 1, 1]
y_pred = [0, 1, 0, 1, 0, 1, 0, 1]
tn, fp, fn, tp = confusion_matrix(y_true, y_pred).ravel()
tn, fp, fn, tp
```

Referència: https://scikit-learn.org/stable/modules/model_evaluation.html#confusion-matrix

8.3. Precisió, reclam i mesura F

La **precisió** (en anglès, *precision*) és la capacitat de classificador de no etiquetar com a positiva una mostra que és negativa, i el **reclam** (en anglès, *recall*) és la capacitat del classificador de trobar tots els exemples positius.

La **mesura F** (F_β i F_1) és una mitjana harmònica ponderada de la precisió i el reclam. Una mesura F_β ateny el seu millor valor a 1 i el pitjor a 0. Amb $\beta = 1$, F_β i F_1 són equivalents, i el reclam i la precisió són igualment importants.

Classificació binària

En una tasca de classificació binària, els termes positiu i negatiu es refereixen a la predicció del classificador, i els termes veritable (*true*) i fals (*false*) indiquen si la predicció correspon a l'observació. Amb aquestes definicions, es pot formula la taula següent.

	classe actual (observació)	
classe predita	tp (true positive) resultat correcte	fp (false positive) resultat inesperat
	fn (false negative) resultat que manca	tn (true negative) manca de resultat, correcta

Taula: *Taula amb tp, fp, fn i tn.*

En aquest context, es defineixen la precisió i el reclam de la forma següent.

$$precision = \frac{tp}{tp + fp}$$

$$recall = \frac{tp}{tp + fn}$$

La mesura F és la mitjana harmònica ponderada de la precisió i el reclam, amb la contribució de la precisió ponderada pel paràmetre β.

$$F_\beta = (1 + \beta^2)\frac{precision \times recall}{\beta^2 precision + recall}$$

Referència: https://scikit-learn.org/stable/modules/model_evaluation.html#precision-recall-and-f-measures

8.4. Corba ROC

Una corba ROC (Receiver Operating Characteristic) és una representació gràfica que il·lustra la relació entre la sensibilitat i l'especificitat d'un sistema classificador per a diferents punts de tall. Les corbes ROC es varen desenvolupar la dècada de 1950 per analitzar senyals amb soroll per caracteritzar el compromís entre encerts i falses alarmes. Són emprades per comparar visualment distints models de classificació.

La corba ROC representa l'ajust entre la FPR (especificitat) i la TRP (sensibilitat). Aquestes dues mesures es defineixen a partir dels vertaders positius (TP, *True Positives*), falsos positius (FP, *False Positives*), vertaders negatius (TN, True Negatives) i falsos negatius (FN).

$$TPR = \frac{TP}{TP + FN}$$

$$FPR = \frac{FP}{FP + TN}$$

El classificador al cantó superior esquerre especifica que el rendiment és millor. La corba és més poc precisa si és més a prop dels 45 graus de l'espai ROC.

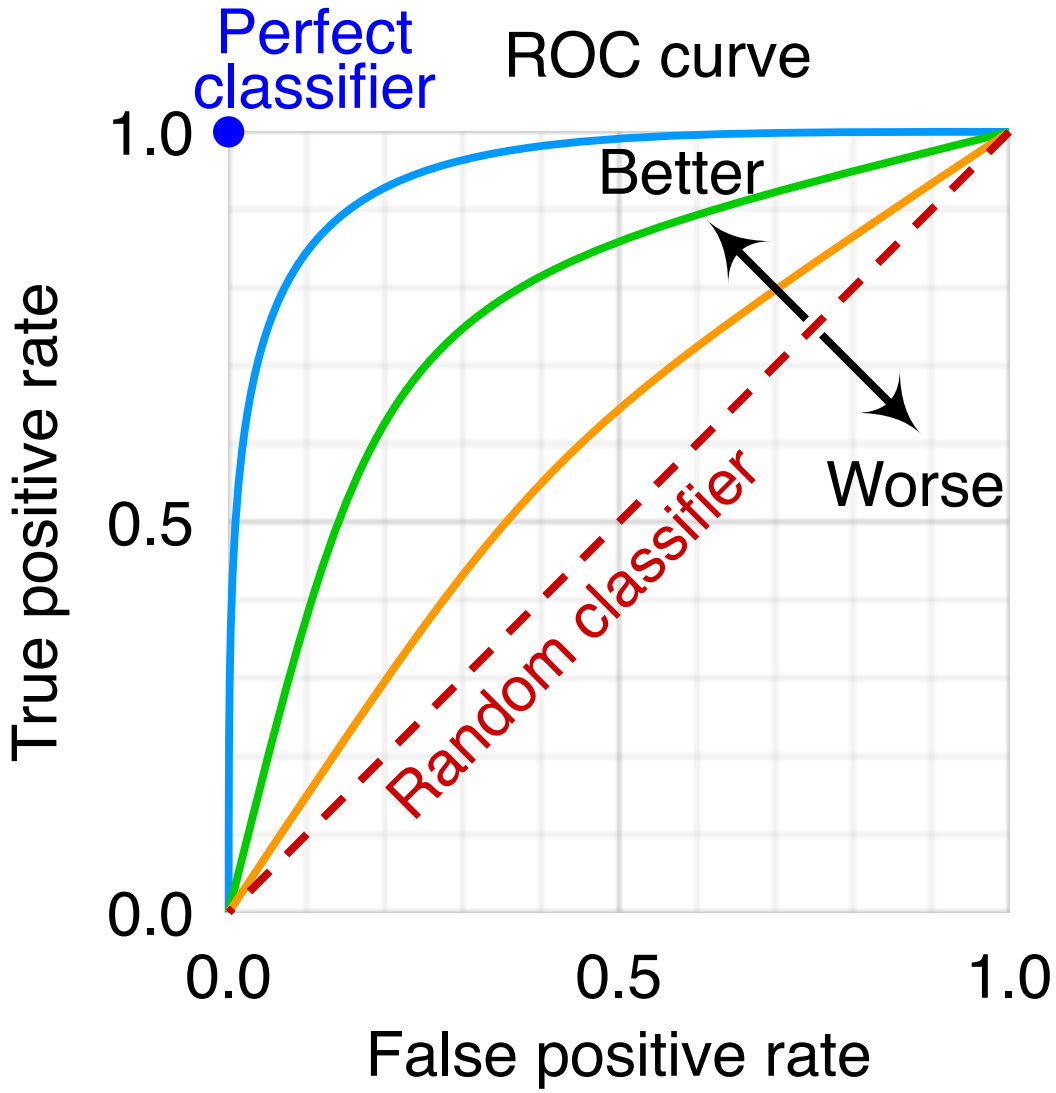
La construcció de la corba segueix l'algorisme següent.

S'usa un classificador que predigui la probabilitat que un exemple E pertanyi a la classe positiva $P(+|E)$

S'ordenen els exemples en ordre decreixent del valor estimat $P(+|E)$.

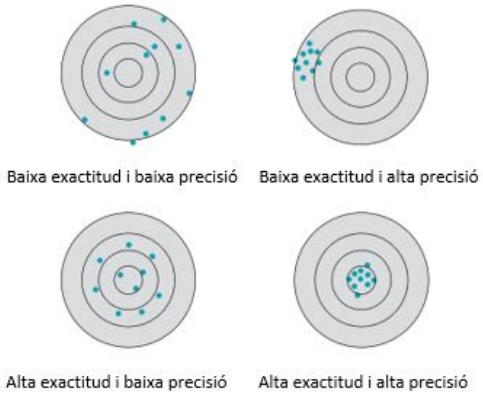
S'aplica un llindar per a cada valor diferent de $P(+|E)$. Per fer-ho, es compta el nombre de TP, FP, TN i FN.

L'àrea sota la corba (*Area Under the Curve*, AUC) és una mesura de la precisió del classificador. Com més a prop sigui de la diagonal (àrea propera a $\frac{1}{2}$, més poc precís serà el model. Un model perfecte tendria àrea 1.



Imatge: *Corba ROC*

8.5. Compromís biaix-variància



Imatge: *Exactitud i precisió*

En anglès, aquest gràfic moltes vegades el trobam amb el nom de **bias-variance tradeoff**.

La filera de dalt és *high bias*, que correspon a baixa exactitud. Per tant, podem traduir bias, a més de com a biaix, com a inexactitud.

La filera de baix és *low bias*, alta exactitud.

La columa de l'esquerra és *high variance*, baixa precisió. Per tant, podem entendre la variància com a imprecisió, manca de precisió al voltant de l'estimació.

La columna de la dreta és *low variance*, alta precisió.

La situació ideal és la d'alta exactitud i alta precisió, *low bias* i *low variance*.

9. Validació en problemes de regressió

El nostre model produirà una sortida per a qualsevol entrada o conjunt d'entrades, de forma que podem comparar aquestes sortides estimades amb els valors reals que intentam predir. Anomenam **residu** o **error** a la diferència entre el valor real i l'estimació del model. Podem calcular el residu per a cada punt del nostre conjunt de dades, i cada un d'aquests residus serà útil a l'avaluació. Aquests residus tendran un paper important a l'hora d'avaluar la utilitat d'un model.

Si el valor agregat del nostre error és petit, això vol dir que el model és bo; si no, si aquests residus són generalment grans, això indica que el model és un mal estimador.

En resum, la qualitat d'un model de regressió mesura com de bé coincideixen les seves prediccions amb els valors reals. Per tal d'avaluar aquesta qualitat s'usaran mesures d'error, i això ens permetrà comparar regressions amb diferents paràmetres. Aquestes mètriques són resums breus i útils de la qualitat dels nostres models.

Variància explicada

La variància explicada mesura la proporció en què un model matemàtic explica la variació (dispersió) d'un conjunt de dades determinat.

Si $\hat{y_i}$ és el valor predit de l'element i, y_i el valor esperat i Var la variància entesa com el quadrat de la desviació típica, aleshores la variància explicada s'estima de la forma següent:

$$variància\_explicada(y,\hat{y}) = 1 - \frac{Var\{y-\hat{y}\}}{Var\{y\}}$$

El valor més bo possible és 1; els valors baixos representen mals resultats.

Error màxim

És el valor més alt de l'error residual, una mètrica que captura el pitjor cas d'error entre el valor predit i el valor real. En un model de regressió de sortida única perfectament ajustat, el valor d'aquesta mètrica seria 0 en el conjunt d'entrenament i, encara que això seria molt poc probable en el món real, aquesta mètrica mostra el grau d'error que tenia el model quan s'ajustà.

Si $\hat{y_i}$ és el valor predit de l'element i, y_i el valor esperat, aleshores l'error màxim es formula així:

$$Error_{max}(y,\hat{y}) = max(|y_i-\hat{y_i}|)$$

Error absolut mitjà

L'error absolut mitjà és el promig del valor absolut de la diferència entre la predicció i el valor esperat.

Si $\hat{y_i}$ és el valor predit de l'element i, y_i el valor esperat, aleshores l'error absolut mitjà sobre $n_{samples}$ elements es formula així:

$$MAE(y,\hat{y}) =\frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} | y_i-\hat{y_i}|$$

El seu ús en Python és el següent

```
from sklearn.metrics import max_error
mean_absolute_error(y_true, y_pred, multioutput='raw_values')
```

El MAE és la mètrica més fàcil d'interpretar, ja que només s'observa la diferència absoluta entre les dades i les prediccions del model. Com que utilitzam el valor absolut del residu, el MAE no indica un rendiment inferior o superior del model (si el model s'ajusta o no a les dades reals). Cada residu contribueix proporcionalment a la quantitat total d'error, cosa que significa que els errors més grans contribuiran linealment a l'error global. D'aquesta forma, un MAE petit suggereix que el model és excel·lent en la predicció, mentre que un MAE gros suggereix que el model pot tenir problemes. Un MAE de 0 significa que el model és un predictor perfecte dels resultats.

Quan s'usa el valor absolut de l'error s'estan tractant per igual tots els errors, però pot ser necessari donar més importància als valors atípics o extrems.

Error quadràtic mitjà

Si $\hat{y_i}$ és el valor predit de l'element i, y_i el valor esperat, l'error quadràtic mitjà MSE sobre $n_{samples}$ elements quedaria definit com:

$MSE(y,\hat{y}) =\frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} (y_i-\hat{y_i})^2$

i la seva utilització en Python és la següent:

```
from sklearn.metrics import max_error
mean_squared_error(y_true, y_pred)
```

Pel fet d'elevar les prediccions al quadrat, l'error creix quadràticament en el cas del MSE. Això significa que els valors atípics a les nostres dades contribuiran a un error total molt més gran en el MSE que no en el MAE. Igualment, els model es veurà més penalitzat per fer prediccions que difereixin molt del valor real corresponent.

Error quadràtic logarítmic mitjà

Correspon al valor esperat de l'error o pèrdua logarítmica elevada al quadrat.

Si $\hat{y_i}$ és el valor predit de l'element i, y_i el valor esperat, l'error quadràtic logarítmic mitjà MSLE sobre $n_{samples}$ elements quedaria definit com:

$MSLE(y,\hat{y}) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} (log(1+y_i)-log(1+\hat{y_i}))^2$

Aquesta mètrica és adequada quan els objectius tenen un creixement exponencial, com els recomptes de població, les vendes mitjanes d'una mercaderia durant un període d'anys, etc. En aquesta mètrica es penalitza més una estimació subestimada que una sobreestimada.

En Python,

```
from sklearn.metrics import mean_squared_log_error
mean_squared_log_error(y_true, y_pred)
```

Error percentual mitjà absolut

L'error percentual mitjà absolut, MAPE, també conegut com a desviació percentual mitjana absoluta MAPD, és una mètrica d'avaluació per als problemes de regressió. La idea d'aquesta mètrica és ésser sensible als errors relatius. Per exemple, no canvia per un escalat global de la variable objectiu.

Si $\hat{y_i}$ és el valor predit de l'element i, y_i el valor esperat, l'error percentual mitjà absolut MSLE sobre $n_{samples}$ elements queda definit com:

$MAPE(y,\hat{y}) = \frac{1}{samples} \sum{i=0}^{n_{samples}-1} \frac{|y_i-\hat{y_i}|}{max(\epsilon,|y_i|)}$,

on ϵ és un nombre arbitrari, petit però positiu, per evitar resultats NaN (*Not a Number*) quan y és zero.

```
from sklearn.metrics import mean_absolute_percentage_error
mean_absolute_percentage_error(y_true,y_pred)
```

El MAPE representa la distància que separa les prediccions del model dels seus resultats corresponents en promig. Igual que el MAE, el MAPE també té una interpretació clara. No obstant això, malgrat tots els seus avantatges, l'ús del MAPE és més limitat que el del MAE. El MAPE pot créixer inesperadament si els valors reals són excepcionalment petits. Finalment, el MAPE està esbiaixat cap a les prediccions que són sistemàticament menors que els propis valors reals. És a dir, el MAPE serà menor quan la predicció sigui menor que la real en comparació amb una predicció que sigui més gran en la mateixa mesura.

Coeficient de determinació R^2

El coeficient de determinació, normalment denominat R^2 representa la proporció de la variància (d'y) que ha estat explicada per les variables independents del model. Proporciona una indicació de la bondat de l'ajust i, per tant, una mesura de la probabilitat que les mostres no vistes siguin predites pel model, a través de la proporció de la variància explicada.

Com que aquesta variància depèn del conjunt de dades, R^2 és possible que no sigui significativament comparable entre diferents conjunts de dades. La millor puntuació possibles és 1 i pot ésser negativa (perquè el model pot ésser arbitràriament pitjor). Un model constant que sempre prediu el valor esperat d'y, sense tenir en compte les característiques d'entrada, obtendria una puntuació R^2 de 0.

Si $\hat{y_i}$ és el valor predit de l'element i, i y_i el valor esperat, el coeficient de determinació R^2 estaria definit com:

$R^2(y,\hat{y}) = 1 -\frac{\sum_{i=1}^n (y_i-\hat{y_i})^2}{ \sum_ {i=1}^n (y_i-\bar{y})^2}$,

on

$\bar{y} = \frac{1}{n} \sum_{i=1}^n y_i$
i

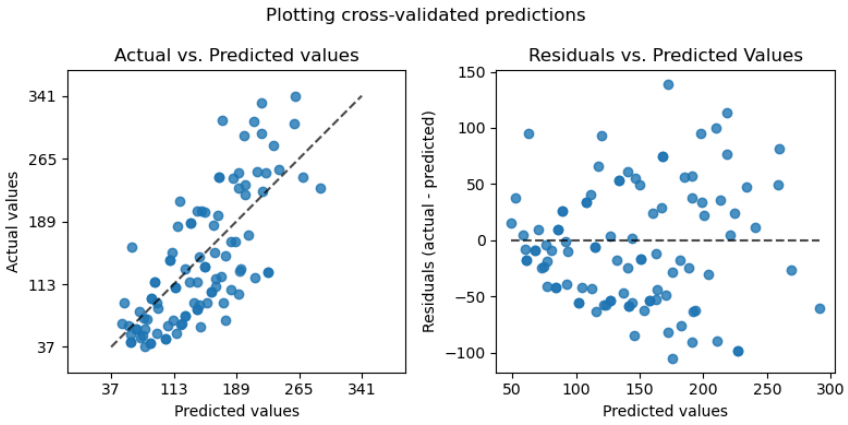
$\sum_{i=1}^n(y_i-\hat{y_i})^2 = \sum_{i=1}^n {\epsilon_i}^2$

El coeficient R^2 es pot incrementar fàcilment afegint un nombre més gran de variables, però això duria a un sobreajust del model. Per compensar aquesta manipulació, es pot recórrer al R^2 ajustat.
Tot i que tant R^2 com R^2 ajustat donen una idea dels punts de dades que cauen a la línia de regressió, l'única diferència que presenten és que el R^2 ajustat troba el percentatge de variació explicat per la variable independent que realment influeix a la variable dependent, mentre que R^2 (sense ajustar) assumeix que cada variable explicació la variació a la variable dependent.

R^2 ajustat és la versió modificada de R^2; el seu valor augmenta només quan la variable en el model li afegeix valor. Com més poc útil sigui la variable present en el model, més petit serà el valor de R^2 ajustat i més gran serà el valor de R^2.

9.1. Avaluació visual de models de regressió

Entre els mètodes per avaluar la qualitat dels models de regressió, scikit-learn facilita la classe PredictionErrorDisplay, que permet inspeccionar els errors de predicció d'un model de dues formes distintes.



Imatge: *Error de predicció*

La gràfica de l'esquerra mostra els valors actuals en funció dels valors predits. En una tasca de predicció sense soroll, un model de regressió perfecte mostraria els punts de dades sobre la diagonal que correspon als valors predits iguals que els valors reals. Com més enfora d'aquesta línia òptima, més gran és l'error del model. En una situació més realista amb soroll irreductible (quan no totes les variacions de \(y\) es poden explicar amb característiques de \(X\)), el millor model portaria a un núvol de punts distribuït densament al voltant de la diagonal.

Tot i que és intuïtiu de llegir, aquest gràfic realment no ens diu què podem fer per millorar el model.

El gràfic de la dreta mostra els residus, és a dir, la diferència entre els valors predits i els valors actuals. Això facilita veure la distribució d'aquests residus.

Quan ajustam un model de regressió de mínims quadrats, podem usar aquest gràfic per comprovar si es compleixen les suposicions del model. Els residus haurien d'estar incorrelats, el seu valor esperat hauria de ser nul i la seva variància constant.

Si no és així, en particular, si els residus segueixen una forma de banana, això és una pista que el model està mal especificat i que una enginyeria de característiques no lineal, o bé passar a un model de regressió no lineal podria ser útil.

En aquest exemple s'il·lustra una avaluació de model que utilitza aquesta representació.

10. Validació en problemes d'anàlisi clúster

Els índexs de validesa dels clústers s'utilitzen per a validar els resultats de l'agrupació i per a trobar un conjunt de clústers que ajusti millors particions naturals per a un conjunt de dades determinat. La majoria dels índexs de validesa existents depenen considerablement del nombre d'objectes de dades en els clústers, dels centroides dels clústers i dels valors mitjans. D'aquesta forma, tenen una tendència a ignorar els clústers petits i els clústers amb baixa densitat.

La presència d'una gran variabilitat en les formes geomètriques dels clústers, les densitats, mides i el nombre de grups, que no sempre poden conèixer a priori, són dificultats importants a l'hora d'agrupar. Molts d'algorismes de clústering (com **k-means**) requereixen que l'usuari defineixi prèviament el nombre de grups abans del procés de clústering. No obstant això, de vegades és impossible conèixer el nombre de clústers amb antelació. Els resultats de l'agrupació depenen de la tria del nombre de grups. La determinacióí del nombre adequat de grups i la validesa de la partició obtinguda són dos problemes fonamentals en el clústering. Trobar el nombre de grups que s'ajusti millor a la partició natural per a un conjunt de dades donat és difícil, ja que per a un mateix conjunt de dades hi ha diverses particions depenent del nivell de detall.

Els índexs de validesa solen utilitzar-se per a obtenir el nombre òptim de grups. Això requereix que un algorisme de clústering s'executi diverses vegades, amb un nombre diferent de grups com a objectiu en cada execució. Una altra opció per identificar el nombre correcte de clústers és millorar la funció d'optimització i descobrir el nombre de clústers dinàmicament durant l'execució de l'algorisme de clústering que satisfaci la nova funció d'optimització.

La majoria dels índexs de validació proposats durant les darreres dècades s'han centrat en la compacitat i la separació. La separació és una mesura de l'aïllament dels clústers entre si i la compacitat és un amesura de la proximitat dels objectes de dades dins un clúster. Un valor baix de variància és un indicador de proximitat. Els membres de cada clúster han d'estar com més a prop millor els uns dels altres, i els clústers han d'estar molt separats. La majoria de les mesures de validesa tenen la tendència a ignorar els clústers amb baixa densitat i no són eficients en la validació de particions que tenen diferents mides i densitats.

10.1. Validació interna

Un índex de validesa interna dels clústers té com a objectiu mesurar com de bé una partició determinada d'un conjunt de dades reflecteix l'estructura subjacent del domini modelitzat.

Coeficient Silhouette

El Coeficient Silhouette representa una de les mètriques de validesa interna més utilitzades en l'aprenentatge automàtic. Una puntuació gran en aquest coeficient es relaciona amb un model amb clústers més ben definits. El Coeficient Silhouette es defineix per a cada element i es compon de dues puntuacions.

- **a**: la distància mitjana entre un element i tota la resta d'elements de la mateixa classe.
- **b**: la distància mitjana entre un element i tota la resta dels elements del clúster més proper.

El Coeficient de Silhouette s per a una sola mostra és defineix com:

$s=\frac{b-a}{max(a,b)}$

El Coeficient Silhouette per a un conjunt de mostres es dona com la mitjana del Coeficient Silhouette per a cada mostra.

La puntuació està limitada entre -1 per a una agrupació incorrecta i $+1$ per a una agrupació molt densa. Les puntuacions al voltant del zero indiquen que els clústers se superposen. La puntuació és més alta quan els clústers són densos i estan ben separats, cosa que es relaciona amb un concepte estàndard de clúster.

El principal inconvenient que presenta aquest coeficient és el seu biaix a favor de les agrupacions convexes, la qual cosa perjudica els grups obtinguts mitjançant criteris de densitat com els de DBSCAN.

El Coeficient Silhouette s'usa habitualment per triar el valor òptim del nombre de clústers a l'algorisme k-means.

```
import numpy as np
from sklearn.cluster import KMeans
kmeans_model = KMeans(n_clusters=5)
kmeans_model.fit(X)
labels=kmeans_model.labels_
metrics.silhouette_score(X, labels, metric='euclidean')
```

Calinski-Harabasz

L'índex Calinski-Harabasz, també conegut com el criteri de relació de variància, es pot usar per avaluar un resultat de clústering. Una puntuació Calinski-Harabasz més alta es relaciona amb un model amb grups més ben definits.

L'índex és el quocient de la suma entre la dispersió entre clústers i la dispersió dins els clústers per a tots els clústers. La dispersió es pren com la suma de distàncies al quadrat.

Per a un conjunt de dades E de mida n_e que s'ha agrupat en k clústers,

$ s=\frac{tr(B_k)}{tr(W_k)}\frac{n_E-k}{k-1} $,

on $tr(B_k)$ és la traça (suma dels elements diagonals) de la matriu de dispersió del grup i $tr(W_k)$ és la traça de la matriu de dispersió del clúster, amb

$ B_k = \sum_{q=1}^k n_q(c_q-c_E)(c_q-c_E)^T $

$ W_k = \sum_{q=1}^k\sum_{x \in C_q} (x-c_q)(x-c_q)^T $

amb C_q el conjunt d'elements del clúster q, c_q el centroide del clúster q, c_E el centre del conjunt de dades E i n_q el nombre d'elements al clúster q.

La puntuació, ràpida de calcular, és més gran quan els clústers són densos i estan ben separats. Això fa que sigui generalment més alt per als clústers convexos que per a altres conceptes de clústers basats en la densitat, tal com passava amb Silhouette.

```
import numpy as np
from sklearn.cluster import KMeans
kmeans_model = KMeans(n_clusters=5)
kmeans_model.fit(X)
labels=kmeans_model.labels_
metrics.calinski_harabasz_score(X,labels)
```

10.2. Validació externa

Aquestes mesures es calculen fent coincidir l'estructura dels clústers amb alguna classificació predefinida d'instàncies a les dades (*ground truth*).

Rand Index

Compara els dos clústers i mira de trobar la proporció d'observacions coincidents i no coincidents entre dues estructures de clústering (\(C\) i \(K\)). El seu valor se situa entre \(0\) i \(1\).

Si \(C\) és una assignació a grups que funciona com a *ground truth* i \(K\) és el clústering aleshores es pot definir:

- a: nombre de parells d'elements que són al mateix grup dins \(C\) i al mateix conjunt dins \(K\).
- b: nombre de parells d'elements que són a diferents grups dins \(C\) i a diferents grups dins \(K\).

El Rand Index no ajustat ve donat per:

\[RI = \frac{a+b}{C_{2}^{n_{samples}} } \]

on \(C_2^{n_{samples}}\) és el nombre total de parells en el conjunt de dades.

Amb codi Python,

```
from sklearn import metrics
metrics.rand_score(labels_true, labels_pred)
```

Com que el Rand Index no garanteix que una distribució aleatòria tengui un valor proper a \(0\), aleshores el \(RI\) s'ajusta com a \(ARI\), descomptant-ne el \(RI\) esperat en una distribució aleatòria.

\[ARI = \frac{RI-E[RI]}{max(RI)-E[RI]}\]

A diferència d'altres índexs, el Rand Index (ajustat o no) necessita el coneixement del *ground truth*, que gairebé mai no està disponible a la pràctica, o l'assignació manual per part d'anotadors humans (com l'aprenentatge supervisat). No obstant això, pot ésser útil en un entorn purament no supervisat com a bloc de construcció per a un índex de consens que es pot utilitzar per a la selecció de models d'agrupació.

```
from sklearn import metrics
metrics.adjusted_rand_score(labels_true, labels_pred)
```

Homogeneïtat, Exhaustivitat i V-Score

Sempre comptant amb la disponibilitat de la referència dels clústers reals (*ground truth*) és possible definir mètriques utilitzant l'anàlisi d'entropia condicional. Es defineixen els dos objectius desitjables següents per a qualsevol assignació de clústers:

- **homogeneïtat**: cada clúster conté només membres d'una única classe.
- **exhaustivitat**: tots els membres d'una classe determinada s'assignen al mateix clúster.

Aquests dos conceptes poden esdevenir mesures de qualitat dels clústers, que queden limitades entre \(0\) i \(1\). Com més gran sigui el valor obtingut, millor. Un gran avantatge d'aquestes mètriques és que no fan cap suposició sobre l'estructura dels clústers.

Les dues mètriques venen definides per les fórmules següents:

\[h = 1 - \frac{H(C|K)}{H(C)} \]

\[c = 1 - \frac{H(K|C)}{H(K)}, \]

on \(H(C|K)\) és l'entropia condiciconal de les classes donades les assignacions dels clústers:

\[H(C|K) = -\sum_{c=1}^{|C|}\sum_{k=1}^{|K|} \frac{n_{c,k}}{n}log\left(\frac{n_{c,k}}{n_k}\right)\]

i \(H(C)\) és l'entropia de les classes i ve donada per

$$H(C) = -\sum_{c=1}^{|C|}\frac{n_c}{n}\log\left(\frac{n_c}{n}\right),$$

amb n el nombre total d'elements, n_c i n_k el nombre d'elements que pertanyen a la classe c i al clúster k, i finalment $n_{c,k}$ el nombre d'elements de la classe c assignats al clúster k.

L'entropia condicional dels clústers donada la classe $H(K|C)$ i l'entropia dels clústers $H(K)$ es defineixen de forma simètrica.

Per equilibrar aquests dos conceptes es proposa la **V-measure** com a mitjana harmònica d'aquestes dues mesures.

$$ v=2\frac{h\cdot c}{h+c} $$

Aquest indicador ofereix una explicació bastant intuïtiva del resultat. Un resultat amb una V-measure baixa ha d'analitzar-se qualitativament en termes d'homogeneïtat i exhaustivitat per conèixer més bé quin tipus d'errors es cometen a l'assignació.

Segons el nombre d'elements, clústers i classes objectiu, un resultat completament aleatori no sempre produirà els mateixos valors d'homogeneïtat, exhaustivitat i V-measure. En particular, no s'obtendran puntuacions nul·les, sobretot quan el nombre de grups sigui gran. Aquest problema es pot ignorar amb més de mil mostres i menys de deu grups. Amb volums de dades més petits o més clústers, és més segur utilitzar un índex ajustat com el Rand Index ajustat (ARI).

```
def calc_metrics (data, labels_pred, labels_true):

  print("Homogeneity: %0.3f" % metrics.homogeneity_score(labels_true, labels_pred))
  print("Completeness: %0.3f" % metrics.completeness_score(labels_true, labels_pred))
  print("V-measure: %0.3f" % metrics.v_measure_score(labels_true, labels_pred))
  print("Adjusted Rand Index: %0.3f"
          % metrics.adjusted_rand_score(labels_true, labels))
  print("Adjusted Mutual Information: %0.3f"
          % metrics.adjusted_mutual_info_score(labels_true, labels))
  print("Silhouette Coefficient: %0.3f"
          % metrics.silhouette_score(data, labels))
```

Mutual Information Index

La informació mútua és una funció que mesura la concordança de les dues assignacions, ignorant les permutacions. Hi ha dues versions normalitzades d'aquesta mesura, la informació mútua normalitzada NMI i la informació mútua ajustada AMI. L'NMI s'utilitza sovint a la literatura, mentre que l'AMI es proposà més recentment i es normalitza respecte de l'atzar.

Les assignacions d'etiquetes aleatòries (uniformes) tenen una puntuació AMI propera a 0 per a qualsevol valor de nombre de grups i d'elements. Els valors propers a 0 indiquen dues assignacions d'etiquetes bàsicament independents, mentre que eles valors propers a 1 indiquen un acord significatiu. A més, un AMI d'exactament 1 indica que les dues assignacions d'etiquetes són iguals (amb permutació o sense).

Per a la informació mútua normalitzada i la informació mútua ajustada, el valor normalitzador sol ésser alguna mitjana generalitzada de les entropies de cada agrupació. Hi ha diverses mitjanes generalitzades i no hi ha regles fixes per preferir-ne una per damunt de les altres. La decisió depèn en gran manera de cada camp d'aplicació.