Desenvolupament d'aplicacions de Dades massives

Resum II

ÍNDEX DE CONTINGUTS

Motor d'analítica de Spark 5
Tractament de dades a escala 5
Patró de processament de dades 5
Apache Spark 6
Plataforma unificada 6
Motor de computació 7
Spark App 8
Avantatges 9
Explosió de big data 10
Velocitat 10
Cost 10
Arquitectura Spark 12
Spark Core 12
Llibreries 13
Spark DataFrames 14
Execució API estructurada 15

Exemples de consultes 17
Spark MLIB 19
Sistemes recomanadors 19
Objectius 20
Ús de recomanadors 20
Procés de dades 21
Arquitectura 22
Tipus d'algoritmes 23
Tipus d'informació 24
Tècniques 24
Models 25
Rating sysytems 25
Challenges 26
Machine Learning amb Spark 27
Aprenentatge supervisat 29
Amb Spark 29
Training 30

Preparació del data set....31
Transformers....32
VectorAssembler....32
Estimadors....32
Avaluadors....33
Pipelines....34
Alternating least squares....34

MOTOR D'ANALÍTICA DE SPARK

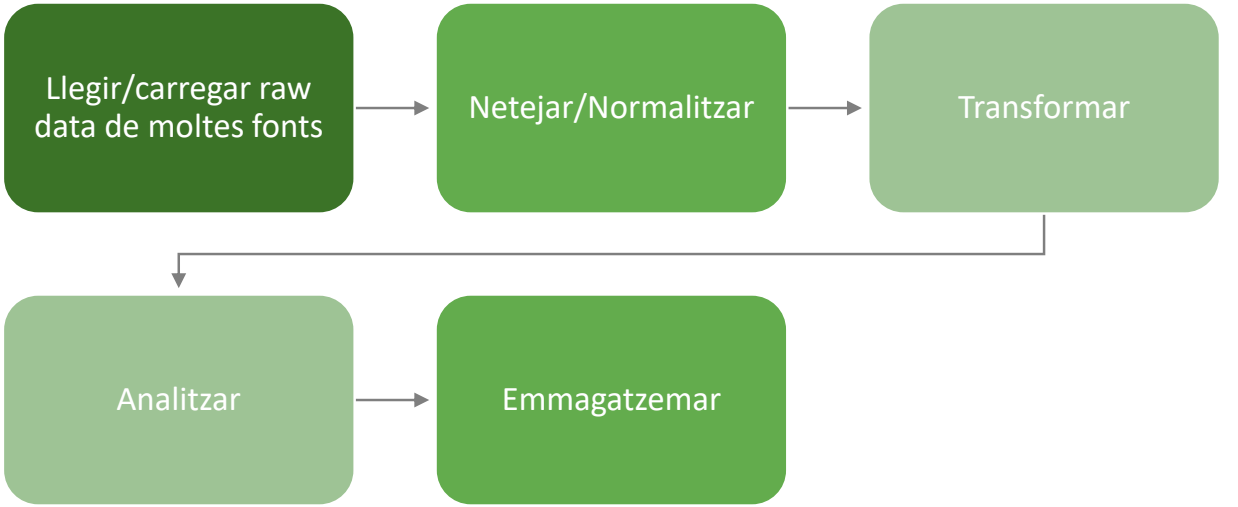
Tractament de dades a escala

Per processar dades a escala:

- Processar dades a escala: Terabytes.
- Utilitzar recursos comuns (locals o al núvol).
- Des d'un marc de tipus Pandas (abstraccions d'alt nivell).

Patró de processament de dades

El flux de dades és:

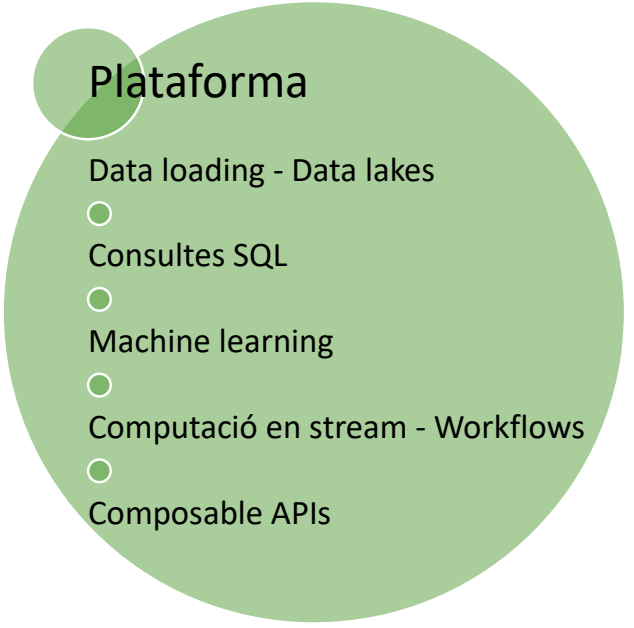


Apache Spark

spark.apache.org (3.0.1 i 2.4.7) és un motor analític unificat per processar una gran escala de dades. De manera que es fa processament paral·lel de dades en clústers d'ordinadors. El clúster més gran de Spark té més de 8000 màquines.

Plataforma unificada

Com hem dit, Spark és una plataforma unificada per escriure aplicacions de big data. Inclou:



Spark té App logic amb Spark APIs, Spark Application i Several runtime concepts.

Motor de computació

Spark gestiona la càrrega de dades i la combinació de sistemes d'emmagatzematge:

- Podem deixar les dades al cloud: Amazon S3.
- Magatzem clau-valor: Cassandra.
- Sistemes de fitxers distribuïts: HDFS. → Això permet processar grans volums de dades.
- Busos de missatges: Kafka.

Movem el càlcul a les dades, no les dades als nuclis informàtics. Així, dividim el programa en diversos sistemes (discos) com Apache YARN o Kubernetes, i s'executa de forma local. Cada sistema fa una part de l'execució.

És important el volum, no com es computa. I no fa falta enviar missatges.

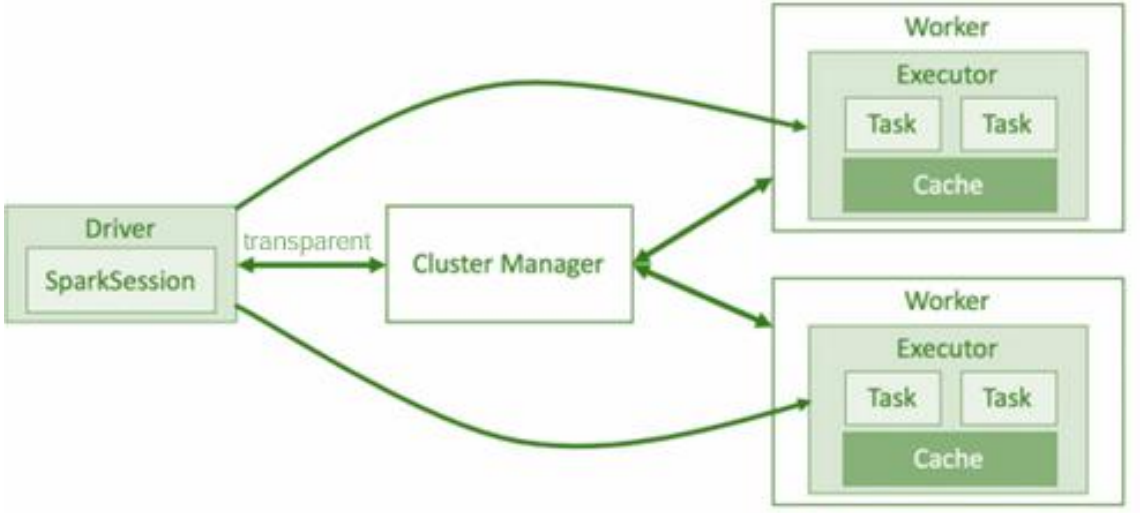
Spark App

Tenim un Driver process per Spark App, N execucions per Driver i 1 CPU core per Tasca.

Seguim una arquitectura master/worker:

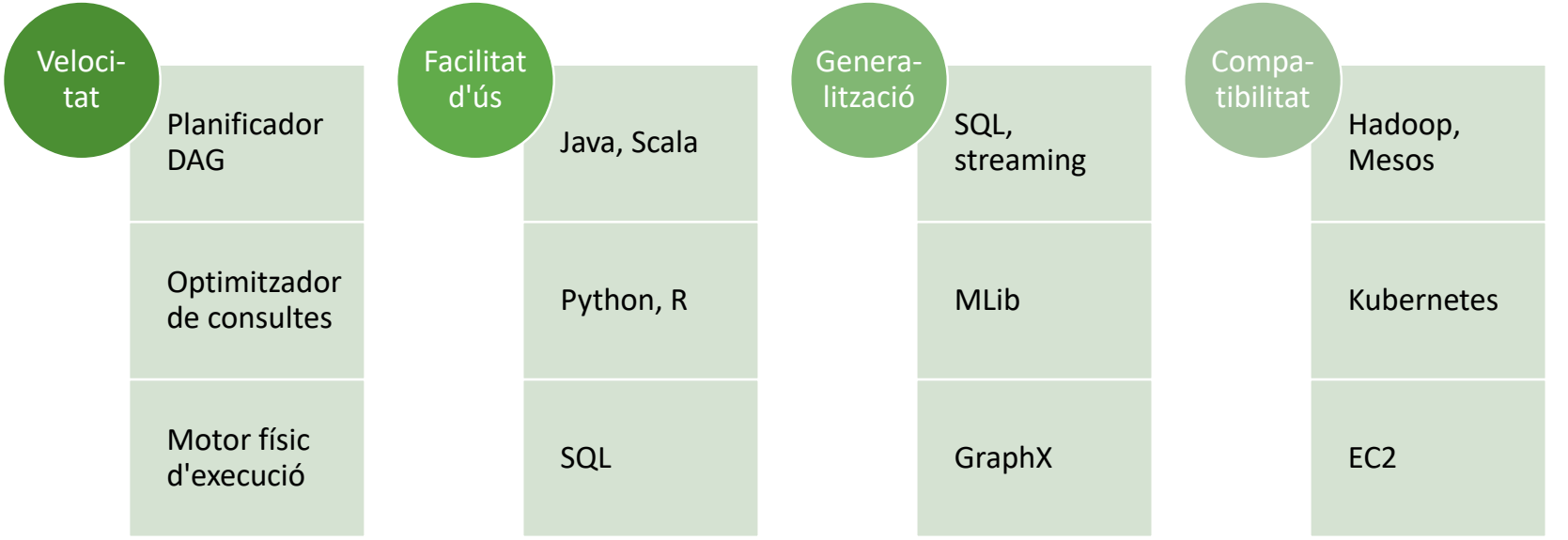
- Master: Driver process
- Worker: Executor process

Les dades es divideixen per mida depenent dels cores que tenim. Per exemple, tenim 8 cores, les dividim en 8 blocs.



Avantatges

Els avantatges d'utilitzar Spark són:



També és important el fet que dividim les tasques en cores. Altres avantatges són:

- Recollir dades és barat.
- Necessitem grans i paral·leles computacions.
- És difícil escalar solucions de programari grans i models tradicionals (SQL).

El 2009 es proposa treballar amb un clúster informàtica amb conjunts de treball del Projecte de recerca de la UC Berkeley.

Explosió de big data

Amb big data tenim velocitat i costos.

Velocitat

En un principi diem que 2xCPU → 2xEnergia. Per això volem programes que es paral·lelitzin sols. Les aplicacions han d'afegir paral·lelisme per funcionar més ràpid.

A partir del 2005: no hi ha CPU més ràpides, però sí un augment dels nuclis de CPU.

Cost

Cost d'emmagatzematge

Tenim una reducció de costos d'emmagatzematge. El cost d'emmagatzematge d'1 TB es redueix a la meitat cada 14 mesos.

Per dur a terme una execució paral·lela tenim els RDDs, Resilient Distributed Datasets:

- Estructures de dades paral·leles i tolerants a errors.
- Read-only, amb col·lecció partida per registres.
- Operem mitjançant transformacions (p. ex., "mapa") i accions (p. ex., "compte").
- Les dades resultants d'una consulta les afegim al final.

Cost d'adquisició d'emmagatzematge

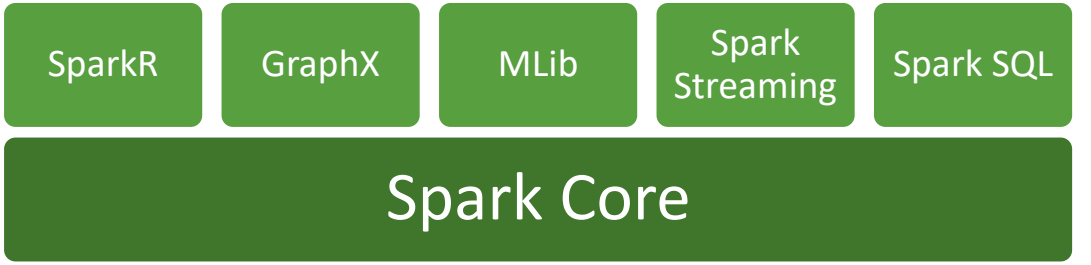
Experimentem una baixada de costos de la tecnologia de recollida de dades:

- La càmera web de 12 megapíxels costa menys de 5€.
- Seqüenciació del genoma menys de 1000€ per persona.

Arquitectura Spark

Pila unificada construïda a la part superior de Spark Core:

- Biblioteques separades orientades a càrregues de treball específiques de processament de dades.
- Les dades flueixen a través de les API sense necessitat d'emmagatzematge intermedi. No ens interessem de l'emmagatzematge.



Spark Core

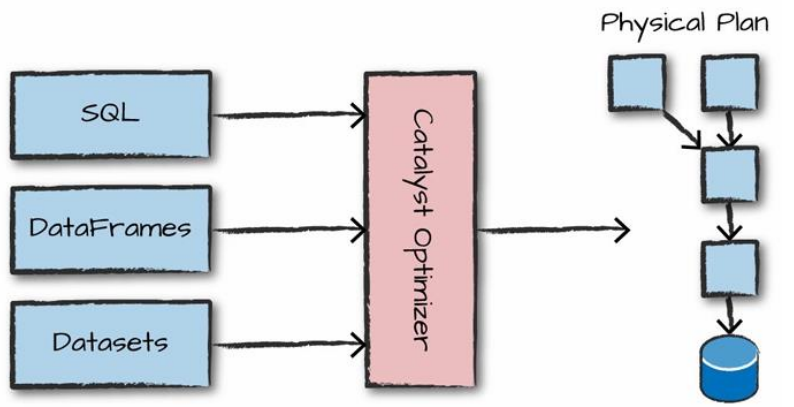
Les característiques del Spark Core són:

- **Tolerància de fallades (RDDs):** a part de dividir, podem dir que per cada bloc quantes rèpliques volem, evitant pèrdua de dades.
- **Computació in-memory:** cache() per guardar en memòria i persist() per guardar al disc.
- **Scheduling i monitorització:** per tenir les dades homogènies.
- **Interacció amb sistemes d'emmagatzematge:** hdfs, s3...

Llibreries

Algunes llibreries de Spark, a part del Spark Core engine, són:

- **Spark SQL:** Introdueix el DataFrame, una API d'alt nivell. Difumina la línia entre RDDs i taules relacionals. De manera que no tenim taules, fem joins.
    - Read/write JSON, CSV, Parquet, etc.
    - Optimitzador de cerques (*catalyst optimizer*)



- **MLiB (Machine learning):** Un cop entrenat el model, guardem els pesos. Es basa en l'API de DataFrame. Té més de 50 algoritmes.
    - Featurization, hyperparamenter tuning, model persistance.
- **Streaming:** Són streaming apps que permeten tolerància en errors.
- **Analítica de grafs:** Computació de grafs paral·lela, no viable amb Python.
- **SparkR:** Anàlisi de dades a gran escala utilitzant R.

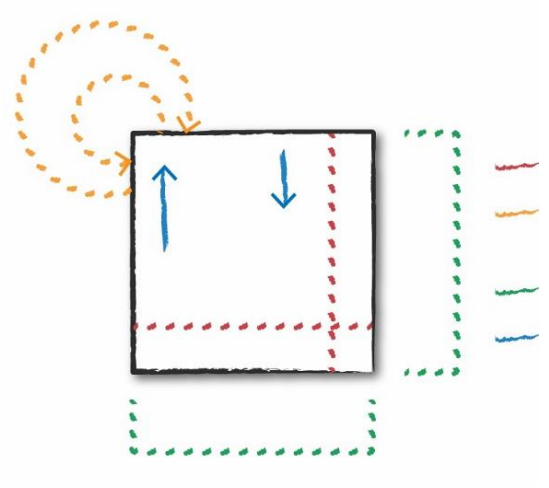
Spark DataFrames

Els Spark DataFrame són una llista de registres en format de files i columnes. Són blocs amb el mateix número de files i columnes.

- Col·leccions distribuïdes com a taules de files i columnes ben definides.
- Cada columna ha de tenir el mateix nombre de files.
- Cada columna té el mateix tipus de dades.

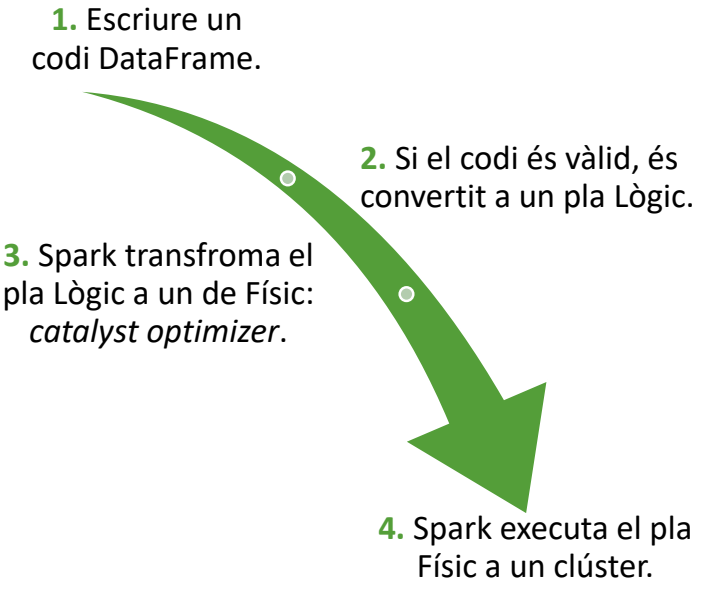
Cada vegada que volem modificar el DataFrame, Spark inicia una operació que planifica.

Acció sobre DataFrames: Spark planifica sobre com manipular files i columnes per calcular el resultat per a l'usuari. Els resultats de Spark generen un altre DataFrame. Algunes transformacions:

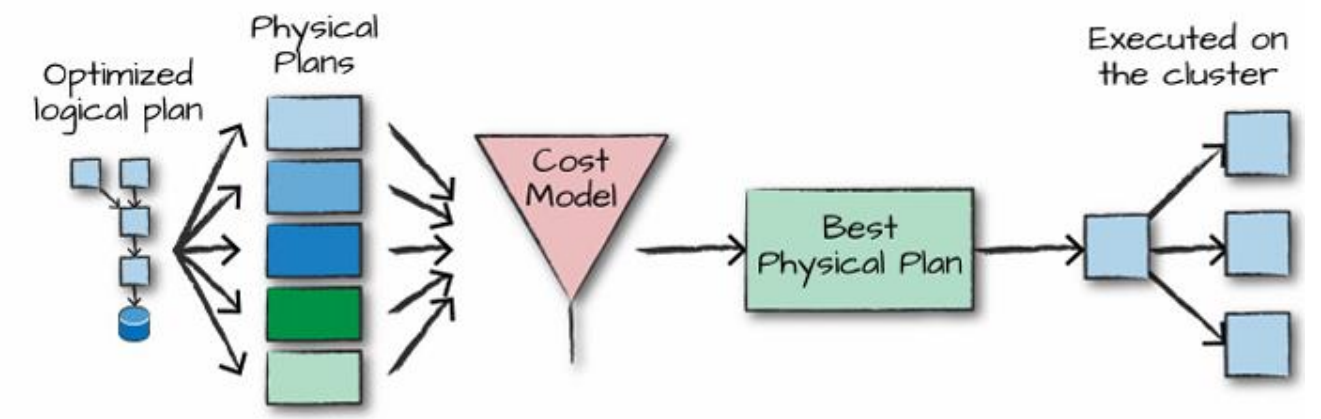


Eliminar columnes o files.
Transformar una fila en una columna o viceversa.
Afegir files o columnes.
Ordenar per un valor en les files.

Execució API estructurada



Es pot variar el model de cost, depenent del clúster amb que treballem.



Exemples de consultes

*Amb Spark **no tenim claus primàries**, podent fer tot amb tot.*

Seleccionar columnes:

```
c.select("customer","product").show
```

Seleccionar una columna i renombrar-la:

```
c.selectExpr("customer as customer_id").show
```

selectExpr ens permet crear nous dataframes, afegint una columna:

```
c.selectExpr("*", "(quantity = price) as equal_price")
```

Filtrar files

```
c.where(expr("quantity < 9")).where(expr("customer != 100")).show
```

Ordenar files

```
c.orderBy(desc("customer"),asc("price")).show
```

Agregacions

```
c.selectExpr("avg(price)","count(customer)")
```

- **Aggregate:** recopilar dades conjuntament de DataFrame.
- **Summarize:** dades numèriques per agrupació personalitzada.
- **Key** d'agrupament: columna que ens fixem.
- **Aggregation function:** com transformar els valors de les columnes.
    - Count, countDistinct, last, min, max, sum, sumDistinct, avg, approx_count_distinct (per velocitat, calculem un número aproximat).

Utilitzant **groupBy**, indicant una o més claus i una o més funcions.

```
c.groupBy("customer").count.show
```

```
c.groupBy("customer").agg(sum("quantity")).show
```

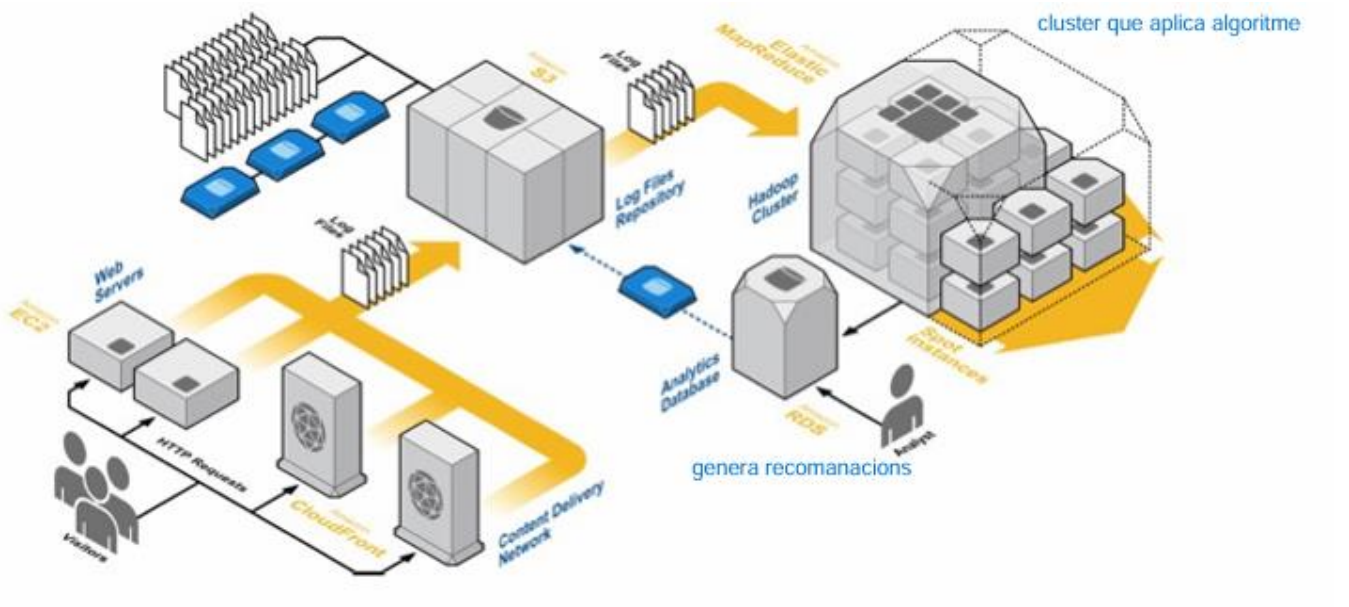
SPARK MLIB

Sistemes recomanadors

Predicció del comportament de l'usuari basada en operacions històriques. Els usuaris poden estar en un bucle de retroalimentació en el qual contribueixen als productes en ús.

Una bona manera d'adaptar-se bé en els usuaris és que els altres usuaris afectin a la decisió d'un usuari.

L'arquitectura cloud d'un sistema recomanador.




MALLORCA
CHAMPIONSHIPS
ATP 250
Mallorca
Illes Balears
R AL MALLORCA CHAMPIONSHIPS
CUENTA COMO ACTIVIDAD
EXTRACURRICULAR, ¿NO?
21-28 Jun 2025
mallorcachampionships.com

Per entendre el comportament de les dades, s'ha de tenir un coneixement del producte i el seu context. És a dir, per interpretar-les hem de saber què passa en el mon.

El punt fort de Google és el seu **poder predictiu**, pot mesurar les tendències globals abans que qualsevol altre mitjà de comunicació ho sàpiga.

Amb l'aparició del chatGPT, les prediccions són difícils perquè ara les dades no només són de font humana.

Objectius

Ajudar els usuaris a descobrir informació rellevant per a ells. Cal aplicar mètodes per introduir esdeveniments:

- ítem per vendre
- acció a fer
- notificació a llegir

Ús de recomanadors

S'utilitzen els recomanadors comunament:

- **E-commerce:** productes i usuaris
- **Campanyes de marketing:** anàlisi de cistella buida i campanyes d'e-mails.
- **Buscador de contingut:** spotify, netflix, ...
- **Predicció:** cyber defensa

Procés de dades

Elements

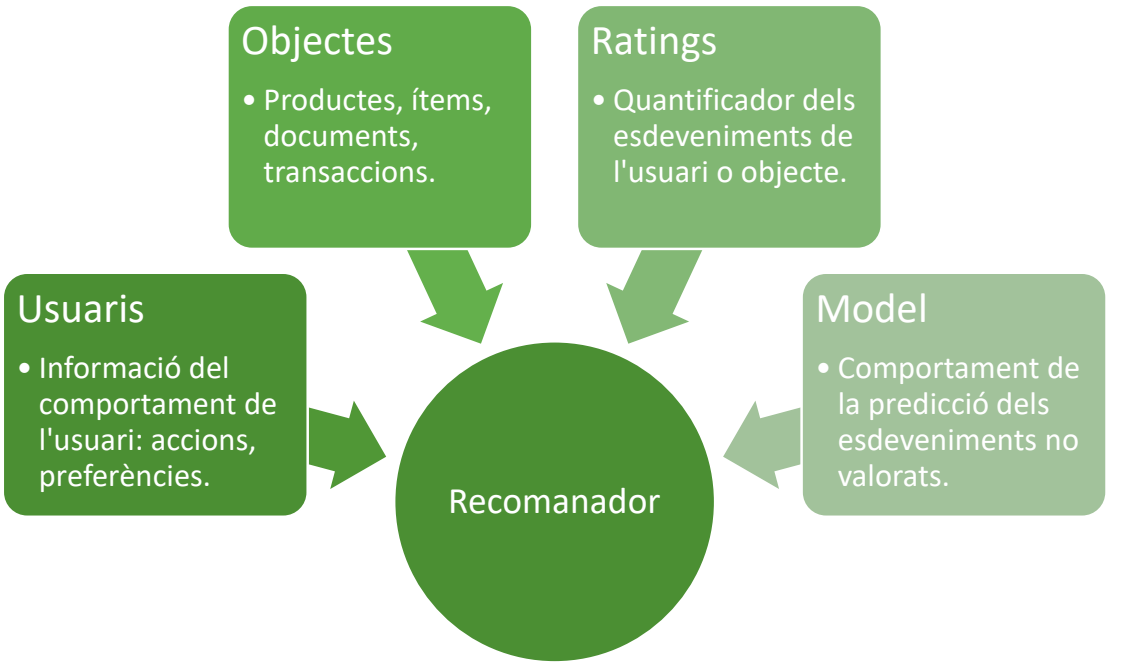
- Machine Learning algorithms
- ML models
- Results of applying model to a data set

Procés

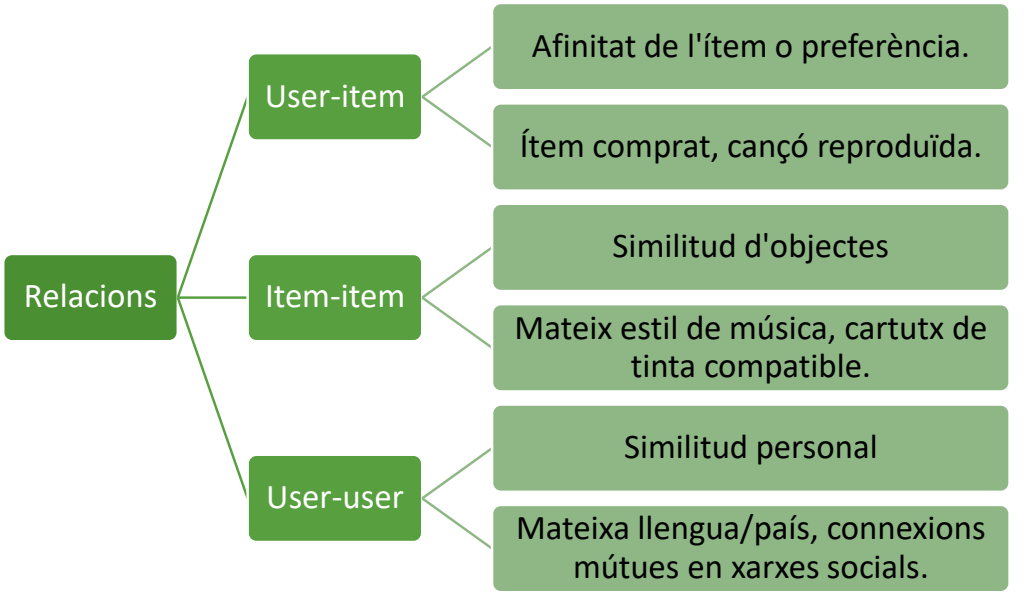
- data storage
- data processing
- data science
- data engineering

Arquitectura

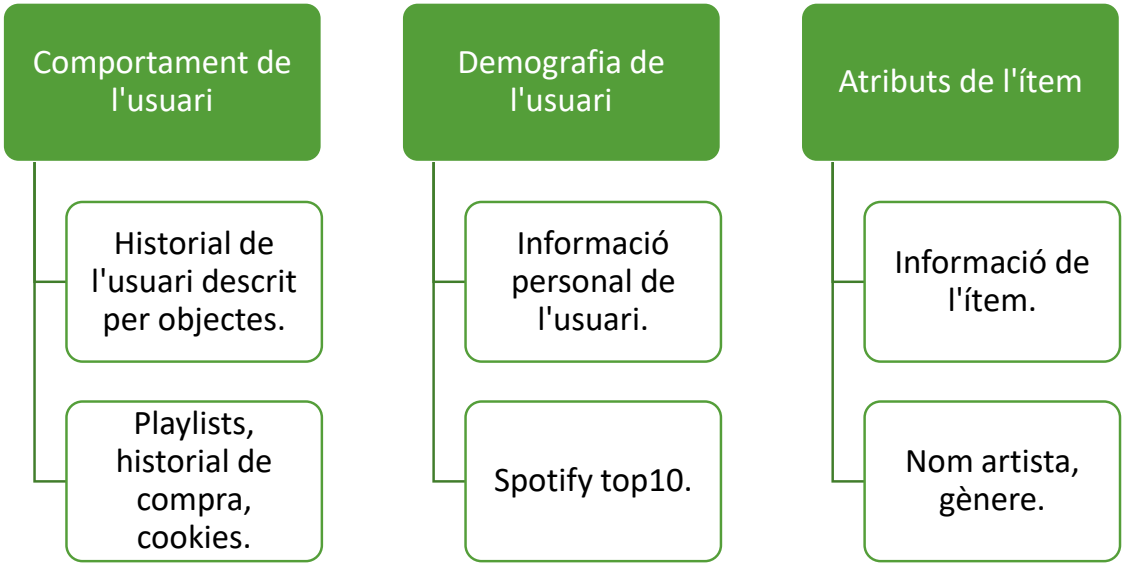
En l'arquitectura d'un sistema recomanador participen:



Tipus d’algoritmes



Tipus d'informació



Tècniques

Les tècniques més comunes són:

- **Collaborative filtering:** Interaccions passades defineixen el futur. Es genera una matriu d'interacció.
- **Content-based filtering:** Utilitza característiques d'elements per seleccionar i retornar elements rellevants per a la consulta d'un usuari. Aquest mètode sovint té en compte les característiques d'altres elements en què un usuari expressa interès. I de vegades es té en compte els elements d'acord amb les característiques descriptives (per exemple, metadades) adjunts als elements en lloc del contingut real d'un element. Productes, documents, accions...

Models

Tenim dos tipus de models recomanadors en collaborative filtering:

- **Basat en memòria:** Basat en interaccions passades, utilitzem els últims registres de l'històric i no tot.
    - Assumeix cap model, normalment mira els veïns més propers.
      "Clients que han vist això també s'ha fixat en"
- **Basat en model:** Basat en el model generatiu subjacent (underlying). Tenim una probabilitat associada.
    - El model explica les interaccions **user-item**.

Rating systems

Tenint una entrada, podem valorar:

- **Explícitament:** directament des de l'usuari.
    - Estrelles, comentaris, feedback, etc. Es vol que els ratings com més millors perquè surtin primer com a opció en l'app Store.
- **Implícitament:** derivat de les interaccions. Hi ha informació que pren un pes que no hauria de ser, s'ha de gestionar. Per exemple, si es comença a escoltar una cançó i no s'acaba.
    - Clicks, visualitzacions, compres.

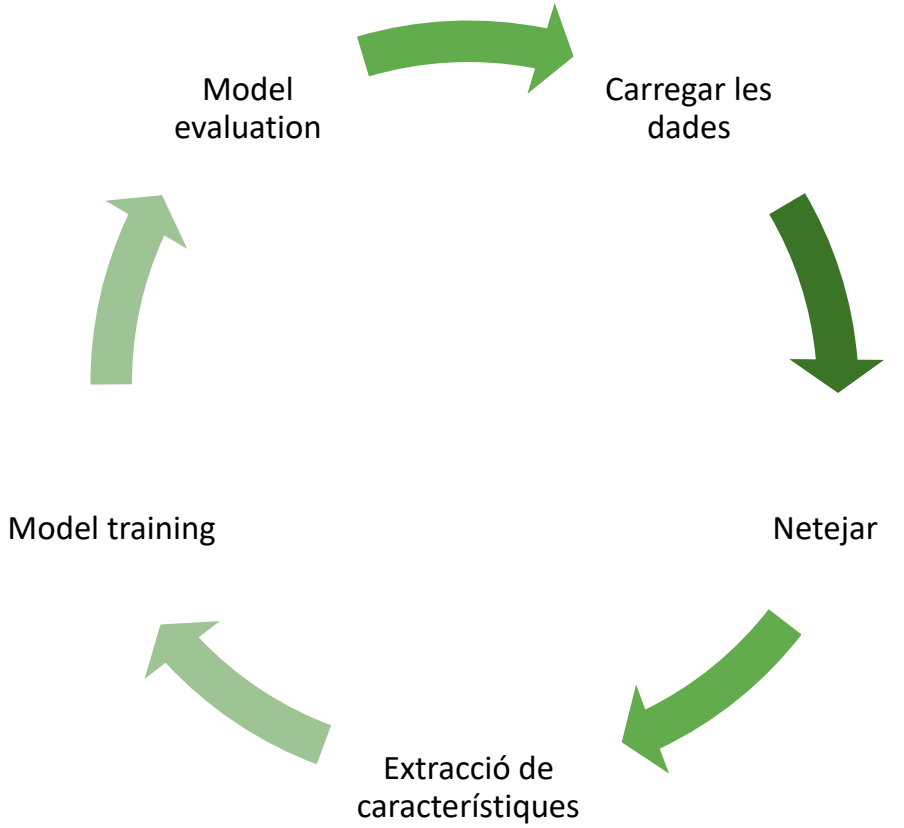
Challenges

Els desafiaments dels recomanadors són:

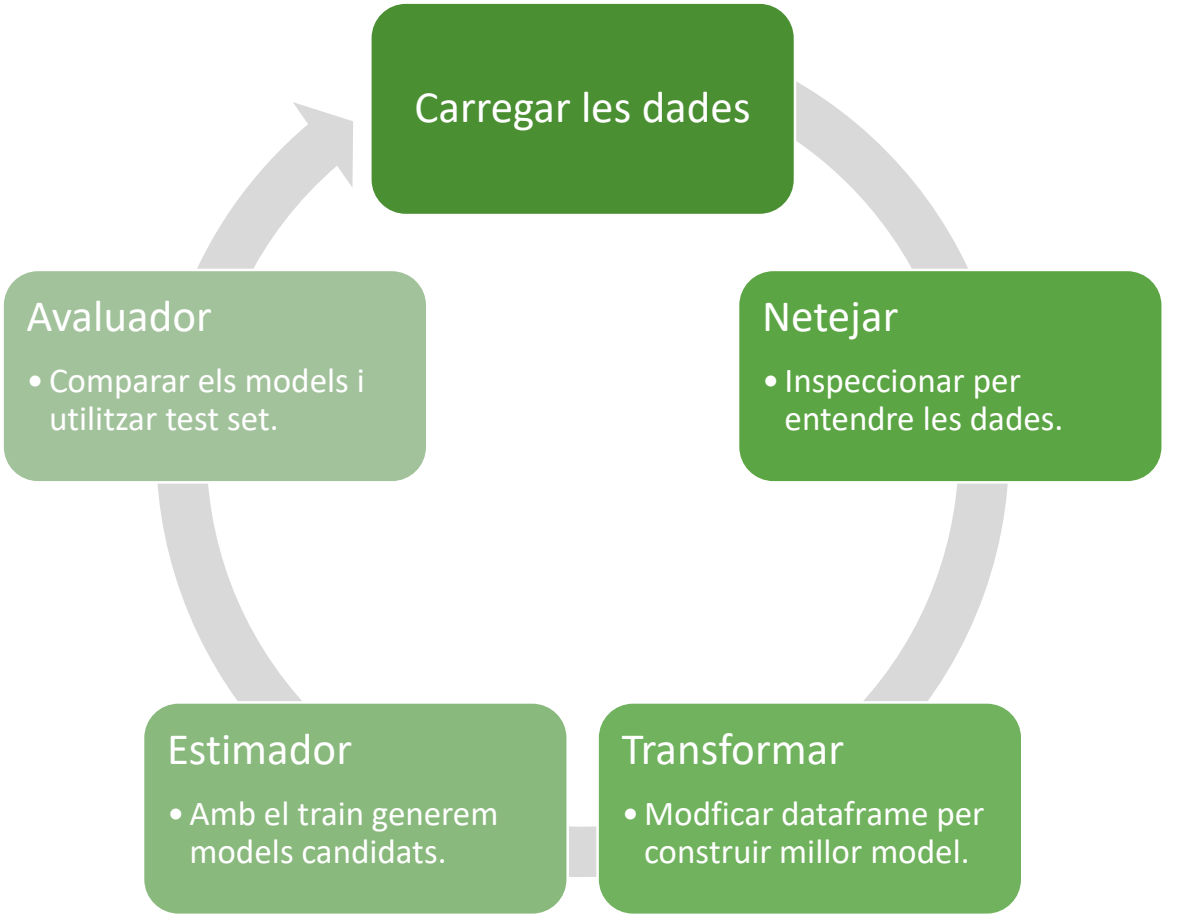
1. Els ratings explícits poden ser difícils de processar.
2. Saber com començar amb pocs usuaris.
    a. Pandora: quan no tenim gaires usuaris fem content-based.
    b. Spotify: collaborative filtering.
3. Escalabilitat. Fer una matriu és difícil de gestionar, volem que sigui ràpid. Per això, utilitzem Spark.

Machine Learning amb Spark

MLib és un sistema distribuït, amb paral·lelisme de dades. No ens preocupem per la infraestructura. En **ML** tenim els passos:



En **Spark MLIB** tenim:



Aprenentatge supervisat

Conjunt de registres d'entrada, cadascun dels quals té etiquetes associades.

Objectiu: predir les etiquetes de sortida a partir d'una entrada nova sense etiquetar.

Les etiquetes de sortida poden ser discretes o contínues.

Dos tipus:

- **Classificació:** Entrenar un algorisme per predir una variable dependent que sigui categòrica (pertanyent a un conjunt de valors discret i finit). El model de classificació farà una predicció que un determinat ítem pertany a un dels grups.
    - *Exemple:* classificació de correu brossa. Ús d'un conjunt de correus electrònics històrics organitzats en grups de correu brossa i no en correus brossa.
- **Regressió:** El valor a predir és un nombre continu. Prediu els valors que el nostre model no ha vist durant l'entrenament.

Amb Spark

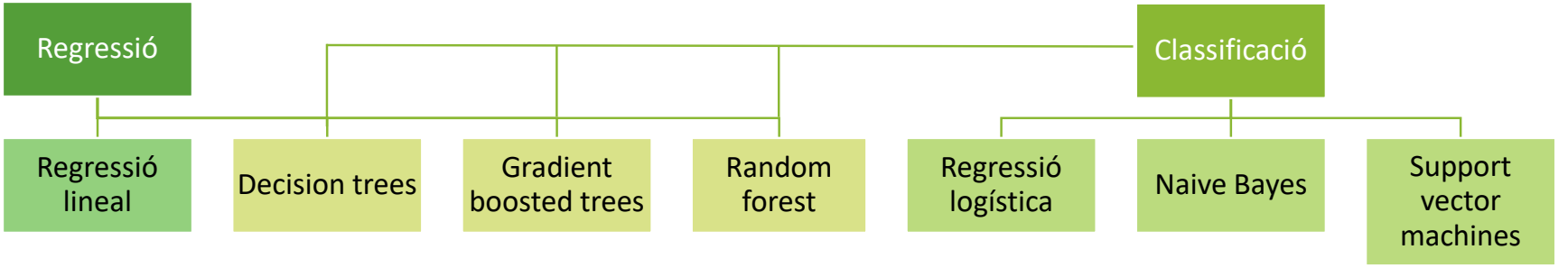
Dades històriques amb etiquetes: variables dependents.

Cal formar un model, predir els valors d'aquestes etiquetes. Ens basem en diverses característiques dels punts de dades.

Exemple
Predir els ingressos d'una persona en funció de l'edat.

- Variable dependent: ingressos
- Característica: edat

Algoritmes



Training

El procés d'entrenament del model acostuma a procedir a través d'un algorisme d'optimització iteratiu com ara el descens del gradient. L'algorisme d'entrenament comença amb un model bàsic i millora gradualment ajustant diversos paràmetres interns (coeficients) durant cada iteració d'entrenament.

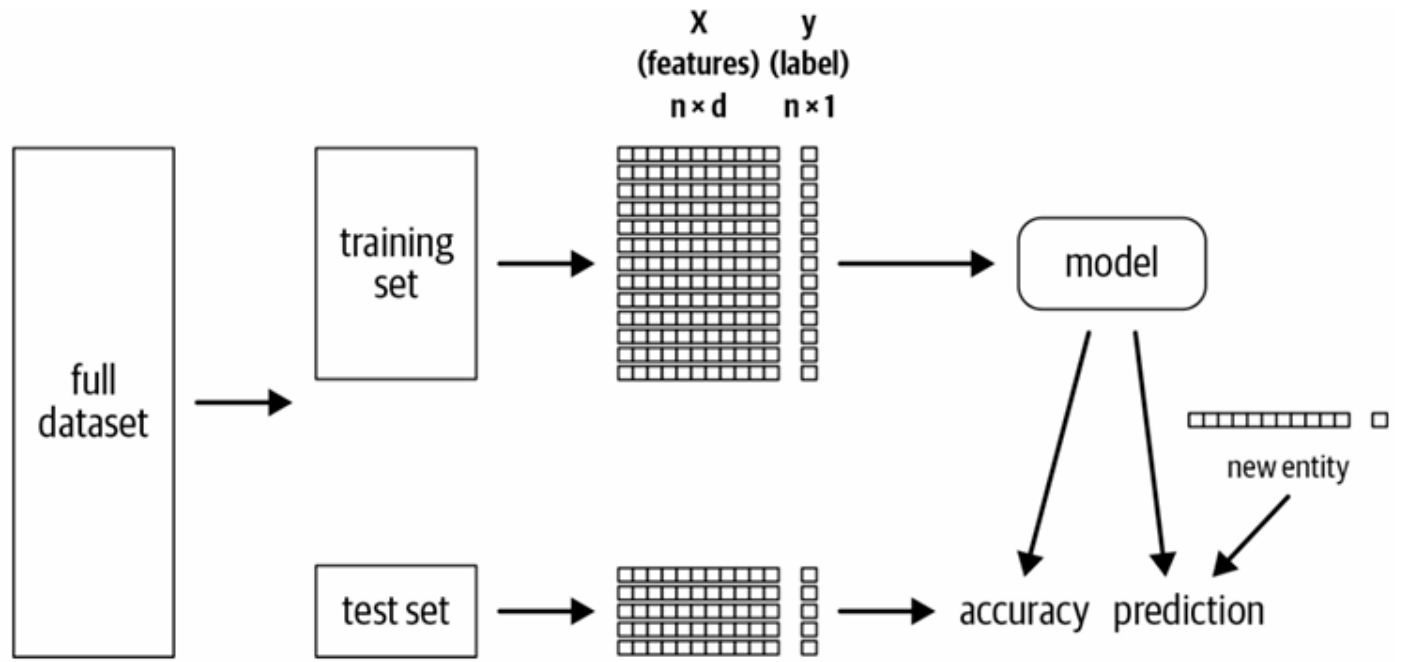
El resultat d'aquest procés és un model entrenat que podem utilitzar per fer prediccions sobre dades noves. De manera que mesurarem l'èxit dels models entrenats: mètriques.

Els pipelines d'aprenentatge ent-to-end han de definir el processament mitjançant transformers, estimadors, avaluadors i pipelines.

Preparació del data set

Podem fer `randomSplit`. Així, dividim el data set en dos grups, normalment 80% train i 20% test. Si construïm un model sobre tot el conjunt de dades, és possible que el model memoritzi o "sobreajusti" les dades d'entrenament que proporcionem.

El rendiment del model al conjunt de proves és un indicador del rendiment que tindrà en dades no vistes, suposant que les dades segueixen distribucions similars.



Transformers

Els transformadors a Spark accepten un DataFrame com a entrada i retornen un nou DataFrame amb normalment una columna afegida.

No aprenen de les nostres dades, però apliquen transformacions basades en regles mitjançant el mètode transform().

S'utilitza en preprocessament i feature engineering.

Exemple: convertir variables categòriques de cadena en valors numèrics.

VectorAssembler

Concatena totes les característiques en un gran vector.

```
(trainDF, testDF) = indexedDF.randomSplit([0.8,0.2],seed=1)
vecAssembler= VectorAssembler(inputCols=trainDF.columns, outputCol = "features")
vecTrainDF = vecAssembler.transform(trainDF)
```

Estimadors

Els estimadors aprenen paràmetres a partir de les nostres dades. Agafa un DataFrame i retorna un Model. Això requereix dues passades sobre les nostres dades: la passada inicial genera els valors d'inicialització i la segona realment aplica la funció generada sobre les dades.

A la nomenclatura de Spark: els algorismes que permeten als usuaris entrenar un model a partir de dades també s'anomenen **estimadors**.

```
lr = ( LinearRegression( featuresCol="features", labelCol="SalePrice", maxIter=10,
                         regParam=0.8, elasticNetParam=0.1, ) )

lrModel = lr.fit(trainDF)

predDF = lrModel.transform(testDF)
```

fit() retorna un transformador i amb el nou transformador apliquem els paràmetres del model a les noves dades per generar prediccions.

Avaluadors

Un avaluador ens permet veure com funciona un model determinat segons els criteris que especifiquem. Utilitzem un avaluador per comprovar la qualitat del model. Aleshores, decidim utilitzar aquest model per fer prediccions.

Avaluadors típics: R2, RMSE (Root Mean Squared Error).

```
lrEvaluator = ( RegressionEvaluator( predictionCol="prediction",
                labelCol="SalePrice", metricName="r2",) )

r2 = lrEvaluator.evaluate(testDF)
```

Pipelines

Podem especificar cadascuna de les transformacions, estimacions i avaluacions una per una. No obstant, és més fàcil especificar els nostres passos com a etapes en un pipeline.

Aquest pipeline és similar al concepte de pipeline de scikit-learn. Si sabem que la fase d'anàlisi està bé creem un pipeline.

```
Pipeline = Pipeline(stages=[vecAssembler, lr])

pipelineModel=pipeline.fit(trainDF)

predDF=pipelineModel.transform(validationDF)
```

Alternating least squares

Tenint una matriu d'opinions d'usuaris.

	Movie 539	Movie 281	Movie 347
Jordi	1	?	4
Mariona	1	3	5

$$= \begin{bmatrix} U_1 \\ U_2 \end{bmatrix} * \begin{bmatrix} P_1 & P_2 & P_3 \end{bmatrix}$$

L'objectiu de l'alternança de mínims quadrats és trobar dues matrius: U i P.

U * P és aproximadament igual a la matriu original d'usuaris i productes.

Podem predir què pensarà en Jordi de la pel·lícula 281 multiplicant la fila 1 de U per la columna 2 de P.

Llavors, ALS és un algoritme **user-to-user**. És un tipus d'estimador que pren un DataFrame i retorna un Model.

```
als = ALS(userCol="userId", itemCol="movieId", ratingCol="rating")
model = als.fit(trainDF)
```

I per obtenir les prediccions de qualsevol usuari.

```
predDF = model.transform(validationDF)
usersRec = model.recommendForAllUsers(5)
userPredictedRatings = usersRec.first().userId
```

```
+------+-------+----------+
|userId|movieId|prediction|
+------+-------+----------+
|   471|    281| 3.0085273|
|   471|   2366| 3.1742785|
+------+-------+----------+
```