1. Spark

Processa dades a escala:

- Terabytes
- Utilitza recursos comuns (local o cloud)
- Des d'un marc de tipus Pandas (abstraccions d'alt nivell)

Patró comú de processament de dades:

- Llegir/agregar dades en brut des de moltes fonts.
- Netejar/normalitzar.
- Transformar.
- Analitzar.
- Emmagatzemar**.**

Apache Spark:

- Motor d'anàlisi unificat per al processament de dades a gran escala.
- Tractament paral·lel de dades en clústers d'ordinadors.

Spark com a plataforma unificada:

Plataforma unificada per escriure aplicacions de big data.

- Càrrega de dades.
- Consultes SQL.
- Machine learning.
- Càlcul de streaming.
- API componibles.

Motor de computació Spark:

- Mou el càlcul a les dades, no les dades als nuclis informàtics.
- Spark gestiona la càrrega de dades i la combinació de sistemes d'emmagatzematge.
- Exemples de com interactua Spark:
    - Pot connectar sistemes d'emmagatzematge al núvol com Amazon S3.
    - Utilitzar bases de dades clau-valor com Cassandra.
    - Utilitzar sistemes de fitxers distribuïts com HDFS.
    - Produir dades utilitzant busos de missatges com Kafka.

Per què Spark?

- *Velocitat*: planificador DAG, optimitzador de consultes, motor d'execució física.
- *Facilitat d'ús*: Java, Scala, Python, R, SQL.
- *Generalitat*: SQL, streaming, MLib, GraphX.
- *S'executa a tot arreu*: Hadoop, Mesos, Kubernetes, EC2.

Característiques que la Big Data ha de complir:

- *Velocitat:* les aplicacions han d'afegir paral·lelisme per funcionar més ràpid.
- *Cost:* reducció de costos d'emmagatzematge i de la tecnologia de recollida de dades

Per què és Spark popular ara?

- La recollida de dades és econòmica.
- Necessitat de càlculs grans i paral·lels.
- Difícil d'escalar solucions de programari grans i models tradicionals (SQL).

Apache Spark: conceptes bàsics.

- Conjunts de dades distribuïts resilients: RDD
    - Estructures de dades paral·leles i tolerants a errors.
    - Col·lecció de registres particionats només de lectura.
    - Operat mitjançant transformacions (p. ex., "map") i accions (p. ex., "count").
- Spark és un sistema distribuït per processar volums molt grans de dades.
- El clúster Spark més gran del món té més de 8.000 màquines.
- Sistemes de gestió de recursos: conceptes mestre i treballador.
    - Apache YARN, Apache Mesos, Kubernetes.
- Funcionament per aplicació Spark: 1 procés controlador, N executors per controlador i 1 nucli de CPU per tasca.
- Arquitectura Master/Worker
    - Driver/Master → Procés del controlador, gestiona l'execució de Spark. S'encarrega d'enviar les tasks als executors pel seu processament.
    - Executor/Worker → Procés executor, són els workers que realment executen les tasks. Cada executor rep les tasks del Driver, llegeix les particions de dades que li toquen, i les processa utilitzant els cores de CPU disponibles.

Spark: és un sistema de processament distribuït utilitzat per a càrregues de treball de big data. Utilitza una execució de consultes optimitzada per a consultes analítiques ràpides amb dades de qualsevol mida.

Possibles preguntes:

Quina és la relació entre CPU core i Task?

1 CPU core per Task. En Spark, una Task és la unitat bàsica de treball i s'executa en un únic core de CPU. Llavors, la quantitat de CPU cores disponibles en un clúster determina la quantitat de tasks que es poden executar simultàniament. Per exemple, si tens un clúster amb 8 cores de CPU, pots executar fins a 8 tasks en paral·lel.

Quina és la relació entre Task i la partició de dades?

1 Task per partició de Dades. Cada Task s'encarrega de processar una partició de dades específica. Quan es distribueixen les dades, Spark les divideix en particions, i cada partició és assignada a una task per ser processada. Això implica que el nombre total de tasks en una aplicació Spark serà igual al nombre de particions de dades.

Com es distribueixen les dades?

Les dades es divideixen en particions, les quals són assignades a task. Cada task té una partició específica. Aquestes tasks són dividides en CPU cores, quantes més CPU cores i particions hi hagi, més tasks es poden executar simultàniament, el que generalment condueix a un processament més ràpid de les dades.

Relació nombre de tasks, CPUs cores i particions de dades

El Nombre de tasks ≠ Nombre de CPU cores ≠ Nombre de particions de dades.

Exemple pràctic:

Si hi ha un clúster amb 8 CPU cores i les dades estan dividides en 100 particions:

- Es crearan 100 tasks per processar les 100 particions.
- Només 8 tasks es poden executar simultàniament perquè només hi ha 8 cores disponibles.
- Les altres 92 tasks esperaran en una cua fins que un core estigui disponible per a la seva execució.

L'arquitectura de Spark

General:

- Pila unificada construïda a sobre 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.

Spark Core:

- Tolerància a errors (és a dir, RDD).
- Càlcul a la memòria: cache() i persist().
- Programació i supervisió.
- Interacció amb sistemes d'emmagatzematge (hdfs, s3, etc.).

Spark SQL:

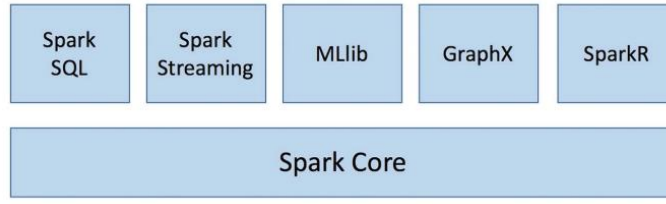
- Presenta l'API d'alt nivell DataFrame.
- Desdibuixa la línia entre RDD i taules relacionals.
- Lectura/escriptura JSON, CSV, Parquet, etc.
- Optimitzador de catalitzadors.

MLlib:

- Basat en l'API DataFrame (des de la versió 2.0)
- +50 algorismes comuns de ML predefinits
- "Featurització" (funcions Spark), ajustament d'hiperparàmetres, persistència del model.

Spark Streaming, GraphX, SparkR:

- Aplicacions de transmissió tolerants a errors.
- Càlcul gràfic paral·lel (no disponible a Python).
- Anàlisi de dades a gran escala mitjançant R.



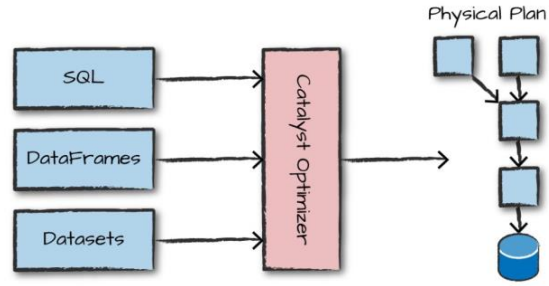
Spark DataFrames

- 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.
- Acció sobre DataFrames: Spark planeja sobre com manipular files i columnes per calcular el resultat per a l'usuari.

Execució d'API estructurada

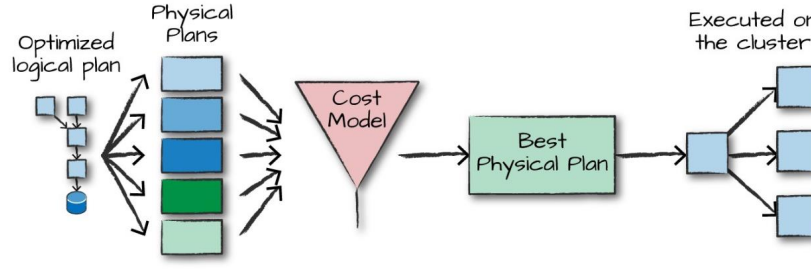
1. Escriu el codi DataFrame, un DataFrame és una llista de registres de tipus Fila i un nombre de columnes.
2. Si el codi és vàlid, es converteix en un pla lògic.
3. Spark transforma el pla lògic en un pla físic: Catalyst optimizer.
4. Spark executa el pla físic al clúster.

Optimitzador de catalitzadors

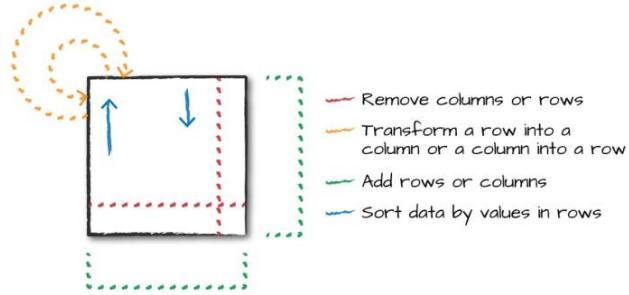


- És una eina potent que s'utilitza per optimitzar el rendiment de les consultes a Spark.
- L'objectiu principal del Catalyst Optimizer és transformar i optimitzar les operacions SQL o DataFrame de l'usuari en un pla d'execució física eficient.

Planificació física: Pla Spark



Transformacions del DataFrame



Exemples de consultes principals

- select: per manipular columnes en el DataFrame.
- selectExpr: és la manera més flexible de descriure una columna o una manipulació d'string d'una columna.

  Es poden crear nous DataFrames.

  És una manera simple de construir expressions complexes.

  Serveix per fer agregacions en tot el DataFrame.
- where: crea una expressió que es valora com a True o False. Serveix per filtrar files amb una expressió igual a False.
- orderBy: per tenir els valors més grans o més petits a la part superior de DataFrame, per defecte és ordena en ordre ascendent.

Agregacions

- Agregat: recull dades conjuntes de DataFrame.
- Resumir dades numèriques per agrupació personalitzada.
- clau d'agrupació: columna en què centrar-se.
- funció d'agregació: com transformar els valors de les columnes.
- groupBy: una o més tecles i una o més funcions.
- opcions més avançades: window, rollup, cube, ...
- Funcions d'agregació:

  count, countDistinct, approx_count_distinct, last, min/max, sum, sumDistinct, avg, var_pop/stddev_pop

Agrupació amb expressions

- Definiu l'expressió d'agregació: "sum(quantity)".
- Utilitzar agg per aplicar la funció a la clau d'interès.

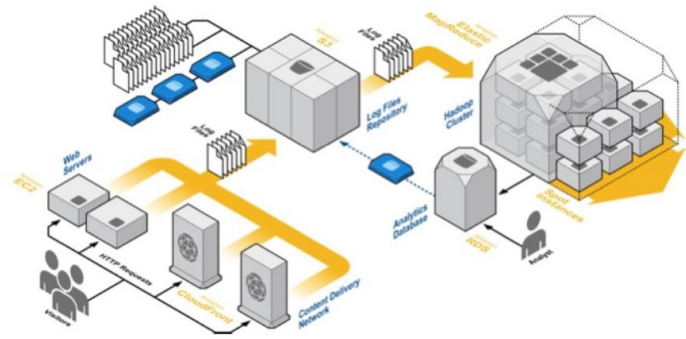
2. Spark MLib

Sistemes de recomanació i pipelines ML

Sistemes de recomanació

- Predicció del comportament de l'usuari basada en operacions històriques.
- Si t'agrada això, també t'agradarà: ...

Arquitectura cloud del sistema de recomanació



Anàlisi de dades de cerques

- L'equip de recerca analitza cerques populars.
- Les cerques poden mostrar patrons repetitius i pics en funció d'esdeveniments específics o contextos culturals.
- Per interpretar aquestes dades i conèixer el seu significat, s'ha de conèixer el context mundial o esdeveniments rellevants que puguin influir en els patrons de cerca.

Exemple d'anàlisi de dades de cerques:

El 22 d'abril del 2001, l'equip de recerca de Google va observar una tendència notable en les cerques relacionades amb el nom de naixement de Carol Brady, un personatge de la sèrie de televisió dels anys 70 "The Brady Bunch". Aquestes cerques es van agrupar en cinc pics distints, tots començant 48 minuts després de l'hora, amb una distribució específica: el primer pic va ser el més gran, seguit de dos pics petits, un quart pic gran i finalment un altre pic petit després d'un temps.

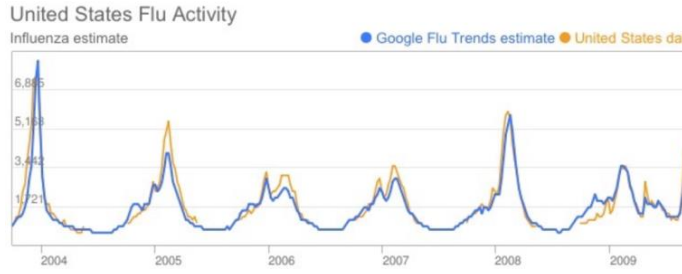
Per comprendre el perquè de la cerca d'aquesta dada, s'ha de considerar els esdeveniments externs que van influir en aquesta activitat de recerca. El 21 d'abril del 2001, en el programa de televisió "Who Wants to Be a Millionaire?", la pregunta per guanyar el milió de dòlars va ser sobre el nom de naixement de Carol Brady. Quan el presentador va formular la pregunta, milers de persones van fer

cerques a Google per trobar la resposta. Això va causar quatre pics de cerques successives a mesura que el programa es va emetre en diferents zones horàries dels Estats Units.

Resumint, les cerques sobre Carol Brady el 22 d'abril del 2001 es van produir a causa d'una pregunta crucial en un popular concurs televisiu, i els pics corresponents reflectien l'emissió del programa a través de les diverses zones horàries.

Prediccions de Google

- El seu punt fort és el seu poder predictiu, pot mesurar les tendències globals abans que qualsevol altre mitjà de comunicació ho sàpiga.
- Exemple de predicció de la grip de Google entre 2003 i 2008.



Recomanacions com a processos de dades

Elements:

- Algorismes d'aprenentatge automàtic
- Models ML
- Resultats de l'aplicació del model a un conjunt de dades

Procés:

- Emmagatzematge de dades
- Tractament de dades
- Ciència de dades
- Enginyeria de dades

Ús comú dels recomanadors

- Comerç electrònic: productes i usuaris.
- Campanyes de màrqueting.
- Navegació de continguts: spotify, netflix.
- Predicció: ciberdefensa.

Objectius dels recomanadors:

- Ajudar els usuaris a descobrir informació rellevant per a ells.
- S'ha d'aplicar mètodes per introduir esdeveniments: article per vendre, acció a fer, notificació per llegir.

Arquitectura del sistema de recomanació

- Usuaris: informació sobre el comportament dels usuaris (accions, preferències).
- Objectes: productes, articles, documents, transaccions.
- Valoracions: quantificació d'esdeveniments d'usuari/objecte.
- Model: predicció del comportament a partir d'esdeveniments d'usuari/objecte no valorat.

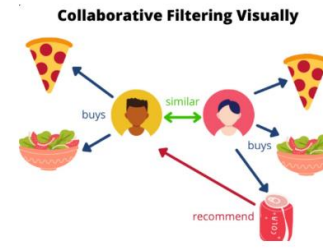
Tipus d'algoritmes

- Usuari – Item: afinitat o preferència d'element (Item comprat o cançó reproduïda).
- Item – Item: semblança d'objectes (cançó del mateix estil o cartutx de tinta compatible).
- Usuari – Usuari: semblança personal (mateix idoma /país o connexions mútues a les xarxes socials).

Tipus d'informació per a sistemes de recomanació

- Comportament de l'usuari: historial d'usuari descrit amb objectes (Llistes de reproducció, historial de compres, cookies).
- Dades demogràfiques dels usuaris: informació personal de l'usuari (Spotify top 10 més popular avui).
- Atributs de l'ítem: informació de l'element (nom de l'artista, gènere).

Tècniques més habituals



- Filtratge col·laboratiu: les interaccions passades defineixen el futur mitjançant la matriu d'interacció usuari-element.
  Exemple: si a l'usuari A i a l'usuari B els agraden els mateixos llibres, i l'usuari A llegeix un llibre nou, es recomanarà aquest llibre a l'usuari B.
- Filtratge basat en contingut: recomana elements analitzant les característiques dels elements que l'usuari ha preferit en el passat.
  Exemple: si a un usuari li agraden les pel·lícules de ciència-ficció, se li recomanaran altres pel·lícules del mateix gènere.

Models de recomanació

- Memòria: basat en interaccions passades, no assumeix cap model. Generalment cerca de veïns més propers.

  Exemple: Els clients que van veure això també van veure allò.
- Model: basat en el model generatiu subjacent. Explica les interaccions usuari-item.

Sistemes de classificació d'usuari/objecte

- Explícit: directament de l'usuari. Són valoracions amb estrelles, ressenyes, comentaris, etc. Com ara Netflix, Instagram i Spotify.
- Implícit: derivat de les interaccions, com ara clics, visualització i compres.

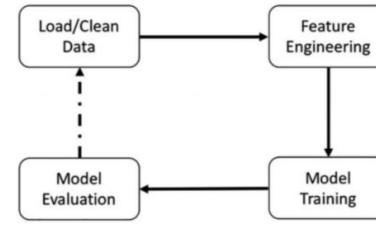
Reptes dels recomanadors

- Les valoracions explícites poden ser difícils de processar.
- Problema d'arrencada en fred: com es comença amb pocs usuaris?
  - pandora (basat en contingut) vs spotify (filtratge col·laboratiu)
- Escalabilitat

Pipelines ML

Passos principals del processament de dades de ML

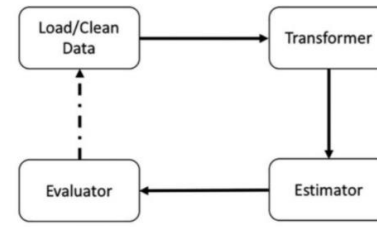
- Carregar dades.
- Neteja les dades.
- Enginyeria de característiques.
- Entrenament del model.
- Avaluació del model.



ML Main Steps

Conceptes principals d'Apache Spark MLiB

- Carregar dades.
- Neteja les dades.
- Transformador.
- Estimador.
- Avaluador.



MLlib Pipeline Main Concepts

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 i regressió.

Aprenentatge supervisat amb Spark

- Dades històriques amb etiquetes: variables dependents.
- Ha d'entrenar un model.
- Predir els valors d'aquestes etiquetes.
- Basat en diverses característiques dels punts de dades.
- Exemple: predir els ingressos d'una persona en funció de l'edat.
    - Variable dependent: ingressos.
    - Característica: edat.

Entrenament d'un model ML

- El procés d'entrenament del model acostuma a procedir a través d'un algorisme d'optimització iteratiu com ara el descens de gradients.
- L'algoritme d'entrenament comença amb un model bàsic i el millora gradualment ajustant diversos paràmetres interns (coeficients) durant cada iteració d'entrenament.
- El resultat d'aquest procés és un model entrenat que podeu utilitzar per fer prediccions sobre dades noves.
- Mesurar l'èxit dels models entrenats: mètriques.

Regressió

- El valor a predir és un nombre continu.
- Prediu valors que el vostre model no ha vist durant l'entrenament.

Classificació

- Entrenar un algorisme per predir una variable dependent que sigui categòrica (pertanyent a un conjunt de valors discrets i finits).
- Model de classificació: fa una predicció que un ítem pertany a un dels dos 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.

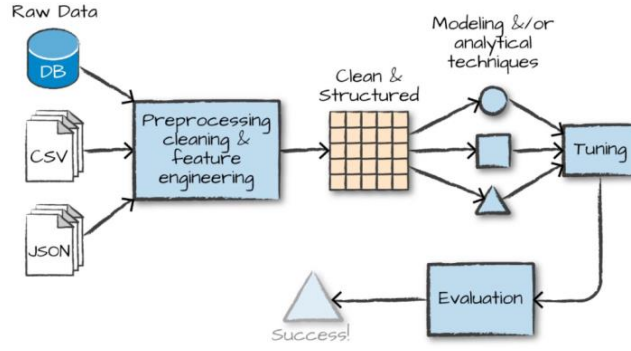
Algoritmes disponibles de Spark ML

Algoritme	Ús
Regressió lineal	Regressió
Regressió logística	Classificació
Decision trees	Regressió/classificació
Gradient boosted trees	Regressió/classificació
Random forests	Regressió/classificació
Naive Bayes	Classificació
Support vector machines	Classificació

Sistemes de recomanació

- Entrada: preferències explícites o implícites de la gent per a diversos productes o articles
    - explícit: valoracions.
    - implícit: comportament observat com els clics.
- L'algoritme fa recomanacions sobre què li pot agradar a un usuari trobant similituds entre usuaris o elements.
- Sortida: recomanacions basades en:
    - Què els ha agradat a usuaris similars (filtratge col·laboratiu).
    - Quins altres productes s'assemblen als que va comprar l'usuari (filtratge basat en contingut).

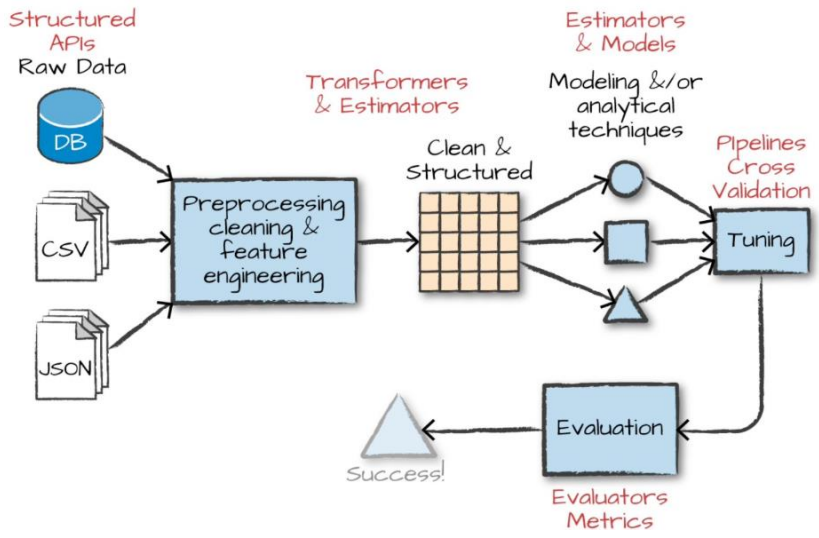
Flux de treball d'aprenentatge automàtic



Passos del flux de treball de ML

1. Recollida i recollida de dades rellevants per a la seva tasca.
2. Neteja i inspecció de les dades per entendre-les millor.
3. Realitzar enginyeria de característiques per permetre que l'algoritme aprofiti les dades d'una forma adequada (p. ex., convertint les dades en vectors numèrics).
4. Utilitzar una part d'aquestes dades com a conjunt d'entrenament per entrenar un o més algorismes per generar alguns models candidats.
5. Avaluar i comparar models amb els criteris d'èxit mesurant objectivament els resultats d'un subconjunt de les mateixes dades que no es van utilitzar per l'entrenament.
6. Utilitzar el model per fer prediccions, detectar anomalies o resoldre reptes empresarials més generals.

Flux de treball de Spark ML



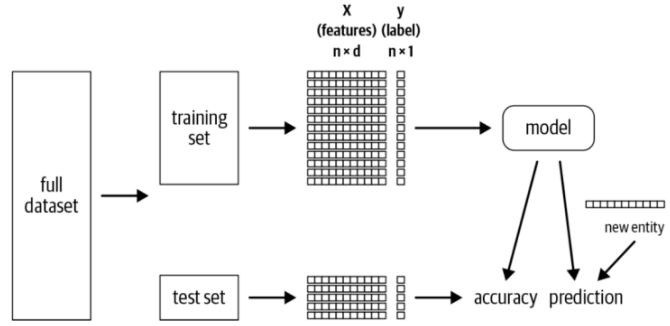
Tipus estructurals MLlib

Els pipelines d'aprenentatge automàtic d'extrem a extrem han de definir el processament mitjançant aquests tipus:

- Transformadors
- Estimadors
- Avaluadors
- Pipelines

Preparació del conjunt de dades: randomSplit

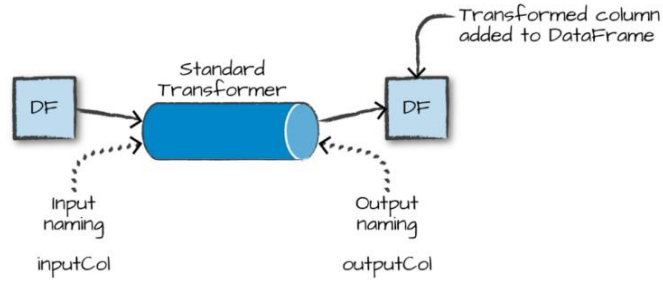
- Dividir el conjunt de dades en dos grups: train i test.
- Molts científics de dades utilitzen 80/20 com a divisió estàndard de train/test.
- Per què? Si es construeix un model sobre tot el conjunt de dades, és possible que el model memoritzés o "sobreajusti" les dades d'entrenament que es proporcionen.
- 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.



Transformadors

- A Spark accepten un DataFrame com a entrada i retornen un nou DataFrame amb normalment una columna afegida.
- Els transformadors no aprenen de les dades, sinó que apliquen transformacions basades en regles mitjançant el mètode transform().
- S'utilitza en preprocessament i enginyeria de funcions (feature engineering).

Exemple: convertir variables categòriques de cadena en valors numèrics.



VectorAssembler transformer: concatena totes les característiques en un gran vector.

Estimadors

- Aprenen paràmetres a partir de les dades.
- Agafa un DataFrame i retorna un Model.
- Això requereix dues passades sobre les 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.

Prediccions amb models

```
lrModel = lr.fit(trainDF) # Retorna un transformador
predDF = lrModel.transform(testDF) # Utilitzem el transformador per aplicar els
                                   paràmetres del model a nous punts de dades
                                   per generar prediccions
```

Avaluadors

- Permet veure com funciona un model determinat segons els criteris que s'especifiquen.
- S'utilitzen per comprovar la qualitat del model.
- Aleshores, es decideix utilitzar aquest model per fer prediccions.
- Avaluadors típics: R^2, RMSE (Root Mean Squared Error).

Pipelines

- Es poden definir les transformacions, estimacions i avaluacions individualment.
- Sovint és més fàcil especificar els passos com a etapes en un pipeline.
- Aquest pipeline és similar al concepte de pipeline de scikit-learn.

Principal conceptes ML?

Els principals conceptes de ML són: carregar dades, neteja les dades, enginyeria de característiques, entrenament del model i avaluació del model.

Principal conceptes Spark MLib?

els principals conceptes de Spark MLib són: carregar dades, neteja les dades, transformador, estimador i avaluador.

ML vs Spark MLib?

- Carregar dades i netejar dades: ambdues disciplines comparteixen aquests passos, donat que són fonamentals per a qualsevol procés de ML.
- Enginyeria de característiques (ML) vs Transformador (Spark MLlib): en ML general, l'enginyeria de característiques inclou totes les transformacions necessàries per preparar les dades. En Spark MLlib, aquestes transformacions es gestionen específicament mitjançant transformadors.
- Entrenament del model (ML) vs Estimador (Spark MLlib): En ML general, l'entrenament del model es refereix a l'ús d'algoritmes per ajustar un model. En Spark MLlib, un estimador és l'objecte que s'ajusta a les dades per produir un model (transformador).
- Avaluació del model (ML) vs Avaluador (Spark MLlib): Ambdós conceptes impliquen mesurar el rendiment del model, però en Spark MLlib, l'avaluador és un objecte específicament dedicat a aquesta tasca.

Conceptes	ML	Spark MLlib
Carregar dades	Sí	Sí
Netejar dades	Sí	Sí
Enginyeria de característiques	Sí	No (ús de Transformador)
Transformador	No	Sí
Entrenament del model	Sí	No (ús d'Estimador)
Estimador	No	Sí
Avaluació del model	Sí	No (ús d'Avaluador)
Avaluador	No	Sí

Spark SQL

Ofereix una integració molt més estreta entre relacional i procedimental en el processament, mitjançant una API de DataFrame declarativa.
Inclou un optimitzador altament extensible, Catalyst, que facilita l'addició de fonts de dades, regles d'optimització i tipus de dades.

- **spark.sql**: per executar sentències SQL directament sobre DataFrame.

Precedents

- MapReduce
  - Interfase de programació low-level.
  - Cost Alt, és necessari optimització manual
- Pig, Hive, Dremel, Shark
  - Avantatges de les declarative queries per a proveir de optimitzacions automàtiques.
  - L'aproximació de la part relacional és insuficient per a aplicacions de Big Data.
    ETL des de/cap font de dades semi-/no estructurades (p.ej JSON) necessita codi específic.
    Analítica avançada (ML, graph processing) són difícils d'expressar en un sistema relacional.

Objectius

- Admet el processament relacional tant dins dels programes Spark com de fonts de dades externes mitjançant una API fàcil de programar.
- Proporciona un alt rendiment mitjançant tècniques de DBMS ben conegudes.
- Dona suport fàcilment a noves fonts de dades, incloses dades semiestructurades i bases de dades externes susceptibles de consultes federades.
- Permet l'extensió amb algorismes d'anàlisi avançats, com ara el processament de gràfics i l'aprenentatge automàtic.

Funcions de transformació de columnes

- select: per seleccionar una o més columnes.
- withColumn: per afegir, reemplaçar o modificar una columna existent amb una nova columna.
- drop: per eliminar una o més columnes d'un DataFrame.

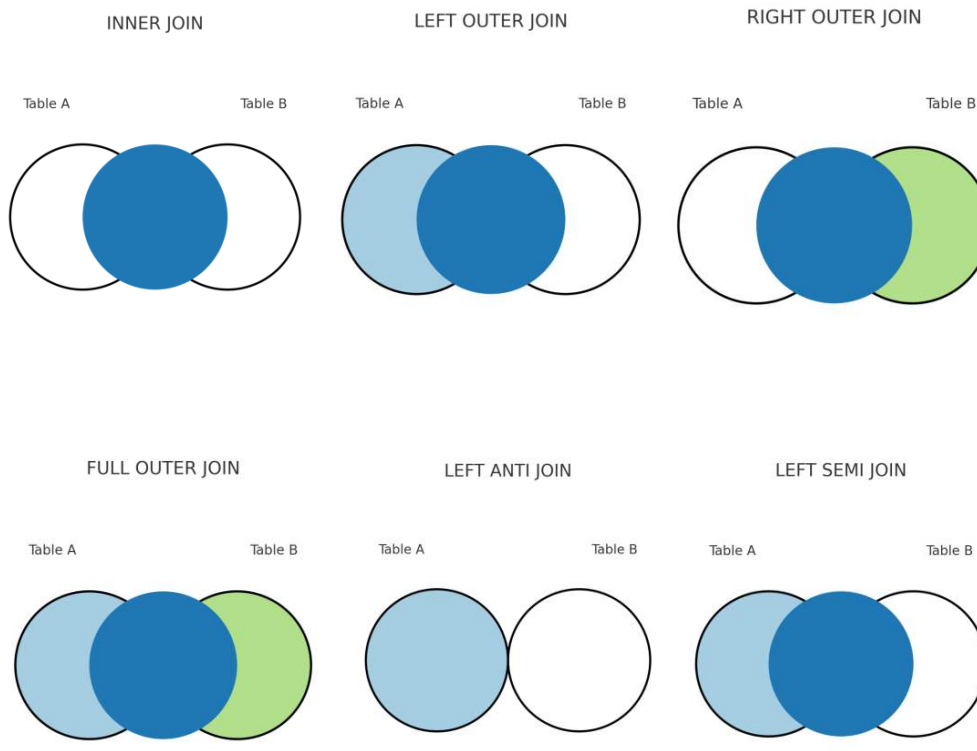
- filter (o where): per filtrar files basades en una condició.
- groupBy: per agrupar dades basades en una o més columnes.
- agg (o groupBy().agg): per realitzar agregacions com ara sumes, mitjanes, màxims, mínims, etc., sobre dades agrupades.
- orderBy (o sort): per ordenar les files d'un DataFrame basades en una o més columnes.

Funcions d'Agregació

- count: per comptar el nombre de files en un DataFrame.
- sum, avg, min, max: per calcular la suma, mitjana, valor mínim i valor màxim d'una columna numèrica.
- groupBy().agg: Per aplicar funcions d'agregació després d'una operació de groupBy.

Operacions Join

- Natural: combina files de dues taules basant-se en les columnes amb el mateix nom i tipus de dades, seleccionant només les files on aquests valors coincideixen.
- Inner: combina files de dues taules basant-se en una condició explícita, retornant només les files que compleixen aquesta condició en ambdues taules.
- Outer: retorna totes les files de les taules combinades, incloent les files que no tenen coincidències a l'altra taula, amb valors nuls per les columnes de la taula que no té coincidència.
- Left outer: retorna totes les files de la taula esquerra i les coincidències de la taula dreta, omplint amb valors nuls si no hi ha coincidència.
- Right outer: retorna totes les files de la taula dreta i les coincidències de la taula esquerra, omplint amb valors nuls si no hi ha coincidència.
- Left anti: retorna només les files de la taula esquerra que no tenen cap coincidència a la taula dreta.
- Left semi: retorna només les files de la taula esquerra que tenen almenys una coincidència a la taula dreta, però sense incloure les columnes de la taula dreta.



Conceptes ML

Transformadors

- VectorAssembler: no necessita un ajustament (fit), donat que combina columnes en un vector.
- StandardScaler: requereix un ajustament (fit) per aplicar la transformació d'escala. Això es duu a terme per calcular estadístiques sobre les dades per la transformació posterior.
- Es poden fer servir de forma complementària.
- Cap dels dos transformadors entrena un model.

Exemple d'ús de Pipeline

```
from pyspark.ml import Pipeline
pipeline=Pipeline(stages=[vecAssembler, scaler, lr])
pipelineModel=pipeline.fit(trainDF)
predDF=pipelineModel.transform(validationDF)
predDF.select("prediction","SalePrice","features").show(5)
```

En un Pipeline s'especifiquen les etapes que es vol que passin les dades, en ordre, i Spark s'encarrega del processament. El mètode fit() s'aplica al Pipeline complet per ajustar tots els seus components a les dades d'entrenament, mentre que transform() s'aplica al Pipeline per aplicar totes les transformacions definides a les dades d'entrada, sigui d'entrenament o de validació.

Linear Regression vs Random Forest:

- Linear Regression necessita vecAssembler, per combinar les característiques en un vector, i StandardScaler, per aplicar la transformació d'escala.
- Random Forest només necessita vecAssembler, per combinar les característiques en un vector. No requereix StandardScaler, donat que les transformacions d'escala no són necessàries.
- Mentre que en la regressió lineal és important escalar les característiques per garantir que el model convergeixi més ràpidament i tingui millors prestacions, en Random Forest això no és necessari, ja que l'algoritme és inherentment més flexible respecte a l'escala de les característiques.

Característica	Linear Regression	Random Forest
Necessita `vecAssembler`	Sí	Sí
Necessita `StandardScaler`	Sí	No
Importància de l'escala de les característiques	Alta (convergència més ràpida i millor rendiment)	Baixa (algoritme flexible respecte a l'escala)

Exemple d'avaluacions:

```
from pyspark.ml.regression import LinearRegression
lr = (
    LinearRegression(
       featuresCol="features",
       labelCol="SalePrice",
       maxIter=10,
       regParam=0.8,
       elasticNetParam=0.1,
    )
)
lrModel = lr.fit(scTrainDF)
lrSummary = lrModel.summary
print(f"RMSE: {lrSummary.rootMeanSquaredError:f}")
print(f"r2: {lrSummary.r2:f}")
```

- **lrSummary.r2**: mesura R^2 calculada internament pel model de regressió lineal sobre les dades d'entrenament.

```
from pyspark.ml.evaluation import RegressionEvaluator
lrEvaluator = (
    RegressionEvaluator(
        predictionCol="prediction",
        labelCol="SalePrice",
        metricName="r2",
    )
)
r2=lrEvaluator.evaluate(predDF)
r2

lrPredictions = pipelineModel.transform(validationDF)

print(f"R Squared (R2) on val data = {lrEvaluator.evaluate(lrPredictions):g}")
```

Per un costat, r2 mesura la R^2 calculada amb un RegressionEvaluator sobre les prediccions del model en les dades de validació (predDF). En canvi, lrEvaluator.evaluate(lrPredictions) mesura R^2 calculada amb un RegressionEvaluator sobre les prediccions del model en les dades de validació (validationDF), després de passar pel PipelineModel.

Examen

1-[0,5 punts] Quins són els principals punts de partida i problemes a resoldre quan es treballa amb bases de dades en memòria?

- Requeriments d'emmagatzemament de dades en un espai d'adreces únic.
- La complexitat de la solució es redueix donat que no cal buscar informació fora de la memòria principal i no cal dissenyar mètodes per mantenir la consistència.
- Accés ràpid a un espai de memòria de gran abast.
- No cal optimitzar les estructures de dades per ser guardades al disc.

4-[1 punt] Què és Spark? Per a que serveix i qui tipus de paral·lelisme ofereix per resoldre les consultes? Quins tipus d'APIs ofereix per escriure fluxos de dades?

Spark és un sistema distribuït per processar volums molt grans de dades. Ofereix estructures de dades paral·leles i tolerants a errors que, mitjançant tasks, CPUs, cores i particions, distribueixen i executen el flux de treball de manera eficient i escalable. Proporciona APIs per escriure fluxos de dades com ara: DataFrames, Spark SQL o Spark MLib, entre vàries.

5-[1 punt] Què és una funció lambda? Pots donar un exemple de funció lambda per calcular el quadrat dels elements d'un vector?

- Funció de mida petita sense nom: anònima.
- S'aplica en paral·lel a cada element del conjunt de dades d'entrada per produir una sortida derivada d'aplicar la funció map(number => number * number)

10-[1,5 punts] Quan dissenyem una aplicació amb Spark Streaming, és possible mantenir un estat global per calcular per exemple el número total de registres analitzats? Com es pot solucionar? Pots donar un exemple de la transformació necessària?

És possible amb un objecte Spark DStream que permet combinar: dades persistents d'un estat global de l'aplicació i noves dades que arriben en processar un mini-batch.

```
val amountState = registerAmount.updateStateByKey((newVals, previousState => { pre
viousState match{
case None => newVals.sum
case Some(total) => newVals.sum + total }
}))
```