Ciencia e Ingeniería de Datos en el Mundo Real

Implementación de modelos estadísticos de principio a fin en entornos productivos

Daniel Bestard Delgado - Lead Data Scientist en Damavis Studio

www.damavis.com | @damavisstudio

CONTENIDO

Contenido

1. Instalar y configurar **Apache Hadoop** para levantar un **HDFS**
2. Instalar y configurar **Apache Spark** e integrarlo con Apache Hadoop
3. Introducción a **arquitectura de software** aplicada a aplicaciones de **Scala Spark**
4. Desarrollar aplicación de Scala Spark con **IntelliJ IDEA**
5. Entrenamiento de **modelo estadístico** de elasticidad de precio
6. Ejecución con **spark-submit** de aplicaciones

Apache Hadoop

¿Qué es Apache Hadoop?

Apache Hadoop es un framework open source utilizado para almacenar y procesar big data de forma distribuida y tolerante a fallos

Módulos principales:

- **Hadoop Distributed File System (HDFS)** → almacenamiento de datos
- **Hadoop MapReduce** → procesamiento de datos
- **Apache YARN** → gestión del clúster

Pasos para instalar Apache Hadoop

1. Identificar directorio donde almacenar software de Apache. Generalmente en `/opt`. En esta demo `~/opt`.
2. Siguiendo buenas prácticas de administrados de sistemas se deberían crear usuarios específicos con permisos limitados para lanzar los softwares a instalar. Por simplicidad se omitirá en esta demo.
3. `wget` del software de Apache Hadoop y descomprimir.
4. Crear link simbólico sin la versión: `ln -s hadoop-3.3.1 hadoop`
5. Instalar Java 8: `apt install openjdk-8-jdk -y`
6. `wget` del software de Apache Hadoop y descomprimir.
7. Crear directorio `config-files` para desacoplar la configuración de una versión concreta de software.
8. Declara la variable de entorno `JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64` en `hadoop/etc/hadoop/hadoop-env.sh` moviendo antes este ficher a `config-files` y creando link simbólico

Pasos para instalar Apache Hadoop

9. Para evitar replicación de datos en un entorno local, modifica la propiedad `dfs.replication` a 1 en `hadoop/etc/hadoop/hdfs-site.xml`, que una vez más debe ser movido a `config-files` y crear link simbólico
10. Especificar que el servicio de HDFS debe ser levantado en local modificando la propiedad `fs.defaultFS` al valor `hdfs://localhost:9000` del fichero `hadoop/etc/hadoop/core-site.xml`, que una vez más debe ser movido a `config-files` y crear link simbólico
11. Comprueba que puedes hacer ssh localhost. En caso contrario, copia tu clave pública a `.ssh/authorized_keys`
12. Para lanzar scripts de Hadoop desde cualquier directorio, modifica la variable de entorno `PATH` del `.bashrc` añadiendo la localización de los binarios de Hadoop (`hadoop/bin`) y carga los cambios con el comando `source`

Pasos para instalar Apache Hadoop

13. Lanza HDFS ejecutando `hdfs namenode -format` y, posteriormente, ejecutando `hadoop/sbin/start-dfs.sh`
14. Crea directorio de entorno de `pre` y otro de `live` usando `hdfs dfs -mkdir hdfs:///{env}`
15. Baja datos de compra de cereales de aquí y descomprime
16. Copia el fichero `region1_company1.csv` a HDFS al directorio `hdfs:///{env}/raw/cereal.csv` usando el comando `hdfs dfs -copyFromLocal region1_company1.csv hdfs:///{env}/raw/cereal.csv`

Apache Spark

¿Qué es Apache Spark?

Apache Spark es un framework open source utilizado para computación en clúster para así poder procesar big data de forma distribuida y tolerante a fallos

Modos de lanzamiento:

- **Standalone** → Un solo nodo. Utilizado en la demo de hoy
- **Apache YARN** → Artículos de Damavis, uno de introducción y otro de personalización
- **Kubernetes** → Clúster de contenedores Docker

Pasos para instalar Apache Spark

1. `wget` del software de Apache Spark sin Hadoop y descomprimir.
2. Crear link simbólico sin la versión: `ln -s spark-3.2.0-bin-without-hadoop spark`
3. Declara la variable de entorno `SPARK_DIST_CLASSPATH=$(hadoop --config /opt/hadoop/etc/hadoop classpath)` en `spark/conf/spark-env.sh` moviendo antes este fichero a `config-files` y creando link simbólico (previamente hay que borrar `spark/conf/spark-env.sh.template`)
4. Para lanzar scripts de Hadoop desde cualquier directorio, modifica la variable de entorno `PATH` del `.bashrc` añadiendo la localización de los binarios de Hadoop (`spark/bin`) y carga los cambios con el comando `source`
5. Lanza `spark-shell` y lee el fichero que hemos copiado en HDFS.

Arquitectura de software aplicada a aplicaciones de Scala Spark

Buenas prácticas en el desarrollo de software

- Principios **SOLID**:
  - **S** → Single responsability
  - **O** → Open closed
  - **L** → Liskov substitution
  - **I** → Interface segregation
  - **D** → Dependency inversion
- Uso de **IDE**:
  - Scala/Java → Intellij IDEA
  - Python → Pycharm
- Gestión de **entornos**:
  - PRE → entorno de prueba y testeo
  - LIVE → entorno de producción
- **Testing**
  - cada subida a LIVE debe tener una capa de seguridad en la que todos los tests pasen satisfactoriamente antes de la subida
- Pasar keys y secrets como **variables de entorno**

Ejemplo arquitectura de software

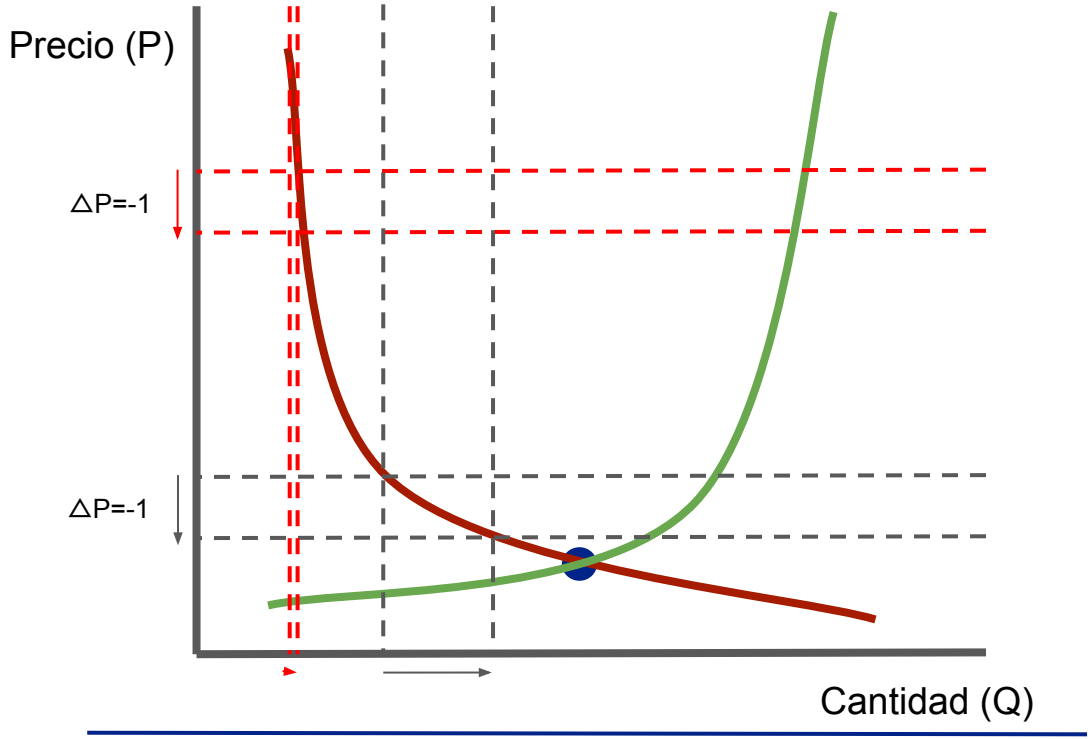
- `src`
  - `main`
    - `resources` → ficheros de configuración con extensión `.conf`
    - `scala`
      - `com…`
        - `stage` → contiene la lógica de negocio. Se le pasan los datos, por lo que no interacciona ni con `repository` ni con `resource`.
        - `pipeline` → se le pasa el `repository` y los `stages`, los cuales se ejecutan en el orden especificado en esta clase
        - `repository` → usando el resource lee/escribe datos concretos de la aplicación.
          - `resource` → lectura y escritura en fuentes. Ajeno a lógica de negocio, es decir, no sabe que lee/escribe, solo sabe de formatos.
      - `main`
  - `test`
- `build.sbt` → Dependencias del aplicativo
- `.gitignore`
- `.scalafmt.conf` → Configuración para formatear el código

Ejemplo aplicación Scala Spark

- Nos bajamos el código de aquí.
- Necesitaremos **sbt**
  - Software open source de build de proyectos scala parecido a Apache MAVEN
  - Instrucciones de instalación aquí
  - `sbt compile` genera un paquete `.jar` que contiene toda la aplicación
  - En el fichero `.build.sbt` se especifican las dependencias que se incluyen en el `.jar` cuando se compila
  - Para ejecutar desde el IDE se deben bajar las dependencias de Apache Spark, por lo que se debe quitar `% Provided` (el Spark instalado en local no es usado).
- Abrimos el proyecto con Intellij IDEA
  - El script requiere de la variable de entorno `MASTER` de Spark
  - Al usar un modo standalone de spark especificamos el valor `local[*]`
- El script recibe un argumento llamado `environment`, que puede ser `live` o `pre`
- Ejecutamos con el comando `runMain main.Main --environment pre`

Cálculo de elasticidad de demanda

Modelo Demanda Multiplicativo



Ref: 1992, W. Nicholson, Microeconomic Theory: Basic Principles and Extensions

$$Q = A * P^{-b}$$

donde **A** y **b** son parámetros

$$\log(Q) = C - b * \log(P)$$

Interpretación Modelo de Demanda Multiplicativo

$$Q = A * P^{-b}$$

$$\log(Q) = \log(A) - b * \log(P)$$

$$\log(Q_1) - \log(Q_0) = [\log(A) - b * \log(P_1)] - [\log(A) - b * \log(P_0)]$$

$$\log(Q_1) - \log(Q_0) = b * [\log(P_1) - \log(P_0)]$$

$$b = [\log(Q_1) - \log(Q_0)] / [\log(P_1) - \log(P_0)]$$

$b \approx \Delta\%Q / \Delta\%P$

Ref: 1992, W. Nicholson, Microeconomic Theory: Basic Principles and Extensions

Si $A = 1*10^7$ y $b = -3 \rightarrow$ ¿$\triangle Q$, si $P_0=100$ y $P_1=105$?

1

Cálculo Exacto

$Q_0 = (1 * 10^7) * 100^{-3} = 10$

$Q_1 = (1 * 10^7) * 105^{-3} = 8.64$

$Q_1 - Q_0 = -1.36$

$\triangle\%Q = -1.36/10 = -13.6\%$

2

Cálculo Aproximado

$\triangle\%P = (105 - 100) / 100 = 5\%$

$-3 \approx \triangle\%Q / 5\%$

$\triangle\%Q \approx -15\%$

Spark submit

Spark submit

Compilamos con las dependencias de Spark como `Provided` dado que emplearemos la instalación que hemos hecho en local

```
spark-submit --class main.Main --master local[*] {JAR} --environment pre
```