Introduction to Big Data

why is needed?

How is implemented?

What tools ?

1

Who is yout host?

- Miguel Angel Rodriguez, Computer Science and IA área in Huelva University
- Dr. in Computer Science by Huelva university
- Thesis in Distributed Data mining and paralllel genetic algorithms
- Teaches Machine Learning, Big data , IOT , Genetic Algorithms

2

What is the aim of this webinar

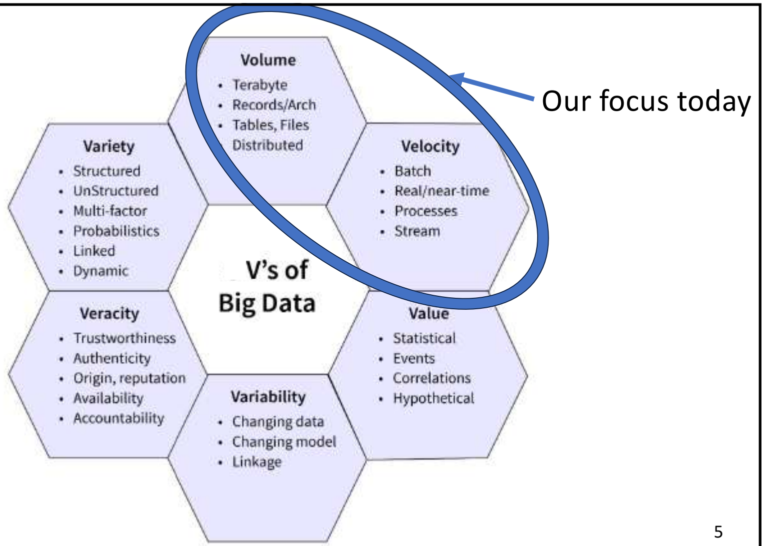
- Get an overall view on the big data concepts
- Get a Good understanding of history, causes, architectures and Issues
- Focus on trending platforms and tools

3

Science of the Data and Big Data

- ***Big Data*** *are data whose* ***scale, diversity*** *and* ***complexity*** *require of* ***new architectures, techniques, algorithms*** *and* ***Models of analysis*** *for handle them****, extract worth*** *and* ***knowledge hidden*** *from they*
- *The* ***Science of the Data:*** *discipline that serves of different others (techniques of AI like Machine Learning, technologies of storage and processing in the field of the Big Data) in order to obtain useful knowledge from data*

4

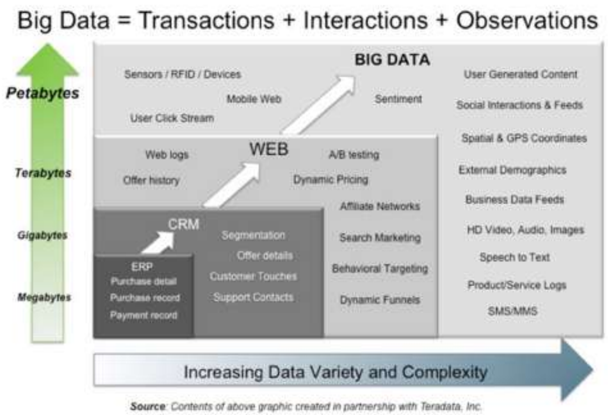


5

5

Volume

Volume: Large amount of data that exceeds the handling capacity of traditional tools.



6

Speed

- Speed: The speed with which data is generated, processed and analyzed.

7

Data Generation Speed

- 456,000 Tweets/Xs are sent per minute, which is 656 million Tweets per day
- Companies like X (former Twitter) generate millions of tweets per minute, requiring fast processing for real-time analysis.

8

Milestones in the Development of Big Data

- 1970 introduction of relational database
- 1980-1990 Emergence of Data Warehousing
- 2000 (mid) Rise of Hadoop – MapReduce
- 2000 (late) : NoSQL Datbases: Cassandra, MongoDB
- 2010 Spark and in memoryProcessing
- 2010+ Streaming
- 2015- RNN- Recurrent Neural Networks
- 2020 Transformers - LLM

9

Data wareHouse

- **Traditional databases** struggled to handle the growing volume of data generated by businesses and the increasing need **for analytical insight**
- Centralized Data Repositories: aimed to provide a unified view of an organization's data for analytical purposes.
- Integration of data from various sources , including not databases
- **OLAP**: multidimensional analysis on large datasets
  - shift from transaction-oriented databases to systems designed for analytical processing
  - Preagregation of data to suport complex queries on historical data
  - **Scaling**: Often involves vertical , migration. complexity

10

The rise of HDFS – MapReduce - Hadoop

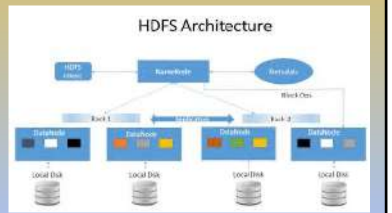
Apache Hadoop Distributed File System (HDFS)

Apache Hadoop by Doug Cutting and Mike Cafarella in 2006.

Inspired by Google's MapReduce and Google File System (GFS)

Hadoop's initial release in 2006 brought attention to HDFS as its primary storage layer, designed to handle large-scale data storage and processing across **clusters of commodity hardware**

Ghemawat, S., Gobioff, H., & Leung, S. T. (2003, October). **The Google file system.** In *Proceedings of the nineteenth ACM symposium on Operating systems principles* (pp. 29-43).
https://shorturl.at/istHL

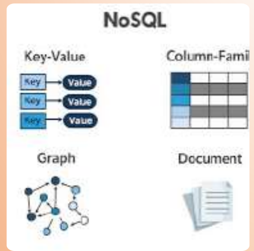


Cited by more than 10,000 Papers!!!

11

11

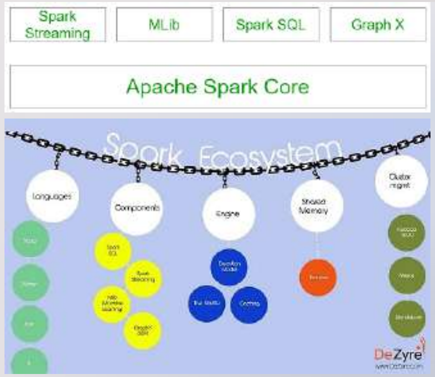
NoSQL Databases



- Flexible Schema: They support semi-structured or unstructured data.
- Horizontal Scalability: Distributing data across nodes.
- High Availability: They ensure data availability with automatic failover.
- Performance: Optimized for fast read and write operations.
- Distributed Architecture: Operates across multiple nodes in a cluster.
- Variety of Data Models: Including key-value, document, column-family, and graph.
- Use Cases: Commonly used for real-time analytics, web apps, IoT, and more.
- **Cassandra, Mongodb** are used in

12

Apache Spark: Memory wins over Hard disk



- Processes data in-memory, reducing disk I/O and improving performance up to 100 times or more.
- DAG engine for optimized task execution and reduced disk writes.
- High-level APIs for batch, interactive, machine learning, and stream processing.
- Ensures fault tolerance through RDD lineage and checkpointing without excessive data replication.

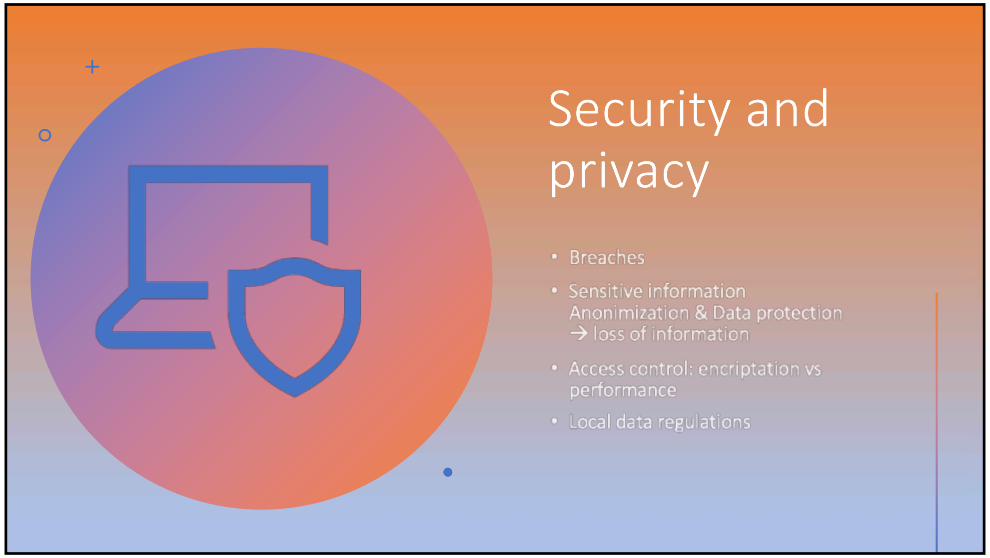
Zaharia M., Chowdhury M., Franklin M.J., Shenker S. (2010) Spark: cluster computing with working sets

13

Problems of handling large volumes of data

- Scalability
- Storage
- Processing
- Security and Privacy

14

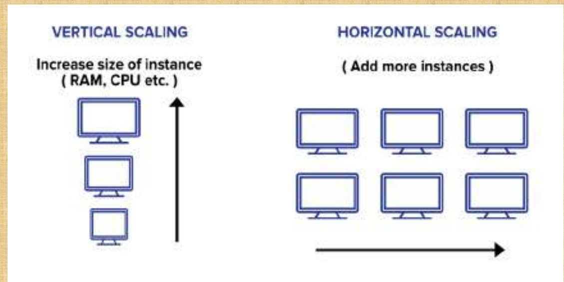

Security and privacy
• Breaches
• Sensitive information
Anonimization & Data protection
→ loss of information
• Access control: encriptation vs performance
• Local data regulations

15


Scalability
Scalability: Need to handle sustained data growth.

16

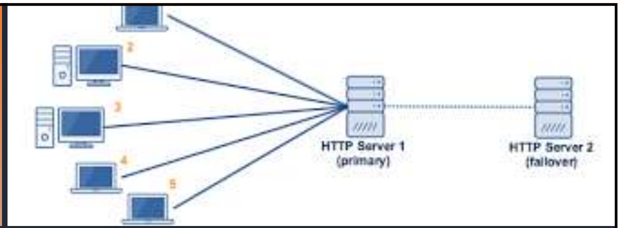
- Scalability refers to a system's ability to handle growth and workload.
  - Vertical:Increasing the capacity of a machine by adding more resources (CPU, RAM, etc.) to a single instance.
  - Horizontal: Increase in capacity by distributing the load among multiple nodes or machines.

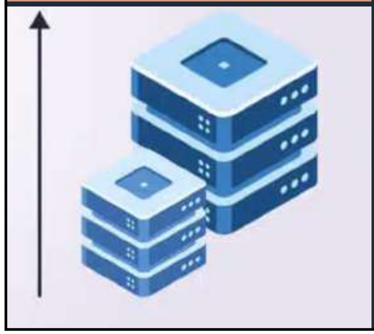


17

17

Vertical

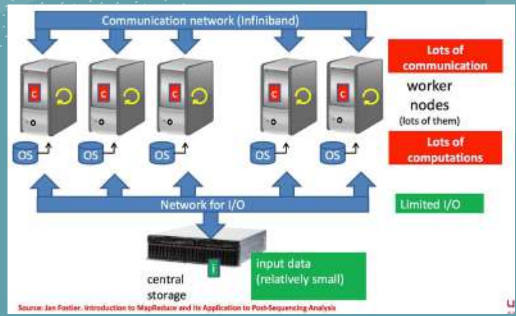




- Advantages:
  - Higher performance per machine.
  - Less management complexity.
- Issues:
  - Fault tolerance
  - Cost associated with more powerful HW.
  - HW limited scalability
- Solutions:
  - Virtualization.
  - Migration to more powerful HW.

18

Horizontal



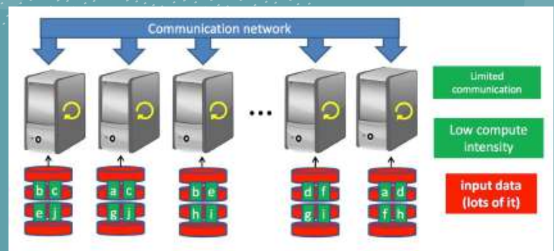
- Advantages:
  - Easy adding of extra resources.
  - Greater fault tolerance.
  - Easier scalability for large volumes of data.
- Issues:
  - Problem: Coordination between nodes can be complex.
  - Need to maintain consistency between nodes.
  - Difficulties in managing large data sets.

19

19

NEW HDFS→

Horizontal



- Implementation of distributed databases that support distributed transactions.
- Use of distributed storage systems such asHadoop Distributed File System (HDFS)or Amazon S3.

20

20

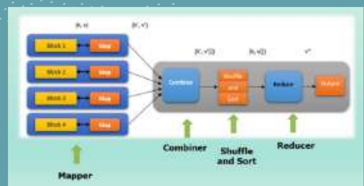
HDFS – The power of distributed computation

- The technology of Big Data was based originally in:
  - Split the sets of data between a set of servers
  - Design the algorithm as a method of resolution of the problem by parts separated, and whose results can be integrated subsequently, for obtain the solution final of the problem.
  - This idea was proposal by Google in a new scalable distributed computation based on MapReduce

21

21

Map Reduce



- A model of programming parallel created and used by Google from 2004
- His aim is treat big sets of data in groups of computers
- Conceptually simple, elegant and extensible for multiple Applications
- Popularized by he project of code open Hadoop, used by Yahoo!, Facebook, amazon,..
- Programmable in C++, Java, Scala or Python

22

22

Processing Capability

- Processing: The ability to process large amounts of data efficiently.
- Algoritmic Scalability is a great concern because tend to have non linear growth
- Not all Algorithm have an equivalent implementation over parallel or distributed arquitectures

23

23

Analysis Speed

The generation and continuous processing of data does not produce knowledge by itself.

Clasical Reports

Big data handling

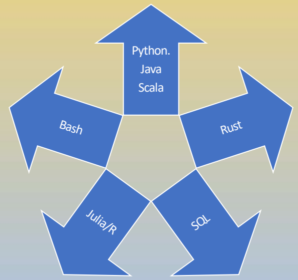
SQL Like

Data Warehouse

The need to generate predictions, clustering, classify

Google analyzes large sets of data in real time to improve the accuracy of search results.

24



What Programming Languages are we going to use??

- **Python is a popular choice due to its user-friendly features, versatility, and strong community support**.
- R and Julia offer statistical computing, data visualization, and machine learning.
- Rust for high-performance and memory management capabilities.
- Bash scripts are useful for automation and data pipelines.
- SQL as it is a declarative languaje, similar to Hive/pig latín concepts for handling data relations.

*Java/Scala Still present in most platforms… but decreasing vs python

25

25

Big Data Tools Categories

- Data Storage and Management Tools
- Data Processing and Analytics Tools
- Data Visualization Tools
- Data Integration and ETL Tools
- Data Governance and Security Tools

26

26

Big Data Tools : Storage , Processing, Analysis

Hadoop: An open-source framework for distributed storage and processing of large datasets.

Apache Spark: A fast and general-purpose cluster computing system for big data processing both **Stream and Batch**.

Apache Kafka: A distributed streaming platform for building real-time data pipelines and applications.

Apache Hive: A data warehouse infrastructure built on top of Hadoop for querying and analyzing large datasets.

Apache HBase: A distributed, scalable, and NoSQL database for big data workloads.Apache

Apache PIG: Scripting language used to interact with HDFS.

27

Machine Learnig Tools:

Weka: graphical user interface and a wide range of machine learning algorithms for data mining. No big data.

MLlib (Apache Spark): MLlib is a scalable machine learning library.

Scikit-learn (sklearn): popular Python library for machine learning. No Big data

Apache Flink:real-time stream and batch processing. Data transformation.Machine Learning (based in MIllib) .monitoring..

NumPy: library for scientific computing in Python. No Big data

Pandas: library for data manipulation and analysis in Python. DataFrames. data cleaning, transformation, aggregation, and visualization. No Big data

TensorFlow: open-source machine learning framework for building and training deep learning models.

PyTorch: deep learning framework developed by Facebook's AI Research labOther Machine Learning Tools

Keras: A high-level neural networks API written in Python

28

Map Reduce

1

MapReduce

- Basics
- A close look at MapReduce data flow
- Additional functionality
- Scheduling and fault-tolerance in MapReduce
- Comparison with existing techniques and models

2

2

MapReduce

- Basics
- A close look at MapReduce data flow
- Additional functionality
- Scheduling and fault-tolerance in MapReduce
- Comparison with existing techniques and models

Problem Scope

- *MapReduce* is a programming model for data processing
- The power of MapReduce lies in its ability to scale to 100s or 1000s of computers, each with several processor cores
- How large an amount of work?
  - Web-Scale data on the order of 100s of GBs to TBs or PBs
  - It is likely that the input data set will not fit on a single computer's hard drive
  - Hence, a distributed file system (e.g., Google File System- GFS) is typically required

MapReduce

Commodity Clusters

- MapReduce is designed to efficiently process large volumes of data by connecting many commodity computers together to work in parallel
- A *theoretical* 1000-CPU machine would cost a very large amount of money, far more than 1000 single-CPU or 250 quad-core machines
- MapReduce ties smaller and more reasonably priced machines together into a single cost-effective *commodity cluster*

5

5

MapReduce

Isolated Tasks

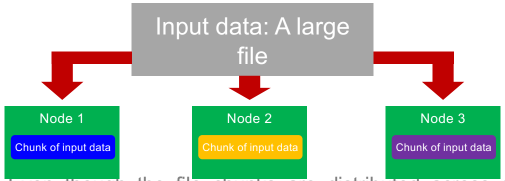
- MapReduce divides the workload into multiple *independent tasks* and schedule them across cluster nodes
- A work performed by each task is done *in isolation* from one another
- The amount of communication which can be performed by tasks is mainly limited for scalability reasons
    - The communication overhead required to keep the data on the nodes synchronized at all times would prevent the model from performing reliably and efficiently at large scale

6

6

MapReduce — Data Distribution

- In a MapReduce cluster, data is distributed to all the nodes of the cluster as it is being loaded in
- An underlying distributed file systems (e.g., GFS) splits large data files into chunks which are managed by different nodes in the cluster



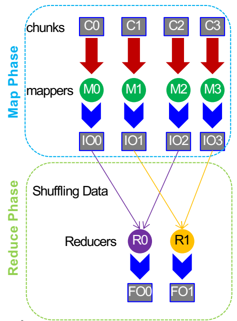
- Even though the file chunks are distributed across several machines, they form *a single namesapce*

7

7

MapReduce — MapReduce: A Bird's-Eye View

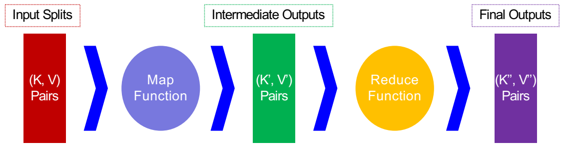
- In MapReduce, chunks are processed in isolation by tasks called *Mappers*
- The outputs from the mappers are denoted as intermediate outputs (IOs) and are brought into a second set of tasks called *Reducers*
- The process of bringing together IOs into a set of Reducers is known as *shuffling process*
- The Reducers produce the final outputs (FOs)
- Overall, MapReduce breaks the data flow into two phases, *map phase* and *reduce phase*



8

MapReduce | Keys and Values

- The programmer in MapReduce has to specify two functions, the *map function* and the *reduce function* that implement the Mapper and the Reducer in a MapReduce program
- In MapReduce data elements are always structured as key-value (i.e., (K, V)) pairs
- The map and reduce functions receive and *emit* (K, V) pairs



9

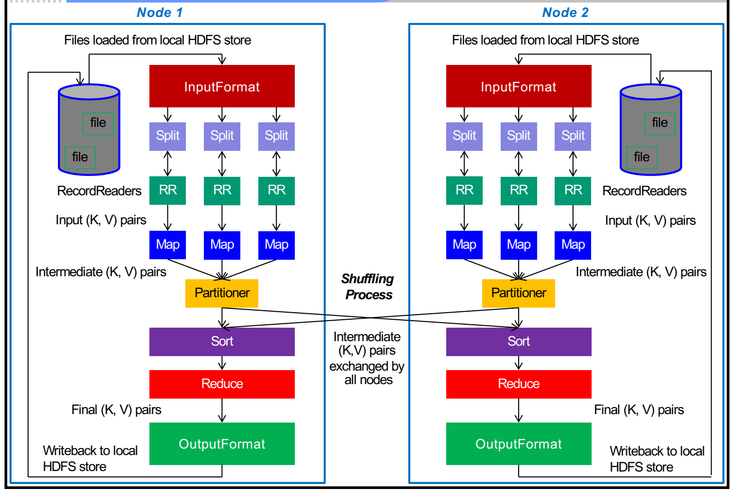
MapReduce | MapReduce

- Basics
- A close look at MapReduce data flow
- Additional functionality
- Scheduling and fault-tolerance in MapReduce
- Comparison with existing techniques and models

10

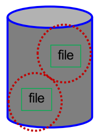
10

MapReduce



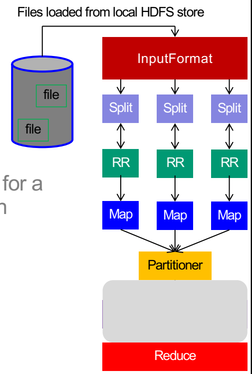
MapReduce — Input Files

- *Input files* are where the data for a MapReduce task is initially stored
- The input files typically reside in a distributed file system (e.g. HDFS)
- The format of input files is arbitrary
  - Line-based log files
  - Binary files
  - Multi-line input records
  - Or something else entirely



MapReduce | Partitioner

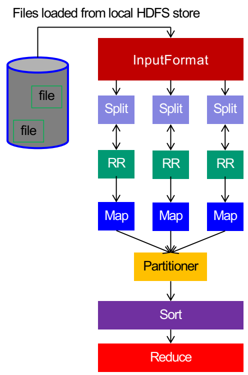
- Each mapper may emit (K, V) pairs to *any* partition
- Therefore, the map nodes must all agree on where to send different pieces of intermediate data
- The *partitioner* class determines which partition a given (K,V) pair will go to
- The default partitioner computes *a hash value* for a given key and assigns it to a partition based on this result



13

MapReduce | Sort

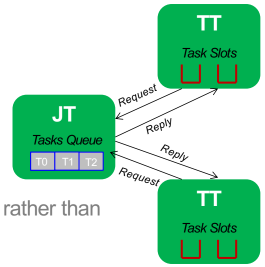
- Each Reducer is responsible for reducing the values associated with (several) intermediate keys
- The set of intermediate keys on a single node is *automatically sorted* by MapReduce before they are presented to the Reducer



14

- In this part, the following concepts of MapReduce will be described:
  - Basics
  - A close look at MapReduce data flow
  - Additional functionality
  - Scheduling and fault-tolerance in MapReduce
  - Comparison with existing techniques and models

- MapReduce adopts a *master-slave architecture*
- The master node in MapReduce is referred to as *Job Tracker* (JT)
- Each slave node in MapReduce is referred to as *Task Tracker* (TT)
- MapReduce adopts a *pull scheduling* strategy rather than a *push one*
  - I.e., JT does not push map and reduce tasks to TTs but rather TTs pull them by making pertaining requests



MapReduce

Fault Tolerance in Hadoop

- MapReduce can guide jobs toward a successful completion even when jobs are run on a large cluster where probability of failures increases

- The primary way that MapReduce achieves fault tolerance is through *restarting tasks*

- If a TT fails to communicate with JT for a period of time (by default, 1 minute in Hadoop), JT will assume that TT in question has crashed

  - If the job is still in the map phase, JT asks another TT to re-execute *all Mappers that previously ran at the failed TT*

  - If the job is in the reduce phase, JT asks another TT to re-execute *all Reducers that were in progress on the failed TT*

17

17

MapReduce

Speculative Execution

- A MapReduce job is dominated by the slowest task

- MapReduce attempts to locate slow tasks (*stragglers*) and run redundant (*speculative*) tasks that will optimistically commit before the corresponding stragglers

- This process is known as *speculative execution*

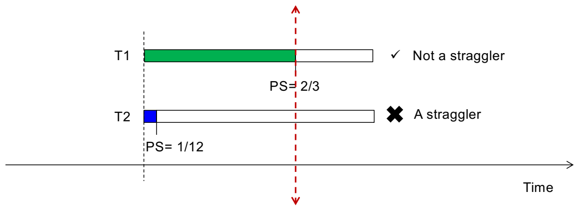
- Only one copy of a straggler is allowed to be speculated

- Whichever copy (among the two copies) of a task commits first, it becomes the definitive copy, and the other copy is killed by JT

18

MapReduce — Locating Stragglers

- How does Hadoop locate stragglers?
  - Hadoop monitors each task progress using a *progress score* between 0 and 1
  - If a task's progress score ***is less than*** (average – 0.2), and the task has run for at least 1 minute, it is marked as a straggler



19

MapReduce — MapReduce

- In this part, the following concepts of MapReduce will be described:
  - Basics
  - A close look at MapReduce data flow
  - Additional functionality
  - Scheduling and fault-tolerance in MapReduce
  - Comparison with existing techniques and models

20

20

Traditional Models

Aspect	Shared Memory	Message Passing	MapReduce
Communication	Implicit (via loads/stores)	Explicit Messages	Limited and Implicit
Synchronization	Explicit	Implicit (via messages)	Immutable (K, V) Pairs
Hardware Support	Typically Required	None	None
Development Effort	Lower	Higher	Lowest
Tuning Effort	Higher	Lower	Lowest

Advantages and Disadvantages of the model MapReduce

- Advantages :
  - Automate aspects of parallelism and of tolerance to failures
  - It allows to the programmer get focus only on functionality
  - The improvements in the library benefit to all the Applications
  - Is scalable and save costs in hardware (*commodity hardware*), programming and administration
- Disadvantages:
  - Not all the Applications can be adapted to this model, some Models are forced with extra stages in order to adjust the application to the model
  - Existence of intermediate values *(keys and values...)*
  - Appropriate only for calculations in *batch* (batch processing), Not interactive

MapReduce

Limitations:

- *Too many times, when you have a hammer, all things look like nails: "MapReduce is good Enought?".*

- Examples of algorithms that does not adapt to the philosophy MapReduce:
  - Iterative Graphs Algorithms
  - Descent Gradient Algorithms
  - A great number of Machine Learning algorithms that hae to handle the whole dataset as a block.

23

23

MapReduce

Enrique Alfonseca (Google Research, Zurich):

- Google employs MapReduce in 80% of his process:
  - *Large-scale Web searching indexing*
  - *Clustering problems for Google News*
  - *Reports for popular you want, ie Google Trends*
  - *Processing of satellite imagery data*
  - *Language model processing for statistical machine translation*
  - *Large-scale machine learning problems*

20% remaining? - *Pregel* (design of Google for iterative processes and the rest that is not adapted to MapReduce)

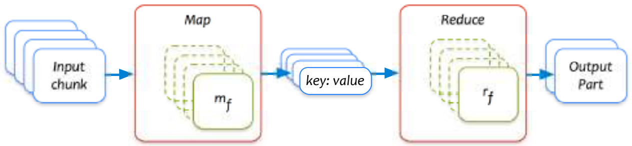
- *Malewicz, G., Austern, M., Bik, TO., Dehnert, J., horn, YO., Leiser, N., and Czajkowski, g. Pregel: TO system for large scale graph processing. ACM SIGMOD 2010.*
- Implementation available in: www.michaelnielsen.org/ddi/pregel *(includes example withPageRank)*

24

24

MapReduce

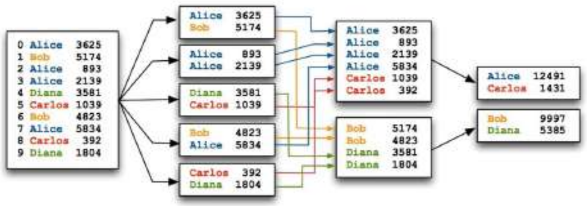
Hadoop



25

MapReduce

Example



26

MapReduce

Mapper

```
public static class MyMapper implements Mapper<LongWritable, Text, Text, IntWritable> {
   private final static IntWritable one = new IntWritable(1);
   private Text word = new Text();
  public void map(LongWritable key, Text value,
                                          OutputCollector<Text, IntWritable> output,
                                          Reporter reporter) throws IOException {

      String line = value.toString();
      StringTokenizer itr = new StringTokenizer(line);
      while (itr.hasMoreTokens()) {
        word.set(itr.nextToken());
        output.collect(word, one);
      }
    }
  }
```

https://developer.yahoo.com/hadoop/tutorial/module4.html

27

MapReduce

Reducer

```
public static class MyReducer implements Reducer<Text, IntWritable, Text, IntWritable> {

  public void reduce(Text key, Iterator<IntWritable> values,
                                          OutputCollector<Text, IntWritable> output,
                                          Reporter reporter) throws IOException {
    int sum = 0;
    while (values.hasNext()) {
      sum += values.next().get();
    }
    output.collect(key, new IntWritable(sum));
  }
}
```

https://developer.yahoo.com/hadoop/tutorial/module4.html

28

g

BIG DATA

ApacheHadoop

1

Hadoop

Index

Topic 4: EcosystemHadoop

Index:

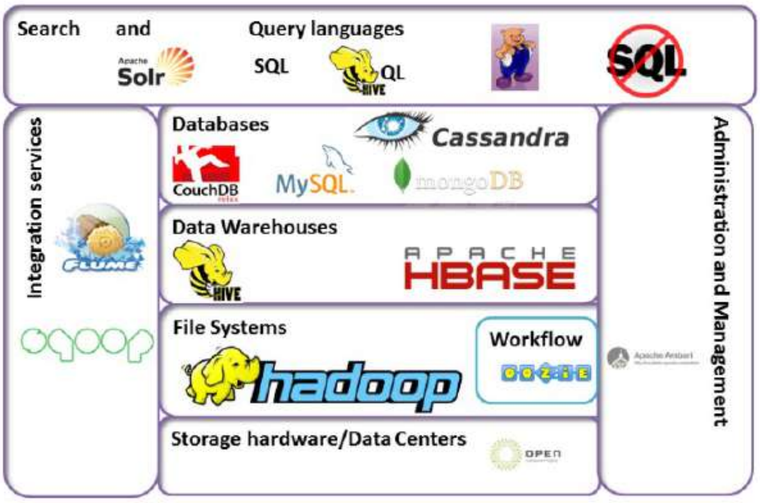
1. Introduction to the Ecosystem Hadoop
2. ApachePig
3. Hive
4. ….

2

2

Hadoop | 1. Introduction

- EcosystemHadoop:

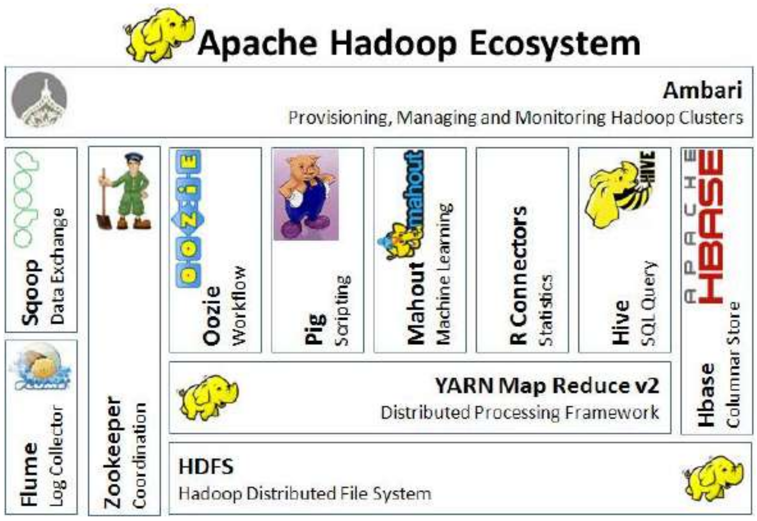


3

3

Hadoop | 1. Introduction

- EcosystemHadoop:



4

4

Hadoop 1. Introduction

- EcosystemHadoop:
  - Apache flume:(Flow Channel)
    - Its function is to upload data files from different applications to HDFS of Hadoop, in order to be able to analyze them.
    - Therefore, it is a distributed java application for gather, and move large amounts of data efficiently from multitude of sources(specialized for*logs*) to a centralized warehouse.
    - It is fault tolerant, and has the appropriate resources for error recovery.

5

5

Hadoop 1. Introduction

- EcosystemHadoop:
  - Apache Hive:(Hive)
    - Is the Data Warehouse of Hadoop, for adding data, perform *queries* tailored, etc., that is, to save us from programming access to files from programming inHadoop, to prepare the data we need.
    - Use the languageHiveQL, which is a substitute for SQL.(It works as if it were a database, withHiveQL, but is not).
    - It works, like so many other applications in the ecosystem Hadoop, in Hadoop, that is, it is a distributed application.
    - You work with it by defining a schema: table with columns with their types, loading data, and subsequently extracting them in the way that suits us.
    - Some distributions of Hadoop already have it installed; if not, it must be downloaded, unzipped, set the environment variable (HIVE_HOME), etc.:

```
exportHIVE_HOME=/usr/local/hive
exportPATH:$PATH:$HIVE_HOME/bin
```

6

6

- EcosystemHadoop:
  - Apache Sqoop:

- It is a tool for transfer data between Hadoop and storage systems structured as relational databases.
- Allows you to import entire databases or individual tables to HDFS
- Allow you to import SQL databases toHive.
- NameSqoopcomes from*SQL-to-Hadoop*
- Among the relational databases it supports areMySQL, Oracle, SQL Server, DB2,PostgresSQL…
- It is managed based on commands:`sqoopC OMMAND [ARGS]`where *command*could be:

codegen	*Generate code to interact with database records*
create-hive-table	*Import a table definition into Hive*
eval	*Evaluate a SQL statement and display the results*
export	*Export an HDFS directory to a database table*
help	*List available commands*
import	*Import a table from a database to HDFS*
import-all-tables	*Import tables from a database to HDFS*
import-mainframe	*Import mainframe datasets to HDFS*
list-databases	*List available databases on a server*
list-tables	*List available tables in a database*
version	*Display version information*

7

7

- EcosystemHadoop:
  - ApachePig:

Source: IBMdeveloperWorks

- Programming tool whose objective is to abstract the programmer from the use of MapReduce for certain things, using a high-level language called *Pig Latin,*which enables express parallel processing of data in the form of a sequence of transformations easy to write, understand and maintain.
- It is ultimately a platform to express analysis of very large data sets along with an infrastructure to evaluate them.
- It integrates a compiler that produces MapReduce program sequences.
- The afore mentioned compiler takes care of optimizing efficiency automatically.
- It allows the programmer to create their own functions to perform specific processing, called UDFs (*User defined functions*), which the programmer writes in Java,Python,javascript, Ruby orgroovy.
- Pig does not have control structures (it does not have conditionals, loops... (although it doesiteratorsabout data)) but is oriented only to data flow.

8

8

BIG DATA Hadoop 1. Introduction

- Ecosystem Hadoop:
  - Apache Pig(II):

IBMdeveloperWorks

- Pig is more performance-oriented than functionality-oriented, to avoid it being complex due to excessive functionality.
- It is called that because of its philosophy: like the*pig*:
  - …eats everything:Pig operates with any type of data, structured,semi-structured or unstructured,
  - …lives anywhere:PigIt is not only aimed at Hadoop although it was initially; now it is oriented towards any parallel processing,
  - … It's a domestic animal:PigIt is made so that its users can easily control, modify and extend it, using the aforementioned UDFs.
- It is used by Yahoo(creator) in at least half of its processes Hadoop; It is also used byLinkedIn, AOL, etc.

9

9

BIG DATA Hadoop 1. Introduction

- What are the most important distributions?

CLOUDERA

HORTONWORKS

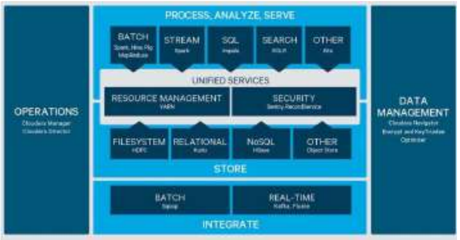
MAPR

10

10

- ## Cloudera

- It was the first distribution on the market.
- It was also the pioneer in creating proprietary components
- There are several distributions within : For example "ClouderaEnterprise" is made up of its own (CDH),**Cloudera Manager** who owns and user supports the components core from CDH
- Its pace of innovation is very strong.
- He has currently partnered with Hortonworks and also gives distributions of this one



- ## MapR

- Focuses on offering maximum performance and fault tolerance
- It has a different architecture, with a more distributed approach that results in performance improvements.
- Own native file system in Unix

Hadoop

1. Introduction

- Hortonworks

HORTONWORKS®

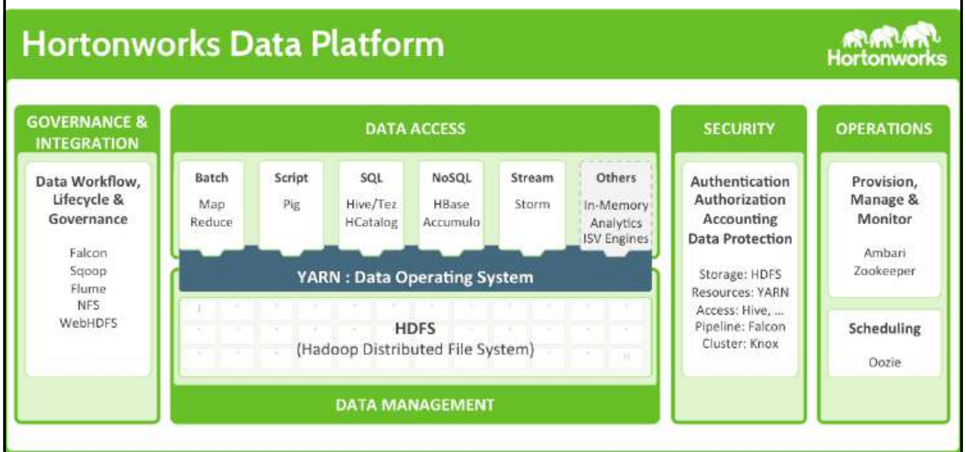
- Philosophy closest to the innovation modelopen source 100%
- First to use Hcatalog for metadata services and optimized the Project Hive, which is the standard used for interactive queries, with the initiative Stinger.
- Use Apache Ambari which is a tool similar to ClouderaManager but it is a non-proprietary software.
- Use a practical “sand-box” with a big number of tutorials.

13

13

Hadoop

1. Introduction



14

14

Hadoop 1. Introduction

- Big Data Suites

- All distributions can run alone or in Big Data Suites
- These integrate them, providing advantages in modeling, code generation, programming Jobs and integration of data from different sources
- They can be open source, such as Talend or Pentaho, or owners
- Most large software companies such as Microsoft, IBM, Oracle,etc, integrate Big Data suites into theirportfolioof software.

pentaho Microsoft IBM talend ORACLE

15

15

Hadoop 1. Introduction

- Which distributions are most important?

	Hortonworks	Cloudera	MapR
Manageability			
Management Tools	Ambari	Cloudera Manager	MapR Control System
Volume Support	No	No	Yes
Heat map, Alarms, Alerts	Yes	Yes	Yes
Integration with REST API	Yes	Yes	Yes
Data and Job Placement Control	No	No	Yes

16

16

Hadoop | 1. Introduction

- Some important tools in the EcosystemHadoop:

  - **Java:**Hadoop's Native Language
  - **Pig:**Query and Workflow Language
  - **Hive:**SQL-Based Language
  - **HBase:**Column-oriented Database forMapReduce

17

17

Hadoop | 1. Introduction

- Some important tools in the EcosystemHadoop:

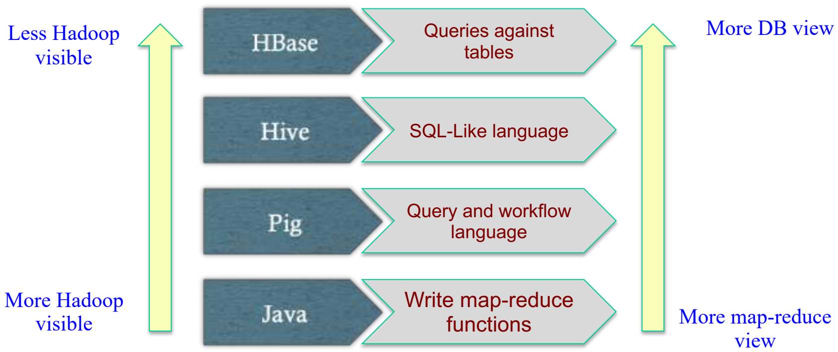
  - **Java:**Is the Native language of Hadoop
    - Hadoop itself is written inJava
    - **Provided Java APIs**
      - For mappers, reducers, combiners,partitioners
      - Formats Input and output
    - Others programming languages ,like, Pig or Hive,converts the queries to code Java & MapReduce

18

18

Hadoop | 1. Introduction

- Some important tools in the EcosystemHadoop:
  - **Levels of Abstraction**

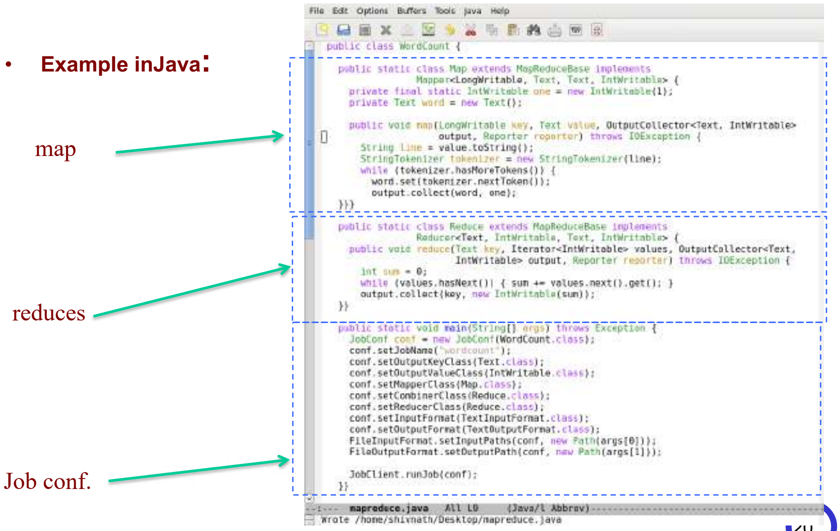


19

19

Hadoop | 1. Introduction

- Some important tools in the EcosystemHadoop:
  - **Example inJava:**




```java
public class WordCount {

  public static class Map extends MapReduceBase implements
                Mapper<LongWritable, Text, Text, IntWritable> {
    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();

    public void map(LongWritable key, Text value, OutputCollector<Text, IntWritable>
                    output, Reporter reporter) throws IOException {
      String line = value.toString();
      StringTokenizer tokenizer = new StringTokenizer(line);
      while (tokenizer.hasMoreTokens()) {
        word.set(tokenizer.nextToken());
        output.collect(word, one);
  }}}

  public static class Reduce extends MapReduceBase implements
                Reducer<Text, IntWritable, Text, IntWritable> {
    public void reduce(Text key, Iterator<IntWritable> values, OutputCollector<Text,
                       IntWritable> output, Reporter reporter) throws IOException {
      int sum = 0;
      while (values.hasNext()) { sum += values.next().get(); }
      output.collect(key, new IntWritable(sum));
  }}

  public static void main(String[] args) throws Exception {
    JobConf conf = new JobConf(WordCount.class);
    conf.setJobName("wordcount");
    conf.setOutputKeyClass(Text.class);
    conf.setOutputValueClass(IntWritable.class);
    conf.setMapperClass(Map.class);
    conf.setCombinerClass(Reduce.class);
    conf.setReducerClass(Reduce.class);
    conf.setInputFormat(TextInputFormat.class);
    conf.setOutputFormat(TextOutputFormat.class);
    FileInputFormat.setInputPaths(conf, new Path(args[0]));
    FileOutputFormat.setOutputPath(conf, new Path(args[1]));

    JobClient.runJob(conf);
  }}
```

20

20

Hadoop | Index

Topic 4: Ecosystem Hadoop

Index:

1. Introduction to the Ecosystem Hadoop
2. Apache Pig
3. Apache Hive
4. ….

21

21

Hadoop | 2.ApachePig

2.ApachePig

- Let's remember……What isPig:
  - A platform for analyze big datasets that consists in the utilization of a High level language for the realization analysis on data.
  - Compile down to MapReduce jobs
  - Developed by Yahoo!
  - Open-source language

22

22

2.ApachePig

- Components inPig:

Two Main Components →

- Language High-Level(Pig Latin)
  - Set of commands
- Two modes execution
  - Local: reads/writes to local file system
  - Mapreduce:Connect to Hadoop Clusters and reads/writes to HDFS

TwoModes →

- Mode Interactive
  - Console
- Batch mode
  - Submit to script

23

23

2.ApachePig

- Why Pig?.....Abstraction:

  - Common design patterns as keywords (joins, distinct, counts)
  - Analysis of data flow (Data flow analysis)
    - the script can be assigned to multiple jobs map-reduce
  - Avoid errors at Java level (not everyone can write Java code)
  - Able to work in interactive mode
    - Run commands on the go and get results!

24

24

2.ApachePig

IBMdeveloperWorks

- Relational operators:

Operator	***Description***
DISTINCT	Remove duplicates in a relationship
FILTER	Selects a set oftuplesof a relationship based on a condition
FOREACH	Iterates the tuples of a relationship, generating a new set of data
GROUP	Groups data into one or more relationships
JOIN	Joins two or more relationships (exists both *inner join and outer join*)
LIMIT	Sets the limit of output tuples
LOAD	Load data from a file system
ORDER	Orders a relationship based on one or more fields
SPLIT	Splits a relationship into one or more relationships
STORE	Stores transformed information in the file system

- Diagnostic operators*(debugging scripts)*:

Operator	***Description***
DUMP	Displays the content of a relationship on the screen
DESCRIBE	Displays on the screen the detailed diagram of a relationship: field and type
EXPLAIN	Shows on the screen how operators are grouped in MapReduce processes

2.ApachePig

- Pig Latin is procedural (data flow programming model)
  - The step-by-step query style is much cleaner and easier to write
- SQL is declarative but not step by step

SQL

```
insert into ValuableClicksPerDMA
select dma, count(*)
from geoinfo join (
                select name, ipaddr
                from users join clicks on (users.name = clicks.user)
                where value > 0;
            ) using ipaddr
group by dma;
```

Pig Latin

```
Users                = load 'users' as (name, age, ipaddr);
Clicks               = load 'clicks' as (user, url, value);
ValuableClicks       = filter Clicks by value > 0;
UserClicks           = join Users by name, ValuableClicks by user;
Geoinfo              = load 'geoinfo' as (ipaddr, dma);
UserGeo              = join UserClicks by ipaddr, Geoinfo by ipaddr;
ByDMA                = group UserGeo by dma;
ValuableClicksPerDMA = foreach ByDMA generate group, COUNT(UserGeo);
store ValuableClicksPerDMA into 'ValuableClicksPerDMA';
```

Hadoop 2.ApachePig

2.ApachePig

- **In Pig Latin**
  - Assessment deferred(Lazy) (data not processed before the STORE command)
  - The data can be stored in any spot during the pipeline
  - The scheme and the type of data are defined in execution time
  - An execution plan can be explicitly defined
    - By means of suggestions of the optimizer
    - Due to the lack of complex optimizers
- **InSQL:**
  - The consultation plans are determined only by the system
  - The data can´t be stored in middle of the execution
  - The scheme and the type of data are defined in creation time

27

27

Hadoop 2.ApachePig

- EcosystemHadoop:
  – ApachePig(VI):

IBMdeveloperWorks

- Advantages of Pig:
  – Use of relational operators such as JOIN, FILTER, GROUP BY, etc., whose implementation in MapReduce is expensive in lines of code.
  – Depuration.
  – Optimization of data flows.
  – PigIt can be used as a simple ETL that works in parallel automatically, (although it is not a true ETL).
- Disadvantages:
  – Certain algorithms can be complicated to write in Pig.
  – If you need very optimal code, writing it directly in MapReduce may be the only choice

28

28

BIG DATA | Hadoop | Index

Topic 4: Ecosystem Hadoop

Index:

29

29

BIG DATA | Hadoop | 3. ApacheHive

Motivation

- Yahoo made Pig for ease the development of Applications developed in Hadoop.
  - Its main need was around non structured data
- Simultaneously Facebook began working in the development of datawarehouse in Hadoop giving as result Hive.
  - They where focused on structured data

30

30

Characteristics

- A data warehouse infrastructure built on top of Hadoop for supplying “data summary, query, and analysis”
- **Hive Provides**
  - ETL (Extract, Transform, Load)
  - Structure
  - Access to different types of storage(HDFS or HBase)
  - Execution of Queries via MapReduce
- **Key Building Principles**
  - SQL is a known yet simple language
  - Extensibility– Types, Functions, Formats, Scripts
  - Performance

Hivedeals with structured data

- Tables
  - Analog to relational databases tables
  - Each table has a directory correspondent in HDFS
  - Are Data serialized and stored as files inside of that directory
  - Hive has a serialization built-in default that supports compression and deserializationlazy (lazy)
  - Users can specify serialization custom – schemes

Examples Commands inHive(HiveDDL)

- **CREATE TABLE**sample (foo INT, bar STRING)**PARTITIONED BY**(ds STRING);
- **SHOW TABLES**'.*s';
- **DESCRIBE**sample;
- **ALTER TABLE**samples**ADD COLUMNS**(new_colINT);
- **DROP TABLE**sample;

A table and in Hive is a directory HDFS in Hadoop

```
create table table_name (
  id              int,
  dtDontQuery     string,
  name            string
)
partitioned by (date string)
```

The Schema isknown in timeofcreation (asaschemeby BD)

Theboards Partitioned have“sub-directories”, one by eachpartition

Advantages

- Good for Analysis
- Low Learning Curve
- Completely transparent to Map-Reduce jobs
- Partitions (speed!)
- Flexibility for load data fromLocal SF/HDFS in Hive tables

Data Science

1

KDD

Goals

- Know the fundamentals of the Mining of Data, as discipline that contains the beginning for the obtaining of knowledge in setsof data "*conventional*"
- Know the aim and relevance of each stage (KDD)

2

2

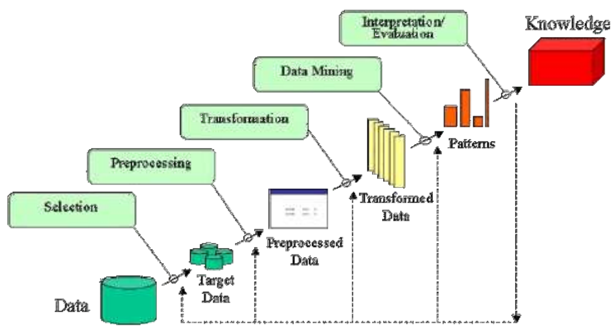
KDD

- Mining of Data (*Data Mining*):
  - Is a stage of the process of ***Knowledge discovery in Databases***, that treats about find patterns in sets of data that be **interpretable** and **tools** (prediction, take of decisions, etc.).
  - For it employs the resources of the areas of Sciences of the computing and Intelligence Artificial (Learning Automatic (*machine Learning*), methods of search, etc.) and of the Statistics.
  - **knowledge** :
    - Valid (general: applicable to futures data)
    - Novel
    - Potentially useful (for example, take a decision from it)
    - Understandable (in he sense of Interpretable) by he user
  - It treats of a discipline "mature", evolved from decades, that offers good results

3

3

KDD



* https://www2.cs.uregina.ca/~dbd/cs831/notes/kdd/1_kdd.html

4

4

KDD

Phases / etages

- Integration: comprehension and familiarization with he domain of the application, ID of the sources of data and creation of the repository initial with the different sources
- Sampling and selection: choose he subset of data appropriate for the study
- Preprocessing: Cleaning ,wrong data, missing values, noise ,transformation to adapt to Learning model
- Mining of data (properly) or modeling: selection and use of the algorithms according to he aim of the study
- Assessment, creation of reports and display: validation of the results obtained and report elaboration, presentations graphs, etc.,to clarify and present the obtained knowledge

5

5

KDD

- A bit of organisation:

1.KDD
2.Definitions about the data
3.Preprocessing
4.Mining of Data

6

6

KDD

- **Domain**: set of values of the same instance; they can be nominal or numerical (*discreet*: integer, *continuous*: real). Examples:
  - nominal: *high, half, low, ...*
  - numeric:
    - discreet: *1, 2, ...*
    - continuous: *1.2, 3.1426, ...*
  - **Attribute** (X_{Y_0}): each characteristics (either variable) to analyze that take their values in a certain domain
- **Instance** or example: tuple of values (x1, x2, ..., xn) where each value corresponds to an attribute of a single individual in the study

7

7

KDD Preprocesing

- Treatments to adapt and cure the data in order to mine it.
- Need / Importance: In Mining of Data, is ***critical*** to use the set of adequate data to the machine Learning algorithm
- What happends with a bad preprocessing?
  - Lost of calculation time : inefficiency of the algorithms, waste of energy...
  - Lack of precision, useless results and obvious outcomes
  - Sub-Stages:
    - Integration and Collection from the different sources
    - Cleaning, Transformation and Selection

8

8

BIG DATA — KDD

Cleaning, Transformation and Selection

- The **cleaning** of the data mostly consists in:
  - Treatment of values missing
  - Treatment of mistakes and values anomalous
- The **transformation** fundamentally consists in:
  - Standardization
  - Discretization
  - Transformations of attributes
- The **selection** basically consists in:
  - Selection of characteristics
  - Selection of instances

9

9

BIG DATA — KDD

Cleaning: Treatment of missing values:

- Ignore: some algorithms they can be robust to are absences (because when arise, they can ignore them, e.g. trees).
- Filter the attribute (eliminate / replace): extreme solution;
- Filter the instances (delete them): could skew the data, because many times the Causes of afact that lack are related with cases or special ones
- Replace by other value: by estatistics (attributes numeric) , mean, mode (attribute nominal). Or even can be *predicted based on the rest* of the data

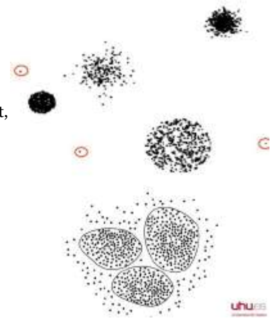
10

10

KDD

Cleaning: Treatment of anomalous values (*outliers*):

- Could be correct values, but **significantly different** to the rest (statistically) or noise
- They impact alot on methods based in wheight, the Artificial Neuronal Networks
- How to detect them:
  - Average distanves
  - Clustering
  - Smothing
  - Regression



11

11

KDD

Transformation: Standardization:

- Consists in carring out a change of scale in a attribute
- Necessary when HE have attributes with scales different and he algorithm is sensitive to this scale
- The standardization further traditional is normalize in he interval [0,1] of the variables

$$v' = \frac{v - \min}{\max - \min}$$

12

12

Transformation: Discretization and Numeration**: Discretization**

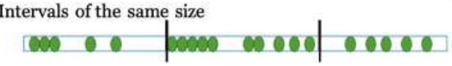
- It is about to find a set of intervals;
- sometimes known as *bin* (➔*binning*)
- Is necessary while many algorithms of Mining of Data can only work with discreet attributes
- For example, a common case is the discretization of the calificactions, that is a worth numeric betweeno and 10, that is discretized:

  Fail [0-5)
  Approved [5,7)
  Remarkable [7,9)
  Outstanding [9,10]

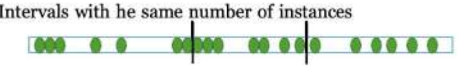
Transformation: Discretization and Numeration: Discretization (III)

Example of the discretization of the same set with different beginning:

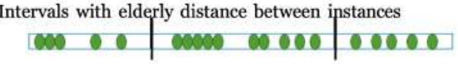
- Intervals of the same size



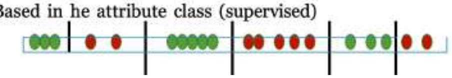
- Intervals with he same number of instances



- Intervals with elderly distance between instances



- Based in he attribute class (supervised)



KDD

- Considerations about the Models of Mining of Data:
  - They can trigger Models interpretable either Models No interpretable:
    - Interpretable, either understandable by the humans in how much to his formulation of theknowledge (rules, trees...), ID3, C4.5, CART, K-means, Networks Bayesian,...
    - No interpretable, either Models of box black whose knowledge learned No isunderstandable by the humans, such as the Networks Neuronal artificial, Machinesof Medium Vector, K-NN, ...

15

15

KDD

Types of learning

Supervised Learning:

- Learns from labeled data.
- Goal: Make predictions or classify data.
- Examples: Classification, regression.
- Evaluation: Accuracy of predictions.

Models: Decission Trees, Nuronal Networks, Rules

Unsupervised Learning:

- Learns from unlabeled data.
- Goal: Discover hidden patterns or structures.
- Examples: Clustering, dimensionality reduction.
- Evaluation: Uncover meaningful insights.
- Models: Clusters, association rules, instances

Key Differences:

Supervision: Labeled vs. unlabeled data.
Goal: Prediction vs. pattern discovery.
Use Cases: Predictive tasks vs. exploring data.

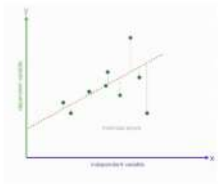
16

16

KDD

- **Predictive Models Regression:**
  - Learn from set of examples with one or more outputs
  - They get a ***model*** that it serves for estimate values that where not found originally in the set of examples
  - Measure : MSE

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

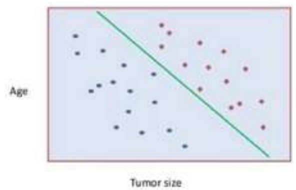


17

17

KDD

- **Predictive Models Classification:**
  - Learn from set of examples with one labaled class in categories
  - They get a ***model*** that it serves for estimate clases that where not found originally in the set of examples
  - Measure : Accuracy, precisión Recall, Roc ...



18

18

KDD

Confusion Matrix and measures

		Real Label	
		Positive	Negative
Predicted Label	Positive	True Positive (TP)	False Positive (FP)
	Negative	False Negative (FN)	True Negative (TN)

$$\text{Precision} = \frac{\sum TP}{\sum TP + FP}$$

$$\text{Recall} = \frac{\sum TP}{\sum TP + FN}$$

$$\text{Accuracy} = \frac{\sum TP + TN}{\sum TP + FP + FN + TN}$$

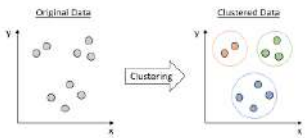
19

19

KDD

Clustering:

Clustering is an unsupervised learning task that **groups data points into clusters based on similarity**.
It doesn't require labeled data and aims to uncover hidden patterns within the dataset.
Clustering algorithms assign data points to clusters based on **distance or density measures**. Popular algorithms include **K-means** and **hierarchical clustering**



Evaluation:

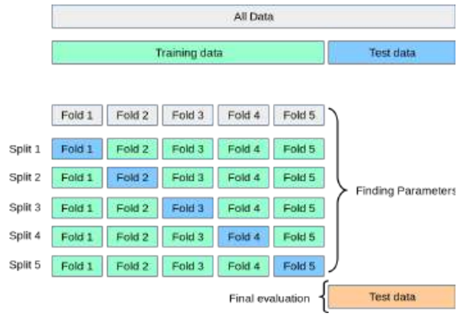
Silhouette Score: Rates how well objects fit within their clusters.
Davies–Bouldin Index: Measures cluster compactness and separation.
Dunn Index: Assesses cluster compactness relative to separation.

20

20

KDD | Data Shift

Cross Fold Validation

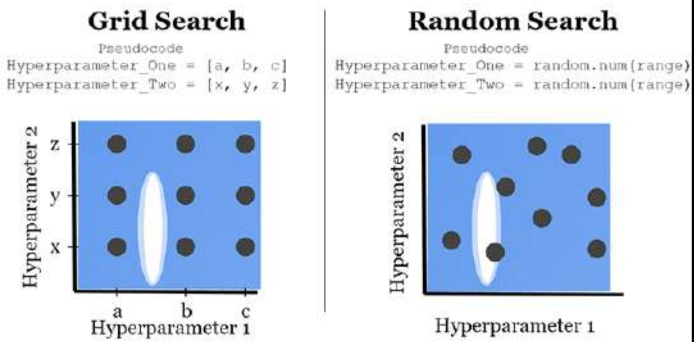


21

21

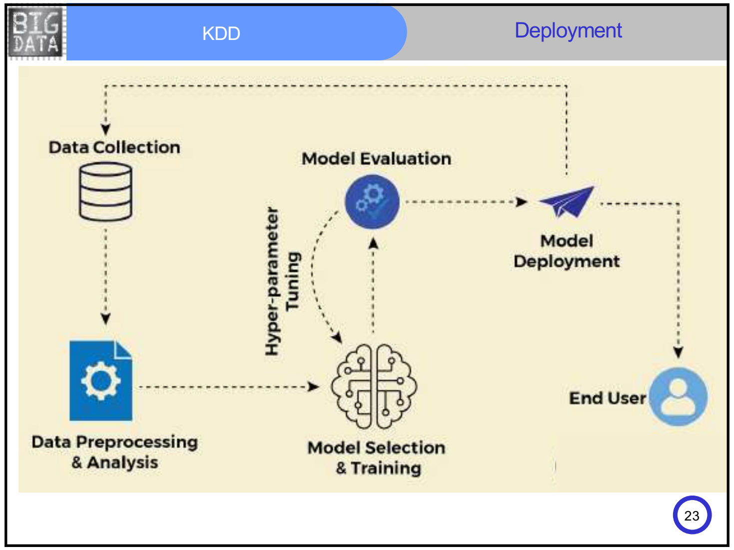
KDD

Parameter Tuning: making the model work



22

22


KDD
Deployment
Data Collection
Model Evaluation
Hyper-parameter Tuning
Model Deployment
End User
Data Preprocessing & Analysis
Model Selection & Training
23

23

BIG DATA

Spark:*SparkML*

1. Introduction

Can define ***Big Data*** as a technology complete able to handle much of the **storage** and processing of **big volumes** of data.

- Complex and unstructured of diverse sources generated athigh speeds.
- **Stored** in distributed nodes.
- **Processed** in parallel.
- **Process move to the data and not the data to the process**

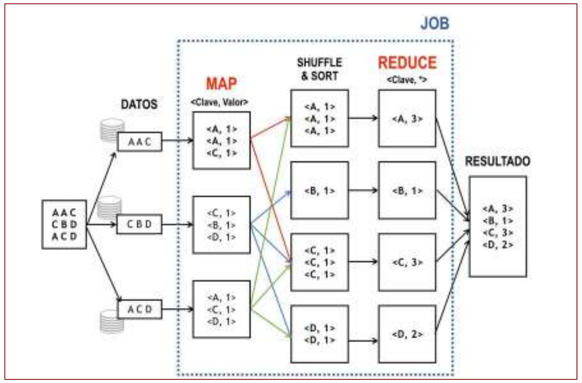
1. Introduction

So, arise ***Frameworks*** that facilitate the implementation of solutions of Big Data:

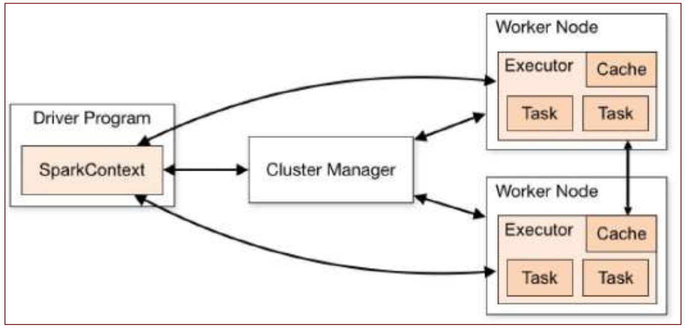
- **Hadoop**: Framework of Big Data based in the implementation of the paradigm MapReduce. Provides **scalability horizontal**. It allows he storage distributed and he prosecution parallel of big volumes of data. Uses as architecture a cluster of servers.
- **Spark**: Framework of Big Data that aims to sort out the issues of Hadoop. Offers a flow of processes more flexible disengaging the functions Reduce of the Map. Besides, implements the procesing in memory.

1. Introduction

MapReduce

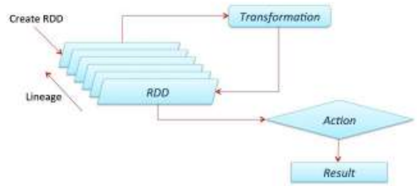


1. Introduction



Operations about RDDs:

- Creation: for generate them
- Action: makes a *operation* about they and gets a result
- Transformation: starting off of a RDD, it process for obtain a new RDD different, butbasedin he original



…futher forward we will see he code and details for each operation.

Spark

- Around of prosecution for Big Data of guy *open source*
- Their goalsare:
  - Better performance: speed
  - Ease of use
  - Ability for carry out sophisticated analysis

Matei Zaharia

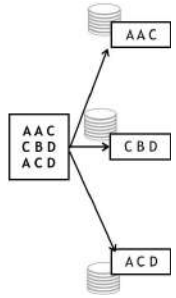
- Original of the AMPLab of the University of California in Berkeley (2009) as Project of investigation; converted in *open source* in 2010, for the project of *Apache Software Foundation* in 2013

7

1. Introduction

the **data bstructure** on Spark are the ***Resilient Distributed Dataset (RDD)***.

- Structure of data in memory.
- Data distributed,divided in partitions in different nodes
- Processed in parallel, in different nodes
- Operated through Transformations and Actions.
- Tolerant to failures.
- Lazy evaluation

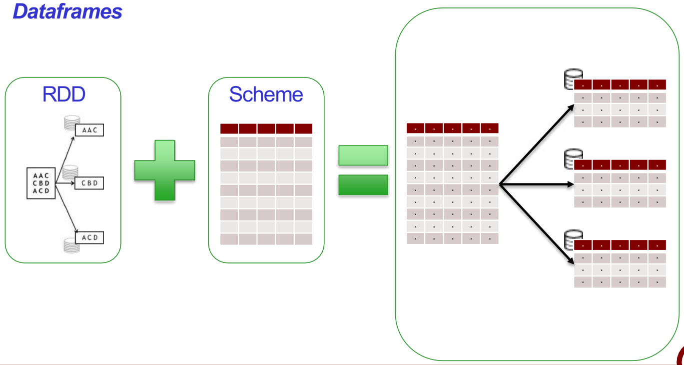


8

1. Introduction

Similar to Relational tables. SQL language, already in Pandas.

Dataframes



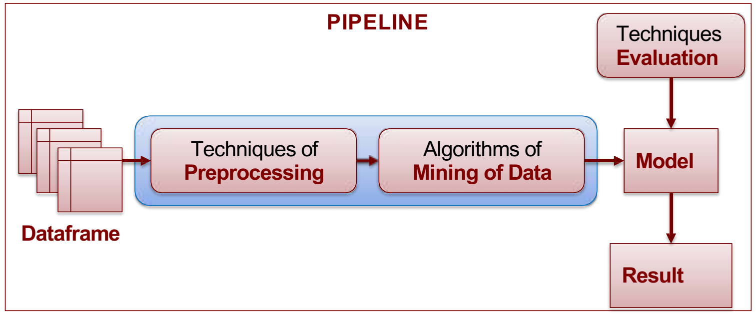
4. SparkML

The **SparkML** us offers a great variety of algorithms on eacb of the stages of the KDD **implemented** in an **scalable platform**

- **Preprocessing: Extraction**, **transformation,** and **selection**
- **Mining of data**
  - **Learning Supervised**
    - **Regression**
    - **Classification**
  - **Learning No Supervised**
    - **Clustering**
- **Evaluation** of the obtained **Models**.

4. SparkML

A **pipeline** is a way of representing the process of D.M. through the concatenation of several **phases** that will performed secuentially.



4. SparkML

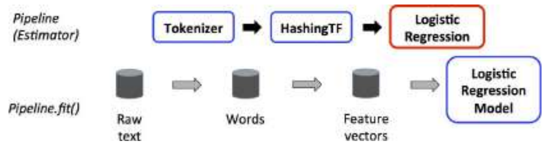
The makn components of the **pipeline** are:

- **Dataframe**: Structure of data that will store our set ofdata. Usually we will use the columns "features", "labels" and"predictions".
- **Transformers**: Algorithm that transform a Dataframe in another. A model of M.L. obtained is a transformer that transform a dataframewith "features" in a dataframe with "predictions".
- **Estimators**: Algorithm that applied over a Dataframe generates a Transformer. An ML algorithm is a estimator that is trained abouthe Dataframe and generate a model of ML.
- **Pipeline**: Sequence of Transformers and Estimators that specify a ML workflow .

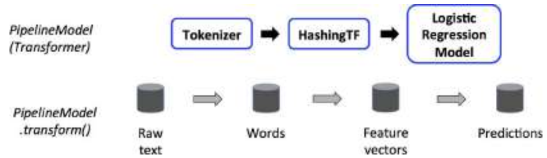
4. SparkML

Execution of the **pipeline**:

- **fit** (Training/Train/Learning):



- **transform** (Test/Test/Inference):

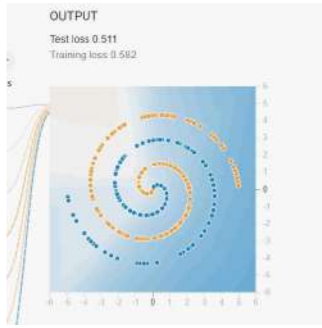


13

4. SparkML

Algorithms of **Classification**:

- **Logistics regression**
- **Decision tree classifier**
- **Random forest classifier**
- **Gradient-boosted treeclassifier**
- **Multilayer perceptron classifier**
- **Linear Support Vector machine**
- **One-vs-Rest classifier (One-vs-All)**
- **Naive Bayes**
- **Factorization machines classifier**

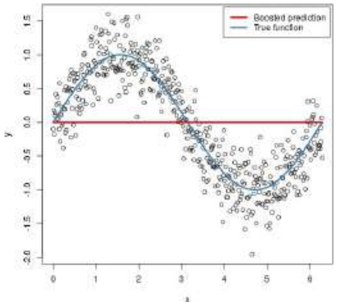


14

4. SparkML

Algorithms of **Regression**:

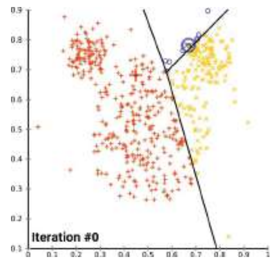
- **Linear regression**
- **Decision tree regression**
- **Random forest regression**
- **Gradient-boosted treeregression**
- **Survival regression**
- **Isotonic regression**
- **Factorization machines regressor**



4. SparkML

Algorithms of **Clustering**:

- **K-means**
- **Latent Dirichlet allocation (LDA)**
- **Bisecting k-means**
- **Gaussian Mixture model (GMM)**
- **power Iteration Clustering (PIC)**



4. SparkML

Techniques of **Evaluation**:

- **REGRESSION:**

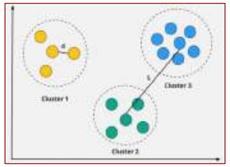
$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

- **CLASSIFICATION:**

$$Accuracy = \frac{Number\ of\ Correct\ Predictions}{Total\ Numeber\ of\ Predictions}$$

		Actual Values	
		Positive (1)	Negative (0)
Predicted Values	Positive (1)	TP	FP
	Negative (0)	FN	TN

- **CLUSTERING:**



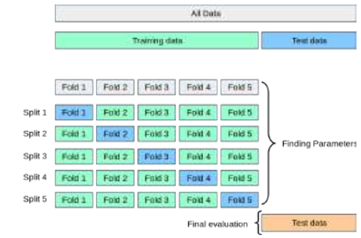
1. **Silhouette**
2. **within Set Sum of squared error**

17

4. SparkML

Methods of **Assessment**:

- **Hold-Out**: Consists in spliting the set of data (Dataset) in two independent sets, set of training (Training Set) and set of test(Test Set).(75%-25%)
- **Cross-Validation**: Consists in carring out several Hold-Out, is say, applies the algorithms over different partitions of Training/Test Sets (Same proportion,but different instances).



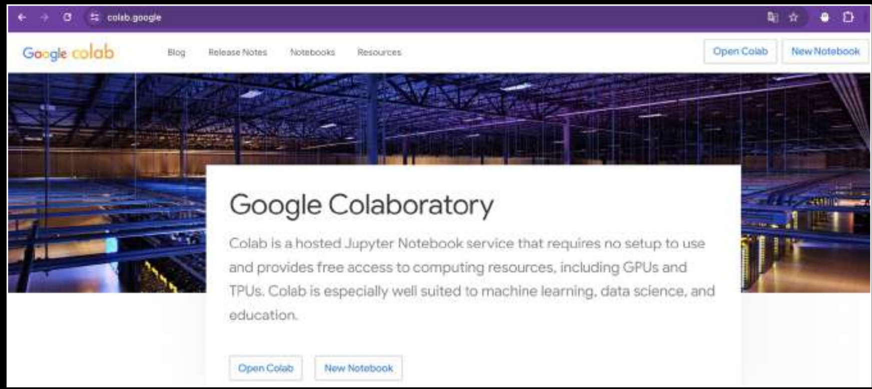
18

1

Python

- Python has emerged as a leading language for machine learning due to its simplicity, flexibility, and extensive ecosystem of libraries and tools.
- Several Python libraries are widely used in machine learning projects, offering various functionalities for data manipulation, preprocessing, modeling, and evaluation.
  - Numpy
  - Pandas
  - Sklearn
  - Matplotlib and seaborn

2



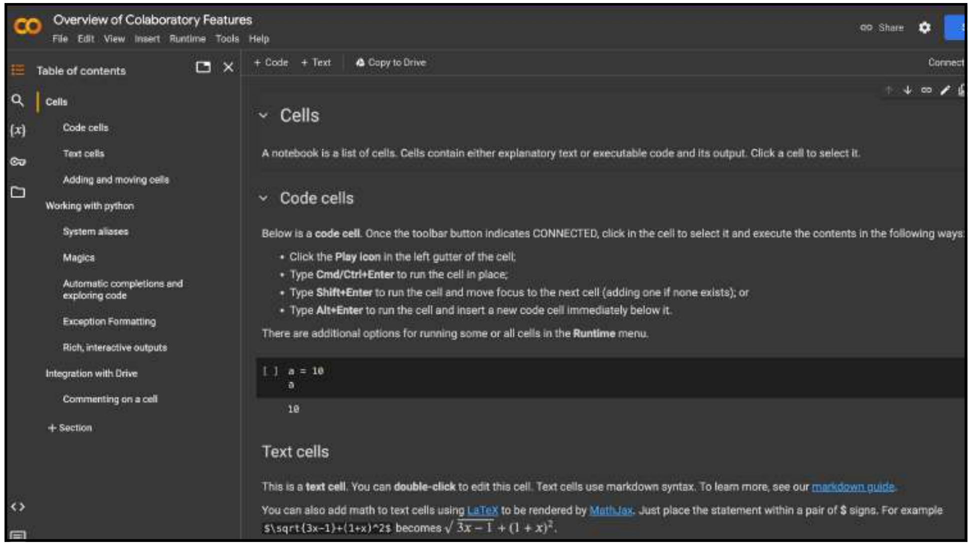
Google Colab

3

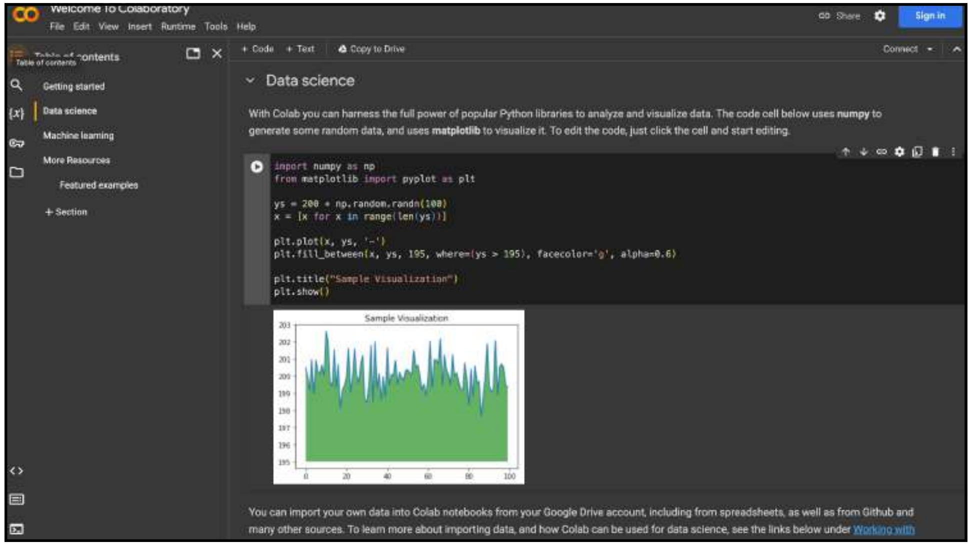
How to set up a Google colab account and use a Jupyter notebook

- **Access**: Go to https://colab.research.google.com/ and sign in with your Google account.
- Create Notebook: Click "File" > "New Notebook" to create a new notebook.
- Code Execution: Write and execute Python code in cells. Use **Shift + Enter** to run a cell.
- Text Cells: Add text cells for documentation using **Markdown syntax.**
- **Upload/Download Files**: Upload files using the file upload button. Download files using the download button.
- Sharing: Share notebooks with others for collaboration.
- Jupyter notebooks are interactive documents combining **live code, equations, visualizations, and text explanations**. They're widely used in data science and research for their flexibility and support for various programming languages through kernels.

4


Overview of Colaboratory Features
File Edit View Insert Runtime Tools Help
Share
Table of contents
+ Code + Text
Copy to Drive
Connect
Cells
Code cells
Text cells
Adding and moving cells
Working with python
System aliases
Magics
Automatic completions and exploring code
Exception Formatting
Rich, interactive outputs
Integration with Drive
Commenting on a cell
+ Section
Cells
A notebook is a list of cells. Cells contain either explanatory text or executable code and its output. Click a cell to select it.
Code cells
Below is a code cell. Once the toolbar button indicates CONNECTED, click in the cell to select it and execute the contents in the following ways:
Click the Play icon in the left gutter of the cell;
Type Cmd/Ctrl+Enter to run the cell in place;
Type Shift+Enter to run the cell and move focus to the next cell (adding one if none exists); or
Type Alt+Enter to run the cell and insert a new code cell immediately below it.
There are additional options for running some or all cells in the Runtime menu.
a = 10
a
10
Text cells
This is a text cell. You can double-click to edit this cell. Text cells use markdown syntax. To learn more, see our markdown guide.
You can also add math to text cells using LaTeX to be rendered by MathJax. Just place the statement within a pair of $ signs. For example
$\sqrt{3x-1}+(1+x)^2$ becomes

5


Welcome To Colaboratory
File Edit View Insert Runtime Tools Help
Share
Sign in
Table of contents
+ Code + Text
Copy to Drive
Connect
Getting started
Data science
Machine learning
More Resources
Featured examples
+ Section
Data science
With Colab you can harness the full power of popular Python libraries to analyze and visualize data. The code cell below uses numpy to generate some random data, and uses matplotlib to visualize it. To edit the code, just click the cell and start editing.
import numpy as np
from matplotlib import pyplot as plt
ys = 200 + np.random.randn(100)
x = [x for x in range(len(ys))]
plt.plot(x, ys, '-')
plt.fill_between(x, ys, 195, where=(ys > 195), facecolor='g', alpha=0.6)
plt.title("Sample Visualization")
plt.show()
Sample Visualization
You can import your own data into Colab notebooks from your Google Drive account, including from spreadsheets, as well as from Github and many other sources. To learn more about importing data, and how Colab can be used for data science, see the links below under Working with

6

Python basics

```python
# Variables
x = 5
y = "Hello"
z = 3.14

# Data Types
print(type(x))  # Output: <class 'int'>
print(type(y))  # Output: <class 'str'>
print(type(z))  # Output: <class 'float'>
```

```
<class 'int'>
<class 'str'>
<class 'float'>
```

7

Python Operations

```python
a = 10
b = 5
print(a + b)  # Addition: Output -> 15
print(a - b)  # Subtraction: Output -> 5
print(a * b)  # Multiplication: Output -> 50
print(a / b)  # Division: Output -> 2.0 (Python 3.x)
print(a // b) # Floor Division: Output -> 2 (Python 3.x)
print(a % b)  # Modulus: Output -> 0
print(a ** b) # Exponentiation: Output -> 100000
```

```
15
5
50
2.0
2
0
100000
```

8

Python Conditions

```python
# If statement
x = 10
if x > 5:
    print("x is greater than 5")
elif x == 5:
    print("x is equal to 5")
else:
    print("x is less than 5")

# Output: x is greater than 5
```

```
x is greater than 5
```

9

Python loops , lists, functions

```python
for i in range(5):
    print(i)

# Output: 0, 1, 2, 3, 4

# While loop
count = 0
while count < 5:
    print(count)
    count += 1
```

```
0
1
2
3
4
0
1
2
3
4
```

```python
my_list = [1, 2, 3, 4, 5]
print(my_list[0])  # Output: 1
print(my_list[-1]) # Output: 5
print(len(my_list)) # Output: 5
```

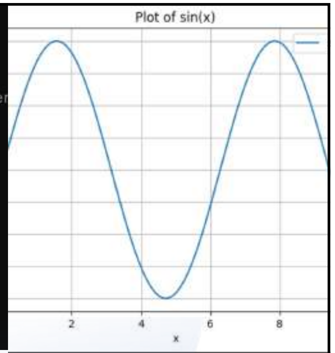
```
1
5
5
```

```python
# Function definition
def greet(name):
    print("Hello, " + name + "!")

# Function call
greet("Alice")  # Output: Hello, Alice!
```

```
Hello, Alice!
```

10

```python
import matplotlib.pyplot as plt

# Generate data
x = np.linspace(0, 10, 100)  # Create 100 evenly spaced points between
y = np.sin(x)                 # Compute the sine of each point

# Plot data
plt.plot(x, y, label='sin(x)')  # Plot x vs y with label 'sin(x)'
plt.xlabel('x')                  # Set the x-axis label
plt.ylabel('sin(x)')             # Set the y-axis label
plt.title('Plot of sin(x)')      # Set the title of the plot
plt.legend()                     # Show legend
plt.grid(True)                   # Show grid
plt.show()                       # Display the plot
```



Numpy & matplotlib :

11

Pandas and Seaborn graphics

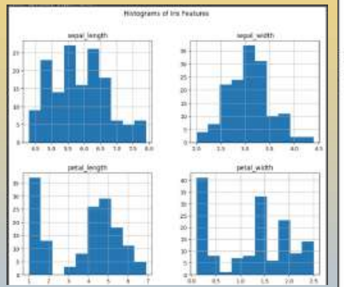
```python
import pandas as pd
import matplotlib.pyplot as plt
# Load Iris dataset from seaborn library
import seaborn as sns
iris = sns.load_dataset('iris')
# Display the first few rows of the dataset
print("First few rows of the Iris dataset:")
print(iris.head())
# Summary statistics
print("\nSummary statistics of the Iris dataset:")
print(iris.describe())
# Number of samples for each species
print("\nNumber of samples for each species:")
print(iris['species'].value_counts())
# Plot histograms for each feature
iris.hist(figsize=(10, 8))
plt.suptitle('Histograms of Iris Features')
plt.show()
# Plot scatter matrix
sns.pairplot(iris, hue='species', height=2.5)
plt.suptitle('Scatter Matrix of Iris Features')
plt.show()
```

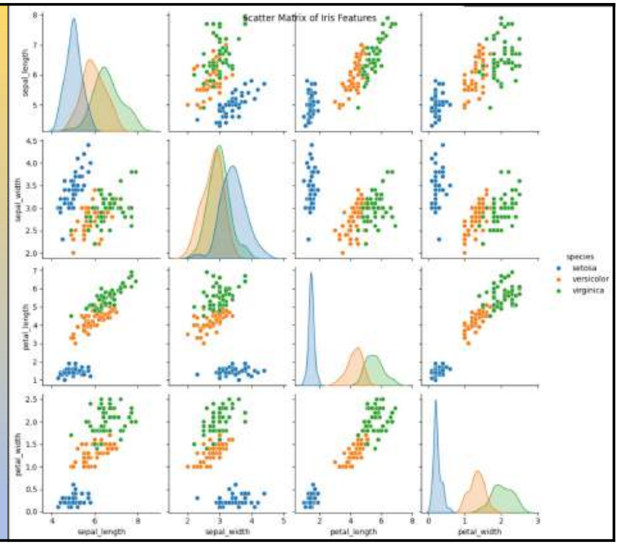
- **Iris dataset** is a classic dataset in machine learning and is often used for learning and demonstration purposes. It contains measurements of four features (sepal length, sepal width, petal length, and petal width) of three species of iris flowers (Setosa, Versicolor, and Virginica).:

```
First few rows of the Iris dataset:
   sepal_length  sepal_width  petal_length  petal_width species
0           5.1          3.5           1.4          0.2  setosa
1           4.9          3.0           1.4          0.2  setosa
2           4.7          3.2           1.3          0.2  setosa
3           4.6          3.1           1.5          0.2  setosa
4           5.0          3.6           1.4          0.2  setosa

Summary statistics of the Iris dataset:
       sepal_length  sepal_width  petal_length  petal_width
 unt     150.000000   150.000000    150.000000   150.000000
 ean       5.843333     3.057333      3.758000     1.199333
std        0.828066     0.435866      1.765298     0.762238
min        4.300000     2.000000      1.000000     0.100000
25%        5.100000     2.800000      1.600000     0.300000
50%        5.800000     3.000000      4.350000     1.300000
75%        6.400000     3.300000      5.100000     1.800000
max        7.900000     4.400000      6.900000     2.500000

Number of samples for each species:
setosa        50
versicolor    50
virginica     50
Name: species, dtype: int64
```

12

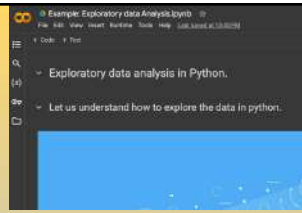
Seaborn





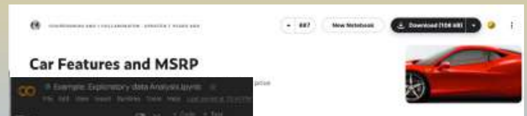
13

Dataframes: Born to EDA

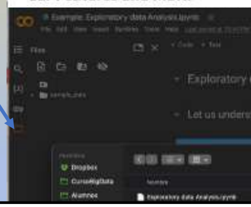


- Open https://shorturl.at/gDJQ1
- Download the dataset in (is also in the notebook):

https://www.kaggle.com/datasets/CooperUnion/cardataset



Upload it to Google colab



14

ChatGPT comes in your help

ChatGPT is a chatbot developed by OpenAI and launched on November 30, 2022. Based on a large language model, it enables users to refine and steer a conversation towards a desired length, format, style, level of detail, and language. Wikipedia

Developer: OpenAI, Microsoft Corporation

Initial release date: November 30, 2022

Engine: GPT-3.5 (free and paid); GPT-4 (paid only)

Do we need to write every single line of code or…. Not?

15

Lets do another EDA … by ChatGPT

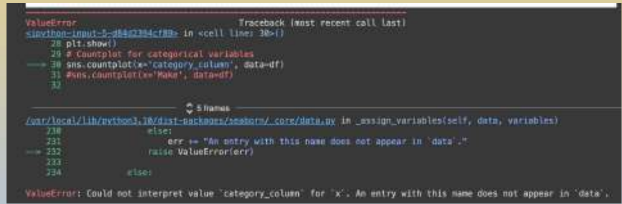
- Write a promp: I want to make an EDA with the dataset data.csv in Python
- Copy the code and paste into a cell of a Google colab new notebook
- Load the data.csv
- Run it….

```
rt necessary libraries
rt pandas as pd
port matplotlib.pyplot as plt
port seaborn as sns
 Load the dataset
df = pd.read_csv('data.csv')
# Display the first few rows of the dataset to get an overview
print(df.head())
# Get summary statistics
print(df.describe())
# Check for missing values
print(df.isnull().sum())
# Visualize the distribution of numerical features
df.hist(bins=20, figsize=(12, 10))
plt.show()
# Correlation matrix heatmap
correlation_matrix = df.corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Matrix')
plt.show()
# Pairplot to visualize relationships between numerical features
sns.pairplot(df)
plt.show()
# Boxplot to identify outliers
plt.figure(figsize=(12, 8))
sns.boxplot(data=df)
plt.title('Boxplot of Numerical Features')
plt.show()
# Countplot for categorical variables
sns.countplot(x='category_column', data=df)
plt.title('Countplot of Categorical Variable')
lt.show()
```

16

I would like some thing Else : Talk to chatGPT

- Its not perfect. Some times you will have errors, so some fixing need to be done:

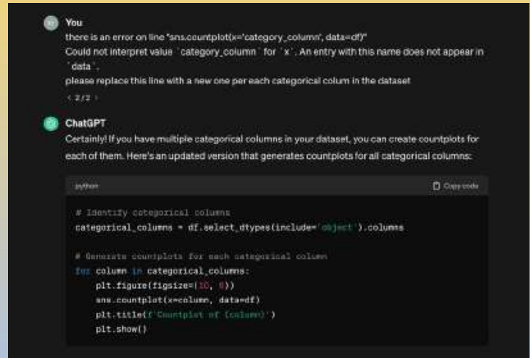


- And the categorical columns have to be written by and or… ask again Chatgpt

17

Lets make ChatGPT fix the problem

- Write the prompt:



18

ARR Metodology ;)

Ask what you want
Refine
Repeat

- Now we want …

You
print how many different MAke there are in the dataser grouping by model

ChatGPT
Certainly! If you want to count the number of different "Make" values in your dataset, grouped by the "Model" column, you can use the following code:

python Copy code
```python
# Assuming 'Make' and 'Model' are columns in your dataset
make_model_counts = df.groupby(['Make', 'Model']).size().reset_index(name='Count')

# Display the result
print(make_model_counts)
```

```
     Make       Model  Count
0    Acura         CL      9
1    Acura        ILX     16
2    Acura  ILX Hybrid      2
3    Acura    Integra     24
4    Acura     Legend     16
..     ...        ...    ...
923  Volvo        V90      1
924  Volvo         XC      1
925  Volvo       XC60     54
926  Volvo       XC70     18
927  Volvo       XC90     17

[928 rows x 3 columns]
```

19

ARR Metodology ;)

Ask what you want
Refine
Repeat

- Now we want the makes in a year…

You
print the list of makes that does have vehicles in the year 2011

ChatGPT
To print the list of makes that have vehicles in the year 2011, you can filter the dataset based on the year and then extract the unique makes. Here's an example:

```python
# Assuming 'Year' and 'Make' are columns in your dataset
makes_2011 = df[df['Year'] == 2011]['Make'].unique()

# Display the result
print("Makes with vehicles in the year 2011:")
print(makes_2011)
```

```
Makes with vehicles in the year 2011:
['BMW' 'Maybach' 'Ferrari' 'Saab' 'Honda' 'Nissan' 'Chevrolet' 'Volvo'
 'Dodge' 'Bentley' 'Ford' 'Mazda' 'Aston Martin' 'Cadillac' 'Mitsubishi'
 'Hyundai' 'Lotus' 'Suzuki' 'Infiniti' 'Lexus' 'Buick' 'Toyota' 'Lincoln'
 'Mercedes-Benz' 'Acura' 'Volkswagen' 'GMC']
```

20

ARR Metodology ;)

Ask what you want
Refine
Repeat

- Now we want it in order…

You
could you order them alphabetically

ChatGPT
Certainly! You can use the `sorted` function to order the list of makes alphabetically before printing. Here's the updated code:

```python
# Assuming 'Year' and 'Make' are columns in your dataset
makes_2011 = df[df['Year'] == 2011]['Make'].unique()

# Order the list alphabetically
makes_2011_sorted = sorted(makes_2011)

# Display the result
print("Makes with vehicles in the year 2011 (ordered alphabetically):")
print(makes_2011_sorted)
```

```
Makes with vehicles in the year 2011 (ordered alphabetically):
['Acura', 'Aston Martin', 'BMW', 'Bentley', 'Buick', 'Cadillac', 'Chevrolet', 'Dodge', 'Ferrari',
'Ford', 'GMC', 'Honda', 'Hyundai', 'Infiniti', 'Lexus', 'Lincoln', 'Lotus', 'Maybach', 'Mazda',
'Mercedes-Benz', 'Mitsubishi', 'Nissan', 'Saab', 'Suzuki', 'Toyota', 'Volkswagen', 'Volvo']
```

21

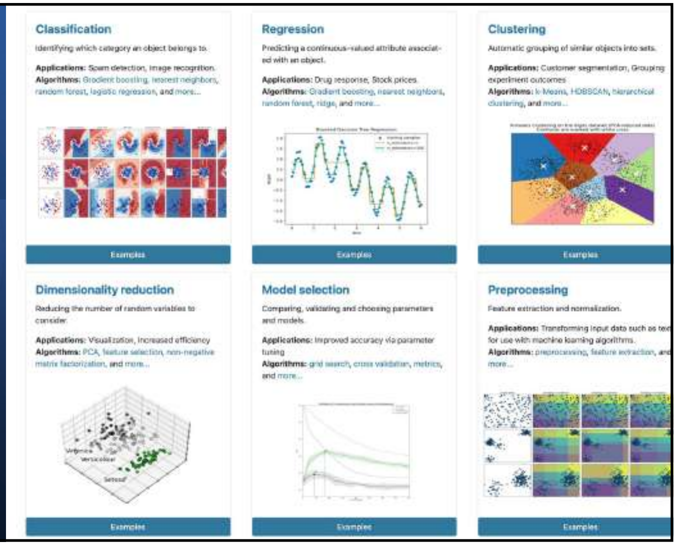
Sources

Google Colab EDA by chatGPT
https://shorturl.at/kPTY6

Script Chat GTP
https://shorturl.at/mrBNZ

22

Sklearn

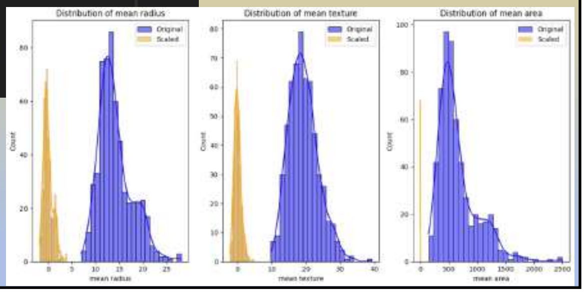


23

Preprocessing

```python
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# Load the breast cancer dataset
cancer = load_breast_cancer()

# Extract features (X) and target variable (y)
X, y = cancer.data, cancer.target

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Perform feature scaling using StandardScaler
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Display the shape of the original and scaled datasets
print("Original training data shape:", X_train.shape)
print("Scaled training data shape:", X_train_scaled.shape)
print("Original testing data shape:", X_test.shape)
print("Scaled testing data shape:", X_test_scaled.shape)
```

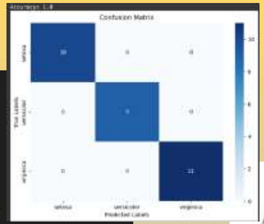
```
Original training data shape: (455, 30)
Scaled training data shape: (455, 30)
Original testing data shape: (114, 30)
Scaled testing data shape: (114, 30)
```

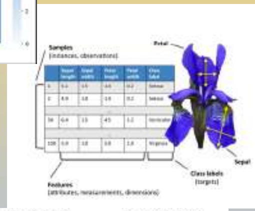


24

Sklearn: Clasification

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix, accuracy_score
import seaborn as sns
import matplotlib.pyplot as plt

iris = load_iris()
X = iris.data
y = iris.target
# Split the dataset into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
svm = SVC()
svm.fit(X_train, y_train)
# Make predictions on the test set
y_pred = svm.predict(X_test)
# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
# Plot confusion matrix
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, cmap='Blues', fmt='g', xticklabels=iris.target_names, yticklabels=iris.target_names)
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.title('Confusion Matrix')
plt.show()
```

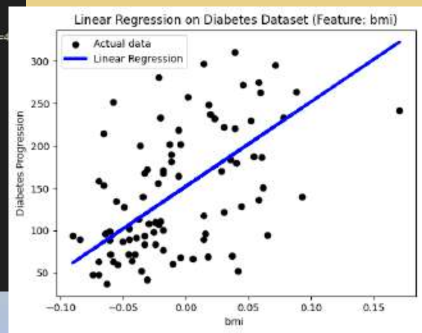




25

Sklearn: Prediction

```
# Load the diabetes dataset
diabetes = load_diabetes()
# Split the data into features (X) and target variable (y)
X = diabetes.data[:, np.newaxis, 2]  # Use a single feature for simplicity
y = diabetes.target
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=4
# Create a linear regression model
model = LinearRegression()
# Train the model
model.fit(X_train, y_train)
# Make predictions on the test set
y_pred = model.predict(X_test)
# Calculate the mean squared error
mse = mean_squared_error(y_test, y_pred)
print(f"Mean Squared Error: {mse:.2f}")
# Plot the results
plt.scatter(X_test, y_test, color='black', label='True values')
plt.plot(X_test, y_pred, color='blue', linewidth=3, label='Linear regression')
plt.title('Linear Regression on Diabetes Dataset')
plt.xlabel('Feature')
plt.ylabel('Target Variable')
plt.legend()
plt.show()
```

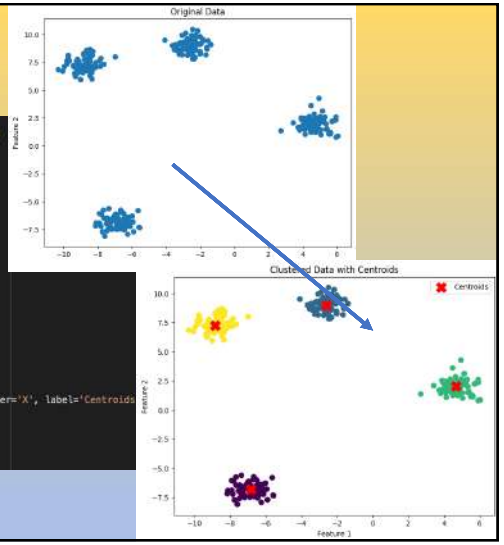


The dataset consists of ten variables: age, sex, body mass index (**BMI**), blood pressure, and six blood measurements.
The target variable is a quantitative measure of **disease progression** one year after baseline.

26

Sklearn: Clustering

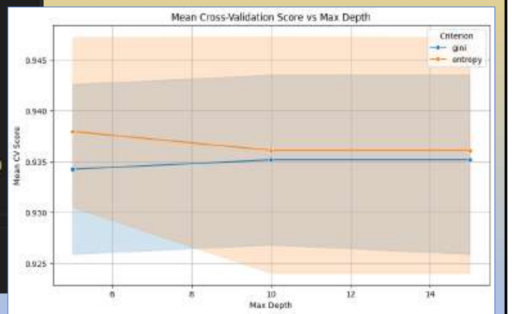
```
# Generate synthetic data
X, _ = make_blobs(n_samples=300, centers=4, cluster_std=0.60, random_state=42)

# Plot the generated data
plt.figure(figsize=(8, 6))
plt.scatter(X[:, 0], X[:, 1], s=50)
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.title('Original Data')
plt.show()

# Perform K-Means clustering
kmeans = KMeans(n_clusters=4)
kmeans.fit(X)

# Get cluster centers and labels
cluster_centers = kmeans.cluster_centers_
cluster_labels = kmeans.labels_

# Plot the clustered data with centroids
plt.figure(figsize=(8, 6))
plt.scatter(X[:, 0], X[:, 1], c=cluster_labels, s=50, cmap='viridis')
plt.scatter(cluster_centers[:, 0], cluster_centers[:, 1], c='red', s=200, marker='X', label='Centroids
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.title('Clustered Data with Centroids')
plt.legend()
plt.show()
```



27

Pipelines and Hyperparameter tuning

```
# Define preprocessing steps
preprocessor = StandardScaler()
# Define classifier
classifier = DecisionTreeClassifier(random_state=42)
# Create a pipeline
pipeline = Pipeline(steps=[('preprocessor', preprocessor),
                           ('classifier', classifier)])
# Define hyperparameters grid for tuning
param_grid = {
    'classifier__criterion': ['gini', 'entropy'],
    'classifier__max_depth': [None, 5, 10, 15],
    'classifier__min_samples_split': [2, 5, 10],
    'classifier__min_samples_leaf': [1, 2, 4]
}
# Perform hyperparameter tuning using GridSearchCV
grid_search = GridSearchCV(pipeline, param_grid, cv=5, scoring='accuracy')
grid_search.fit(X_train, y_train)
# Print the best parameters and best score
print("Best Parameters:", grid_search.best_params_)
print("Best Score:", grid_search.best_score_)
# Evaluate the best model on the test set
best_model = grid_search.best_estimator_
test_accuracy = best_model.score(X_test, y_test)
print("Test Accuracy:", test_accuracy)
```



28

Pyspark: instalación

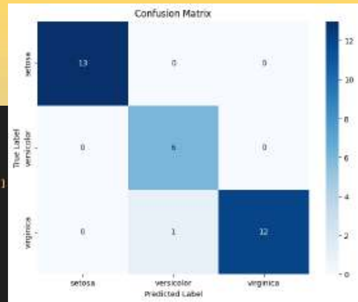
```
!apt-get install openjdk-8-jdk-headless -qq > /dev/null
!wget -q https://archive.apache.org/dist/spark/spark-3.0.3/spark-3.0.3-bin-hadoop2.7.tgz
!tar xf spark-3.0.3-bin-hadoop2.7.tgz
!pip install -q findspark
```

```
import os
os.environ["JAVA_HOME"] = "/usr/lib/jvm/java-8-openjdk-amd64"
os.environ["SPARK_HOME"] = "/content/spark-3.0.3-bin-hadoop2.7"

import findspark
findspark.init()
```

29

Pyspark

```
spark = SparkSession.builder.appName("RandomForestExample").getOrCreate()
# Load Iris dataset
iris = datasets.load_iris()
X = iris.data
y = iris.target
# Convert to Spark DataFrame
data = [(float(X[i][0]), float(X[i][1]), float(X[i][2]), float(X[i][3]), float(y[i])) for i in range(len(X))]
columns = ["feature1", "feature2", "feature3", "feature4", "label"]
df = spark.createDataFrame(data, columns)
# Assemble features into a single column
feature_cols = ["feature1", "feature2", "feature3", "feature4"]
assembler = VectorAssembler(inputCols=feature_cols, outputCol="features")
df = assembler.transform(df)
# Split the data into training and testing sets
(train_data, test_data) = df.randomSplit([0.8, 0.2], seed=42)
# Initialize the RandomForestClassifier
rf_classifier = RandomForestClassifier(labelCol="label", featuresCol="features", numTrees=100)
# Train the model
model = rf_classifier.fit(train_data)
# Make predictions on the test set
predictions = model.transform(test_data)
# Evaluate the model
evaluator = MulticlassClassificationEvaluator(labelCol="label", predictionCol="prediction", metricName="accuracy")
accuracy = evaluator.evaluate(predictions)
print(f"Accuracy: {accuracy * 100:.2f}%")

# Stop the Spark session
spark.stop()
```



30