B.M.S College of Engineering

P.O. Box No.: 1908 Bull Temple Road,
Bangalore-560 019

DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING

Course – Big Data Analytics
Course Code – 20IS6PEBDA
AY 2021-2022

APACHE SPARK

Submitted to – Yogesh N
(Assistant Professor)

Submitted by -
Niranjan Hegde 1BM19IS103
Prashanth Jaganathan 1BM19IS115
Prateek M Gummaraju 1BM19IS117
Samartha S 1BM19IS219

B.M.S College of Engineering

P.O. Box No.: 1908 Bull Temple Road,
Bangalore-560 019

DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING

BMS COLLEGE OF ENGINEERING

(Autonomous College under VTU)
Bull Temple Road, Basavanagudi, Bangalore – 560019

C E R T I F I C A T E

This is to certify that the presentation entitled "**Apache Spark**" is a bona-fide work carried out by **Niranjan Hegde**, **Prashanth Jaganathan, Prateek Gummaraju and Samartha S** bearing USN: **1BM19IS103**, **1BM19IS115**, **1BM19IS117** and **1BM19IS219** respectively, in partial fulfillment of the requirements for the VI Semester degree in **Bachelor of Engineering in Information Science & Engineering** of **Visvesvaraya Technological University, Belgaum** as a part of for the **course Big Data Analytics, Course Code - 20IS6PEBDA** during academic year 2021-2022

Faculty Name: Yogesh N

Designation: Assistant Professor

Signature:

Table of Content

Sl. No.	Topic	Page No.
1	Abstract	4
2	Introduction to Spark	5
3	Features of Spark	6
4	Components of Spark	8
5	Spark vs Hadoop	10
6	PySpark	13
7	Lumify	15
8	Data Wrapper	16
9	References	18

Abstract

Big data is larger, more complex data sets, especially from new data sources. These data sets are so voluminous that traditional data processing software just can't manage them. But these massive volumes of data can be used to address business problems you wouldn't have been able to tackle before. The definition of big data is data that contains greater variety, arriving in increasing volumes and with more velocity. To process big data, we need specialized tools. One such tool is Apache Spark. Apache Spark is an open source parallel processing framework for running large-scale data analytics applications across clustered computers. It can handle both batch and real-time analytics and data processing workloads. In this report, we explore Spark in detail, covering its features, components and comparing it with Hadoop. We also look at PySpark, Lumify and DataWrapper.

Introduction to Spark

Spark was introduced by Apache Software Foundation for speeding up the Hadoop computational computing software process. As against a common belief, Spark is not a modified version of Hadoop and is not, really, dependent on Hadoop because it has its own cluster management. Hadoop is just one of the ways to implement Spark.

Apache Spark is a lightning-fast cluster computing technology, designed for fast computation. It is based on Hadoop MapReduce and it extends the MapReduce model to efficiently use it for more types of computations, which includes interactive queries and stream processing. The main feature of Spark is its in-memory cluster computing that increases the processing speed of an application.

Spark is designed to cover a wide range of workloads such as batch applications, iterative algorithms, interactive queries and streaming. Apart from supporting all these workload in a respective system, it reduces the management burden of maintaining separate tools.

Spark is one of Hadoop's sub project developed in 2009 in UC Berkeley's AMPLab by Matei Zaharia. It was Open Sourced in 2010 under a BSD license. It was donated to Apache software foundation in 2013, and now Apache Spark has become a top level Apache project from Feb-2014.

Features of Spark

Apache Spark has the following features.

1. Lighting-fast processing speed: Big Data processing is all about processing large volumes of complex data. Hence, when it comes to Big Data processing, organizations and enterprises want such frameworks that can process massive amounts of data at high speed. As we mentioned earlier, Spark apps can run up to 100x faster in memory and 10x faster on disk in Hadoop clusters. It relies on Resilient Distributed Dataset (RDD) that allows Spark to transparently store data on memory and read/write it to disc only if needed. This helps to reduce most of the disc read and write time during data processing.

2. Ease of use: Spark allows you to write scalable applications in Java, Scala, Python, and R. So, developers get the scope to create and run Spark applications in their preferred programming languages. Moreover, Spark is equipped with a built-in set of over 80 high-level operators. You can use Spark interactively to query data from Scala, Python, R, and SQL shells.

3. It offers support for sophisticated analytics: Not only does Spark support simple "map" and "reduce" operations, but it also supports SQL queries, streaming data, and advanced analytics, including ML and graph algorithms. It comes with a powerful stack of libraries such as SQL & DataFrames and MLlib (for ML), GraphX, and Spark Streaming. What's fascinating is that Spark lets you combine the capabilities of all these libraries within a single workflow/application.

4. Real-time stream processing: Spark is designed to handle real-time data streaming. While MapReduce is built to handle and process the data that is already stored in Hadoop clusters, Spark can do both and also manipulate data in real-time via Spark Streaming. Unlike other streaming solutions, Spark Streaming can recover the lost work and deliver the exact semantics out-of-the-box without requiring extra code or configuration. Plus, it also lets you reuse the same code for batch and stream processing and even for joining streaming data to historical data.

5. It is flexible: Spark can run independently in cluster mode, and it can also run on Hadoop YARN, Apache Mesos, Kubernetes, and even in the cloud. Furthermore, it can access diverse data sources. For instance, Spark can run on the YARN cluster manager and read any existing Hadoop data. It can read from any Hadoop data sources like HBase, HDFS, Hive, and Cassandra. This aspect of Spark makes it an ideal tool for migrating pure Hadoop applications, provided the apps' use-case is Spark-friendly.

6. Active and expanding community: Developers from over 300 companies have contributed to design and build Apache Spark. Ever since 2009, more than 1200 developers

have actively contributed to making Spark what it is today! Naturally, Spark is backed by an active community of developers who work to improve its features and performance continually. To reach out to the Spark community, you can make use of mailing lists for any queries, and you can also attend Spark meetup groups and conferences.

Components of Spark

The following illustration depicts the different components of Spark.

Apache Spark Core

Spark Core is the underlying general execution engine for spark platform that all other functionality is built upon. It provides In-Memory computing and referencing datasets in external storage systems.

Spark SQL

Spark SQL is a component on top of Spark Core that introduces a new data abstraction called SchemaRDD, which provides support for structured and semi-structured data.

Spark Streaming

Spark Streaming leverages Spark Core's fast scheduling capability to perform streaming analytics. It ingests data in mini-batches and performs RDD (Resilient Distributed Datasets) transformations on those mini-batches of data.

MLlib (Machine Learning Library)

MLlib is a distributed machine learning framework above Spark because of the distributed memory-based Spark architecture. It is, according to benchmarks, done by the MLlib developers

against the Alternating Least Squares (ALS) implementations. Spark MLlib is nine times as fast as the Hadoop disk-based version of Apache Mahout (before Mahout gained a Spark interface).

GraphX

GraphX is a distributed graph-processing framework on top of Spark. It provides an API for expressing graph computation that can model the user-defined graphs by using Pregel abstraction API. It also provides an optimized runtime for this abstraction.

Spark vs Hadoop

Spark and Hadoop are currently the 2 most popular Big Data Tools. They are both developed by the Apache Software Foundation and are widely used open source frameworks for big data architecture. Each framework contains an extensive ecosystem of open-source technologies that prepare, process, manage and analyze big data sets.

Hadoop

Apache Hadoop is an open-source software utility that allows users to manage big data sets by enabling a network of computers (nodes) to solve vast and intricate data problems. It is a highly scalable, cost-effective solution that stores and processes structured, semi-structured and unstructured data. Hadoop got its start as a Yahoo project in 2006, which became a top-level Apache open-source project afterwards. It's a general-purpose form of distributed processing that has several components: the Hadoop Distributed File System (HDFS), stores files in a Hadoop-native format and parallelizes them across a cluster; YARN, a schedule that coordinates application runtimes; and MapReduce, the algorithm that actually processes the data in parallel. Hadoop is built in Java, and accessible through many programming languages, for writing MapReduce code, including Python, through a Thrift client.
It's available either open-source through the Apache distribution, or through vendors such as Cloudera, MapR, or HortonWorks.

The Hadoop ecosystem consists of four primary modules:

- Hadoop Distributed File System (HDFS): Primary data storage system that manages large data sets running on commodity hardware. It also provides high-throughput data access and high fault tolerance.

- Yet Another Resource Negotiator (YARN): Cluster resource manager that schedules tasks and allocates resources (e.g., CPU and memory) to applications.

- Hadoop MapReduce: Splits big data processing tasks into smaller ones, distributes the small tasks across different nodes, then runs each task.

- Hadoop Common (Hadoop Core): Set of common libraries and utilities that the other three modules depend on.

Spark

Spark is a newer project, initially developed in 2012, at the AMPLab at UC Berkeley. It's a top-level Apache project focused on processing data in parallel across a cluster, but the biggest difference is that it works in memory. Apache Spark is a data processing engine for big data sets. Like Hadoop, Spark splits up large tasks across different nodes. However, it tends to perform faster than Hadoop and it uses random access memory (RAM) to cache and process data instead of a file system. This enables Spark to handle use cases that Hadoop cannot.

The Spark ecosystem consists of five primary modules:

- Spark Core: Underlying execution engine that schedules and dispatches tasks and coordinates input and output (I/O) operations.

- Spark SQL: Gather information about structured data to enable users to optimize structured data processing.

- Spark Streaming and Structured Streaming: Both add stream processing capabilities. Spark Streaming takes data from different streaming sources and divides it into micro-batches for a continuous stream. Structured Streaming, built on Spark SQL, reduces latency and simplifies programming.

- Machine Learning Library (MLlib): A set of machine learning algorithms for scalability plus tools for feature selection and building ML pipelines. The primary API for MLlib is DataFrames, which provides uniformity across different programming languages like Java, Scala and Python.

- GraphX: User-friendly computation engine that enables interactive building, modification and analysis of scalable, graph-structured data.

Differences between Hadoop and Spark

1. Performance

Spark is faster because it uses random access memory (RAM) instead of reading and writing intermediate data to disks. Hadoop stores data on multiple sources and processes it in batches via MapReduce.

2. Cost

Both Spark and Hadoop are available for free as open-source Apache projects, meaning you could potentially run it with zero installation costs. However, it is important to consider the total cost of ownership, which includes maintenance, hardware and software purchases, and hiring a team that understands cluster administration. The general rule of thumb for on-prem installations is that Hadoop requires more memory on disk and Spark requires more RAM, meaning that setting up Spark clusters can be more expensive. Additionally, since Spark is the newer system, experts in it are rarer and more costly.

3. Fault tolerance

Hadoop is highly fault-tolerant because it was designed to replicate data across many nodes. Each file is split into blocks and replicated numerous times across many machines, ensuring that if a single machine goes down, the file can be rebuilt from other blocks elsewhere. Spark's fault tolerance is achieved mainly through RDD operations. Initially, data-at-res is stored in HDFS, which is fault-tolerant through Hadoop's architecture. As an RDD is built, so is a lineage, which remembers how the dataset was constructed, and,

since it's immutable, can rebuild it from scratch if need be.

4. Data Processing

Though both platforms process data in a distributed environment, Hadoop is ideal for batch processing and linear data processing. Spark is ideal for real-time processing and processing live unstructured data streams

5. Ease of Use

Hadoop's MapReduce has no interactive mode and is complex. It needs to handle low-level APIs to process the data, which requires a lot of coding. Spark on the other hand supports user-friendly APIs for different languages. It has an interactive mode and provides intermediate feedback for queries and actions.

6. Language Support

Hadoop framework is developed in Java programming language. While, MapReduce applications are written in Python, R and C++. Spark is developed in Scala language and supports other programming languages like Python, R and Java.

7. Scalability

Hadoop is highly scalable as we can add 'n' number of nodes in the cluster. Yahoo reportedly used a 42,000 node Hadoop cluster. Whereas the largest known spark cluster has 8000 nodes. But as Big Data grows, it is expected that cluster sizes will increase to maintain throughput expectations.

PySpark

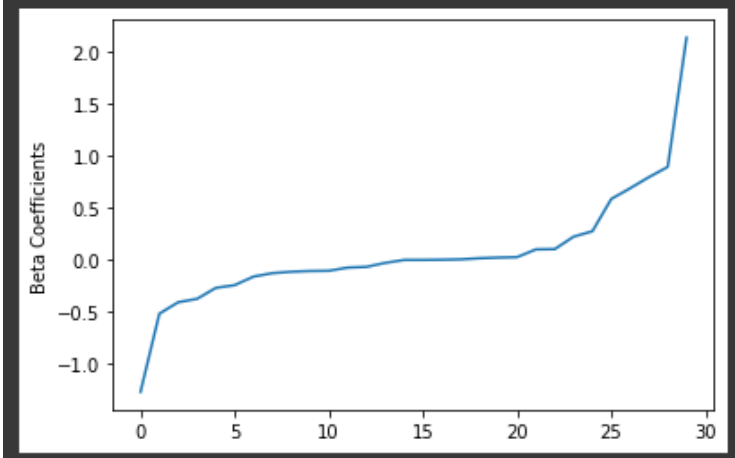
PySpark is the Python API for Apache Spark

```
df.show()
```

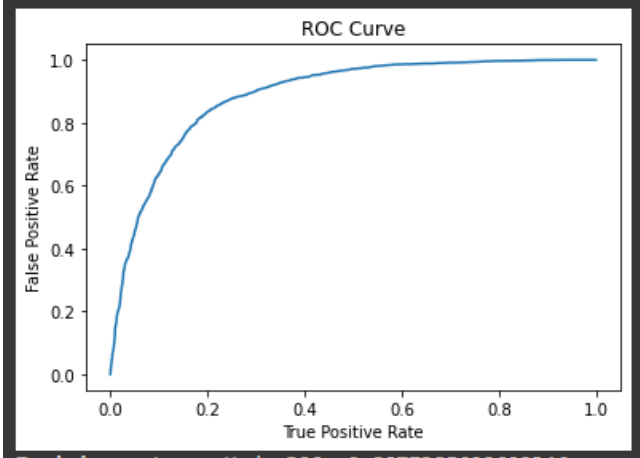
```
+---+-----------+--------+---------+-------+-------+-------+----+-------+---+-----+--------+--------+-----+--------+--------+-------+
|age|        job| marital|education|default|balance|housing|loan|contact|day|month|duration|campaign|pdays|previous|poutcome|deposit|
+---+-----------+--------+---------+-------+-------+-------+----+-------+---+-----+--------+--------+-----+--------+--------+-------+
| 59|     admin.| married|secondary|     no|   2343|    yes|  no|unknown|  5|  may|    1042|       1|   -1|       0| unknown|    yes|
| 56|     admin.| married|secondary|     no|     45|     no|  no|unknown|  5|  may|    1467|       1|   -1|       0| unknown|    yes|
| 41| technician| married|secondary|     no|   1270|    yes|  no|unknown|  5|  may|    1389|       1|   -1|       0| unknown|    yes|
| 55|   services| married|secondary|     no|   2476|    yes|  no|unknown|  5|  may|     579|       1|   -1|       0| unknown|    yes|
| 54|     admin.| married| tertiary|     no|    184|     no|  no|unknown|  5|  may|     673|       2|   -1|       0| unknown|    yes|
| 42| management|  single| tertiary|     no|      0|    yes| yes|unknown|  5|  may|     562|       2|   -1|       0| unknown|    yes|
| 56| management| married| tertiary|     no|    830|    yes| yes|unknown|  6|  may|    1201|       1|   -1|       0| unknown|    yes|
| 60|    retired|divorced|secondary|     no|    545|    yes|  no|unknown|  6|  may|    1030|       1|   -1|       0| unknown|    yes|
| 37| technician| married|secondary|     no|      1|    yes|  no|unknown|  6|  may|     608|       1|   -1|       0| unknown|    yes|
| 28|   services|  single|secondary|     no|   5090|    yes|  no|unknown|  6|  may|    1297|       3|   -1|       0| unknown|    yes|
| 38|     admin.|  single|secondary|     no|    100|    yes|  no|unknown|  7|  may|     786|       1|   -1|       0| unknown|    yes|
| 30|blue-collar| married|secondary|     no|    309|    yes|  no|unknown|  7|  may|    1574|       2|   -1|       0| unknown|    yes|
| 29| management| married| tertiary|     no|    199|    yes| yes|unknown|  7|  may|    1689|       4|   -1|       0| unknown|    yes|
| 46|blue-collar|  single| tertiary|     no|    460|    yes|  no|unknown|  7|  may|    1102|       2|   -1|       0| unknown|    yes|
| 31| technician|  single| tertiary|     no|    703|    yes|  no|unknown|  8|  may|     943|       2|   -1|       0| unknown|    yes|
| 35| management|divorced| tertiary|     no|   3837|    yes|  no|unknown|  8|  may|    1084|       1|   -1|       0| unknown|    yes|
| 32|blue-collar|  single|  primary|     no|    611|    yes|  no|unknown|  8|  may|     541|       3|   -1|       0| unknown|    yes|
| 49|   services| married|secondary|     no|     -8|    yes|  no|unknown|  8|  may|    1119|       1|   -1|       0| unknown|    yes|
| 41|     admin.| married|secondary|     no|     55|    yes|  no|unknown|  8|  may|    1120|       2|   -1|       0| unknown|    yes|
| 49|     admin.|divorced|secondary|     no|    168|    yes| yes|unknown|  8|  may|     513|       1|   -1|       0| unknown|    yes|
+---+-----------+--------+---------+-------+-------+-------+----+-------+---+-----+--------+--------+-----+--------+--------+-------+
only showing top 20 rows
```

```
from pyspark.ml.classification import LogisticRegression
lr = LogisticRegression(featuresCol = 'features', labelCol = 'label', maxIter=10)
lrModel = lr.fit(train)
```

```
import matplotlib.pyplot as plt
import numpy as np
beta = np.sort(lrModel.coefficients)
plt.plot(beta)
plt.ylabel('Beta Coefficients')
plt.show()
```



```
trainingSummary = lrModel.summary
roc = trainingSummary.roc.toPandas()
plt.plot(roc['FPR'],roc['TPR'])
plt.ylabel('False Positive Rate')
plt.xlabel('True Positive Rate')
plt.title('ROC Curve')
plt.show()
print('Training set areaUnderROC: ' + str(trainingSummary.areaUnderROC))
```




```
Training set areaUnderROC: 0.8877385690600346
```

```
predictions.select('prediction').show()
```

```
+----------+
|prediction|
+----------+
|       0.0|
|       0.0|
|       1.0|
|       0.0|
|       0.0|
|       0.0|
|       0.0|
|       0.0|
|       0.0|
|       0.0|
|       1.0|
|       0.0|
|       0.0|
|       0.0|
|       0.0|
|       0.0|
|       0.0|
|       0.0|
|       0.0|
|       0.0|
+----------+
only showing top 20 rows
```

Big Data Analytics and Visualization

- ## Big Data Visualization using Lumify

Lumify is a big data fusion, analysis, and visualization platform. Like all big data analytics tools, it too enables you to understand connections and explore the relationship between your data.
Lumify is considered as a good big data analytics tool because it facilitates its users to get a set of analytics options that include graph visualizations, full-text faceted search, dynamic histograms, interactive geospatial views, and collaborative workspaces that can be shared in real-time.
Lumify offers both 2D and 3D graph visualizations with automatic layouts. It also provides a plethora of options to analyze the links between different entities in a graph.
Lumify comes with specific ingest processing and interface elements for textual content, images, and videos. The platform allows you to organize your work in different workspaces.
The platform is built on proven,scalable big data technologies. It is secure, scalable, and backed by a motivated full-time development team.

Lumify is a free and open source tool for big data fusion/integration, analytics, and visualization.

Its primary features include full-text search, 2D and 3D graph visualizations, automatic layouts, link analysis between graph entities, integration with mapping systems, geospatial analysis, multimedia analysis, real-time collaboration through a set of projects or workspaces.

Advantages of Lumify:

1. Scalable
2. Secure
3. Supported by a dedicated full-time development team.
4. Supports the cloud based environment. Works well with Amazon’s AWS.

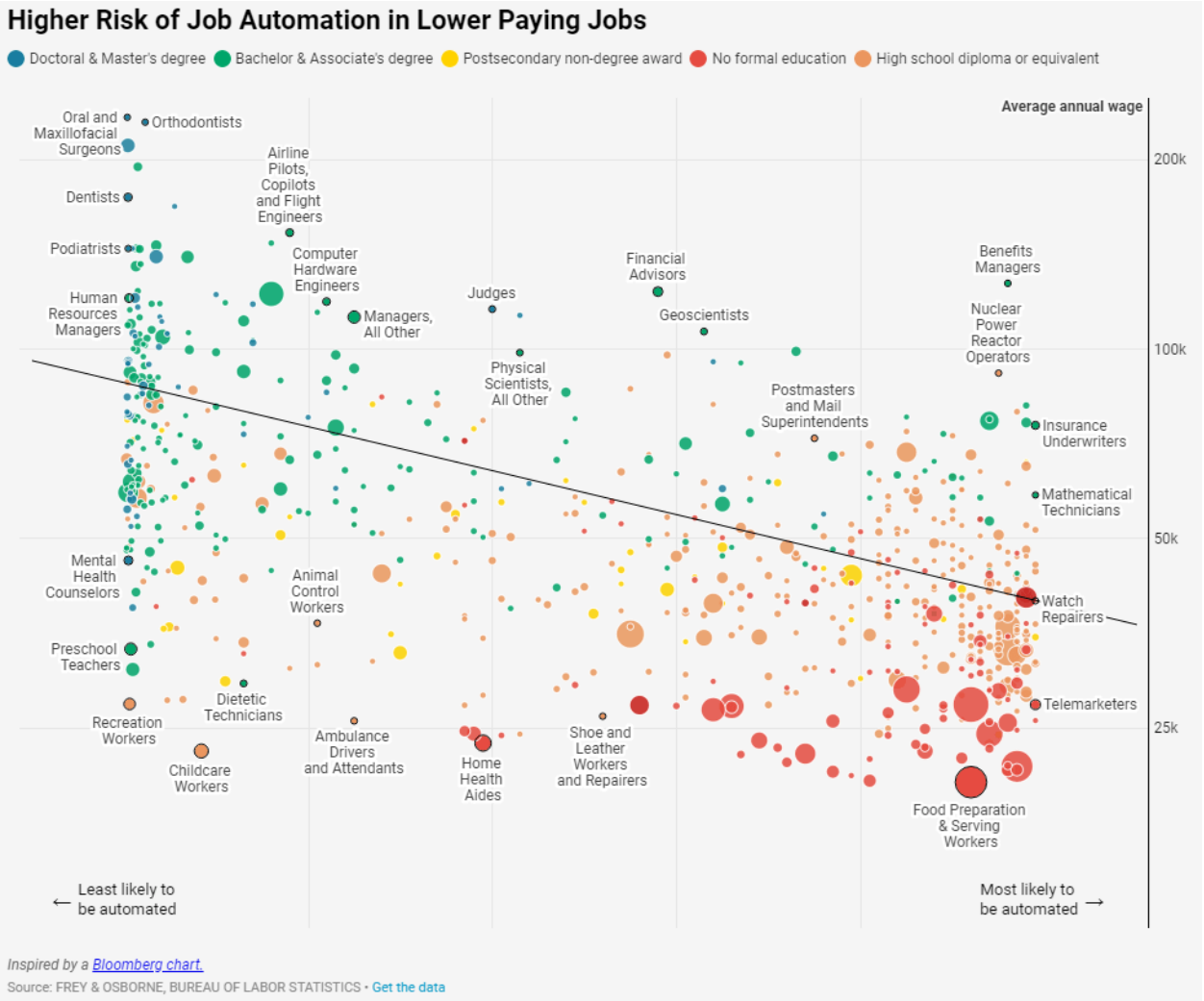
- **Big Data Visualization using DataWrapper**

Datawrapper is an open source platform for data visualization that aids its users to generate simple, precise and embeddable charts very quickly.

Its major customers are newsrooms that are spread all over the world. Some of the names include The Times, Fortune, Mother Jones, Bloomberg, Twitter etc.

Pros:

- Device friendly. Works very well on all type of devices – mobile, tablet or desktop.
- Fully responsive
- Fast
- Interactive
- Brings all the charts in one place.
- Great customization and export options.
- Requires zero coding.



A use case of DataWrapper

REFERENCES

1. https://spark.apache.org/
2. https://databricks.com/spark/about
3. https://www.ibm.com/cloud/blog/hadoop-vs-spark#:~:text=Like%20Hadoop%2C%20Spark%20splits%20up,use%20cases%20that%20Hadoop%20cannot.
4. https://logz.io/blog/hadoop-vs-spark/
5. https://www.datawrapper.de/