NAME : EDWARD **Course : Big Data Processing**
NIM : 2201741971 **Course Code : COMP6579**
CLASS : LB-08 **Faculty / Department : School of Computer Science**

I. CASE (100%)

1. **[15%]** YARN (Yet Another Resource Negotiator) is a very important component in the Big Data ecosystem.
   a. Each application under YARN will be handled by the Application Master. **Explain what is the main function of the Application Master**, **how the mechanism will run in the event of Application Master failure?**

Answer:

- **Application Master Main Function**
  - **Requests the container from the node manager** by **sending a Container Launch Context(CLC)** which includes everything an application needs to run.
  - **Once the application is started**, it **sends the health report to the resource manager from time-to-time**.
- **Base Concept of YARN Application Master Failure**
  - **Applications in YARN are retried in the event of failure**.
  - **The default value is 2, so if a MapReduce application master fails twice it will not be tried again and the job will fail**.
  - **If want to increase the number of MapReduce application master attempts, must increase the YARN setting on the cluster, too**.
- **Mechanism Run in the Event of Application Master Failure**
  - **Application master sends periodic heartbeats to the resource manager**.
  - **In the event of application master failure**, the **resource manager will detect the failure** and **start a new instance of the master running in a new container** which is **managed by a node manager**.
  - It will **use the job history to recover the state of the tasks that were already run by the application**, so they **don't have to be rerun from the beginning again**.

NAME : EDWARD **Course : Big Data Processing**
NIM : 2201741971 **Course Code : COMP6579**
CLASS : LB-08 **Faculty / Department : School of Computer Science**

b. **What does Negotiator mean in YARN? When the negotiations in YARN take place?**

Answer:

- **Negotiator Meaning in YARN**
  - **Allows multiple engine access thus giving organizations a benefit of multi-tenancy**.
  - **If the job is very important then it can hold the resource for long period**.
  - **Report about node resources with another protocol**.
- **YARN Negotiator Take Place**

  Take Place When Node Failed to Occur. **Steps to do it:**
  - **Resource Manager works with Application Manager, Node Manager and Containers**.
  - **Along with Node Manager it makes possible the hosting of any application on YARN**.
  - **It is responsible for managing the cluster and providing the resources by following pluggable scheduling policy**.
  - **When node failure occurs, it is reported to Application Master**.
  - **When Application Master Send Resource Request to Resource Manager then it provides the Container class of the cluster to Application Master** and on continuous resource demand, **Resource Manager's decisions are purely based on cluster's scheduling policies**.
  - **The new model which uses Resource Manager and Container class allows dynamic utilization of nodes** which **results in increasing the overall utilization of the cluster**.

NAME : EDWARD **Course : Big Data Processing**
NIM : 2201741971 **Course Code : COMP6579**
CLASS : LB-08 **Faculty / Department : School of Computer Science**

2. **[10%]** Explain **how Zookeper can prevent a "split brain" condition where there are two Resource Managers active at the same time**.

Answer:

- **What is "split brain"?**
  - Split-Brain is like **when have two nodes in cluster, they both know that a master needs to be elected in this cluster**.
  - Then **when there is no problem in the communication between the two of them**, **a consensus will be reached and one of them will be selected as the master**.
  - But **if there is a problem with the communication between them**, **both nodes will feel that there is no master now, so each elects itself as the master**. **So there will be Two Masters in the cluster**.
- **How does Zookeeper solve the "split brain" problem?**
  - **Zookeeper uses Quorums by default to prevent the "split brain" phenomenon**.
  - **Only more than half of the nodes in the cluster can vote for the Leader**.
  - This way **can ensure the uniqueness of the leader, either choose the only leader, or the election fails**.
  - **Suppose a leader dies, and the rest of the followers elect a new leader**. **At this time, the old leader is resurrected and still considers himself a leader**.
  - At this time, **it will also refuse to write requests to other followers**. **Because each time a new leader is generated, an epoch label (identifying the current rule period of that leader) is generated**.
  - **This epoch is incremented**. **If the followers confirm the existence of the new leader and know its epoch**, **they will reject the epoch less than the current leader All requests for epoch**.

NAME : EDWARD **Course : Big Data Processing**
NIM : 2201741971 **Course Code : COMP6579**
CLASS : LB-08 **Faculty / Department : School of Computer Science**

3. **[15%]** Hadoop provides two scripting languages i.e., PIG and HiveQL to help Big Data application developers to develop a program that use the Map Reduce programming model. **What do you know about the two scripting languages? What are the advantages and disadvantages of PIG and HiveQL?**

Answer:

- **About Pig and HiveQL**
  - **Pig**
    - To explore large data sets.
    - High level and dataflow language.
    - Compatible with any versions of Hadoop.
  - **HiveQL**
    - To process structured data in Hadoop a data warehouse infrastructure tool.
    - It was developed by Facebook initially and later taken up by Apache Software Foundation.
    - It is not a relational database and language for row-level updates and real-time queries.
    - It serves both the purposes i.e. Batch Query processing(Large data set processing) and Interactive Query processing(Real time processing).
- **Advantages of Pig and HiveQL**
  - **Pig**
    - Fast execution that works with MapReduce, Spark.
    - Its ability to process almost any amount of data, regardless of size.
    - Features that let it join forces with other tools like Hive and DBMS to improve their functionality.
    - A strong documentation process that helps new users learn Pig Latin.
    - Local and remote interoperability that lets professionals work from anywhere with a reliable connection.

NAME : EDWARD **Course : Big Data Processing**
NIM : 2201741971 **Course Code : COMP6579**
CLASS : LB-08 **Faculty / Department : School of Computer Science**

- **HiveQL**
  - Simple querying for anyone already familiar with SQL.
  - Scalability that can seek reinforcements from multiple servers when needed.
  - The option to generate ad hoc queries for data analysis.
  - How well it handles long-running queries.
  - Its ability to connect with a variety of relational databases, including Postgres and MySQL.
  - Options to write custom functions with Java and Python.
  - Simplifying the Hadoop experience, especially when people without technical backgrounds participate in data projects.

- **Disadvantages of Pig and HiveQL**
  - **Pig**
    - Inability to solve complex mathematical problems.
    - Difficulty implementing sequential checks.
    - Few options for looping across data, which can add to a user's workload.
    - Domain-specific language (Pig Latin) that some people will have difficulty mastering.
  - **HiveQL**
    - Lack of support for processing data online.
    - Inability to support subqueries,
    - Complex approach to updating data.
    - Slow ad-hoc querying speeds.
    - Lack of security controls that let admins assign specific roles to users.
    - Putting ease of use above processing speeds, especially when it comes to batch processing.

NAME : EDWARD **Course : Big Data Processing**
NIM : 2201741971 **Course Code : COMP6579**
CLASS : LB-08 **Faculty / Department : School of Computer Science**

4. **[15%]** One of the NoSQL database available in the Big Data ecosystem is the Columnar Database (column-based database). **What do you know about the Columnar Database? In which database operations (insert, update, or delete) a columnar database is superior to a row-based database, and in which database operations is a columnar database no better than a row-based database? Explain why this is so.**

Answer:

- **About Columnar Database**
  - More efficient than a traditional database because data is stored in columns instead of rows.
  - Used in data warehouses where companies send huge amounts of data from multiple sources for B analysis.
  - Query function is faster because column design keeps data closer, reducing search time.
  - Each client's name appears in a "name" column, and all addresses are displayed in an "address" column.
  - Preferred for analytics applications because allow to quickly retrieve columns from data.
  - Designed for data storage and large data processing because they use distributed clusters of cost-effective hardware to increase economies of scale.
- **Database operations a columnar database is superior to a row-based database**.
  - **Best suited for Inserts and Deletions**.
  - Can be marked by many short atomic database operations which are very common in day-to-day planning.
- **Database operations is a columnar database no better than a row-based database**.
  - **Not best suited for Updates**.
  - Organize data by absorbing all data and keeping it in memory side by side.

NAME : EDWARD **Course : Big Data Processing**
NIM : 2201741971 **Course Code : COMP6579**
CLASS : LB-08 **Faculty / Department : School of Computer Science**

5. **[20%] Give an example of Big Data implementation where Stream Processing is required in the Data Ingestion or Data Analytics step. What components of the Big Data ecosystem can be implemented in that case. Explain why Stream Processing capabilities are needed in this case, what is the impact if batch processing is used?**

Answer:

- **Example of Big Data implementation.**
  - **Stream Processing helps in presenting real time information to the user.**
  - **Examples of big data implementation in stream processing is needed are:**
    - Healthcare: Real time monitoring of health conditions.
    - Finance: Involving lot of transactions need real time updates and consists of big data.
    - Netflix: Has a lot of data and comes under category of big data which involves stream processing real time.

- **Big Data ecosystem can be implemented**
  - **HDFS**
    - ➔ Responsible for storing large data sets of structured or unstructured data across various nodes and thereby maintaining the metadata in the form of log files.
  - **MapReduce**
    - ➔ Performs sorting and filtering of data and thereby organizing them in the form of group. Map generates a key-value pair-based result which is later on processed by the summarization by aggregating the mapped data.

NAME : EDWARD **Course : Big Data Processing**
NIM : 2201741971 **Course Code : COMP6579**
CLASS : LB-08 **Faculty / Department : School of Computer Science**

- **YARN**
  - ➔ Jelps to manage the resources across the clusters. In short, it performs scheduling and resource allocation for the Hadoop System.

- **PIG**
  - ➔ Structuring the data flow, processing and analyzing huge data sets by executing commands and in the background, all the activities of MapReduce are taken care of. After the processing, pig stores the result in HDFS.

- **HIVE**
  - ➔ Reading and writing of large data sets. Establishing the data storage permissions and connection whereas HIVE Command line helps in the processing of queries.

- **Mahout**
  - ➔ Allows Machine Learnability to a system or application. Provides various libraries or functionalities such as collaborative filtering, clustering, and classification which are nothing but concepts of Machine learning.

- **Zookeeper**
  - ➔ Performing synchronization, inter-component-based communication, grouping, and maintenance.

- **Oozie**
  - ➔ Oozie workflow is the jobs that need to be executed in a sequentially ordered manner.
  - ➔ Oozie Coordinator jobs are those that are triggered when some data or external stimulus is given to it.

NAME : EDWARD **Course : Big Data Processing**
NIM : 2201741971 **Course Code : COMP6579**
CLASS : LB-08 **Faculty / Department : School of Computer Science**

❖ **Application Layer**

➔ Having real time updates on the user application layer to show the data clearly and updated on the spot.

❖ **Data Processing layer**

➔ One of the most complex, but important task is of this layer in which all the data is being processed and further supplied to be shown on the application layer.

❖ **Data collection layer**

➔ Next component that plays an important role is this layer. Big data has 4Vs: Volume, velocity, variety, and value. All these are controlled in this layer.

❖ **Platform Management layer**

➔ This is the management layer which manages everything.

- **Why Stream Processing Capabilities are Needed**
  - ✓ The data is moved and processed by batches. This batch must be queried by a user.
  - ✓ Thus, the system would understand when to fetch data and how to process it and how to present it.
  - ✓ In Data analytics which is required in real time, in this case, data can't be real time if we use batch processing.
  - ✓ Thus, stream processing capabilities are needed.

NAME : EDWARD **Course : Big Data Processing**
NIM : 2201741971 **Course Code : COMP6579**
CLASS : LB-08 **Faculty / Department : School of Computer Science**

6. **[25%]** Create a Python script that uses Apache Spark Machine Learning Library (Spark ML lib) to classify Iris data (https://archive.ics.uci.edu/ml/datasets/iris). Use the Naïve Bayes algorithm with 70% training data and the rest for testing. Show the accuracy of the results of the classification process.

Answer:

Link Google Colab:

https://colab.research.google.com/drive/15rjUXpyXk1OVwxWx5MWoZiFbsEWd2PWO?usp=sharing

Install Spark ML Library

```
In [1]: !apt-get install openjdk-8-jdk-headless -qq > /dev/null
        !wget -q https://downloads.apache.org/spark/spark-3.0.1/spark-3.0.1-bin-hadoop2.7.tgz
        !tar -xvf spark-3.0.1-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-3.0.1-bin-hadoop2.7"
        import findspark
        findspark.init()
```

Membuat Builder untuk Spark Session

```
[113]: #Memakai Apache Spark
       from pyspark.sql import SparkSession

       spark = SparkSession.builder.getOrCreate()
```

Membaca Iris Dataset

```
[122]: #Membaca File CSV dari Iris Dataset
       data = spark.read.option("inferSchema", "true").csv("Iris.csv", header=True)
       #Splitting Dataset menjadi 70% Training dan 30% Testing
       training, testing = data.randomSplit([0.7, 0.3])
```

Hasil Splitting Dataset

```
In [123]: #Hasil dari Splitting, Training(70%) dan Testing(30%)
          print("Total Data Awal     : ",data.count())
          print("Total Data Training : ",training.count())
          print("Total Data Testing  : ",testing.count())

Total Data Awal     :  150
Total Data Training :  108
Total Data Testing  :  42
```

NAME : EDWARD **Course : Big Data Processing**
NIM : 2201741971 **Course Code : COMP6579**
CLASS : LB-08 **Faculty / Department : School of Computer Science**

Menampilkan 5 Data Teratas Data Training

In [124]:
```
# Data Training
print("5 Data Paling Atas dari Training")
training.show(5)
```

```
5 Data Paling Atas dari Training
+---+-------------+------------+-------------+------------+-----------+
| Id|SepalLengthCm|SepalWidthCm|PetalLengthCm|PetalWidthCm|    Species|
+---+-------------+------------+-------------+------------+-----------+
|  1|          5.1|         3.5|          1.4|         0.2|Iris-setosa|
|  2|          4.9|         3.0|          1.4|         0.2|Iris-setosa|
|  5|          5.0|         3.6|          1.4|         0.2|Iris-setosa|
|  7|          4.6|         3.4|          1.4|         0.3|Iris-setosa|
|  9|          4.4|         2.9|          1.4|         0.2|Iris-setosa|
+---+-------------+------------+-------------+------------+-----------+
only showing top 5 rows
```

Menampilkan 5 Data Teratas Data Testing

In [125]:
```
# Data Testing
print("5 Data Paling Atas dari Testing")
testing.show(5)
```

```
5 Data Paling Atas dari Testing
+---+-------------+------------+-------------+------------+-----------+
| Id|SepalLengthCm|SepalWidthCm|PetalLengthCm|PetalWidthCm|    Species|
+---+-------------+------------+-------------+------------+-----------+
|  3|          4.7|         3.2|          1.3|         0.2|Iris-setosa|
|  4|          4.6|         3.1|          1.5|         0.2|Iris-setosa|
|  6|          5.4|         3.9|          1.7|         0.4|Iris-setosa|
|  8|          5.0|         3.4|          1.5|         0.2|Iris-setosa|
| 11|          5.4|         3.7|          1.5|         0.2|Iris-setosa|
+---+-------------+------------+-------------+------------+-----------+
only showing top 5 rows
```

Menghapus Kolom “Id” di Data Training dan Testing

In [126]:
```
# Drop Kolom "Id" karena tidak diperlukan untuk kasus ini
training = training.drop("Id")
testing = testing.drop("Id")
```

In [127]:
```
# Data Testing
print("5 Data Paling Atas dari Testing")
training.show(5)
```

```
5 Data Paling Atas dari Testing
+-------------+------------+-------------+------------+-----------+
|SepalLengthCm|SepalWidthCm|PetalLengthCm|PetalWidthCm|    Species|
+-------------+------------+-------------+------------+-----------+
|          5.1|         3.5|          1.4|         0.2|Iris-setosa|
|          4.9|         3.0|          1.4|         0.2|Iris-setosa|
|          5.0|         3.6|          1.4|         0.2|Iris-setosa|
|          4.6|         3.4|          1.4|         0.3|Iris-setosa|
|          4.4|         2.9|          1.4|         0.2|Iris-setosa|
+-------------+------------+-------------+------------+-----------+
only showing top 5 rows
```

In [128]:
```
# Data Testing
print("5 Data Paling Atas dari Testing")
testing.show(5)
```

```
5 Data Paling Atas dari Testing
+-------------+------------+-------------+------------+-----------+
|SepalLengthCm|SepalWidthCm|PetalLengthCm|PetalWidthCm|    Species|
+-------------+------------+-------------+------------+-----------+
|          4.7|         3.2|          1.3|         0.2|Iris-setosa|
|          4.6|         3.1|          1.5|         0.2|Iris-setosa|
|          5.4|         3.9|          1.7|         0.4|Iris-setosa|
|          5.0|         3.4|          1.5|         0.2|Iris-setosa|
|          5.4|         3.7|          1.5|         0.2|Iris-setosa|
+-------------+------------+-------------+------------+-----------+
only showing top 5 rows
```

NAME : EDWARD **Course : Big Data Processing**
NIM : 2201741971 **Course Code : COMP6579**
CLASS : LB-08 **Faculty / Department : School of Computer Science**

Memilih Fitur yang digunakan

In [129]:
```
training = training.select("SepalLengthCm", "SepalWidthCm", "PetalLengthCm", "PetalWidthCm","Species")
testing = testing.select("SepalLengthCm", "SepalWidthCm", "PetalLengthCm", "PetalWidthCm","Species")
```

In [130]:
```
training = training.na.drop()
testing = testing.na.drop()
```

In [131]:
```
from pyspark.sql.functions import when

training = training.withColumn("Species", when(training["Species"] == "Iris-setosa", 2).
                                          when(training["Species"] == "Iris-versicolor", 1).
                                          when(training["Species"] == "Iris-virginica", 0))

testing = testing.withColumn("Species", when(testing["Species"] == "Iris-setosa", 2).
                                        when(testing["Species"] == "Iris-versicolor", 1).
                                        when(testing["Species"] == "Iris-virginica", 0))
```

Normalisasi Datanya dengan StandardScaler

In [132]:
```
from pyspark.ml.feature import VectorAssembler, StandardScaler

cols = training.columns
cols.remove("Species")

training = VectorAssembler(inputCols=cols, outputCol="Features").transform(training)
testing = VectorAssembler(inputCols=cols, outputCol="Features").transform(testing)
```

In [133]:
```
scaler = StandardScaler(inputCol="Features",outputCol="Scaled Features")
training = scaler.fit(training).transform(training)
testing = scaler.fit(testing).transform(testing)
```

Gunakan Model Naïve Bayes

In [136]:
```
from pyspark.ml.classification import NaiveBayes

modelNaiveBayes =  NaiveBayes(featuresCol="Scaled Features", labelCol="Species", modelType="multinomial").fit(training)

predictionDataTest = modelNaiveBayes.transform(testing)
predictionDataTest.show(5)
```

```
+-------------+------------+-------------+------------+-------+-----------------+--------------------+--------------------+----
----------------+----------+
|SepalLengthCm|SepalWidthCm|PetalLengthCm|PetalWidthCm|Species|         Features|     Scaled Features|       rawPrediction|
probability|prediction|
+-------------+------------+-------------+------------+-------+-----------------+--------------------+--------------------+----
----------------+----------+
|          4.7|         3.2|          1.3|         0.2|      2|[4.7,3.2,1.3,0.2]|[5.78968276392355...|[-15.895927164215...|[0.1
1720619491329...|       2.0|
|          4.6|         3.1|          1.5|         0.2|      2|[4.6,3.1,1.5,0.2]|[5.66649802426560...|[-15.755794897315...|[0.1
3563637218522...|       2.0|
|          5.4|         3.9|          1.7|         0.4|      2|[5.4,3.9,1.7,0.4]|[6.65197594152918...|[-19.325754677397...|[0.1
1632293891083...|       2.0|
|          5.0|         3.4|          1.5|         0.2|      2|[5.0,3.4,1.5,0.2]|[6.15923698289739...|[-16.925940989587...|[0.1
0922880282449...|       2.0|
|          5.4|         3.7|          1.5|         0.2|      2|[5.4,3.7,1.5,0.2]|[6.65197594152918...|[-18.096087081859...|[0.0
8674973603295...|       2.0|
+-------------+------------+-------------+------------+-------+-----------------+--------------------+--------------------+----
----------------+----------+
only showing top 5 rows
```

Cetak Akurasi dari Prediksi “Species”

In [138]:
```
from pyspark.ml.evaluation import BinaryClassificationEvaluator

evaluator=BinaryClassificationEvaluator(rawPredictionCol="rawPrediction",labelCol="Species")

result=evaluator.evaluate(predictionDataTest)

print("Accuracy: {}%".format(result * 100))
```

```
Accuracy: 99.32659932659934%
```