BIG DATA ANALYTICS PROJECT

Dharshani Anandkumar

23MSP3068

```
!pip install pyspark
```

```
    Collecting pyspark
      Downloading pyspark-3.5.1.tar.gz (317.0 MB)
         ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 317.0/317.0 MB 2.1 MB/s eta 0:00:00
      Preparing metadata (setup.py) ... done
    Requirement already satisfied: py4j==0.10.9.7 in /usr/local/lib/python3.10/dist-packages (from pyspark) (0.10.9.7)
    Building wheels for collected packages: pyspark
      Building wheel for pyspark (setup.py) ... done
      Created wheel for pyspark: filename=pyspark-3.5.1-py2.py3-none-any.whl size=317488491 sha256=48924ad0fbec9a87463a3ab11f015d071b0f
      Stored in directory: /root/.cache/pip/wheels/80/1d/60/2c256ed38dddce2fdd93be545214a63e02fbd8d74fb0b7f3a6
    Successfully built pyspark
    Installing collected packages: pyspark
    Successfully installed pyspark-3.5.1
```

```
from pyspark import SparkContext
from pyspark.sql import SparkSession,SQLContext

from pyspark.sql.types import StructType,StructField,DoubleType,StringType,IntegerType,NumericType

from pyspark.sql.functions import col
from pyspark.sql import functions as func

import time
import numpy as np

from scipy import linalg

from scipy.stats import norm

from scipy.sparse import csr_matrix
from pyspark.ml.linalg import Vectors

import matplotlib.pyplot as plt

from pyspark.ml.feature import StringIndexer,VectorAssembler,StandardScaler,PCA
from pyspark.ml.regression import LinearRegression
from pyspark.ml.evaluation import RegressionEvaluator
from pyspark.ml.tuning import ParamGridBuilder,CrossValidator

from pyspark.ml import Pipeline
from pyspark.ml.regression import GeneralizedLinearRegression
from pyspark.ml.classification import LogisticRegression,LinearSVC,OneVsRest
from pyspark.ml.evaluation import BinaryClassificationEvaluator,MulticlassClassificationEvaluator

from pyspark.sql.functions import approxCountDistinct,col,exp,mean,stddev

from pyspark.ml.clustering import KMeans,GaussianMixture
from pyspark.ml.evaluation import ClusteringEvaluator

import pandas as pd

from nltk.tokenize import RegexpTokenizer

from sklearn.decomposition import TruncatedSVD
from sklearn.feature_extraction.text import TfidfVectorizer

from pyspark.ml.feature import Tokenizer,StopWordsRemover,CountVectorizer
from pyspark.ml.clustering import LDA

from IPython.display import Image
```

```
sc = SparkContext().getOrCreate()
```

```
spark = SparkSession.builder.appName('Project Demo').getOrCreate()
```

Module 1

1. Apache Spark is an excellent tool for handling large datasets that come in various formats, such as structured, semi-structured, and unstructured data. Here's how Spark can be used to integrate these different types of data into a single view for comprehensive business analysis:

Data Collection: The first step in the process is data collection. Spark can read data from multiple sources like databases, live streams, and file systems.

- Structured data can be collected from traditional database systems using JDBC connectors.
- Semi-structured data such as JSON or XML can be directly ingested from file systems like HDFS or cloud storage services.
- Unstructured data, like logs or text files, can also be loaded into Spark.

Data Processing: Once data is collected, Spark provides several components and APIs for processing:

- DataFrame API: This is used to handle structured and semi-structured data. It allows you to manipulate data in a tabular form.
- RDD (Resilient Distributed Dataset): For lower-level transformations and actions on any type of data, RDDs provide fine-grained control.
- Datasets API: This combines the best features of RDDs and DataFrames, offering type-safety and object-oriented programming interfaces.

Data Integration: To create a single view from multiple data sources:

- Spark SQL: Use Spark SQL to run SQL queries across datasets. It can seamlessly mix SQL commands with Spark programs and can read data from Hive, Avro, Parquet, ORC, JSON, and JDBC.
- Join Operations: Data from different sources can be joined using common identifiers. For instance, customer transaction data can be joined with web traffic data on customer ID.
- UDFs (User Defined Functions): You can write custom transformations using UDFs in Spark SQL to handle data transformations that are not natively supported by SQL.

Data Analysis:

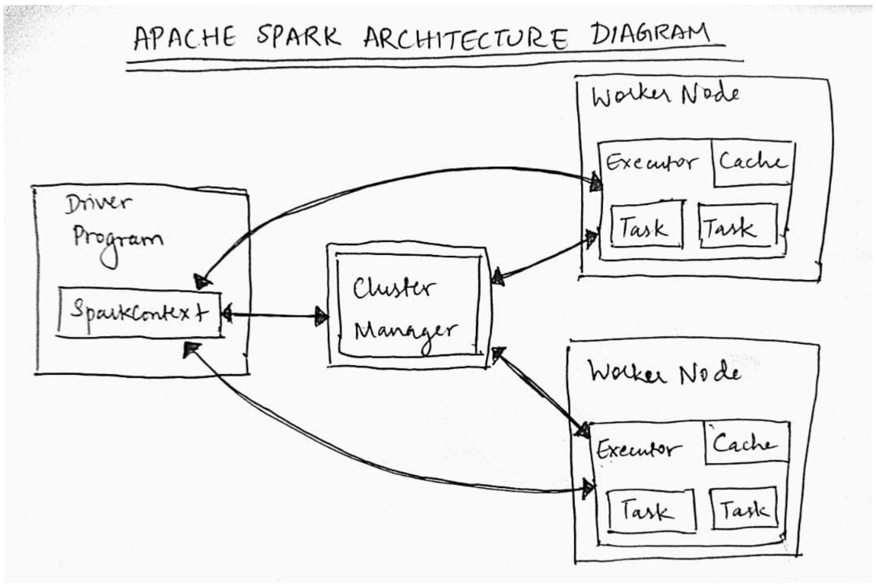
- MLlib: Use Spark’s machine learning library (MLlib) for predictive analytics and to uncover patterns in data.
- GraphX: For graph-based analysis, where relationships between data points need to be analyzed.
- Spark Streaming: For real-time data processing and analytics if the e-commerce platform requires analysis of live data streams.

Data Storage: After processing and analysis, the results can be stored back into a database, file system, or live dashboard systems for further action or visualization.

Spark Components Involved:

- Spark Core: The foundation of the system providing distributed task dispatching, scheduling, and basic I/O functionalities.
- Spark SQL: For seamless integration and querying of data.
- Spark MLlib: For applying machine learning algorithms.
- Spark Streaming: For processing real-time data streams.

```
image1 = "/content/da1.jpg"
display(Image(filename=image1))
```



2. Big data in healthcare is characterized by the 5 Vs, and these characteristics can be utilized to improve patient outcomes in several impactful ways: Big data in healthcare is characterized by the 5 Vs, and these characteristics can be utilized to improve patient outcomes in several impactful ways:

- **Volume:** Analyzing large volumes of electronic health records (EHRs) can help hospitals identify disease patterns, evaluate treatment effectiveness, and optimize resource allocation.
- **Velocity:** Real-time monitoring data from patients can alert providers to immediate health changes, enabling swift interventions that can save lives, especially in critical care.
- **Variety:** Integrating diverse data types, such as EHRs, imaging (X-rays, MRI), and operational data, allows for a comprehensive analysis, enhancing diagnostic accuracy and patient treatment plans.
- **Veracity:** Maintaining data accuracy is essential. Implementing data cleansing to ensure reliability improves treatment decisions and patient safety by reducing errors.
- **Value:** Data insights can lead to personalized medicine, where genetic information is used alongside health records to tailor treatments, improving effectiveness and reducing side effects.

By making use of these big data characteristics, healthcare institutions can improve diagnostics, predict patient risks, enhance preventive care, and optimize treatments, significantly boosting patient outcomes.

3. The Hadoop ecosystem is an ideal platform for managing and analyzing large volumes of data due to its capacity for distributed processing and scalability. A digital media company can leverage various components of the Hadoop ecosystem to understand viewer preferences, enhance content recommendation algorithms, and identify trending topics efficiently. Here's how each component can play a role:

i. Hadoop Distributed File System (HDFS)

- Role: HDFS serves as the backbone of the Hadoop ecosystem, providing a reliable and scalable storage system for handling large datasets. It stores data across multiple machines without prior organization, making it perfect for unstructured data such as video views and comments.
- Application: Store vast amounts of video interaction data, including views, likes, shares, and user-generated comments, distributed across different nodes to ensure high availability and fault tolerance.

ii. MapReduce

- Role: MapReduce is the processing engine in Hadoop. It processes large datasets in a parallel and distributed manner across the nodes in a Hadoop cluster.

- Application: Analyze large datasets by mapping out data elements and reducing them to results, like calculating the average watch time per video, identifying the most liked or shared content, or aggregating user demographic metrics.

iii. Apache Hive

- Role: Hive provides a SQL-like interface to query data stored in HDFS. It is designed for data summarization, query, and analysis.
- Application: Run queries to understand user behavior patterns, such as finding correlations between demographic factors and preferences in video content. Hive can also be used to perform complex analyses to track and predict trending topics based on viewer engagement metrics.

iv. Apache HBase

- Role: HBase is a NoSQL database that runs on top of HDFS. It is useful for real-time read/write access to large datasets.
- Application: Manage real-time data interactions, especially for features that require quick updates and retrievals such as counting likes or comments in real-time, which is crucial for recommending trending content dynamically.

v. Apache Pig

- Role: Pig is a high-level platform for creating MapReduce programs used with Hadoop. It is designed to handle any type of data, hence very suitable for handling semi-structured or unstructured data.
- Application: Transform and process user interaction data more easily than writing complex Java MapReduce programs. Pig allows for complex data transformations and analysis, such as extracting user engagement patterns and preparing data for machine learning models in recommendation systems.

vi. Apache Mahout

- Role: Mahout offers a framework for building scalable machine learning algorithms, primarily focusing on collaborative filtering, clustering, and classification.
- Application: Utilize viewer interaction data to build and train recommendation algorithms that suggest videos based on user preferences and viewing habits.

vii. Apache Spark

- Role: Although not originally a part of the Hadoop ecosystem, Spark can run on HDFS and is known for its speed and ability to handle real-time streaming.
- Application: Process live data streams of user interactions for immediate insights, which can be used to adjust content recommendations on the fly. Spark's machine learning library (MLlib) can be employed to enhance algorithms based on new user data.

viii. Apache Oozie

- Role: Oozie is a workflow scheduler system to manage Hadoop jobs.
- Application: Coordinate complex data processing jobs, such as sequencing data ingestion, processing, and analytical workflows, ensuring that all processes are completed in an orderly and timely manner.

By integrating these Hadoop components, the digital media company can effectively handle the diverse data types generated by user interactions, enabling scalable processing and sophisticated analysis to drive better content recommendations, identify trends, and understand viewer preferences comprehensively.

4.

```
rdd = sc.textFile('/content/drive/MyDrive/Big Data/Datasets/data.txt')
rdd
```

```
/content/drive/MyDrive/Big Data/Datasets/data.txt MapPartitionsRDD[1] at textFile at NativeMethodAccessorImpl.java:0
```

```
words =rdd.flatMap(lambda line:line.split(" "))
print(words.collect())
```

```
['this', 'is', 'a', 'sample', 'input', 'text', 'file', 'for', 'wordcount', 'program', 'wordcount', 'program', 'is', 'being', 'impler
```

```
words_count = words.map(lambda x:(x,1))
print(words_count.collect())
```

```
[('this', 1), ('is', 1), ('a', 1), ('sample', 1), ('input', 1), ('text', 1), ('file', 1), ('for', 1), ('wordcount', 1), ('program',
```

```
result =words_count.reduceByKey(lambda x,y:x+y)
print(result.collect())
```

```
[('this', 1), ('is', 6), ('input', 1), ('using', 1), ('stored', 1), ('hdfs', 2), ('an', 1), ('execution', 1), ('engine', 1), ('libra
for words,count in result.collect():
  print("words is:",words,"count is:",count)
```

```
    words is: this count is: 1
    words is: is count is: 6
    words is: input count is: 1
    words is: using count is: 1
    words is: stored count is: 1
    words is: hdfs count is: 2
    words is: an count is: 1
    words is: execution count is: 1
    words is: engine count is: 1
    words is: library count is: 1
    words is: of count is: 1
    words is: python count is: 1
    words is: used count is: 1
    words is: opensource count is: 1
    words is: a count is: 2
    words is: sample count is: 1
    words is: text count is: 2
    words is: file count is: 3
    words is: for count is: 2
    words is: wordcount count is: 2
    words is: program count is: 2
    words is: being count is: 1
    words is: implemented count is: 1
    words is: pyspark count is: 3
    words is: will count is: 1
    words is: be count is: 1
    words is: on count is: 1
    words is: distributed count is: 1
    words is: system count is: 1
    words is: spark count is: 1
    words is: data count is: 1
    words is: processing count is: 1
```

5. The two major components of the Hadoop ecosystem that form its core are the Hadoop Distributed File System (HDFS) and MapReduce. Both are essential for Hadoop's ability to handle large volumes of data efficiently. Here's an illustration of how each component works:

Hadoop Distributed File System (HDFS)

HDFS is designed to store very large data sets reliably, and to stream those data sets at high bandwidth to user applications. In a typical HDFS cluster, you have two types of nodes operating in a master-slave pattern: the NameNode (master) and several DataNodes (slaves).

Working:

- **NameNode:** The NameNode manages the file system namespace. It maintains the file system tree and metadata for all the files and directories in the system. This metadata is stored in memory, which provides very fast access to this information. The NameNode also knows the DataNodes on which all the blocks for a given file are located, though it does not store block locations persistently as this information is reconstructed from DataNodes when the system starts.
- **DataNodes:** These nodes manage the storage attached to the nodes that they run on. DataNodes are responsible for serving read and write requests from the file system’s clients. They also perform block creation, deletion, and replication upon instruction from the NameNode.
- **Process:** When a client needs to add a file to HDFS, the client splits the file into one or more blocks and these are stored in a set of DataNodes. The NameNode orchestrates the block storage and the replication policy, which ensures redundancy and fault tolerance.

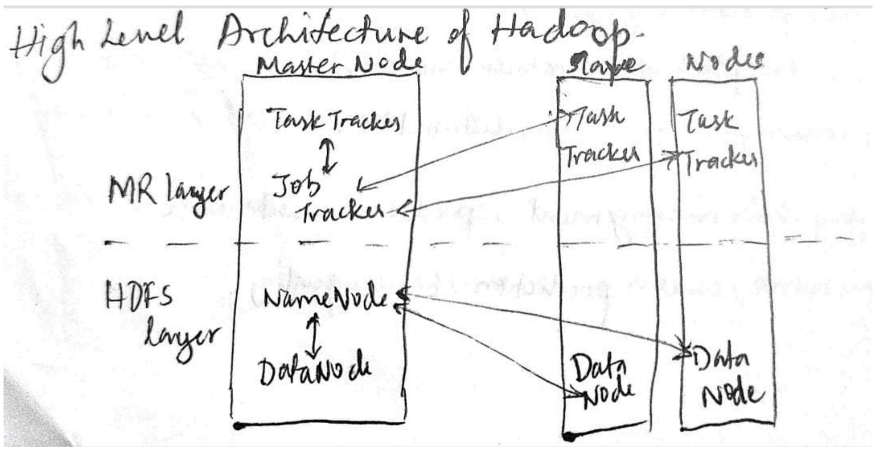
MapReduce

MapReduce is a programming model and processing technique that allows for massive scalability across hundreds or thousands of servers in a Hadoop cluster. It works by breaking down the application into many small fragments of work, each of which may be executed on any node in the cluster.

Working:

- **Job Tracker and Task Trackers:** The Job Tracker is the daemon service for submitting and tracking MapReduce jobs in Hadoop. Each slave node in the cluster has a Task Tracker daemon that communicates with the Job Tracker. The Job Tracker assigns tasks to different nodes depending on their availability, and the Task Trackers execute these tasks under its supervision.
- **Map Phase:** This phase processes the input data, typically in key-value pairs, to produce intermediate key-value pairs. For example, if analyzing text, the Map function counts the number of occurrences of each word.
- **Shuffle and Sort:** After the map tasks have processed the data, the results are "shuffled" (redistributed) across the reducers based on the output keys. Each reducer receives all values associated with the same key from the map outputs and sorts them to facilitate processing.
- **Reduce Phase:** This phase processes the shuffled data to perform a summary operation and produce a smaller, combined output. For example, in the word count scenario, it might sum up all counts per word received from the shuffle phase.
- **Output:** The output of the reduce phase is then written back to the HDFS, often as a single output file, though there can be multiple output files depending on how the MapReduce job is configured.

```
image2 = "/content/da2.jpg"
display(Image(filename=image2))
```



6.

```
sqlContext = SQLContext(spark)
```

```
/usr/local/lib/python3.10/dist-packages/pyspark/sql/context.py:113: FutureWarning: Deprecated in 3.0.0. Use SparkSession.builder.get
  warnings.warn(
```

A.

```
data= sqlContext.read.options(mode="DROPMALFORMED").json('/content/drive/MyDrive/Big Data/Datasets/iris.json')
data.show()
```

```
+-----------+----------+-----------+----------+-------+
|petalLength|petalWidth|sepalLength|sepalWidth|species|
+-----------+----------+-----------+----------+-------+
|        1.4|       0.2|        5.1|       3.5| setosa|
|        1.4|       0.2|        4.9|       3.0| setosa|
|        1.3|       0.2|        4.7|       3.2| setosa|
|        1.5|       0.2|        4.6|       3.1| setosa|
|        1.4|       0.2|        5.0|       3.6| setosa|
|        1.7|       0.4|        5.4|       3.9| setosa|
|        1.4|       0.3|        4.6|       3.4| setosa|
|        1.5|       0.2|        5.0|       3.4| setosa|
|        1.4|       0.2|        4.4|       2.9| setosa|
|        1.5|       0.1|        4.9|       3.1| setosa|
|        1.5|       0.2|        5.4|       3.7| setosa|
|        1.6|       0.2|        4.8|       3.4| setosa|
|        1.4|       0.1|        4.8|       3.0| setosa|
|        1.1|       0.1|        4.3|       3.0| setosa|
|        1.2|       0.2|        5.8|       4.0| setosa|
|        1.5|       0.4|        5.7|       4.4| setosa|
|        1.3|       0.4|        5.4|       3.9| setosa|
|        1.4|       0.3|        5.1|       3.5| setosa|
|        1.7|       0.3|        5.7|       3.8| setosa|
|        1.5|       0.3|        5.1|       3.8| setosa|
+-----------+----------+-----------+----------+-------+
only showing top 20 rows
```

```
data.printSchema()
```

```
root
 |-- petalLength: double (nullable = true)
 |-- petalWidth: double (nullable = true)
 |-- sepalLength: double (nullable = true)
 |-- sepalWidth: double (nullable = true)
 |-- species: string (nullable = true)
```

B.

```
data.show(10)
```

```
+-----------+----------+-----------+----------+-------+
|petalLength|petalWidth|sepalLength|sepalWidth|species|
+-----------+----------+-----------+----------+-------+
|        1.4|       0.2|        5.1|       3.5| setosa|
|        1.4|       0.2|        4.9|       3.0| setosa|
|        1.3|       0.2|        4.7|       3.2| setosa|
|        1.5|       0.2|        4.6|       3.1| setosa|
|        1.4|       0.2|        5.0|       3.6| setosa|
|        1.7|       0.4|        5.4|       3.9| setosa|
|        1.4|       0.3|        4.6|       3.4| setosa|
|        1.5|       0.2|        5.0|       3.4| setosa|
|        1.4|       0.2|        4.4|       2.9| setosa|
|        1.5|       0.1|        4.9|       3.1| setosa|
+-----------+----------+-----------+----------+-------+
only showing top 10 rows
```

C.

```
df = data.select("sepalLength")
sepalLength_rdd = df.rdd
sepalLength_rdd.collect()
```

```
[Row(sepalLength=5.1),
 Row(sepalLength=4.9),
 Row(sepalLength=4.7),
 Row(sepalLength=4.6),
 Row(sepalLength=5.0),
 Row(sepalLength=5.4),
 Row(sepalLength=4.6),
 Row(sepalLength=5.0),
 Row(sepalLength=4.4),
 Row(sepalLength=4.9),
 Row(sepalLength=5.4),
 Row(sepalLength=4.8),
 Row(sepalLength=4.8),
 Row(sepalLength=4.3),
 Row(sepalLength=5.8),
 Row(sepalLength=5.7),
 Row(sepalLength=5.4),
 Row(sepalLength=5.1),
 Row(sepalLength=5.7),
 Row(sepalLength=5.1),
 Row(sepalLength=5.4),
 Row(sepalLength=5.1),
 Row(sepalLength=4.6),
 Row(sepalLength=5.1),
 Row(sepalLength=4.8),
 Row(sepalLength=5.0),
 Row(sepalLength=5.0),
 Row(sepalLength=5.2),
 Row(sepalLength=5.2),
 Row(sepalLength=4.7),
 Row(sepalLength=4.8),
 Row(sepalLength=5.4),
 Row(sepalLength=5.2),
 Row(sepalLength=5.5),
 Row(sepalLength=4.9),
 Row(sepalLength=5.0),
 Row(sepalLength=5.5),
 Row(sepalLength=4.9),
 Row(sepalLength=4.4),
 Row(sepalLength=5.1),
 Row(sepalLength=5.0),
 Row(sepalLength=4.5),
 Row(sepalLength=4.4),
 Row(sepalLength=5.0),
 Row(sepalLength=5.1),
 Row(sepalLength=4.8),
 Row(sepalLength=5.1),
 Row(sepalLength=4.6),
 Row(sepalLength=5.3),
 Row(sepalLength=5.0),
 Row(sepalLength=7.0),
 Row(sepalLength=6.4),
 Row(sepalLength=6.9),
 Row(sepalLength=5.5),
 Row(sepalLength=6.5),
 Row(sepalLength=5.7),
 Row(sepalLength=6.3),
 Row(sepalLength=4.9),
```

D.

```
sepalLength_rdd.getNumPartitions()
```

```
    1
```

```
sepalLength_rdd = sepalLength_rdd.repartition(5)
sepalLength_rdd.getNumPartitions()
```

```
    5
```

E.

```
sepalLength_rdd.filter(lambda x: (x.sepalLength>5.4)).collect()
```

```
    [Row(sepalLength=5.8),
     Row(sepalLength=5.7),
     Row(sepalLength=5.7),
     Row(sepalLength=5.9),
     Row(sepalLength=6.0),
     Row(sepalLength=6.1),
     Row(sepalLength=5.6),
     Row(sepalLength=6.7),
     Row(sepalLength=5.6),
     Row(sepalLength=5.8),
     Row(sepalLength=6.2),
     Row(sepalLength=5.6),
     Row(sepalLength=6.5),
     Row(sepalLength=6.4),
     Row(sepalLength=6.8),
     Row(sepalLength=5.7),
     Row(sepalLength=5.8),
     Row(sepalLength=6.4),
     Row(sepalLength=6.5),
     Row(sepalLength=7.7),
     Row(sepalLength=7.7),
     Row(sepalLength=6.0),
     Row(sepalLength=5.9),
     Row(sepalLength=6.1),
     Row(sepalLength=6.3),
     Row(sepalLength=6.1),
     Row(sepalLength=6.4),
     Row(sepalLength=6.6),
     Row(sepalLength=6.8),
     Row(sepalLength=6.7),
     Row(sepalLength=6.0),
     Row(sepalLength=5.7),
     Row(sepalLength=6.9),
     Row(sepalLength=5.6),
     Row(sepalLength=7.7),
     Row(sepalLength=6.3),
     Row(sepalLength=6.7),
     Row(sepalLength=7.2),
     Row(sepalLength=6.2),
     Row(sepalLength=6.1),
     Row(sepalLength=6.4),
     Row(sepalLength=7.2),
     Row(sepalLength=5.5),
     Row(sepalLength=5.5),
     Row(sepalLength=5.5),
     Row(sepalLength=5.5),
     Row(sepalLength=5.8),
     Row(sepalLength=6.0),
     Row(sepalLength=6.0),
     Row(sepalLength=6.7),
     Row(sepalLength=6.3),
     Row(sepalLength=5.6),
     Row(sepalLength=5.5),
     Row(sepalLength=7.4),
     Row(sepalLength=7.9),
     Row(sepalLength=6.4),
     Row(sepalLength=6.3),
     Row(sepalLength=6.1),
```

F.

```
sepalLength_rdd.flatMap(lambda x: x*3).collect()
```

```
    [5.4,
     5.4,
     5.4,
     4.8,
     4.8,
      4.8,
      4.8,
      4.8,
      4.8,
      4.3,
      4.3,
      4.3,
      5.8,
      5.8,
      5.8,
      5.7,
      5.7,
      5.7,
      5.4,
      5.4,
      5.4,
      5.1,
      5.1,
      5.1,
      5.7,
      5.7,
      5.7,
      5.1,
      5.1,
      5.1,
      5.0,
      5.0,
      5.0,
      5.9,
      5.9,
      5.9,
      6.0,
      6.0,
      6.0,
      6.1,
      6.1,
      6.1,
      5.6,
      5.6,
      5.6,
      6.7,
      6.7,
      6.7,
      5.6,
      5.6,
      5.6,
      5.8,
      5.8,
      5.8,
      6.2,
      6.2,
      6.2,
      5.6,
```

G.

```
sepalLength_rdd.distinct().collect()
```

```
     [Row(sepalLength=5.6),
      Row(sepalLength=4.7),
      Row(sepalLength=6.6),
      Row(sepalLength=7.4),
      Row(sepalLength=4.5),
      Row(sepalLength=7.3),
      Row(sepalLength=4.8),
      Row(sepalLength=5.7),
      Row(sepalLength=6.7),
      Row(sepalLength=6.5),
      Row(sepalLength=5.5),
      Row(sepalLength=4.9),
      Row(sepalLength=7.6),
      Row(sepalLength=5.8),
      Row(sepalLength=5.1),
      Row(sepalLength=5.9),
      Row(sepalLength=6.1),
      Row(sepalLength=6.8),
      Row(sepalLength=7.7),
      Row(sepalLength=6.9),
      Row(sepalLength=4.3),
      Row(sepalLength=5.0),
      Row(sepalLength=6.0),
      Row(sepalLength=6.2),
      Row(sepalLength=5.2),
      Row(sepalLength=4.4),
      Row(sepalLength=7.9),
      Row(sepalLength=7.1),
      Row(sepalLength=5.4),
      Row(sepalLength=6.4),
      Row(sepalLength=4.6),
      Row(sepalLength=6.3),
      Row(sepalLength=7.2),
      Row(sepalLength=5.3),
      Row(sepalLength=7.0)]
```

H.

```
rdd1 = data.select("sepalLength","species").rdd
rdd2 = data.select("petalLength","species").rdd
```

```
rdd_join = rdd1.join(rdd2)
rdd_join.collect()
```

```
     [(4.9, ('setosa', 'versicolor')),
      (4.9, ('setosa', 'versicolor')),
      (4.9, ('setosa', 'virginica')),
      (4.9, ('setosa', 'virginica')),
      (4.9, ('setosa', 'virginica')),
      (4.9, ('setosa', 'versicolor')),
      (4.9, ('setosa', 'versicolor')),
      (4.9, ('setosa', 'virginica')),
      (4.9, ('setosa', 'virginica')),
      (4.9, ('setosa', 'virginica')),
      (4.9, ('setosa', 'versicolor')),
      (4.9, ('setosa', 'versicolor')),
      (4.9, ('setosa', 'virginica')),
      (4.9, ('setosa', 'virginica')),
      (4.9, ('setosa', 'virginica')),
      (4.9, ('setosa', 'versicolor')),
      (4.9, ('setosa', 'versicolor')),
      (4.9, ('setosa', 'virginica')),
      (4.9, ('setosa', 'virginica')),
      (4.9, ('setosa', 'virginica')),
      (4.9, ('versicolor', 'versicolor')),
      (4.9, ('versicolor', 'versicolor')),
      (4.9, ('versicolor', 'virginica')),
      (4.9, ('versicolor', 'virginica')),
      (4.9, ('versicolor', 'virginica')),
      (4.9, ('virginica', 'versicolor')),
      (4.9, ('virginica', 'versicolor')),
      (4.9, ('virginica', 'virginica')),
      (4.9, ('virginica', 'virginica')),
      (4.9, ('virginica', 'virginica')),
      (4.7, ('setosa', 'versicolor')),
      (4.7, ('setosa', 'versicolor')),
      (4.7, ('setosa', 'versicolor')),
      (4.7, ('setosa', 'versicolor')),
      (4.7, ('setosa', 'versicolor')),
      (4.7, ('setosa', 'versicolor')),
      (4.7, ('setosa', 'versicolor')),
      (4.7, ('setosa', 'versicolor')),
      (4.7, ('setosa', 'versicolor')),
      (4.7, ('setosa', 'versicolor')),
      (4.6, ('setosa', 'versicolor')),
      (4.6, ('setosa', 'versicolor')),
      (4.6, ('setosa', 'versicolor')),
      (4.6, ('setosa', 'versicolor')),
      (4.6, ('setosa', 'versicolor')),
      (4.6, ('setosa', 'versicolor')),
      (4.6, ('setosa', 'versicolor')),
      (4.6, ('setosa', 'versicolor')),
      (4.6, ('setosa', 'versicolor')),
      (4.6, ('setosa', 'versicolor')),
      (4.6, ('setosa', 'versicolor')),
      (4.6, ('setosa', 'versicolor')),
      (4.4, ('setosa', 'versicolor')),
      (4.4, ('setosa', 'versicolor')),
      (4.4, ('setosa', 'versicolor')),
      (4.4, ('setosa', 'versicolor')),
      (4.4, ('setosa', 'versicolor')),
      (4.4, ('setosa', 'versicolor')),
```

```
rdd_left_outer_join = rdd1.leftOuterJoin(rdd2)
rdd_left_outer_join.collect()
```

```
     [(4.9, ('setosa', 'versicolor')),
      (4.9, ('setosa', 'versicolor')),
      (4.9, ('setosa', 'virginica')),
      (4.9, ('setosa', 'virginica')),
      (4.9, ('setosa', 'virginica')),
      (4.9, ('setosa', 'versicolor')),
      (4.9, ('setosa', 'versicolor')),
      (4.9, ('setosa', 'virginica')),
      (4.9, ('setosa', 'virginica')),
      (4.9, ('setosa', 'virginica')),
      (4.9, ('setosa', 'versicolor')),
      (4.9, ('setosa', 'versicolor')),
      (4.9, ('setosa', 'virginica')),
      (4.9, ('setosa', 'virginica')),
      (4.9, ('setosa', 'virginica')),
      (4.9, ('setosa', 'versicolor')),
      (4.9, ('setosa', 'versicolor')),
      (4.9, ('setosa', 'virginica')),
      (4.9, ('setosa', 'virginica')),
      (4.9, ('setosa', 'virginica')),
      (4.9, ('versicolor', 'versicolor')),
      (4.9, ('versicolor', 'versicolor')),
      (4.9, ('versicolor', 'virginica')),
      (4.9, ('versicolor', 'virginica')),
      (4.9, ('versicolor', 'virginica')),
      (4.9, ('virginica', 'versicolor')),
      (4.9, ('virginica', 'versicolor')),
      (4.9, ('virginica', 'virginica')),
      (4.9, ('virginica', 'virginica')),
      (4.9, ('virginica', 'virginica')),
      (4.7, ('setosa', 'versicolor')),
      (4.7, ('setosa', 'versicolor')),
      (4.7, ('setosa', 'versicolor')),
      (4.7, ('setosa', 'versicolor')),
      (4.7, ('setosa', 'versicolor')),
      (4.7, ('setosa', 'versicolor')),
      (4.7, ('setosa', 'versicolor')),
      (4.7, ('setosa', 'versicolor')),
      (4.7, ('setosa', 'versicolor')),
      (4.7, ('setosa', 'versicolor')),
      (4.6, ('setosa', 'versicolor')),
      (4.6, ('setosa', 'versicolor')),
      (4.6, ('setosa', 'versicolor')),
      (4.6, ('setosa', 'versicolor')),
      (4.6, ('setosa', 'versicolor')),
      (4.6, ('setosa', 'versicolor')),
      (4.6, ('setosa', 'versicolor')),
      (4.6, ('setosa', 'versicolor')),
      (4.6, ('setosa', 'versicolor')),
      (4.6, ('setosa', 'versicolor')),
      (4.6, ('setosa', 'versicolor')),
      (4.6, ('setosa', 'versicolor')),
      (4.4, ('setosa', 'versicolor')),
      (4.4, ('setosa', 'versicolor')),
      (4.4, ('setosa', 'versicolor')),
      (4.4, ('setosa', 'versicolor')),
      (4.4, ('setosa', 'versicolor')),
      (4.4, ('setosa', 'versicolor')),
```

```
rdd_right_outer_join = rdd1.rightOuterJoin(rdd2)
rdd_right_outer_join.collect()
```

```
[(4.9, ('setosa', 'versicolor')),
 (4.9, ('setosa', 'versicolor')),
 (4.9, ('setosa', 'virginica')),
 (4.9, ('setosa', 'virginica')),
 (4.9, ('setosa', 'virginica')),
 (4.9, ('setosa', 'versicolor')),
 (4.9, ('setosa', 'versicolor')),
 (4.9, ('setosa', 'virginica')),
 (4.9, ('setosa', 'virginica')),
 (4.9, ('setosa', 'virginica')),
 (4.9, ('setosa', 'versicolor')),
 (4.9, ('setosa', 'versicolor')),
 (4.9, ('setosa', 'virginica')),
 (4.9, ('setosa', 'virginica')),
 (4.9, ('setosa', 'virginica')),
 (4.9, ('setosa', 'versicolor')),
 (4.9, ('setosa', 'versicolor')),
 (4.9, ('setosa', 'virginica')),
 (4.9, ('setosa', 'virginica')),
 (4.9, ('setosa', 'virginica')),
 (4.9, ('versicolor', 'versicolor')),
 (4.9, ('versicolor', 'versicolor')),
 (4.9, ('versicolor', 'virginica')),
 (4.9, ('versicolor', 'virginica')),
 (4.9, ('versicolor', 'virginica')),
 (4.9, ('virginica', 'versicolor')),
 (4.9, ('virginica', 'versicolor')),
 (4.9, ('virginica', 'virginica')),
 (4.9, ('virginica', 'virginica')),
 (4.9, ('virginica', 'virginica')),
 (4.7, ('setosa', 'versicolor')),
 (4.7, ('setosa', 'versicolor')),
 (4.7, ('setosa', 'versicolor')),
 (4.7, ('setosa', 'versicolor')),
 (4.7, ('setosa', 'versicolor')),
 (4.7, ('setosa', 'versicolor')),
 (4.7, ('setosa', 'versicolor')),
 (4.7, ('setosa', 'versicolor')),
 (4.7, ('setosa', 'versicolor')),
 (4.7, ('setosa', 'versicolor')),
 (4.6, ('setosa', 'versicolor')),
 (4.6, ('setosa', 'versicolor')),
 (4.6, ('setosa', 'versicolor')),
 (4.6, ('setosa', 'versicolor')),
 (4.6, ('setosa', 'versicolor')),
 (4.6, ('setosa', 'versicolor')),
 (4.6, ('setosa', 'versicolor')),
 (4.6, ('setosa', 'versicolor')),
 (4.6, ('setosa', 'versicolor')),
 (4.6, ('setosa', 'versicolor')),
 (4.6, ('setosa', 'versicolor')),
 (4.6, ('setosa', 'versicolor')),
 (4.4, ('setosa', 'versicolor')),
 (4.4, ('setosa', 'versicolor')),
 (4.4, ('setosa', 'versicolor')),
 (4.4, ('setosa', 'versicolor')),
 (4.4, ('setosa', 'versicolor')),
 (4.4, ('setosa', 'versicolor')),
```

```
rdd_full_outer_join = rdd1.fullOuterJoin(rdd2)
rdd_full_outer_join.collect()
```

```
[(4.9, ('setosa', 'versicolor')),
 (4.9, ('setosa', 'versicolor')),
 (4.9, ('setosa', 'virginica')),
 (4.9, ('setosa', 'virginica')),
 (4.9, ('setosa', 'virginica')),
 (4.9, ('setosa', 'versicolor')),
 (4.9, ('setosa', 'versicolor')),
 (4.9, ('setosa', 'virginica')),
 (4.9, ('setosa', 'virginica')),
 (4.9, ('setosa', 'virginica')),
 (4.9, ('setosa', 'versicolor')),
 (4.9, ('setosa', 'versicolor')),
 (4.9, ('setosa', 'virginica')),
 (4.9, ('setosa', 'virginica')),
 (4.9, ('setosa', 'virginica')),
 (4.9, ('setosa', 'versicolor')),
 (4.9, ('setosa', 'versicolor')),
 (4.9, ('setosa', 'virginica')),
 (4.9, ('setosa', 'virginica')),
 (4.9, ('setosa', 'virginica')),
 (4.9, ('versicolor', 'versicolor')),
 (4.9, ('versicolor', 'versicolor')),
 (4.9, ('versicolor', 'virginica')),
 (4.9, ('versicolor', 'virginica')),
 (4.9, ('versicolor', 'virginica')),
 (4.9, ('virginica', 'versicolor')),
 (4.9, ('virginica', 'versicolor')),
 (4.9, ('virginica', 'virginica')),
 (4.9, ('virginica', 'virginica')),
 (4.9, ('virginica', 'virginica')),
 (4.7, ('setosa', 'versicolor')),
 (4.7, ('setosa', 'versicolor')),
 (4.7, ('setosa', 'versicolor')),
 (4.7, ('setosa', 'versicolor')),
 (4.7, ('setosa', 'versicolor')),
 (4.7, ('setosa', 'versicolor')),
 (4.7, ('setosa', 'versicolor')),
 (4.7, ('setosa', 'versicolor')),
 (4.7, ('setosa', 'versicolor')),
 (4.7, ('setosa', 'versicolor')),
 (4.6, ('setosa', 'versicolor')),
 (4.6, ('setosa', 'versicolor')),
 (4.6, ('setosa', 'versicolor')),
 (4.6, ('setosa', 'versicolor')),
 (4.6, ('setosa', 'versicolor')),
 (4.6, ('setosa', 'versicolor')),
 (4.6, ('setosa', 'versicolor')),
 (4.6, ('setosa', 'versicolor')),
 (4.6, ('setosa', 'versicolor')),
 (4.6, ('setosa', 'versicolor')),
 (4.6, ('setosa', 'versicolor')),
 (4.6, ('setosa', 'versicolor')),
 (4.4, ('setosa', 'versicolor')),
 (4.4, ('setosa', 'versicolor')),
 (4.4, ('setosa', 'versicolor')),
 (4.4, ('setosa', 'versicolor')),
 (4.4, ('setosa', 'versicolor')),
 (4.4, ('setosa', 'versicolor')),
```

I.

```
rdd_cache = sepalLength_rdd.cache()
rdd_cache.collect()
```

```
[Row(sepalLength=5.4),
 Row(sepalLength=4.8),
 Row(sepalLength=4.8),
 Row(sepalLength=4.3),
 Row(sepalLength=5.8),
 Row(sepalLength=5.7),
 Row(sepalLength=5.4),
 Row(sepalLength=5.1),
 Row(sepalLength=5.7),
 Row(sepalLength=5.1),
 Row(sepalLength=5.0),
 Row(sepalLength=5.9),
 Row(sepalLength=6.0),
 Row(sepalLength=6.1),
 Row(sepalLength=5.6),
 Row(sepalLength=6.7),
 Row(sepalLength=5.6),
 Row(sepalLength=5.8),
 Row(sepalLength=6.2),
 Row(sepalLength=5.6),
 Row(sepalLength=6.5),
 Row(sepalLength=6.4),
 Row(sepalLength=6.8),
 Row(sepalLength=5.7),
 Row(sepalLength=5.8),
 Row(sepalLength=6.4),
 Row(sepalLength=6.5),
 Row(sepalLength=7.7),
 Row(sepalLength=7.7),
 Row(sepalLength=6.0),
 Row(sepalLength=5.4),
 Row(sepalLength=5.1),
 Row(sepalLength=4.6),
 Row(sepalLength=5.1),
 Row(sepalLength=4.8),
 Row(sepalLength=5.0),
 Row(sepalLength=5.0),
 Row(sepalLength=5.2),
 Row(sepalLength=5.2),
 Row(sepalLength=4.7),
 Row(sepalLength=5.9),
 Row(sepalLength=6.1),
 Row(sepalLength=6.3),
 Row(sepalLength=6.1),
 Row(sepalLength=6.4),
 Row(sepalLength=6.6),
 Row(sepalLength=6.8),
 Row(sepalLength=6.7),
 Row(sepalLength=6.0),
 Row(sepalLength=5.7),
 Row(sepalLength=6.9),
 Row(sepalLength=5.6),
 Row(sepalLength=7.7),
 Row(sepalLength=6.3),
 Row(sepalLength=6.7),
 Row(sepalLength=7.2),
 Row(sepalLength=6.2),
 Row(sepalLength=6.1),
```

Module 2

7.

A.

```
book_schema = StructType([StructField("Book_Id", IntegerType(), True),
                     StructField("Book_Name", StringType(), True),
                     StructField("Author_Name", StringType(), True),
                     StructField("Price", DoubleType(), True)])

book_data = [
    (1, "Handmaid's Tale", "Margaret Atwood", 500.0),
    (2, "To Sleep In A Sea Of Stars", "Christopher Paolini", 560.0),
    (3, "Mistborn", "Brandon Sanderson", 120.0),
    (4, "Crooked Kingdom", "Leigh Bardugo", 220.0),
    (5, "Inheritance", "Christipher Paolini", 520.0)
]

book_details = spark.createDataFrame(book_data,book_schema)

print(book_details.printSchema())

bill_schema = StructType([StructField("Bill_Number",IntegerType(),True),
                     StructField("Book_Id",IntegerType(),True),
                     StructField("Qty",StringType(),True)])

bill_data = [
    (100,1,"2"),
    (101,3,"4"),
    (102,2,"1"),
    (103,5,"6"),
    (104,4,"2")
]

bill_details = spark.createDataFrame(bill_data,bill_schema)

print(bill_details.printSchema())
```

```
root
 |-- Book_Id: integer (nullable = true)
 |-- Book_Name: string (nullable = true)
 |-- Author_Name: string (nullable = true)
 |-- Price: double (nullable = true)

None
root
 |-- Bill_Number: integer (nullable = true)
 |-- Book_Id: integer (nullable = true)
 |-- Qty: string (nullable = true)

None
```

```
book_details.show()
```

```
+-------+--------------------+-------------------+-----+
|Book_Id|           Book_Name|        Author_Name|Price|
+-------+--------------------+-------------------+-----+
|      1|     Handmaid's Tale|    Margaret Atwood|500.0|
|      2|To Sleep In A Sea...|Christopher Paolini|560.0|
|      3|            Mistborn|  Brandon Sanderson|120.0|
|      4|     Crooked Kingdom|      Leigh Bardugo|220.0|
|      5|         Inheritance|Christipher Paolini|520.0|
+-------+--------------------+-------------------+-----+
```

B.

```
(book_details.filter('Price > 500.0')).show()
```

```
+-------+--------------------+-------------------+-----+
|Book_Id|           Book_Name|        Author_Name|Price|
+-------+--------------------+-------------------+-----+
|      2|To Sleep In A Sea...|Christopher Paolini|560.0|
|      5|         Inheritance|Christipher Paolini|520.0|
+-------+--------------------+-------------------+-----+
```

C.

```
(book_details.select('Book_Id')).count()
```

```
5
```

D.

```
joined_table_inner = book_details.join(bill_details,"Book_Id","inner")
joined_table_inner.show()
```

```
+-------+--------------------+-------------------+-----+-----------+---+
|Book_Id|           Book_Name|        Author_Name|Price|Bill_Number|Qty|
+-------+--------------------+-------------------+-----+-----------+---+
|      1|     Handmaid's Tale|    Margaret Atwood|500.0|        100|  2|
|      2|To Sleep In A Sea...|Christopher Paolini|560.0|        102|  1|
|      3|            Mistborn|  Brandon Sanderson|120.0|        101|  4|
|      4|     Crooked Kingdom|      Leigh Bardugo|220.0|        104|  2|
|      5|         Inheritance|Christipher Paolini|520.0|        103|  6|
+-------+--------------------+-------------------+-----+-----------+---+
```

```
joined_table_left = book_details.join(bill_details,"Book_Id","left")
joined_table_left.show()
```

```
+-------+--------------------+-------------------+-----+-----------+---+
|Book_Id|           Book_Name|        Author_Name|Price|Bill_Number|Qty|
+-------+--------------------+-------------------+-----+-----------+---+
|      1|     Handmaid's Tale|    Margaret Atwood|500.0|        100|  2|
|      2|To Sleep In A Sea...|Christopher Paolini|560.0|        102|  1|
|      3|            Mistborn|  Brandon Sanderson|120.0|        101|  4|
|      5|         Inheritance|Christipher Paolini|520.0|        103|  6|
|      4|     Crooked Kingdom|      Leigh Bardugo|220.0|        104|  2|
+-------+--------------------+-------------------+-----+-----------+---+
```

```
joined_table_right = book_details.join(bill_details,"Book_Id","right")
joined_table_right.show()
```

```
+-------+--------------------+-------------------+-----+-----------+---+
|Book_Id|           Book_Name|        Author_Name|Price|Bill_Number|Qty|
+-------+--------------------+-------------------+-----+-----------+---+
|      1|     Handmaid's Tale|    Margaret Atwood|500.0|        100|  2|
|      3|            Mistborn|  Brandon Sanderson|120.0|        101|  4|
|      5|         Inheritance|Christipher Paolini|520.0|        103|  6|
|      4|     Crooked Kingdom|      Leigh Bardugo|220.0|        104|  2|
|      2|To Sleep In A Sea...|Christopher Paolini|560.0|        102|  1|
+-------+--------------------+-------------------+-----+-----------+---+
```

```
joined_table_full = book_details.join(bill_details,"Book_Id","full")
joined_table_full.show()
```

```
+-------+--------------------+-------------------+-----+-----------+---+
|Book_Id|           Book_Name|        Author_Name|Price|Bill_Number|Qty|
+-------+--------------------+-------------------+-----+-----------+---+
|      1|     Handmaid's Tale|    Margaret Atwood|500.0|        100|  2|
|      2|To Sleep In A Sea...|Christopher Paolini|560.0|        102|  1|
|      3|            Mistborn|  Brandon Sanderson|120.0|        101|  4|
|      4|     Crooked Kingdom|      Leigh Bardugo|220.0|        104|  2|
|      5|         Inheritance|Christipher Paolini|520.0|        103|  6|
+-------+--------------------+-------------------+-----+-----------+---+
```

```
joined_table_outer = book_details.join(bill_details,"Book_Id","outer")
joined_table_outer.show()
```

```
+-------+--------------------+-------------------+-----+-----------+---+
|Book_Id|           Book_Name|        Author_Name|Price|Bill_Number|Qty|
+-------+--------------------+-------------------+-----+-----------+---+
|      1|     Handmaid's Tale|    Margaret Atwood|500.0|        100|  2|
|      2|To Sleep In A Sea...|Christopher Paolini|560.0|        102|  1|
|      3|            Mistborn|  Brandon Sanderson|120.0|        101|  4|
|      4|     Crooked Kingdom|      Leigh Bardugo|220.0|        104|  2|
|      5|         Inheritance|Christipher Paolini|520.0|        103|  6|
+-------+--------------------+-------------------+-----+-----------+---+
```

```
joined_table_semi = book_details.join(bill_details,"Book_Id","semi")
joined_table_semi.show()
```

```
+-------+--------------------+-------------------+-----+
|Book_Id|           Book_Name|        Author_Name|Price|
+-------+--------------------+-------------------+-----+
|      1|     Handmaid's Tale|    Margaret Atwood|500.0|
|      2|To Sleep In A Sea...|Christopher Paolini|560.0|
|      3|            Mistborn|  Brandon Sanderson|120.0|
|      4|     Crooked Kingdom|      Leigh Bardugo|220.0|
|      5|         Inheritance|Christipher Paolini|520.0|
+-------+--------------------+-------------------+-----+
joined_table_semi = book_details.join(bill_details,"Book_Id","anti")
joined_table_semi.show()
```

```
+-------+---------+-----------+-----+
|Book_Id|Book_Name|Author_Name|Price|
+-------+---------+-----------+-----+
+-------+---------+-----------+-----+
```

E.

```
bill_details.count()
```

```
5
```

F.

```
bill_details.select('Book_Id').distinct().count()
```

```
5
```

G.

```
bill_details_updated = joined_table_inner.withColumn("Total_cost",func.try_multiply("Price","Qty"))
bill_details_updated.show()
```

```
+-------+--------------------+-------------------+-----+-----------+---+----------+
|Book_Id|           Book_Name|        Author_Name|Price|Bill_Number|Qty|Total_cost|
+-------+--------------------+-------------------+-----+-----------+---+----------+
|      1|     Handmaid's Tale|    Margaret Atwood|500.0|        100|  2|    1000.0|
|      2|To Sleep In A Sea...|Christopher Paolini|560.0|        102|  1|     560.0|
|      3|            Mistborn|  Brandon Sanderson|120.0|        101|  4|     480.0|
|      4|     Crooked Kingdom|      Leigh Bardugo|220.0|        104|  2|     440.0|
|      5|         Inheritance|Christipher Paolini|520.0|        103|  6|    3120.0|
+-------+--------------------+-------------------+-----+-----------+---+----------+
```

H.

```
bill_details_updated.select("Price").orderBy(bill_details_updated.Price.desc()).show(5)
```

```
+-----+
|Price|
+-----+
|560.0|
|520.0|
|500.0|
|220.0|
|120.0|
+-----+
```

9.

A.

```
book_details.createOrReplaceTempView("book_table")
bill_details.createOrReplaceTempView("bill_table")
```

```
all_records = spark.sql("SELECT * FROM book_table")
all_records.show()
```

```
+-------+--------------------+-------------------+-----+
|Book_Id|           Book_Name|        Author_Name|Price|
+-------+--------------------+-------------------+-----+
|      1|     Handmaid's Tale|    Margaret Atwood|500.0|
|      2|To Sleep In A Sea...|Christopher Paolini|560.0|
|      3|            Mistborn|  Brandon Sanderson|120.0|
|      4|     Crooked Kingdom|      Leigh Bardugo|220.0|
|      5|         Inheritance|Christipher Paolini|520.0|
+-------+--------------------+-------------------+-----+
```

```
all_records = spark.sql("SELECT * FROM bill_table")
all_records.show()
```

```
    +-----------+-------+---+
    |Bill_Number|Book_Id|Qty|
    +-----------+-------+---+
    |        100|      1|  2|
    |        101|      3|  4|
    |        102|      2|  1|
    |        103|      5|  6|
    |        104|      4|  2|
    +-----------+-------+---+
```

B.

```
starts_S = spark.sql("SELECT Book_Name FROM book_table WHERE Book_Name LIKE 'S%'")
starts_S.show()
```

```
    +---------+
    |Book_Name|
    +---------+
    +---------+
```

C.

```
joined_sql = spark.sql("SELECT * FROM book_table AS b1 INNER JOIN book_table AS b2 ON b1.Book_Id <> b2.Book_Id")
joined_sql.show()
```

```
    +-------+--------------------+-------------------+-----+-------+--------------------+-------------------+-----+
    |Book_Id|           Book_Name|        Author_Name|Price|Book_Id|           Book_Name|        Author_Name|Price|
    +-------+--------------------+-------------------+-----+-------+--------------------+-------------------+-----+
    |      1|     Handmaid's Tale|    Margaret Atwood|500.0|      2|To Sleep In A Sea...|Christopher Paolini|560.0|
    |      2|To Sleep In A Sea...|Christopher Paolini|560.0|      1|     Handmaid's Tale|    Margaret Atwood|500.0|
    |      1|     Handmaid's Tale|    Margaret Atwood|500.0|      3|            Mistborn|  Brandon Sanderson|120.0|
    |      1|     Handmaid's Tale|    Margaret Atwood|500.0|      4|     Crooked Kingdom|      Leigh Bardugo|220.0|
    |      1|     Handmaid's Tale|    Margaret Atwood|500.0|      5|         Inheritance|Christipher Paolini|520.0|
    |      2|To Sleep In A Sea...|Christopher Paolini|560.0|      3|            Mistborn|  Brandon Sanderson|120.0|
    |      2|To Sleep In A Sea...|Christopher Paolini|560.0|      4|     Crooked Kingdom|      Leigh Bardugo|220.0|
    |      2|To Sleep In A Sea...|Christopher Paolini|560.0|      5|         Inheritance|Christipher Paolini|520.0|
    |      3|            Mistborn|  Brandon Sanderson|120.0|      1|     Handmaid's Tale|    Margaret Atwood|500.0|
    |      3|            Mistborn|  Brandon Sanderson|120.0|      2|To Sleep In A Sea...|Christopher Paolini|560.0|
    |      4|     Crooked Kingdom|      Leigh Bardugo|220.0|      1|     Handmaid's Tale|    Margaret Atwood|500.0|
    |      4|     Crooked Kingdom|      Leigh Bardugo|220.0|      2|To Sleep In A Sea...|Christopher Paolini|560.0|
    |      5|         Inheritance|Christipher Paolini|520.0|      1|     Handmaid's Tale|    Margaret Atwood|500.0|
    |      5|         Inheritance|Christipher Paolini|520.0|      2|To Sleep In A Sea...|Christopher Paolini|560.0|
    |      3|            Mistborn|  Brandon Sanderson|120.0|      4|     Crooked Kingdom|      Leigh Bardugo|220.0|
    |      3|            Mistborn|  Brandon Sanderson|120.0|      5|         Inheritance|Christipher Paolini|520.0|
    |      4|     Crooked Kingdom|      Leigh Bardugo|220.0|      3|            Mistborn|  Brandon Sanderson|120.0|
    |      4|     Crooked Kingdom|      Leigh Bardugo|220.0|      5|         Inheritance|Christipher Paolini|520.0|
    |      5|         Inheritance|Christipher Paolini|520.0|      3|            Mistborn|  Brandon Sanderson|120.0|
    |      5|         Inheritance|Christipher Paolini|520.0|      4|     Crooked Kingdom|      Leigh Bardugo|220.0|
    +-------+--------------------+-------------------+-----+-------+--------------------+-------------------+-----+
```

E.

```
distinct_ids = spark.sql("SELECT distinct(Book_Id) FROM bill_table")
distinct_ids.show()
```

```
    +-------+
    |Book_Id|
    +-------+
    |      1|
    |      3|
    |      5|
    |      4|
    |      2|
    +-------+
```

F.

```
min_price = spark.sql("SELECT min(Price) as Min_Price FROM book_table")
min_price.show()
```

```
    +---------+
    |Min_Price|
    +---------+
    |    120.0|
    +---------+
```

G.

```
max_qty = spark.sql("SELECT max(Qty) AS Max_Qty FROM bill_table")
max_qty.show()
```

```
+-------+
|Max_Qty|
+-------+
|      6|
+-------+
```

H.

```
count_book_id = spark.sql("SELECT Book_Id, count(Book_Id) FROM bill_table GROUP BY Book_Id")
count_book_id.show()
```

```
+-------+--------------+
|Book_Id|count(Book_Id)|
+-------+--------------+
|      1|             1|
|      3|             1|
|      5|             1|
|      4|             1|
|      2|             1|
+-------+--------------+
```

9.

```
labels = np.array([1,0,1,1,0])
predictions = np.array([0.9,0.3,0.8,0.6,0.2])
start = time.time()
summation = np.sum(np.square(labels-predictions))
n = len(labels)
error = np.multiply(1/n,summation)
print(f"Error value is: {error}")
end = time.time()
print(f"Total computation time is: {end-start}")
```

```
Error value is: 0.06799999999999999
Total computation time is: 0.007539033889770508
```

10.

```
numpy_matrix = np.arange(30).reshape(6,5)
print(f"{numpy_matrix}\n")
print(numpy_matrix * 2)
```

```
[[ 0  1  2  3  4]
 [ 5  6  7  8  9]
 [10 11 12 13 14]
 [15 16 17 18 19]
 [20 21 22 23 24]
 [25 26 27 28 29]]

[[ 0  2  4  6  8]
 [10 12 14 16 18]
 [20 22 24 26 28]
 [30 32 34 36 38]
 [40 42 44 46 48]
 [50 52 54 56 58]]
```

11.

```
sc = SparkContext.getOrCreate()
x = np.arange(25).reshape(5,5)
print(x)
mat_rdd = sc.parallelize(x)
beta = np.array([4,5,3,2,6])
result = mat_rdd.map(lambda x : x*beta).reduce(lambda x,y: x+y)
print(f"Theta value is: {result}")
```

```
[[ 0  1  2  3  4]
 [ 5  6  7  8  9]
 [10 11 12 13 14]
 [15 16 17 18 19]
 [20 21 22 23 24]]
Theta value is: [200 275 180 130 420]
```

12.

```
data_mat = np.array([[1,2], [3,4]])
inv_data_mat = linalg.inv(data_mat)
print(f"{inv_data_mat}\n")
print(data_mat)
```

```
[[-2.   1. ]
 [ 1.5 -0.5]]

[[1 2]
 [3 4]]
```

```
a = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
b = np.array([[1], [2], [3]])
a[2] = a[0] + a[1]
x = linalg.solve(a, b)
print(x)
```

```
[[-0.33333333]
 [ 0.66666667]
 [ 0.        ]]
<ipython-input-185-c373415ae86f>:4: LinAlgWarning: Ill-conditioned matrix (rcond=1.23358e-17): result may not be accurate.
  x = linalg.solve(a, b)
```

13.

- **Sparse Matrix:** A sparse matrix is one in which most of the elements are zero. Storing only the non-zero elements saves significant amounts of memory and computational resources, especially useful in large matrices. This is typically achieved using formats like Compressed Sparse Row (CSR) or Compressed Sparse Column (CSC).
- **Dense Matrix:** A dense matrix is one where most of the elements are non-zero. These matrices require storage for every element in the matrix, regardless of the element's value, typically leading to higher memory usage compared to sparse matrices. Dense matrices are usually stored in traditional 2D arrays.

```
row = np.array([0,1,2])
col = np.array([0,1,2])

data = np.array([1,1,1])

sparseMatrix = csr_matrix((data,(row,col)),shape=(3,3))

print(sparseMatrix)
```

```
  (0, 0)	1
  (1, 1)	1
  (2, 2)	1
```

14.

Code optimization is the process of modifying a software system to make some aspect of it work more efficiently or use fewer resources. The goal is to improve the performance of the code, which can include faster execution times, reduced memory usage, or other performance metrics. Optimization can occur at different stages of the software development process, such as during coding (by writing more efficient algorithms), at compile time (through compiler optimizations), or during runtime (via just-in-time compilation).

```
size = 4
data = [(0,Vectors.dense(np.random.rand(size)),),(1,Vectors.dense(np.random.rand(size)),),(1,Vectors.dense(np.random.rand(size)),)]
data_spark = spark.createDataFrame(data,["label","features"])
data_spark.show()
```

```
+-----+--------------------+
|label|            features|
+-----+--------------------+
|    0|[0.10367931412651...|
|    1|[0.22244202085247...|
|    1|[0.52231757297834...|
+-----+--------------------+
```

15.

Cluster configuration refers to the setup and management of a cluster of computers that work together to perform computing tasks. This can involve both hardware (setting up the physical machines and their connections) and software (configuring the network settings, operating systems, and applications that allow the machines to communicate and distribute tasks among them). Cluster configuration is crucial for achieving high availability, scalability, and efficient processing of large volumes of data or high-performance computing tasks. It determines how resources are allocated, managed, and maintained across the cluster.

```
spark_config = SparkSession.builder.appName('Cluster Configuration').master("spark://master-url:7077").config("spark.executor.memory","2
spark_config.stop()
```

Module 3

1. Data Modeling is the process of creating a visual representation (a model) of a system or process in a database to support business processes, enable data-driven decisions, and guide the development and design of database systems. It involves defining, structuring, and documenting the data resources and relationships between them.

Types of Data Modeling:

- **Conceptual Data Model:** Provides a high-level view of the system, focusing on the major entities and the relationships between them without going into details.
- **Logical Data Model:** More detailed than the conceptual model, it includes all entities, relationships, key attributes, and shows how data elements interconnect.
- **Physical Data Model:** Specifies the complete details of how data is stored in a database including tables, columns, indexes, primary and foreign keys, and the physical storage structure.

Four Steps to be Followed by a Data Analyst:

1. Requirement Gathering: Understand business requirements through interviews, questionnaires, or sessions with stakeholders.
2. Data Collection and Preparation: Gather the required data and clean it to ensure quality and relevance.
3. Model Construction: Build the model using appropriate modeling techniques based on the collected data and prepared schema.
4. Iteration and Validation: Continuously refine the model based on feedback and new data, and validate the model to ensure it meets business needs.

2. **Monte Carlo Markov Chain (MCMC)** is a class of algorithms used to sample from probability distributions based on constructing a Markov chain that has the desired distribution as its equilibrium distribution. The MCMC method is particularly useful for calculating numerical approximations in complex probabilistic models where direct sampling is difficult.

Types of MCMC:

- **Metropolis-Hastings Algorithm:** Generates a sequence of sample values in such a way that, as more sample values are produced, the distribution of values more closely approximates the desired distribution. Adjustments are made at each step based on a proposal distribution and acceptance probability.
- **Gibbs Sampling:** A special case of the Metropolis-Hastings algorithm that is used when the joint distribution is known but the conditional distributions are easier to sample from. It involves sequentially sampling from conditional distributions of each variable while holding all others fixed.
- **Hamiltonian Monte Carlo (HMC):** A variant that uses concepts from classical mechanics to inform the proposal distribution—specifically, it uses ideas about potential and kinetic energy to create efficient proposals that are likely to be accepted.

3. Optimization is the process of making a system or design as effective or functional as possible. Various types of optimization techniques are used in different fields, primarily categorized based on the nature of the functions involved and the methods used:

- **Linear Optimization:** Deals with problems in which the objective function and the constraints are linear. A common method used here is linear programming.
- **Nonlinear Optimization:** Involves objective functions or constraints that are nonlinear. It is typically more complex and uses methods like gradient descent or quadratic programming.
- **Convex Optimization:** A subset of optimization where the objective function is convex (the line segment between any two points on the graph of the function does not lie below the graph at any point). Solutions to convex optimization problems are global optima.
- **Discrete Optimization:** Involves optimization problems with discrete variables (e.g., integer values). Techniques used include integer programming and dynamic programming.

A.

```
data = [(1.0,2.0),(2.0,4.0),(3.0,6.0),(4.0,8.0)]
df = spark.createDataFrame(data,['x','y'])
df.show()
```

```
+---+---+
|  x|  y|
+---+---+
|1.0|2.0|
|2.0|4.0|
|3.0|6.0|
|4.0|8.0|
+---+---+
```

```python
def compute_gradient(df,w,b):
  N = df.count()
  dw = 1/N*df.rdd.map(lambda row:row.x*(row.y-(w*row.x+b))).sum()
  db = 1/N*df.rdd.map(lambda row:row.y-(w*row.x+b)).sum()
  return dw,db

def mse(df,w,b):
  N = df.count()
  mse_err = 1/(2*N)*df.rdd.map(lambda row:(row.y-(w*row.x+b))**2).sum()
  return mse_err


w = 0.5
b = 0
iter = 100
learning_rate = 0.01
for i in range(iter):
  dw,db = compute_gradient(df,w,b)
  w = w - learning_rate * dw
  b = b - learning_rate * db
  mse_err = mse(df,w,b)
  print(f"Iteration is {i+1}: w = {w} and b = {b} and mse = {mse_err}")
```

```
Iteration is 1: w = 0.3875 and b = -0.0375 and mse = 9.902460937499999
Iteration is 2: w = 0.265625 and b = -0.0781875 and mse = 11.622285158203127
Iteration is 3: w = 0.1335921875 and b = -0.12232875 and mse = 13.641313457875562
Iteration is 4: w = -0.009446617187500006 and b = -0.1702122328125 and mse = 16.01160099314561
Iteration is 5: w = -0.16441041929687503 and b = -0.2221505205703125 and mse = 18.79425941718463
Iteration is 6: w = -0.33229496375839845 and b = -0.2784822862584375 and mse = 22.061032518619022
Iteration is 7: w = -0.5141791431967393 and b = -0.33957448321498185 and mse = 25.896145988298244
Iteration is 8: w = -0.7112319410168693 and b = -0.40582470662705017 and mse = 30.39847900636999
Iteration is 9: w = -0.9247199542588107 and b = -0.4776637522187424 and mse = 35.684113639591295
Iteration is 10: w = -1.1560155446336902 and b = -0.5555583885974001 and mse = 41.88932777986865
Iteration is 11: w = -1.4066056701961518 and b = -0.6400143610992164 and mse = 49.174108790822665
Iteration is 12: w = -1.6781014544883437 and b = -0.7315796464651123 and mse = 57.726278454564266
Iteration is 13: w = -1.9722485547365973 and b = -0.8308479792919721 and mse = 67.7663355719386
Iteration is 14: w = -2.2909383958241416 and b = -0.9384626729533068 and mse = 79.5531410731630 5
Iteration is 15: w = -2.6362203423347847 and b = -1.0551207595784433 and mse = 93.39059221663571
Iteration is 16: w = -3.0103148869993546 and b = -1.1815774757325974 and mse = 109.63545795842226
Iteration is 17: w = -3.415627940417621 and b = -1.3186511226649071 and mse = 128.70657750944054
Iteration is 18: w = -3.8547663140155652 and b = -1.4672283324019968 and mse = 151.09565924748546
Iteration is 19: w = -4.3305544958767825 and b = -1.628269773576406 and mse = 177.3799584115538
Iteration is 20: w = -4.846052827406951 and b = -1.8028163337090897 and mse = 208.23716044693066
Iteration is 21: w = -5.4045771978051995 and b = -1.9919958177313544 and mse = 244.4628537376345
Iteration is 22: w = -6.009720383083874 and b = -2.197030205853798 and mse = 286.991042224872 1
Iteration is 23: w = -6.66537516696151 and b = -2.4192435174894324 and mse = 336.91822678739567
Iteration is 24: w = -7.375759392420859 and b = -2.6600703318383645 and mse = 395.5316762728772
Iteration is 25: w = -8.145443105148383 and b = -2.9210650199672696 and mse = 464.3426170909916
Iteration is 26: w = -8.979377963533693 and b = -3.203911747795652 and mse = 545.125197093952
Iteration is 27: w = -9.882929104493611 and b = -3.510435314361951 and mse = 639.9622283484242
Iteration is 28: w = -10.86190967018968 and b = -3.842612895117911 and mse = 751.2988881825902
Iteration is 29: w = -11.922618217831854 and b = -4.202586765823832 and mse = 882.0057630799715
Iteration is 30: w = -13.071879253314838 and b = -4.592678088927867 and mse = 1035.4528608778064
Iteration is 31: w = -14.317087149536649 and b = -5.015401851150017 and mse = 1215.5964995230677
Iteration is 32: w = -15.666253732030647 and b = -5.473483048399934 and mse = 1427.0813126348905
Iteration is 33: w = -17.128059838142942 and b = -5.969874222184699 and mse = 1675.3600018781065
Iteration is 34: w = -18.71191118155828 and b = -6.507774460360119 and mse = 1966.8339237177838
Iteration is 35: w = -20.427998881684154 and b = -7.090649984502678 and mse = 2309.0181352965596
Iteration is 36: w = -22.287365047423034 and b = -7.722256456389808 and mse = 2710.735154804972
Iteration is 37: w = -24.301973837389507 and b = -8.406663147139282 and mse = 3182.3424320594527
Iteration is 38: w = -26.484788453872202 and b = -9.148279124545413 and mse = 3735.9993941513662
Iteration is 39: w = -28.849854566026252 and b = -9.951881627137672 and mse = 4385.980951392685
Iteration is 40: w = -31.412390699156663 and b = -10.822646807559705 and mse = 5149.045546667595
Iteration is 41: w = -34.18888617178241 and b = -11.766183043114218 and mse = 6044.867237589416
Iteration is 42: w = -37.19720721074394 and b = -12.788567027839921 and mse = 7096.542951817088
Iteration is 43: w = -40.456711927245735 and b = -13.89638287838692 and mse = 8331.187994071559
Iteration is 44: w = -43.98837489374884 and b = -15.096764505351933 and mse = 9780.635158781897
Iteration is 45: w = -47.8149221234138 and b = -16.397441522749173 and mse = 11482.255473550125
Iteration is 46: w = -51.960977320738564 and b = -17.80678899106201 and mse = 13479.921734625521
Iteration is 47: w = -56.45322034457051 and b = -19.333881313991096 and mse = 15825.139677178906
Iteration is 48: w = -61.32055890326308 and b = -20.98855063574527 and mse = 18578.375945287724
Iteration is 49: w = -66.59431458690143 and b = -22.7814501146843 and mse = 21810.617100620562
Iteration is 50: w = -72.30842443378614 and b = -24.72412248050368 and mse = 25605.19986566982
Iteration is 51: w = -78.4996593283327 and b = -26.82907431615337 and mse = 30059.959790609086
Iteration is 52: w = -85.2078606358614 8 and b = -29.10985654252322 and mse = 35289.75374275325
Iteration is 53: w = -92.47619659711417 and b = -31.581151623844992 and mse = 41429.4212558504
Iteration is 54: w = -100.35144013249385 and b = -34.2588680550113 and mse = 48637.26109153709
Iteration is 55: w = -108.88426984380617 and b = -37.160242738873755 and mse = 57099.112648980125
Iteration is 56: w = -118.12959615056347 and b = -40.30395191235765 and mse = 67033.14745350048
Iteration is 57: w = -128.14691465966467 and b = -43.71023133524531 and mse = 78695.4942629089
Iteration is 58: w = -139.00068904252066 and b = -47.40100651508938 and mse = 92386.84282340933
```

4.

```
data = spark.read.options(header=True,inferSchema=True).csv('/content/drive/MyDrive/Big Data/Datasets/advertising (2).csv')
```

```
data.printSchema()
```

```
root
 |-- TV: double (nullable = true)
 |-- Radio: double (nullable = true)
 |-- Newspaper: double (nullable = true)
 |-- Sales: double (nullable = true)
```

```
data.show()
```

```
+-----+-----+---------+-----+
|   TV|Radio|Newspaper|Sales|
+-----+-----+---------+-----+
|230.1| 37.8|     69.2| 22.1|
| 44.5| 39.3|     45.1| 10.4|
| 17.2| 45.9|     69.3| 12.0|
|151.5| 41.3|     58.5| 16.5|
|180.8| 10.8|     58.4| 17.9|
|  8.7| 48.9|     75.0|  7.2|
| 57.5| 32.8|     23.5| 11.8|
|120.2| 19.6|     11.6| 13.2|
|  8.6|  2.1|      1.0|  4.8|
|199.8|  2.6|     21.2| 15.6|
| 66.1|  5.8|     24.2| 12.6|
|214.7| 24.0|      4.0| 17.4|
| 23.8| 35.1|     65.9|  9.2|
| 97.5|  7.6|      7.2| 13.7|
|204.1| 32.9|     46.0| 19.0|
|195.4| 47.7|     52.9| 22.4|
| 67.8| 36.6|    114.0| 12.5|
|281.4| 39.6|     55.8| 24.4|
| 69.2| 20.5|     18.3| 11.3|
|147.3| 23.9|     19.1| 14.6|
+-----+-----+---------+-----+
only showing top 20 rows
```

```
data = data.select('Radio').rdd.flatMap(lambda x: x).collect()
```

```
sc = SparkContext.getOrCreate()
```

```
data = sc.parallelize(data)
```

```
pdf_data = data.map(lambda x:norm.pdf(x))
print(pdf_data.collect())
```

```
[2.149048933896e-311, 0.0, 0.0, 0.0, 1.8743724023417964e-26, 0.0, 9.665456273536337e-235, 1.5192385847961196e-84, 0.04398359598042
```

```
ppf_data = data.map(lambda x:norm.ppf(x))
print(ppf_data.collect())
```

```
[nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan,
```

5. Newton's method, also known as the Newton-Raphson method, is a powerful technique used in optimization to find the stationary points of a function (where the gradient is zero) by solving the equation of the derivatives equal to zero. It is particularly effective for finding local maxima or minima of real-valued functions.

The fundamental idea behind Newton's method is to use an iterative approach to find successively better approximations to the roots (or zeroes) of a real-valued function. The method uses the first and second derivatives of the function to understand its curvature and to predict where the function's gradient will be zero.

Newton's method is widely used in numerical optimization, especially in scenarios where high precision is required and derivatives can be conveniently calculated. Its effectiveness and efficiency make it a staple method in fields such as economics, engineering, and machine learning model optimization.

Variations and Extensions:

- Modified Newton's Method: This variation involves modifications to handle cases where the Hessian is zero or nearly zero, such as by adding a small number to the denominator.
- Quasi-Newton Methods: These are popular alternatives that approximate the Hessian matrix instead of calculating it directly, significantly reducing computational overhead. Examples include the BFGS and L-BFGS algorithms.

Module 4

1.

```
data = spark.read.options(header='true',inferSchema='true').csv('/content/drive/MyDrive/Big Data/Datasets/Salary_Data (1).csv')
```

```
data.printSchema()
```

```
root
 |-- YearsExperience: double (nullable = true)
 |-- Salary: double (nullable = true)
```

```
data.show()
```

```
+---------------+-------+
|YearsExperience| Salary|
+---------------+-------+
|            1.1|39343.0|
|            1.3|46205.0|
|            1.5|37731.0|
|            2.0|43525.0|
|            2.2|39891.0|
|            2.9|56642.0|
|            3.0|60150.0|
|            3.2|54445.0|
|            3.2|64445.0|
|            3.7|57189.0|
|            3.9|63218.0|
|            4.0|55794.0|
|            4.0|56957.0|
|            4.1|57081.0|
|            4.5|61111.0|
|            4.9|67938.0|
|            5.1|66029.0|
|            5.3|83088.0|
|            5.9|81363.0|
|            6.0|93940.0|
+---------------+-------+
only showing top 20 rows
```

```
data.describe().show()
```

```
+-------+------------------+------------------+
|summary|   YearsExperience|            Salary|
+-------+------------------+------------------+
|  count|                30|                30|
|   mean|5.3133333333333335|           76003.0|
| stddev| 2.837888157662718|27414.429784582302|
|    min|               1.1|           37731.0|
|    max|              10.5|          122391.0|
+-------+------------------+------------------+
```

```
assembler = VectorAssembler(inputCols=['YearsExperience'],outputCol='Independent Feature')
data_final = assembler.transform(data)
```

```
scaler = StandardScaler(inputCol='Independent Feature',outputCol='Scaled Feature')
data_scaled = scaler.fit(data_final).transform(data_final)
```

```
train_data,test_data = data_scaled.randomSplit([0.70,0.30],seed=44)
```

```
lr_model = LinearRegression(featuresCol='Scaled Feature',labelCol='Salary',regParam=0.0,elasticNetParam=0.0,maxIter=100,solver='l-bfgs')
para_grid = ParamGridBuilder().addGrid(lr_model.regParam,[0.1,0.01,1.0]).build()
```

```
eval = RegressionEvaluator(labelCol='Salary',predictionCol='prediction',metricName='r2')
cross_eval = CrossValidator(estimator=lr_model,estimatorParamMaps=para_grid,evaluator=eval,numFolds=5,seed=44)
```

```
cv_model = cross_eval.fit(train_data)
```

```
best_model = cv_model.bestModel
predictions = best_model.transform(test_data)
predictions.show()
```

```
+---------------+--------+-------------------+--------------------+------------------+
|YearsExperience|  Salary|Independent Feature|      Scaled Feature|        prediction|
+---------------+--------+-------------------+--------------------+------------------+
|            1.1| 39343.0|              [1.1]|[0.38761217457771...| 35736.60704161503|
|            1.3| 46205.0|              [1.3]|[0.45808711541003...|  37659.0672509933|
|            2.0| 43525.0|              [2.0]|[0.7047494083231237]| 44387.67798381724|
|            2.2| 39891.0|              [2.2]|[0.7752243491554361]| 46310.13819319551|
|            3.2| 54445.0|              [3.2]| [1.127599053316998]| 55922.43924008685|
|            3.7| 57189.0|              [3.7]| [1.303786405397779]| 60728.58976353253|
|            3.9| 63218.0|              [3.9]|[1.3742613462300912]|  62651.0499729108|
|            4.0| 55794.0|              [4.0]|[1.4094988166462474]|63612.280077599935|
|            5.3| 83088.0|              [5.3]|[1.8675859320562778]| 76108.27143855867|
|            9.6|112635.0|              [9.6]| [3.382797159950994]|117441.16594019148|
|           10.3|122391.0|             [10.3]|[3.6294594528640873]|124169.77667301541|
+---------------+--------+-------------------+--------------------+------------------+
```

```
r2 = eval.evaluate(predictions)
print(f"MSE value : {r2}")
```

```
MSE value : 0.9661780679979106
```

2.

```
data = spark.read.options(header='true',inferSchema='true').csv('/content/drive/MyDrive/Big Data/Datasets/heart (1).csv')
```

```
data.printSchema()
```

```
root
 |-- age: integer (nullable = true)
 |-- sex: integer (nullable = true)
 |-- cp: integer (nullable = true)
 |-- trestbps: integer (nullable = true)
 |-- chol: integer (nullable = true)
 |-- fbs: integer (nullable = true)
 |-- restecg: integer (nullable = true)
 |-- thalach: integer (nullable = true)
 |-- exang: integer (nullable = true)
 |-- oldpeak: double (nullable = true)
 |-- slope: integer (nullable = true)
 |-- ca: integer (nullable = true)
 |-- thal: integer (nullable = true)
 |-- target: integer (nullable = true)
```

```
data.describe().show()
```

```
+-------+------------------+-------------------+------------------+------------------+------------------+-------------------+------
|summary|               age|                sex|                cp|          trestbps|              chol|                fbs|
+-------+------------------+-------------------+------------------+------------------+------------------+-------------------+------
|  count|               303|                303|               303|               303|               303|                303|
|   mean|54.366336633663366| 0.6831683168316832| 0.966996699669967|131.62376237623764|246.26402640264027| 0.1485148514851485|0.5280
| stddev|  9.08210098983786|0.46601082333962385|1.0320524894832983|  17.5381428135171| 51.83075098793005|0.35619787492797644|0.5258
|    min|                29|                  0|                 0|                94|               126|                  0|
|    max|                77|                  1|                 3|               200|               564|                  1|
+-------+------------------+-------------------+------------------+------------------+------------------+-------------------+------
```

```
assembler = VectorAssembler(inputCols=['age','sex','cp','trestbps','chol','fbs','restecg','thalach','exang','oldpeak','slope','ca','tha
scaler = StandardScaler(inputCol='Independent Feature',outputCol='Scaled Feature')
svm_model = LinearSVC()
one_model = OneVsRest(featuresCol='Scaled Feature',labelCol='target',classifier=svm_model)
lr_model = LogisticRegression(featuresCol='Scaled Feature',labelCol='target')
pipeline_lr = Pipeline(stages=[assembler,scaler,lr_model])
pipeline_svm = Pipeline(stages=[assembler,scaler,one_model])
```

```
train_data,test_data = data.randomSplit([0.7,0.3],seed=45)
```

```
lr_model = pipeline_lr.fit(train_data)
svm_model = pipeline_svm.fit(train_data)
```

```
prediction = lr_model.transform(test_data)
prediction.show()
```

```
+---+---+---+--------+----+---+-------+-------+-----+-------+-----+---+----+------+--------------------+--------------------+------
|age|sex| cp|trestbps|chol|fbs|restecg|thalach|exang|oldpeak|slope| ca|thal|target| Independent Feature|      Scaled Feature|
+---+---+---+--------+----+---+-------+-------+-----+-------+-----+---+----+------+--------------------+--------------------+------
| 29|  1|  1|     130| 204|  0|      0|    202|    0|    0.0|    2|  0|   2|     1|[29.0,1.0,1.0,130...|[3.30284055574669...|[-3.128
| 34|  1|  3|     118| 182|  0|      0|    174|    0|    0.0|    2|  0|   2|     1|[34.0,1.0,3.0,118...|[3.87229582397888...|[-4.533
| 35|  1|  1|     122| 192|  0|      1|    174|    0|    0.0|    2|  0|   2|     1|[35.0,1.0,1.0,122...|[3.98618687762532...|[-2.877
| 37|  1|  2|     130| 250|  0|      1|    187|    0|    3.5|    0|  0|   2|     1|[37.0,1.0,2.0,130...|[4.21396898491820...|[-0.129
| 41|  0|  2|     112| 268|  0|      0|    172|    1|    0.0|    2|  0|   2|     1|[41.0,0.0,2.0,112...|[4.66953319950395...|[-4.249
| 41|  1|  0|     110| 172|  0|      0|    158|    0|    0.0|    2|  0|   3|     0|(13,[0,1,3,4,7,10...|(13,[0,1,3,4,7,10...|[-0.696
| 42|  0|  2|     120| 209|  0|      1|    173|    0|    0.0|    1|  0|   2|     1|[42.0,0.0,2.0,120...|[4.78342425315039...|[-4.987
| 42|  1|  0|     136| 315|  0|      1|    125|    1|    1.8|    1|  0|   1|     0|[42.0,1.0,0.0,136...|[4.78342425315039...|[1.9002
| 42|  1|  0|     140| 226|  0|      1|    178|    0|    0.0|    2|  0|   2|     1|[42.0,1.0,0.0,140...|[4.78342425315039...|[-1.587
| 42|  1|  3|     148| 244|  0|      0|    178|    0|    0.8|    2|  2|   2|     1|[42.0,1.0,3.0,148...|[4.78342425315039...|[-1.837
| 43|  0|  0|     132| 341|  1|      0|    136|    1|    3.0|    1|  0|   3|     0|[43.0,0.0,0.0,132...|[4.89731530679682...|[2.9212
| 43|  1|  0|     110| 211|  0|      1|    161|    0|    0.0|    2|  0|   3|     1|[43.0,1.0,0.0,110...|[4.89731530679682...|[-0.751
| 43|  1|  0|     150| 247|  0|      1|    171|    0|    1.5|    2|  0|   2|     1|[43.0,1.0,0.0,150...|[4.89731530679682...|[-0.029
| 43|  1|  2|     130| 315|  0|      1|    162|    0|    1.9|    2|  1|   2|     1|[43.0,1.0,2.0,130...|[4.89731530679682...|[-0.510
| 44|  0|  2|     108| 141|  0|      1|    175|    0|    0.6|    1|  0|   2|     1|[44.0,0.0,2.0,108...|[5.01120636044326...|[-5.408
| 44|  1|  0|     110| 197|  0|      0|    177|    0|    0.0|    2|  1|   2|     0|[44.0,1.0,0.0,110...|[5.01120636044326...|[-1.538
| 44|  1|  0|     112| 290|  0|      0|    153|    0|    0.0|    2|  1|   2|     0|[44.0,1.0,0.0,112...|[5.01120636044326...|[-0.128
| 44|  1|  1|     120| 220|  0|      1|    170|    0|    0.0|    2|  0|   2|     1|[44.0,1.0,1.0,120...|[5.01120636044326...|[-2.709
| 44|  1|  2|     130| 233|  0|      1|    179|    1|    0.4|    2|  0|   2|     1|[44.0,1.0,2.0,130...|[5.01120636044326...|[-2.527
| 44|  1|  2|     140| 235|  0|      0|    180|    0|    0.0|    2|  0|   2|     1|[44.0,1.0,2.0,140...|[5.01120636044326...|[-3.115
+---+---+---+--------+----+---+-------+-------+-----+-------+-----+---+----+------+--------------------+--------------------+------
only showing top 20 rows
```

```
eval = BinaryClassificationEvaluator(labelCol='target')
a_ROC = eval.evaluate(prediction)
print(f'area under ROC is {a_ROC}')
```

```
area under ROC is 0.9033119658119658
```

```
pred = svm_model.transform(test_data)
pred.show()
```

```
+---+---+---+--------+----+---+-------+-------+-----+-------+-----+---+----+------+--------------------+--------------------+------
|age|sex| cp|trestbps|chol|fbs|restecg|thalach|exang|oldpeak|slope| ca|thal|target| Independent Feature|      Scaled Feature|
+---+---+---+--------+----+---+-------+-------+-----+-------+-----+---+----+------+--------------------+--------------------+------
| 29|  1|  1|     130| 204|  0|      0|    202|    0|    0.0|    2|  0|   2|     1|[29.0,1.0,1.0,130...|[3.30284055574669...|[-2.186
| 34|  1|  3|     118| 182|  0|      0|    174|    0|    0.0|    2|  0|   2|     1|[34.0,1.0,3.0,118...|[3.87229582397888...|[-3.519
| 35|  1|  1|     122| 192|  0|      1|    174|    0|    0.0|    2|  0|   2|     1|[35.0,1.0,1.0,122...|[3.98618687762532...|[-2.030
| 37|  1|  2|     130| 250|  0|      1|    187|    0|    3.5|    0|  0|   2|     1|[37.0,1.0,2.0,130...|[4.21396898491820...|[0.2105
| 41|  0|  2|     112| 268|  0|      0|    172|    1|    0.0|    2|  0|   2|     1|[41.0,0.0,2.0,112...|[4.66953319950395...|[-3.193
| 41|  1|  0|     110| 172|  0|      0|    158|    0|    0.0|    2|  0|   3|     0|(13,[0,1,3,4,7,10...|(13,[0,1,3,4,7,10...|[-0.462
| 42|  0|  2|     120| 209|  0|      1|    173|    0|    0.0|    1|  0|   2|     1|[42.0,0.0,2.0,120...|[4.78342425315039...|[-3.389
| 42|  1|  0|     136| 315|  0|      1|    125|    1|    1.8|    1|  0|   1|     0|[42.0,1.0,0.0,136...|[4.78342425315039...|[1.0294
| 42|  1|  0|     140| 226|  0|      1|    178|    0|    0.0|    2|  0|   2|     1|[42.0,1.0,0.0,140...|[4.78342425315039...|[-1.004
| 42|  1|  3|     148| 244|  0|      0|    178|    0|    0.8|    2|  2|   2|     1|[42.0,1.0,3.0,148...|[4.78342425315039...|[-1.557
| 43|  0|  0|     132| 341|  1|      0|    136|    1|    3.0|    1|  0|   3|     0|[43.0,0.0,0.0,132...|[4.89731530679682...|[1.9734
| 43|  1|  0|     110| 211|  0|      1|    161|    0|    0.0|    2|  0|   3|     1|[43.0,1.0,0.0,110...|[4.89731530679682...|[-0.411
| 43|  1|  0|     150| 247|  0|      1|    171|    0|    1.5|    2|  0|   2|     1|[43.0,1.0,0.0,150...|[4.89731530679682...|[0.1428
| 43|  1|  2|     130| 315|  0|      1|    162|    0|    1.9|    2|  1|   2|     1|[43.0,1.0,2.0,130...|[4.89731530679682...|[-0.567
| 44|  0|  2|     108| 141|  0|      1|    175|    0|    0.6|    1|  0|   2|     1|[44.0,0.0,2.0,108...|[5.01120636044326...|[-3.521
| 44|  1|  0|     110| 197|  0|      0|    177|    0|    0.0|    2|  1|   2|     0|[44.0,1.0,0.0,110...|[5.01120636044326...|[-1.088
| 44|  1|  0|     112| 290|  0|      0|    153|    0|    0.0|    2|  1|   2|     0|[44.0,1.0,0.0,112...|[5.01120636044326...|[-0.410
| 44|  1|  1|     120| 220|  0|      1|    170|    0|    0.0|    2|  0|   2|     1|[44.0,1.0,1.0,120...|[5.01120636044326...|[-1.959
| 44|  1|  2|     130| 233|  0|      1|    179|    1|    0.4|    2|  0|   2|     1|[44.0,1.0,2.0,130...|[5.01120636044326...|[-1.847
| 44|  1|  2|     140| 235|  0|      0|    180|    0|    0.0|    2|  0|   2|     1|[44.0,1.0,2.0,140...|[5.01120636044326...|[-2.408
+---+---+---+--------+----+---+-------+-------+-----+-------+-----+---+----+------+--------------------+--------------------+------
only showing top 20 rows
```

```
a_ROC = eval.evaluate(pred)
print(f'area under ROC is {a_ROC}')
```

```
area under ROC is 0.8974358974358975
```

```
data = spark.read.options(header='true',inferSchema='true').csv('/content/drive/MyDrive/Big Data/Datasets/heart_disease_health_indicato
```

```
data.printSchema()
```

```
root
 |-- HeartDiseaseorAttack: double (nullable = true)
 |-- HighBP: double (nullable = true)
 |-- HighChol: double (nullable = true)
 |-- CholCheck: double (nullable = true)
 |-- BMI: double (nullable = true)
 |-- Smoker: double (nullable = true)
 |-- Stroke: double (nullable = true)
 |-- Diabetes: double (nullable = true)
 |-- PhysActivity: double (nullable = true)
 |-- Fruits: double (nullable = true)
 |-- Veggies: double (nullable = true)
 |-- HvyAlcoholConsump: double (nullable = true)
 |-- AnyHealthcare: double (nullable = true)
 |-- NoDocbcCost: double (nullable = true)
 |-- GenHlth: double (nullable = true)
 |-- MentHlth: double (nullable = true)
 |-- PhysHlth: double (nullable = true)
 |-- DiffWalk: double (nullable = true)
 |-- Sex: double (nullable = true)
 |-- Age: double (nullable = true)
 |-- Education: double (nullable = true)
 |-- Income: double (nullable = true)
```

```
data.describe().show()
```

```
+-------+--------------------+-------------------+-------------------+-------------------+------------------+-------------------+--
|summary|HeartDiseaseorAttack|             HighBP|           HighChol|          CholCheck|               BMI|             Smoker|
+-------+--------------------+-------------------+-------------------+-------------------+------------------+-------------------+--
|  count|              253680|             253680|             253680|             253680|            253680|             253680|
|   mean| 0.09418558814254178| 0.4290011037527594| 0.4241209397666351| 0.9626695048880479|28.382363607694735|0.44316855881425415|0.
| stddev|  0.2920873147507536|0.49493446268990043|0.49420980465688596|0.18957075436272514| 6.608694201406001| 0.4967606667785607| 0
|    min|                 0.0|                0.0|                0.0|                0.0|              12.0|                0.0|
|    max|                 1.0|                1.0|                1.0|                1.0|              98.0|                1.0|
+-------+--------------------+-------------------+-------------------+-------------------+------------------+-------------------+--
```

```
data.show()
```

```
+--------------------+------+--------+---------+----+------+------+--------+------------+------+-------+-----------------+---------
|HeartDiseaseorAttack|HighBP|HighChol|CholCheck| BMI|Smoker|Stroke|Diabetes|PhysActivity|Fruits|Veggies|HvyAlcoholConsump|AnyHealth
+--------------------+------+--------+---------+----+------+------+--------+------------+------+-------+-----------------+---------
|                 0.0|   1.0|     1.0|      1.0|40.0|   1.0|   0.0|     0.0|         0.0|   0.0|    1.0|              0.0|
|                 0.0|   0.0|     0.0|      0.0|25.0|   1.0|   0.0|     0.0|         1.0|   0.0|    0.0|              0.0|
|                 0.0|   1.0|     1.0|      1.0|28.0|   0.0|   0.0|     0.0|         0.0|   1.0|    0.0|              0.0|
|                 0.0|   1.0|     0.0|      1.0|27.0|   0.0|   0.0|     0.0|         1.0|   1.0|    1.0|              0.0|
|                 0.0|   1.0|     1.0|      1.0|24.0|   0.0|   0.0|     0.0|         1.0|   1.0|    1.0|              0.0|
|                 0.0|   1.0|     1.0|      1.0|25.0|   1.0|   0.0|     0.0|         1.0|   1.0|    1.0|              0.0|
|                 0.0|   1.0|     0.0|      1.0|30.0|   1.0|   0.0|     0.0|         0.0|   0.0|    0.0|              0.0|
|                 0.0|   1.0|     1.0|      1.0|25.0|   1.0|   0.0|     0.0|         1.0|   0.0|    1.0|              0.0|
|                 1.0|   1.0|     1.0|      1.0|30.0|   1.0|   0.0|     2.0|         0.0|   1.0|    1.0|              0.0|
|                 0.0|   0.0|     0.0|      1.0|24.0|   0.0|   0.0|     0.0|         0.0|   0.0|    1.0|              0.0|
|                 0.0|   0.0|     0.0|      1.0|25.0|   1.0|   0.0|     2.0|         1.0|   1.0|    1.0|              0.0|
|                 0.0|   1.0|     1.0|      1.0|34.0|   1.0|   0.0|     0.0|         0.0|   1.0|    1.0|              0.0|
|                 0.0|   0.0|     0.0|      1.0|26.0|   1.0|   0.0|     0.0|         0.0|   0.0|    1.0|              0.0|
|                 0.0|   1.0|     1.0|      1.0|28.0|   0.0|   0.0|     2.0|         0.0|   0.0|    1.0|              0.0|
|                 0.0|   0.0|     1.0|      1.0|33.0|   1.0|   1.0|     0.0|         1.0|   0.0|    1.0|              0.0|
|                 0.0|   1.0|     0.0|      1.0|33.0|   0.0|   0.0|     0.0|         1.0|   0.0|    0.0|              0.0|
|                 0.0|   1.0|     1.0|      1.0|21.0|   0.0|   0.0|     0.0|         1.0|   1.0|    1.0|              0.0|
|                 0.0|   0.0|     0.0|      1.0|23.0|   1.0|   0.0|     2.0|         1.0|   0.0|    0.0|              0.0|
|                 0.0|   0.0|     0.0|      0.0|23.0|   0.0|   0.0|     0.0|         0.0|   0.0|    1.0|              0.0|
|                 0.0|   0.0|     1.0|      1.0|28.0|   0.0|   0.0|     0.0|         0.0|   0.0|    0.0|              1.0|
+--------------------+------+--------+---------+----+------+------+--------+------------+------+-------+-----------------+---------
only showing top 20 rows
```

```
assembler = VectorAssembler(inputCols=['HighBP','HighChol','CholCheck','BMI','Smoker','Stroke','Diabetes','PhysActivity','Fruits','Vegg
data_final = assembler.transform(data)
```

```
scaler = StandardScaler(inputCol='Independent Feature',outputCol='Scaled Feature')
data_scaled = scaler.fit(data_final).transform(data_final)
```

```
train_data,test_data = data_scaled.randomSplit([0.70,0.30],seed=44)
```

```
glm = GeneralizedLinearRegression(labelCol='HeartDiseaseorAttack',featuresCol='Scaled Feature',maxIter=100,family='gaussian')
```

```
model = glm.fit(train_data)
model.summary
```

```
Coefficients:
          Feature Estimate Std Error  T Value P Value
      (Intercept)  -0.1517    0.0069 -21.9848  0.0000
 Scaled Feature_0   0.0162    0.0007  21.8439  0.0000
 Scaled Feature_1   0.0194    0.0007  27.9522  0.0000
 Scaled Feature_2   0.0030    0.0006   4.6224  0.0000
 Scaled Feature_3  -0.0081    0.0007 -11.6771  0.0000
 Scaled Feature_4   0.0114    0.0007  17.0254  0.0000
 Scaled Feature_5   0.0371    0.0007  56.4336  0.0000
 Scaled Feature_6   0.0163    0.0007  23.2572  0.0000
 Scaled Feature_7   0.0019    0.0007   2.7023  0.0069
 Scaled Feature_8   0.0018    0.0007   2.7057  0.0068
 Scaled Feature_9   0.0020    0.0007   2.9454  0.0032
Scaled Feature_10  -0.0040    0.0006  -6.0839  0.0000
Scaled Feature_11   0.0019    0.0007   2.8086  0.0050
Scaled Feature_12   0.0021    0.0007   3.0952  0.0020
Scaled Feature_13   0.0358    0.0009  41.5989  0.0000
Scaled Feature_14  -0.0012    0.0007  -1.6459  0.0998
Scaled Feature_15   0.0085    0.0008  10.4174  0.0000
Scaled Feature_16   0.0184    0.0008  23.2893  0.0000
Scaled Feature_17   0.0269    0.0007  40.6517  0.0000
Scaled Feature_18   0.0346    0.0007  46.6849  0.0000
Scaled Feature_19   0.0007    0.0007   0.9377  0.3484
Scaled Feature_20  -0.0075    0.0008  -9.5049  0.0000

(Dispersion parameter for gaussian family taken to be 0.0727)
    Null deviance: 15217.5656 on 177608 degrees of freedom
Residual deviance: 12914.8190 on 177608 degrees of freedom
AIC: 38511.7305
```

```
predictions = model.transform(test_data)
predictions.show()
```

```
+--------------------+------+--------+---------+----+------+------+--------+------------+------+-------+-----------------+---------
|HeartDiseaseorAttack|HighBP|HighChol|CholCheck| BMI|Smoker|Stroke|Diabetes|PhysActivity|Fruits|Veggies|HvyAlcoholConsump|AnyHealthc
+--------------------+------+--------+---------+----+------+------+--------+------------+------+-------+-----------------+---------
|                 0.0|   0.0|     0.0|      0.0|14.0|   1.0|   0.0|     0.0|         1.0|   1.0|    1.0|              0.0|
|                 0.0|   0.0|     0.0|      0.0|16.0|   0.0|   0.0|     0.0|         1.0|   0.0|    1.0|              0.0|
|                 0.0|   0.0|     0.0|      0.0|16.0|   1.0|   0.0|     0.0|         0.0|   0.0|    1.0|              0.0|
|                 0.0|   0.0|     0.0|      0.0|16.0|   1.0|   0.0|     0.0|         1.0|   0.0|    0.0|              0.0|
|                 0.0|   0.0|     0.0|      0.0|17.0|   0.0|   0.0|     0.0|         0.0|   0.0|    0.0|              0.0|
|                 0.0|   0.0|     0.0|      0.0|17.0|   0.0|   0.0|     0.0|         0.0|   1.0|    0.0|              0.0|
|                 0.0|   0.0|     0.0|      0.0|17.0|   0.0|   0.0|     0.0|         0.0|   1.0|    1.0|              0.0|
|                 0.0|   0.0|     0.0|      0.0|17.0|   0.0|   0.0|     0.0|         1.0|   0.0|    1.0|              0.0|
|                 0.0|   0.0|     0.0|      0.0|17.0|   0.0|   0.0|     0.0|         1.0|   1.0|    1.0|              0.0|
|                 0.0|   0.0|     0.0|      0.0|17.0|   1.0|   0.0|     0.0|         1.0|   0.0|    0.0|              0.0|
|                 0.0|   0.0|     0.0|      0.0|17.0|   1.0|   0.0|     0.0|         1.0|   0.0|    1.0|              1.0|
|                 0.0|   0.0|     0.0|      0.0|17.0|   1.0|   0.0|     0.0|         1.0|   1.0|    1.0|              1.0|
|                 0.0|   0.0|     0.0|      0.0|17.0|   1.0|   1.0|     0.0|         1.0|   1.0|    1.0|              1.0|
|                 0.0|   0.0|     0.0|      0.0|18.0|   0.0|   0.0|     0.0|         1.0|   1.0|    1.0|              0.0|
|                 0.0|   0.0|     0.0|      0.0|18.0|   0.0|   0.0|     0.0|         1.0|   1.0|    1.0|              0.0|
|                 0.0|   0.0|     0.0|      0.0|18.0|   0.0|   0.0|     0.0|         1.0|   1.0|    1.0|              0.0|
|                 0.0|   0.0|     0.0|      0.0|18.0|   0.0|   0.0|     0.0|         1.0|   1.0|    1.0|              0.0|
|                 0.0|   0.0|     0.0|      0.0|18.0|   0.0|   0.0|     0.0|         1.0|   1.0|    1.0|              0.0|
|                 0.0|   0.0|     0.0|      0.0|18.0|   1.0|   0.0|     0.0|         0.0|   0.0|    1.0|              1.0|
|                 0.0|   0.0|     0.0|      0.0|18.0|   1.0|   0.0|     0.0|         0.0|   1.0|    1.0|              0.0|
+--------------------+------+--------+---------+----+------+------+--------+------------+------+-------+-----------------+---------
only showing top 20 rows
```

```
evaluator = RegressionEvaluator(labelCol='HeartDiseaseorAttack',predictionCol='prediction',metricName='rmse')
rmse = evaluator.evaluate(predictions)
print(f"RMSE Value: {rmse}")
```

```
RMSE Value: 0.26879609774691215
```

```
evaluator = RegressionEvaluator(labelCol='HeartDiseaseorAttack',predictionCol='prediction',metricName='mse')
mse = evaluator.evaluate(predictions)
print(f"MSE Value: {mse}")
```

```
MSE Value: 0.07225134216396756
```

```
evaluator = RegressionEvaluator(labelCol='HeartDiseaseorAttack',predictionCol='prediction',metricName='mae')
mae = evaluator.evaluate(predictions)
print(f"MAE Value: {mae}")
```

```
MAE Value: 0.1618541017871356
```

```
evaluator = RegressionEvaluator(labelCol='HeartDiseaseorAttack',predictionCol='prediction',metricName='r2')
r2 = evaluator.evaluate(predictions)
print(f"R2 Value: {r2}")
```

```
R2 Value: 0.1447841115076205
```

3.

```
data = spark.read.options(header='true',inferSchema='true').csv('/content/drive/MyDrive/Big Data/Datasets/heart (1).csv')
data.printSchema()
```

```
root
 |-- age: integer (nullable = true)
 |-- sex: integer (nullable = true)
 |-- cp: integer (nullable = true)
 |-- trestbps: integer (nullable = true)
 |-- chol: integer (nullable = true)
 |-- fbs: integer (nullable = true)
 |-- restecg: integer (nullable = true)
 |-- thalach: integer (nullable = true)
 |-- exang: integer (nullable = true)
 |-- oldpeak: double (nullable = true)
 |-- slope: integer (nullable = true)
 |-- ca: integer (nullable = true)
 |-- thal: integer (nullable = true)
 |-- target: integer (nullable = true)
```

```
def detect_continuous_variables(df,threshold):
  continuous_columns = []
  for column in df.columns:
    dtype = df.schema[column].dataType
    if isinstance(dtype,(IntegerType,NumericType)):
      distinct_count = df.select(approxCountDistinct(column)).collect()[0][0]
      if distinct_count > threshold:
        continuous_columns.append(column)
  return continuous_columns
```

```
continous_variables = detect_continuous_variables(data,10)
continous_variables
```

```
/usr/local/lib/python3.10/dist-packages/pyspark/sql/functions.py:3796: FutureWarning: Deprecated in 2.1, use approx_count_distinct i
  warnings.warn("Deprecated in 2.1, use approx_count_distinct instead.", FutureWarning)
['age', 'trestbps', 'chol', 'thalach', 'oldpeak']
```

```
def iqr_treat_outlier(df,columns,factor=1.5):
  for column in columns:
    quantile = df.approxQuantile(column,[0.25,0.75],0.01)
    q1,q3 = quantile[0],quantile[1]
    iqr = q3-q1
    lower_bound = q1-factor*iqr
    upper_bound = q3+factor*iqr
    df = df.filter((col(column)>=lower_bound)&(col(column)<=upper_bound))
  return df
```

```
df_outlier_treated = iqr_treat_outlier(data,continous_variables,1.5)
df_oulier_pandas = df_outlier_treated.toPandas()
df_oulier_pandas.head()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	tha
0	63	1	3	145	233	1	0	150	0	2.3	0	0	
1	37	1	2	130	250	0	1	187	0	3.5	0	0	
2	41	0	1	130	204	0	0	172	0	1.4	2	0	
3	56	1	1	120	236	0	1	178	0	0.8	2	0	
4	57	0	0	120	354	0	1	163	1	0.6	2	0	

```
def z_score_outlier(df,columns,threshold=3.0):
  for column in columns:
    stats = df.select(mean(col(column)).alias('mean'),stddev(col(column)).alias('stddev')).collect()[0]
    df = df.withColumn(f'{column}_z_score',(col(column)-stats['mean'])/stats['stddev']).filter(f'abs({column}_z_score)<={threshold}').dı
  return df
```

```
df_z_score_output = z_score_outlier(data,continous_variables,3.0)
df_z_score_output.count()
df_z_score_pandas = df_z_score_output.toPandas()
df_z_score_pandas.head()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	tha
0	63	1	3	145	233	1	0	150	0	2.3	0	0	
1	37	1	2	130	250	0	1	187	0	3.5	0	0	
2	41	0	1	130	204	0	0	172	0	1.4	2	0	
3	56	1	1	120	236	0	1	178	0	0.8	2	0	
4	57	0	0	120	354	0	1	163	1	0.6	2	0	

Module 5

1.

```
data = spark.read.options(header='true',inferSchema='true').csv('/content/drive/MyDrive/Big Data/Datasets/Mall_Customers (2).csv')
data.printSchema()
```

```
root
 |-- CustomerID: integer (nullable = true)
 |-- Genre: string (nullable = true)
 |-- Age: integer (nullable = true)
 |-- Annual Income (k$): integer (nullable = true)
 |-- Spending Score (1-100): integer (nullable = true)
```

```
data.summary().show()
```

```
+-------+------------------+------+-----------------+------------------+----------------------+
|summary|        CustomerID| Genre|              Age|Annual Income (k$)|Spending Score (1-100)|
+-------+------------------+------+-----------------+------------------+----------------------+
|  count|               200|   200|              200|               200|                   200|
|   mean|             100.5|  NULL|            38.85|             60.56|                  50.2|
| stddev|57.879184513951124|  NULL|13.96900733155888| 26.26472116527124|    25.823521668370173|
|    min|                 1|Female|               18|                15|                     1|
|    25%|                50|  NULL|               28|                40|                    34|
|    50%|               100|  NULL|               36|                61|                    50|
|    75%|               150|  NULL|               49|                78|                    73|
|    max|               200|  Male|               70|               137|                    99|
+-------+------------------+------+-----------------+------------------+----------------------+
```

```
indexer = StringIndexer(inputCol="Genre",outputCol="Gender_num")
data = indexer.fit(data).transform(data)
data.show()
```

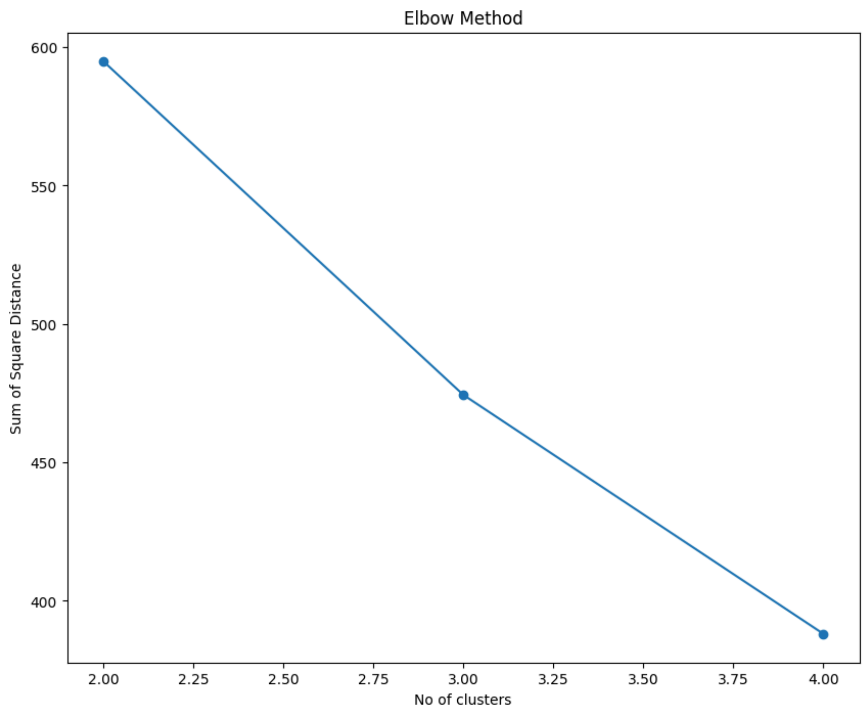
```
+----------+------+---+------------------+----------------------+----------+
|CustomerID| Genre|Age|Annual Income (k$)|Spending Score (1-100)|Gender_num|
+----------+------+---+------------------+----------------------+----------+
|         1|  Male| 19|                15|                    39|       1.0|
|         2|  Male| 21|                15|                    81|       1.0|
|         3|Female| 20|                16|                     6|       0.0|
|         4|Female| 23|                16|                    77|       0.0|
|         5|Female| 31|                17|                    40|       0.0|
|         6|Female| 22|                17|                    76|       0.0|
|         7|Female| 35|                18|                     6|       0.0|
|         8|Female| 23|                18|                    94|       0.0|
|         9|  Male| 64|                19|                     3|       1.0|
|        10|Female| 30|                19|                    72|       0.0|
|        11|  Male| 67|                19|                    14|       1.0|
|        12|Female| 35|                19|                    99|       0.0|
|        13|Female| 58|                20|                    15|       0.0|
|        14|Female| 24|                20|                    77|       0.0|
|        15|  Male| 37|                20|                    13|       1.0|
|        16|  Male| 22|                20|                    79|       1.0|
|        17|Female| 35|                21|                    35|       0.0|
|        18|  Male| 20|                21|                    66|       1.0|
|        19|  Male| 52|                23|                    29|       1.0|
|        20|Female| 35|                23|                    98|       0.0|
+----------+------+---+------------------+----------------------+----------+
only showing top 20 rows
```

```
assembler = VectorAssembler(inputCols=['Age','Gender_num','Annual Income (k$)','Spending Score (1-100)'],outputCol="Independent Feature'
data = assembler.transform(data)
```

```
sc = StandardScaler(inputCol="Independent Feature",outputCol="Scaled Feature")
data = sc.fit(data).transform(data)
```

```
k_range = range(2,5)
ssd = []
sil_score = []
for k in k_range:
  km = KMeans(featuresCol='Scaled Feature',k=k)
  model = km.fit(data)
  eval = ClusteringEvaluator(featuresCol='Scaled Feature')
  prediction = model.transform(data)
  s_score = eval.evaluate(prediction)
  sil_score.append(s_score)
  sum_squ = model.summary.trainingCost
  ssd.append(sum_squ)
```

```
plt.figure(figsize=(10,8))
plt.plot(k_range,ssd,marker='o')
plt.xlabel('No of clusters')
plt.ylabel('Sum of Square Distance')
plt.title('Elbow Method')
plt.show()
```



```
sil_score
```

```
    [0.41960650030709834, 0.39166384459523845, 0.441485161289118]
```

```
kmm = KMeans(k=4,featuresCol='Scaled Feature',maxIter=50,seed=44)
model = kmm.fit(data)
predict = model.transform(data)
predict.show()
```

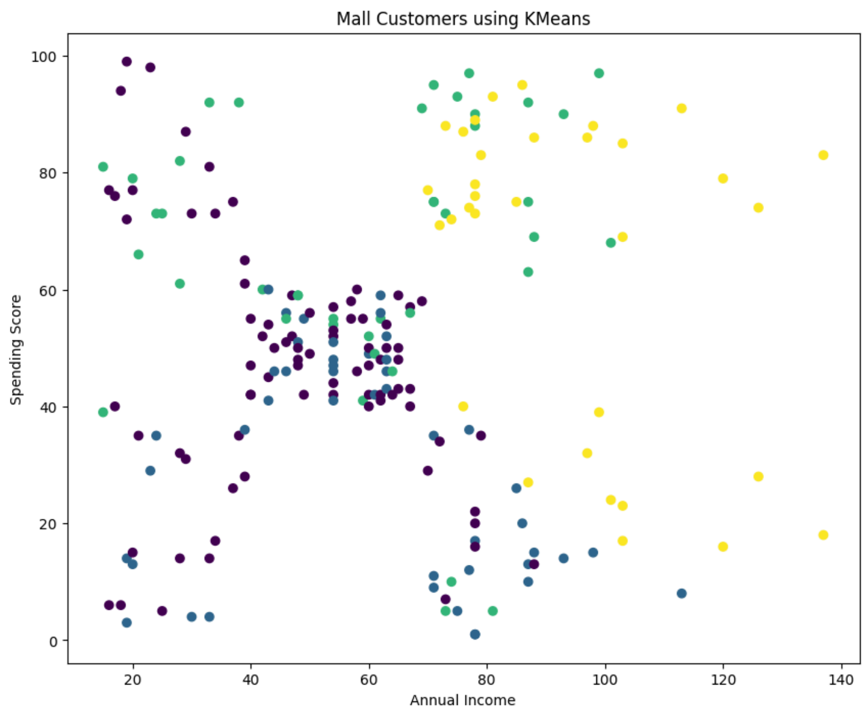
```
+----------+------+---+------------------+----------------------+----------+--------------------+--------------------+----------+
|CustomerID| Genre|Age|Annual Income (k$)|Spending Score (1-100)|Gender_num| Independent Feature|      Scaled Feature|prediction|
+----------+------+---+------------------+----------------------+----------+--------------------+--------------------+----------+
|         1|  Male| 19|                15|                    39|       1.0|[19.0,1.0,15.0,39.0]|[1.36015391423519...|         2|
|         2|  Male| 21|                15|                    81|       1.0|[21.0,1.0,15.0,81.0]|[1.50332801047048...|         2|
|         3|Female| 20|                16|                     6|       0.0| [20.0,0.0,16.0,6.0]|[1.43174096235284...|         0|
|         4|Female| 23|                16|                    77|       0.0|[23.0,0.0,16.0,77.0]|[1.64650210670576...|         0|
|         5|Female| 31|                17|                    40|       0.0|[31.0,0.0,17.0,40.0]|[2.21919849164690...|         0|
|         6|Female| 22|                17|                    76|       0.0|[22.0,0.0,17.0,76.0]|[1.57491505858812...|         0|
|         7|Female| 35|                18|                     6|       0.0| [35.0,0.0,18.0,6.0]|[2.50554668411747...|         0|
|         8|Female| 23|                18|                    94|       0.0|[23.0,0.0,18.0,94.0]|[1.64650210670576...|         0|
|         9|  Male| 64|                19|                     3|       1.0| [64.0,1.0,19.0,3.0]|[4.58157107952909...|         1|
|        10|Female| 30|                19|                    72|       0.0|[30.0,0.0,19.0,72.0]|[2.14761144352926...|         0|
|        11|  Male| 67|                19|                    14|       1.0|[67.0,1.0,19.0,14.0]|[4.79633222388201...|         1|
|        12|Female| 35|                19|                    99|       0.0|[35.0,0.0,19.0,99.0]|[2.50554668411747...|         0|
|         13|Female| 58|                 20|                     15|       0.0|[58.0,0.0,20.0,15.0]|[4.15204879082323...|         0|
|         14|Female| 24|                 20|                     77|       0.0|[24.0,0.0,20.0,77.0]|[1.71808915482340...|         0|
|         15|  Male| 37|                 20|                     13|       1.0|[37.0,1.0,20.0,13.0]|[2.64872078035275...|         1|
|         16|  Male| 22|                 20|                     79|       1.0|[22.0,1.0,20.0,79.0]|[1.57491505858812...|         2|
|         17|Female| 35|                 21|                     35|       0.0|[35.0,0.0,21.0,35.0]|[2.50554668411747...|         0|
|         18|  Male| 20|                 21|                     66|       1.0|[20.0,1.0,21.0,66.0]|[1.43174096235284...|         2|
|         19|  Male| 52|                 23|                     29|       1.0|[52.0,1.0,23.0,29.0]|[3.72252650211738...|         1|
|         20|Female| 35|                 23|                     98|       0.0|[35.0,0.0,23.0,98.0]|[2.50554668411747...|         0|
+----------+------+---+-----------------+---------------------+----------+--------------------+--------------------+----------+
only showing top 20 rows
```

```
i = 0
for center in model.clusterCenters():
  print(f'Cluster Center for cluster {i} is :{center}')
  i = i+1
```

```
Cluster Center for cluster 0 is :[2.80250037 0.         1.80592432 1.81908995]
Cluster Center for cluster 1 is :[3.7225265  1.96485882 2.28950968 1.17808185]
Cluster Center for cluster 2 is :[1.96602478 2.00951471 2.27050369 2.57847736]
Cluster Center for cluster 3 is :[2.45782199 0.18268316 3.59855911 2.50652384]
```

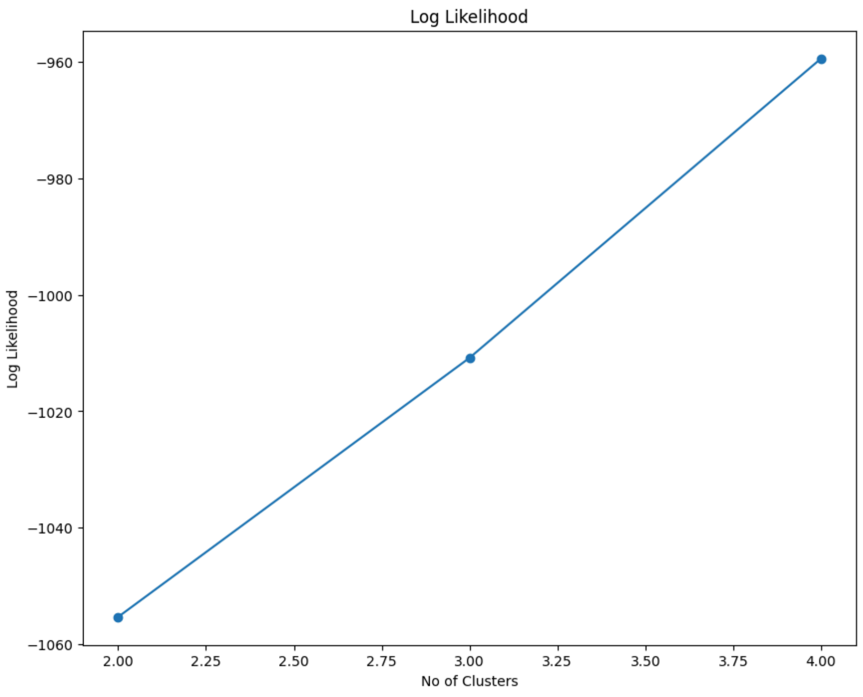
```
prediction_data = predict.toPandas()
```

```
plt.figure(figsize=(10,8))
plt.scatter(prediction_data['Annual Income (k$)'],prediction_data['Spending Score (1-100)'],c=prediction_data['prediction'],cmap='viridis
plt.xlabel('Annual Income')
plt.ylabel('Spending Score')
plt.title('Mall Customers using KMeans')
plt.show()
```



```
k_range = range(2,5)
log_likelihood = []
for k in k_range:
  gm = GaussianMixture(featuresCol='Scaled Feature',k=k)
  model = gm.fit(data)
  log_val = model.summary.logLikelihood
  log_likelihood.append(log_val)
```

```
plt.figure(figsize=(10,8))
plt.plot(k_range,log_likelihood,marker='o')
plt.xlabel('No of Clusters')
plt.ylabel('Log Likelihood')
plt.title('Log Likelihood')
plt.show()
```



```
gm = GaussianMixture(k=4,featuresCol='Scaled Feature',seed=44)
model = gm.fit(data)
pred = model.transform(data)
pred.show()
```

```
+----------+------+---+------------------+----------------------+----------+--------------------+--------------------+------------
|CustomerID| Genre|Age|Annual Income (k$)|Spending Score (1-100)|Gender_num| Independent Feature|      Scaled Feature|         proba
+----------+------+---+------------------+----------------------+----------+--------------------+--------------------+------------
|         1|  Male| 19|                15|                    39|       1.0|[19.0,1.0,15.0,39.0]|[1.36015391423519...|[1.2745738414
|         2|  Male| 21|                15|                    81|       1.0|[21.0,1.0,15.0,81.0]|[1.50332801047048...|[3.0807430581
|         3|Female| 20|                16|                     6|       0.0| [20.0,0.0,16.0,6.0]|[1.43174096235284...|[2.0507353664
|         4|Female| 23|                16|                    77|       0.0|[23.0,0.0,16.0,77.0]|[1.64650210670576...|[3.1375090915
|         5|Female| 31|                17|                    40|       0.0|[31.0,0.0,17.0,40.0]|[2.21919849164690...|[4.5835901455
|         6|Female| 22|                17|                    76|       0.0|[22.0,0.0,17.0,76.0]|[1.57491505858812...|[7.0312215681
|         7|Female| 35|                18|                     6|       0.0| [35.0,0.0,18.0,6.0]|[2.50554668411747...|[2.0309324254
|         8|Female| 23|                18|                    94|       0.0|[23.0,0.0,18.0,94.0]|[1.64650210670576...|[6.5092223302
|         9|  Male| 64|                19|                     3|       1.0| [64.0,1.0,19.0,3.0]|[4.58157107952909...|[2.9995993285
|        10|Female| 30|                19|                    72|       0.0|[30.0,0.0,19.0,72.0]|[2.14761144352926...|[1.1133876914
|        11|  Male| 67|                19|                    14|       1.0|[67.0,1.0,19.0,14.0]|[4.79633222388201...|[1.3408600905
|        12|Female| 35|                19|                    99|       0.0|[35.0,0.0,19.0,99.0]|[2.50554668411747...|[1.8152524409
|        13|Female| 58|                20|                    15|       0.0|[58.0,0.0,20.0,15.0]|[4.15204879082323...|[2.4237800638
|        14|Female| 24|                20|                    77|       0.0|[24.0,0.0,20.0,77.0]|[1.71808915482340...|[1.0495704647
|        15|  Male| 37|                20|                    13|       1.0|[37.0,1.0,20.0,13.0]|[2.64872078035275...|[1.9158921936
|        16|  Male| 22|                20|                    79|       1.0|[22.0,1.0,20.0,79.0]|[1.57491505858812...|[4.6192856259
|        17|Female| 35|                21|                    35|       0.0|[35.0,0.0,21.0,35.0]|[2.50554668411747...|[6.3415822081
|        18|  Male| 20|                21|                    66|       1.0|[20.0,1.0,21.0,66.0]|[1.43174096235284...|[9.7566958868
|        19|  Male| 52|                23|                    29|       1.0|[52.0,1.0,23.0,29.0]|[3.72252650211738...|[1.1677064662
|        20|Female| 35|                23|                    98|       0.0|[35.0,0.0,23.0,98.0]|[2.50554668411747...|[8.4238451192
+----------+------+---+------------------+----------------------+----------+--------------------+--------------------+------------
only showing top 20 rows
```

```
i = 0
for center in model.gaussiansDF.collect()[0]:
  print(f'Cluster Center for cluster {i} is :{center}')
  i = i+1
```

```
     Cluster Center for cluster 0 is :[3.1174713150722932,0.8694667730250536,2.1296237835631975,1.911426723303635]
     Cluster Center for cluster 1 is :DenseMatrix([[ 1.37736248e+00,  1.30184535e-01, -7.04749488e-02,
                   -2.06293004e-05],
                  [ 1.30184535e-01,  9.91233797e-01, -2.08661837e-02,
                    1.42535681e-02],
                  [-7.04749488e-02, -2.08661837e-02,  1.12037188e-01,
                   -2.36294952e-02],
                  [-2.06293004e-05,  1.42535681e-02, -2.36294952e-02,
                    6.52266917e-02]])
```

```
eval = ClusteringEvaluator(featuresCol='Scaled Feature')
score = eval.evaluate(pred)
print(score)
```

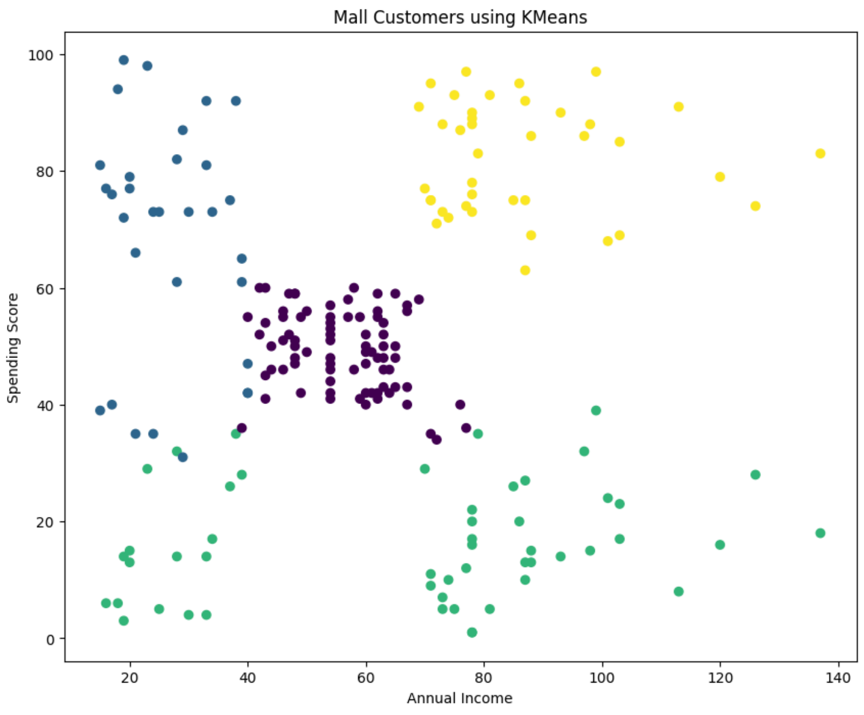
```
     0.26131604489712135
```

```
cluster_data = pred.groupBy('prediction').count().orderBy('prediction')
cluster_data.show()
```

```
     +----------+-----+
     |prediction|count|
     +----------+-----+
     |         0|   79|
     |         1|   31|
     |         2|   51|
     |         3|   39|
     +----------+-----+
```

```
pred_data = pred.toPandas()
```

```
plt.figure(figsize=(10,8))
plt.scatter(pred_data['Annual Income (k$)'],pred_data['Spending Score (1-100)'],c=pred_data['prediction'],cmap='viridis')
plt.xlabel('Annual Income')
plt.ylabel('Spending Score')
plt.title('Mall Customers using KMeans')
plt.show()
```



2.

```
data = spark.read.options(header="true",inferSchema="true").csv("/content/drive/MyDrive/Big Data/Datasets/ANSUR_II_FEMALE_Public.csv")
```

```
data.printSchema()
```

```
root
 |-- SubjectId: integer (nullable = true)
 |-- abdominalextensiondepthsitting: integer (nullable = true)
 |-- acromialheight: integer (nullable = true)
 |-- acromionradialelength: integer (nullable = true)
 |-- anklecircumference: integer (nullable = true)
 |-- axillaheight: integer (nullable = true)
 |-- balloffootcircumference: integer (nullable = true)
 |-- balloffootlength: integer (nullable = true)
 |-- biacromialbreadth: integer (nullable = true)
 |-- bicepscircumferenceflexed: integer (nullable = true)
 |-- bicristalbreadth: integer (nullable = true)
 |-- bideltoidbreadth: integer (nullable = true)
 |-- bimalleolarbreadth: integer (nullable = true)
 |-- bitragionchinarc: integer (nullable = true)
 |-- bitragionsubmandibulararc: integer (nullable = true)
 |-- bizygomaticbreadth: integer (nullable = true)
 |-- buttockcircumference: integer (nullable = true)
 |-- buttockdepth: integer (nullable = true)
 |-- buttockheight: integer (nullable = true)
 |-- buttockkneelength: integer (nullable = true)
 |-- buttockpopliteallength: integer (nullable = true)
 |-- calfcircumference: integer (nullable = true)
 |-- cervicaleheight: integer (nullable = true)
 |-- chestbreadth: integer (nullable = true)
 |-- chestcircumference: integer (nullable = true)
 |-- chestdepth: integer (nullable = true)
 |-- chestheight: integer (nullable = true)
 |-- crotchheight: integer (nullable = true)
 |-- crotchlengthomphalion: integer (nullable = true)
 |-- crotchlengthposterioromphalion: integer (nullable = true)
 |-- earbreadth: integer (nullable = true)
 |-- earlength: integer (nullable = true)
 |-- earprotrusion: integer (nullable = true)
 |-- elbowrestheight: integer (nullable = true)
 |-- eyeheightsitting: integer (nullable = true)
 |-- footbreadthhorizontal: integer (nullable = true)
 |-- footlength: integer (nullable = true)
 |-- forearmcenterofgriplength: integer (nullable = true)
 |-- forearmcircumferenceflexed: integer (nullable = true)
 |-- forearmforearmbreadth: integer (nullable = true)
 |-- forearmhandlength: integer (nullable = true)
 |-- functionalleglength: integer (nullable = true)
 |-- handbreadth: integer (nullable = true)
 |-- handcircumference: integer (nullable = true)
 |-- handlength: integer (nullable = true)
 |-- headbreadth: integer (nullable = true)
 |-- headcircumference: integer (nullable = true)
 |-- headlength: integer (nullable = true)
 |-- heelanklecircumference: integer (nullable = true)
 |-- heelbreadth: integer (nullable = true)
 |-- hipbreadth: integer (nullable = true)
 |-- hipbreadthsitting: integer (nullable = true)
 |-- iliocristaleheight: integer (nullable = true)
 |-- interpupillarybreadth: integer (nullable = true)
 |-- interscyei: integer (nullable = true)
 |-- interscyeii: integer (nullable = true)
 |-- kneeheightmidpatella: integer (nullable = true)
```

```
continuous_features = detect_continuous_variables(data,10)
continuous_features
```

```
/usr/local/lib/python3.10/dist-packages/pyspark/sql/functions.py:3796: FutureWarning: Deprecated in 2.1, use approx_count_distin
  warnings.warn("Deprecated in 2.1, use approx_count_distinct instead.", FutureWarning)
['SubjectId',
 'abdominalextensiondepthsitting',
 'acromialheight',
 'acromionradialelength',
 'anklecircumference',
 'axillaheight',
 'balloffootcircumference',
 'balloffootlength',
 'biacromialbreadth',
 'bicepscircumferenceflexed',
 'bicristalbreadth',
 'bideltoidbreadth',
 'bimalleolarbreadth',
 'bitragionchinarc',
 'bitragionsubmandibulararc',
 'bizygomaticbreadth',
 'buttockcircumference',
 'buttockdepth',
 'buttockheight',
 'buttockkneelength',
 'buttockpopliteallength',
 'calfcircumference',
 'cervicaleheight',
 'chestbreadth',
 'chestcircumference',
 'chestdepth',
 'chestheight',
 'crotchheight',
 'crotchlengthomphalion',
 'crotchlengthposterioromphalion',
 'earbreadth',
 'earlength',
 'earprotrusion',
 'elbowrestheight',
 'eyeheightsitting',
 'footbreadthhorizontal',
 'footlength',
 'forearmcenterofgriplength',
 'forearmcircumferenceflexed',
 'forearmforearmbreadth',
 'forearmhandlength',
 'functionalleglength',
 'handbreadth',
 'handcircumference',
 'handlength',
 'headbreadth',
 'headcircumference',
 'headlength',
 'heelanklecircumference',
 'heelbreadth',
 'hipbreadth',
 'hipbreadthsitting',
 'iliocristaleheight',
 'interpupillarybreadth',
```

```
indexer = StringIndexer(inputCol="WritingPreference",outputCol="WritingPreference_num")
data = indexer.fit(data).transform(data)
data.show()
```

```
+---------+------------------------------+--------------+---------------------+------------------+------------+--------------------
|SubjectId|abdominalextensiondepthsitting|acromialheight|acromionradialelength|anklecircumference|axillaheight|balloffootcircumferen
+---------+------------------------------+--------------+---------------------+------------------+------------+--------------------
|    10037|                           231|          1282|                  301|               204|        1180|
|    10038|                           194|          1379|                  320|               207|        1292|
|    10042|                           183|          1369|                  329|               233|        1271|
|    10043|                           261|          1356|                  306|               214|        1250|
|    10051|                           309|          1303|                  308|               214|        1210|
|    10053|                           272|          1428|                  326|               228|        1326|
|    10061|                           261|          1352|                  306|               223|        1246|
|    10070|                           229|          1383|                  327|               205|        1281|
|    10077|                           213|          1237|                  308|               193|        1135|
|    10080|                           281|          1301|                  278|               208|        1194|
|    10095|                           274|          1395|                  318|               245|        1291|
|    10101|                           284|          1360|                  331|               220|        1260|
|    10105|                           279|          1368|                  342|               206|        1252|
|    10111|                           214|          1368|                  325|               220|        1258|
|    10121|                           265|          1315|                  320|               194|        1218|
|    10127|                           225|          1371|                  312|               223|        1265|
|    10129|                           193|          1337|                  307|               218|        1237|
|    10131|                           201|          1387|                  302|               197|        1286|
|    10136|                           233|          1374|                  342|               211|        1281|
|    10138|                           214|          1309|                  294|               209|        1213|
+---------+------------------------------+--------------+---------------------+------------------+------------+--------------------
only showing top 20 rows
```

```
assembler = VectorAssembler(inputCols=continuous_features,outputCol="Independent Feature")
data = assembler.transform(data)
data.show()
```

```
+---------+------------------------------+--------------+---------------------+------------------+------------+--------------------
|SubjectId|abdominalextensiondepthsitting|acromialheight|acromionradialelength|anklecircumference|axillaheight|balloffootcircumferen
+---------+------------------------------+--------------+---------------------+------------------+------------+--------------------
|    10037|                           231|          1282|                  301|               204|        1180|
|    10038|                           194|          1379|                  320|               207|        1292|
|    10042|                           183|          1369|                  329|               233|        1271|
|    10043|                           261|          1356|                  306|               214|        1250|
|    10051|                           309|          1303|                  308|               214|        1210|
|    10053|                           272|          1428|                  326|               228|        1326|
|    10061|                           261|          1352|                  306|               223|        1246|
|    10070|                           229|          1383|                  327|               205|        1281|
|    10077|                           213|          1237|                  308|               193|        1135|
|    10080|                           281|          1301|                  278|               208|        1194|
|    10095|                           274|          1395|                  318|               245|        1291|
|    10101|                           284|          1360|                  331|               220|        1260|
|    10105|                           279|          1368|                  342|               206|        1252|
|    10111|                           214|          1368|                  325|               220|        1258|
|    10121|                           265|          1315|                  320|               194|        1218|
|    10127|                           225|         1371|                 312|               223|       1265|
|    10129|                           193|         1337|                 307|               218|       1237|
|    10131|                           201|         1387|                 302|               197|       1286|
|    10136|                           233|         1374|                 342|               211|       1281|
|    10138|                           214|         1309|                 294|               209|       1213|
+---------+------------------------------+-------------+--------------------+------------------+-----------+--------------------
only showing top 20 rows
```

```
sc = StandardScaler(inputCol="Independent Feature",outputCol="Scaled Feature")
data = sc.fit(data).transform(data)
```

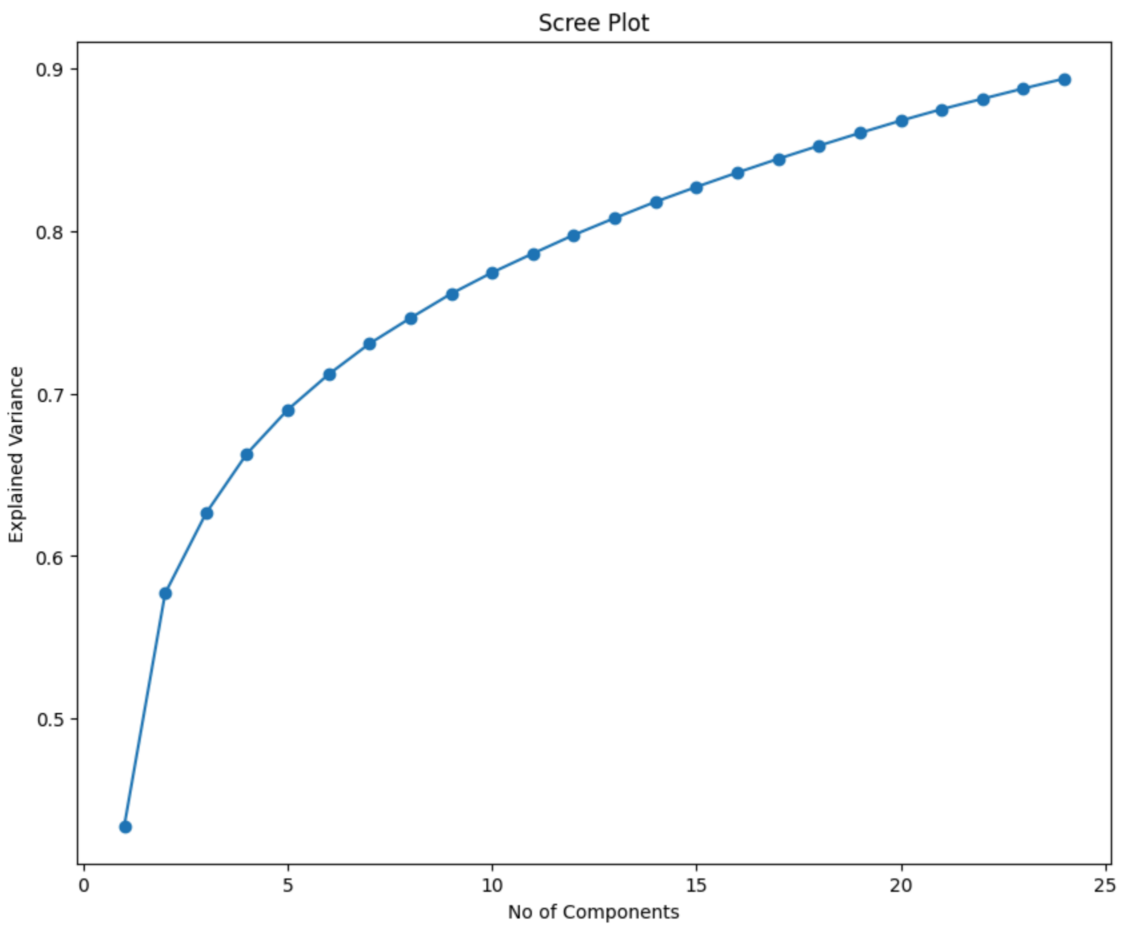
```
len(continuous_features)
```

```
98
```

```
pca_model = PCA(k=24,inputCol="Scaled Feature",outputCol="PCA Feature")
model = pca_model.fit(data)
variance_explained = model.explainedVariance.cumsum()
```

```
k_comp = len([var for var in variance_explained if var<=0.95])+1
```

```
plt.figure(figsize=(10,8))
plt.plot(range(1,len(variance_explained)+1),variance_explained,marker='o',linestyle='-')
plt.xlabel("No of Components")
plt.ylabel("Explained Variance")
plt.title("Scree Plot")
plt.show()
```



```
pca_model = PCA(k=5,inputCol='Scaled Feature',outputCol='PCA Feature')
model = pca_model.fit(data)
final_data = model.transform(data)
final_data.show()
```

```
+---------+------------------------------+-------------+--------------------+------------------+-----------+--------------------
|SubjectId|abdominalextensiondepthsitting|acromialheight|acromionradialelength|anklecircumference|axillaheight|balloffootcircumferer
+---------+------------------------------+-------------+--------------------+------------------+-----------+--------------------
|    10037|                           231|         1282|                 301|               204|       1180|
|    10038|                           194|         1379|                 320|               207|       1292|
|    10042|                           183|         1369|                 329|               233|       1271|
|    10043|                           261|         1356|                 306|               214|       1250|
|    10051|                              309|          1303|                    308|                 214|        1210|
|    10053|                              272|          1428|                    326|                 228|        1326|
|    10061|                              261|          1352|                    306|                 223|        1246|
|    10070|                              229|          1383|                    327|                 205|        1281|
|    10077|                              213|          1237|                    308|                 193|        1135|
|    10080|                              281|          1301|                    278|                 208|        1194|
|    10095|                              274|          1395|                    318|                 245|        1291|
|    10101|                              284|          1360|                    331|                 220|        1260|
|    10105|                              279|          1368|                    342|                 206|        1252|
|    10111|                              214|          1368|                    325|                 220|        1258|
|    10121|                              265|          1315|                    320|                 194|        1218|
|    10127|                              225|          1371|                    312|                 223|        1265|
|    10129|                              193|          1337|                    307|                 218|        1237|
|    10131|                              201|          1387|                    302|                 197|        1286|
|    10136|                              233|          1374|                    342|                 211|        1281|
|    10138|                              214|          1309|                    294|                 209|        1213|
+---------+-------------------------------+--------------+-----------------------+------------------+------------+--------------------
only showing top 20 rows
```

```
lr =  LogisticRegression(featuresCol="PCA Feature",labelCol="WritingPreference_num")
train_data,test_data = final_data.randomSplit([0.70,0.30],seed=44)
lr_model = lr.fit(final_data)
pred = lr_model.transform(final_data)
pred.show()
```

```
+---------+-------------------------------+--------------+-----------------------+------------------+------------+--------------------
|SubjectId|abdominalextensiondepthsitting|acromialheight|acromionradialelength|anklecircumference|axillaheight|balloffootcircumferer
+---------+-------------------------------+--------------+-----------------------+------------------+------------+--------------------
|    10037|                              231|          1282|                    301|                 204|        1180|
|    10038|                              194|          1379|                    320|                 207|        1292|
|    10042|                              183|          1369|                    329|                 233|        1271|
|    10043|                              261|          1356|                    306|                 214|        1250|
|    10051|                              309|          1303|                    308|                 214|        1210|
|    10053|                              272|          1428|                    326|                 228|        1326|
|    10061|                              261|          1352|                    306|                 223|        1246|
|    10070|                              229|          1383|                    327|                 205|        1281|
|    10077|                              213|          1237|                    308|                 193|        1135|
|    10080|                              281|          1301|                    278|                 208|        1194|
|    10095|                              274|          1395|                    318|                 245|        1291|
|    10101|                              284|          1360|                    331|                 220|        1260|
|    10105|                              279|          1368|                    342|                 206|        1252|
|    10111|                              214|          1368|                    325|                 220|        1258|
|    10121|                              265|          1315|                    320|                 194|        1218|
|    10127|                              225|          1371|                    312|                 223|        1265|
|    10129|                              193|          1337|                    307|                 218|        1237|
|    10131|                              201|          1387|                    302|                 197|        1286|
|    10136|                              233|          1374|                    342|                 211|        1281|
|    10138|                              214|          1309|                    294|                 209|        1213|
+---------+-------------------------------+--------------+-----------------------+------------------+------------+--------------------
only showing top 20 rows
```

```
eval = MulticlassClassificationEvaluator(labelCol="WritingPreference_num",predictionCol="prediction",metricName="accuracy")
accuracy = eval.evaluate(pred)
print(accuracy)
```

```
0.8927492447129909
```

```
svm_model = LinearSVC()
one = OneVsRest(featuresCol='PCA Feature',labelCol='WritingPreference_num',classifier=svm_model)
train_data,test_data = final_data.randomSplit([0.70,0.30],seed=44)
one_model = one.fit(final_data)
prediction = one_model.transform(final_data)
prediction.show()
```

```
+---------+-------------------------------+--------------+-----------------------+------------------+------------+--------------------
|SubjectId|abdominalextensiondepthsitting|acromialheight|acromionradialelength|anklecircumference|axillaheight|balloffootcircumferer
+---------+-------------------------------+--------------+-----------------------+------------------+------------+--------------------
|    10037|                              231|          1282|                    301|                 204|        1180|
|    10038|                              194|          1379|                    320|                 207|        1292|
|    10042|                              183|          1369|                    329|                 233|        1271|
|    10043|                              261|          1356|                    306|                 214|        1250|
|    10051|                              309|          1303|                    308|                 214|        1210|
|    10053|                              272|          1428|                    326|                 228|        1326|
|    10061|                              261|          1352|                    306|                 223|        1246|
|    10070|                              229|          1383|                    327|                 205|        1281|
|    10077|                              213|          1237|                    308|                 193|        1135|
|    10080|                              281|          1301|                    278|                 208|        1194|
|    10095|                              274|          1395|                    318|                 245|        1291|
|    10101|                              284|          1360|                    331|                 220|        1260|
|    10105|                              279|          1368|                    342|                 206|        1252|
|    10111|                              214|          1368|                    325|                 220|        1258|
|    10121|                             265|          1315|                  320|                194|       1218|
|    10127|                             225|          1371|                  312|                223|       1265|
|    10129|                             193|          1337|                  307|                218|       1237|
|    10131|                             201|          1387|                  302|                197|       1286|
|    10136|                             233|          1374|                  342|                211|       1281|
|    10138|                             214|          1309|                  294|                209|       1213|
+---------+------------------------------+--------------+---------------------+-------------------+-----------+--------------------
only showing top 20 rows
```

```
eval = MulticlassClassificationEvaluator(labelCol="WritingPreference_num",predictionCol="prediction",metricName="accuracy")
accuracy = eval.evaluate(prediction)
print(accuracy)
```

```
0.8927492447129909
```

Module 6

```
document = pd.read_csv('/content/drive/MyDrive/Big Data/Datasets/npr.csv')
document
```

	Article
0	In the Washington of 2016, even when the polic...
1	Donald Trump has used Twitter — his prefe...
2	Donald Trump is unabashedly praising Russian...
3	Updated at 2:50 p. m. ET, Russian President Vl...
4	From photography, illustration and video, to d...
...	...
11987	The number of law enforcement officers shot an...
11988	Trump is busy these days with victory tours,...
11989	It’s always interesting for the Goats and Soda...
11990	The election of Donald Trump was a surprise to...
11991	Voters in the English city of Sunderland did s...

11992 rows × 1 columns

```
document_list = document['Article']
```

```
tokenizer = RegexpTokenizer(r'\w+')
tfidf = TfidfVectorizer(lowercase=True,stop_words='english',ngram_range=(1,1),max_features=1000,tokenizer=tokenizer.tokenize,max_df=0.5
train_data = tfidf.fit_transform(document_list)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/feature_extraction/text.py:528: UserWarning: The parameter 'token_pattern' will not
  warnings.warn(
```

```
n_components = 10
svd = TruncatedSVD(n_components=n_components,n_iter=100,random_state=42)
svd.fit_transform(train_data)
```

```
array([[ 0.42561111,  0.44666762,  0.01360668, ...,  0.02022168,
         0.10812173,  0.00835765],
       [ 0.37796404,  0.42833585,  0.00170011, ..., -0.00613033,
         0.00593589,  0.09518064],
       [ 0.40815687,  0.56938431,  0.05043533, ...,  0.00693158,
         0.05322761,  0.06737707],
       ...,
       [ 0.33187054, -0.10945567, -0.09565988, ..., -0.07681644,
         0.06641655,  0.11307751],
       [ 0.31811317,  0.29630719,  0.12700604, ...,  0.11885517,
         0.0543419 , -0.07197287],
       [ 0.34310391,  0.01726595, -0.09290905, ...,  0.06577147,
         0.0387871 ,  0.00337945]])
```

```
sigma = svd.singular_values_
V_transpose = svd.components_.T
terms = tfidf.get_feature_names_out()
```

```
for index,components in enumerate(svd.components_):
  zipped = zip(terms,components)
  top_terms_keys = sorted(zipped,key=lambda t:t[1],reverse=True)[:10]
  top_terms_list = list(dict(top_terms_keys).keys())
  print('Topic '+str(index)+":",top_terms_list)
```

```
Topic 0: ['trump', 'u', 'president', 'think', 'know', 'state', 'don', 'clinton', 'women', 'year']
Topic 1: ['trump', 'clinton', 'president', 'campaign', 'donald', 'obama', 'republican', 'election', 'house', 'presidential']
Topic 2: ['music', 'trump', 'album', 'song', 'songs', 'band', 'clinton', 'love', 'think', 'really']
Topic 3: ['police', 'u', 'reports', 'attack', 'music', 'city', 'russia', 'killed', 'court', 'officers']
Topic 4: ['clinton', 'sanders', 'police', 'women', 'voters', 'black', 'hillary', 'state', 'democratic', 'vote']
Topic 5: ['music', 'album', 'health', 'song', 'band', 'songs', 'care', 'insurance', 'clinton', 'medicaid']
Topic 6: ['students', 'police', 'school', 'trump', 'court', 'law', 'schools', 'education', 'federal', 'care']
Topic 7: ['students', 'school', 'schools', 'education', 'music', 'state', 'student', 'u', 'teachers', 'album']
Topic 8: ['women', 'court', 'u', 'supreme', 'obama', 'president', 'men', 'woman', 'senate', 'war']
Topic 9: ['women', 'students', 'music', 'zika', 'school', 'album', 'trump', 'virus', 'study', 'song']
```

```
data = spark.read.options(inferSchema='true',mode='DROPMALFORMED',header='true').csv("/content/drive/MyDrive/Big Data/Datasets/npr.csv"
data.show()
```

```
+--------------------+
|             Article|
+--------------------+
|In the Washington...|
|  Donald Trump ha...|
|  Donald Trump is...|
|Updated at 2:50 p...|
|From photography,...|
|I did not want to...|
|With a   who has ...|
|I was standing by...|
|If movies were tr...|
|Eighteen years ag...|
|For years now, so...|
|For years now, so...|
|The Colorado Rive...|
|For the last inst...|
|Being overweight ...|
|Who’s the YouTube...|
|Here’s a quick ro...|
|Ben Johnston does...|
|David Bowie, Prin...|
|In November, the ...|
+--------------------+
only showing top 20 rows
```

```
tokenizer = Tokenizer(inputCol='Article',outputCol='Words')
word_data = tokenizer.transform(data)
word_data.show()
```

```
+--------------------+--------------------+
|             Article|               Words|
+--------------------+--------------------+
|In the Washington...|[in, the, washing...|
|  Donald Trump ha...|[, , donald, trum...|
|  Donald Trump is...|[, , donald, trum...|
|Updated at 2:50 p...|[updated, at, 2:5...|
|From photography,...|[from, photograph...|
|I did not want to...|[i, did, not, wan...|
|With a   who has ...|[with, a, , , who...|
|I was standing by...|[i, was, standing...|
|If movies were tr...|[if, movies, were...|
|Eighteen years ag...|[eighteen, years,...|
|For years now, so...|[for, years, now,...|
|For years now, so...|[for, years, now,...|
|The Colorado Rive...|[the, colorado, r...|
|For the last inst...|[for, the, last, ...|
|Being overweight ...|[being, overweigh...|
|Who’s the YouTube...|[who’s, the, yout...|
|Here’s a quick ro...|[here’s, a, quick...|
|Ben Johnston does...|[ben, johnston, d...|
|David Bowie, Prin...|[david, bowie,, p...|
|In November, the ...|[in, november,, t...|
+--------------------+--------------------+
only showing top 20 rows
```

```
remover = StopWordsRemover(inputCol='Words',outputCol='Filtered')
filter_data = remover.transform(word_data)
filter_data.show()
```

```
+--------------------+--------------------+--------------------+
|             Article|               Words|            Filtered|
+--------------------+--------------------+--------------------+
|In the Washington...|[in, the, washing...|[washington, 2016...|
|  Donald Trump ha...|[, , donald, trum...|[, , donald, trum...|
|  Donald Trump is...|[, , donald, trum...|[, , donald, trum...|
|Updated at 2:50 p...|[updated, at, 2:5...|[updated, 2:50, p...|
|From photography,...|[from, photograph...|[photography,, il...|
|I did not want to...|[i, did, not, wan...|[want, join, yoga...|
|With a   who has ...|[with, a, , , who...|[, , publicly, su...|
|I was standing by...|[i, was, standing...|[standing, airpor...|
|If movies were tr...|[if, movies, were...|[movies, trying, ...|
|Eighteen years ag...|[eighteen, years,...|[eighteen, years,...|
|For years now, so...|[for, years, now,...|[years, now,, bes...|
|For years now, so...|[for, years, now,...|[years, now,, bes...|
|The Colorado Rive...|[the, colorado, r...|[colorado, river,...|
|For the last inst...|[for, the, last, ...|[last, installmen...|
|Being overweight ...|[being, overweigh...|[overweight, rais...|
|Who’s the YouTube...|[who’s, the, yout...|[who’s, youtube, ...|
|Here’s a quick ro...|[here’s, a, quick...|[here’s, quick, r...|
|Ben Johnston does...|[ben, johnston, d...|[ben, johnston, d...|
|David Bowie, Prin...|[david, bowie,, p...|[david, bowie,, p...|
|In November, the ...|[in, november,, t...|[november,, typic...|
+--------------------+--------------------+--------------------+
only showing top 20 rows
```

```
cv = CountVectorizer(inputCol='Filtered',outputCol='features')
cv_model = cv.fit(filter_data)
vector_data = cv_model.transform(filter_data)
vector_data.show()
```

```
+--------------------+--------------------+--------------------+--------------------+
|             Article|               Words|            Filtered|            features|
+--------------------+--------------------+--------------------+--------------------+
|In the Washington...|[in, the, washing...|[washington, 2016...|(248931,[0,1,4,6,...|
|  Donald Trump ha...|[, , donald, trum...|[, , donald, trum...|(248931,[0,1,4,5,...|
|  Donald Trump is...|[, , donald, trum...|[, , donald, trum...|(248931,[0,1,4,7,...|
|Updated at 2:50 p...|[updated, at, 2:5...|[updated, 2:50, p...|(248931,[0,1,2,7,...|
|From photography,...|[from, photograph...|[photography,, il...|(248931,[0,1,2,3,...|
|I did not want to...|[i, did, not, wan...|[want, join, yoga...|(248931,[0,1,2,3,...|
|With a   who has ...|[with, a, , , who...|[, , publicly, su...|(248931,[0,1,2,4,...|
|I was standing by...|[i, was, standing...|[standing, airpor...|(248931,[0,1,2,4,...|
|If movies were tr...|[if, movies, were...|[movies, trying, ...|(248931,[0,1,2,4,...|
|Eighteen years ag...|[eighteen, years,...|[eighteen, years,...|(248931,[0,1,2,4,...|
|For years now, so...|[for, years, now,...|[years, now,, bes...|(248931,[0,1,3,4,...|
|For years now, so...|[for, years, now,...|[years, now,, bes...|(248931,[0,1,4,5,...|
|The Colorado Rive...|[the, colorado, r...|[colorado, river,...|(248931,[0,1,2,4,...|
|For the last inst...|[for, the, last, ...|[last, installmen...|(248931,[0,6,8,26...|
|Being overweight ...|[being, overweigh...|[overweight, rais...|(248931,[0,1,2,3,...|
|Who’s the YouTube...|[who’s, the, yout...|[who’s, youtube, ...|(248931,[0,1,3,4,...|
|Here’s a quick ro...|[here’s, a, quick...|[here’s, quick, r...|(248931,[0,2,4,5,...|
|Ben Johnston does...|[ben, johnston, d...|[ben, johnston, d...|(248931,[0,1,2,3,...|
|David Bowie, Prin...|[david, bowie,, p...|[david, bowie,, p...|(248931,[0,1,3,4,...|
|In November, the ...|[in, november,, t...|[november,, typic...|(248931,[0,1,2,4,...|
+--------------------+--------------------+--------------------+--------------------+
only showing top 20 rows
```

```
vocab = cv_model.vocabulary
vocab
```

```
['',
 '—',
 'says',
 'people',
 'one',
 'it’s',
 'like',
 'said',
 'new',
 'also',
 'says.',
 'trump',
 'u.',
 'many',
 's.',
 'get',
 'even',
 'first',
 '.',
 'that’s',
 'think',
 'don’t',
 '”i',
 'two',
 'time',
 'going',
 'make',
 'president',
 'last',
 'years',
 '”the',
 'much',
 'back',
 'way',
 'really',
 'health',
 'state',
 'know',
 'may',
 'still',
 'say',
 'told',
 '..',
 'made',
 'see',
 'percent',
 'want',
 'said.',
 'there’s',
 'take',
 'go',
 'lot',
 'part',
 'called',
 'white',
 'according',
 'work',
 'something',
```

```
num_topics = 10
lda = LDA(k=num_topics,maxIter=10)
ldaModel = lda.fit(vector_data)
topics = ldaModel.describeTopics(maxTermsPerTopic=5)
topics.show(truncate=False)
```

```
+-----+-------------------------+-----------------------------------------------------------------------------------------------------
|topic|termIndices              |termWeights
+-----+-------------------------+-----------------------------------------------------------------------------------------------------
|0    |[11550, 0, 246, 422, 1074]|[1.3880066284309593E-5, 1.2739454397687274E-5, 1.2425172875177488E-5, 1.1401479854166915E-5, 1.057
|1    |[0, 2, 1, 12, 3]         |[0.001311474895336409, 1.9768850216327955E-4, 1.4883797098004723E-4, 1.431773896012635E-4, 1.4307
|2    |[0, 2, 10, 8, 5]         |[8.442514887680619E-4, 1.4532189355664904E-4, 1.3365644276840336E-4, 1.2863954040857257E-4, 1.197
|3    |[0, 1, 3, 4, 5]          |[0.0037488361532890917, 5.981713284816944E-4, 4.372701929151953E-4, 4.0675329380071286E-4, 3.7906
|4    |[0, 1, 11, 7, 2]         |[0.002143540083530728, 2.728146753358114E-4, 2.538374095637789E-4, 2.067181059130293E-4, 1.99268
|5    |[0, 7, 54, 67, 4680]     |[1.241341514060456E-4, 6.910385361248523E-5, 4.5255030516775535E-5, 4.2875286996610374E-5, 4.0890
|6    |[0, 1, 2, 3, 4]          |[0.04717915849325078, 0.007572535153391111, 0.0036125716300022146, 0.003602167362939701, 0.00359
|7    |[0, 1, 11, 7, 4]         |[0.003948581697040057, 8.204240863561499E-4, 4.6846759747715046E-4, 2.738605768310781E-4, 2.66353
|8    |[0, 1, 4, 2, 5]          |[0.0061909935242382055, 0.0011225281588237745, 4.244575894616563E-4, 3.7951798472506246E-4, 3.628
|9    |[0, 11, 1, 7, 9]         |[0.0029946544784648023, 4.7350898304646143E-4, 4.4577068574923453E-4, 3.741544597659725E-4, 1.785
+-----+-------------------------+-----------------------------------------------------------------------------------------------------
```

```
def topic_render(topic):
  terms = topic['termIndices']
  return [vocab[idx] for idx in terms]

topic_words = topics.rdd.map(lambda topic:(topic['topic'],topic_render(topic)))
for topic,words in topic_words.collect():
  print(f"Topics is:{topic} with words {words}")
```

```
Topics is:0 with words ['gymnastics', '', 'report', 'russian', 'baby']
Topics is:1 with words ['', 'says', '—', 'u.', 'people']
Topics is:2 with words ['', 'says', 'says.', 'new', 'it’s']
Topics is:3 with words ['', '—', 'people', 'one', 'it’s']
Topics is:4 with words ['', '—', 'trump', 'said', 'says']
Topics is:5 with words ['', 'said', 'white', 'house', 'conway']
Topics is:6 with words ['', '—', 'says', 'people', 'one']
Topics is:7 with words ['', '—', 'trump', 'said', 'one']
Topics is:8 with words ['', '—', 'one', 'says', 'it’s']
Topics is:9 with words ['', 'trump', '—', 'said', 'also']
```

Topic modeling is a type of statistical modeling for discovering the abstract "topics" that occur in a collection of documents. It is a frequently used text-mining tool to extract hidden semantic structures in text bodies. This technique is primarily used for categorizing text in a document