*This study investigates the relationship between Big Data management tools such as Relational Database Management System (RDBMS), Hadoop, and APIs, and how they can be separately interlinked with advanced data analytics, specifically neural networks. The purpose of the study is to compare a RDBMS with Hadoop when processing a 1.31 GB dataset, and then apply a Neural Network. To expand the scope, this study will also include the usage of APIs (Keras library) for implementing Neural Networks. This study was conducted using my personal laptop to load a 1.31 GB dataset into a RDBMS and Spark. I utilized Jupyter Notebooks to interact with these two technologies, exploring computing times, roadblocks faced, and other insights. Following this, we applied the same Neural Network to predict if certain jobs are more popular based on gender. Another aspect of the study involves utilizing an API; for this, we are employing Keras and a Convolutional Neural Network (CNN). Our aim is to evaluate the performance of the CNN model in classifying movie reviews as positive or negative based on their sentiment. The research findings indicate that using RDBMS or Hadoop for data processing is not as quick and straightforward as using an API like Keras, where you simply import the data without the need to worry about how to push it into databases, this becomes clear when modelling data using NN via Jupyter Notebooks.*

Relational Database Management Systems have been well-established since the late 1970s; at that time, the concept of Big Data was not the same as it is today. As technology rapidly advanced, the industry needed to process large amounts of data. To address this need, an open-source framework for writing and running distributed applications, called Hadoop, entered the scene. These two technologies, RDMS and Hadoop, are great; however, the implementation of both requires a high level of technical software skill. This is where APIs offer a solution to this problem, which the industry refers to as Machine Learning as a Service (MLaaS), e.g., Azure ML or AWS ML, just to mention a few

The intention of this paper is to explore all three technologies—RDBMS, Hadoop, and APIs—to determine which one is the best fit for data extraction and processing in the context of Neural Networks implementation. This consideration is crucial, given that many individuals interested in Machine Learning are not software developers, and the need for a 'plug-in' to deploy their ML models is evident.

The chosen topic is Big Data and Neural Networks, with NN being considered a type of Machine Learning (ML) process known as Deep Learning . The field of Big Data is constantly growing and encompasses a need for efficient data management and processing tools. Two well-known tools for handling and analyzing large datasets are Relational Database Management Systems (RDBMS) and Hadoop. However, the rampant advancement of Machine Learning and Neural Networks, the integration of these data management tools with advanced analytics technologies is the focus of this paper.

How do Relational Database Management Systems (RDBMS) and Hadoop compare in terms of efficiency and effectiveness in processing large datasets for the application of neural networks, and how can APIs, particularly the Keras library, streamline the implementation of neural network models in data analytics.

• Examine the current state of RDBMS, Hadoop, and APIs when used in modeling NN.

• Store a 1.31 GB dataset in both an RDBMS (SQL) and Hadoop, and then retrieve the data into a Jupyter Notebook to model a neural network.

• Utilize an API (Keras) to model a neural network and compare its performance in conjunction with RDBMS and Hadoop.

• Discuss the rationale behind the selection of the NN model for both scenarios

The current state of RDBMS has evolved substantially with enhancements in storage, speed, and scalability by using cloud-based solutions. The future holds a shift for RDBMS transitioning to a NoSQL database. To understand why NoSQL is taking over RDBMS, it is crucial to talk about: Schemas, where NoSQL uses dynamic instead of static schemas; the type of data to be stored, with NoSQL databases offering advantages for hierarchical data storage due to their flexible data models and scalability, while RDBMS are not that flexible; scalability, with NoSQL depending on horizontal scalability and RDBMS on vertical scalability; and other points where NoSQL surpasses RDBMS, including data warehouse, complexity, cloud, and big data handling, and output performance.

In the era of Big Data, where we sometimes run out of storage and face difficulties on a single host due to the volume of data, Hadoop came into the scene to tackle this by offering computational capabilities over huge amounts of data. The present and future look bright for Hadoop, as some of the major Big Data companies, such as Google, Facebook, eBay, Twitter and Spotify, rely on this technology.

Machine Learning APIs have helped developers integrating data flows into complex algorithms without requiring deep expertise. These APIs were once primarily used for basic tasks like picture and speech recognition, but they have since grown to include a variety of machine learning activities, such as predictive analytics and natural language processing. Today, they are essential to leading tech companies and cutting edge industries, being ML accessible and customizable than in the past. Looking at the future ML APIs will remain as they have simplified model development across diverse environments.

The development of Deep Learning Neural Networks (DLNNs) traces back to the 1950s. The method has improved since the introduction of Convolutional Neural Networks (CNNs) by LeCun et al. in the late 1980s, which showed how good deep architectures are for image processing. In 2012 this field achieved another milestone with AlexNet's success in the ImageNet challenge, showcasing DLNNs' potential in image recognition. The introduction of Transformer models by Vaswani et al. in 2017 marked another significant advancement, in natural language processing. Neural Networks have the future guaranteed as all points mentioned earlier RDMS and Hadoop store and process data for Neural Networks models.

This section aims to explain the demo work with all its components.

Hardware and software configuration of the host laptop and the VM.

• Laptop: HP 250 G8 PC.

o Operating System (OS): Microsoft Windows 10 Pro.

o Processor: 11th Gen Intel® Core™ i7-1165G7 @ 2.80GHz, 2803 Mhz, 4 cores, 8 logical processors.

o RAM: 16GB.

o Hard Disk Drive (HDD): 237GB.

• VirtualBox: Version 7.0.14.

o OS: Ubuntu 22.04 LTS (Jammy Jellyfish) (64-bit).

o Processor: Configured with 2 cores and 2 logical processors from the host's 11th Gen Intel® Core™ i7-1165G7 processor.

o RAM: 4GB.

o Memory (Disk Space): 100GB.

Essential software versions installed on VM:

• Hadoop: 3.3.6.

• Spark: 3.4.2.

• MySQL: 8.0.36.

• MySQL Workbench: 8.0.36.

• Jupyter Notebook: 6.4.8.

The dataset loaded into Hadoop and MySQL was sourced from the Datablist website (www.datablist.com, n.d.). It comprises nine columns and two hundred rows, containing personal data such as names, surnames, gender, job positions, etc. A significant consideration with this type of data is privacy; however, the data was randomly generated by the Python Faker package, as seen on the Datablist GitHub account (GitHub, 2023). This approach ensures GDPR compliance, carefully avoiding any conflict with it. Importantly, this dataset does not require a license for use, as this data is dummy generated for testing purposes.

At first people.csv weighted 0.23 GB however for testing purposes it was increased to 1.6 GB using 1. Increasing\_dataset\_size.ipynb script. The dataset was duplicated seven times, resulting in people\_increased.csv. This choice was made with the intention of approaching Big Data. Although 1.6 GB is not near what Big Data looks like nowadays, it is close to concept in terms of overwhelming most conventional applications. For instance, Excel CSV grid will crash automatically when attempting to open this file. Yes, we can use Notepad ++ to how the data looks but no more than that.

### *Hadoop people\_increased.csv load: First step creating a new directory and moving the dataset into it:*

After a successful load Hadoop UI shows:

Above figure shows a 1.52GB file that is replicated once with a block size of 128MB. To get a sense of how HDFS works we need to run the following command:

After running it, we see the file divided and stored across 13 blocks:

This means the HDFS has filled 12 blocks completely 128MB (134,217,728 bytes) in size, consistent with HDFS's default block size setting except the last one as is the remainder, with 20.24MB (21,234,682 bytes). This is common as the final block not using the default block size unless is a multiple of that block size.

1. MySQL people\_increased.csv load: Before loading the file an schema and a table inside must be created:

After successful schema and table creation, data will be loaded via 2.Importing\_1.6GB\_CSV\_to\_MySQL.ipynb script. This approach was chosen because MySQL Workbench server import failed, displaying the error : ”Error Code: 2013. Lost connection to MySQL server during query”. MySQL workbench appears to struggle with importing large CSV files into a schema. However, the script did with a time of 7 minutes and 28 seconds. Let us examine the size of the table:

3) HDFS vs MySQL loading process: After this implementation step, it is clear that HDFS is quicker in terms loading time. It required just two commands and took only 5 to 10 seconds for the file to be integrated in the system whereas, MySQL took 7 minutes and 28 seconds. Furthermore, MySQL required the creation of a schema and table. A second advantage of HDFS is memory comsumption; from an intial 1.63GB csv file when loaded, it was reduced to 1.52GB, whereas MySQL, upon loading, increased to 1.87GB.