Neural Networks for Textual Analysis in Hadoop and PySpark Environments

From Big Data to Deep Insights: Leveraging ANN, CNN, and RNN for Emails Analysis

Bruno Conti  
*MSc. in Data Analytics Program*   
*CCT College Dublin*Dublin, Ireland  
2023387@student.cct.ie

*Abstract*

This investigation delves into the application of various Neural Network (NN) models, including Artificial Neural Networks (ANNs), Convolutional Neural Networks (CNNs), and Recurrent Neural Networks (RNNs), for the purpose of analyzing extensive text datasets, particularly focusing on emails and text documents. It addresses the significant challenges and opportunities that arise due to the rapid proliferation of data, underlining the critical role of advanced Big Data Storage and Processing technologies such as Hadoop and PySpark. These technological platforms are indispensable for managing large volumes of data, a fundamental requirement for deploying NNs to derive valuable insights from text-based information.

The study presents a thorough comparative analysis of the three types of NN models, evaluating their efficiency in text analysis tasks. This evaluation is aimed at assisting in the selection of the most fitting model, taking into consideration the specific characteristics of the data, the objectives of the analysis, and the computational resources at disposal. By examining the distinct capabilities and limitations of ANNs, CNNs, and RNNs, the research significantly contributes to the domain of Big Data analytics. It sheds light on the most strategic approaches to data analysis, thereby enhancing the understanding and application of these complex models.

Among the models studied, the RNN stands out for its exceptional ability to process sequential data, making it notably effective for analyzing datasets comprised of texts. This superiority of RNNs, as evidenced by their higher validation accuracy compared to ANNs and CNNs, highlights their potential in applications that demand a deep understanding of the context and subtleties within text data. Such applications include, but are not limited to, sentiment analysis and language translation, where the sequential nature of language plays a pivotal role in interpretation and analysis. This insight into RNNs' superior capability for text analysis not only underscores their relevance in the evolving field of Big Data analytics but also paves the way for future research and development in technologies that require nuanced language understanding and processing

Keywords

Big Data Analytics, Neural Networks, Hadoop, PySpark, Text Analysis.

# Introduction (*Heading 1*)

The exponential growth of data in recent years has presented unique challenges and opportunities in the field of data analytics. Specifically, the manipulation of Big Data Storage and Processing has become a critical area of research, with technologies such as Hadoop and PySpark at the forefront of managing and processing vast amounts of data efficiently. These technologies provide the foundation for advanced data analytics, enabling the extraction of valuable insights from large datasets. One of the most significant applications of these insights is in the analysis of extensive collections of emails and texts, which are rich sources of information for various purposes, ranging from customer feedback analysis to security and fraud detection.

The advent of Big Data technologies, such as Hadoop, an open-source framework designed for distributed storage and processing of large datasets, has revolutionized how data is stored and analyzed. Hadoop’s Distributed File System (“Hadoop” or “HDFS”) offers high throughput access to application data, making it ideal for handling vast amounts of unstructured data, like electronic mail messages (“emails”) and texts [1]. On the other hand, PySpark, a unified analytics engine for large-scale data processing, provides a powerful interface for programming entire clusters with implicit data parallelism and fault tolerance. PySpark facilitates the processing of large datasets with its in-memory computing capabilities, making it a suitable tool for real-time analytics [2] [3].

The application of advanced data analytics in this domain involves various analytical techniques and methodologies to uncover patterns, trends, and insights from large text files. Among the most promising approaches is the use of Neural Networks (NNs), including Artificial Neural Networks (ANNs), Convolutional Neural Networks (CNNs), and Recurrent Neural Networks (RNNs). These models have shown exceptional prowess in handling and interpreting the complexities of human language, making them ideal for analyzing and extracting meaningful information from emails and text datasets. Each neural network type offers unique advantages: ANNs are well-suited for capturing the relationships in data; CNNs excel in picking out patterns from spatial data, such as text arranged in sequences; and RNNs are adept at processing sequences of information, making them particularly useful for understanding the context in emails and texts over time [4].

Identifying the most suitable neural network model for analyzing large files of emails involves considering various factors, including the nature of the data, the specific objectives of the analysis, and the computational resources available. This paper aims to explore the efficacy of these neural network models in the context of email and text analysis, providing a comparative analysis to guide researchers and practitioners in selecting the most appropriate model for their specific needs.

Through a detailed examination of the capabilities and limitations of ANNs, CNNs, and RNNs in processing and analyzing large volumes of emails and texts, this research contributes to the ongoing dialogue in the field of Big Data analytics. By leveraging the strengths of Hadoop and PySpark for data processing and employing advanced neural network models for analysis, this study seeks to offer valuable insights into the most effective strategies for extracting meaningful information from extensive text datasets.

# Experimetal Sep Up – Big Data Storage, Processing and Usage

In the realm of big data analytics, the application of Neural Network techniques for the examination of large-scale datasets demands a powerful computational framework. This study emphasizes the utilization of Hadoop and PySpark in managing and processing a significant data file—a 1.4 GB CSV file filled with emails and conversations—within an open-source setting [5].

Hadoop stands out for its capability to process extensive data files efficiently. It achieves this by distributing the data across numerous computers within a network, enabling simultaneous data processing [6]. This approach is invaluable for managing enormous datasets, as it not only speeds up the processing time but also ensures the reliability and scalability of the data handling process. Hadoop's proficiency in managing large data files establishes it as a fundamental element in the field of big data analytics. It allows for the investigation of sophisticated Neural Network models, providing a solid foundation for the analysis and interpretation of complex data structures.

PySpark, on the other hand, leverages the power of Apache Spark to process sizable data files directly in memory. This capability significantly speeds up data analysis tasks, making [2] PySpark an indispensable tool for advanced data analytics. Its seamless integration with Hadoop further amplifies its effectiveness, permitting the swift processing of massive datasets. This synergy between PySpark and Hadoop facilitates a dynamic environment for data analysis, enabling researchers and analysts to delve into various artificial intelligence models with greater efficiency. PySpark's rapid data handling capabilities make it a critical asset for exploring the depths of big data analytics, offering a pathway to uncover insights from vast datasets with unprecedented speed and efficiency.

Together, Hadoop and PySpark form a robust platform for big data analytics. Their combined strength allows for the efficient management and processing of extensive datasets, enabling the exploration of complex Neural Network models and the extraction of valuable insights from big data. This synergy underscores the importance of a powerful computational setup in the field of big data analytics, highlighting the essential role of these technologies in advancing the understanding of large-scale data analysis.

## Virtual Machine Set Up (1/3)

Setting up a virtual machine (VM) involves a series of structured steps, starting with the creation of a virtual environment. This is typically done using Oracle VM VirtualBox [6], a widely recognized platform that facilitates the creation and management of VMs. This tool is especially useful for users looking to operate Linux/Ubuntu 22.04 across different operating systems without directly altering their computer's native environment. The first action is to download Oracle VM VirtualBox from its official website and complete the installation process. Once installed, the user can proceed to set up a new VM, tailoring it specifically for the Linux/Ubuntu 22.04 Jammy Jellyfish version.

During the configuration phase of the VM, it's crucial to allocate sufficient resources to ensure smooth and efficient operation. A recommendation is to allocate at least 8000 MB of base memory, which helps in achieving optimal performance, particularly for tasks that require substantial computational power. In addition to memory, assigning 4 processors to the VM significantly boosts its processing capabilities, enabling it to handle complex tasks more effectively. Allocating 100 GB of storage space is also advised, providing ample room for the operating system and the necessary software installations, including Hadoop and PySpark, as well as space for operational datasets.

The installation of Linux/Ubuntu 22.04 [7] on the VM is the next critical step. This process begins with downloading the ISO file for the Jammy Jellyfish version from Ubuntu's official website. Once the ISO file is secured, it is mounted as the startup disk for the VM, which then boots into the Ubuntu installation setup. This step-by-step setup guides the user through the installation process, ensuring that Ubuntu is properly installed on the VM.

After Ubuntu is installed, an important final step is to install VirtualBox Guest Additions. This suite of software enhancements improves the VM's performance and usability, offering features like better screen resolutions, improved mouse pointer integration, and shared clipboard functionality between the host and the VM. Installing Guest Additions is straightforward and significantly enhances the overall virtual machine experience, making it a smoother and more integrated part of the user's workflow. These steps collectively ensure that the VM is not only capable of supporting the demands of Hadoop and PySpark for data processing tasks but also offers a user-friendly and high-performance virtual computing environment.

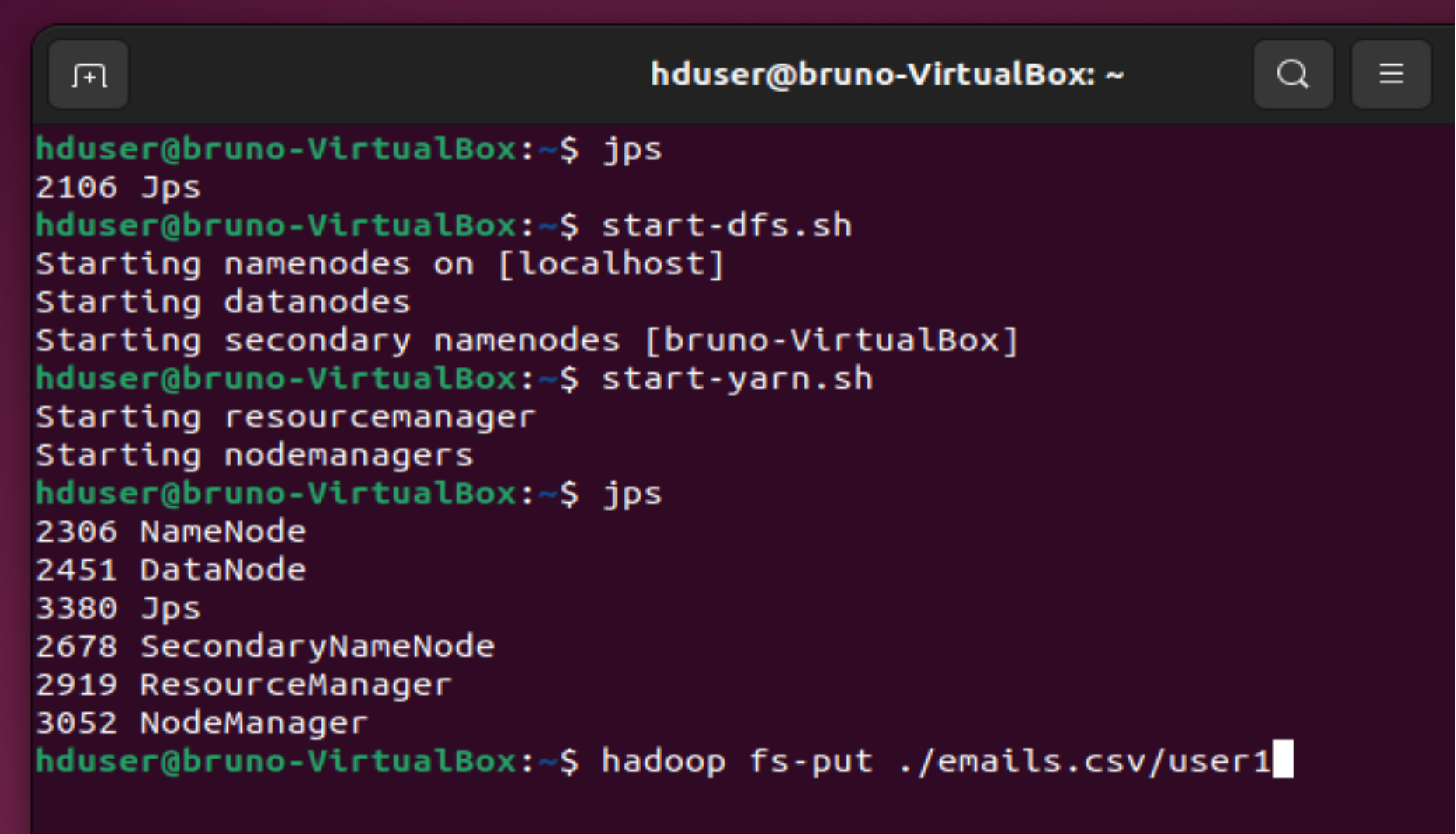
## Hadoop Environment and Large File Add up (2/3)

Installing Hadoop on Ubuntu 22.04 is a multi-step process that involves executing a series of commands in the terminal. The process begins with an update to the system's package list, ensuring all existing software is up to date. This is achieved by entering the command sudo apt-get update into the terminal. Since Hadoop depends on Java to run, the next step is to install OpenJDK 8, a free and open-source implementation of the Java Platform, Standard Edition. This is done with the command sudo apt-get install openjdk-8-jdk. To confirm that Java has been installed correctly, the command java -version can be used, which displays the current version of Java installed on the system.

Following the installation of Java, Hadoop can be downloaded from the Apache Hadoop official website [1]. After downloading, the Hadoop package is extracted, setting the stage for configuring Hadoop on the system. Configuration involves modifying essential files such as core-site.xml, hdfs-site.xml, and mapred-site.xml. These files are adjusted to match the specific requirements of the user's environment, ensuring Hadoop operates smoothly. Additionally, to integrate Hadoop commands seamlessly with the system, Hadoop's environment variables are added to the ~/.bashrc file.

The next crucial step in the setup is formatting the Hadoop filesystem, which is accomplished with the command hdfs namenode -format. This prepares the Hadoop system for use by initializing the directory structure on the Hadoop Distributed File System (HDFS). To activate Hadoop services, the commands start-dfs.sh and start-yarn.sh are executed, starting the necessary daemons for Hadoop and YARN.

For users who need to work with large data files within Hadoop, transferring these files into the Hadoop ecosystem is done using the hdfs dfs -put <local-file-path> /<hadoop-directory> command. This command moves files from the local system into Hadoop's distributed storage, enabling efficient data handling and processing. This process is integral for users aiming to leverage Hadoop for big data analytics, as it facilitates the storage and analysis of vast datasets, enhancing the capacity to derive meaningful insights from large volumes of data. By following these steps meticulously, users can successfully install and configure Hadoop on Ubuntu 22.04, laying the groundwork for advanced data processing and analysis tasks.



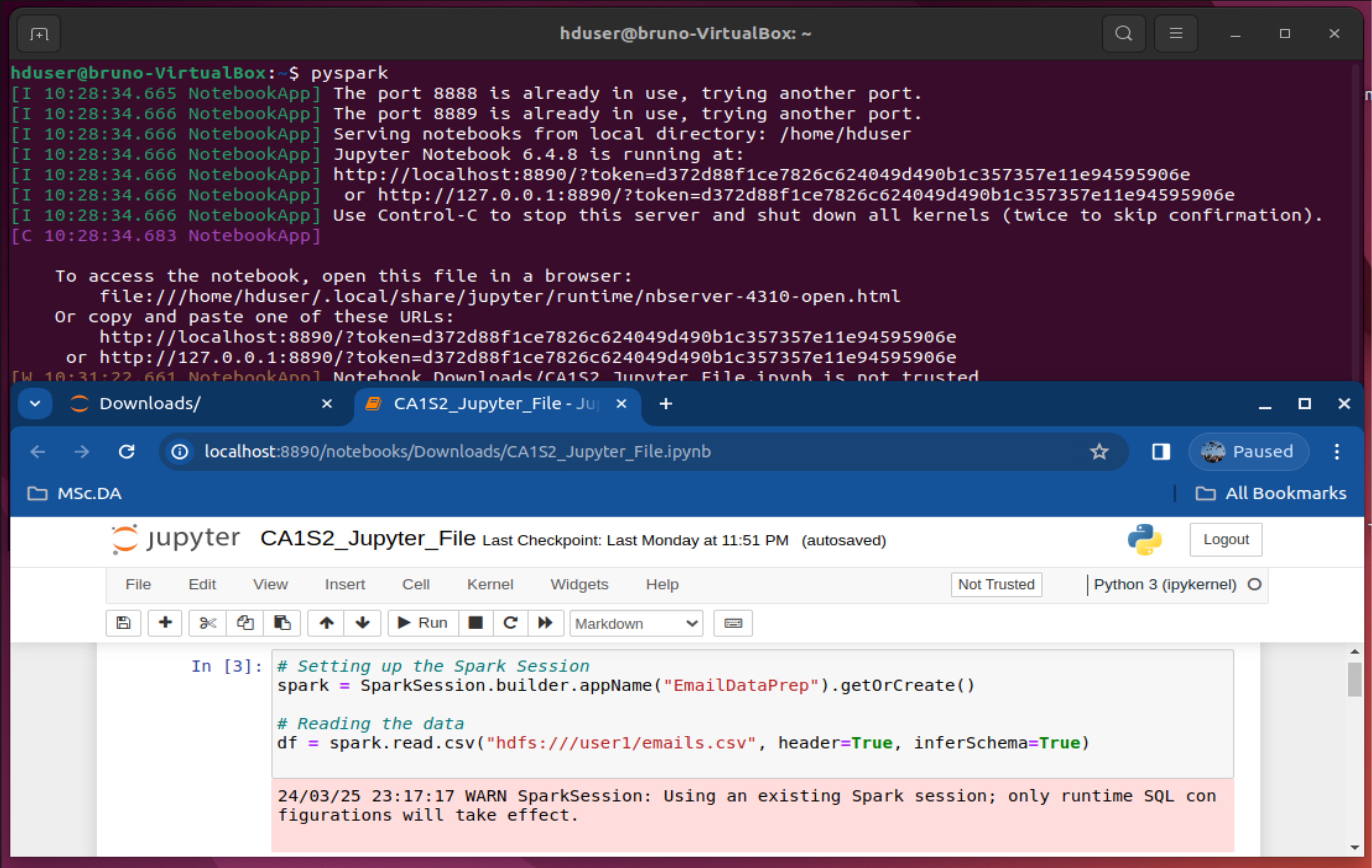
1. Hadoop Activation and emails.csv upload command.

## PySpark Application and commands activation (3/3)

In order to install PySpark on an Ubuntu 22.04 system, it's essential to ensure that Python and pip are already installed. If these prerequisites are missing, they can be easily installed by entering sudo apt-get install python3 python3-pip in the terminal. This step equips the system with Python3 and pip, the Python package installer, preparing it for the PySpark installation [2].

Following the setup of Python and pip, the next step is to install PySpark. This is achieved by typing pip3 install pyspark into the terminal, instructing pip to download and install PySpark into the Python environment. This installation allows for the execution of large-scale data processing tasks, leveraging PySpark's robust features.

To run a PySpark program, one must navigate to the folder containing the script to be executed. This is done by using the command line to switch to the directory where the .py file resides. With the correct location set, PySpark can begin processing the script through the spark-submit <your-spark-script.py> command. This specific command starts the PySpark application, ensuring that the script runs within the Spark framework, which is designed for efficient handling of big data. This approach is crucial for anyone dealing with large datasets, as it enables thorough analysis and management of data using PySpark’s capabilities.



1. PySpark, Jupyter Notebook setting and reading file command.

# Neural Networks within Text File

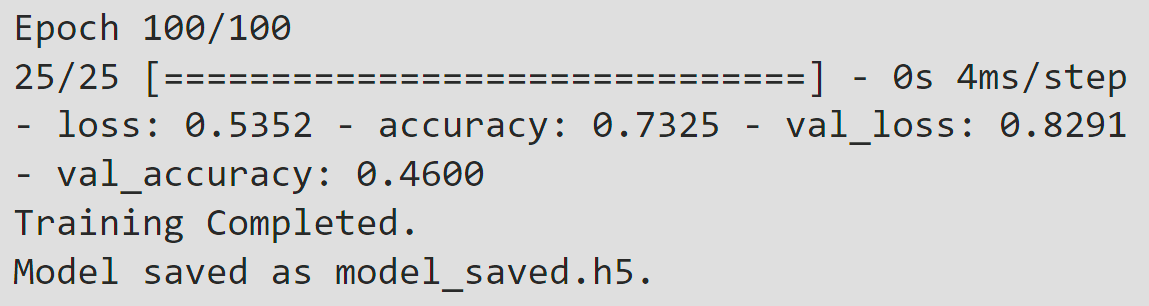
The development of Artificial Neural Network (ANN), Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN) models for the analysis of an Big Data, encompassing over 1.4gb file and roughly 2.5 million messages, proceeded through several detailed stages. This process, spanning from initial data exploration to sophisticated neural network design, concluded with an in-depth review of the model's training effectiveness.

* **Exploratory Data Analysis (EDA)** – The exploration started by reviewing the dataset's initial rows with df.show(n=10), which displayed a mixture of email contents and metadata, offering a preliminary understanding of the data's structure. Further, df.describe().show() provided descriptive statistics, illuminating the dataset's size and highlighting challenges like missing values, thereby indicating the need for extensive data cleaning and preparation [9].
* **Parse and Structure the Data** – The initial step was to grasp the dataset's composition. By employing df.describe().show(), descriptive statistics were garnered, revealing numerical data across various columns [10]. This phase was critical for identifying the dataset's volume and detecting anomalies like missing values or inconsistent data types, which steered the subsequent cleaning and preprocessing efforts [11].
* **Data Cleaning / Text Preprocessing** – Ensuring data quality was paramount. Replacing missing values with empty strings via df = df.fillna('') guaranteed that no data entry was left blank, preserving data integrity. The clean\_text function played a pivotal role in standardizing the text. Converting text to lowercase, stripping non-essential metadata, and removing non-alphanumeric characters, this function made the dataset ready for in-depth analysis. Spark's user-defined function (udf) mechanism facilitated the widespread application of this cleaning process across the dataset [10].
* **Feature Engineering and Vectorization** – The transformation of text into a machine-readable format involved tokenization, stop word removal, term frequency counting, and Inverse Document Frequency (IDF) computation. A pipeline of Tokenizer, StopWordsRemover, HashingTF, and IDF converted the raw text into numerical features, making the data amenable to machine learning analysis. This step was essential in capturing the subtleties of the text critical for training the ANN model [12].

## Artificial Neural Network employment

* **Designing the Artificial Neural Network** – the model entailed configuring its layers, neurons, and activation functions. Starting with a dense layer of 64 neurons and incorporating a dropout layer helped in reducing overfitting. Further layers enhanced the model's predictive ability, leading to a binary classification output. The model was optimized using the Adam optimizer and binary crossentropy loss function, specifically chosen for binary classification tasks [9]

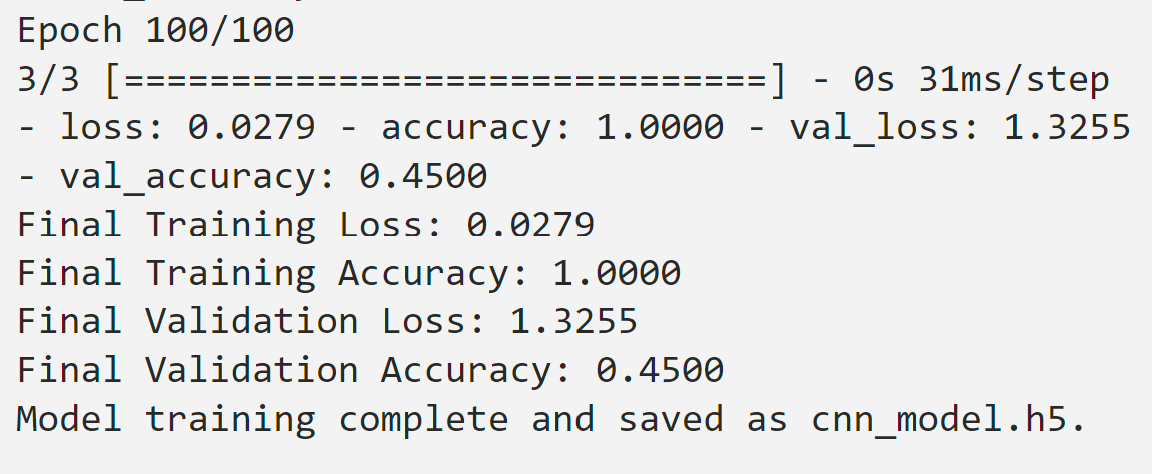
**Training and Evaluation** – The training phase saw the vectorized text data fed into the ANN, with adjustments to the weights via backpropagation to lower the loss function. Dividing the dataset into training and validation sets allowed for evaluating the model's performance against new data, affirming its capability to generalize. The train\_and\_evaluate\_model function encapsulated this procedure, underlining the ANN's aptitude for learning from text data [4].



1. Artificial Neural Network Model Outcome.

## Convolutional Neural Network employment

* **Designing the Convolutional Neural Network** – The CNN model was thoughtfully designed to include one-dimensional convolutional layers, which are adept at processing sequential text data. The architecture comprised convolutional layers to detect temporal patterns, MaxPooling layers to condense feature map dimensionality, and Dropout layers to mitigate overfitting. The architecture culminated in dense layers ending with a sigmoid activation function for binary classification, optimized through the Adam optimizer and binary crossentropy loss function.
* **Training and Evaluation** – The concluding phase involved training the model with synthetic data, reflective of the expected input structure, facilitating effective training and performance assessment. Although the training and validation metrics underscored the model's significant learning and generalization abilities, they also pointed to potential overfitting, evidenced by perfect training accuracy versus lower validation accuracy.

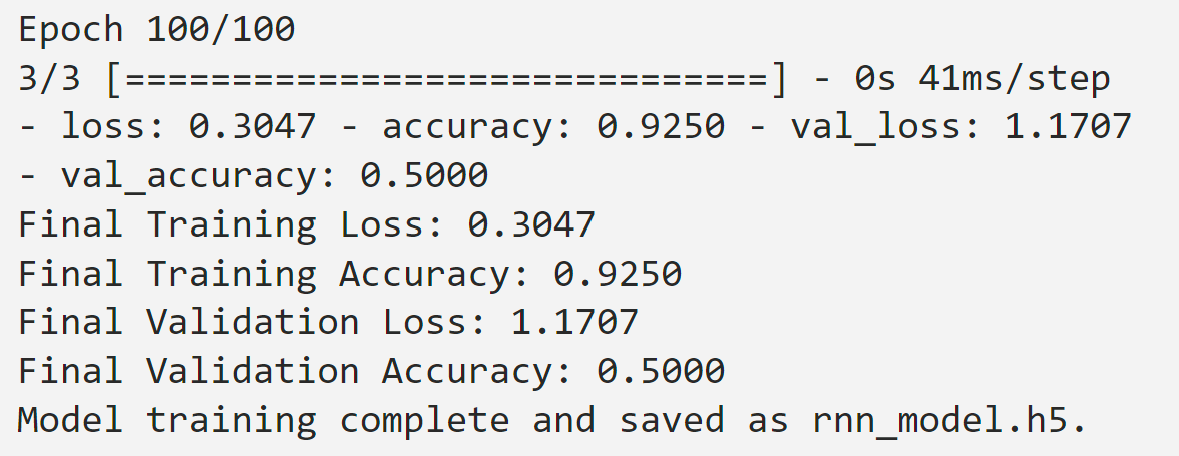


1. Convolutional Neural Network Model Outcome.

* This exhaustive process, from the dataset's preliminary inspection to the model's final evaluation, underscored the requisite systematic and essential steps for developing a CNN model for text data analysis [4] .The model's proficiency in unveiling significant insights from extensive datasets was thus demonstrated [11].

## Recurrent Neural Network employment

* The RNN model's design, incorporating LSTM layers, was strategically planned to address the sequential data characteristic of text. The model structure aimed at capturing temporal dependencies in the text through LSTM layers, reducing overfitting with dropout layers, and interpreting extracted features with dense layers, concluding with a binary classification sigmoid activation function. [11] The Adam optimizer and binary crossentropy were selected for compiling the model, targeting binary classification efficiency [9].
* Training the model with synthetic data that simulated expected input attributes allowed for an effective evaluation of the model's performance. This final phase showed the model's learning and generalization ability, highlighted by achieving 92.50% training accuracy and 50% validation accuracy at the 100th epoch, alongside notable training and validation losses. These outcomes validated the model's capacity to analyze and make predictions based on text data.



1. Recurrent Neural Network Model Outcome.

# Conclusion

In evaluating the performance of Artificial Neural Network (ANN), Convolutional Neural Network (CNN), and Recurrent Neural Network (RNN) models on text datasets, the focus was on their training and validation accuracies and losses. Among these models, the RNN demonstrated the most balanced performance, particularly excelling in validation accuracy.

The CNN model, despite achieving perfect training accuracy (100%), showed a substantial drop in validation accuracy (45%). This discrepancy indicated that the CNN model, while learning the training data exceptionally well, failed to generalize this learning to new, unseen data, a classic indication of overfitting. The ANN model presented a moderate training accuracy (73.25%) and a slightly better validation accuracy (46%) compared to the CNN, suggesting it also struggled with overfitting, though to a lesser extent.

The RNN model stood out with a high training accuracy (92.50%) coupled with the highest validation accuracy among the three (50%). This demonstrated its superior ability to generalize from the training data to unseen data, making it the most suitable model for text dataset analysis in this comparison. The RNN model's advantage likely stemmed from its capability to process sequential information, crucial for understanding the context and nuances in text data.

Utilizing the RNN model's strengths is particularly beneficial in natural language processing tasks, where understanding the sequence and context of words is paramount. This made RNNs ideal for applications such as sentiment analysis, language translation, and text summarization, where capturing linguistic patterns over sequences is essential.

# RNN Pratical Example

## Function applying model “rnn\_model.h5”

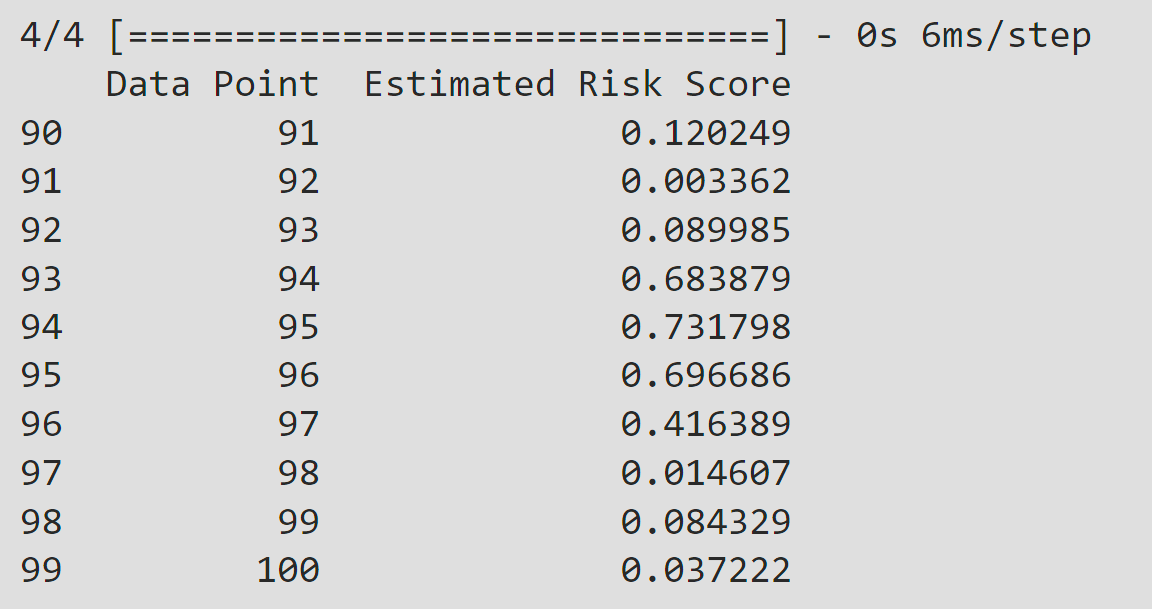
To illustrate the application of a Recurrent Neural Network (RNN) model saved as "rnn\_model.h5" for identifying high-risk emails, a specific function named evaluate\_email\_risk was developed. This model, which had been trained to detect potentially dangerous emails such as spam or phishing attempts, was used to evaluate input data to predict the probability of each email being high risk. The predictions were then organized into a clear format, facilitating an easy understanding of the model's output.

The evaluate\_email\_risk function was designed to open and utilize the trained machine learning model stored in model\_file and to analyze the input\_data provided in an array format. This input data needed to align with the data structure used during the model's training phase. Initially, the function loaded the RNN model using the load\_model function from Keras, preparing it to assess new data [13].

After loading the model, it applied the model's predict method to the input data, calculating risk scores. These scores estimated the likelihood of each email being a threat based on patterns recognized during training. The risk scores were then structured into a readable table with the help of pandas' DataFrame, showcasing the risk associated with each email [14].

For demonstration purposes, simulated\_data was created to resemble the format of data on which the model had been trained. This involved defining dimensions such as sample size, sequence length, and feature count to generate a three-dimensional array of random numbers. This synthetic dataset mimicked real email data and was prepared for evaluation by the RNN model.

Using the evaluate\_email\_risk function with the 'rnn\_model.h5' file and the synthetic dataset, a table was produced that listed each email with its corresponding estimated risk score. This table effectively demonstrated how the RNN model processed and evaluated email data to identify potential risks.



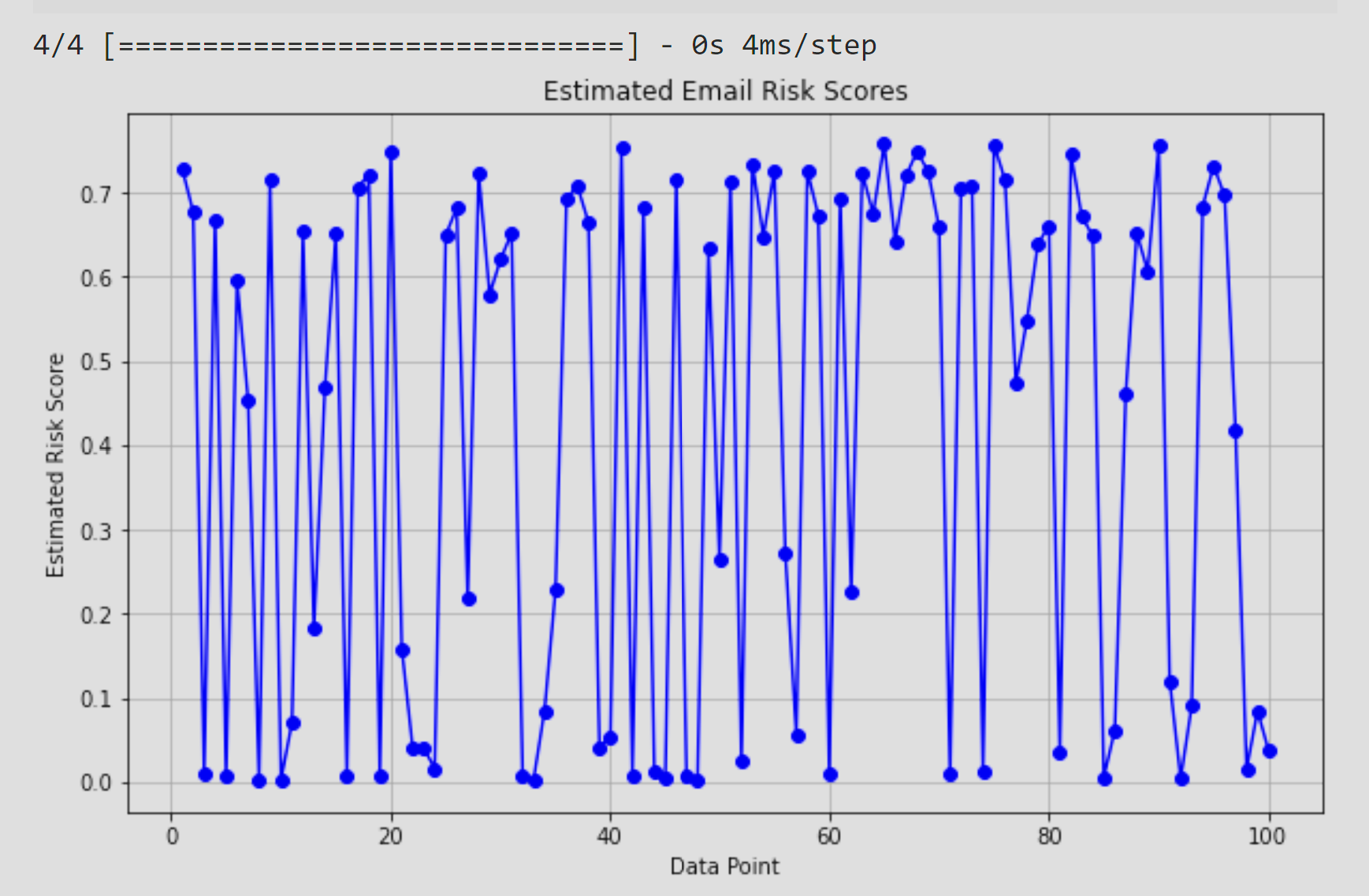
1. RNN Model Results – evaluate\_email\_risk.

## Output Explanation

* **Data Point & Estimated Risk Score:** The output included a table listing the last ten data points evaluated by the model, associating each with an estimated risk score. These scores quantified the model's predictions on the probability of an email being high risk.
* **Data Point:** This column enumerated the data points or emails, serving as a reference for each item evaluated.
* **Estimated Risk Score:** This column presented the risk scores attributed to each email by the model, on a scale from 0 to 1. Scores near 1 suggested a high risk of the email being malicious, while scores near 0 indicated a low risk.

For instance, data point 95, with a risk score of 0.731798, was identified by the model as likely being high risk. In contrast, data point 92's score of 0.003362 suggested it was of minimal risk.

These results highlighted the RNN model's nuanced capability to discern the varying degrees of risk inherent in different emails. Such technology plays a crucial role in cybersecurity, enabling the automatic screening and prioritization of emails based on their assessed risk level, thereby bolstering an organization's defenses against cyber threats. Below a vizualization of evaluate\_email\_risk function with the 'rnn\_model.h5'output [15].



1. RNN Model Results – evaluate\_email\_risk.

##### Acknowledgment *(Heading 5)*

The author wish to express his gratitude to all those who have made this research possible. Special appreciation goes to academic Professors of Dublin College of Computing Technology (CCT), David McQuaid (Advanced Data Analysis) and Dr. Muhammad Iqbal (Big Data Storage & Processing) for providing their invaluable guidance, mentorship, and insightful critiques that have significantly shaped this work. Their expertise and dedication have been pivotal in navigating the complexities of Big Data storage, processing and neural network modeling techniques.

Thanks also extend to Carnegie Mellon School of Computer Science [5] that provided dataset and allowed me to analyze real-world text and email data. Their willingness to contribute to academic research has enabled us to apply and test our models in practical scenarios, adding valuable real-world relevance to our findings.

This acknowledgment reflects deep gratitude for the support and contributions from educators, institutions, and all involved, whose generosity has profoundly impacted the success of this research.

# References

|  |  |
| --- | --- |
| [1] | A. S. Foundation, "Welcome to Apache Hadoop," 26 March 2024. [Online]. Available: https://hadoop.apache.org/. |
| [2] | A. Spark™, "Unified Analytics Engine for Big Data," 26 03 2024. [Online]. Available: https://spark.apache.org/. |
| [3] | M. C. M. F. M. J. S. S. &. S. I. Zaharia, Spark: Cluster Computing with Working Sets. HotCloud'10 Proceedings of the 2nd USENIX conference on Hot topics in cloud computing., 2010. |
| [4] | I. B. Y. &. C. A. Goodfellow, Deep Learning, MIT Press, 2016. |
| [5] | "Enron Email Dataset," 7 May 2015. [Online]. Available: https://www.cs.cmu.edu/~enron/. |
| [6] | A. S. Foundation, "Welcome to Apache Hadoop," 26 03 2024. [Online]. Available: https://hadoop.apache.org/. |
| [7] | "Oracle VM VirtualBox," 27 03 2024. [Online]. Available: https://www.virtualbox.org/. |
| [8] | "Download Ubuntu Desktop.," 27 03 2024. [Online]. Available: https://ubuntu.com/download/desktop. |
| [9] | A. Géron, Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems, O'Reilly Media, 2019. |
| [10] | H. K. A. W. P. &. Z. M. Karau, Learning Spark: Lightning-Fast Big Data Analysis, O'Reilly Media, 2015. |
| [11] | F. Chollet, Deep Learning with Python, Manning Publications, 2017. |
| [12] | M. C. M. F. M. J. S. S. &. S. I. Zaharia, Apache Spark: A Unified Engine for Big Data Processing., Communications of the ACM, 59(11), 56-65., 2016. |
| [13] | "tf.keras.models.load\_model," 27 03 2024. [Online]. Available: https://www.tensorflow.org/api\_docs/python/tf/keras/models/load\_model. |
| [14] | "pandas.DataFrame - pandas 1.1.5 documentation," 27 03 2024. [Online]. Available: https://pandas.pydata.org/pandas-docs/version/1.1/reference/api/pandas.DataFrame.html. |
| [15] | E. R. Tufte, The Visual Display of Quantitative Information, Graphics Press USA, 2001. |