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 study explores the utilization of Neural Network (NN) models, specifically Artificial Neural Networks (ANNs), Convolutional Neural Networks (CNNs), and Recurrent Neural Networks (RNNs), in analyzing large text datasets, with a focus on emails and texts. The research highlights the challenges and opportunities presented by the exponential growth of data, emphasizing the importance of efficient Big Data Storage and Processing technologies like Hadoop and PySpark. These platforms enable the handling of vast datasets, which is crucial for the application of NNs in extracting meaningful insights from text-based data. The paper conducts a comparative analysis of ANNs, CNNs, and RNNs, assessing their effectiveness in text analysis to guide the selection of the most appropriate model based on the data's nature, analysis objectives, and available computational resources. Through a detailed exploration of each model's capabilities and limitations, the research contributes to the field of Big Data analytics, offering insights into the most effective strategies for data analysis. The RNN model, noted for its ability to process sequential information, emerges as particularly suitable for text dataset analysis, outperforming ANNs and CNNs in terms of validation accuracy. This finding underscores the potential of RNNs in applications requiring an understanding of context and nuances in text data, such as sentiment analysis and language translation.

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 sphere of big data analytics, employing Neural Network techniques to analyze extensive datasets necessitates a robust computational setup. This research also aims to highlights the use of Hadoop and PySpark for managing and processing a substantial data file - a 1.4 GB CSV [5] containing emails and conversations, within an open-source environment.

Hadoop efficiently processes big data files by distributing them across multiple computers in a network, allowing for parallel data processing. This method is crucial for handling vast datasets as it enhances processing speed and ensures data reliability and scalability. Hadoop's ability to manage sizeable data files makes it a cornerstone for big data analytics, facilitating the exploration of complex Neural Network models [1].

PySpark controls the power of Apache Spark to process large data files in memory, which significantly accelerates data analysis tasks. Its compatibility with Hadoop enhances its utility, enabling it to process massive datasets quickly. PySpark is essential for conducting advanced data analytics and exploring various artificial intelligence models due to its efficiency and speed in handling big data [2].

## Virtual Machine Set Up (1/3)

Firstly, the initial step involves setting up a virtual environment using Oracle VM VirtualBox [6]. This platform allows for the creation and management of virtual machines (VMs), enabling users to run Linux/Ubuntu 22.04 on various operating systems. To begin, Oracle VM VirtualBox is downloaded and installed from the official website. Following installation, a new VM is created specifically for Linux/Ubuntu 22.04 Jammy Jellyfish.

During the VM configuration phase, it is recommended to allocate at least 4000 MB of base memory to ensure optimal performance. Additionally, assigning 2 processors and provisioning 100 GB of storage space will accommodate the operating system along with the Hadoop and PySpark installations and their operational datasets.

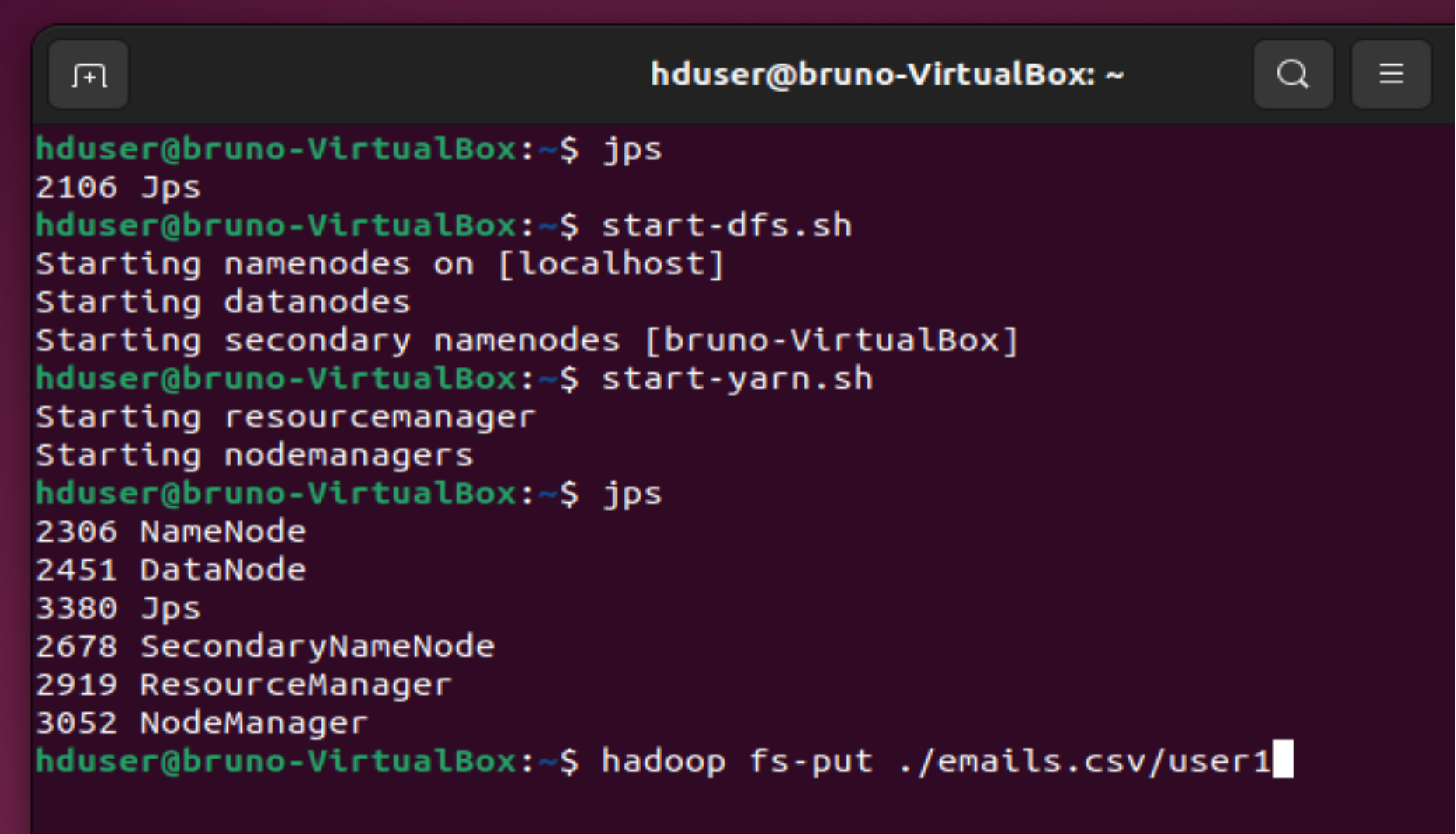
Once the VM is configured, the next step involves installing Linux/Ubuntu 22.04. This process starts with downloading the ISO file for Jammy Jellyfish from Ubuntu's official site [7]. After mounting the ISO file as the startup disk, the VM boots into the installation setup, guiding users through the installation process. Post-installation, installing VirtualBox Guest Additions enhances the VM's performance and usability.

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

The installation of Hadoop on Ubuntu 22.04 necessitates several terminal commands. Initially, the system's package list is updated using sudo apt-get update. Hadoop requires Java; thus, OpenJDK 8 is installed with sudo apt-get install openjdk-8-jdk. Verification of Java installation is done via java -version.

Hadoop is then downloaded from the Apache Hadoop official site and extracted [8]. Essential configuration files such as core-site.xml, hdfs-site.xml, and mapred-site.xml are modified to reflect the specific environment setup. Hadoop's environment variables are added to the ~/.bashrc file, ensuring the system recognizes Hadoop commands.

To format the Hadoop filesystem, the command hdfs namenode -format is executed. Starting Hadoop services requires start-dfs.sh and start-yarn.sh. For inserting large files into Hadoop, the command follows the pattern hdfs dfs -put <local-file-path> /<hadoop-directory>, facilitating data storage within the Hadoop ecosystem.

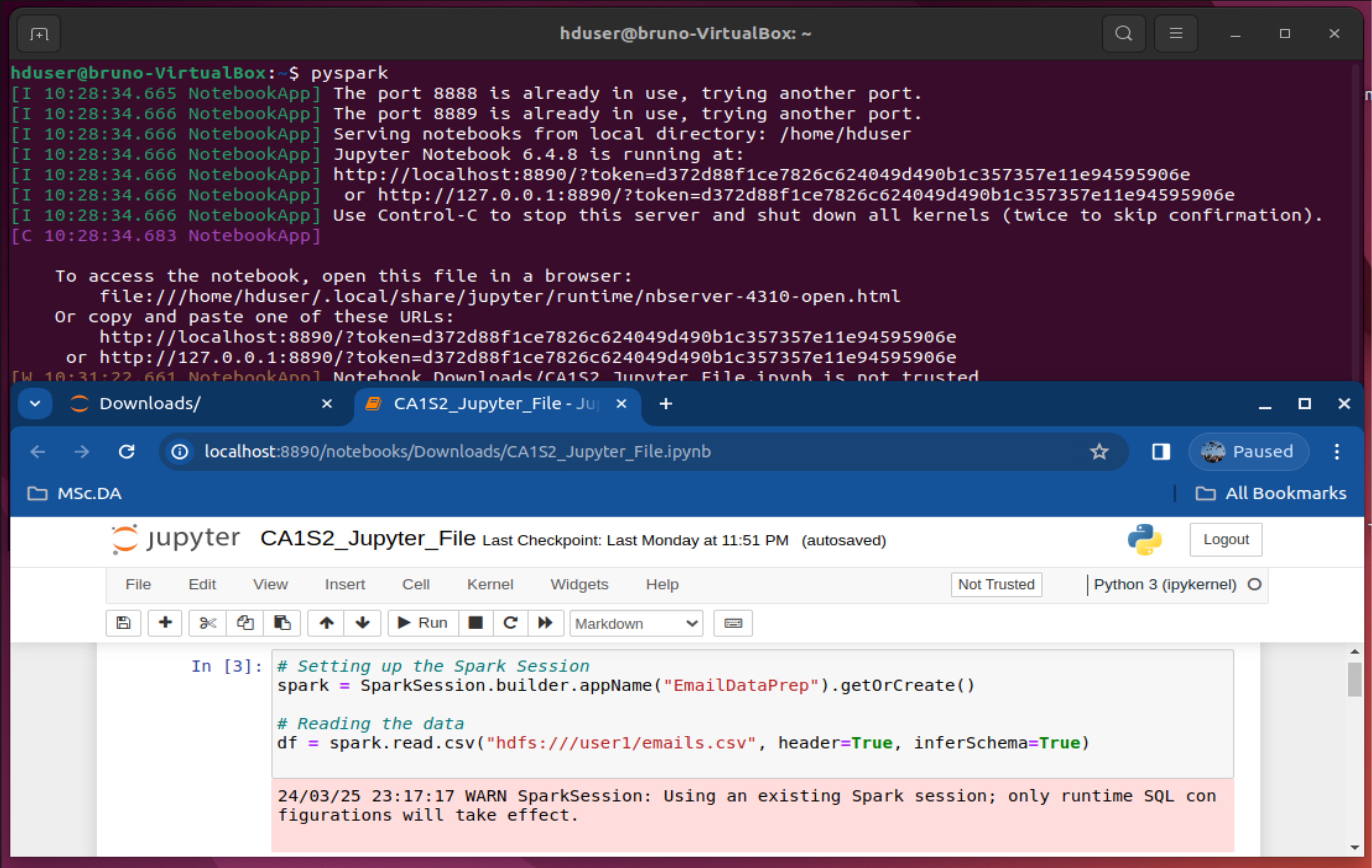


1. Hadoop Activation and emails.csv upload command.

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

PySpark installation on Ubuntu 22.04 begins with ensuring Python and pip are present. If absent, they are installed via sudo apt-get install python3 python3-pip. PySpark is then installed using pip with pip3 install pyspark [2].

Running a PySpark application involves navigating to the script's directory and executing spark-submit <your-spark-script.py>. This command initiates the PySpark application, processing the specified script.



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

# Neural Networks within Text File

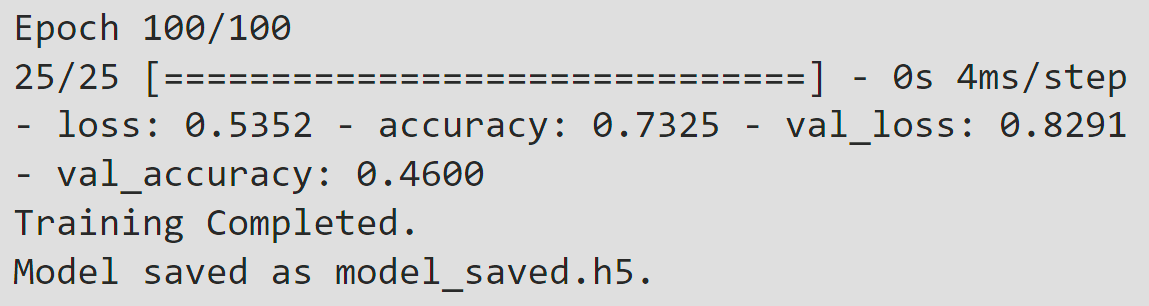
The development of Artificial Neural Network (ANN), Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN) models for the analysis of an Big Data, encompassing over 1.4gb file and roughly 2.5 million messages, proceeded through several detailed stages. This process, spanning from initial 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]

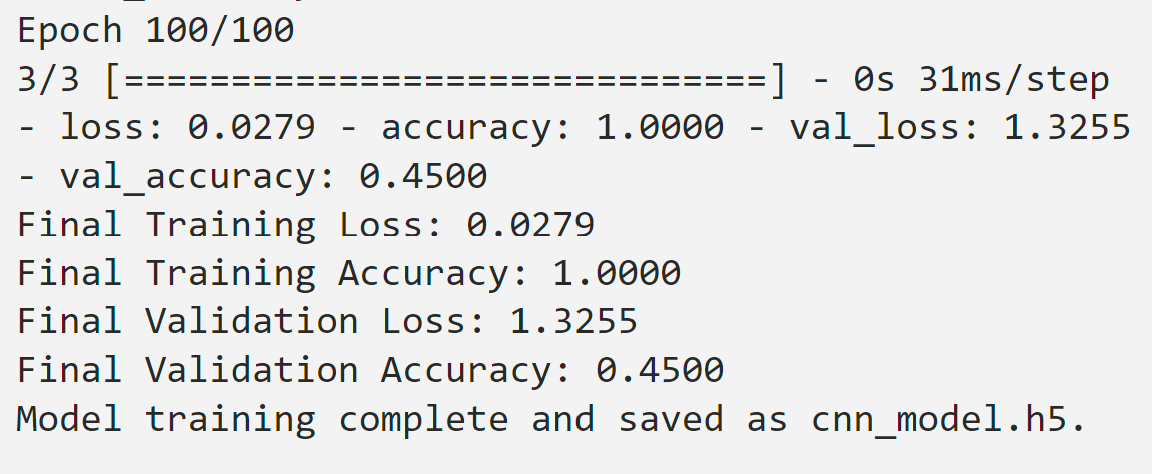
**Training and Evaluation** – The training phase saw the vectorized text data fed into the ANN, with adjustments to the weights via backpropagation to lower the loss function. Dividing the dataset into training and validation sets allowed for evaluating the model's performance against new data, affirming 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.

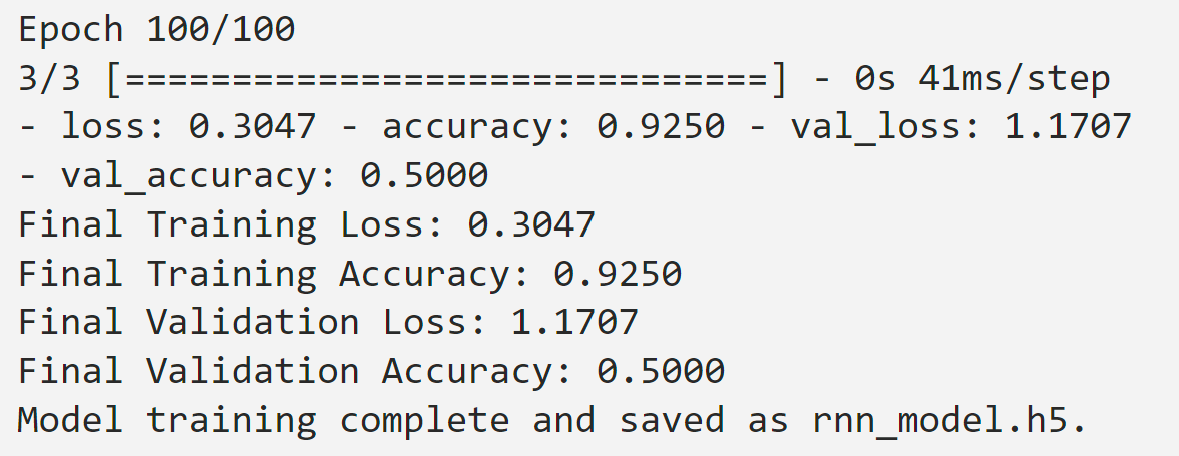


1. Convolutional Neural Network Model Outcome.

* 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 70% 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”

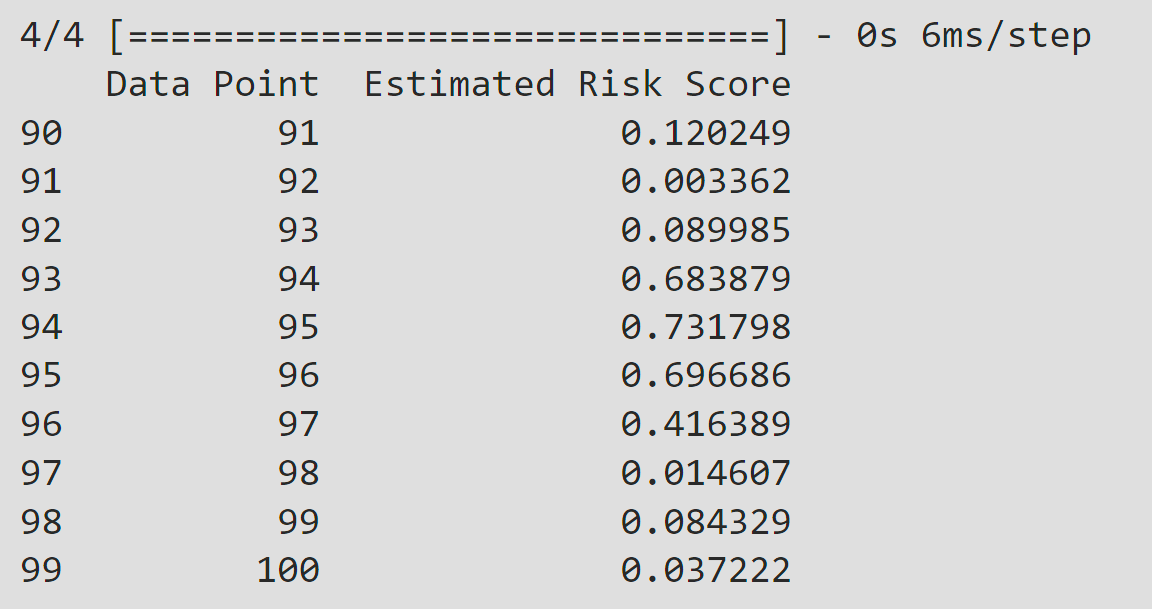
To illustrate the application of a Recurrent Neural Network (RNN) model saved as "rnn\_model.h5" for identifying high-risk emails, a specific function named evaluate\_email\_risk was developed. This model, which had been trained to detect potentially dangerous emails such as spam or phishing attempts, 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.



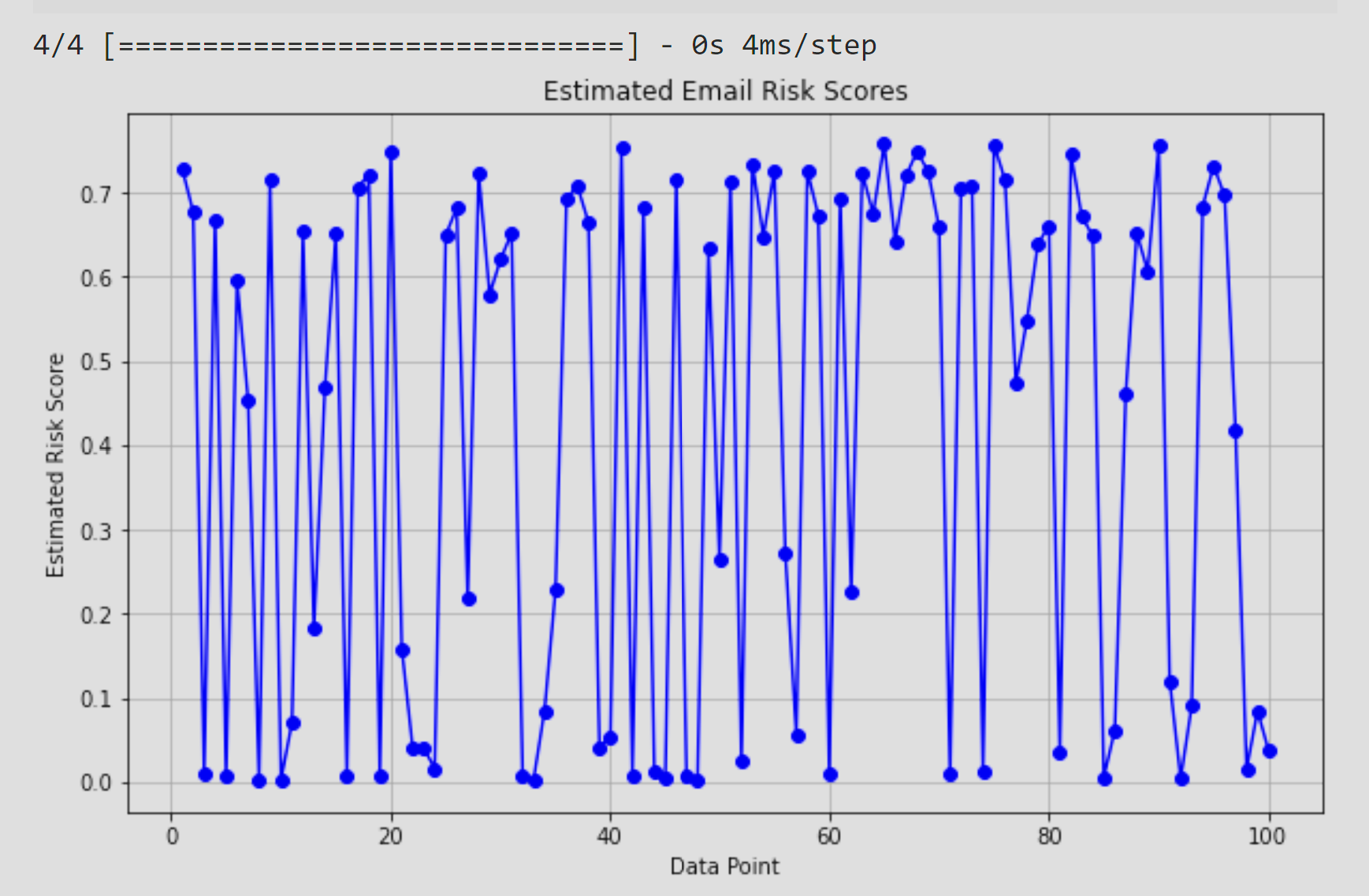
1. RNN Model Results – evaluate\_email\_risk.

## 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] | "Oracle VM VirtualBox," 27 03 2024. [Online]. Available: https://www.virtualbox.org/. |
| [7] | "Download Ubuntu Desktop.," 27 03 2024. [Online]. Available: https://ubuntu.com/download/desktop. |
| [8] | A. S. Foundation, "Welcome to Apache Hadoop," 26 03 2024. [Online]. Available: https://hadoop.apache.org/. |
| [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. |