

10922984: Using machine learning to identify the key variables in data sets

Submission date: 19-Sep-2022

Submission ID: 10922984

File name: Dissertation_10922984_FINAL.pdf

Word count: 15297

Character count: 94766

USING MACHINE LEARNING TO IDENTIFY THE KEY VARIABLES IN DATA SETS



The University of Manchester

A DISSERTATION SUBMITTED TO THE UNIVERSITY OF
MANCHESTER FOR THE DEGREE OF MSc DATA SCIENCE IN THE
FACULTY OF SOCIAL SCIENCE

YEAR:2022
STUDENT ID:10922984
SCHOOL OF SOCIAL SCIENCES

Contents

List of Tables:	5
List of Figures:	5
List of Equations:	6
ABSTRACT	7
DECLARATION	8
INTELLECTUAL PROPERTY STATEMENT	9
ACKNOWLEDGEMENTS.....	10
CHAPTER 1: INTRODUCTION	11
INTRODUCTION.....	11
1.1 Problem Definition:	13
1.2 Project Goals:	13
1.3 Proposed Methodology	14
1.4 Dissertation Structure:	14
CHAPTER 2: LITERATURE REVIEW	16
Background and Research	16
2.1 Privacy:	16
2.2 Effect of Digitization and Digitalization:	16
2.3 Report on Data Responsibility by KPMG:	17
2.4 The growing need for Privacy:	19
2.5 Privacy seen in the past and now:.....	19
2.6 Rise in data breaches and concerns to privacy:	20
2.7 Data Anonymization process:	20
2.8 Web applications and API's using JSON structure:	21
2.9 Deep diving into Text Classification techniques and implementation:	23
2.10 Automatic Anonymization of Textual Documents: Detecting Sensitive Information via Word Embeddings:	36
CHAPTER 3: DESIGN AND METHODOLOGIES	39
3.1 Project Approach and Design Aspects	39
3.1.1 Implementation Process Flow Chart:	40
3.1.2 Version information for Python libraries:	41
3.2 Libraries used:	43
3.3 Understanding the Dataset:	43
3. 4 Data Pre-processing:	45

3.5 Exploratory Data Analysis (EDA):	47
3.6 Tokenisation:	52
3.7 MODEL IMPLEMENTATION	55
3.7.1 Approach 1: Implementing Machine Learning Models	56
3.7.2 Approach 2: Implementing Neural Network based model	59
CHAPTER 5: TESTING	70
5.1 JSON Flattening:	72
5.2 Data Pre-processing:	72
5.3 Tokenization:	73
5.4 Phonetic and Spelling Match:	73
5.4.1 Phonetic Level Match:	74
5.4.2 Spelling Level Match.....	74
5.5 Word Replacement:	75
5.6 TEST CASES:.....	75
CHAPTER 5: EVALUATION AND RESULT	77
5.1 Performance indicators:	77
5.1.1 Confusion Matrix:.....	77
5.1.2 Accuracy:	78
5.1.3 Classification Report:	78
5.2 Evaluation of Machine Learning Models:	79
5.2.1 Decision Tree Classifier:.....	79
5.2.2 Random Forest Tree Classifier:	80
5.2.3 Logistic Regression Classifier:.....	81
5.3 Evaluation of Deep Neural Network using LSTM:.....	83
5.2.1 Model 1 - LSTM with solely word2vec Embedding Layer:	83
5.2.2 Model 2 - LSTM with word2vec ‘gensim’ Embedding Layer:	85
CHAPTER 5: CONCLUSION, LIMITATIONS, FUTURE WORK	88
5.1 CONCLUSION:	88
5.1.1 Achievements:	89
5.2 Limitations:	89
5.3 Future Work:	90
References	92
References	Error! Bookmark not defined.

List of Tables:

Table 1:Version information about different utilities installed	42
Table 2: TEST CASES AND RESULT	76
Table 3: Classification scores for Decision Tree	80
Table 4: Classification Table for Random Forest Tree	81
Table 5: Classification Table for Logistic Regression Classifier	82
Table 6: Classification Table for LSTM model 1	84
Table 7: Classification Table for LSTM Model 2	86
Table 8: Train and Validation scores(Model 1 vs Model 2)	87

List of Figures:

Figure 1:Anonymization Process for Personal Data	21
Figure 2:JSON Document	22
Figure 3: JSON transfer mechanism	23
Figure 4: Text classification process.....	24
Figure 5: Sigmoid function.....	29
Figure 6 :Decision Tree structure	30
Figure 7:Random Forest tree structure [20]	31
Figure 8:Neural Network structure	32
Figure 9: Recurrent neural vs Feed-Forward Neural Network.....	33
Figure 10: Word embedding model & Training and Classification	38
Figure 11: Implementation Process flow.....	40
Figure 12: The runtime configuration setup used for Jupyter notebook.....	41
Figure 13: The runtime configuration setup used for Google Collab Notebook.	41
Figure 14: Libraries used.....	43
Figure 15: Initial Dataset.....	43
Figure 16: Dataset Description	44
Figure 17:Identifiers Dataset	44
Figure 18: count of null values in the main dataset	45
Figure 19: Modified Dataframe after data pre-processing	46
Figure 20: Total count of records grouped by label.....	47
Figure 21: Word count in each record.....	48
Figure 22: Mean word length for each label.....	48
Figure 23: Frequency of occurrences vs length of the word	49
Figure 24: Top 20 most occurring words in the dataset.....	50
Figure 25: Word Cloud (the size of the key is proportional to the number of times the key has appeared in the dataset)	51
Figure 26: example sentence	54
Figure 27: Sentence converted to respective tokenized ids by BERT tokenizer	54
Figure 28: Example list of strings	55

Figure 29: Unique word with index	55
Figure 30 : BERT Tokenizer parameters	56
Figure 31:Modified Dataframe using BERT	57
Figure 32: A BSLSTM for 7-word input sequence with word embedding and hidden units .	61
Figure 33:Average error values per each activation function for the MNIST data set	66
Figure 34: Model 1 configuration	67
Figure 35: Model 2 configuration	68
Figure 36: Testing a new datapoint.....	71
Figure 37: Example JSON file for testing	72
Figure 38: Output after flattening the JSON file	72
Figure 39: Tokenized words	73
Figure 40: Nearest phonetic matches	74
Figure 41:Spelling correction	74
Figure 42: Replacement of unknown words in the test phase	75
Figure 43: Test Case output example 1	76
Figure 44: Test Case output example 2	76
Figure 45: Confusion Matrix for Decision Tree	80
Figure 46: Confusion Matrix for Random Forest Tree	81
Figure 47: Confusion Matrix for Logistic Regression Classifier	82
Figure 48:Train Vs Validation Accuracy	83
Figure 49: Train vs Validation loss.....	83
Figure 50: Confusion Matrix for LSTM model 1	84
Figure 51: Train Vs Validation Accuracy	85
Figure 52: Train vs Validation loss.....	85
Figure 53: Confusion Matric for LSTM Model 2	86

List of Equations:

Equation 1: equation for one explanatory variable	27
Equation 2: equation for multiple explanatory variable	28
Equation 3:Sigmoid function.....	28
Equation 4:LSTM hidden state.....	60
Equation 5: Accuracy formula.....	78
Equation 6: Precison formula	78
Equation 7: Recall formula.....	79
Equation 8: F1 score formula	79

ABSTRACT

A little more than 20 years ago, with the burst of modern technology and web applications running, the rate at which data is being generated has risen at an alarming pace. Many more people are online today than they were at the start of the millennium. During this time period, nearly 90% of all the digital data in the globe has been created. Perhaps this can be deemed as the most dramatic consequence of the digital revolution. Businesses have been forced to re-evaluate their strategies in order to keep up with the growing demands for data-privacy and protection to remain viable and thrive in this rapidly changing environment of increasing data. To address these concerns of data collection, 137 out of 194 countries had legislation in place to assure the security of data and privacy. For instance, in accordance with the GDPR, personal data can refer to a wide variety of characteristics, ranging from direct identifiers like names to indirect identifiers like demographics and even more sophisticated data formats like location information. Categorizing this text/data within a JSON file is considered to be a Sentiment Analysis problem.

In this thesis, key variables are retrieved from a JSON file to determine whether or not the 'sensitive' data has been appropriately labelled. A number of pre-processing were required in order to clean and format the text sequence to allow it to be used as input. On the basis of the type of data in question, a Machine Learning and Deep Neural Network model is proposed in which tokenization methods such as BERT and KERAS tokenizer are implemented. On achieving the results, a Python pipeline is then created to assess a JSON path, taken directly from the JSON file, to determine whether it is capable of successfully classifying a new datapoint. This model exhibits a competitive edge over the previously obtained models with its implementation of Word2Vec embedding, effectively avoiding spelling errors.

The validity of the LSTM-based NN model is examined by performing a metric comparison with the findings obtained in light of earlier models made by other researchers. This model demonstrates promising results with an accuracy of 98% on the validation set and shows a significant improvement in precision and recall values for the data classed as 'sensitive'.

Key words: Text classification, LSTM, BERT, Natural Language Processing, word2vec, JSON

DECLARATION

I certify that this dissertation submitted for the award of the degree of MSc Data Science (Business & Management) is my own work in its all entirety, and doesn't support or advertise any other academic degree, qualification, or institution of learning. Any references from external works have been duly cited as and when required and is in accordance with the regulations and ethics guidelines applicable for the postgraduate study at the University of Manchester.

INTELLECTUAL PROPERTY STATEMENT

1. The author of this dissertation (inclusive of any appendix/schedules to this report) owns certain copyright or similar in “the Copyright” and s/he has authorized the University of Manchester to utilize such Copyright, that includes administrative purposes.
2. The copies of this dissertation, either complete or in the form of partial scripts, and electronic or hard copy, is allowed only in line with the Copyright, Designs and Patents Act 1988 (as modified) and regulations issued herewith, or wherever appropriate, in accordance with licensing agreements which the University has entered into. This page must form part of any such copies made.
3. The ownership of certain Copyright, patents, designs, watermarks, trademarks, and any other intellectual property (tagged under “Intellectual Property”) and any reproduction of the copyright material in the dissertation report, for instance, graphs and tabular representations (“Reproductions”), which is allowed for description in this dissertation, may not be solely owned by the author but may be allowed ownership by external parties. Making these Intellectual property and/or Reproductions available in any form for internal or external use without the written consent of the owners is disallowed and shouldn’t be carried out under any circumstances.
4. The [University IP Policy](#) covers all the clauses on grounds of which disclosure, publication, and commercialization of this dissertation report, the Copyright, Intellectual Property and/or Reproductions that may take place, in any relevant Dissertation restriction declarations deposited in the University Library, and the [University Library’s regulations](#).

ACKNOWLEDGEMENTS

It is an honor and a privilege for me to take this opportunity to convey my profound appreciation and gratitude to my supervisors and mentors Prof. Mark Elliot, Prof. Richard Allemendinger and Prof. Claire Little. The completion of my task is entirely and primarily due to their commitment, acute curiosity and above all their enormous willingness to assist me through my dissertation. Their timely advice, comprehensive examination, scholarly counsel, and scientific methodology have greatly assisted me in completing this assignment.

Furthermore, I would like to thank the industry partners Fred Purcell and Millie Gibbons for their genuine interest in my study at every stage of my research. Their prompt inspirations, timely suggestions with kindness, enthusiasm and dynamism have enabled me to complete my thesis. In addition, I would like to thank the University of Manchester's library for providing me with such a large array of useful resources for the completion of this thesis.

Lastly, I would want to thank my family and friends for their unending support, love, and kindness throughout this journey.

CHAPTER 1: INTRODUCTION

INTRODUCTION

The rapid advancement in the Technology and Internet has come as a boon for world. Currently Machine Learning and Deep Learning systems have provided solutions to the problems that were once perceived impossible and the applications of it are limitless. Like a coin with two sides, the evolving applications of Internet brings one of the major concerns in the modern world, **Privacy**.

Privacy, it is the ability of individual to recognize or control how, when and for what purpose their personal information is used by them or by the organization, has become one of the most important aspects of the world. This poses a big question - 'Is anyone truly private'. Despite knowing whether one is truly private or not, the **important** thing which matters right now is that how seriously Privacy is viewed now.

Privacy can be seen as a growing concern for the modern world which is affecting the corporations and people equally. There are numerous instances where a Data breach has happened and due to which people's personal information has been leaked in the internet. The GDPR regulation in European Union introduced in 2016 dictated several data protection laws which was very vital in for the protection of people privacy. This is where the vital role of Data Anonymization comes up.

Data anonymization is the process by which private or sensitive information is safeguarded through masking or a variety of other methods, making it impossible to identify a specific person using the information included. There are many ways for data anonymization that are now in use, but one of the key aspects of anonymization is the detection of sensitive data. There are several ways for the detection of the sensitive data but the area which remains unexplored is the detection of sensitive data in a JSON file.

We know that data is the fuel which runs the application, with the exponential growth of the data, there are many factors which are required now to be dealt with. One of the risks as mentioned earlier is the with the upcoming of modern technologies due to which it has become a huge deal of importance while the data has been sent and received through

applications. There are many unexplored grounds that should be investigated. One such area is the detection of sensitive information from a JSON file which we will understand in this essay.

As the technology has evolved and almost all the services are now provided to us in our hand and shows a significant growth in the transfer of data within the application. API (Application Programmable Interface) is one of the famous data transfer mechanisms, that is been used nowadays in most of the application because of its feasibility and integration within the applications. The format that is commonly used in API's is JSON. So, a lot of research needs to undertaken for the classification of the sensitive data as the data in JSON is unstructured.

Text Classification:

Text is one of the most frequent forms of unstructured data, accounting for an estimated 80% of all unstructured data. Due to the disorderly nature of text, evaluating, comprehending, organising, and sorting through text data is difficult and time-consuming; thus, most businesses do not utilise it to its full potential.

This is where machine learning and text classification comes in. Using text classifiers, businesses may be able to solve an age-old problem which is related with the data privacy. Advance Machine Learning models such as Neural networks would be able to provide solutions for accurately predicting a text sequence. This would help businesses to save time studying text data for the identification of sensitive keys and thus enable them to boost data security by flagging them.

Detecting sensitive information from a text especially from an unstructured text is a broad area of research. Unstructured text data require special attention on data pre-processing tasks as the text appears in several different formats. So, in this research work we go through several different techniques and implement Machine Learning and Neural Network models in order to identify the sensitive keys in a JSON file.

1.1 Problem Definition:

A crucial aspect is the key variable problem: which variables may an attacker use to attack a dataset? To properly safeguard a data collection, it is therefore essential to be able to identify its key variables; if you do not know what information is crucial, you cannot protect the data set. While the Industrial partner 'eXate' has procedures and systems in place to secure data, this process still requires expert judgement and manual involvement due to the inability to automatically detect critical variables in a dataset. This topic has perplexed scholars and practitioners for over three decades. Nevertheless, new machine learning techniques may provide some hope for a solution.

1.2 Project Goals:

This project will examine the use of machine learning and deep learning approaches to detect and categorise sensitive and non-sensitive variables automatically. The project will focus on unstructured data present in a JSON file and seek to address three major challenges:

- Examine the use of various machine learning and deep learning methods for detecting sensitive and not sensitive key variables.
- Constructing and optimising a machine learning or a deep learning model which is capable of identifying key variables within a dataset.
- Using string matching techniques to replace a string in which the model is not trained on.

1.3 Proposed Methodology

This study requires an extensive knowledge about the data pre-processing and various machine learning models which will suite for this type of text classification. First and foremost, data pre-processing is involved where the string sequences are converted to a required format. An exploratory data analysis is conducted which provides a detailed analysis on the type of dataset we are dealing with. Based on this understanding '*BERT*' and '*Keras*' Tokenizer is used to tokenize the text corpus. The tokenized dataset is then used to train Machine Learning and Neural Network models.

1.4 Dissertation Structure:

The whole dissertation is composed of 6 chapters which is outlined below:

- **Chapter 1: Introduction** – In this chapter we give a brief introduction about the problem context and define the aims and goals for the project.
- **Chapter 2: Literature Review** - This chapter gives us a detailed knowledge about various topics related to our study. It talks about the rise of data security concerns due to Privacy and why organisations are now taking Privacy seriously. In the later sections it gives a brief information about the technologies involved and research work from which this paper was based on.
- **Chapter 3: Design and Methodologies** - This chapter outlines the proposed methodology and different approaches involved in the whole project. Here we discuss about the technical definition of the steps involved and why performing this step are important for the implementation of Machine Learning model. Further all the steps involved in modelling different models are explained with the parameters they are configured.
- **Chapter 4: Testing** – In this chapter we design a testing pipeline and add a inference checkpoint for testing a new datapoint in a trained model.

- **Chapter 5: Evaluation and Results** – In this chapter we discuss and evaluate the different machine learning and neural network models based on some performance metrics. The final model is decided for the testing phase and finally is tested to different test cases.
- **Chapter 6: Conclusion, Limitation, Future Work-** In this section a conclusion is provided for the project work that was conducted. Further limitations were explained for which the work was limited to certain methods. Finally, some suggestions were provided to which this research could be extended.

CHAPTER 2: LITERATURE REVIEW

Background and Research

2.1 Privacy:

In the year 1890, Samuel D. Warren II and Louis Brandeis were two major American Lawyers who published an article in the *Harvard Law Review* [1], one of the first articles in the history of Privacy which shares some of first insights about the about the Right to Privacy which primarily focuses on the ‘right to be let alone’ [2]. Privacy is the claim of an individual ,group or an organisation to dictate the terms for themselves on what, how, when and to what extent their information can be used or communicated to others [3].

We are now in an era where we know that Privacy and technology go hands in hands. Data privacy is being directly impacted with the digitization. Digitization means that conversion of something which once was stored or used as analogue to a digital format. For example, documents such as Bank statements, Payslip, offer letter which was sent via mails are now sent directly to our emails electronically.

2.2 Effect of Digitization and Digitalization:

Digitization also changed things how people are accessing services. Nearly 20 years back people used to go to bank for making any payments or transfers whereas people now can do that sitting at home with the ease of their hands.

Digitalization, the use of digital technologies to enhance a business model and add an extra value to them which can reap benefits, is simply a process to move into a digital business [4]. Transforming a business model into a digital has made the business much more data centric. Organisations have started using analytics in which they started learning about the customer patterns and behaviour. In doing so they started making personalised adds and created unique benefits for each customer [5]. While this gave a strategic edge to the businesses, on the other side the privacy concerns grew over the use of the customer’s data. Currently all the business, even small or big they have started collecting personal details about their customers.

The information that businesses get about their clients is highly valued. Businesses can use this data to send out customised advertising, forecast sales patterns, and enhance their products. But customers naturally have a different perspective. Many people view data collection as an invasion of their privacy and an easily exploitable process, which makes many corporations as a substance of suspicion and mistrust [6].

2.3 Report on Data Responsibility by KPMG:

There's a report released by KPMG, *Corporate Data Responsibility: Bridging the Trust Chasm*, [8] which provides a detailed overview about the growing concerns among the clients about the methodologies of collection of data which is been increasing in recent times and also provides guidance to the corporation on how to deal with such concerns. This report has been based on the two online surveys that were conducted by KPMG itself. The first survey consists of findings that are taken from 2000 United State Adults whereas the second survey consists of 250 decision makers which are involved in privacy and security at corporations having more than 1000 employees.

Below are the following key findings that were found during the first survey,

- Around 86% of the population stated that the data privacy is a growing concern for them
- Around 68% of people are concerned about how much data firms are collecting.
- 40% don't trust businesses to utilise their data in an ethical manner.
- 30% of people said they would never reveal their personal information.

Below are the following key findings that were found during the second survey,

- 70% claim that during the last year, their business has boosted the gathering of customer personal data.
- 62% of respondents believe their business might improve its current data protection policies.
- Consumers should be worried about how their firm uses their personal data, according to 33% of respondents.

Data collection has really been getting more common despite the rising worry over it. 70% of the businesses KPMG examined increased the amount of information they collected on individual customers last year. 95% of the business executives surveyed indicated their organisation had good or very strong data protection mechanisms in place, and 75% said they are happy with the amount of data collection.

However, even for companies that do gather data, there are cautionary indicators. 62% of the company executives polled said that their organisations should take additional steps to secure client data. A third of them said that customers should be more worried about how their firm uses their data, and 29% acknowledged that their organisation occasionally utilised unethical methods to gather sensitive information.

Consumers have become progressively more dubious and apprehensive of the collection of their personal information. 86% of those surveyed indicated they have rising concerns about data privacy, while 78% voiced trepidation over the volume of data being gathered. A little over 40% of customers polled don't believe businesses would utilise their data in an ethical manner, and 13% don't even believe their own employers.

Consumers are concerned about how their data may be corrupted or sold to third parties, in addition to the act of data collection itself. In comparison to 51% who were worried about their data being sold, 47% of respondents indicated they were worried about the prospect of their data being stolen. Ironically, just 17% of the business executives polled admitted that their firm sells data to third parties, indicating that companies need to be more upfront about this practise in order to allay customer concerns.

This gap between corporate and consumer attitude is not new, but its persistence demonstrates that businesses still have a long way to go to gain the public's trust in their data collection, use, and security practises, according to Orson Lucas, KPMG's U.S. privacy services leader. A genuine risk of losing access to the priceless data and insights that fuel growth exists if this barrier is not closed.

People are less eager to divulge personal information as concerns over data gathering increase. 30% of the people polled claimed they would never share their personal information

with companies. Just 12% of respondents said they would give data to make advertising more relevant, and 17% said they would contribute data to assist businesses improve their goods and services.

Consumers are open to sharing information in particular circumstances notwithstanding their concern. While 52% of respondents were okay with firms recording calls for training and quality purposes, 57% of respondents agreed that the use of face recognition technology in criminal investigations is appropriate.

2.4 The growing need for Privacy:

We humans always had the need for privacy. In the writings of past there was book name 'One of Politics' in which Socrates and other Greek philosophers mentioned about Privacy [7]. In the book a distinction is made between the 'outer' and the 'inner', between public and private, and between society and solitude.

In the current times we have to understand that everybody needs to have Privacy. The definition of Privacy differs from region to region, culture to culture and also to person to person but at the same time we must acknowledge that the need to privacy is absolute and should be balanced against the other needs, for example for example the need for fighting terrorism, criminality, and fraud.

2.5 Privacy seen in the past and now:

Privacy has always been attacked in different situations with different ways. In the paper (History of Privacy), A distinction has been made between three different generations in what ways the privacy was invaded. We know that, during 1980 there was a major revolution introduced to the importance of Privacy after the publication of article 'The Right to Privacy' in the Harvard Law Review [1].

So, the article describes three generations, the situation before 1980, the situation after 1980 to 2008 and the invasion to the Privacy that can happen in the future.

In the past actions such as trespassing, correspondence, the press, Instantaneous Photography, Wiretapping, Psychological Testing and Lie Detectors were considered as the attacks to privacy. With the time and the advancement of the technology, things such as Video Surveillance, Biometric Identification, Genetic Data, Data Warehousing and Data Mining, Global Positioning System (GPS), Internet, Radio Frequency Identification (RFID), Wireless Networking increased the ways in which privacy could be invaded.[3] In the future things such as Ambient Technology, Neurolinguistics, Memetics, Grid Technology [3] have also been considered as a use through which privacy could be invaded.

2.6 Rise in data breaches and concerns to privacy:

In Recent times, the cyberattacks has been widely growing.[9] A cyberattack is a person's attempt to damage or shut down a network, computer system, or numerous systems in order to obtain the information for personal gain. As per the statistics published in an article it has been mentioned that a maximum breach size of roughly 200 million has happened from the years 2000 to 2015 and over the next 5 years, it is anticipated to increase by 50%.[10] The model which has also projected that within the next five years, estimates show that the entire amount of compromised information will quadruple from two to four billion pieces, surpassing the number of Internet users.[10]

The enormous and unchecked release of personal information from the organisations creates serious privacy issues has facilitated widespread identity fraud and has led Government in the countries to think about the very importance of the Privacy. In the light of this, several steps have been taken by government of several countries. For example, UK and EU started to enforce a Data Protection act, known as GDPR (General Data Protection Regulation) in the year 2018. This law dictates that ‘Everyone in charge of using personal data is required to abide by stringent guidelines known as “data protection principles”’.[12]

2.7 Data Anonymization process:

Corporations today are collecting a large amount of personal Identifiable Information (PII). Information that may be used alone or in combination with other pertinent data to identify a specific person is known as personally identifiable information (PII) which is being stored in

the databases and has become a common practise currently. As this might create a significant privacy concern for the database. Many methods for preserving privacy have been put forth, including perturbation [13], anonymization [14], and cryptography [15].

There are several anonymization techniques such as Generalization, Suppression, Distortion, Swapping, Masking which are in place for anonymizing the data.[16] Each type has its own unique way to anonymize the Data.

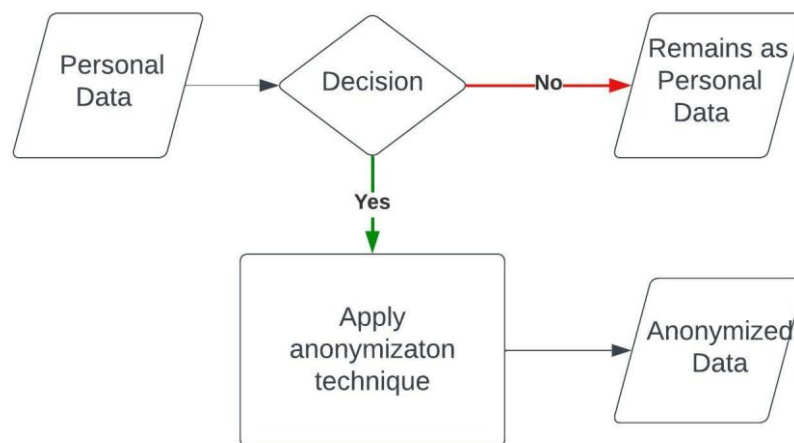


Figure 1:Anonymization Process for Personal Data

The above flowchart shows a simple process description of Anonymization of the Personal Data. But before the process of Anonymization there is one of important step which is the detection of the Sensitive Data. Detection of sensitive data is one of difficult problems as textual data might be in different formats which also needs specific operations for each of them.

2.8 Web applications and API's using JSON structure:

Now a days JSON (JavaScript Object Notation) has become one of the popular methodologies for data transfer among the web applications and services. It's and key-value based data exchange format is JSON. [17] JSON's simplicity makes it simple to read and write for humans as well as to produce and interpret for machines. Below mentioned is an

JSON example, where the text mentioned as “emp_details”,” name”,” id” are the keys and the words followed after the “:” (colon symbol) are the values pairs for each one of them.

```
obj= {"emp_details":[  
    {"name": "a",  
     "id": "123"  
    },  
    {"name": "b",  
     "id": "345"  
    }  
]
```

Figure 2:JSON Document

We can see that under employee details there are two keys which can be termed as a direct identifier and thus can be regarded as Personal Identifiable Information. In this paper we are mostly focusing on the detection of the key pairs.

We already have known that JSON’s type file format has become one of common and easiest way of data transfers mechanisms in web-based applications and services.

During a data transfer there are usually two parties involved in it one is the client side and another is the server side. In the Client side the data is being sent to the server side as URL encoded JSON string as a part of the HTTP request. There are some common methods such as HEAD, GET, or POST which are used to send the object. The REST method in the server side allows the JSON string to be received and then be parsed. Then according to the request received a response is provided to the client as a JSON object string.

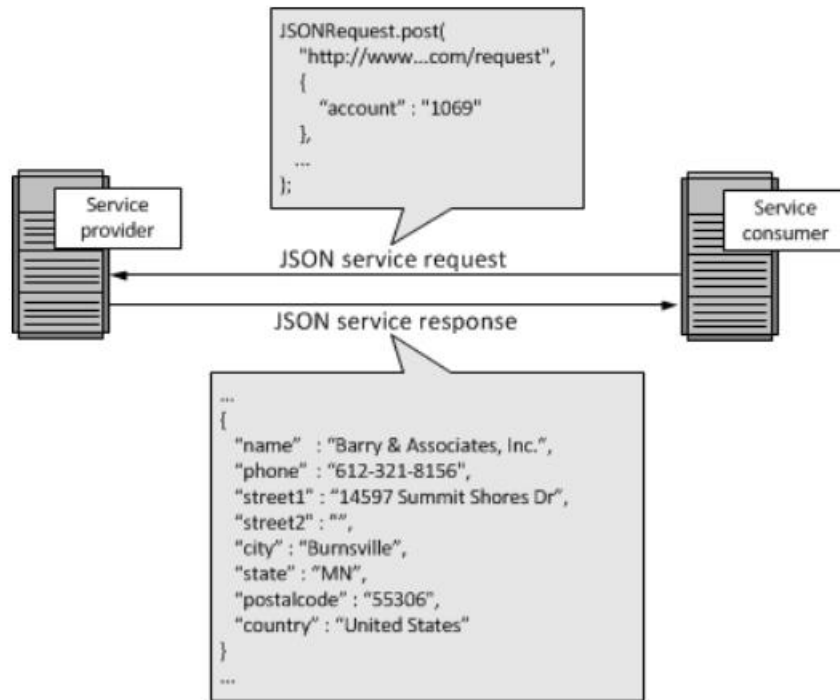


Figure 3: JSON transfer mechanism [38]

As the number of web-based applications have been increasing significantly the number of JSON transaction between servers have also proliferated. As per the GDPR protocols it is imperative now that the information that is been sent or received has to be anonymized. As the JSON file which is transferred could contain some personal data which could be also treated as a Personal Identifiable Information.

The main problem objective of our paper is to be able to identify the key variables which can be also termed as PII using various machine learning algorithm and approaches.

2.9 Deep diving into Text Classification techniques and implementation:

With the recent advancement of NLP (Natural Language Processing) and text mining approaches, organisations and academics are investing substantially in text classification applications. In the study [20], multiple text algorithms are explained briefly, focusing on the extraction of various text characteristics and the usage of existing dimensionality reduction methods, algorithms, and approaches.

There are typically four main scopes that may be used to text categorization, which are described below.

- Document level: This method retrieves the required document categories at the document level.
- Paragraph Level: This method retrieves the appropriate categories of a single paragraph at the paragraph level (a portion of a document).
- Sentence Level: At the sentence level, retrieves the necessary categories for a single sentence (a paragraph fragment) [39].
- Sub-Sentence Level: The method extracts the appropriate categories of sub-expressions within a phrase at the sub-sentence level (a portion of a sentence)

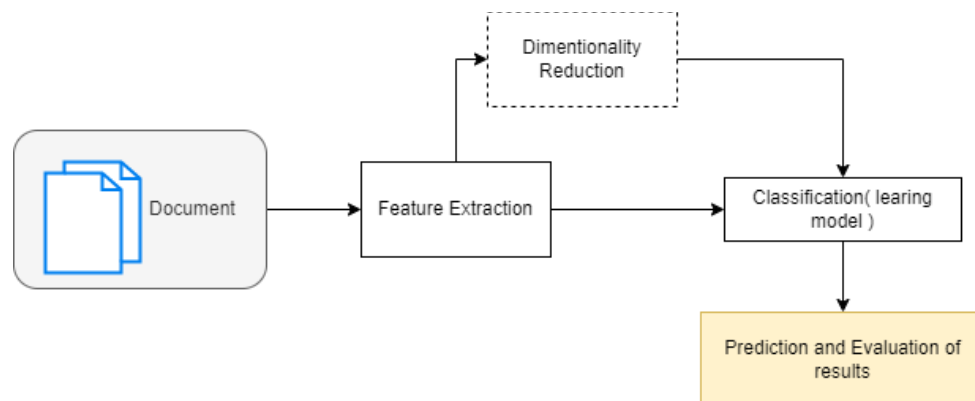


Figure 4: Text classification process

In the above figure 4, a flow chart represents the process of text classification is short.

2.9.1 Feature Extraction:

This is the first stage in solving any text-based problem that involves categorization. In most cases, the text or documents that we obtain arrive in the form of an unstructured dataset. This means that the dataset has to be pre-processed and tokenized before it can be turned into

features. Therefore, the transformation of an unstructured text sequence into a structured feature space is of the utmost significance for a text classifier that would make use of mathematical modelling.

In the first phase we have to clean the dataset to remove the unnecessary characters and words. Only after cleaning the dataset, then we can proceed to for the feature extraction.

Some of the common feature extraction methods are:

- Term Frequency-Inverse Document Frequency (TD-IDF) [40]
- Term Frequency [21]
- Word2Vec [22]
- Global Vectors for Word Representation (GloVe) [23]

Term Frequency-Inverse Document Frequency (TD-IDF):

This statistical metric assesses a word's relevance to a document or corpus of text. [24] This is accomplished by multiplying two metrics: the number of times a phrase or a word appears in a document and its inverse document frequency over a range of documents. It has a number of uses, most notably in automated text analysis, and may be quite beneficial when used for scoring words in machine learning and data science methods for Natural language Processing (NLP) [24].

Term Frequency:

Term Frequency is a statistical metric that indicates the frequency with which a phrase or word appears in a text. Documents can be of different lengths, and the frequency of appearance of a term in a document can vary from the shorter ones to the longer ones, so conventionally, the term frequency is divided by the total number of terms in the document in order to achieve normalization.[21]

Word2Vec:

Word2Vec is a recent advancement in the field of NLP. It was directed by Tomas Mikolov, a Czech computer scientist and CIIRC researcher (Czech Institute of Informatics, Robotics and Cybernetics). Word2Vec is one of the prevalent techniques for generating word embeddings [41] in which the text is envisioned as a vectorized representation. All the individual words are transformed into unique integers in a vector space. They can be applied in a range of applications, including sentiment analysis, text similarity checks, and recommendation systems, among others. [22]

GloVe:

GloVe is an acronym that stands for global vectors for word representation. It is a Stanford-developed unsupervised learning system for producing word embeddings by aggregating a corpus (collection of written documents) global word-word co-occurrence matrix. In vector space, the resultant embeddings reveal intriguing linear substructures of the word.[\[23\]](#)

2.9.2 Dimensionality Reduction:

Since text or document data collections often contain many unique words, data pre-processing operations can be time- and memory-intensive. Utilizing affordable algorithms is a frequent method for solving this problem. However, in some data sets, the performance of these inexpensive algorithms falls short of expectations. To minimise dimensionality severity during pre-processing, words that do not contribute to the classification job (such as articles, verbs, and prepositions) are eliminated.[25] Many academics choose to employ dimensionality reduction to lower their applications' time and memory complexity to avoid a decline in performance. Using dimensionality reduction for pre-processing may be more effective than constructing affordable classifiers.

2.9.3 Classification techniques:

Choosing the optimal classifier is the most crucial stage in the classification process. We cannot choose the most successful model for a text categorization application without a comprehensive conceptual knowledge of each approach.

Some of the classification methods that we are going to discuss would be logistic regression classifier, tree-based classifiers such as decision tree and random tree and deep learning approaches such as the use of LSTM and other network models. On a variety of tasks, such as image classification, natural language processing, and facial recognition, etc., deep learning techniques have recently outperformed older machine learning methods. The effectiveness of these deep learning algorithms is dependent on their ability to represent complicated and nonlinear data connections [26].

Logistic Regression:

It is one of the most popular approaches used for binary classification. It is a classification technique that uses supervised learning to anticipate observations for a limited set of classes. In practice, it is used to categorise observations into distinct groups. Consequently, its output is discrete. Logit Regression is another name for Logistic Regression. It is one of the simplest, straight forward and adaptable classification algorithms which is used to tackle classification issues [42].

Implementation:

The Logistic Regression approach predicts a response value by executing a linear equation using independent or explanatory factors. Consider the example of the number of hours spent studying and the chance of passing the exam. Number of hours studied is the explanatory variable given by x_1 in this instance. z denotes the probability of passing the exam, which is the response or target variable.

The equation for one explanatory variable can be stated as:

Equation 1: equation for one explanatory variable

$$z = \beta_0 + \beta_1 x_1$$

where the coefficients β_0 and β_1 are the parameters of the model,

If there are more than one independent variable, then the equation above can be extended as

Equation 2: equation for multiple explanatory variable

$$z = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_n x_n$$

where the coefficients β_0 , β_1 , β_2 and β_n are the parameters of the model

the predicted response value is in the above two equation is denoted as z .

This projected response value, represented by z , is then transformed into a probability value ranging from 0 to 1. The sigmoid function is used to convert anticipated values to probability values. This sigmoid function then converts any actual number to a probability between 0 and 1.

The sigmoid function is used to transfer predictions to probabilities in machine learning. The sigmoid function has a curve with an S shape. This is also known as the sigmoid curve.

A Sigmoid function is a particular instance of a Logistic function. The following mathematical formula describes it.

Equation 3: Sigmoid function

$$\phi(z) = \frac{1}{1 + e^{-z}}$$

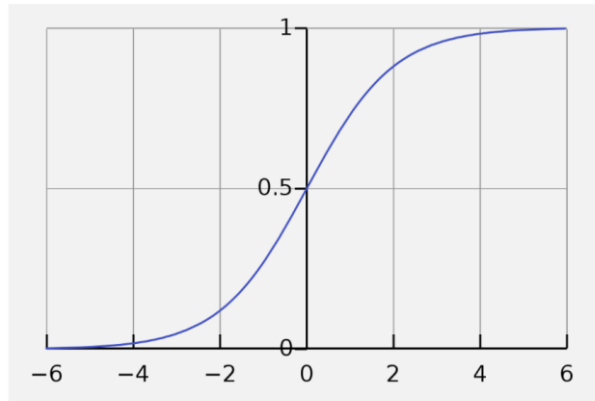


Figure 5: Sigmoid function [43]

The figure above shows the curve for the sigmoid function. The sigmoid function produces a probability value ranging from 0 to 1. Then, this probability value is transferred to a discrete class consisting of "0" or "1". We set a threshold value in order to transfer this probability value to a discrete class (pass/fail, yes/no, true/false). This number is known as the Decision boundary. Above this level, the probability values will be mapped into class 1; below it, they will be mapped into class 0.

Limitation:

Logit regression necessitates that each data point be independent of the others. If observations are interrelated, then the model will tend to overestimate their relevance.[29]

Decision Tree Classifier:

Decision tree can be termed as one of the popular supervised machine learning algorithms. It uses a set of rules to make decisions, like how humans make decisions.

It leverages the dataset characteristics to generate branches containing yes or no questions, and then repeatedly separates the dataset until all data points are isolated into their respective groups. After the outcome of each yes or no question, the dataset is partitioned depending on the specific value and extra nodes are created. The original node is known as the Root node,

whereas the last nodes formed are known as Leaf nodes. Figure 6 depicts the process flow of a Decision tree.

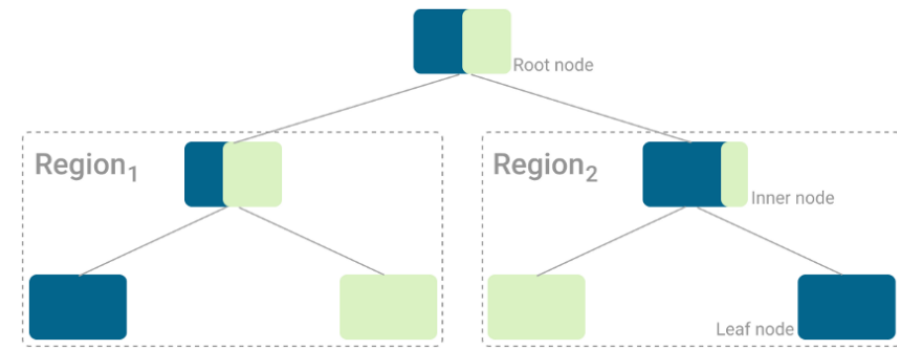


Figure 6 :Decision Tree structure [44]

Limitations:

The decision tree is a very fast method for both learning and prediction; nevertheless, it is highly sensitive to tiny data perturbations [30] and is readily overfit [31]. This is a grey area, although validation procedures and trimming can mitigate these consequences [30]. This model has issues with out-of-sample prediction as well [32].

Random Forest Classifier

On the other hand, a random forest consists of large number of decision trees instead of relying in a single Decision tree. Each Decision tree present within the random forest is created and trained from a subset of the dataset.

For example, if a Random Forest has 5 decision tress and 10 features.

At the first (root) node of the first decision tree, the algorithm would randomly choose three characteristics (the square root of 10 is 3.2). It then determines the optimal split and selects three random characteristics to assess the subsequent split. This is repeated for every split in every tree.

This procedure creates trees that have been exposed to various types of data, which are then combined to give a final prediction. Then the majority of the votes [28] are taken into account from the given predicted values as the final result.

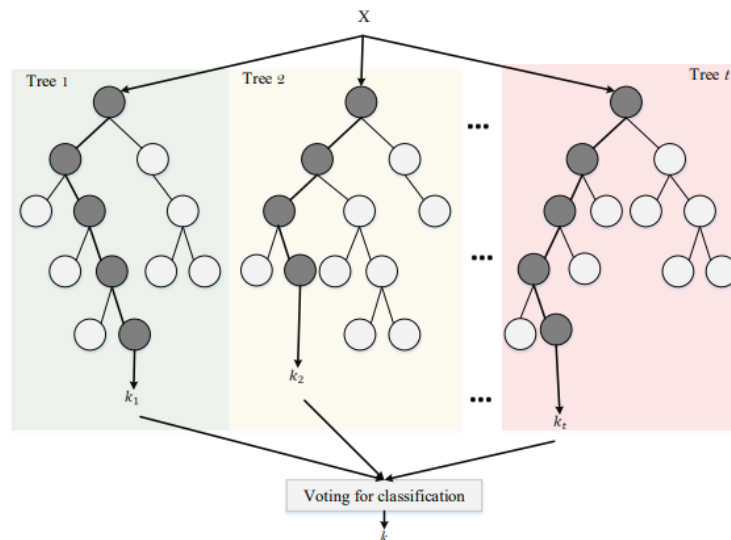


Figure 7: Random Forest tree structure [20]

Limitations:

While training the text datasets in Random Forest Tree, it is usually very fast and takes very less time compared to the deep neural networks. But once they are trained, they are quite slow to create predictions [33]. In order to obtain a speedier structure, it is necessary to limit the number of trees in the Random Forest. As the number of trees in a Random Forest rise, so does the time complexity of the prediction step.

Deep learning:

When it comes to NLP tasks neural networks have significantly provided impressive results compared to the conventional Machine Learning algorithms. Deep neural networks process data in complex ways by employing sophisticated math modelling. DNN have shown impressive results for variety of applications such as image classification, speech recognition etc or the tasks dealing with huge amount of data.

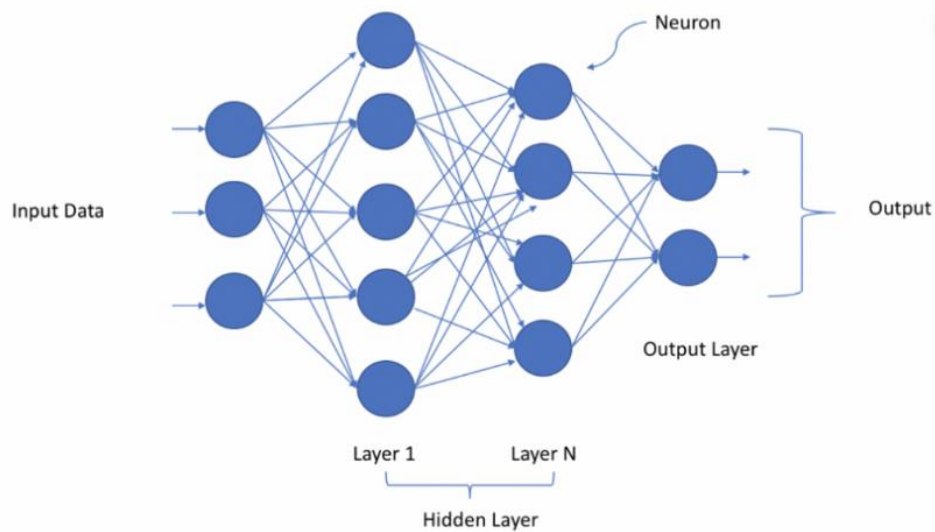


Figure 8: Neural Network structure

Deep neural networks (DNN) are meant to learn via multi-connection of layers such that each layer only receives connections from the previous layer and only offers connections to the next layer in the hidden section

In the deep neural network, Figure 8 illustrates the construction of a typical DNN. The input comprises the relationship between the input feature space and the DNN's first hidden layer. The input layer may be created with TF-IDF, word embedding, or another type of feature extraction. For multi-class classification, the output layer is equal to the number of classes, while for binary classification, it is a single layer. Each learning model is created in multi-class DNNs (the number of nodes in each layer and the number of layers is fully arbitrary).

ANN:

ANN (Artificial Neural Network) is a type of feedforward network [46]. This means that the information flows from one layer to another through the nodes without touching the same node twice. It comprises of three layers which are input layer, hidden layer and output layer. The input layer is responsible for accepting the inputs, the hidden layer processes input, and the output layer produces the output.

ANN performs well when it comes to 1-dimensional tabular data, image data and text data but the complexity increases while solving an image classification problem. While feeding the input layer, the data in 2 dimensional has to be converted to 1 dimensional before training the model.

There are some drawbacks which arise from these which are:

- The number of trainable parameters rises dramatically as the size of the picture increases.
- The spatial properties of a picture are lost via ANN. Spatial characteristics are the arrangement of pixels inside an image.
- ANN is incapable of capturing sequential information in input data, which is necessary for processing sequence data.

RNN:

To overcome the limitation in the previous ANN model, RNN (Recurrent Neural Network) is introduced. The structure of the RNN is the same as that of ANN, as it also primarily consists of three layers. But while ANN has a feed-forward information transverse, On the hidden state, RNN features a recurrent connection [\[47\]](#). This looping requirement ensures that the supplied data has sequential information.

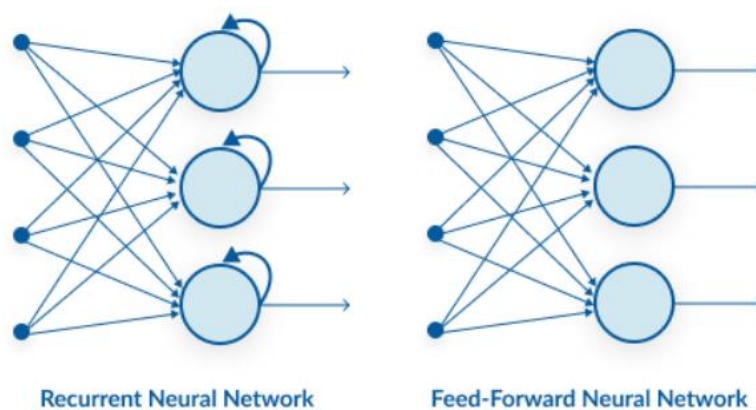


Figure 9: Recurrent neural vs Feed-Forward Neural Network [48]

RNN works well with textual data because it can capture the sequential information included in the input data. This enables the system to comprehend the relationship between the words in the text when generating predictions.

Even though RNN performs well with sequential text input, its fundamental disadvantage is the vanishing gradient and exploding gradient problem [36], which is a prevalent issue across neural network types. The ANN network likewise exhibited the same issue.

LSTM:

Long-Term Short-Term Memory is what LSTM stands for. The LSTM neural network model eliminates the drawbacks that are inherent to the other neural network models that have been discussed.

LSTM can be explained as a type of recurrent neural network, although it has superior memory over typical recurrent neural networks. Having a firm grasp on retaining certain patterns, LSTMs function considerably better. When with any other Neural Network, LSTM may have numerous hidden layers, and as it traverses each layer, only relevant information is retained, and all irrelevant information is eliminated in each cell. Importantly, it eliminates the vanishing gradient [35] and exploding gradient problem, which was considered a significant shortcoming in above discussed neural networks.

Working of LSTM:

LSTM has three types of gates which are mentioned below:

- FORGET Gate
- INPUT Gate
- Output Gate

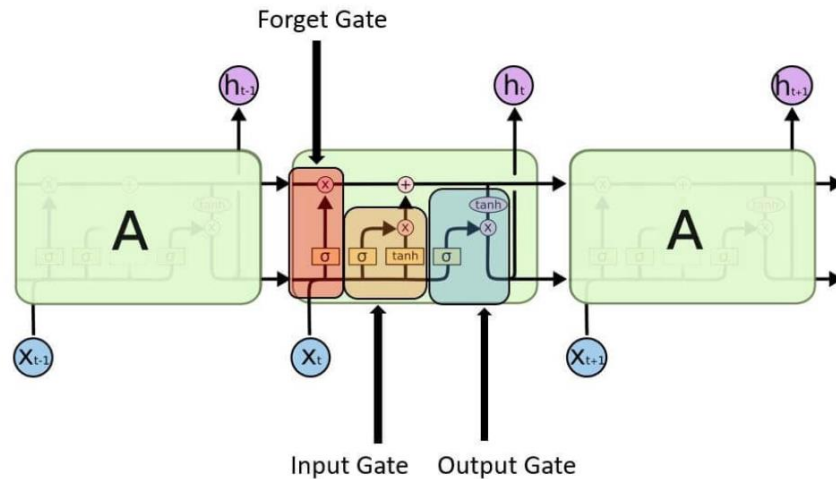


Figure 10: LSTM Structure [49]

FORGET Gate:

This gate is in charge of determining which pieces of information should be stored in order to accurately calculate the state of the cell, and which pieces of information are superfluous and may be discarded.

INPUT Gate:

The Input Gate is responsible for bringing the current state of the cell up to date and determining which pieces of information are vital and which are not. In the same way that the forget gate is helpful in discarding information, the input gate is useful in determining what information is essential and storing data in the memory that is pertinent to the situation.

OUTPUT Gate:

The last gate, known as the Output gate, is responsible for determining what the next hidden state will be.

LSTM for text classification:

Short-term memory is an issue that impacts traditional neural networks. The problem of disappearing gradients is another significant downside of this method. (During the process of backpropagation, the gradient approaches zero as it gets smaller and smaller, and a neuron in which this occurs is useless for further processing.) LSTMs are able to effectively boost performance by memorising the pertinent information that is necessary and locating the pattern.

If we examine various non-neural network classification algorithms, we find that they are trained on many words as distinct inputs that have no actual meaning as a sentence, and when predicting the class, the output is based on statistics rather than meaning. This implies that each and every word is sorted into one of the groups.

In LSTM, this is not the same. In LSTM, a string of numerous words can be used to determine the class to which it belongs. This is beneficial when working with Natural language processing. If we employ proper layers of embedding and encoding in LSTM, the model will be able to determine the real meaning of the input text and provide the most precise output class.

2.10 Automatic Anonymization of Textual Documents: Detecting Sensitive Information via Word Embeddings:

This research journal talks about the identifying the sensitive information using word embeddings. This paper mostly focuses on the anonymizing texts which are unstructured [17].

Structured and unstructured texts are the two major categories that texts can be placed into. By far though extracting sensitive information from structured text is a highly challenging task, doing it from unstructured text presents even greater challenges. This research suggests employing word embedding models based on neural networks or by-word vectorization to compute the linguistics to assess the relatedness of words to identify sensitive keywords.

This paper also discusses about the NER (Named Entity recognition) [18] method that was typically used for detecting the sensitive information. The major drawback in this method was that for the training of NER Classifiers, a large amount of manually tagged training data was required which takes a considerable effort [17]. Mostly in the process of Document Anonymization there are main two steps involved, one is the detection of the pieces of information and then second step to anonymize the same with an appropriate masking method.

Word Embedding is a method where it maps the words into high dimensions. In this concept we try to arrange the words which have the similar meaning in same dimension.[17] The semantic relationship between the words is assessed by some of statistical methods such as distributional or probabilistic models.

The neural network is either trained to predict the current word from a window of adjacent words (continuous bag of words model) or to predict adjacent words depending on the current word (skip-gram) [17]. In order to accomplish this, the network utilises a collection of documents or phrases as input data and constructs a vocabulary from the terms in the collection. For each word in the vocabulary, the weights acquired by training the neural network are utilised as the vector associated with that word [17].

Here, the skip-gram model is used, which is more accurate than the continuous bag of words model [50] and produces output probabilities representing the likelihood of locating a vocabulary word in the neighborhood of the input word. Due to this useful property, the skip-gram word embedding model was employed in this study to detect phrases in a text that may jeopardize the privacy of a particular entity.

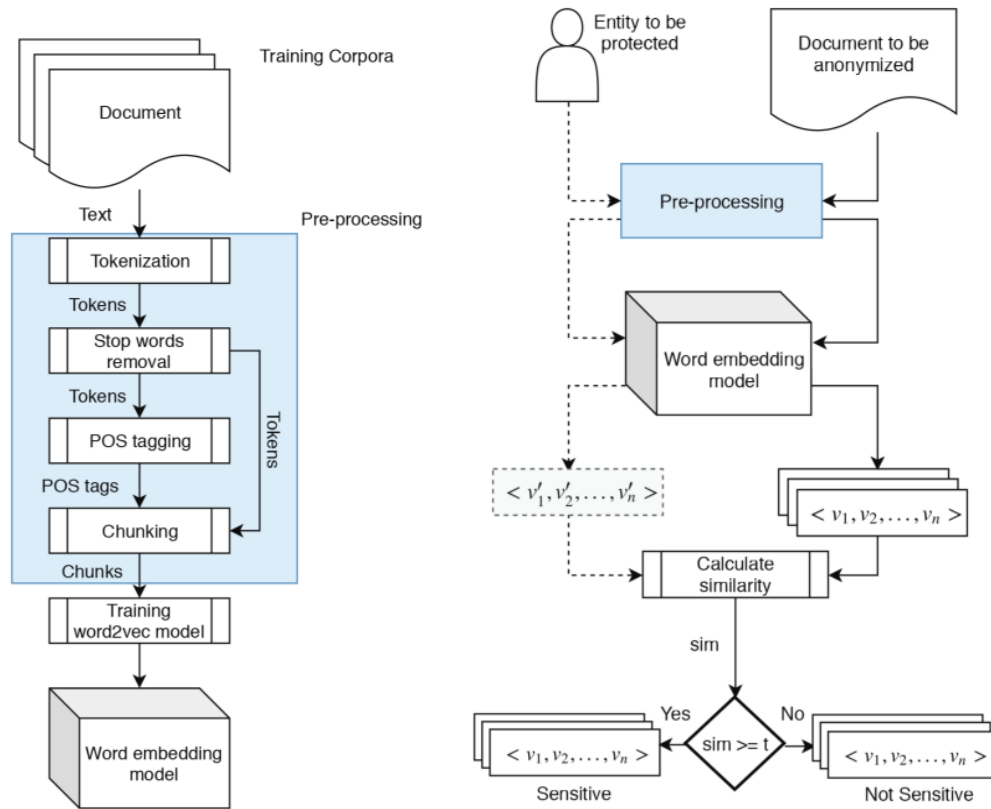


Figure 11: Word embedding model & Training and Classification [17]

The strategy described in the article consists of two steps. In the first phase, a huge corpus is utilized to train a custom-tailored word embedding model that captures the semantic link that may induce disclosure. In the second phase, a neural network model is trained to identify the protected document's words.

CHAPTER 3:DESIGN AND METHODOLOGIES

In this chapter, we discuss the approaches that we have implemented with the knowledge that we have obtained from the previous papers and the research work that has been carried out. The topics mentioned in this section are: all of the pre-processing activities that were performed on the dataset, the information that was gathered from the exploratory data analysis, tokenization methods and the various modelling techniques that were used to train the data. **eXate** (A private company which streamlines, automates, and simplifies the processes of storing, interpreting and extracting value from data assets) was kind enough to supply us with the dataset that we utilised for this project. In order to determine the most effective strategy for resolving the issue, a variety of natural language processing (NLP) activities, machine learning, and deep learning functionalities have been utilised. The code for the software was developed in the Python programming language, and an IDE (Integrated Development Environment) [51] like Jupyter or Google Collab was utilised in order to run the code on a computer. A discussion of the many different techniques is presented in chapter 4 of this paper.

3.1 Project Approach and Design Aspects

We have explored different methods and approaches on modelling techniques. In the first approach on classifying the text as sensitive or not sensitive we have user BERT tokenizer to tokenize the text where we have utilised conventional machine learning algorithms such as Decision tree, Random Forest classifiers which are tree-based structures and Logistic Regression Classifier which is a logistic model to classify the labels. Then in the second part we have used Keras Tokenizer and then implemented DNN based LSTM with different embedding layer.

3.1.1 Implementation Process Flow Chart:

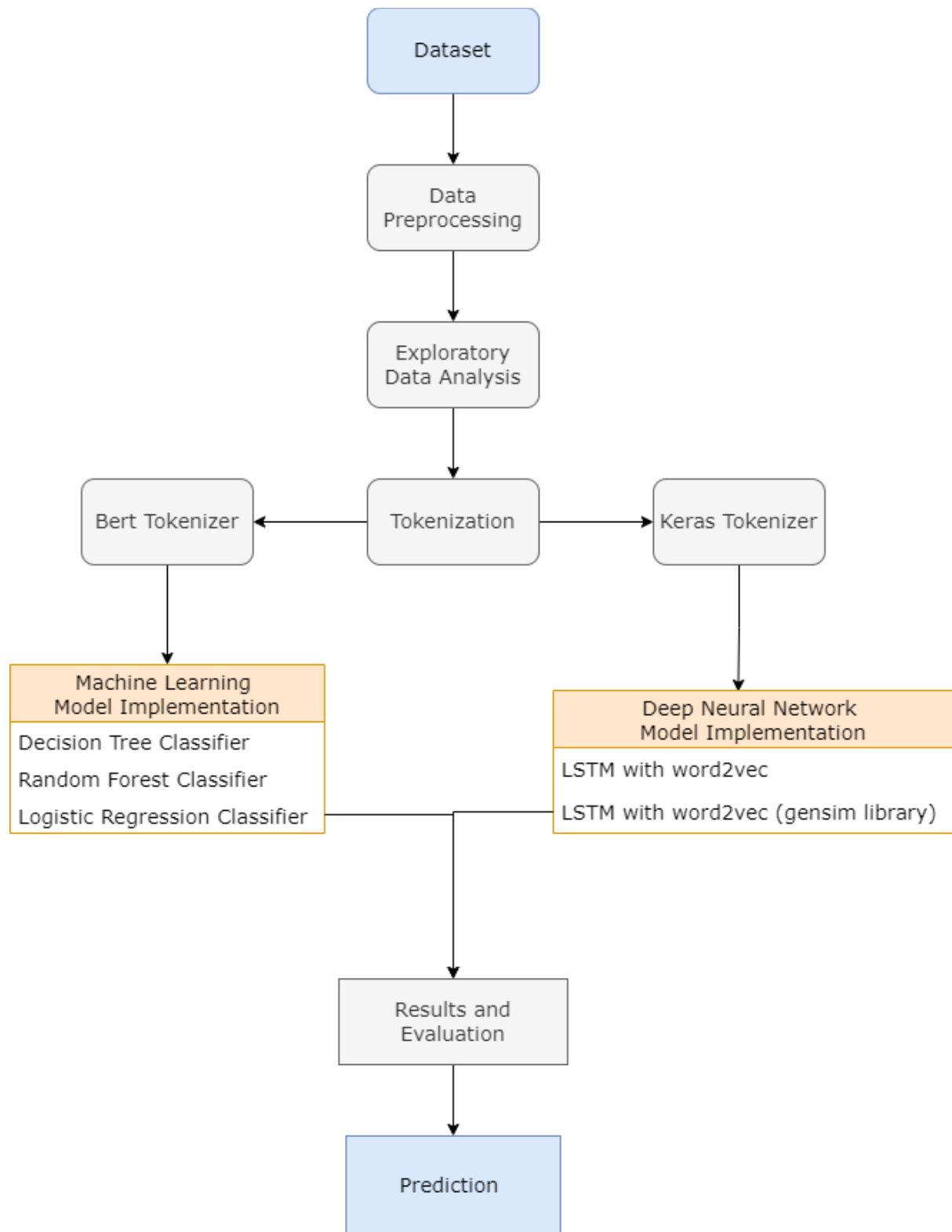


Figure 12: Implementation Process flow

3.1.2 Version information for Python libraries:

Most of the program code is run on Jupyter notebook and Google Collab. Google collab has been utilised mostly due to its computation capabilities for training neural networks.

```
!nvidia-smi
```

```
Tue Sep 13 18:46:51 2022
```

NVIDIA-SMI 516.94				Driver Version: 516.94				CUDA Version: 11.7			
GPU	Name	TCC/WDDM		Bus-Id	Disp.A	Volatile	Uncorr.	ECC			
Fan	Temp	Perf	Pwr:Usage/Cap	Memory-Usage		GPU-Util	Compute	M.			
											MIG M.
0	NVIDIA GeForce ...	WDDM		00000000:01:00.0	Off					N/A	
N/A	49C	P8	1W / N/A	0MiB / 4096MiB		0%	Default		N/A		

```
Processes:
```

GPU	GI	CI	PID	Type	Process name	GPU Memory
ID	ID					Usage

```
No running processes found
```

Figure 13: The runtime configuration setup used for Jupyter notebook

```
!nvidia-smi
```

```
Tue Sep 13 17:47:27 2022
```

NVIDIA-SMI 460.32.03				Driver Version: 460.32.03				CUDA Version: 11.2			
GPU	Name	Persistence-M		Bus-Id	Disp.A	Volatile	Uncorr.	ECC			
Fan	Temp	Perf	Pwr:Usage/Cap	Memory-Usage		GPU-Util	Compute	M.			
											MIG M.
0	Tesla T4		Off	00000000:00:04.0	Off					0	
N/A	68C	P8	11W / 70W	0MiB / 15109MiB		0%	Default		N/A		

```
Processes:
```

GPU	GI	CI	PID	Type	Process name	GPU Memory
ID	ID					Usage

```
No running processes found
```

Figure 14: The runtime configuration setup used for Google Collab Notebook.

The package information which was needed to be installed in the Collab Notebook are mentioned below in the table.

Table 1:Version information about different utilities installed

Package (Utility)	Jupyter Notebook version	Google Notebook version
Python implementation	Cpython	CPython
Python version	3.8.8	3.7.14
IPython version	7.29.0	7.9.0
numpy	1.20.3	1.21.6
pandas	1.3.4	1.3.5
keras	2.8.0	2.8.0
transformers	4.21.1	4.22.1
wordcloud	1.8.2.2	1.8.2.2
tensorflow	2.8.0	2.8.2
nltk	3.6.5	3.7

Below mentioned are various operations were conducted in this project:

1. Installation of all the required libraries
2. Understanding the Dataset
3. Data Pre-processing
4. Tokenisation methods
5. Machine Learning and Deep learning modelling
6. Model Implementation

3.2 Libraries used:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import transformers
import wordcloud
import tensorflow
import nltk
from keras.preprocessing import sequence
from keras.models import Sequential
from keras.layers import Dense, Dropout, Embedding, LSTM, Bidirectional

import warnings
warnings.filterwarnings("ignore")
```

Figure 15: Libraries used

The libraries mentioned above in figure 14 have been utilized majorly to perform all the operations in the creation of different machine and deep learning models.

3.3 Understanding the Dataset:

Understanding the dataset gives us a broader picture of the whole dataset and let us the know the types of pre-processing tasks that we are supposed to perform in the dataset.

The dataset provided is loaded into a ‘pandas’ dataframe.

Unnamed: 0		Data	Label
0	1	\$candidates.candidateId	NaN
1	2	\$candidates.workspace	JP
2	3	\$candidates.tealId	NaN
3	4	\$candidates.title	NaN
4	5	\$candidates.firstName	FINA
...
18387	18388	\$paymentMethods.check.zip	PC
18388	18389	\$paymentMethods.check.countryId	CC
18389	18390	\$hp	NaN
18390	18391	\$subDistrict	CITY
18391	18392	\$urbanVillage	NaN

Figure 16: Initial Dataset

Figure 16 shows the data information that the dataset contains. The column named 'Data' stores the JSON path keywords whereas the keywords which are termed as sensitive are assigned with an abbreviation in the 'Label' column. Rest of the values which are not sensitive are blank and so the dataframe sets them as 'NaN' (i.e.) null values.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 18392 entries, 0 to 18391
Data columns (total 3 columns):
#   Column      Non-Null Count  Dtype
---  ---
0   Unnamed: 0  18392 non-null  int64
1   Data        18390 non-null  object
2   Label       2647 non-null   object
dtypes: int64(1), object(2)
memory usage: 431.2+ KB
```

Figure 17: Dataset Description

The 'describe()' function shows the description of the dataset. It shows the count of not null values in each column and the data type of each column.

Abbreviation		Keys
0	BP	Birthplace
1	CC	City
2	CCN	Credit card number
3	CITY	City
4	CR	Criminal record
5	DI	Digital identity
6	DL	Drivers license number
7	DOB	Date of birth
8	EMAIL	email
9	FACE	Face
10	FINA	First name
11	FN	Full name
12	FP	Fingerprints
13	GI	Genetic information
14	GN	Gender
15	GRA	Grades
16	HADD	Home address
17	IP	IP address
18	JP	Job position
19	LN	Last name
20	NI	National identification number
21	PC	Postcode
22	PP	Passport number
23	RACE	Race
24	SCH	School
25	SL	Salary
26	ST	State
27	TN	Telephone number
28	VR	Vehicle registration
29	WP	Workplace

Figure 18:Identifiers Dataset

The imported dataframe in figure 18 shows the description of each sensitive keyword.

```
Unnamed: 0      0
Data           2
Label        15745
dtype: int64
```

Figure 19: count of null values in the main dataset

The above figure 19 shows the count of null values in each of the columns of the data frame.

3. 4 Data Pre-processing:

Data pre-processing is a phase in the data mining and data analysis process that converts raw data into a format that computers and machine learning algorithms can understand and evaluate.

Machines like to process information that is neat and orderly; they interpret data as 1s and 0s. Therefore, it is simple to calculate structured data like whole numbers and percentages. However, unstructured data must first be cleaned and prepared in the form of text and graphics before analysis.

Importance of Data Pre-processing in text analysis:

Garbage in in, garbage out is a saying that is frequently used while training machine learning models utilising data sets. This implies that if you train your model using faulty or "dirty" data, the model will be poor and inadequately trained and won't truly be useful for your research. The importance of Data pre-processing is a major criterion for a dataset which are used for text classification. The representation of the features is a critical for the Machine Learning algorithm to understand the underlying pattern.

To the extent that machine learning models trained on bad data might potentially be damaging to the analysis that we are attempting to accomplish, giving us "junk" findings.

Thus good, pre-processed data is even more vital than most of strong machine learning algorithms.

The "features" that make up a data set can be used to describe or convey the data set itself. This may depend on factors including size, location, age, time, colour, etc. Features are sometimes referred to as attributes, variables, fields, and characteristics. They appear as columns in datasets.

Various Pre-processing tasks were performed on the initial dataset. Below mentioned are the operations that were involved.

- ‘\$’ was removed from all of the columns as because it was unnecessary in this scenario.
- The column Label was converted to binary variables of 0s and 1s. The attributes were converted to 1s and the null values were replaced with 0s.
- The text in ‘Data’ column was split into individual words when an “.” Or a Camel case appeared

Unnamed: 0		Data	Label	Separated	Text_Split	word_length
0	1	\$candidates.candidateid	0	candidates.candidate.id	candidates candidate id	3
1	2	\$candidates.workspace	1	candidates.workspace	candidates workspace	2
2	3	\$candidates.teaid	0	candidates.tea.id	candidates tea id	3
3	4	\$candidates.title	0	candidates.title	candidates title	2
4	5	\$candidates.firstname	1	candidates.first.name	candidates first name	3
...
18387	18388	\$paymentmethods.check.zip	1	payment.methods.check.zip	payment methods check zip	4
18388	18389	\$paymentmethods.check.countryid	1	payment.methods.check.country.id	payment methods check country id	5
18389	18390	\$hp	0	hp	hp	1
18390	18391	\$subdistrict	1	sub.district	sub district	2
18391	18392	\$urbanvillage	0	urban.village	urban village	2

Figure 20: Modified Dataframe after data pre-processing

The column “Text_Split” is the final result after all the pre-processing tasks were performed. In the above figure 20, we can see that the whole keyword is properly converted individual words of different length.

3.5 Exploratory Data Analysis (EDA):

Conducting an EDA is an important part of the Machine Learning process. It provides a detailed study aimed to identify the underlying structure of a data set and is significant for a corporation because it exposed trends, pattern linkages that are not easily visible in the dataset.

Preferably the text data we are dealing with is massive and we need a careful examination to understand the type of data we dealing with an analytical lens. So, some text analytics have been performed on the dataset so to infer some information that can help us to understand the type of features that we are dealing with. Further it assists us to reveal the hidden patterns in the data and draw conclusions from it.

First, let's check the count of the occurrences for the text which are labelled as 0s and 1s.

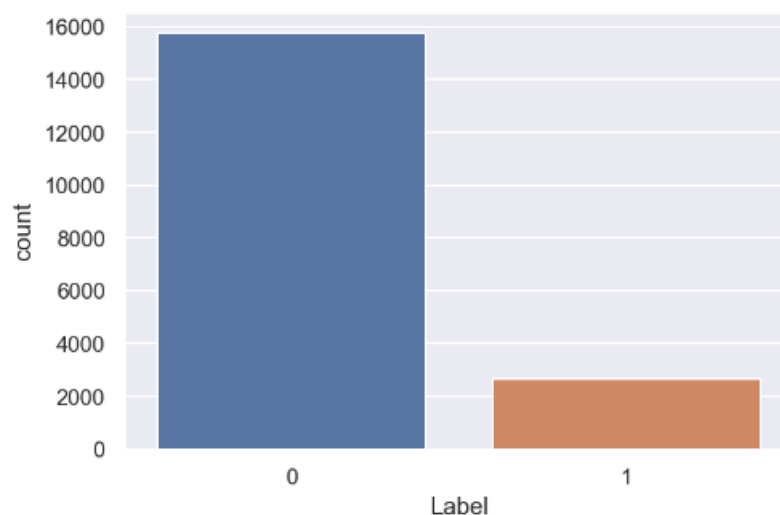


Figure 21: Total count of records grouped by label

In the preceding figure 21, it is evident that the dataset is unbalanced. There are way more non-sensitive key messages than sensitive ones. As deep learning techniques are being applied, it is not necessary to sample the dataset in this instance.

Then an additional column is created in the dataframe which stores the count of the word length for each of the record. This shows the maximum number of words that are present in the in the text.

Text_Split	word_length
candidates candidate id	3
candidates workspace	2
candidates tea id	3
candidates title	2
candidates first name	3
...	...
payment methods check zip	4
payment methods check country id	5
hp	1
sub district	2
urban village	2

Figure 22: Word count in each record

For each binary variable, the word length mean is computed and shown in the below table.

Unnamed: 0	word_length
Label	
0	9260.876326 3.868513
1	8819.009067 4.035890

Figure 23: Mean word length for each label

The mean of the labels is displayed in the figure above, indicating that the average word length of the sensitive and non-sensitive key messages is closer. After then the labels are filtered and then a plot is drawn between Frequency of the key messages to the message length.

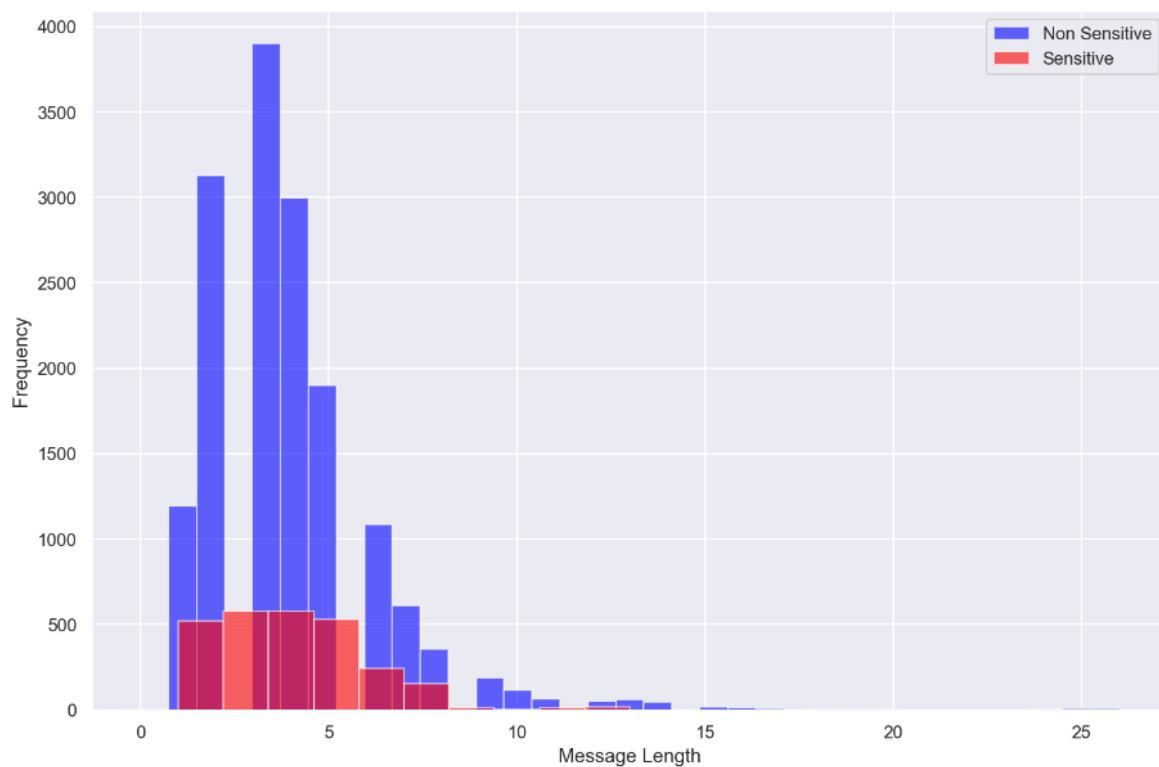


Figure 24: Frequency of occurrences vs length of the word

In the preceding figure 24, we can observe that sensitive key messages have shorter word counts than their non-sensitive counterparts. The majority of sensitive keywords vary from one to eight words in length, although the length of non-sensitive keywords varied.

Next, (Natural Language Toolkit) library NLTK has been used to tokenise the lines to show the frequency of the top 20 common occurring words in the dataset.

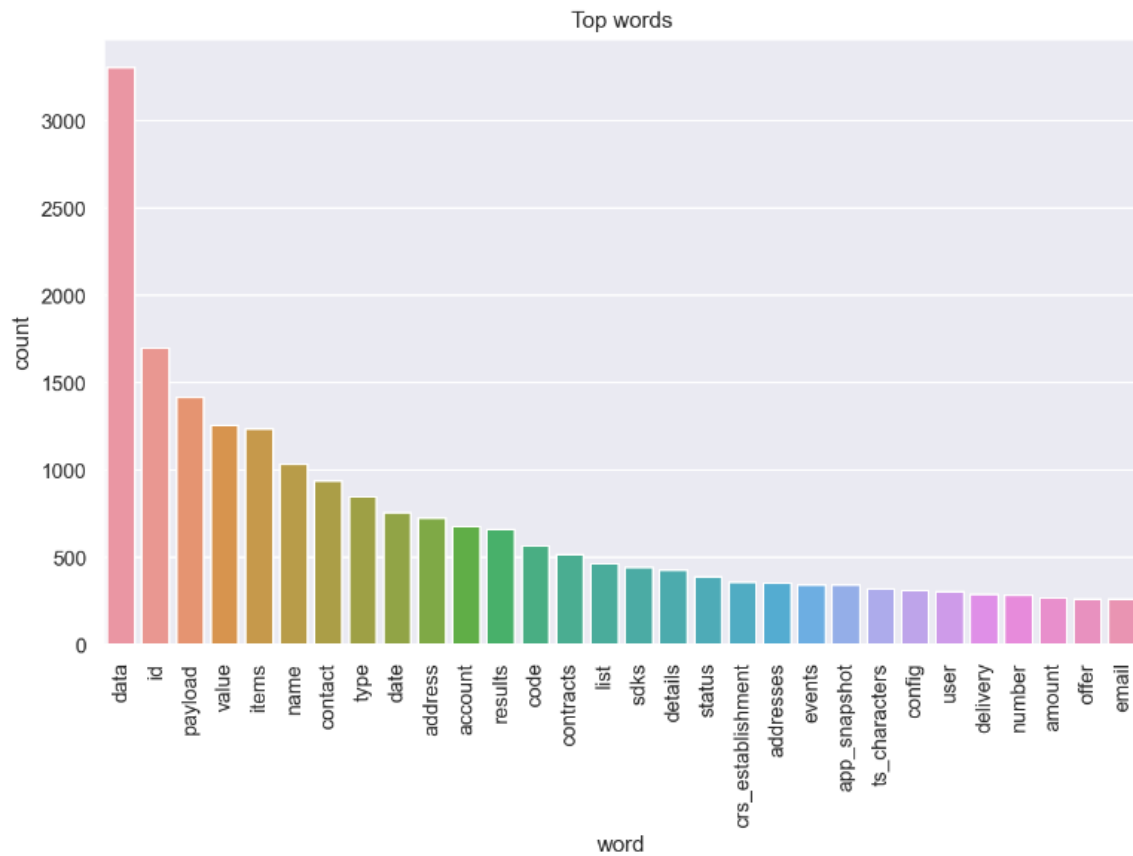


Figure 25: Top 20 most occurring words in the dataset

Next, Word clouds have been implemented for the graphical representations of the frequency of each word in a given document. It lends more prominence to the more common words, which are larger than the less frequent terms.

3.6 Tokenisation:

This is the most critical part for any text analysis phase. In order for our computer to comprehend any text, we must deconstruct the word in a form that it can comprehend. That is where the notion of tokenization in Natural Language Processing (NLP) comes in.

Tokenization is a straightforward technique that transforms raw data into a meaningful data string. In natural language processing, tokenization is used to divide paragraphs and phrases into smaller pieces that can be ascribed meaning more simply.

The initial phase of the NLP procedure is to collect the data (a sentence) and break it down into manageable chunks (words). For example, given a sentence *“Data will create a new order”*

After the tokenization is performed on the sentence which break the sentence into individual words which will look like,

“Data”, “will”, “create”, “a”, “new”, “order”

Breaking a statement into its constituent components enables a computer to comprehend the parts as well as the whole. This will assist the algorithm grasp both the individual words and their purpose within the bigger text. This is particularly significant for bigger volumes of text, since it enables the computer to determine the frequency of specific terms and the locations where they occur often. This is crucial for natural language processing's next phases.

There are some challenges during the tokenisation also. For example, we should properly clean the sentence to remove the punctuations or symbols which can cause trouble for the machines attempting to extract their meaning from a string of data. Failure to tokenize every portion of the text correctly might result in misconceptions during the NLP process.

Tokenization may be broken down into three distinct categories: word, character, and sub-word (also referred to as N-gram) tokenization. The most common type of tokenization is word tokenization.

In the project there are two types of tokenizing approaches that was performed on the dataset which are BERT and KERAS tokenizer:

1.BERT Tokenizer:

BERT is a tokenizer which incorporates its own vocabulary to tokenize the sentence into a desired format. BERT refers to (Bidirectional Encoder Representation from Transformers) [19]. Transformers are self-sufficient deep learning models that analyse their input and output data representations. BERT employs Transformer, an attention mechanism that discovers contextual relationships between words (or sub-words) in a text. In its default configuration, Transformer consists of two distinct mechanisms: an encoder that reads the text input and a decoder that generates a prediction for the task. Since the objective of BERT is to produce a language model, only the encoder mechanism is required [19].

In this context we are using Bert Tokenizer in the Transformers Library. This is a pre trained tokenizer from the Transformer library which is used to tokenize the sentence into unique tokens. BERT uses its own pre-existing library to tokenize the sentence into tokens.

There are several markers that BERT contains for tokenization that are mentioned below:

[SEP]: this is marker for the separator token

[CLS]: this token is added to the start of each sentence so that BERT knows that classification task is being performed

[PAD]: this token is used when we want to pad the sentence to a certain length. Bert checks the maximum length provided in the parameter and then adds a [PAD] token to each of the empty places.

[UNK]: BERT uses this token for the things that it is not able to tokenize, or which is not present in the library

For a Sample text “*The demon appears in the crossroads*”

BERT tokenises the sentence into this:

```
tz.tokenize('demon appears in the crossroads')  
['demon', 'appears', 'in', 'the', 'cross', '##roads']
```

Figure 27: example sentence

Then the BERT embeds the tokens and adds the specials token as per the parameter provided.

```
print(bert_transform('demon appears in the crossroads'))  
tensor([[ 101,  5725,  2691,  1107,  1103,  2771, 16038,  102,    0,    0,  
         0,    0,    0,    0,    0,    0,    0,    0,    0,    0,  
         0,    0,    0,    0,    0,    0,    0,    0,    0,    0,  
         0,    0,    0,    0,    0,    0,    0,    0,    0,    0,  
         0,    0,    0,    0,    0,    0,    0,    0,    0,    0,  
         0,    0,    0,    0]])
```

Figure 28: Sentence converted to respective tokenized ids by BERT tokenizer

In the above figure 28 we can see that Bert has added additions tokens in the starting and the end of the token and then has padded the list to the given length.

2. Keras Tokenizer:

In this implementation, the tokenizer class of Keras is used covert the text input into a integer sequence. There are several methods in this class which can be used for tokenization of a text sequence but the method we are using over here is “fit_on_texts”. Once this method is implemented on a text corpus, the object returned by this method can be used to derive more information. The attributes which can be returned are mentioned below:

- *word_count* : This returns a dictionary of words in the whole corpus along with the count.

- `word_docs` : This returns a count of documents for each of the unique words.
- `word_index` : This assigns a unique index to each of the word present in the text.
These generated unique words are used for training purpose.
- `document_count` : This returns the count of text sequences(documents) in the whole corpus

In this project implementation the ‘word_index’ attribute is used to convert each of the word into a unique integer.

For example, a list is created which contains 4 documents which is shown below in figure 29.

```
fit_text = ['Machine Learning ', 'Nueral Network',
            'Deep Learning',
            'ANN CNN LSTM']
```

Figure 29: Example list of strings

The method ‘word_index’ will return a sequence shown below in the figure 30.

```
{'learning': 1,
 'machine': 2,
 'nueral': 3,
 'network': 4,
 'deep': 5,
 'ann': 6,
 'cnn': 7,
 'lstm': 8}
```

Figure 30: Unique word with index

3.7 MODEL IMPLEMENTATION

For the model implementation the approach has been divided into two segments. In the first segment we have used conventional Machine learning model approach and in the second segment we have used Bi-directional LSTM neural network to train the data in two tokenization approach.

3.7.1 Approach 1: Implementing Machine Learning Models

In the first approach the words in each of the rows are tokenised using the BERT transformer class tokenizer. Then various machine learning models such as Decision Tree, Random Forest and Logistic Regression Classifier are implemented. The models trained on this dataset are then evaluated based on the performance metrics such as confusion matrix and classification report.

Steps involved in model implementation are discussed below:

- Creation of a new dataframe based on the BERT tokenizer:

A function is created to tokenize the documents in the corpus with the use of BERT tokenizer.

The parameters which are set for the BERT tokenizer are mentioned below in figure 31.

```
def bert_transform(sent):  
    encoded = tz.encode_plus(  
        text=sent, # the sentence to be encoded  
        add_special_tokens=True, # Add [CLS] and [SEP]  
        max_length = 64, # maximum length of a sentence  
        pad_to_max_length=True, # Add [PAD]s  
        return_attention_mask = True, # Generate the attention mask  
        return_tensors = 'pt', # ask the function to return PyTorch tensors  
    )  
    input_ids = encoded['input_ids']  
    return input_ids
```

Figure 31 : BERT Tokenizer parameters

After setting the parameter the function is ran on the whole text corpus. The result after running the function is a modified dataframe of shape 64 columns and 18390 records.

	0	1	2	3	4	5	6	7	8	9	...	54	55	56	57	58	59	60	61	62	63
0	101.0	4765	3234	25021	102	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
0	101.0	4765	1759	12204	102	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
0	101.0	4765	5679	25021	102	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
0	101.0	4765	1641	102	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
0	101.0	4765	1148	1271	102	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
...
0	101.0	7727	4069	4031	195	9717	102	0	0	0	...	0	0	0	0	0	0	0	0	0	0
0	101.0	7727	4069	4031	1583	25021	102	0	0	0	...	0	0	0	0	0	0	0	0	0	0
0	101.0	6857	102	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
0	101.0	4841	1629	102	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
0	101.0	3953	1491	102	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
18390 rows × 64 columns																					

Figure 32: Modified Dataframe using BERT

- Splitting of dataframe for training and testing purpose:

To train this dataset into different Machine Learning model we have split the dataframe using the method 'train_test_split' which is present in the library of 'sklearn'. Here the ratio with which the dataset is split is 3:1 (i.e., 75 % of the data is meant for training purpose and the rest 25% of the data for testing purpose). Stratification of dependent variable is done so that the ratio of binary values are same in train and test dataset.

- Machine Learning models for training

1. Decision Tree Classifier:

Classifiers based on decision trees produce a classification model that is potentially accurate in a variety of application situations, including text categorization. The greatest benefit of decision trees is that they simplify the interpretation and visualisation of nonlinear data patterns.

The decision tree model is built on the train dataset and then is validated against the test dataset.

2. Random Forest Classifier:

The fact that decision tree classifiers work well on smaller datasets is one of their limitations. Random Forest, on the other hand, makes accurate predictions and effectively manages enormous datasets. In most situations, the Random Forest model predicts outcomes with more precision than the decision tree method.

The random forest tree is built on the same dataset mentioned above. Here an additional parameter is included which is 'n_estimators'.

“n_estimators” : This is the number of trees you want to build before taking the maximum voting or averages of predictions. A greater number of trees improves speed but slows down the code.

Here the value of this parameter is kept as 100 for better accuracy.

3. Logistic Regression Classifier:

Logistic Regression is one of the most efficient techniques for solving classification problems. Logistic regression is easier to implement, interpret, and very efficient to train. It is very fast at classifying unknown records.

Logistic Classifier is also built on the same dataset as the other two model implemented before. The fit method has a solver parameter which is set as ‘liblinear’. We have used ‘liblinear’ solver as it applies automatic parameter selection and is recommended for dataset having high dimension which is helpful for solving large scale classification problem.

The performance of all these models is evaluated in the Chapter 4 of this report.

3.7.2 Approach 2: Implementing Neural Network based model

In the second approach, Keras tokenizer was implemented to tokenize the text corpus. This was imported from the Keras Preprocessing library. Here an object of Keras tokenizer was created which takes a parameter which sets the number of unique words. The number of unique words is set to 10000 so that all the unique words are taken in account while fitting the tokenizer.

Then the method “text_to_sequences” is fitted on the train and the test dataset which converts all the words in the text sequences to their unique integer values.

Padding:

In order to feed the training data to the neural network, the shape of all the sequences should be same. For this all the word integers sequences are padded to a same length.

Use of **BIDIRECTIONAL LSTM** for Text Classification?

Texts with variable length are always a challenge for any text classification task employing Recurrent neural networks. The RNNs specialize in processing word sequences by initially undertaking the tokenization of the words into vectors, which finally forms a matrix with two dimensions comprising of – a) Feature-vector b) Time-step dimension. The purpose is to obtain a vector with fixed length, for which most base models commonly use the 1D max pooling operation or an attention-based operation, but that works only on the time-based

dimension. Implementing the same thing for feature vector has a limitation – the features for the feature-based vector are interdependent. In this respect the workaround of application of a 1D pooling operation over the time-step dimension will degrade the topology of the feature representation. This gives rise to the need for 2-d pooling operation, which when done over both dimensions yield useful features from sequence modelling perspective. This is where the concept of Bidirectional Long Short-Term Memory Networks (BLSTM) 2-d max pooling comes in. BLSTM is first used to convert text into vectors, followed by the two-dimensional max pooling operation for obtaining a constant length vector [51].

The main purpose of LSTM was to evade the vanishing gradient problem faced by the RNNs. The actual purpose was introduction of an ‘adaptive gating mechanism’ which will decide the degree of memory retention for the previous state features for a given input dataset. For instance, LSTM operates word-by-word on a sequence $X = \{x_1, x_2, \dots, x_l\}$, with ‘l’ being the length of the input text. During the time-step t , the hidden states and the memory get refreshed with new values as shown in the equations below:

Equation 4: LSTM hidden state

$$\begin{bmatrix} i_t \\ f_t \\ o_t \\ \hat{c}_t \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{bmatrix} W \cdot [h_{t-1}, x_t]$$

$$\begin{aligned} c_t &= f_t \odot c_{t-1} + i_t \odot \hat{c}_t \\ h_t &= o_t \odot \tanh(c_t) \end{aligned}$$

Where x_t is the input at current time-step, i , is input gate activation function, f is forget gate, and o is the output gate activation function. \hat{c} denotes the current state of the cell, and σ represents the sigmoid activation function. \odot symbolizes the multiplication of all the elements.

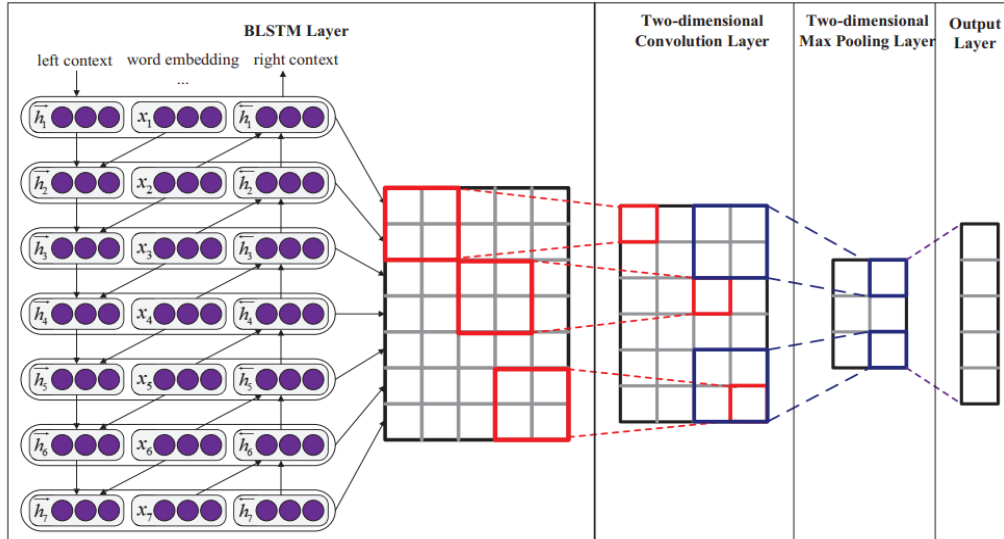


Figure 33: A BiLSTM for 7-word input sequence with word embedding and hidden units

The text classification tasks involving sequence modelling benefits from having both the previous and the upcoming context. A group of scientists called Schuster and Paliwal in 1997 [52] suggested the use of BLSTM as an extension of the traditional LSTMs, which is unidirectional in nature. This was accomplished by the introduction of another hidden layer. This helps in the inter-layer flow of information in an opposite temporal sequence, which enables the model to retain information both from the past and the future. This gives BLSTM the edge over the conventional LSTMs, as the hidden state h_t is related to other words in a sentence. Extraction of information from a matrix is simpler as it comprises of feature vectors resembling an image. This simplifies the application of 2D convolution and 2d max pooling operations on the matrix, which results in extraction of meaningful information.

This tokenizer class calculates the frequency of each individual word and arranges them in descending order. Then, each unique word is turned into an integer. This approach of tokenization and embedding is completely different from the BERT tokenizer.

Before configuring the BLSTM neural network, let's discuss about the terminologies involved in it.

1. Embedding layer:

Embedding layer is one of the essential parts of a neural network dealing with text data. This layer is available in the Keras module which is mainly used for Natural Language Processing tasks such as language modelling [34]. When addressing NLP issues, we can utilise pre-trained word embeddings like GloVe[23]. Here we have implemented Keras embedding layer to train our own embeddings.

Need for Embedding:

While dealing with the textual data we need to convert the text into numbers before feeding it into any Machine Learning Model, including the neural network models.

To understand the concept of embedding in model, words can be represented as categorical variables for convenience. To transform categorical characteristics into integers, we employ one-hot encoding. To do this, we generate dummy features for each category and fill them with 0s and 1s.

Similarly, if we employ one-hot encoding on the words in textual data, we will have a dummy feature for each word, resulting in 10,000 features for a vocabulary of 10,000 words. This embedding method is infeasible because it requires a great deal of storage space for the word vectors and decreases model efficiency.

The embedding layer enables us to turn each word into several vectors in a vector space of fixed length and size. The generated vector is dense and contains actual values as opposed to only 0s and 1s. The constant length of word vectors enables us to express words with fewer dimensions and in a more efficient manner.

The embedding layer has three arguments which are adjusted.

“input_dim” : The whole vocabulary size is taken as a input for the parameter. In this case we are setting the word count to 3316.

“output_dim”: This is the number of dimension each of the unique integer is converted into. Here the number is set to 100.

“input_length”: This is the length of input sequences which is set to the maximum length within all the text sequences.

Steps taken to handle overfitting in the model:

Overfitting occurs when the machine learning model attempts to accommodate for all or more data points than are present in the given dataset. Due to this, the model begins caching noise and erroneous values contained in the dataset, which reduces the model's efficiency and precision. An overfitted model performs well on the training data points but does not perform well when new data points are fed to it. In order to find an optimum level of fitting certain techniques are applied while the model is trained. Some of the techniques which are implemented are mentioned below.

2. Dropout layer:

A drop out layer has been added to after each of the layers. Neural networks can be optimized with the different parameters and hyperparameters. Due to this it is capable of learning extremely complex functions which also causes the model to overfit during the training of data. Thus, dropout layers are added to each layer which remove random neurons every time the information is passed through the network. This is an effective way to avoid the model from overfitting. In this model there are three dropout layers in which the drop out percentage is set as 0.2.

3. Regularization Methods:

To avoid the model from overfitting, there are several regularization techniques. The one which is used in the configuration:

Early Stopping (Regularization Technique):

The library '*Keras*' has a method called early stopping which is used to stop the training process in a neural network. The argument "monitor" allows us set the performance metric in order to end the training process.

Some of the arguments for this method are mentioned below:

"monitor": This argument allows us to set the performance metric in order to end the training process.

"mode": Based on the choice of the performance metric, the "mode" argument needs to be set as whether the object of the chosen metric is set to increase (i.e., 'max') or to decrease (i.e., 'min').

"verbose": This argument is set to 1 in order to discover the training epoch at which the training is stopped.

"patience": Often, the first indication of stagnation is not the ideal time to cease training. This is due to the fact that the model may reach a plateau of no progress or even decline somewhat before significantly improving.

This may be accounted for by adding a delay to the trigger based on the number of epochs during which no improvement is desired. This is possible by configuring the "patience" option.

4. Activation function:

A Neuron Activation Function determines whether or not a neuron should be activated. This means that simpler mathematical procedures will be used to determine whether the neuron's input to the network is significant or not throughout the process of prediction.

The purpose of Activation Function is to generate output from a node's input values (or a layer).

Use of activation function in NN, and especially Sigmoid activation function?

The concept of activation function is fairly common in the context of the recurrent neural network such as Long-short term memory. The most frequently used activation function in LSTMs is sigmoid and hyperbolic tangent, the latter more commonly known as Tanh activation function. The LSTM architecture consists of many blocks of recurrently connected units. In the recurrent networks, the issue of vanishing gradient occurs, which is the exponential decay of the slope of the error function. Selection of a proper activation function impacts the model at large along with other factors such as network structure and learning algorithm. The activation functions derive the decision boundaries and the signal strength for the input and the output activation function for a given node. Aside from affecting the network complexity, activation functions also affect the convergence of a network, and hence are paramount to a model's performance.

There are a certain set of properties that must be fulfilled by an activation function:

- a. It must be continuous and bounded
- b. It must be sigmoidal function, or else the infinity limits must fulfil certain mathematical limit constraints

From an array of activation functions, sigmoid and tanh are one of the most commonly used activation function for the LSTMs. There are more than 12 activation functions, which vary in performance for various datasets and network topology. The performance for each of the activation function is evaluated by changing the network weights and batches randomly during the course of the experiment. The error is measured once a batch ends, along with application of dropout method used to avoid model overfitting. By the end of the experimental setup, sigmoid performed significantly well, however not outperforming the others. On referring another set of experiment, sigmoid activation function performed the best architecturally. The use of this activation function improved the function results by 65%, also the training and testing values for each epoch [54].

Activation function	No. of hidden blocks	
	64	128
Aranda	2.13 (1.84–2.42)	2.10 (1.85–2.34)
Bi-sig1	2.6 (2.35–2.84)	2.00 (1.75–2.24)
Bi-sig2	2.36 (1.98–2.74)	2.03 (1.65–2.41)
Bi-tanh1	3.03 (2.74–3.32)	2.50 (2.25–2.74)
Bi-tanh2	2.13 (1.5–2.75)	1.93 (1.55–2.31)
Cloglog	2.30 (2.05–2.54)	2.26 (1.88–2.64)
Cloglogm	2.70 (2.26–3.13)	2.13 (1.75–2.51)
Elliott	2.16 (1.29–3.03)	1.66 (1.28–2.04)
Gaussian	2.16 (1.64–2.68)	1.96 (1.67–2.25)
Logarithmic	2.80 (2.14–3.45)	2.70 (2.2–3.19)
Loglog	5.46 (4.94–5.98)	8.16 (3.79–12.54)
Logsigm	3.90 (3–4.79)	4.53 (3.59–5.47)
Log-sigmoid	2.30 (1.8–2.79)	2.16 (1.59–2.74)
Modified Elliott	2.43 (1.91–2.95)	2.03 (1.74–2.32)
Rootsig	2.13 (1.98–2.27)	1.90 (1.65–2.14)
Saturated	3.43 (3.14–3.72)	3.83 (1.82–5.84)
Sech	2.33 (1.95–2.71)	2.06 (1.68–2.44)
Sigmoidalm	2.23 (1.47–2.99)	2.36 (1.74–2.99)
Sigmoidalm2	2.83 (2.31–3.35)	2.66 (2.14–3.18)
Sigt	2.66 (2.52–2.81)	2.76 (2.47–3.05)
Skewed-sig	5.43 (3.07–7.79)	4.73 (4.35–5.11)
Softsign	1.83 (1.68–1.97)	1.66 (1.52–1.81)
Wave	2.56 (2.04–3.08)	2.26 (1.88–2.64)

Figure 34: Average error values per each activation function for the MNIST data set [53]

Model Configuration:

The model is initialized as a Sequential model as the input and the output of the model is the form of text sequences.

3.7.2.1 Model 1: LSTM with solely word2vec embedding layer

The embedding layer in this model is a simple word2vec, the parameters for the same are mentioned below:

‘input_dim’ = 5050 (the number represent the whole vocabulary size)

‘output_dim’ = 100 (vector space dimension to which each individual word is converted to)

‘input_length’ = 26 (maximum length of each of the text sequence)

In this model the role of the embedding layer is to convert each of word in the vocabulary into a vector of 100 dimension. This means that each word is represented by 100 unique numbers. The benefit is doing embedding is such that if we have some words which are co-related then their vectors would be also co-related. This embedding layer defined above will provide a 3D (3 dimensional) matrix. The output of this layer will be fed to the LSTM layer as it only supports 3D tensors.

While defining the LSTM layer, the return sequence of the first two layers is set as True and the last layer is kept as False. This is done as we are using are stacking multiple LSTM layers in this configuration. Stacking LSTM hidden layers increases the depth of the model, more appropriately qualifying it as a deep learning approach.

```
Model: "sequential_1"
```

Layer (type)	Output Shape	Param #
embedding_1 (Embedding)	(None, 171, 100)	1000000
dropout_3 (Dropout)	(None, 171, 100)	0
bidirectional_3 (Bidirectional)	(None, 171, 200)	161600
dropout_4 (Dropout)	(None, 171, 200)	0
bidirectional_4 (Bidirectional)	(None, 171, 400)	643200
dropout_5 (Dropout)	(None, 171, 400)	0
bidirectional_5 (Bidirectional)	(None, 200)	401600
dense_1 (Dense)	(None, 1)	201

```

=====
Total params: 2,206,601
Trainable params: 2,206,601
Non-trainable params: 0
=====

```

Figure 35: Model 1 configuration

3.7.2.2 Model 2: LSTM with word2vec embedding layer (pretrained genism library)

Word2vec embedding method is one of the robust methods of embedding. Here we are using a pre-trained data which is available in the library 'gensim'. The 'gensim' library contains a list of pre-trained word file ('glove-wiki-gigaword-100') which is loaded. All the words present in the list are represented by vectors of dimension 100.

Another function is defined which converts each individual word in our text corpus into a 100-dimensional vector space. Every word that is present in the 'gensim' model vocabulary is represented by matrices of 100 vectors. These are already pre-trained vectors. This method then compares every unique word in our corpus with its own vocabulary and then assigns a vector of 100 dimensions to it. If a word is not present in the model, then the word is assigned with a matrix of 0. This means that the LSTM model embedding layer does not have to trained to and is set to false.

The embedding layer in this network

weights = [gensim_weight_matrix], trainable = False))

```
Model: "sequential_1"
```

Layer (type)	Output Shape	Param #
embedding_1 (Embedding)	(None, 171, 100)	1000000
dropout_3 (Dropout)	(None, 171, 100)	0
bidirectional_3 (Bidirectional)	(None, 171, 200)	161600
dropout_4 (Dropout)	(None, 171, 200)	0
bidirectional_4 (Bidirectional)	(None, 171, 400)	643200
dropout_5 (Dropout)	(None, 171, 400)	0
bidirectional_5 (Bidirectional)	(None, 200)	401600
dense_1 (Dense)	(None, 1)	201

```
=====  
Total params: 2,206,601  
Trainable params: 1,206,601  
Non-trainable params: 1,000,000  
=====
```

Figure 36: Model 2 configuration

After the model is configured, EarlyStopping is initialized with the below parameters.

```
monitor = 'val_loss'  
mode = 'min'  
verbose = 1  
patience = 5
```

Finally, the model1 and model 2 are trained with the below mentioned parameters.

```
epochs = 25,  
batch_size = 120  
validation_data=(X_test_pad, y_test),  
verbose = 1  
callbacks= [es, mc]
```

The performance of model 1 and model 2 is evaluated in the Chapter 4 of this report.

CHAPTER 5: TESTING

After the evaluation of the all the classifier models we can conclude that LSTM neural network with the word2vec embedding layer has performed the best. This was known after comparing the performance metrics of all the model the data was trained on. In this section we discuss about the data pre-processing steps and the methods that were implemented to test a new data point. In this section we are discussing about: flattening the JSON document to a proper readable format, replacing an unknown word after performing phonetic level and spelling level match and ultimately to test the final processed text in the model built.

There were two primary objectives of this project work. The first was to investigate whether or not it would be possible to utilise a machine learning or a deep learning model in order to correctly determine whether or not a JSON file contains sensitive information. The second objective was focused on handling the key values that appeared new to the built classifier. Handling this key value was important as because if the machine or deep learning model is not trained with the same word, during prediction the model would not be able to classify the input properly and would provide us inaccurate results. Thus, to test a new datapoint, handling the new words by examining the keys contained within a JSON document is the method that may be used to determine whether or not the path to a JSON file is sensitive.

Several operations were implemented on a JSON document to represent them in a manner that could be used for testing. The testing phase consists of the below mentioned operations:

1. JSON Flattening
2. Data Pre-processing
3. Tokenization
4. Phonetic and Spelling Match
5. Word Replacement
6. Testing

Testing Process flow:

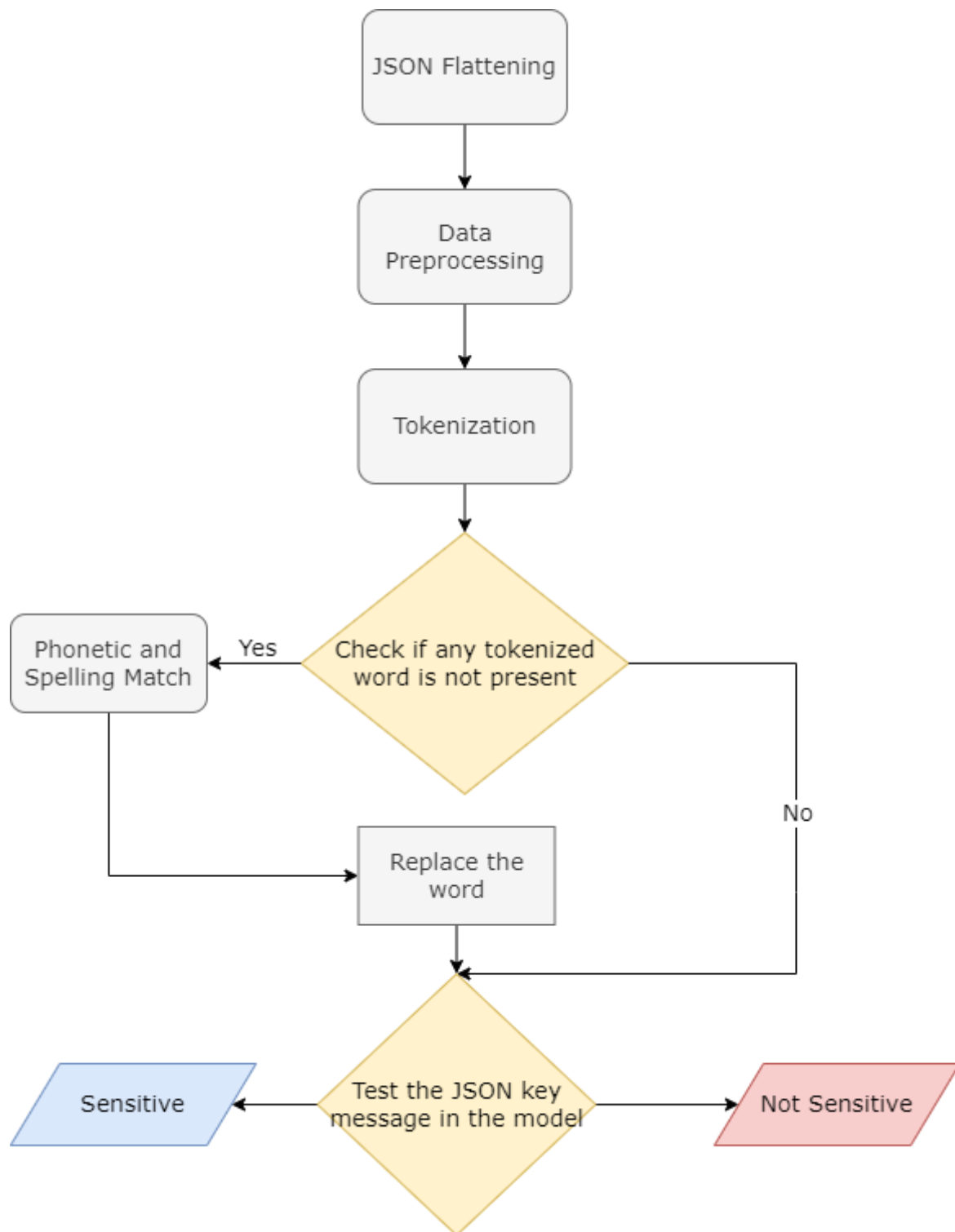


Figure 37: Testing a new datapoint

5.1 JSON Flattening:

As discussed in the above chapter 2, JSON readily has become one of the famous file formats through which the messages are being relayed within the web applications.

Below a JSON file format example is provided.

```
obj= {"employee":[  
    {"first": [{"name":"Kumar"}],  
    "p_id": "123",  
    "age":"29"  
    },  
    {"last":[{"name":"patel"}],  
    "dept_id":"345"  
    }  
    ]  
}
```

Figure 38: Example JSON file for testing

As the JSON file contains many punctuations and symbols, we have implemented a function which filters out just the key values from the JSON object. The output of the JSON file after implementing the defined method would look like:

```
['employee[0].first[0].name',  
'employee[0].p_id',  
'employee[0].age',  
'employee[1].last[0].name',  
'employee[1].dept_id']
```

Figure 39: Output after flattening the JSON file

5.2 Data Pre-processing:

After flattening the JSON file, we can see that string values obtained still have many unwanted characters that need to be removed. So, some amount of data pre-processing is required to ensure that the messages are in the right format.

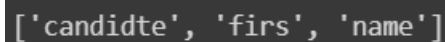
In the data pre-processing part, there were various tasks involved, some of which comprised of the following mentioned:

- Using 'regex' function to remove the numbers inside the string literals
- Replacing some characters such as symbols and punctuations
- Splitting the camel cased words (the words which have capitalized letter)
- Splitting the words and joining them to a string literal.

5.3 Tokenization:

The purpose tokenization here is to check and identify all the individual words, Keras Tokenizer was invoked to split the string into individual words.

For a string '*candidte.firs.name*' the result after performing tokenization is showed in the figure 40.



```
['candidte', 'firs', 'name']
```

Figure 40: Tokenized words

5.4 Phonetic and Spelling Match:

This is one of the important steps in this phase as it deals with the words which are unfamiliar to the model.

A new function is defined which deals with the unfamiliar words. First of all, the function checks if the tokenized word is present in the list of unique words.

If the words are present in the vocabulary of the dataset, then no replacement is performed. If the word is not present, then two levels of matching are performed which are discussed below:

5.4.1 Phonetic Level Match:

A library called “FuzzyWuzzy” is installed and utilized to perform phonetic level spelling match. It is a simple to use library of Python which can be used for string matching. It shows a score out of one hundred with the words that have the nearest matches. The score tells us how close the two strings are equal by providing a similarity index. This involves the use of Levenshtein Distance [37] to calculate the differences between the two sequences.

```
process.extract('pers', myset)

[('last_updated_by_person_id', 90),
 ('persons', 90),
 ('working_key_persisted', 90),
 ('superseding_space_booking_id', 90),
 ('on_behalf_of_person_id', 90)]
```

Figure 41: Nearest phonetic matches

The above figure 41 indicates the closest matches with the word ‘pers’. We can see that 5 results were populated with the similarity scores.

5.4.2 Spelling Level Match

Here we have utilized a built-in library which is present in Python. This is one of the interesting features in python which has various string operations such as finding the changed words, indicating the differences, getting the closest matches and string modification. In this project we have used the function “get_close_matches”, which finds the nearest matches of a word. For example, if there is a scenario where a word ‘the’ is misspelled as ‘teh’ then when the misspelled word is matched with a list of words and the closest matching word is showed.

```
difflib.get_close_matches('teh', ['the', 'they', 'thee'])

['the']
```

Figure 42: Spelling correction

5.5 Word Replacement:

If a string literal is not found in the list, then the above step (Phonetic and Spelling Match) is performed. For each of the strings the closest matches are displayed to the user and then the user is prompted to input a replacement of the word.

This portion of the procedure involves the user's manual interaction. The user must search for the word using the phonetic and spelling alternatives and replace the string with their own conscience.

```
' candidte ' string is not present in the database
The nearest phonetic matches are:
[('candidate', 94), ('i', 90), ('and', 90), ('an', 90), ('n', 90)]
The nearest spelling matches are:
['candidate', 'candidates', 'changed_date']
Enter the word to be replaced: candidate
The modified word list : ['candidate', 'firs', 'name']
' firs ' string is not present in the database
The nearest phonetic matches are:
[('firstame', 90), ('i', 90), ('ir', 90), ('first_output', 90), ('first_day_pay', 90)]
The nearest spelling matches are:
['first', 'firms', 'firm']
Enter the word to be replaced: first
The modified word list : ['candidate', 'first', 'name']
String to be input to the model: candidate first name
```

Figure 43: Replacement of unknown words in the test phase

For testing purpose, a list of individual words which contained `['candidte', 'firs', 'name']` was provided to the code snippet. In the result, we can see that the strings 'employee' and 'Id' were replaced with 'employee' and 'id'.

5.6 TEST CASES:

Finally, after performing all the steps some inputs are provided to the final decided model to predict the string if it is sensitive or not.

```
Input the sentence : candidate first name
Sensitive | 99.94%
```

Figure 44: Test Case output example 1

```
Input the sentence : department id
Non Sensitive | 99.96%
```

Figure 45: Test Case output example 2

Table 2: TEST CASES AND RESULT

INPUT	Class	Percentage (Sensitive or Non Sensitive)
<i>payment methods check state id</i>	Sensitive	78.38
<i>candidates marital status</i>	Non Sensitive	85.07
<i>candidates last name</i>	Sensitive	99.25
<i>candidates tea id</i>	Non Sensitive	99.95
<i>candidates gender</i>	Sensitive	99.78
<i>urban village</i>	Non Sensitive	98.76
<i>payment methods check zip</i>	Sensitive	99.28
<i>payment methods ach account number</i>	Non Sensitive	97.05

In the above results we can see that the model that we have built is properly able to classify if a string is sensitive or not. This also provides us with a probability percentage of how close a string is sensitive or non-sensitive.

CHAPTER 5: EVALUATION AND RESULT

In this chapter we evaluate the models we have implemented based on some performance metrics such as accuracy of the model, confusion matrix and classification report. These metrics helps us understand the model behaviour towards the dataset after it is trained on them. Finally, after comparing the scores of different models the best model is decided.

5.1 Performance indicators:

5.1.1 Confusion Matrix:

It is a 2x2 matrix that provides a summary of predictions for classification problems. The total number of accurate and inaccurate predictions are tallied and split down by class label. The primary purpose of the confusion matrix is to demonstrate how confused our classification model is during prediction.

There are 4 terms involved in a confusion matrix which are:

1. **True Positive (TP):** This is total count of the values which are Predicted True and actually are True.
2. **True Negative (TN):** This is the total count of the values which are predicted False and actually are False.
3. **False Positive (FP):** This is the total count of values which are False but predicted True.
4. **False Negative (FN):** This is the total count of values which are True but predicted False.

5.1.2 Accuracy:

The accuracy percentage tells us how accurately our model is correctly able to predict the instances.

It is calculated by the formula:

Equation 5: Accuracy formula

$$\text{Accuracy} = (\text{TP} + \text{TN}) / (\text{TP} + \text{TN} + \text{FP} + \text{FN})$$

5.1.3 Classification Report:

A classification report is used to measure quality of predictions from a classification algorithm. It displays 4 metrics which are Precision, Recall, f1-Score and support on the basis of the class. All these metrics are calculated by using True Positive, False Positive, True Negative and False Negative.

The metrics used in classification report are defined below.

1. Precision:

The precision percentage tells us that how much of our predictions were correct. Precision is defined as the ability of the classifier not to label a datapoint positive which is actually negative.

Precision can be calculated by the formula:

Equation 6: Precision formula

$$\text{Precision} = \text{TP} / (\text{TP} + \text{FP})$$

2. Recall:

The recall percentage tells us that how much of positive cases were correct. Recall is defined as the ratio of true positives to the sum of true positives and false negatives.

Recall can be calculated by the formula:

Equation 7: Recall formula

$$\text{Recall} = \text{TP} / (\text{TP} + \text{FN})$$

3. F1 score:

The f1 score provides a percentage indicating the proportion of accurate positive predictions. The F1 score is the weighted harmonic mean of accuracy and recall. The highest possible score is 1, while the lowest is 0.

F1 score can be calculated by the formula:

Equation 8: F1 score formula

$$\text{F1 score} = 2 * (\text{Recall} * \text{Precision}) / (\text{Recall} + \text{Precision})$$

4. Support:

Support is the number of real instances of a class in a given dataset. The requirement for stratified sampling or rebalancing may be indicated by unbalanced support in the training data, which may imply fundamental problems in the reported scores of the classifier.

5.2 Evaluation of Machine Learning Models:

5.2.1 Decision Tree Classifier:

- Accuracy:

The Decision tree scored an accuracy score of 0.89 when it was validated on the test data.

- Confusion Matrix:

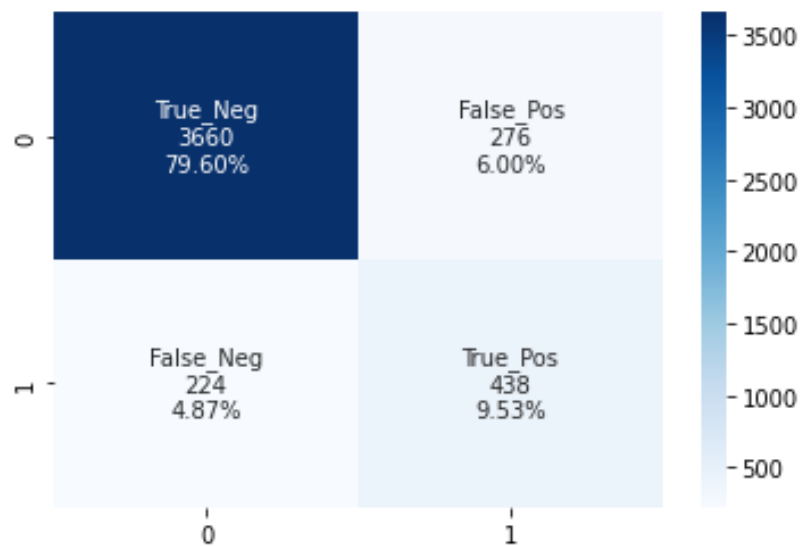


Figure 46: Confusion Matrix for Decision Tree

- Classification Report Table:

Table 3: Classification scores for Decision Tree

Class	Precision	Recall	F1-score	support
0	0.94	0.93	0.93	3936
1	0.60	0.66	0.63	662

5.2.2 Random Forest Tree Classifier:

- Accuracy:

The Random Forest tree scored an accuracy score of 0.93 when it was validated on the test data.

- Confusion Matrix:

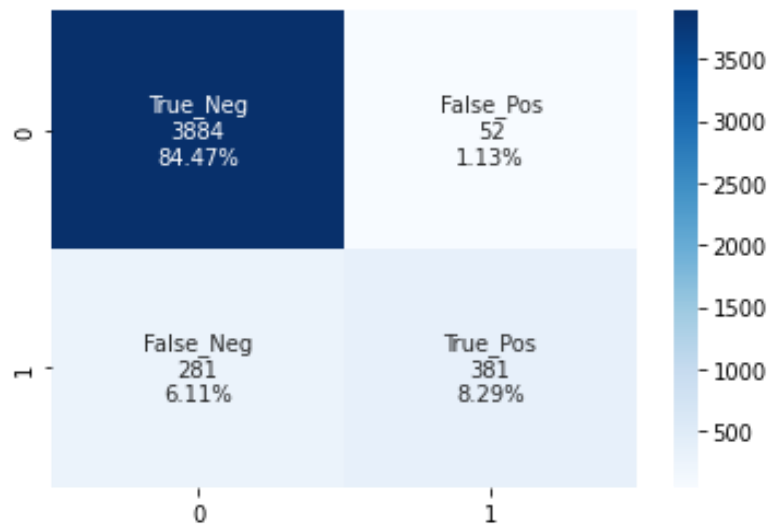


Figure 47: Confusion Matrix for Random Forest Tree

- Classification Report Table:

Table 4: Classification Table for Random Forest Tree

Class	Precision	Recall	F1-score	support
0	0.94	0.99	0.93	3961
1	0.87	0.60	0.71	637

5.2.3 Logistic Regression Classifier:

- Accuracy:

The logistic classifier model scored an accuracy score of 0.93 when it was validated on the test data.

- Confusion Matrix:

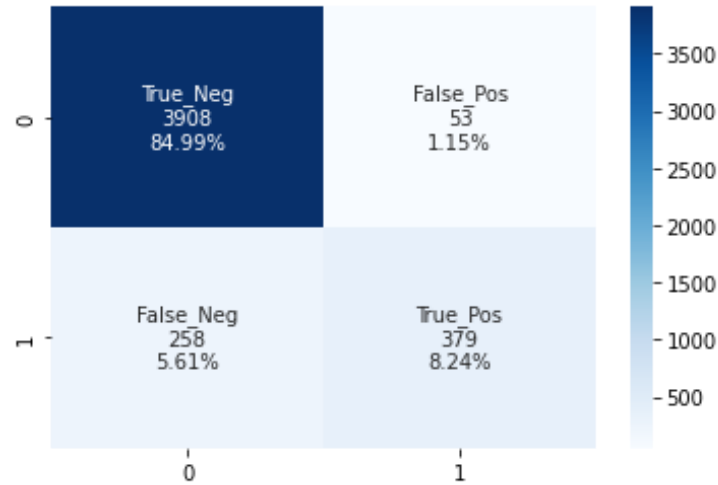


Figure 48: Confusion Matrix for Logistic Regression Classifier

- Classification Report Table:

Table 5: Classification Table for Logistic Regression Classifier

Class	Precision	Recall	F1-score	support
0	0.94	0.99	0.96	3961
1	0.88	0.59	0.71	637

In the tree-based classifiers we could see that the precision and recall scores for classifying the non-sensitive terms were fairly good. The precision score in both the Decision Tree and Random Forest Tree classifiers was 0.94 and the Recall scores were also very significant, having a scores of 0.93 in the Decision Tree and 0.99 in the Random Forest. Despite performing good in predicting non-Sensitive class these models struggled to classify the Sensitive variables as the recall scores came down to around 0.60 in both the models. Further on the implementation of the logistic classifier did not show any significant improvement over the precision and recall values for the Sensitive class.

5.3 Evaluation of Deep Neural Network using LSTM:

5.2.1 Model 1 - LSTM with solely word2vec Embedding Layer:

- Training plots

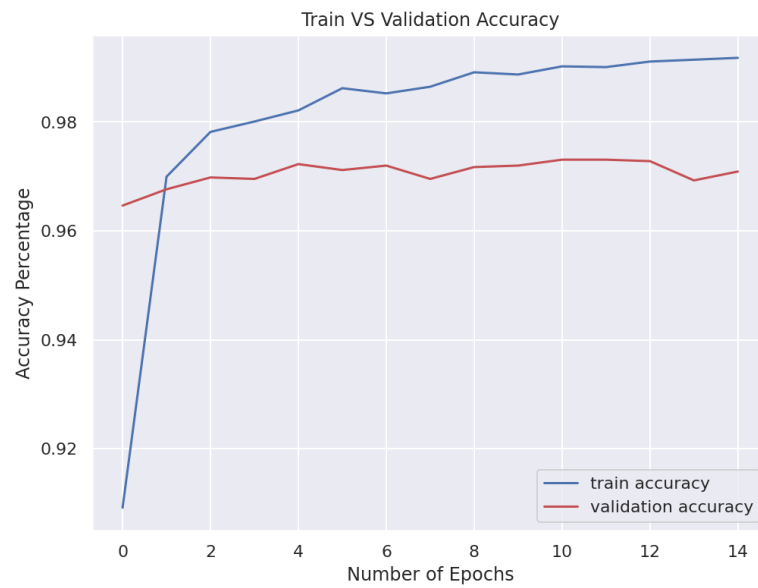


Figure 49: Train Vs Validation Accuracy

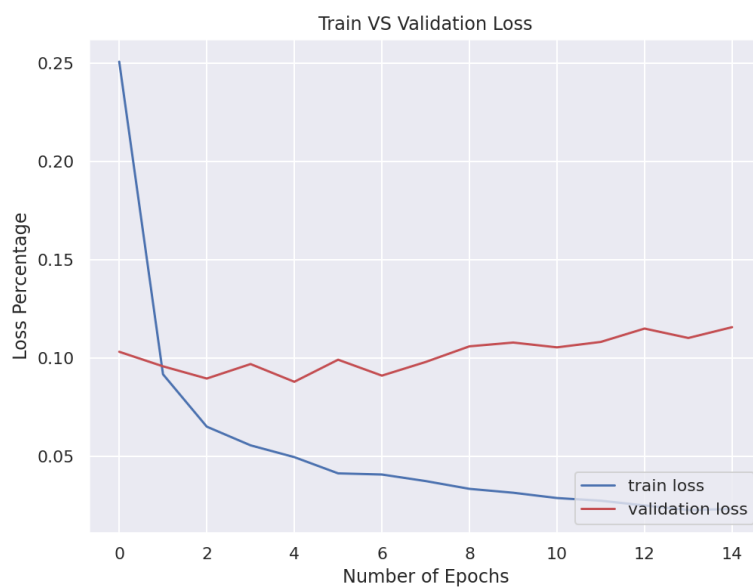


Figure 50: Train vs Validation loss

- Confusion Matrix:

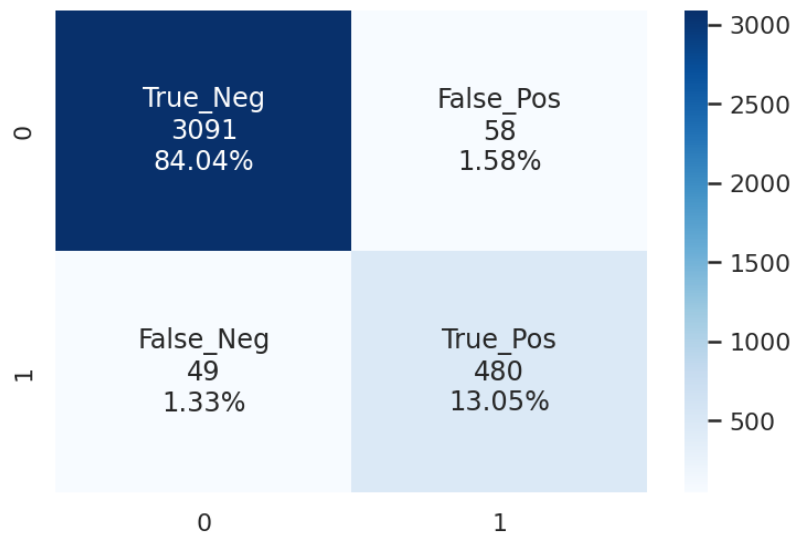


Figure 51: Confusion Matrix for LSTM model 1

- Classification Report Table:

Table 6: Classification Table for LSTM model 1

Class	Precision	Recall	F1-score	support
0	0.98	0.98	0.98	3140
1	0.91	0.89	0.90	538

5.2.2 Model 2 - LSTM with word2vec 'gensim' Embedding Layer:

- Training plots

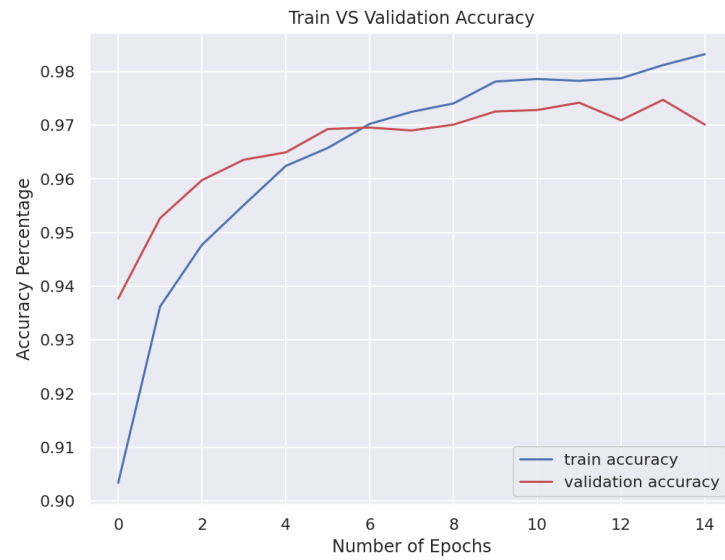


Figure 52: Train Vs Validation Accuracy

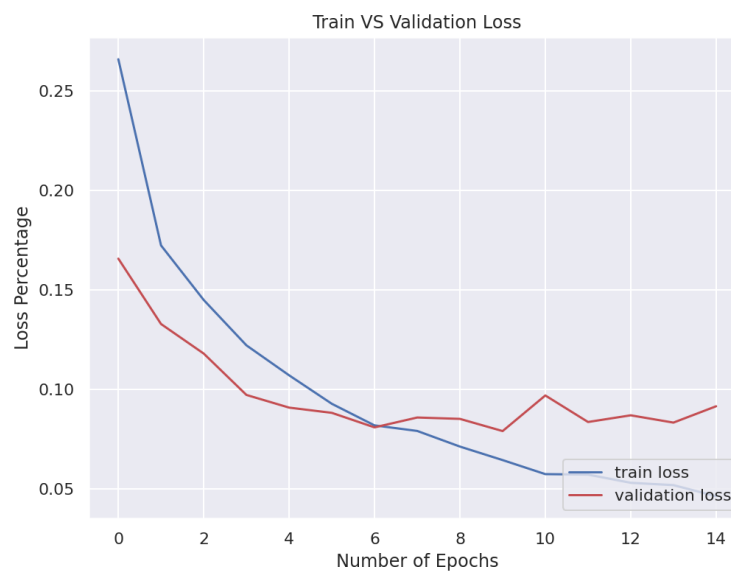


Figure 53: Train vs Validation loss

- Confusion Matrix:

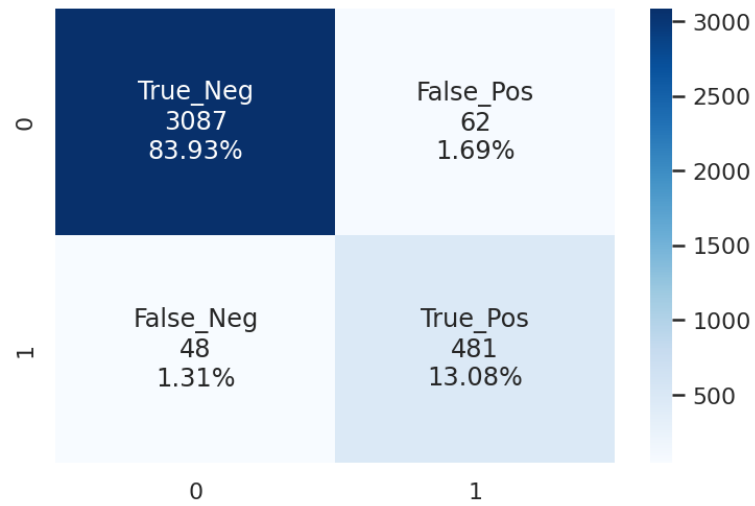


Figure 54: Confusion Matrix for LSTM Model 2

- Classification Report Table:

Table 7: Classification Table for LSTM Model 2

Class	Precision	Recall	F1-score	support
0	0.98	0.98	0.98	3135
1	0.91	0.89	0.90	543

- Training Scores for LSTM models:

Table 8: Train and Validation scores(Model 1 vs Model 2)

LSTM	Train Loss	Train Accuracy	Validation Loss	Validation Accuracy
Model 1	0.0246	0.9906	0.1224	0.9695
Model 2	0.0474	0.9833	0.0952	0.9736

As the conventional Machine learning were not able to satisfy the project requirements of predicting the Sensitive variables, NN based Bi-directional LSTM was implemented. The major advantage this model had was that they overcame the major limitation of machine learning models being trained in unbalanced dataset which affected the Recall values for detecting the Sensitive keywords. The recall values for Sensitive class in both the LSTM models significantly increased when compared with the previous models. Both model 1 and model 2 showed a Recall value of 0.89.

The validation accuracy of both the models was very promising. Model 1 had a validation accuracy of 0.9695 whereas Model 2 was 0.9736. When the training plots were compared model 1 seemed to be little overfit as the Train Accuracy was 0.9906 and Validation Accuracy 0.9695 whereas for Model 2 it was 0.9833 and 0.9736. The difference between the accuracy scores during the training and validation was very small which showed that the model had a good fit. Finally, after all the models were compared Model 2 (Bi-direction LSTM with word2vec genism) was chose for testing purpose.

CHAPTER 5: CONCLUSION, LIMITATIONS, FUTURE WORK

In this chapter, the entire research project is summarized by providing a conclusion. Further we discuss about the limitations and future work which can be extended to his project.

5.1 CONCLUSION:

In this research project, the knowledge gained from various research work and articles online were used towards solving a new problem ‘which was the identification of sensitive information using the JSON key variables. Different approaches were involved during the model implementation as the problem statement that this research work was based upon was niche. This included applying various data pre-processing tasks on the dataset. The processed data was then used to train Machine Learning and Deep Learning models in order to find an optimal performing model for new datapoints.

Machine Learning models such as Decision tree, Random Forest tree and Logistic Regression classifier proved to efficient while detecting the non-sensitive terms, but the performance of the models decreased during the prediction of sensitive classes. As the detection of the sensitive classes was of huge importance, Deep neural network based Bi-directional LSTM was utilized in order to overcome the limitations of the previous models. In this type of neural networks, word2vec embedding layer was implemented to convert each of the unique words to a vectorised representation. Implementation of this approach significantly increased the recall scores of predicting the sensitive classes.

Further a testing pipeline was created in python. This pipeline was created to test whether the model was able to classify new data points. Certain techniques such as Phonetic and Spelling matches were performed in order to replace an unfamiliar word (i.e., the word in which the model is not trained on). Finally, the processed data string was tested on the model. After several test cases it was seen that the model was successfully able to classify a new datapoint to a sensitive class or a non-sensitive class.

5.1.1 Achievements:

- The primary goal of this project was to approach the given problem statement in different ways. We were successfully able to perform various pre-processing tasks to clean the dataset and then perform different tokenization techniques to build two types of datasets.
- After comparing several performance metrics from different Machine Learning and Neural Network models we were successfully able to build a model which was accurately able to classify a JSON string.

5.2 Limitations:

- The major limitation of this project work was the size of the dataset. For neural networks dealing with Natural Language Processing tasks such as Text classification, a large amount of data is required for the model to understand the complex relationship between the input and the output instances. Thus, a large dataset would have helped to get a lower estimation variance [55] and thus have produced much better predictive performance.
- There were some cases where the arrangement of the key words played an important criterion on classifying a text sequence as sensitive or not sensitive. For machine learning or deep learning models to understand this, the data needs to be pre-processed and manually labelled. Thus, a large amount of manually tagged data was needed which was not present in the dataset.
- Due to the unbalanced classes of the presented dataset, resampling was necessary for Machine Learning models such as Decision Tree, Random Forest, and Logistic Regression classifiers. No resampling techniques, including oversampling and undersampling, were employed because oversampling would have resulted in the duplication of minority class samples in the training set, which might have led to overfitting. In addition, undersampling the dataset would have resulted in the loss of significant information for the model due to the deletion of cases from the majority class.

- Computation of Neural Networks in Jupyter Notebook, which is an offline IDE for Python Development, was computation heavy thus we had to limit the number of epochs while training the model.

5.3 Future Work:

- There are lots of Natural Language Processing techniques which could be implemented in this area of study. Some of the techniques such as Stemming and Lemmatization [56] could be used during the data pre-processing phase. Also, some method can be initialized which could generate number of variations for every unique word. Doing this would lead increase the number of records in the dataset and help the model during the training phase.
- Some of the key variables when arranged in different order changes the label of the class. For example, if a text string consisted of keywords such as 'Employee' and 'Name', this would have been classified as sensitive as per the labelled data whereas 'Department' and 'Name' when put together comes up as a non-sensitive class. To do this a lot of manually tagged data was required for the machine learning model to understand this sort of relationship.
- In such cases tokenisation techniques such as N-Gram [57] tokenization could be implemented to generate variations in the order of text strings.
- Further a machine learning model could have been built which would have been able to determine the category of the sensitive key variable. Such that whether it represented a name, date of birth, address, location and other personal identifiable information. This would provide a more detailed information about the type of sensitive key variable.
- Certain data pre-processing techniques can be implemented to detect a key variable which is not known to the model. For example, a trained model knows that when the keywords such as 'personal' and 'id' appear together, it can be classified as a sensitive information but when keywords such as 'pers' and 'id' together, the model

that is built right now would not have been capable to classify it as Sensitive label. The machine does not know that the word ‘pers’ is same as ‘personal’ and would require a manual intervention because the model is not trained with such kind of data.

- Transfer learning can be implemented in this model which can be highly useful for such kinds of work. Transfer learning is a technique for machine learning in which a model created for one task is utilised as the basis for a model for a second task. It is a common practise in deep learning to use pre-trained models as a starting point for computer vision and natural language processing tasks, due to the vast computational and time resources required to develop neural network models for these problems and the enormous skill gains that these models provide on related problems.

References

- [1] Warren, Samuel; Brandeis, Louis (December 15, 1890). "The Right to Privacy". *Harvard Law Review*. **IV** (5): 193–220. Retrieved 4 June 2021 – via Internet Archive.
- [2] Warren & Brandeis, paragraph 1.
- [3] Jan Holvast, *The Future of Identity in the Information Society*, 2009, Volume 298
- [4] <https://www.truqcapp.com/digitization-vs-digitalization-differences-definitions-and-examples/>
- [5] <https://www.boldare.com/blog/5-examples-of-digital-transformation/#:~:text=Digitization%20refers%20to%20converting%20analog,format%20on%20a%20company's%20drive.>
- [6] <https://www.techrepublic.com/article/data-privacy-is-a-growing-concern-for-more-consumers/#:~:text=A%20full%20of%20the,even%20trust%20their%20own%20employers.>
- [7] Moore Jr., B.: *Studies in Social and Cultural History*. M.E. Sharpe, Inc., Armonk (1984)
- [8] https://advisory.kpmg.us/articles/2021/bridging-the-trust-chasm.html?utm_source=vanity&utm_medium=referral&utm_mid=m-00005652&utm_campaign=c-00107353&utm_cid=c-00107353
- [9] List of Cyberattacks https://en.wikipedia.org/wiki/List_of_cyberattacks
- [10] Wheatley, S., Maillart, T. and Sornette, D., 2016. The extreme risk of personal data breaches and the erosion of privacy. *The European Physical Journal B*, 89(1).
- [11] Utzerath, J., Dennis, R. Numbers and statistics: data and cyber breaches under the General Data Protection Regulation. *Int. Cybersecur. Law Rev.* **2**, 339–348 (2021). <https://doi.org/10.1365/s43439-021-00041-8>
- [12] <https://www.gov.uk/data-protection#:~:text=The%20Data%20Protection%20Act%202018%20is%20the%20UK's%20implementation%20of,used%20fairly%2C%20lawfully%20and%20transparently>
- [13] <https://www.techopedia.com/definition/25013/data-perturbation>
- [14] https://en.wikipedia.org/wiki/Data_anonymization
- [15] <https://en.wikipedia.org/wiki/Cryptography>
- [16] S. Murthy, A. Abu Bakar, F. Abdul Rahim and R. Ramli, "A Comparative Study of Data Anonymization Techniques," *2019 IEEE 5th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing, (HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS)*, 2019, pp. 306-309, doi: 10.1109/BigDataSecurity-HP SC-IDS.2019.00063.
- [17] Hassan, Hassan, F., Sánchez, D., Soria-Comas, J. and Domingo-Ferrer, J., 2019, August. Automatic anonymization of textual documents: detecting sensitive information via word embeddings. In *2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)* (pp. 358-365). IEEE.
- [18] https://en.wikipedia.org/wiki/Named-entity_recognition
- [19] Devlin, J., Chang, M., Lee, K. and Toutanova, K., 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *Google AI Language*,.

- [20] Kowsari, K.; Jafari Meimandi, K.; Heidarysafa, M.; Mendu, S.; Barnes, L.; Brown, D. Text Classification Algorithms: A Survey. *Information* **2019**, *10*, 150. <https://doi.org/10.3390/info10040150>
- [21] Salton, G.; Buckley, C. Term-weighting approaches in automatic text retrieval. *Inf. Process. Manag.* **1988**, *24*, 513–523.
- [22] Goldberg, Y.; Levy, O. Word2vec explained: Deriving mikolov et al.'s negative-sampling word-embedding method. *arXiv* **2014**, arXiv:1402.3722.
- [23] Pennington, J.; Socher, R.; Manning, C.D. Glove: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar, 25–29 October 2014; Volume 14, pp. 1532–1543.
- [24] Aizawa, A., 2003. An information-theoretic perspective of tf–idf measures. *Information Processing & Management*, *39*(1), pp.45-65.
- [25] Basha, S.R. and Rani, J.K., 2019. A comparative approach of dimensionality reduction techniques in text classification. *Engineering, Technology & Applied Science Research*, *9*(6), pp.4974-4979.
- [27] LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436–444
- [28] Wu, T.F.; Lin, C.J.; Weng, R.C. Probability estimates for multi-class classification by pairwise coupling. *J. Mach. Learn. Res.* **2004**, *5*, 975–1005.
- [29] Huang, K. Unconstrained Smartphone Sensing and Empirical Study for Sleep Monitoring and Self-Management. Ph.D. Thesis, University of Massachusetts Lowell, Lowell, MA, USA, 2015.
- [30] Giovanelli, C.; Liu, X.; Sierla, S.; Vyatkin, V.; Ichise, R. Towards an aggregator that exploits big data to bid on frequency containment reserve market. In *Proceedings of the 43rd Annual Conference of the IEEE Industrial Electronics Society (IECON 2017)*, Beijing, China, 29 October–1 November 2017; pp. 7514–7519.
- [31] Quinlan, J.R. Simplifying decision trees. *Int. J. Man-Mach. Stud.* **1987**, *27*, 221–234
- [32] Jasim, D.S. Data Mining Approach and Its Application to Dresses Sales Recommendation. Available online: https://www.researchgate.net/profile/Dalia_Jasim/publication/293464737_main_steps_for_doing_data_mining_project_using_weka/links/56b8782008ae44bb330d2583/main-steps-for-doing-datamining-project-using-weka.pdf
- [33] Bansal, H.; Shrivastava, G.; Nhu, N.; Stanciu, L. *Social Network Analytics for Contemporary Business Organizations*; IGI Global: Hershey, PA, USA, 2018.
- [34] Arisoy, E., Sainath, T.N., Kingsbury, B. and Ramabhadran, B., 2012, June. Deep neural network language models. In *Proceedings of the NAACL-HLT 2012 Workshop: Will We Ever Really Replace the N-gram Model? On the Future of Language Modeling for HLT* (pp. 20-28).
- [35] Hochreiter, S. (1998). The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, *06*(02), pp.107–116. doi:10.1142/s0218488598000094.
- [36] Hanin, B., 2018. Which neural net architectures give rise to exploding and vanishing gradients?. *Advances in neural information processing systems*, *31*.
- [37] Yujian, L. and Bo, L., 2007. A normalized Levenshtein distance metric. *IEEE transactions on pattern analysis and machine intelligence*, *29*(6), pp.1091-1095.
- [38] <https://www.service-architecture.com/articles/web-services/javascript-object-notation-json.html>

- [39] <https://gackarur.com/econtents/pg/english/research%20iv.pdf>
- [40] Hakim, A.A., Erwin, A., Eng, K.I., Galinium, M. and Muliady, W., 2014, October. Automated document classification for news article in Bahasa Indonesia based on term frequency inverse document frequency (TF-IDF) approach. In 2014 6th international conference on information technology and electrical engineering (ICITEE) (pp. 1-4). IEEE.
- [41] Lebrete, R.P., 2016. *Word embeddings for natural language processing* (No. THESIS). EPFL.
- [42] Dreiseitl, S. and Ohno-Machado, L., 2002. Logistic regression and artificial neural network classification models: a methodology review. *Journal of biomedical informatics*, 35(5-6), pp.352-359.
- [43] <https://towardsdatascience.com/activation-functions-b63185778794>
- [44] <https://towardsdatascience.com/decision-tree-classifier-explained-in-real-life-picking-a-vacation-destination-6226b2b60575>
- [45] <https://towardsdatascience.com/a-laymans-guide-to-deep-neural-networks-ddcea24847fb>
- [46] Wang, S.C., 2003. Artificial neural network. In *Interdisciplinary computing in java programming* (pp. 81-100). Springer, Boston, MA.
- [47] Mikolov, T., Karafiát, M., Burget, L., Cernocký, J. and Khudanpur, S., 2010, September. Recurrent neural network based language model. In *Interspeech* (Vol. 2, No. 3, pp. 1045-1048)
- [48] <https://www.analyticsvidhya.com/blog/2021/06/lstm-for-text-classification/>
- [49] <https://www.analyticsvidhya.com/blog/2022/03/an-overview-on-long-short-term-memory-lstm>
- [50] Kim, H., Kim, H. and Cho, S., 2017. Bag-of-concepts: Comprehending document representation through clustering words in distributed representation. *Neurocomputing*, 266, pp.336-352.
- [51] Zhou, P., Qi, Z., Zheng, S., Xu, J., Bao, H. and Xu, B., 2016. Text classification improved by integrating bidirectional LSTM with two-dimensional max pooling. *arXiv preprint arXiv:1611.06639*.
- [52] Schuster and Paliwal1997] Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *Signal Processing, IEEE Transactions on*, 45(11):2673–2681
- [53] Farzad, A., Mashayekhi, H. and Hassanpour, H., 2019. A comparative performance analysis of different activation functions in LSTM networks for classification. *Neural Computing and Applications*, 31(7), pp.2507-2521
- [54] Pratiwi, H. κ.ά. (2020) ‘Sigmoid Activation Function in Selecting the Best Model of Artificial Neural Networks’, *Journal of Physics: Conference Series*. IOP Publishing, 1471(1), σ. 012010. doi: 10.1088/1742-6596/1471/1/012010.
- [55] Wolter, K.M. and Wolter, K.M., 2007. *Introduction to variance estimation* (Vol. 53). New York: Springer.
- [56] Balakrishnan, V. and Lloyd-Yemoh, E., 2014. Stemming and lemmatization: A comparison of retrieval performances.
- [57] Cavnar, W.B. and Trenkle, J.M., 1994, April. N-gram-based text categorization. In *Proceedings of SDAIR-94, 3rd annual symposium on document analysis and information retrieval* (Vol. 161175).