Czech University of Life Sciences Prague

Faculty of Economics and Management

Department of Information Technologies

Master's Thesis

Developing an Email Classification algorithm

Ali Khalil

CZECH UNIVERSITY OF LIFE SCIENCES PRAGUE

Faculty of Economics and Management

DIPLOMA THESIS ASSIGNMENT

Ali Khalil, Dipl. Ing.

Systems Engineering and Informatics
Informatics

Thesis title

Developing an Email Classification algorithm.

Objectives of thesis

NLP is a branch of Artificial Intelligence (AI) that enables machines to understand the human language and deal with texts and automatically perform tasks like translation, spell check, or topic classification.

The goal of this study is to create an algorithm in the NLP branch that can understand the text inside emails and decide which categories they should belong to.

Methodology

1-Collect a big database (more than 4000 emails) to use in training and testing.

2-Text Processing.

1-2-Tokenization.

2-2-Clean the data:

1-2-A-Remove HTML tags

2-2-B-Remove extra whitespaces

3-2-C-Convert accented characters to ASCII characters.

4-2-D-Remove special characters.

5-2-E-Remove numbers.

6-2-F-Remove stopwords.

3-2-Lowercase all texts.

4-2-Normalization:

A-Stemming.

B-Lemmatization.

3-Using many Machine learning classification algorithms to choose the best accuracy.

4-Evaluation.

5-To demonstrate the algorithm's power, I will apply it to a real-world problem and compare the results with the results of the previous evaluation

The proposed extent of the thesis

60 pages

Keywords

machine learning, conventional neural network, deep learning,Stemming, Lemmatization,tokenization

---

Recommended information sources

GÉRON, A. *Hands-on machine learning with Scikit-Learn and TensorFlow : concepts, tools, and techniques to build intelligent systems.* Beijing ; Boston ; Farnham ; Sevastopol ; Tokyo: O'Reilly, 2019. ISBN 978-1-492-03264-9.

CHOLLET, F. – ALLAIRE, J J. *Deep learning with R.* Shelter Island, NY: Manning Publications Co., 2018. ISBN 9781617295546.

Expected date of thesis defence

2021/22 WS – FEM

The Diploma Thesis Supervisor

doc. Ing. Arnošt Veselý, CSc.

Supervising department

Department of Information Engineering

Electronic approval: 1. 11. 2021

Ing. Martin Pelikán, Ph.D.

Head of department

Electronic approval: 23. 11. 2021

Ing. Martin Pelikán, Ph.D.

Dean

Prague on 21. 08. 2022

Declaration

I declare that I have worked on my master's thesis titled "Developing an Email Classification algorithm" by myself and I have used only the sources mentioned at the end of the thesis. As the author of the master's thesis, I declare that the thesis does not break any copyrights.

In Prague on 29/11/2022 _____

Acknowledgment

I would like to thank doc. Ing. Arnošt Veselý, CSc, for giving me the opportunity to work on this project and for providing valuable guidance and feedback.
I am grateful for the support I received from the Ministry of interior.
To my big inspiration my father Taher (R.I.P).
Thanks to my mother Afaf, my sisters(Lina, and Layal), my brother (Ibrahim), my brother-in-law-(Hayan), and my nephews (Mohammad, and Miral) for your motivation.
Finally, I'd like to express my gratitude to everyone who supported me (Kamila, Youshaa, Mehyar, Zein, Ammar, Waseem, Osama, and Ali).

Developing an Email Classification Algorithm

Abstract:

Artificial intelligence and machine learning have shown the ability to automate decision-making and improve different aspects of human lives. This thesis uses the power of machine learning to build classification models that can be used to recognize the topic of an email based on its context. The purpose is to have a model that can be deployed in companies and used to correctly detect the topic or category of each email and help the employees to prioritize their email reading based on the topic. Another application may be to direct an email to the correct/suitable person or department depending on its topic.

Bearing the above in mind, this thesis is composed of two main parts. The first part is a theoretical part where an overview of the basics of artificial intelligence, machine learning, natural language processing, and artificial neural networks is presented. The second part is a practical part that shows the practical steps followed in order to build the email classification model. The practical part itself includes two sequential steps. The first step is data preprocessing within which the emails are cleaned and converted into a numerical form making them suitable for the training of machine learning mathematical models. The second step is, of course, the training and testing of the machine learning models.

For comparison purposes, four machine learning models have been trained. Those models are Logistic Regression, Random Forest, and Naive Bayes in addition to Artificial Neural Networks. The results confirm the applicability of the email's classification concept and also affirm the linear nature of the classification problem in scope based on the high prediction accuracy reached by the Linear Logistic Regression. As a result of the performed analysis, the Logistic Regression and Artificial Neural Networks have shown the highest performance among the four compared algorithms.

Keywords: Artificial Intelligence, Machine Learning Algorithms, Artificial Neural Network, Natural Language Processing, Activation Functions.

Contents

1 Introduction:

Electronic email is a method to send and receive information through electronic communication systems, whether it is the internet or communication networks within companies or home institutions. Email allows users to send files, images, links, or texts. Companies use emails to communicate with employees and clients.

As a result of the development and the massive amounts of data sent via the Internet, classifying emails into semantic groups has become a necessity. The classification system of emails saves time and reduces effort, and efficiency is increased. For example, in G-mail the emails are categorized as Primary, Social, Promotions, Updates, and forums.

Natural Language Processing (NLP) is a subfield of artificial intelligence, NLP centers on processing and analyzing text into a form that helps machine learning models to understand and extract information from it and use this information for the desired output.

2 Objectives and Methodology:

2.1 Objectives

The goal of this thesis is to build a machine-learning model that is able to classify emails and predict which category each email belongs to. The aim is to exploit Natural Language Processing techniques to analyze and understand the language of the emails and then, to train several machine learning algorithms (i.e., Logistic Regression, Naive Bayes, Random Forest) to predict the category of each email. In addition to the previous algorithms, the advanced Artificial Neural Network (ANN) is also used to build a non-linear model for the same purpose, i.e., the prediction of each email's category. The use of ANNs is motivated by their ability to extract more complex and non-linear models between the inputs (the emails) and the output (the corresponding category). The various trained models are tested on a test set of emails and then, compared using the test set prediction accuracy as the model's quality criteria.

2.2 Methodology:

The thesis includes two parts: theoretical and practical. The theoretical part is an overview of artificial intelligence and applications of artificial intelligence as well as natural language processing. Moreover, the theoretical part covers the basic principles and applications of machine learning algorithms, Artificial neural networks, and Deep Learning algorithms.
The practical part, on the other hand, includes the detailed processes followed to fulfill the objectives of this thesis. These processes include the data collection, and its pre-processing steps as follows:

- Data splitting
- Tokenization
- Data cleaning
- Lemmatization
- Term frequency-inverse document frequency(TF-IDF)

The pre-processing steps are followed by training the following classification models:

- Logistic Regression
- Naive Bayes
- Random Forest
- ANN

The last step includes the comparison between the trained models in terms of the reached classification accuracy on the test set. In addition to the description of the results, the conclusion of this thesis is provided.

3 Literature Review:

3.1 Artificial Intelligence:

3.1.1 History of AI:

Different ideas about humanoid robots have been implemented. Daedalus is an example of this, who attempts to create artificial humans when he ruled the methodology of the wind. In 1884 Charles Babbage began working on a mechanical machine capable of showing intelligent behavior, but he concluded that he would be unable to create a machine with the same intelligence as a human being. In 1950 Alan Turing asked one of the most important questions in history "Can a machine think", and he did a test to check the intelligence of computers, and the results of the test were deemed convincing. Then in 1956 a conference on artificial intelligence was at Dartmouth college for the first time in this period the first artificial intelligence applications were created, and these applications are based on logic theorem and chess games, the programs created at this time were distinct from the geometric forms used in intelligence tests, leading to the notion that intelligence computers can be created. (Mijwel, 2015)

The most important Milestones of AI:

Artificial Intelligence isn't a novel notion, it has been for a long time, there have been several important milestones that have aided in the advancement of artificial intelligence capabilities. According to (Mijwel, 2015) between 1920 and 1950

1923: The concept of the robot was presented in the theater play Rossum's Universal Robots.

1927: Featured artificial intelligent robot in a science fiction film (Metropolis).

1943: The concept that logical functions could be fulfilled through networks of artificial neurons emerged because of collaboration between Warren McCulloch and Walter Pitts

Then after the 40's, there was a rapid development in the theory of AI. According to (Marr, 2021) :

1950: a collection of fictional stories called I Robot was published by Isacc Asimov. and Asimov indicated the three laws of Robotics, as well as a computer that could answer questions due to its capability to store human knowledge.

1950: Alan Turing represented the concept of the Turing test.

1956: The first conference about the term artificial intelligence was organized at Dartmouth College by Professor John McCarthy.

1966: The world's first text chatbot (Eliza) was created by Joseph Weizenbaum at MIT.

1985: The companies started investing huge money in the field of AI, because from 1980 to 1986: The XCON expert learning system from digital equipment cooperation was credited with saving the company 40 million per year.

1988: The first use of AI strategy of the probability of different outcomes by preparing the data and processing it then training the machines on these prepared data is marked in the paper "A Statistical Approach to Language Translation"

1997: IBM designed a chess-playing supercomputer [Deep Blue] that defeated the international grandmaster chess player Garry Kasparov.

2005: In the race of 100 kilometers of road in the Mojave desert, 5 autonomous vehicles accomplished the race, whereas in 2004 no autonomous vehicle did it.

2011: Apple company designed the voice-controlled virtual assistant.

2015: The accuracy of image recognition reached 97,3% in the contest of ImageNet challenge Where it was 73% in 2014.

In the last four years, AI has been developing faster than expected, as it was mentioned in (Milestones, 2022) :

2018: Groove X created Lovot which is the first emotional robot, which can sense human change moods and act according to these changes.

2019: An autonomous battle tank Ripsaw M5 unveiled at the army expo in Washington.

2021: The first AI system to detect potential COVID-19 patients within one hour was designed by Oxford University. This curial AI can distinguish between other respiratory patients with more than 90 % accuracy.

Waymo self-car driving is open to the public. Users can call a taxi to their location by downloading this app car on their smartphones. In the previous, safety personal would accompany self-driving cars, but this was the first time a car was driven without any human but a passenger.

3.1.2 Subfields of Artificial Intelligence:

According to (AI, 2022) the subfields of AI are :

- Natural language processing.
- Machine learning.
- Neural Networks.
- Deep learning.
- Cognitive computing.
- Computer vision.

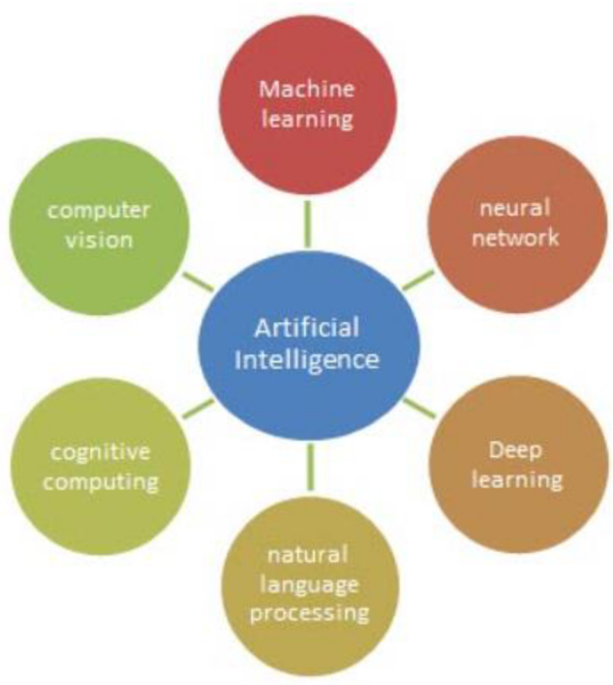


Figure 1: Sub-fields of Artificial Intelligence

(Source: www.softwaretestinghelp.com/what-is-artificial-intelligence)

3.2 Machine Learning:

3.2.1 Introduction:

Machine learning is a subset of artificial intelligence, which aims to study and create programs that learn patterns over time when it's exposed to additional data rather than being explicitly programmed. It learns from observation data. Feeding the machine learning algorithm with more data enables the algorithm to better learn the underlying pattern (Géron, 2019)

Utilizing machine learning programs has increased in recent years. Determining if the email is spam or not spam, an automatic recommendation of which movies to watch, and a recommendation of which order to buy are examples of a machine learning program. Websites like Netflix, and Facebook have machine-learning algorithms (Müller & Guido, 2016)

3.2.2 Machine Learning Systems:

There are different systems of machine learning, these systems are categorized according to special cases:

- If the system is trained or not with human supervision (supervised, unsupervised, semi-supervised, reinforcement learning).
- If the system can or can't learn incrementally on the fly (online, batch learning).
- Whether they create predictive models by identifying patterns in the training data (model-based learning) or simply comparing new data points to known data points (instance-based versus).
- It could combine some of these systems when needed, spam filter is an example of a state that learns on the fly using deep neural networks, The model trained on spam and ham. This makes it an online, model-based, supervised learning system.

The basic categories are Supervised, Unsupervised, Semi-Supervised, and Reinforcement learning. (Géron, 2019)

3.2.3 Supervised Learning:

In this type of machine learning the training data which feeds the algorithm is labeled. There are two types of supervised learning:

3.2.3.1 Classification:

Classification aims to predict a class label (category), which is a choice from a predefined list of categories. Classification is divided into binary classification and multiclass classification. Binary classification is the process of differentiating between two classes, whereas multiclass classification differentiates between more than two classes. An example of binary classification is spam or not spam email, whereas classifying email into more than two categories is an example of multiclass classification. (Müller & Guido, 2016).
According to (Gong, 2022), There are several machine learning models for classification problems:

- Logistic Regression.
- K-Nearest Neighbors(K-NN).
- Support Vector Machines(SVM).
- Naive Bayes.
- Decision Tree Classification.
- Random Forest Classification

Logistic Regression:

Logistic Regression is a classification algorithm that attempts to learn a function that is close to P(Y/X). Its central assumption is that P(Y/X) can be approximated as a sigmoid function applied to a linear combination of input features.

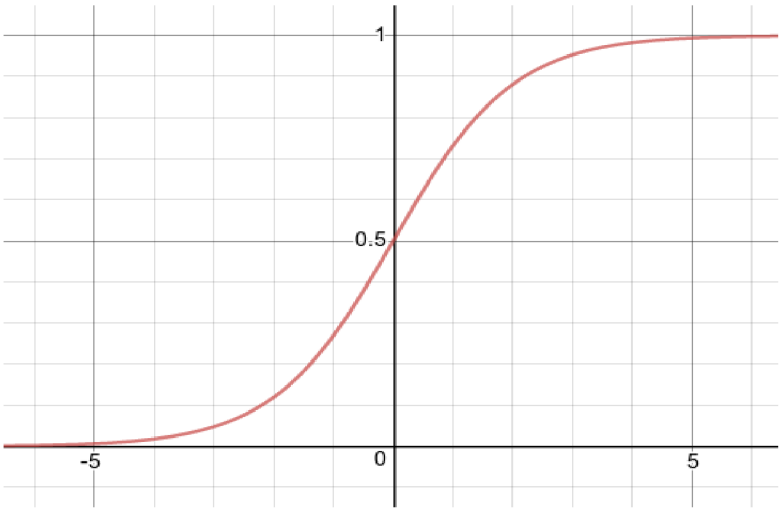


Figure 2: Sigmoid Function

(Source: https://pdfcoffee.com/220-logistic-regression-pdf-free.html)

This is a sigmoid function, its equation is :

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

The sigmoid function converts real numbers to a range between [0,1]. In Logistic regression, σ(z) converts an arbitrary value "z" into a number between [0,1], which is known as probability. Positive numbers of (z) refer to high probabilities, whereas negative ones refer to low probabilities.

The importance of Logistic Regression is because it is considered the major building block of artificial neural networks. Logistic Regression is used to predict if a new observation relates to one of two possible classes such as email filter to spam or not spam, and also predicts if the tumor is benign or malignant. (Monroe, 2017)

Naive Bayes:

The Naive Bayes classifier is a machine-learning model that uses the Bayes theorem, for probabilistic classification. And it can classify multiple classes directly.

The major idea of Naive Bayes is that each feature in the class (input data) is independent of another feature in the same class. An example of the process of Naive Bayes calculation is if the dependent feature is a watermelon fruit and the independent features are green, round, and the diameter is about 15 cm. The Naive Bayes considers that if the watermelon is green, the probability is 40% if the watermelon is round the probability is 60% if the diameter is close to 15 the probability is 95 %. It is called "Naive" because independent features contribute to decision-making.

Naive Bayes theorem is :

$$p(C_j|x) = \frac{p(C_j)p(x|C_j)}{p(x)} \qquad (2)$$

C_j: is the number of classes

x: is the feature vector.

$p(C_j|x)$: is the posterior probability of class C_j where x is given.

$p(C_j)$: is the prior probability of class.

$p(x|C_j)$: is the likelihood.

$p(x)$: is the prior probability of the predictor.

The advantages of Naive Bayes are:

- Simple and quick implementation.
- Working better with categorical input data than numerical data.

The Disadvantage of Naive Bayes is the big challenge is to implement the assumption of independence between features in real-world applications. (Krishnan, 2021)

Random Forest Classification:

To understand the Random Forest, it is necessary to have an overview of the decision tree which is the foundation of the Random Forest.

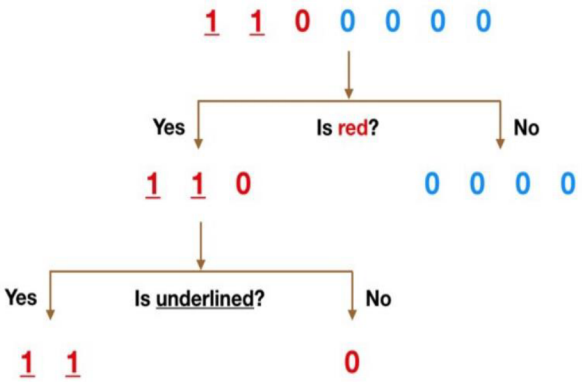


Figure 3: Decision Tree

(Source: (Yiu, 2019))

The above example illustrates the principle of the decision tree, and the aim is to classify data sets that consist of two (1s) and five (0s) into two classes according to their features. The methodology of separation here will be based on two features (if the color observation is red or blue, also if the observation is underlined or not).

Because one of the 0s is red, the first question to classify this dataset(is the observations color red)? If not, the path won't go down more because the color of observations is blue, so it's done here. If yes, the path will go down and there is another separation because the set now consists of two 1s that are underlined and one 0s isn't, so the second question will be is the observations underlined? The two 1s go down the yes branch because they are underlined and the 0 goes down to the no branch. Finally, the data has been separated into blue, underlined, and not underlined observations due to the methodology of the decision tree.

The most important concept Is ensemble learning which refers to the process of taking multiple decision trees/machine learning algorithms and putting them together to create one bigger algorithm, an example of this is a Random Forest that combines a lot of decision tree methods.

A Random Forest is a combination of many decision trees that operate as an ensemble. Each one of those trees is being built on a randomly selected subset from a dataset. Even though each one of those trees might not be ideal overall on average, the whole system can perform very well and that's a major advantage of this algorithm. It's kind of leveraging the power of the crowd so to speak or sort of just relying on one tree, it's checking what all the trees are going to decide and then just taking the majority vote and deciding the classification decide on that. (Yiu, 2019)

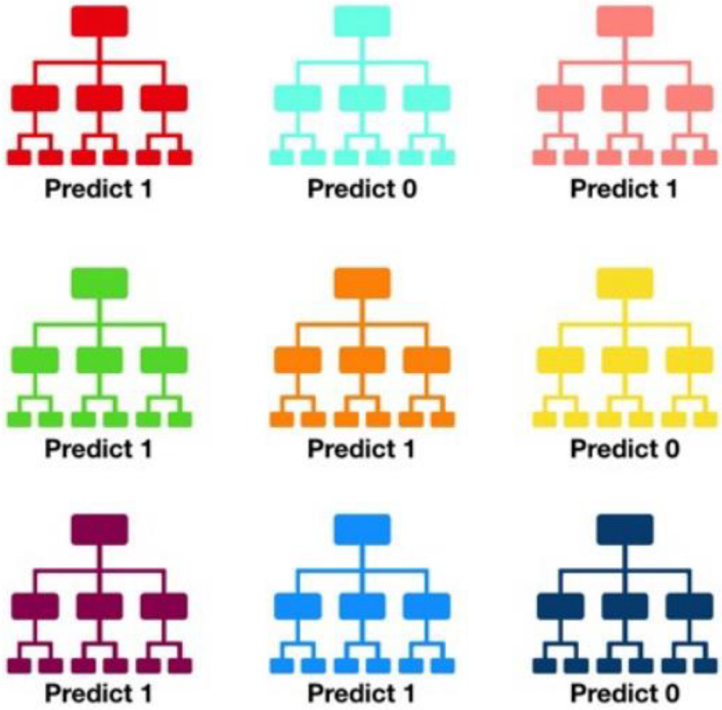


Figure 4: Random Forest

(Source: (Yiu, 2019))

3.2.3.2 Regression:

Regression is used for predicting a real value. There are two models of regression linear and non-linear. An example of regression is predicting the annual income of employees according to their age, education, and experience. The predicted value could be any number in each range. Another example is the predicting of yields of corn farms according to

weather, previous yield, and the number of workers on the farms the outcome(yields) could be an arbitrary number.

It's easy to distinguish between regression and classification if the outcomes in the data set are continuous so the task is regression-like predicting salary, and age. Whereas if the outcomes are discrete, then the task is classification -like predicting the category of the email. (Müller & Guido, 2016)

There are several machine learning models for regression problems:

- Simple Linear Regression.
- Multiple Linear Regression.
- Polynomial Regression.
- Support Vector for Regression [SVM].
- Decision Tree Regression.
- Random Forest Regression.

3.2.4 Unsupervised Learning:

The idea of unsupervised learning is that the data is unlabeled, and the system tries to determine some segments or clusters in the data. (Géron, 2019)

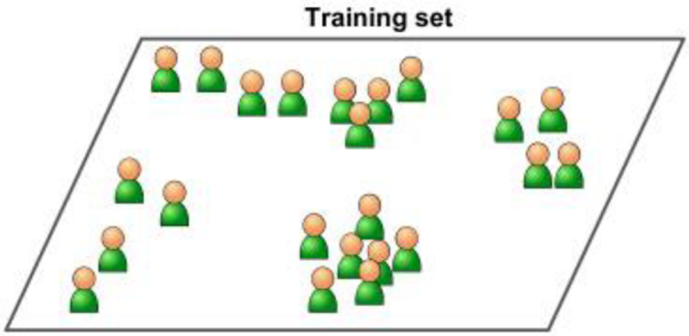


Figure 5: An unlabeled training set for unsupervised learning

(Source: (Géron, 2019))

There are several important unsupervised algorithms :

- Clustering
  - K-Means.
  - DBSCAN.
  - Hierarchical Cluster Analysis(HCA).
- Anomaly detection and novelty detection
  - One class SVM.
  - Isolation Forest.
- Visualization and dimensionality reduction
  - Principal Component Analysis (PCA)
  - Kernel PCA.
  - Local Linear Embedding (LLE).
  - T-distributed Stochastic Neighbor Embedding.
- Association rule learning
  - Apriori
  - Eclat

3.2.5 Semi-Supervised Learning:

Semi-supervised learning algorithms are applied when the training data consists of both many unlabeled data and a few labeled data. An example of it is photo-hosting services like Google Photos, when the user uploads a family photo to the service, it recognizes that person A is in photo 1,3,5 and person B in photo 2,6,8, this part of the algorithm is clustering, which is unsupervised learning. The user should label some photos per person, then the model can name everyone in each photo. (Géron, 2019)

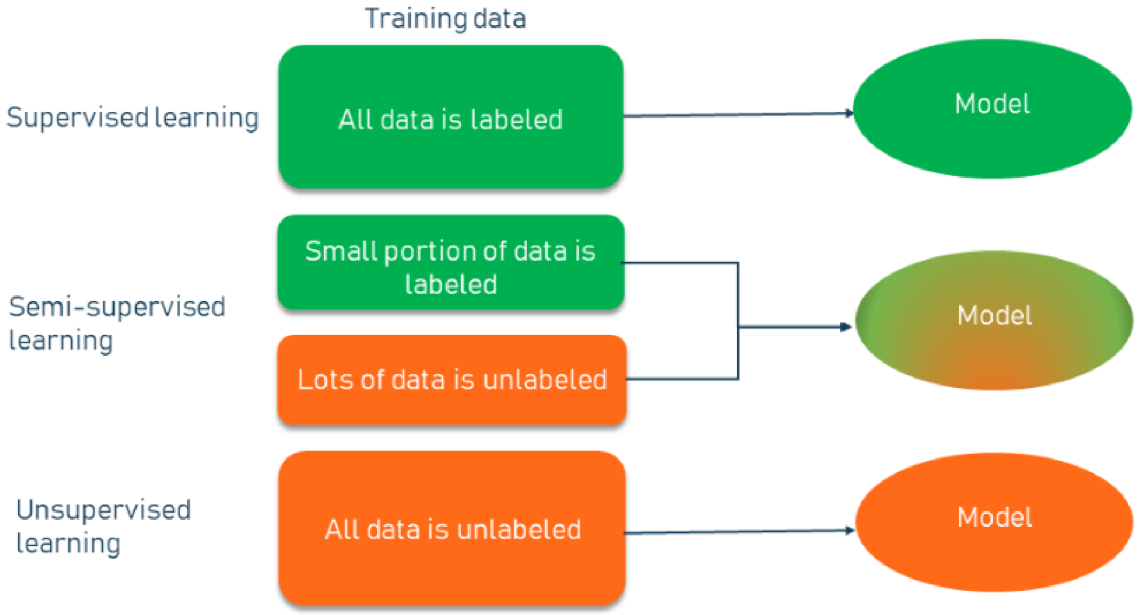


Figure 6: Semi-Supervised Learning Concept

(Source: https://www.altexsoft.com/blog/semi-supervised-learning/)

3.2.6 Reinforcement Learning:

Reinforcement learning is a powerful subset of machine learning. The methodology is divided into two stages. First, the learning system observes the environment, chooses, and carries out actions, and receives rewards in return (or penalties in the negative form) In this context the learning system is called an agent. Second, it must learn by itself what is the best strategy, and this is called the policy, to receive the reward over time. In another hand, the policy determines how the agent should act in certain situations.

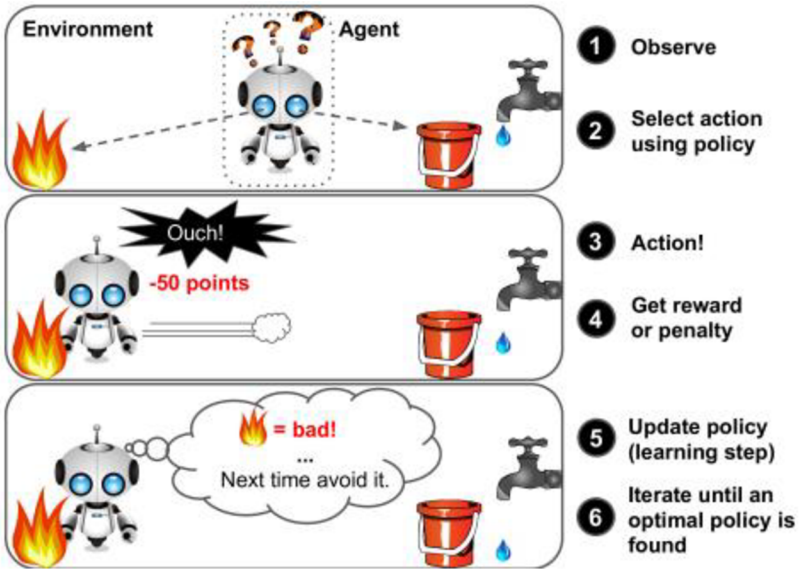


Figure 7: Reinforcement Learning

(Source: (Géron, 2019))

Many robots perform algorithms of learning systems to learn how to walk. An example of reinforcement learning is DeepMind's AlphaGo program which beat the famous world champion Ke Jie in the game of Go. The winning policy methodology involves learning by itself to analyze millions of games and then playing a large number of games against itself. It is worth noting that learning was turned off during the game. AlphaGo was simply implementing the policy it had learned. (Géron, 2019)

3.2.7 Terminologies in Machine Learning

3.2.7.1 Overfitting & Underfitting:

To figure out the underlying reasons behind low model accuracy, it's crucial to understand model fitting. This comprehension will direct to take corrective steps. The difference between prediction error on training data and evaluation (test) data is the key to knowing if the predictive model is underfitting or overfitting the training data.

When an over-complex machine learning model is trained and overly tuned to the training set, overfitting occurs. Because the model works well on training data, and it can't create good predictions on unseen data. On the contrary, the underfitting happens when the trained model is very simple and/or not well tuned/trained to lead to poor performance on the training as well as the test set.

3.2.7.2 Model & Algorithm:

There are two important terms in machine learning, which make confusion for non-expert people (e.g., people from other fields). The first term is “Machine Learning Algorithm” which represents the process and the steps that are applied to data to generate a “Machine Learning Model”, which is the second term. Machine learning algorithms implement “Pattern recognition” and learn from data or fit on a dataset. algorithms in classification like Logistic Regression and Naive Bayes and Random Forest and algorithms in regression and clustering are examples of machine learning algorithms.

There are several properties of machine learning algorithms:

- Math and pseudocode can be used to describe machine learning algorithms.
- Machine learning algorithms can be executed by any programming language.
- Machine learning effectiveness can be analyzed and described.

Scikit-learn is an example of a library that consists of many classification, regression, and clustering machine learning algorithms in Python.

Machine Learning Model refers to the output of a machine learning algorithm applied to data and shows what the machine learning algorithm learned. In precise, the model represents the final mathematical formula or rules that are used to do perdition. This output results from running the machine learning algorithm and can be saved and deployed to perform prediction.

Some examples of models:

- The results of linear regression in a model are represented by a linear equation connecting the input to the prediction. The coefficients of the linear equation terms are normally stored as a vector of values.
- The results of a Random Forest in a model represent a tree if-then rules with specific values.

There is a difference between machine learning models and other algorithms in computer science. The output of the sorting algorithm is a sorted list which isn’t really a model. It could be to think of a machine learning model as a “program”. The best definition of a machine learning model is a program that involves both data and the process of utilizing the data to make predictions. (Brownlee, 2020)

3.2.7.3 Cross-Validation

The train-test split aims to evaluate the performance of the machine learning algorithm. The technique comprised dividing the dataset into two separate sets. The first set is the training set which is utilized to fit the machine learning model and the second set is the test set which is utilized to evaluate the performance. To select the best parameters for the model, it is preferable to train the model many times, each one on a different training set, Then the model resulting in the highest accuracy on the test set is the best. This is the idea of cross-validation which helps estimate the expected performance of the trained model when deployed in production. (Lakshana, 2022)

3.3 Natural Language Processing:

3.3.1 What is NLP:

Natural Language Processing (NLP) is applying Machine Learning models to text and language. Teaching machines to understand what is said in the spoken and written word is the focus of Natural Language Processing. NLP is a subfield of linguistics, computer science, and artificial intelligence that uses algorithms to manipulate and analyze and transform written and spoken human language into numbers which enable machine learning models to predict the desired output.

3.3.2 The Origin of NLP:

In late 1940, The first AI systems were created. Hence the processing and analysis of Natural Language to understand texts have emerged. In 1950 Alan Turing who is regarded as the father of computer science and natural language processing did a test known as the Turing test. The test was initially called the “imitation game”. It measures a machine’s ability to exhibit intelligent behavior that matches or surpasses a human. The task involves understanding and writing back natural language effectively measuring the ability of a machine to fool humans into believing they are talking to a real person.

3.3.3 Evolution Of Natural Language Processing:

In 1956, happening the first translation of 60 sentences containing 250 words from English to the Russian language by computer in a few seconds, because of a collaboration between Georgetown University and IBM. On the face of it, it seems the authors made a breakthrough

even claiming that machine translation would be solved in 5 years, but it turned out the translation was more complicated than originally thought. (Hutchins, 2004)

The first emergence of the term "artificial intelligence" Was at Dartmouth College, New Hampshire in 1956. The book Syntactic Structures written by Noam Chomsky in 1957, involved a methodology to convert natural language words and sentences to a form that computers could understand. (Lichtig, 2011)

According to (Lichtig, 2011) between (1958-1990):

A language LISP(Locator/Identifier Separation Protocol) was created by John McCarthy in 1958. Then the first chatbot(Eliza) humans was released in 1966 by Joseph Weizenbaum and simulates a conversation between a person and a psychiatrist doctor.

The fast development in increasing the speed and power of computers in the period between 1960-1980 was reflected positively in natural language processing. SHRDLU was a project created in 1970. It involved reordering blocks, and cones according to users' input. SHDLU understands and performs sentences like" Put the red cube on top of the blue cube". It introduces the concepts of real-world applications as opposed to concepts of translation or software control.

The concept of the chatbot was developed in 1982 when researchers started work on the project Jabberwocky Chabot. The project is an AI program that aims to mimic human natural conversation in an interesting, entertaining way. Jabberwocky Chabot was the first attempt to pass the Turing test and added another application of NLP.

NLP started growing faster than ever in the early 1990s due to the widespread use of the internet and the massive quantity of data utilized. Machine learning algorithms concerned with machine translation have developed because of huge quantities of Canadian texts spread in English and French language.

Then according to (Bouargane, 2021) In 1994, a big advance in NLP was made, where the capability of machines to read increased almost 400 times faster than humans, But still not as quickly as human translators. After that in 2006, Google Translate was released, which uses several machine learning algorithms to translate words and sentences into other languages. Then in 2010, IBM released a system called Watson, which can analyze and understand questions and utilize machine learning algorithms to give the right answer.

3.3.4 Examples of NLP:

3.3.4.1 Email Filters:

The process of email filters means separating emails, this process includes filtering the email into two categories spam or not spam. Spam email is defined as an irrelevant message sent to users' computers using the internet as the medium with bad mood advertisement, phishing, or releasing malware. Because of the development of NLP, emails could be filtered into categories based on their content and each email is recognized according to its category, an example of it: g-mail categories(Primary, Social, and Promotions) (Tableau, 2022)

3.3.4.2 Language Translation:

Nowadays, NLP is considered a great and powerful technique. The utilization of NLP with deep learning algorithms aims to get the best accuracy of the translation. A language translator helps to communicate when a person talks to a person in another language and also helps researchers when trying to read the information in another language and get accurate outputs. (Tableau, 2022)

3.3.4.3 Smart Assistant:

Natural language generation is a subset of NLP which is utilized to analyze voices that able the machine to understand them. An example of a smart assistant is Alexa, Siri. (Tableau, 2022)

3.3.4.4 Search Engine Results :

NLP is utilized in search engines, when a user searches for a term on Google or another search engine, the results would be relevant and like what the intended user. If the user inserts a term on a Google search, The results wouldn't only match the exact term. Because Google search looks at the big picture and recognizes what the user types rather than the exact search term. For example, when a user types the term car on a google search, the predicted results include anything related to a car such as a car service, car brand, car game, car engine, etc. Another example is when the user inserts the term tree, the results would be about forests, Christmas, and trees in the data structure. Hence these differences when accomplishing a search as NLP in search connects an ambiguous query to the relative entity and returns great results. (Tableau, 2022)

3.3.4.5 Predictive Text:

As a result of the development of NLP and machine learning and deep learning algorithms, Predictive text become so common and necessary in smartphones and notebooks. Because predictive text includes autocorrecting and auto-completing a sentence or a word. When a user types a sentence, especially in another language, predictive text will change words in this sentence to make sense, and correct grammar if it's needed.

The Grammarly application is a great example of artificial intelligence. It's an application that teaches the user how to communicate and write better by autocorrect and suggestion. (Tableau, 2022)

3.3.4.6 Sentiment Analytics:

Nowadays, companies deal with enormous quantities of data, these data must be structured to save time and effort. Text analytics aids to organize unstructured data into significant data. Sentiment analysis is used in a wide area It helps companies to learn how a specific sort of user feels about a topic or product. With NLP, text analysis, and other techniques, companies determine whether the person is positive, negative, or natural about their products and services.

The sentiment is used by political candidates when they want to know political stances and who supports them. The governments also utilize sentiment analysis to have an overview of public opinion and determine potential dangers to the country's security. (Tableau, 2022)

3.3.5 NLP Techniques:

NLP technique means analyzing and processing the data(texts) in a form that enables machines to understand natural language. The most NLP technique used are:

3.3.5.1 Tokenization:

Tokenization is the process of breaking down text into phrases or words. This method streamlines the text analysis for future steps. When tokenization is applied to the sentence "I live in Prague" the following is the result :["I", "live", "in", "Prague"]. (NLP Techniques, 2022)

3.3.5.2 Stemming:

The principle of stemming is to reduce the word to its base, this process is based on the idea that a word with a slightly different spelling, but a roughly identical meaning should be in the same token. In another meaning, removing the suffixes from the given word to get the root of the word, so stemming uses fixed rules such as removing (ing, able, s) to derive a base word.

By applying stemming on these words(playing, walks, Prague, ate)the following are the results: play, walk, Pragu, ate

As a result, in some cases, applying stemming may have no meaning for the converted word like Pragu.

3.3.5.3 Lemmatization:

Lemmatization has the same principle as stemming, but it uses knowledge of a language(linguistic knowledge) to derive a base word.

By applying stemming on the word (ability), the result is (abil) whereas applying lemmatization to the same word, the result is (ability), also applying stemming on (ate), the result is (ate), whereas applying lemmatization the result is (eat).

From the examples above the lemma gives a meaningful word. So, lemmatization is more powerful than stemming but it requires more time.

Both stemming and lemmatization are used, stemming is used in applications like sentiment classifier positive, and negative sentiment analysis, whereas lemmatization can be used in chatbots also in question-answer applications because the response gets from those applications should be meaningful.

3.3.5.4 Bag of Words:

The basic idea of a bag of words is to create a matrix that contains the frequency of each word in the text. This process aims to generate a numerical format to can be fed into the model.

Below is an example of a bag of words where the document has three sentences:

‘I have a nice dog; my neighbor has a big dog. It is a nice cat. The dog and cat are nice.’

After applying lemmatization and stop of words (removing anything that has no meaning) the sentences would be:

‘Nice dog neighbor big dog, nice cat, dog cat nice’

then after counting the frequency of each word the results are nice 3, dog 3, cat 2, big 1, neighbor 1 then after applying BOW, the result is:

	Dog	Cat	Nice	neighbor	big
Sent 1	2	0	1	1	1
Sent 2	0	1	1	0	0
Sent 3	1	1	1	0	0

Table 1: Bag of words

(Source: Own work)

The disadvantage of BOW has only represented the frequency of each word and doesn't represent the importance and the semantics of each word, an example is nice 3, dog 3. To solve this problem there is a method called: Term frequency-inverse document frequency.

3.3.5.5 Term Frequency-Inverse Document Frequency:

Term frequency-inverse document frequency calculates the importance of each keyword in the document. Term frequency (TF) calculates the number of the repetition of each keyword in the sentence divided by the total of words in the sentence (Chouinard, 2022), from the example above by applying TF, the result of TF is :

	Sen 1	Sen 2	Sen3
Dog	2/5	0	1/3
Cat	0	½	1/3
Nice	1/5	½	1/3
neighbor	1/5	0	0
Big	1/5	0	0

Table 2: TF

(Source: Own work)

Inverse document frequency (IDF) =log(number of the total sentences divided by the number of sentences containing the keyword).

Keywords	IDF
Dog	Log(3/2)
Cat	Log(3/2)
Nice	Log(3/3)
Neighbor	Log(3/1)
Big	Log(3/1)

Table 3: IDF

(Source: Own work)

Then by multiplying TF*IDF the results represent the importance of each keyword.

3.3.5.6 Stop Words Removal:

Stop words removal means removing words that don't hold much value in the text such as [the, a, or, he, she,4, @....]. This technique aims to save time and effort and concentrate on meaningful words. The implementation of stop word removal isn't efficient in every NLP model, it relies on the task. When the task is an email classifier, so this technique is efficient, but if the task is machine translation, the stop words removal is inefficient.

There are several libraries for stop words removal, and it could be added some new words to the library according to the task. (NLP Techniques, 2022)

3.3.5.7 Topic Modeling:

Topic means a repeating group of statistically significant words in the corpus, statistically significant means a group of words occurring together in the document and they have similar ranges of TF-IDF values. A document is a group of topics. Topic modeling is the process of finding the major topics in the text, which means minimizing the big text into a small number of topics, and this is an unsupervised technique. Latent Dirichlet Allocation(LDA) is One of the best topic modeling algorithms. The principle of this algorithm is to determine the important topics in the document, and then assign documents to each topic according to a way that the topic must include all words in the document. (NLP Techniques, 2022)

3.3.5.8 Word Embedding:

Word embedding is one of the methods that convert the natural language to a numerical form to feed machine learning and deep learning algorithms. With this technique, words with

similar meanings have similar representations. Word embedding refers to numerical vectors, which means each word is presented as a numerical vector

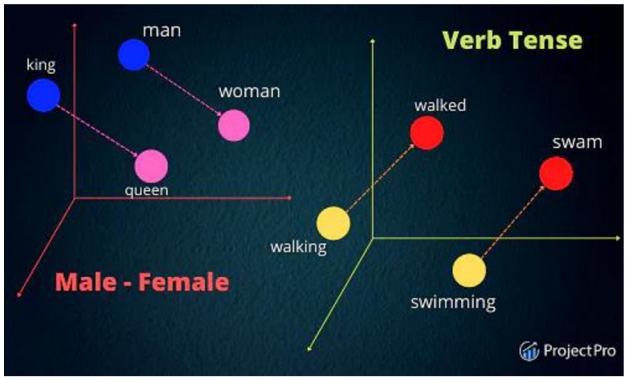


Figure 8: Word Embedding Concept

(Source: (10 NLP Techniques, 2022))

This figure illustrates that Each word is represented in a 3-3-dimensional space, and the distance between words with similar meanings is closer than words with a different meaning, the distance between man and swam is greater than the distance between swam and swimming. Also, this technique is helpful to release relationships between words. Such as the relationship between man and woman, also king and queen. In the vector space, the distance between man and woman is roughly equal to the distance between king and queen (10 NLP Techniques, 2022).

3.4 Deep Learning:

3.4.1 Introduction:

Deep learning is a subfield of machine learning, which exactly includes three or more layers of the neural network. It utilizes an artificial neural network and a huge dataset to simulate the behavior of the human brain and realize patterns that can be utilized for decision-making. Deep learning enhances automation and carries out physical and analytical tasks without the need for human interventions. The technologies of deep learning are used every day in many different areas like the detection of credit card fraud, digital assistants, voice, health care, and self-driving cars). (IBM Cloud Education, 2020)

3.4.2 Why Deep Learning:

As a result of exponential growth in data like using social media platforms (Facebook, Instagram, YouTube, …), it was difficult to create machine learning algorithms able to model a large amount of dataset accurately. Therefore, deep learning is utilized to solve these kinds of problems where there is a big amount of data. This is the essential difference between machine learning and deep learning.

Figure 9 illustrates that X-axis is the amount of data and the Y-axis is the performance of the algorithms, as it shows the amount of data is increasing with respect to the older learning algorithms(any machine learning algorithms) and at a specific time the performance remains almost constant and didn't increase. In the case of deep learning as the amount of data increasing also the performance is increasing, so this exponential growth of data leads to deep learning models. (Narayan, 2019)

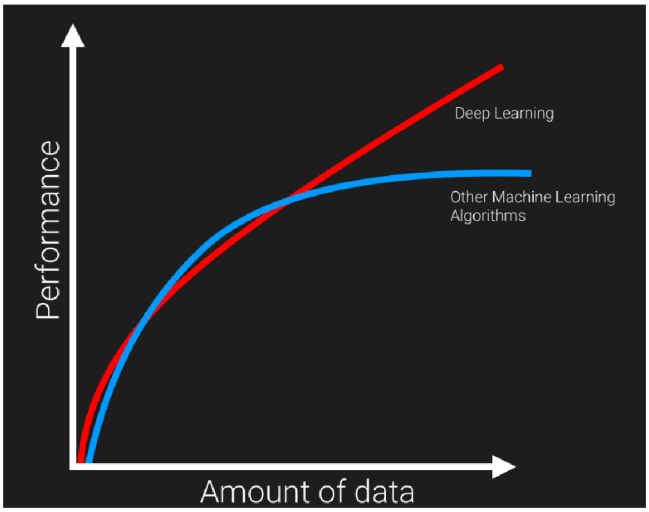


Figure 9: Importance of Deep Learning When Using Big Amount of Data

(Source: (Narayan, 2019))

3.4.3 Artificial Neural Network:

Neural networks are a set of algorithms that aim to understand the implicit relationship in a set of data by a process that simulates how the human brain works. Therefore, neural networks are systems of neurons that can be organic or artificial in nature. Neural networks

create the best results without having to redesign the output criteria because of their ability to adapt to change the input.

The neuron is a mathematical function that gathers and categorizes information based on a specific architecture. The network is quite similar to statistical techniques like regression analysis and curve fitting.

A neural network is made up of layers of nodes that are linked together. The node is a perceptron, which is like multiple linear regression. The signal produced by multiple linear regression is fed into the nonlinear activation function by perceptron (CHEN, 2021)

3.4.3.1 Perceptron:

Perceptron is invented in 1957 by Frank Rosenblatt. The process of Perceptron is based on threshold logic unit (TLU): first, inputs and outputs are numbers instead of binary, second, all inputs(Xi) are multiplied by their weights(Wi). Then The sum of weighted inputs is calculated (z=w1x1+w2x2+w3x3=…). The final step is applying the step function to this sum z.

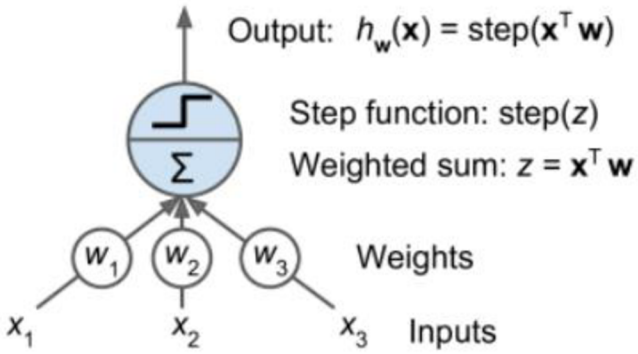


Figure 10: Threshold Logic Unit

(Source: (Géron, 2019))

The Heaviside step function is commonly used in perceptron. Sometimes the sign function is used.

$$\text{Heaviside}(z) = \begin{cases} 0 \; if \; z < 0 \\ 1 \; if \; z \geq 0 \end{cases} \qquad sign(z) = \begin{cases} -1 \; if \; z < 0 \\ 1 \; if \; z \geq 0 \end{cases}$$

The process of a single TLU is used in simple binary classification. TLU calculates the linear combination of the inputs. Then according to the result if exceeds the threshold the output is a positive class otherwise it's a negative class (like Logistic Regression or SVM). Training TLU leads to determining the right values for w1,w2, and w3.

A perceptron consists of a single layer of TLUs where each TLU is connected to all inputs. The fully connected layer is when all neurons in a layer are connected to every neuron in the previous layer.

Figure 11 represents a perceptron with two inputs and three outputs. The first layer is the input layer which represents that every input neuron is sent to every TLU. The output of the input layer is whatever the input is fed. The input layer also includes the bias neuron where the value of x is 1, therefore, the output of it is 1. This perceptron classifies instances together into three different binary classes which is a multiple output classifier.

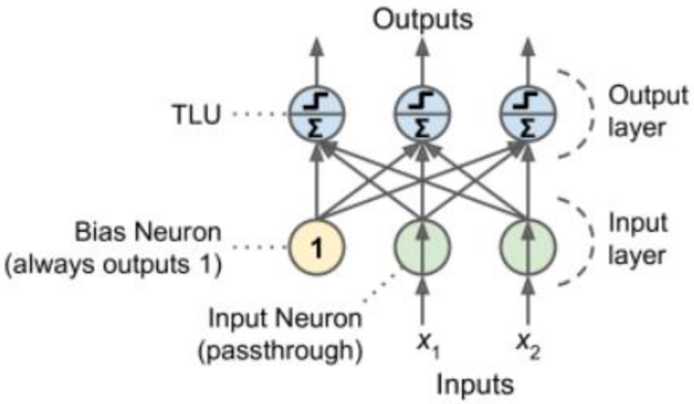


Figure 11: Perceptron Diagram

(Source: (Géron, 2019))

The equation for computing the output of a fully connected layer is :

$$h_{w,b}(x) = \emptyset(xw + b) \qquad (3)$$

X: matrix of features where columns for features and rows for instance.

W: matrix of weights which consists of all connection weights except the bias neuron.

B: bias vector consists of all connection weights between artificial neurons and bias neurons.

Ø: is the activation function. When the artificial neurons are TLUs, the activation function is a step function.

When the pattern is complex the perceptron can't learn of complex patterns like Logistic Regression. Therefore, when the learning instance is linearly separable, the solution would be by this algorithm, and this is called the Perceptron convergence theorem.

The difference between the Logistic Regression classifier and Perceptron is that the output of the Logistic Regression classifier is a class probability whereas the output of Perceptron is a prediction based on a hard threshold.

The weaknesses of perceptron have been noted by Marvin Minsky and Seymour Papert in 1969. One of the weaknesses is that the Perceptron can't solve an XOR classification problem.

Multi-layer Perceptron(MLP) able to solve the XOR problem as it's shown in figure 12: when the inputs of the network are (0,0) or (1,1) the outputs are 0, and when the inputs are (0,1) or (1,0) the outputs are 1. The weights of the connections are equal to 1 except the four connections where the weight is shown. (Géron, 2019)

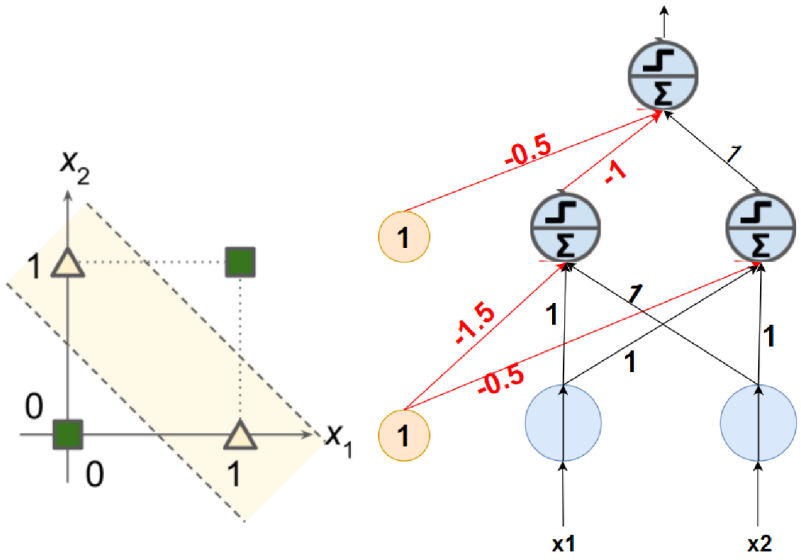


Figure 12: How MLP Can Solve XOR Classification Problem

(Source: (Géron, 2019))

3.4.3.2 Multi-Layer Perceptron:

MLP consists of one input layer, one or more TLU layers (hidden layer), and one output layer. The essential point in the architecture of MLP is the step function is replaced with a logistic function or other activation functions. In the step function, there is no gradient because it includes a flat segment therefore (Gradient descent can't move on a flat surface). The essential feature of the activation function is adding non-linearity to a neural network. Therefore, if there is a complex problem the (MLP) can solve this problem like the XOR problem. (Géron, 2019)

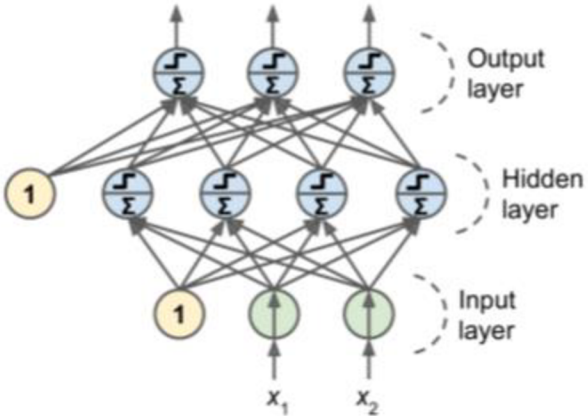


Figure 13: Multi-Layer Perceptron

(Source: (Géron, 2019))

3.4.4 Forward/Backward Propagation:

Forward and backward propagation processes, which are independent, are one of the fundamentals of understanding the process of the training model in neural networks.

Forward propagation:

During forward error propagation, an output prediction with some error is made. An overview of what occurs in each neuron should be done before going too deeply into this process.

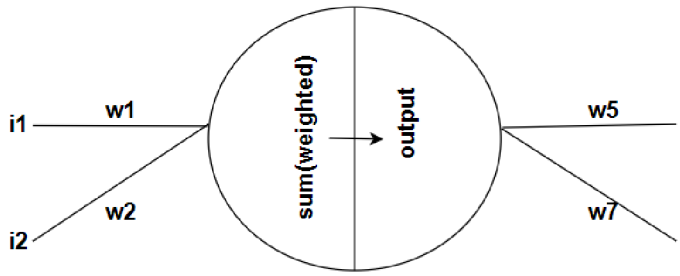


Figure 14: Inside h1 (first neuron of the hidden layer)

(Source: (Lamsal, 2021))

There are two operations inside every single neuron:

1. Sum of product: includes multiplying the weight vector by the given input vector.
2. Pass the sum through the activation function: The goal of this process is to pass the sum of the product from the input layer in every layer to give the output layer.

The output of the first layer becomes an input in the next layer and multiplies it by the weight of this layer. This process continues till the output layer.

Figure (15) is an example of a neural network that includes an input layer with two neurons (i1,i2), one hidden layer with two neurons (h1,h2), and an output layer with two neurons(o1,o2). The w1,w2,w3…w8 are the weights. B1,b2,b3,b4 are the bias for the neurons h1,h2,o1,o2 respectively. The used activation function is a sigmoid function.

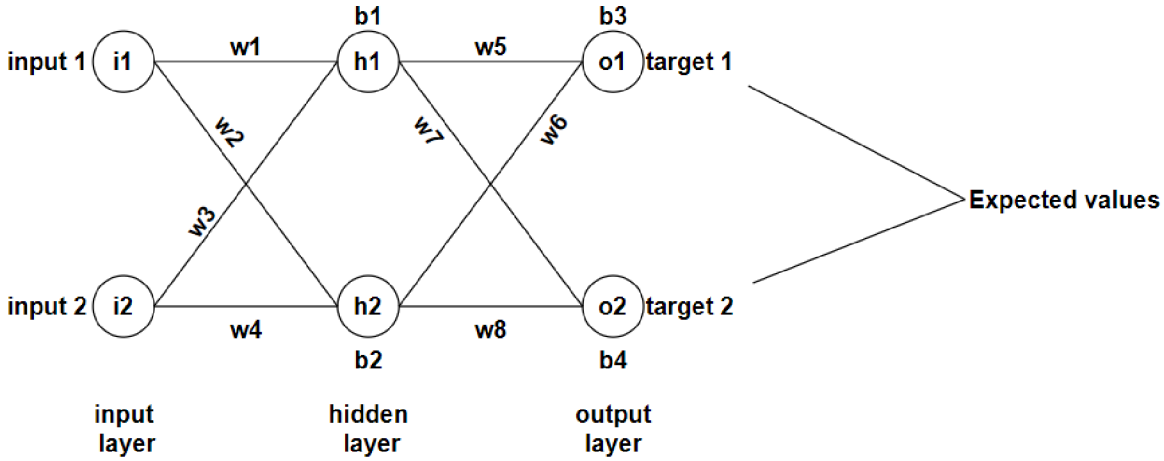


Figure 15: Neural Network example

(Source: (Lamsal, 2021))

let's get begin with the forward pass process.

For h_1:

$$sum_{h1} = i_1 * w_1 + i_2 * w_3 + b_1$$

Now passing sum_{h1} into a sigmoid function to squash the weighted sum into a range[0,1].

$$output_{h1} = \frac{1}{1 + e^{-sum_{h1}}}$$

Now the same operations for the neuron h_2

$$sum_{h2} = i_1 * w_2 + i_2 * w_4 + b_2$$

$$output_{h2} = \frac{1}{1 + e^{-sum_{h2}}}$$

Now $output_{h1}$ and $output_{h2}$ are considered inputs to the next layer.

For o_1:

$$sum_{O1} = output_{h1} * w_5 + output_{h2} * w_6 + b_3$$

$$output_{O1} = \frac{1}{1 + e^{-sum_{O1}}}$$

For o_2:

$$sum_{O2} = output_{h1} * w_7 + output_{h2} * w_8 + b_4$$

$$output_{O2} = \frac{1}{1 + e^{-sum_{O2}}}$$

Now the final step is computing the total error:

$$E_{total} = \sum \frac{1}{2}(taget - output)^2$$

Where target is the real value.

To compute E_{total} we need to compute errors at o_1 and o_2:

$$E1 = \frac{1}{2}(target1 - output_{O1})^2$$

$$E2 = \frac{1}{2}(target2 - output_{O2})^2$$

$$E_{total} = \sum \frac{1}{2}(E1 + E2)^2$$

Backward propagation:

The goal of Backpropagation (backward pass) is to distribute the total error back to the network to update the weights and minimize the cost function (loss). The weights are updated in such a way that when the next forward pass uses the updated weights, the total error is reduced by a certain margin (until the minimum is reached).

Updating weights (w_5, w_6, w_7, w_8) in the output layer:

for w_5:

to update w_5 we will use the chain rule. From figure(15) $E1$ is affected by $output_{O1}$, $output_{O1}$

is affected by sum_{O1}, sum_{O1} is affected by w_5.

$$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial output_{O1}} * \frac{\partial output_{O1}}{\partial sum_{O1}} * \frac{\partial sum_{O1}}{\partial w_5}$$

Computing the first component: partial derivative of Error w.r.t. Output

$$E_{total} = \sum \frac{1}{2}(E1 + E2)^2$$

$$E_{total} = \frac{1}{2}(target1 - output_{O1})^2 + \frac{1}{2}(target2 - output_{O2})^2$$

$$\frac{\partial E_{total}}{\partial output_{O1}} = 2 * \frac{1}{2}(target1 - output_{O1}) * -1$$

$$\frac{\partial E_{total}}{\partial output_{O1}} = output_{O1} - target1$$

Computing the second component: partial derivative of $output_{O1}$ w.r.t. sum_{O1}

The output section of a neuron of a neural network uses a non-linear activation function which is a sigmoid function:

$$f(x) = \frac{1}{1 + e^{-x}}$$

Notice that :

$$\frac{1}{1 + e^{-x}} = 1 - \frac{1}{1 + e^{-x}}$$

By using this fact the partial derivate could be written as :

$$\frac{d}{dx} f(x) = f(x) * (1 - f(x))$$

Therefore, the derivative of the Logistic function is equal to the output multiplied by (1 – output).

$$\frac{\partial output_{O1}}{\partial sum_{O1}} = output_{O1}(1 - output_{O1})$$

Computing the third component: partial derivative of sum_{o1} w.r.t. Weight5

$$sum_{O1} = output_{h1} * w_5 + output_{h2} * w_6 + b_3$$

$$\frac{\partial sum_{O1}}{\partial w_5} = output_{h1}$$

Putting them together:

$$\frac{\partial E_{total}}{\partial w_5} = (output_{O1} - target1) * output_{O1}(1 - output_{O1}) * output_{h1}$$

Therefore:

$$w_5(new) = w_5 - n * \frac{\partial E_{total}}{\partial w_5}$$

n: learning rate.

Remaining weights(w_6, w_7, w_8) can be updated in the same way.

Updating weights (w_1, w_2, w_3, w_4) in the hidden layer:

The same process to update weights is the hidden layer but the chain rule becomes a bit longer

For w_1 (with respect to E_1):

$$\frac{\partial E_1}{\partial w_1} = \frac{\partial E_1}{\partial output_{O1}} * \frac{\partial output_{O1}}{\partial sum_{O1}} * \frac{\partial sum_{O1}}{\partial output_{h1}} * \frac{\partial output_{h1}}{\partial sum_{h1}} * \frac{\partial sum_{h1}}{dw_1}$$

Computing the first component:

$$\frac{\partial E_1}{\partial output_{O1}} = output_{O1} - target1$$

The second component is already computed.

Computing the third component:

$$sum_{O1} = output_{h1} * w_5 + output_{h2} * w_6 + b_3$$

$$\frac{\partial sum_{O1}}{\partial output_{h1}} = w_5$$

Computing the fourth component:

$$\frac{\partial output_{h1}}{\partial sum_{h1}} = output_{h1}(1 - output_{h1})$$

Computing the fifth component:

$$sum_{h1} = i_1 * w_1 + i_2 * w_3 + b_1$$

$$\frac{\partial sum_{h1}}{\partial w_1} = i_1$$

Putting them together:

$$\frac{\partial E_1}{\partial w_1} = (output_{O1} - target1) * \left(output_{O1}(1 - output_{O1})\right) * w_5$$
$$* output_{h1}(1 - output_{h1}) * i_1$$

For w_1 with respect to (E_2):

$$\frac{\partial E_2}{\partial w_1} = (output_{O2} - target2) * \left(output_{O2}(1 - output_{O2})\right) * w_7$$
$$* output_{h1}(1 - output_{h1}) * i_1$$

Now:

$$\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_1}{\partial w_1} + \frac{\partial E_2}{\partial w_1}$$

$$w_1(new) = w_1 - n * \frac{\partial E_{total}}{\partial w_1}$$

Remaining weights (w_2, w_3, w_4) can be updated in the same way.

After we've calculated all of the new weights, we must update all of the old weights with the new weights. One backpropagation cycle is completed once the weights are updated. The forward pass is now complete, and the total new error is computed. The weights are then updated based on the newly computed total error. This process is repeated until the loss value

converges to minima. In this manner, a neural network begins with random weight values and eventually converges to optimal values. (Lamsal, 2021)

It should be noted that the previous example considers the weights update. The bias update, on other hand, can be updated in the same way.

The example represents a regression problem (not a classification problem) for this reason the loss function is a mean squared error. However, the same backward propagation process can be applied to any other loss function for a regression or classification problem.

3.4.5 Activation Function:

The aim of the activation function is to add non-linearity to each layer in the neural network. This step is essential during forward propagation.

When the neural network is working without adding activation functions, each neuron will implement a linear transformation on inputs utilizing the weight and bias. Therefore, the number of hidden layers isn't important in this case, because the output of combining two linear functions is a linear function so the neural network will work simply and couldn't learn any complex task.

3.4.6 Types of Activation Functions:

3.4.6.1 Binary Step Function:

This function depends on the value of the threshold which determines if the neuron should be activated or not. The principle is to compare a certain threshold to the input fed to the activation function. If the input is greater than this value so the neuron is activated which means the output of this neuron is passed to the next hidden later. If the input isn't greater than this value so the neuron isn't activated.

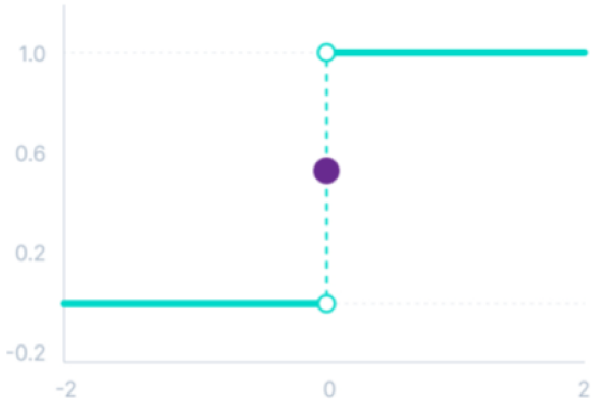


Figure 16: Binary Step Function

(Source: (Baheti, 2022))

Binary step:

$$f(x) = \begin{cases} 0 \; for \; x < 0 \\ 1 \; for \; x \geq 0 \end{cases} \quad (4)$$

The disadvantages of the binary step function are:

- It can't be utilized for a multiclass classification problem.
- The gradient of this function is zero which led to an obstacle in the backward propagation process. (Baheti, 2022)

3.4.6.2 Linear Activation Function:

This function is known as "no activation". The output of This function is the same as the value it was given because the linear activation function doesn't modify the sum of weights of the input.

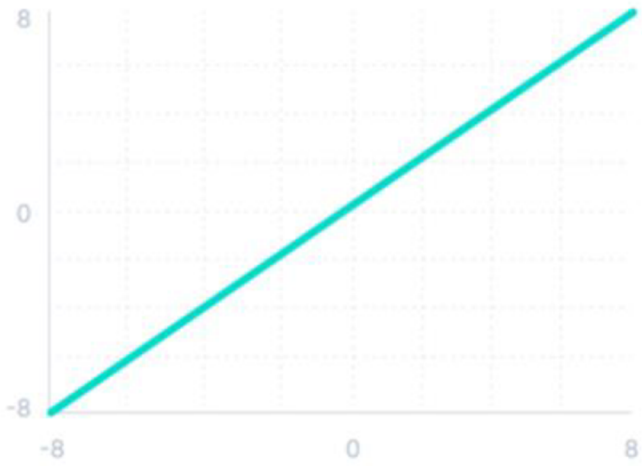


Figure 17: Linear Activation Function

(Source: (Baheti, 2022))

Linear:

$$f(x) = x \qquad (5)$$

The disadvantages of the linear activation function are:

- Because the derivative of this function is a constant and hasn't relation to the input x, it's impossible to implement backpropagation.
- A linear function will cause the neural network's layers to merge into one. The last layer of the neural network will be a linear function of the first layer, regardless of the number of layers, thus, a linear activation function effectively reduces the neural network to a single layer.

3.4.6.3 Non-Linear Activation Function:

The Advantages of non-linear activation functions are:

- Because the derivative of these functions isn't a constant and is related to the input x, it's possible to implement backward propagation and determine which inputs in neurons provide a better prediction.
- The output passed through the multiple layers would be a combination of non-linear activation functions, and they allow for a stack of multiple layers of neurons.

There are several types of non-linear activation functions:

3.4.6.4 Sigmoid/Logistic Activation Function:

The sigmoid function transforms the value of the input to a value between the range [0,1]. When the model is used to predict probability, the sigmoid function is the best choice because the probability is in the range [0,1]. The gradient provided by this function is smooth and this function is differentiable.

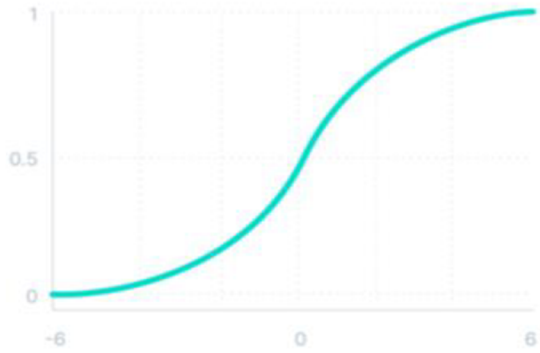


Figure 18: Sigmoid Function

(Source: (Baheti, 2022))

The formula of the sigmoid function is:

$$f(x) = \frac{1}{1+e^{-x}} \qquad (6)$$

The disadvantage of the sigmoid function is the vanishing gradient when doing backward propagation, which means the weight updating is getting updated by a small number. Also, the function output isn't zero-centered.

3.4.6.5 Tanh Or Hyperbolic Tangent Activation:

Tanh function is like the sigmoid function and even has the same shape. The difference between sigmoid and tanh is that tanh has a range from [-1,1].
This function is differentiable, the function output is zero-centered. (Sharma, 2017)

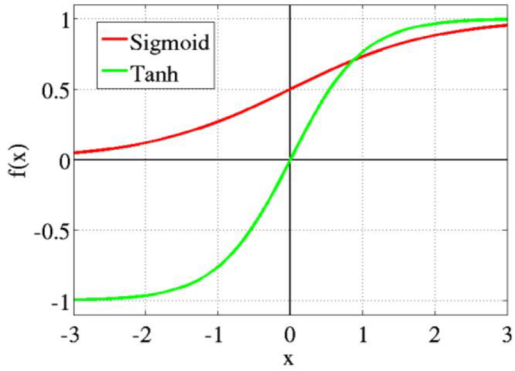


Figure 19: Tanh Activation Function

(Source: (Sharma, 2017))

3.4.6.6 RELU Activation Function:

RELU stands for Rectified Linear Unit. It's the most utilized activation function. Almost all convolutional neural networks and deep learning used it.

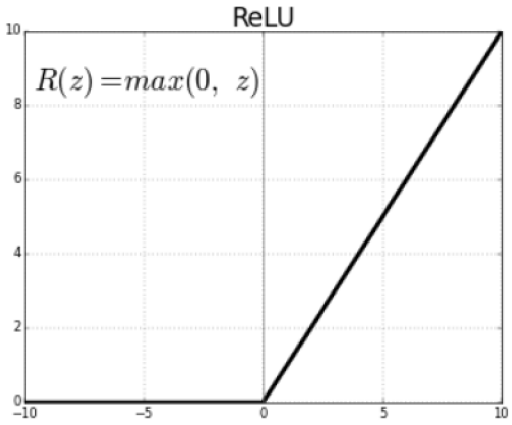


Figure 20: RELU Activation Function

(Source: (Sharma, 2017))

As it's shown this function is half rectified and the output of RELU is zero when z is less than zero and the output is z when z is larger than 0. The range is from 0 to infinity. This function and its derivative are monotonic.

The disadvantage of this function is that takes any negative value and turns it into zero, which reduces the model's ability to effectively fit or train from the data. The resulting graph of this process doesn't properly map the negative values. (Sharma, 2017)

3.4.6.7 Leaky RELU:

The leaky function solves the dying problem of the RELU function because in the negative area it has a small positive slope. The formula of Leaky RELU is:

$$f(x) = \max(o.1x, x) \qquad (7)$$

RELU and Leaky RELU functions are the same, as well as Leaky performs backpropagation for negative input values.

This small modification ensures that the gradient of the graph's left side is non-zero for negative input values. As a result, the dead neurons wouldn't be anymore. (Baheti, 2022)

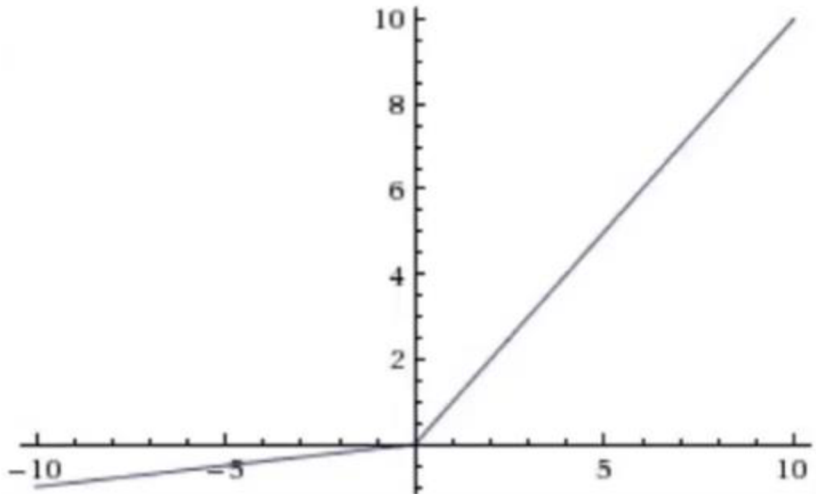


Figure 21: Leaky ReLU

(Source: (Sharma, 2017))

3.4.6.8 Softmax

The aim of softmax is to compute the sum of multiple sigmoid functions where the output of the sigmoid function is a probability in the range[0,1]. Softmax is the most used activation function in the last layer of ANN. The formula of Softmax is:

$$softmax(Z_i) = \frac{\exp(Z_i)}{\sum i \exp(Z_i)} \qquad (8)$$

An example of using softmax:

If there are three classes it means there are three neurons in the output layer and the output of neurons is [1.8, 0.9, 0.68]. Applying softmax over these values will give the probability of each class[0.58, 0.23, 0.19]. The function returns 1 for the largest probability index and 0 for the remaining two array indexes. Index 0 is given full weight, while indexes 1 and 2 are given no weight. As a result, the output would be the class associated with the first neuron (index 0) of three. (Baheti, 2022)

3.4.7 Terminologies in ANN:

There are several terminologies in ANN:

3.4.7.1 Sample

A sample is a single data row, that contains inputs for the algorithm. For example, a sample represents a single image for image classification tasks or a single email for email classification tasks. When supervised learning is utilized, every sample is accompanied by an output (i.e., a label) which is compared to the predicted output of that sample during the training process (the error or the difference between the label and the prediction is to be minimized during training iterations). A dataset for training consists of many samples.
The sample is also called an observation or instance or an input vector. (Brownlee, 2020)

3.4.7.2 Batch and Epoch

When training a neural network, it is common to use Gradient Decent (GD) to tune the neural network weights such that the loss function is minimized. However, to speed up the training, a subset of the dataset samples is usually enough to estimate the loss gradient and update the weights. Therefore, GD is usually applied to a subset of the data samples in every iteration. This subset is called a batch. Based on that, the dataset is split into non-overlapping batches. Then the training iterates over the batches and uses GD on each batch updating the weights in every iteration. When all batches are processed once, this is called an epoch of training. The process repeats as the training process is done over many epochs. In this case, GD is called “Stochastic Gradient Decent” (SGD).
The batch size and the number of training epochs are both hyperparameters that can be tuned by the machine learning engineer or the data scientist performing the training.

Note that if the batch size is set to be equal to the size of the whole data set, then there is only one batch, and every single epoch is a single training iteration. In this case, The algorithm would be "Gradient Decent" (i.e., not stochastic). On the other hand, if the batch size is set to one, the number of batches would be equal to the data set samples and the number of iterations to complete a single training epoch would also be equal to the number of data set samples as well. In this case, the optimization algorithm is called "Mini-batch Gradient Decent". (Brownlee, 2020)

3.4.7.3 Loss Function:

The performance of the model is measured by the loss, which represents how much the model differs from the desired outcome. Hence, the lower the loss, the better the model. The loss function is one of the most crucial components of neural networks which, together with the optimization algorithm, are directly in charge of fitting the model to the provided training data. The principle is to compare the target and predicted output values and to measure how well the neural network models training data. When training, the goal is to minimize the loss between the target and predict value. (Yathish, 2022)

Common types of loss function:

- Regression loss functions: Mean Squared Error, Mean Absolute Error.
- Classification loss functions: Binary Cross-Entropy, Categorical Cross-Entropy. (Yathish, 2022)

Binary Cross-Entropy/Log Loss:

It's used in binary classification cases where there are two actual values of y 0 or 1.

The forum of cross entropy is:

$$CE\ LOSS = -\frac{1}{N}\sum_{i=1}^{N} (y_i \,.\log(p_i) + (1 - y_i).\log(1 - p_i) \qquad (9)$$

To determine the loss between the predicted and actual values, it's necessary to compare the actual value 0 or 1 with the probability that the input aligns with that category.

p(i): represents the probability where the category is 1.

1 — p(i): represents the probability where the category is 0.

N: represents the number of samples.

y_i: represents the label/class (1 or 0). (Pawar, 2021)

Categorical Cross-Entropy Loss:

It's used when the number of classes is greater than two. The formula of categorical cross-entropy is a generalization of the previous formula from (9) to the multi-class case and can be written as:

$$CE\ LOSS = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij} \cdot \log(p_{ij}) \quad (10)$$

N: represents the number of samples.

M: represents the number of possible classes.

y_{ij}: represents a binary indicator of whether the real label of sample i is class j or not.

p_{ij}: represents the predicted probability of class j for sample i. (Yathish, 2022)

Note that other loss functions are out of the scope of this thesis and, hence, there is no need to show/explain more details about those other loss functions.

4 Practical part:

4.1 Data set

4.1.1 Data acquisition

As expected, data sets with email contents are of limited availability due to privacy issues. Therefore, we have used a data set available for research purposes from the well-known website "www. kaggle.com".

The data set includes 2000 emails including the emails' content text. For each email, a corresponding topic out of three available topics is attached. Those three topics are "Crime", "Politics" and "Science". If an email does not belong to any of those three topics, this email is considered to be "Others" referring to any other topic that is not crime, politics, or science. Based on that, each of the emails is classified into one of the four available classes, i.e., Crime, Politics, Science, and Others.

It is important to note that the data is acquired such that the balance between the four classes is maintained. In precise, the 2000 emails include 500 emails (25% of the whole data) for each of the four classes. This keeps the equality of the number of samples for each of the classes and prevents any possible bias in the trained models due to the class imbalance problems.

The data set is provided by a major newspaper in which the editor was receiving a high number of emails from his/her journalists. Those emails were separated into categories indicating each email's topic separated into categories. The editor grew tired of these emails because the newspaper's IT department did a poor job of maintaining the data, resulting in text files labeled in multiple folders. 14147.txt, for example, is labeled as 'Crime', 'Science', and 'Others'. The task is to build a model that is able to classify all the emails into their proper categories based on their content in order to correct the IT department's mistakes.

4.1.2 Data pre-processing

In this sub-section, the data pre-processing steps are described. The goal is to have a clean numeric input that a machine learning model can understand and be trained based upon.

4.1.2.1 Get folders and load emails:

This step is to get a folder in the directory and load every email and put features in x and the labels(email's topic) in y, and then look at the shapes of the inputs and one sample.
then transform x from list to array.
the code developed to perform the analysis presented in this thesis is in the form of a jupyter notebook.

```python
# load every email and put the features (emails' text) in x and the labels (emails' topics) in y
x=[]
y=[]
for directory in directories:
    path2= path +"/"+ directory + "/"
    with os.scandir(path2) as entries:
        for entry in entries:
            with open(entry, "r", encoding = "latin1") as file:
                m=file.read()
                if len(m)!= 0:  # avoid empty emails (data quality issue)
                    x.append(m)
                    y.append(directory)


# take a look at the shapes of the inputs and check one sample
print('Input features shape : ', np.shape(x))
print('Labels shape : ', np.shape(y))
print ('The label/topic of the first email is : ', y[0])
print ('The first email in the topic' + "(" + y[0] + ")" + " is :" +"\n" + x[0])
```

✓ 1.5s

```
Output exceeds the size limit. Open the full output data in a text editor
Input features shape :  (1997,)
Labels shape :  (1997,)
The label/topic of the first email is :  Crime
The first email in the topic(Crime) is :

Archive-name: ripem/faq
Last-update: Sun, 7 Mar 93 21:00:00 -0500

ABOUT THIS POSTING
------------------
This is a (still rather rough) listing of likely questions and
information about RIPEM, a program for public key mail encryption.  It
(this FAQ, not RIPEM) was written and will be maintained by Marc
VanHeyningen, <mvanheyn@whale.cs.indiana.edu>.  It will be posted to a
variety of newsgroups on a monthly basis; follow-up discussion specific
to RIPEM is redirected to the group alt.security.ripem.
```

Figure 22: Code for loading every email

4.1.2.2 Data splitting

This step is typical in every model development process. The idea is to split the whole data set into two parts, a training, and a testing part (referred to as the training set and the test set respectively). The remaining data pre-processing steps and the model training are applied to the training set only. Then, the whole pre-processing and the trained models are evaluated on the test set. The test set represents the set of emails that would be received in the future when the trained and tested model is put into production. In such case, the emails would be inserted into the model and the model has to predict the category of each email. Bearing this in mind, the reachable accuracy on the test set is an estimate of the expected performance of the model when deployed in production.

Based on the pre-described principle, the 2000 emails are split into a training and test set with a split ratio of 80:20. This means that 1600 emails are used for the training (training set) and the remaining 400 emails are used for the testing (test set).

Although the split is done randomly, it is important to note that the “stratified” train-test split is considered. In the stratified train-test split, the class balance is kept in the training as well as the test set. In detail, the training set includes 400 emails for each of the four classes (making a total of 1600 emails in the training set). Similarly, the test set includes 100 emails from each of the four classes (making a total of 400 emails for the test set). This keeps the fairness between the training and the test phases guaranteeing unbiased training on the training set and a fair evaluation of the model on the test set.

The block of code of transforming x from list to array and train-test split and the results are shown in Figure 23.

```
# transform x from list to numpy array
x = np.array(x).reshape(-1)
print(x.shape)

# split data into train/test sets via a stratified fashion
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size = 0.20, stratify = y, shuffle= True, random_state = 1200)
print(y_train[1])
print(X_train[1])
```

✓ 2.3s

```
Output exceeds the size limit. Open the full output data in a text editor
(1997,)
Politics

In article <visser.735260518@convex.convex.com> (alt.conspiracy,talk.politics.misc,talk.religion.misc), visser@convex.com (Lance Visser) writes:
] In <bskendigC5qyJ2.GEw@netcom.com> bskendig@netcom.com (Brian Kendig) writes:
]
] +>b645zaw@utarlg.uta.edu (Stephen Tice) writes:
] +>>
] +>>One way or another -- so much for patience. Too bad you couldn't just
] +>>wait. Was the prospect of God's Message just too much to take?
]
] +>So you believe that David Koresh really is Jesus Christ?
]
]        They cut off the water, there were no fire trucks present and
] the FBI/ATF go blasting holes into the builing and firing gas munitions.
] The building burns, almost everyone dies.  It probably doesn't bother
] you much, but it bothers many other people.....most of whom dont believe
] particularly in Koresh or his message.
]
]        Four ATF agents and 90 branch Davidians are now dead because of
] crazy tactics on the part of the ATF and FBI.
]
]        Attorney General Vampira tells us that todays events were suppose
] to "save" those in the compound.  Blowing holes in a building and
] gassing those inside was supposed to "save" them?
```

Figure 23: Train-Test split

4.1.2.3 Label encode the labels:

This step is to encode the labels of the (y) matrix and check the labels. To do this, the code used for labeling (y) and inspecting the labels is shown in Figure 24.

```
# Label encode the labels (0: , 1: ,2: ,3:)
le = LabelEncoder()
print('The existing unique email topics : ', pd.unique(y_train))
y_train = le.fit_transform(y_train)
y_test=le.transform(y_test)

# check the labels:

print('The resulted labels for each email topic : ', pd.unique(y_train))
print(len(y_train))

print(np.shape(X_train[y_train==0]))
print(np.shape(X_train[y_train==1]))
print(np.shape(X_train[y_train==2]))
print(np.shape(X_train[y_train==3]))

print(np.shape(X_test[y_test==0]))
print(np.shape(X_test[y_test==1]))
print(np.shape(X_test[y_test==2]))
print(np.shape(X_test[y_test==3]))
```

✓ 1.8s

```
The existing unique email topics :  ['Other' 'Politics' 'Crime' 'Science']
The resulted labels for each email topic :  [1 2 0 3]
1597
(399,)
(400,)
(400,)
(398,)
(100,)
(100,)
(100,)
(100,)
```

Figure 24: Code of labeling (y)

4.1.2.4 Tokenization, Data cleaning, and Lemmatization

In this subsection, the process of applying the standard text data preparation is described as follows.

First, the content of every email is separated into a list of words (tokenization). Then, the following data-cleaning steps are applied:

- All words are changed into lowercase (small letters)
- All Latin or non-alphabet words in addition to the words with less or equal to 3 letters are excluded from the list of words
- All remaining words are converted into their root or first form (lemmatization)

The aforementioned cleaning steps guarantee that all redundant words or letters (not beneficial for the identification of the email's topic) are excluded. It also guarantees the same word structure (or tense) over the various emails.

The code used for cleaning every email is shown in Fig.25 shown below:

```
X_train_before = X_train.copy()

def text_cleaning_function(X_train):
    stop = stopwords.words('english')
    corpus=[]
    wordnet_lemmatizer = WordNetLemmatizer()

    for punct in punctuation:
        stop.append(punct)

    for text in X_train:
        sentences= WordPunctTokenizer().tokenize(text.lower())
        review=[regex.sub(u'\p{^Latin}', u'', w) for w in sentences if w.isalpha() and len(w) > 3]#
        review=[wordnet_lemmatizer.lemmatize(w, pos="v") for w in review if not w in stop]
        review = ' '.join(review)
        corpus.append(review)
    return corpus


# transform emails (cleaning ..etc)
X_train = text_cleaning_function(X_train_before)
print(X_train[1])
```

Figure 25: Code for cleaning every email

The result after executing this step to the same email above from Fig.23 is shown :

```
article visser convex convex conspiracy talk politics misc talk religion misc visser convex lance visser write netcom bskendig netcom brian
much take believe david koresh really jesus christ water fire truck present blast hole builing fire munition build burn almost everyone die
four agents branch davidians dead crazy tactics part attorney general vampira tell todays events suppose save compound blow hole build gas
```

```
black batf earn would multiply many time also mourn deaths batf agents although deceptions carry government nothing
```

```
organization start whole debacle rick usenet rick howtek america online disclaimer employer view orthogonal early bird worm
```

Note that three emails out of the 1600 emails of the training set remained empty after this preprocessing step. Hence, these emails were excluded from the analysis. The remaining emails used for the training were 1597 emails/training samples.

4.1.2.5 TFIDF

The Term Frequency-Inverse Document Frequency (TF-IDF) is used to convert the emails into a numeric matrix with the shape of the number of samples (emails), x the number of words. The number of words equals the number of all distinct words over the whole training set. Each cell in the matrix represents the importance of the corresponding word (defined by the column) in the corresponding data sample/email (defined by the row).
This step concludes the preprocessing of data such that the emails are in a form of an input numerical matrix that can be used to train a variety of machine learning models.
The code and the result are shown in Figure 26.

```
# vectorise email texts (Tfid words --> numbers)
vectorizer_data = TfidfVectorizer()
X = vectorizer_data.fit_transform(X_train).toarray()
print(X.shape)

✓ 1.9s
(1597, 20809)
```

Figure 26: TF-IDF-code

Note that the number 1597 refers to the number of the training set samples (i.e., emails) while the number 20809 refers to the number of all distinct words over the whole training set.
The table below in Fig. 27 illustrates that, for example, the sample (email) in row 82 has three words that are used in this email with the following importance: 0.217, 0.09315, and

0.1148. Also, the sample in row 84 includes the word represented by column 41 with importance of 0.0562. Note that the table from Fig. 27 was generated using Spyder IDE.

	37	38	39	40	41	42	43	44	45	46	47
76	0	0	0	0	0	0	0	0	0	0	0
77	0	0	0	0	0	0	0	0	0	0	0
78	0	0	0	0	0	0	0	0	0	0	0
79	0	0	0	0	0	0	0	0	0	0	0
80	0	0	0	0	0	0	0	0	0	0	0
81	0	0	0	0	0	0	0	0	0	0	0
82	0	0	0	0	0	0	0	0	0.217642	0.0941553	0.114884
83	0	0	0	0	0	0	0	0	0	0	0
84	0	0	0	0	0.0562204	0	0	0	0	0	0
85	0	0	0	0	0	0	0	0	0	0	0
86	0	0	0	0	0	0	0	0.0756071	0	0	0
87	0	0	0	0	0	0	0	0	0	0	0
88	0	0	0	0	0	0	0	0	0	0	0
89	0	0	0	0	0	0	0	0	0	0	0
90	0	0	0	0	0.0441426	0	0	0	0	0	0

Figure 27: The “matrix of words” representing the emails

4.2 Models training

In this section, we go over the training of the various machine learning models used for email classification. Of course, all the trained models were tested on a test set to guarantee the avoidance of overfitting. Note that the emails of the test set were also preprocessed using the preprocessors that were fit on the training set. The test set preparation code is the following.

```
# prepare test data

# clean
X_test = text_cleaning_function(X_test)

# vecotrise emails (use transform where the vectoriser is fit on training data)
X_test = vectorizer_data.transform(X_test).toarray()
```

Figure 28: Test data preparation

4.2.1 Logistic Regression

A Logistic Regression model was trained first to do the email classification task. scikit-learn library from Python was used. The default Logistic Regression parameters were kept(

penalty='l2' C:float, default=1.0 , intercept-scaling=1,fit_intercept=True*)* The code used is the following:

```
# train logistic regression
LR_classifier = LogisticRegression(random_state = 0)
LR_classifier.fit(X, y_train)
acc_training_data = 100*LR_classifier.score(X, y_train)
print('Logistic Regression : The accuracy on the training data is : ',  acc_training_data, '%')
```

```
Logistic Regression : The accuracy on the training data is :  99.87476518472135 %
```

Figure 29: Logistic Regression code - Training

The reached total accuracy of the trained Logistic Regression model was 99.8%. However, to examine the performance of the trained Logistic Regression model on unseen data, the model was tested on the test set. The code used for testing is shown below:

```
# predict test output using the traind model
LR_y_pred = LR_classifier.predict(X_test)

# confusion matrix
LR_cm = confusion_matrix(y_test, LR_y_pred)
print('Logistic Regression : The confusion matrix values are : ', LR_cm)

# accuracy calculation
LR_acc = 100*accuracy_score(y_test, LR_y_pred)
print('Logistic Regression : The accuracy on the test set is : ', LR_acc, '%')
```

```
Logistic Regression : The confusion matrix values are :  [[97  0  1  2]
 [ 0 99  1  0]
 [ 0  0 99  1]
 [ 0  0  3 97]]
Logistic Regression : The accuracy on the test set is :  98.0 %
```

Figure 30: Logistic Regression code - Testing

The block of code used to visualize the confusion matrix on the test set and its outcome are shown in Fig 31 and Fig 32, respectively.

```
plt.figure()
ax = sns.heatmap(LR_cm, annot=True, cmap = 'Blues')
ax = ax.set(xlabel='Predicted',ylabel='True',title='Logistic Regression - Confusion Matrix',
            xticklabels=(directories),
            yticklabels=(directories))

plt.savefig('Logistic_Regression_Confusion_Matrix.png', dpi = 300)
```

Figure 31: Code to visualize Logistic Regression confusion matrix

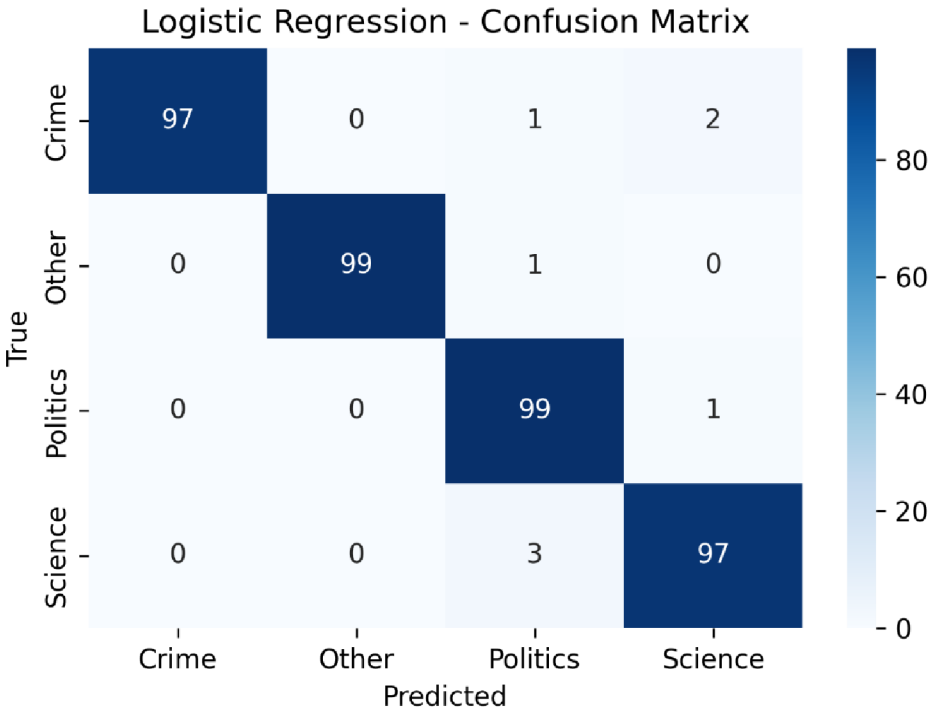


Figure 32: Logistic Regression confusion matrix on the test set

The confusion matrix shows that the total accuracy of the Logistic Regression model on the test set is 98% model. The confusion matrix also confirms that the balanced data lead to a relatively balanced performance of the trained model in each of the four classes. However, small differences between the accuracies in the four classes were spotted. Precisely, the accuracy in both the “Politics” and “Other” classes was 99% while the accuracy in the “Crime” and Science classes was 97%. This indicates that the linear hyperplane created by the Logistic Regression is more able to split the “Politics” and “Other” classes from the remaining classes compared to its ability to split the ”Crime” and “Science” classes from the remaining classes. However, such a performance is almost perfect, and we can state that there is a clear linear relationship/mapping between the inputs (the emails) and the output (the emails categories/classes). Nevertheless, we have trained more complex and non-linear models in the following subsections to confirm the possibility of reaching better performance.

4.2.2 Random Forest

In addition to the Logistic Regression model, a Random Forest model was trained for email classification and scikit-learn library from Python was used. The default Random Forest

parameters were kept (n-estimators (The number of trees in the forest.=100), max_depth=None, criterion='gini'). The code used is the following is shown in Fig.33:

```
# train random forest
RF_classifier = RandomForestClassifier(random_state = 0)
RF_classifier.fit(X, y_train)
RF_acc_training_data = 100*RF_classifier.score(X, y_train)
print('Random Forest : The accuracy on the training data is : ',  RF_acc_training_data, '%')
```

```
Random Forest : The accuracy on the training data is :  100.0 %
```

Figure 33: Random Forest code - Training

The reached total accuracy by the trained Random Forest model was 100% which is very high and indicates possible overfitting. However, this can be confirmed by testing the performance of the trained Random Forest model on unseen data. The code used for Random Forest testing (on the unseen test set) is shown below in Fig.34.

```
# predict test output using the traind model
RF_y_pred = RF_classifier.predict(X_test)

# confusion matrix
RF_cm = confusion_matrix(y_test, RF_y_pred)
print('Random Forest : The confusion matrix values are : ', RF_cm)

# accuracy calculation
RF_acc = 100*accuracy_score(y_test, RF_y_pred)
print('Random Forest : The accuracy on the test set is : ', RF_acc, '%')
```

```
Random Forest : The confusion matrix values are :  [[ 92   0   2   6]
 [  0 100   0   0]
 [  1   0  91   8]
 [  0   0   4  96]]
Random Forest : The accuracy on the test set is :  94.75 %
```

Figure 34: Random Forest code - Testing

Visualization of the confusion matrix on the test set:

The block of code used to visualize the confusion matrix on the test set and its outcome are shown in Fig 35 and Fig 36, respectively.

```
plt.figure()
ax = sns.heatmap(RF_cm, annot=True, cmap = 'Blues')
ax = ax.set(xlabel='Predicted',ylabel='True',title='Random Forest - Confusion Matrix',
            xticklabels=(directories),
            yticklabels=(directories))

plt.savefig('Random_Forest_Confusion_Matrix.png', dpi = 300)
```

Figure 35: Code to visualize Random Forest confusion matrix

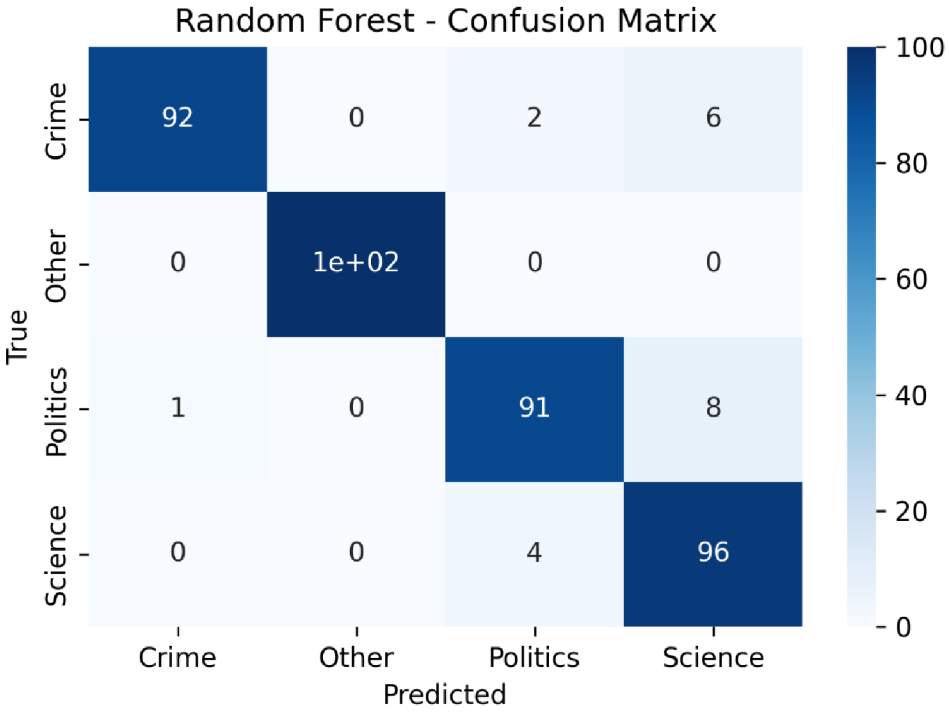


Figure 36: Random Forest confusion matrix on the test set

The total accuracy of the Random Forest model on the test set is 94.7 which is slightly low compared to the 100% accuracy on the training set. This confirms that the model slightly overfits the data. However, this may be expected due to the existence of a linear relationship between the input and the output as shown in the Logistic Regression results. Therefore, the use of a more complex non-linear model such as a Random Forest led to the pre-shown overfitting.

Going into more details from the confusion matrix, the Random Forest model has resulted more overfitting on the “Politics” and “Crime” classes (91% and 92% test set accuracy, respectively) compared to the “Science” class (96% test set accuracy). On the “Other”

class, the Random Forest model showed a perfect performance without any overfitting (100% test set as well as training set accuracies).

4.2.3 Naive Bayes

A Naive Bayes model was also trained for the purpose of comparison with the previously trained models. Similar to the previous models, the scikit-learn library from Python was use The code used is the Fig.37:

```
# train naive bayes
gnb_classifier = GaussianNB()
gnb_classifier.fit(X, y_train)
gnb_acc_training_data = 100*gnb_classifier.score(X, y_train)
print('Naive Bayes : The accuracy on the training data is : ',  gnb_acc_training_data, '%')
```

```
Naive Bayes : The accuracy on the training data is :  99.93738259236068 %
```

Figure 37: Naive Bayes code-Training

The reached total accuracy by the trained Naive Bayes model was 99.9% which is also very high and indicates possible overfitting. To confirm, the models' performance on the test set was checked similarly to the previous two models. The code used for Naive Bayes model testing (on the unseen test set) is shown in Fig.38.

```
# predict test output using the traind model
gnb_y_pred = gnb_classifier.predict(X_test)

# confusion matrix
gnb_cm = confusion_matrix(y_test, gnb_y_pred)
print('Naive Bayes : The confusion matrix values are : ', gnb_cm)

# accuracy calculation
gnb_acc = 100*accuracy_score(y_test, gnb_y_pred)
print('Naive Bayes : The accuracy on the test set is : ', gnb_acc, '%')
```

```
Naive Bayes : The confusion matrix values are :  [[96  2  2  0]
 [ 0 99  0  1]
 [ 3  2 95  0]
 [ 2  1  6 91]]
Naive Bayes : The accuracy on the test set is :  95.25 %
```

Figure 38: Naive Bayes code- Testing

The block of code used to visualize the confusion matrix on the test set and its outcome are shown in Fig 39 and Fig 40, respectively.

```
plt.figure()
ax = sns.heatmap(gnb_cm, annot=True, cmap = 'Blues')
ax = ax.set(xlabel='Predicted',ylabel='True',title='Naive Bayes - Confusion Matrix',
            xticklabels=(directories),
            yticklabels=(directories))

plt.savefig('Naive_Bayes_Confusion_Matrix.png', dpi = 300)
```

Figure 39: Code to visualize Naive Bayes confusion matrix

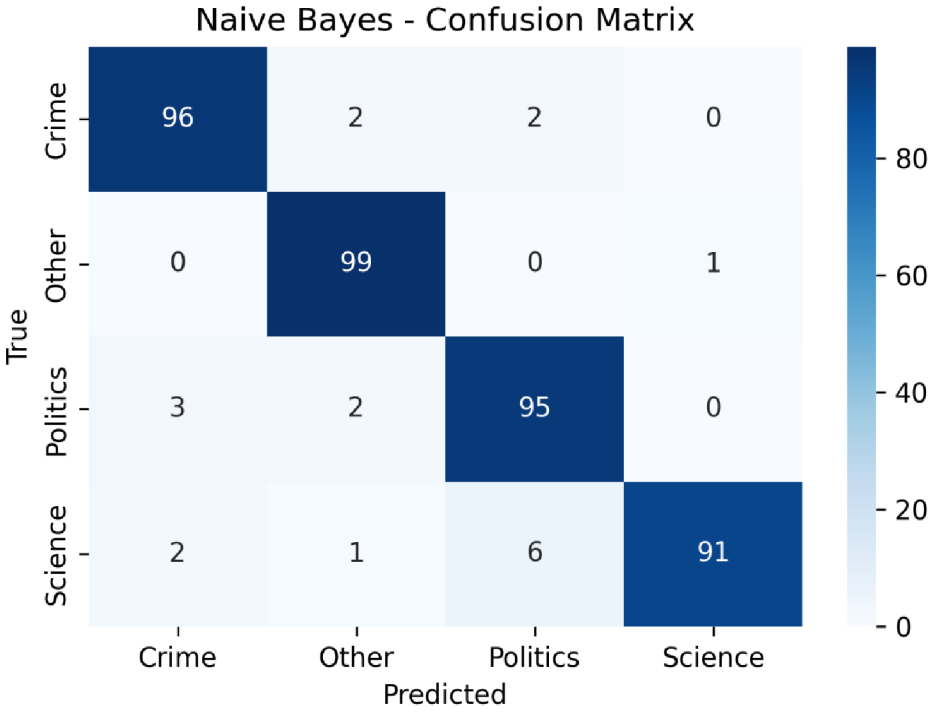


Figure 40: Naive Bayes confusion matrix on the test set

The total accuracy of the Naive Bayes model on the test set is 95.2 which is slightly low compared to the 99.9% accuracy on the training set. Although, this confirms that the model slightly overfits the data, the average overfitting is lower than the overfitting observed in the Random Forest model training.

Looking at each class separately in the confusion matrix, the Naive Bayes model has resulted more overfitting on the "Science" class (91% test set accuracy) compared to the "Politics" and "Crime" classes (95% and 96% test set accuracy, respectively). Similar to the case of Random Forest, the "Other" class shows (almost) a perfect performance without any overfitting.

4.2.4 Artificial Neural Networks

Finally, powerful Artificial Neural Networks (ANNs) were used to train an email's classification model. The selected structure of the trained ANN was an input layer, a hidden layer with five neurons and a RELU activation function, and a SoftMax output layer with four neurons (equal to the number of four possible classes/categories). The used loss function is the multi-class cross entropy suitable for multi-class classification problems. The prediction accuracy is used as the main metric for the evaluation of the model's performance. Keras library from Python was used for the modeling. The optimizer used for the minimization of the loss function and the weights' update is ADAM with the default initial learning rate set in Keras (The learning rate. Defaults to 0.001.). The number of epochs was set to 30 epochs. Note that the previous ANN structure and parameters were set based on the trial-and-error approach. The block of code used to train the ANN and the training progress are shown in Fig.41.

```
# ANN
ann_classifier = tf.keras.models.Sequential()
ann_classifier.add(tf.keras.layers.Dense(units=5, activation='relu'))
ann_classifier.add(tf.keras.layers.Dense(units=4, activation='softmax'))
ann_classifier.compile(optimizer="adam", loss="SparseCategoricalCrossentropy", metrics=["accuracy"])
hist=ann_classifier.fit(X, y_train, epochs = 30)
```

Figure 41: ANN code-Training

The training progress of the ANN (i.e., the loss function value and the reached accuracy at every training epoch) is shown in Figure 42 :

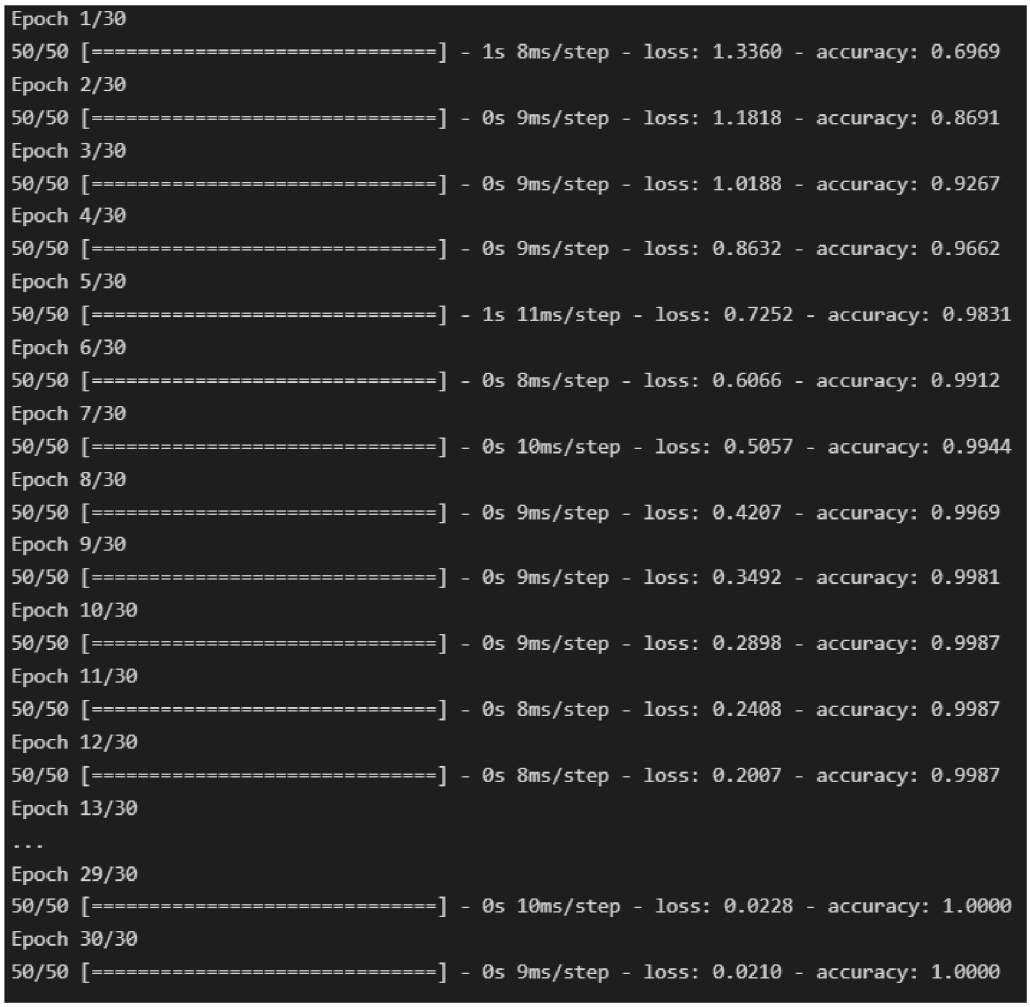


Figure 42: Training progress of the ANN

The training progress is monitored by following the loss reduction and the prediction accuracy increment over the training epochs. The block of code used in training progress over the training epochs is included in Fig 43 and Fig 44 showing the loss decrease and the accuracy increase .

```
plt.figure()
plt.plot(hist.history['loss'])
plt.ylabel('Loss')
plt.xlabel('Epoches')
plt.grid(True)
plt.savefig('ANN_loss_progress.png', dpi = 300)
plt.show()
```

Figure 43: Code to visualize the Loss

```
plt.figure()
plt.plot(hist.history['accuracy'])
plt.ylabel('Accuracy')
plt.xlabel('Epoches')
plt.grid(True)
plt.savefig('ANN_accuracy_progress.png', dpi = 300)
plt.show()
```

Figure 44: Code to visualize the Accuracy

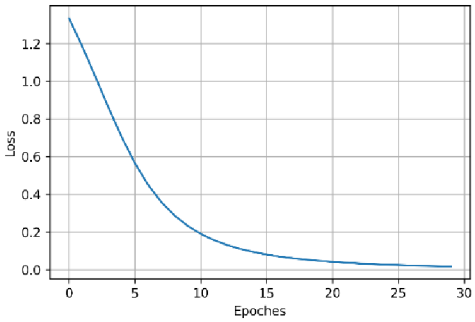


Figure 45: Loss progress

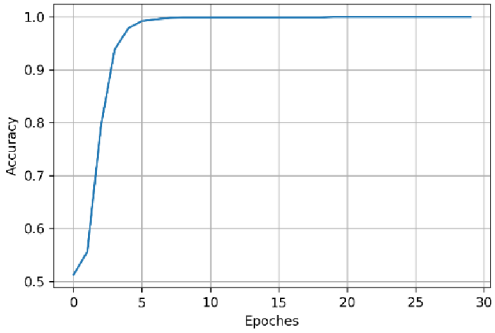


Figure 46: Accuracy progress

The prediction accuracy achieved by the ANN was 100%. Such a high accuracy on the training set motivates towards checking the accuracy in the test set to examine whether the model overfits the training data. The code used for ANN testing is shown in Fig.47

```
# ANN
# predict test output using the traind model
ann_y_pred = np.argmax(ann_classifier.predict(X_test), axis = 1)

# confusion matrix
ann_cm = confusion_matrix(y_test, ann_y_pred)
print('Artificial Neural Network : The confusion matrix values are : ', ann_cm)

# accuracy calculation
ann_acc = 100*accuracy_score(y_test, ann_y_pred)
print('Artificial Neural Network : The accuracy on the test set is : ', ann_acc, '%')
```

```
Artificial Neural Network : The confusion matrix values are :  [[98  0  2  0]
 [ 0 99  1  0]
 [ 1  0 99  0]
 [ 1  0  2 97]]
Artificial Neural Network : The accuracy on the test set is :  98.25 %
```

Figure 47: ANN code-Testing

The confusion matrix on the test set is shown below:

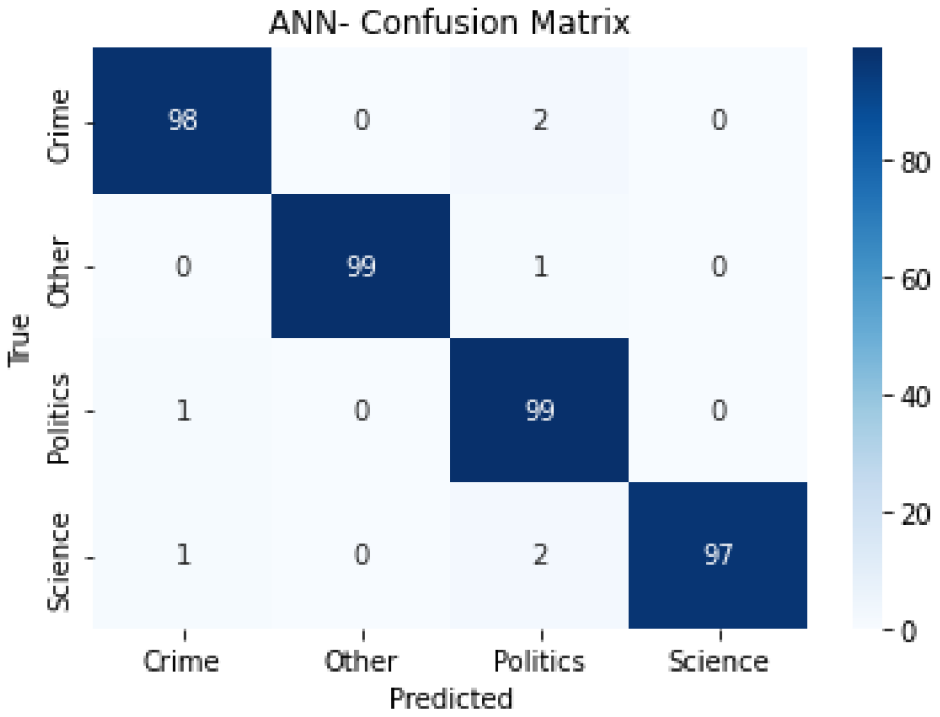
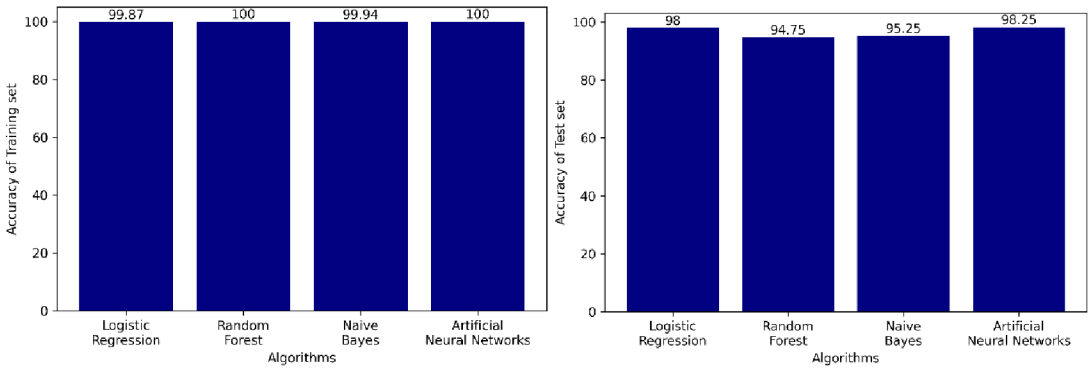


Figure 48: ANN test confusion matrix

As shown in the previous figure, the trained ANN was able to reach a very high total accuracy on the test set (98.25%). Moreover, the accuracy of the individual classes is balanced (only very small differences are spotted in the accuracies of each of the four classes).

4.3 Models' comparison

In order to compare the performance of the trained models, we show the total accuracy of each model on the training set (Figure 49. a) as well as the test set (Figure 49. b).



(a) Training set (b) Test set

Figure 49: Comparison between the trained models in terms of total accuracy

The accuracy of the training set for all four models is almost perfect. However, the accuracy of the test set varies between the models indicating the differences in the expected performance if the models would be deployed in production. On the first hand, the ANN and Logistic Regression both show a very high performance where the ANN slightly overcomes the Logistic Regression model but with a negligible difference. On the second hand, the Random Forest and Naive Bayes show lower performance compared to their performance on the training set indicating some overfitting as already discussed in the previous subsection. It is also worth reminding that the ability of Logistic Regression to reach such high performance without overfitting emphasizes the existence of a linear relation between the input and the output.

5 Conclusion

This thesis aims to build a classification model that is able to predict an email's topic based on that email's content. The practical benefit is to help companies to either know what to expect in an email and prioritize their readings of emails or to even direct each email to the corresponding right department or person.

The data was collected from Kaggle.com where 2000 emails were used in the analysis of this thesis. Each email was attached to a category that represents the label of this email. The four categories are "Crime", "Politics", "Science" and "Other". The data were preprocessed using the following steps. First, the data was split into a training and a test set. Then, Tokenization, data cleaning, and Lemmatization were applied. The words were then converted into a numerical form using Term Frequency-Inverse Document Frequency.

The preprocessed data were used to train various classification models. Logistic Regression, Random Forest, Naive Bayes, and Artificial neural networks were used. The four algorithms were tested on the test set in terms of accuracy (i.e., the percentage of correctly classified emails from the test set).

The confusion matrix of each model was used to evaluate the performance and also to compare the performance of the various models. The Logistic Regression algorithm was able to reach an accuracy of 99.87% on the training set and 98% on the test set. This very high performance indicates the presence of a linear relation between the input and the output in our classification problem. However, the non-linear model of Random Forest should have a high performance on the training set (100% accuracy) and a notably lower performance on the test set (94.75%). This suggests that Random Forest overfit the training data and, hence, would perform poorly (compared to the Logistic Regression for example) when deployed in production. The Naive Bayes results are quite similar to the Random Forest. The accuracy on the training set is 99.94% while the accuracy on the test set is 95.25% and overfitting occurred. Finally, the Artificial Neural Networks achieved the best prediction accuracy on the test set (98.25%) and a perfect accuracy on the training set (100%). However, the difference in the test set accuracy between the Artificial Neural Networks and the Logistic Regression is negligible.

Based on the performed analysis, it is recommended to use Logistic Regression or Artificial Neural Networks to solve the problem targeted in this thesis. The reason is that these two algorithms showed the highest performance on the test data without overfitting. Therefore, these two algorithms are expected to perform well on unseen data when put in production.

The analysis performed in this thesis also confirms the feasibility to apply email classification in various companies. The goals of email classification may vary even to other goals not included in this thesis such as directing every email to the correct person or department. The principles and steps followed in this thesis would remain the same anyway and only the business application would change.

6 References

IBM Cloud Education, 2020. *Deep Learning.* [Online]
Available at: https://www.ibm.com/cloud/learn/deep-learning#:~:text=Deep%20learning%20is%20a%20subset,from%20large%20amounts%20of%20data
[Accessed 11 07 2022].

10 NLP Techniques, 1. N. T. E. D. S. S. K., 2022. [Online]
Available at: https://www.projectpro.io/article/10-nlp-techniques-every-data-scientist-should-know/415
[Accessed 11 07 2022].

AI, W. I. A. I. D. &. S.-F. O., 2022. [Online]
Available at: https://www.softwaretestinghelp.com/what-is-artificial-intelligence/
[Accessed 07 06 2022].

Baheti, P., 2022. *Activation functions in neural networks.* [Online]
Available at: https://www.v7labs.com/blog/neural-networks-activation-functions
[Accessed 22 07 2022].

Bouargane, A., 2021. *HISTORY OF NATURAL LANGUAGE PROCESSING AND ITS DIRECTION FOR GROWTH.* [Online]
Available at: https://www.bbntimes.com/technology/history-of-natural-language-processing-and-its-direction-for-growth
[Accessed 05 07 2022].

Brownlee, J., 2020. *Difference Between Algorithm and Model in Machine Learning.* [Online]
Available at: https://machinelearningmastery.com/difference-between-algorithm-and-model-in-machine-learning/
[Accessed 28 06 2022].

CHEN, J., 2021. *Neural Network.* [Online]
Available at: https://www.investopedia.com/terms/n/neuralnetwork.asp
[Accessed 15 07 2022].

Chouinard, J. C., 2022. *TF-IDF: Term frequency-inverse document frequency.* [Online]
Available at: https://www.jcchouinard.com/author/jean-christophe-chouinard/
[Accessed 10 07 2022].

Géron, A., 2019. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems.* 2nd ed. Sebastopol: O'Reilly Media.

Gong, D., 2022. *Top 6 Machine Learning Algorithms for Classification.* [Online]
Available at: https://towardsdatascience.com/top-machine-learning-algorithms-for-classification-2197870ff501
[Accessed 25 06 2022].

Hutchins, J., 2004. *6th Conference of the Association for Machine Translation in the Americas.* Washington, Springer Berlin, Heidelberg.

Krishnan, S., 2021. *Biomedical Signal Analysis for Connected Healthcare.* 1 ed. s.l.:Academic Press,.

Lakshana, 2022. *Cross-Validation Techniques.* [Online]
Available at: https://www.analyticsvidhya.com/blog/2021/05/4-ways-to-evaluate-your-machine-learning-model-cross-validation-techniques-with-python-code/
[Accessed 11 07 2022].

Lamsal, R., 2021. *A step by step forward pass and backpropagation.* [Online]
Available at: https://theneuralblog.com/forward-pass-backpropagation-example/
[Accessed 04 11 2022].
Lichtig, R., 2011. *The History of Natural Language Processing.* [Online]
Available at: https://ethw.org/The_History_of_Natural_Language_Processing
[Accessed 02 07 2022].
Marr, B., 2021. *The Most Significant AI Milestones So Far.* [Online]
Available at: https://bernardmarr.com/the-most-significant-ai-milestones-so-far/
[Accessed 04 06 2022].
Mijwel, M. M., 2015. History of Artificial Intelligence. *Computer science, college of science, ,* 1(1), pp. 1-6.
Milestones, T. G. T. o. A., 2022. [Online]
Available at: https://achievements.ai/
[Accessed 05 06 2022].
Monroe, W., 2017. *logistic-regression.* [Online]
Available at: https://www.coursehero.com/file/38187601/220-logistic-regressionpdf/
[Accessed 20 07 2022].
Müller, A. C. & Guido, S., 2016. *Introduction to Machine Learning with Python: A Guide for Data Scientists.* 1 ed. Sebastopol: O'Reilly Media, Inc.
Narayan, L., 2019. *Machine Learning versus Artificial Intelligence.* [Online]
Available at: https://medium.com/@lalithnarayan/machine-learning-versus-artificial-intelligence-c9f2f87212e6
[Accessed 14 07 2022].
NLP Techniques, S. N. T. i. D. S. w. R., 2022. [Online]
Available at: https://techvidvan.com/tutorials/nlp-techniques-in-data-science/
[Accessed 10 07 2022].
Pawar, K., 2021. *Binary Cross-Entropy.* [Online]
Available at: https://insideaiml.com/blog/BinaryCross-Entropy-1038
[Accessed 04 11 2022].
Sharma, S., 2017. *Activation Functions in Neural Networks.* [Online]
Available at: https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6
[Accessed 24 07 2022].
Tableau, 2022. *https://www.tableau.com/learn/articles/natural-language-processing-examples.* [Online]
Available at: https://www.tableau.com/learn/articles/natural-language-processing-examples
[Accessed 06 07 2022].
Yathish, V., 2022. *Loss Functions and Their Use In Neural Networks.* [Online]
Available at: https://towardsdatascience.com/loss-functions-and-their-use-in-neural-networks-a470e703f1e9
[Accessed 04 11 2022].
Yiu, T., 2019. *Understanding Random Forest.* [Online]
Available at: https://towardsdatascience.com/understanding-random-forest-58381e0602d2
[Accessed 28 06 2022].

7 List of Figures and Tables

7.1 List of Figures

7.2 List of Tables