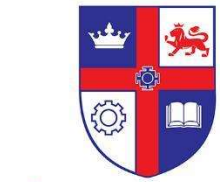
LONDON METROPOLITAN UNIVERSITY



Islington college
(इस्लिङ्टन कलेज)

Module Code & Module Title

CU6051NI Artificial Intelligence

Assessment Weightage & Type

75% Individual Coursework

2023-24 Autumn

Student Name: Amit Baniya

London Met ID: 21039840

College ID: NP01CP4A210096

Assignment Due Date: Wednesday, January 17, 2024

Assignment Submission Date: Wednesday, January 17, 2024

I confirm that I understand my coursework needs to be submitted online via MySecondTeacher under the relevant module page before the deadline in order for my assignment to be accepted and marked. I am fully aware that late submissions will be treated as non-submission and a marks of zero will be awarded.

Abstract

Spam has been one of the biggest problems in our lives, in this document we will be seeing, how AI and machine learning have helped us in different sectors. The AI industry is growing rapidly each year, and it is the future. Therefore, there are ways that we can implement it in our lives for a better living style. We will be discussing the small part of machine learning usage which is spam filtering. There are many ways to filter spam whether it is a traditional way by not using machine learning or by using machine learning. Nowadays, all the major companies use machine learning for spam filtering. However, the impact of spam is still high as spam covers half of our mail. Therefore, we will be researching about what are the possible machine learning algorithms that can be used to identify spam. After researching about the algorithms from different research papers done by others. We will be selecting three algorithms, learning about them, and seeing how we will be able to implement them to detect spam. We will be using a dataset found from the website called Kaggle which is one of the best places to find datasets. We have modelled the dataset with different algorithms in this documentation. Support Vector Machine (SVM), K-Nearest Neighbours (KNN) and Naïve Bayes algorithms has been used to create models. We have also tuned hyper parameter in order to extract more performance out of models. Finally, we have compared all the three algorithms and with each other to see which algorithms performs the best when it comes to spam detection.

Table of Contents

Table of Figures

List of Tables

Table of equations

1. Introduction

Artificial intelligence (AI) is the abilities of a computer or robot to carry out operations typically performed by intelligent entities by computer or robot themselves. The phrase is commonly used to describe the attempt of creating computer systems that include human-like cognitive functions, like reasoning, meaning-finding, generalization, and experience-based learning. It has been shown that computers can perform extremely complicated jobs, including finding proofs for mathematical theorems or playing chess, with remarkable proficiency ever since the digital computer was developed in the 1940s. Nevertheless, despite ongoing improvements in computer memory and processing power, no software can yet fully simulate human adaptability over a larger range of areas or in jobs requiring a great deal of common knowledge. However, in certain limited applications, such as medical diagnosis, computer search engines, voice, or handwriting recognition, and chatbots, artificial intelligence has evolved to the point where some programs can perform at the same levels as humans. (Copeland, 2023)

Figure 1 - AI and human (poptika/shutterstock.com)

Since defining intelligence can be difficult, AI specialists usually distinguish between strong and weak AI.

Strong AI

Strong artificial intelligence, sometimes referred to as artificial general intelligence, is the ability of a machine to solve problems like those on which it has never been trained. The robots from different movies are examples of this type of artificial intelligence (AI). There isn't really any AI like this yet. Though the complexity of gaining this level of achievement hasn't decreased over time, strong AI, in contrast to weak AI, indicates a computer with a full set of cognitive abilities and an equally large selection of use cases. (Schroer, 2023)

Weak AI

A form of artificial intelligence that is limited to a particular or narrow field is known as weak artificial intelligence (AI), also known as narrow AI. Human cognition is simulated by weak AI. It can help society by automating laborious tasks and doing data analysis in ways that people aren't always able to. We can compare weak AI to strong AI as the weak AI are theoretical kinds of computer intelligence comparable to human intelligence. (Frankenfield, 2022)

Some examples of Weak AI are as follows:

- Self-driving cars
- Google search
- Email Spam filters
- Recommendation systems
- Smart assistants like google home, Alexa etc.

Deep Learning vs Machine Learning

While the words "deep learning" and "machine learning" are often used in discussions about artificial intelligence, they should not be used synonymously. Machine learning, a branch of artificial intelligence, includes deep learning as one of its forms.

Deep Learning

A kind of machine learning called deep learning processes inputs via a neural network architecture that takes reference from biology. The data is processed through a number of hidden layers in neural networks, which enable the machine to learn "deeply," form connections, and weight input for optimal outcomes. (Schroer, 2023)

Machine Learning

Artificial intelligence (AI)'s machine learning (ML) field gives computers the capacity to autonomously learn from data and past experiences, finding patterns to generate predictions with little to no human input. Without explicit programming, computers can function independently thanks to machine learning techniques. Applications for machine learning are fed fresh data and have the ability to learn, grow, evolve, and adapt on their own. (Kanade, 2022)

Types of Machine Learning Techniques

There are three types of machine learning techniques, and they are:

Supervised Learning

The target value is chosen beforehand by supervised learning algorithms, which map labelled inputs to known outputs. Machine learning models are trained using supervised learning techniques, which require outside supervision. Thus, the term "supervised." To achieve the intended result, they require instruction and further details. Supervised learning can be used in filtering spams, prediction weather conditions, share market etc. (Menon, 2023)

Unsupervised Learning

A machine learning that trains algorithms using unlabelled data. Data without labels lacks a specific output variable. After learning from the data and identifying its patterns and features, the model produces an output. In order to figure out the output, unsupervised learning analyzes trends and patterns in the data. In considering this, the model attempts to label the data using the attributes of the input data. Unsupervised

learning approaches do not require supervision during the training stage in order to construct models. They make predictions and pick up knowledge on their own. They can be used to segment customers in businesses, identification of accident-prone areas, classification of heavenly bodies. (Menon, 2023)

Reinforcement Learning

A machine can be trained through learning to maximize its rewards and take the appropriate actions in a given scenario in this kind of learning technique. It generates actions and rewards by utilizing an agent and an environment. There are start and finish states for the agent. However, there may be multiple routes, much to a maze, to arrive at the destination. This learning method does not have a predetermined target variable. It can be used to make AI in games, or make robots do human tasks. (Menon, 2023)

Types of AI

AI are divided in four types, and they are as follows:

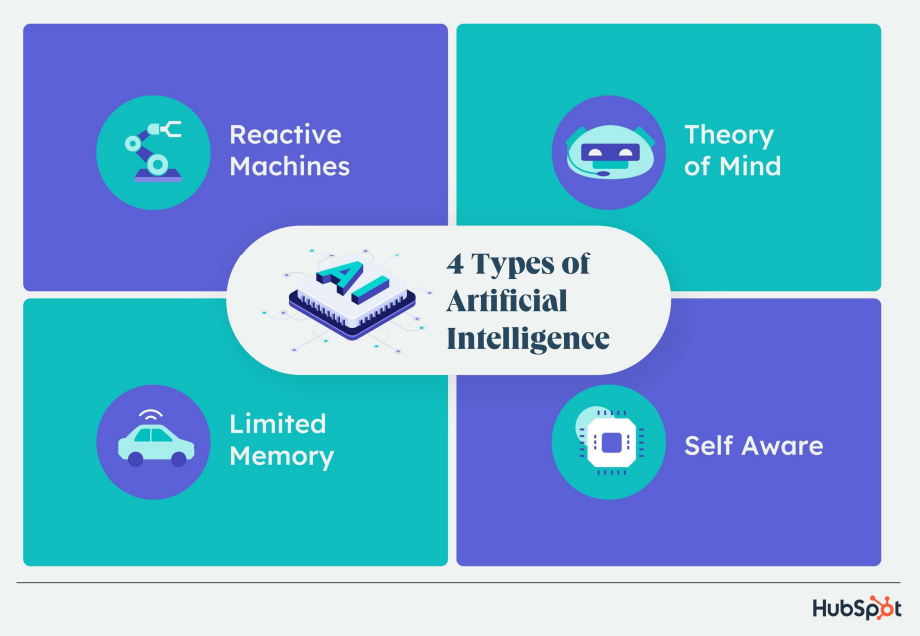


Figure 2 - Types of AI - (Erin Rodrigue,Flori Needle, 2023)

Reactive Machines

Reactive machines, as the name indicates react and respond to various inputs. It achieves this without the aid of recollection or a deeper awareness of the situation. In game design, this kind of AI is frequently utilized to create opponents. The opponent is not aware of the main goal of the game, but they will react to your moves, attacks, and activities in real time. Furthermore, because it lacks memory, it is unable to draw lessons from the past or modify its gameplay. Numerous marketing solutions are powered by reactive AI. Chatbots are one such instance. Reactive AI is used by these algorithms to provide the appropriate information in response to messages or inputs. (Erin Rodrigue,Flori Needle, 2023)

Limited Memory

In order to process information and weigh possible actions, limited memory AI can store historical facts and forecasts. This is effectively looking into the past for hints about what might happen next. Reactive machines are less complex than limited memory AI, which offers more opportunities. Limited memory AI is developed when a group of people consistently trains a model to evaluate and make use of fresh data, or when an environment is developed specifically for AI to enable automatic model renewal and training. Self-Driving cars can be said to be the example of Limited Memory AI. (Schroer, 2023)

Theory of Mind

This type of AI as the name suggests itself, it only exits as a concept. It stands for a specialized category of technology capable of interpreting human brain states. For example, Google Maps does not become angry or provide emotional support if you yell at it because you missed a turn. As a response, it chooses another path.

The goal of theory of mind is to build machines that, by understanding human wants, objectives, and motivations, will be better able to interact with humans. An AI system, for instance, can react more delicately if it realizes the annoyances of a dissatisfied consumer. Theory of mind AI has the potential to significantly impact marketing in the long run. However, it's still in the early phases, so it's hard to say when it will happen. (Erin Rodrigue,Flori Needle, 2023)

Self-Awareness

The last stage of AI development will be self-awareness, which will happen long after theory of mind has been created. This type of artificial intelligence is conscious on the same level with humans, recognizing both its own presence and the presence and emotional states of others. It would have the ability to comprehend what other people could require based on both what they say to them and how they say it.

In order to include self-awareness into AI, human researchers must first grasp the fundamentals of consciousness before figuring out how to replicate it in machines. (Schroer, 2023)

1.1 Explanation of topic and concepts

1.1.1 Artificial Intelligence and Machine Learning

Artificial intelligence, essentially, is the ability of a machine to replicate intelligent human behaviour and machine learning is the subset of the artificial intelligence, meaning machine learning is AI. Systems using artificial intelligence are utilized to carry out difficult jobs in a manner comparable to how people solve issues. AI aims to build computer models which demonstrate "intelligent behaviours" identical to those of humans, as stated by CSAIL main research scientist and leader of the InfoLab Group Boris Katz. This includes devices that can identify an image, read a document written in natural language, or carry out a task in the real world. (Brown, 2021)

Artificial intelligence is a concept that has been widely used in computer science and related fields. Recent advancements in machine learning and artificial intelligence have led to a significant increase in the popularity of this term. Machine learning is the branch of artificial intelligence where machines are thought to be smarter than people and are in charge of automating everyday tasks. Robotics and the Internet of Things have elevated machines to a new degree of intelligence and thought, to the point that they can now outsmart people. They are known to learn, adapt, and function far more quickly than humans are expected or programmed to. (Pedamkar, 2023)

The AI industry has grown significantly these past few years. As per below image from, the Statista Market Insights, the AI market reached about 108 billion USD dollars in

2020 and increased by double in 2021 which was about 227.61 billion USD dollars. However, due to the Russia-Ukraine war the market of AI declined to 134.64 billion dollars. However, it is projected that by the end of 2023 the market of AI will reach whopping 241.90 billion USD Dollars. And with this rate it will reach about 738 billion USD dollars by the end of 2030. This shows us how AI is growing rapidly and how important it is going to be in our lives. (Insights, 2023)

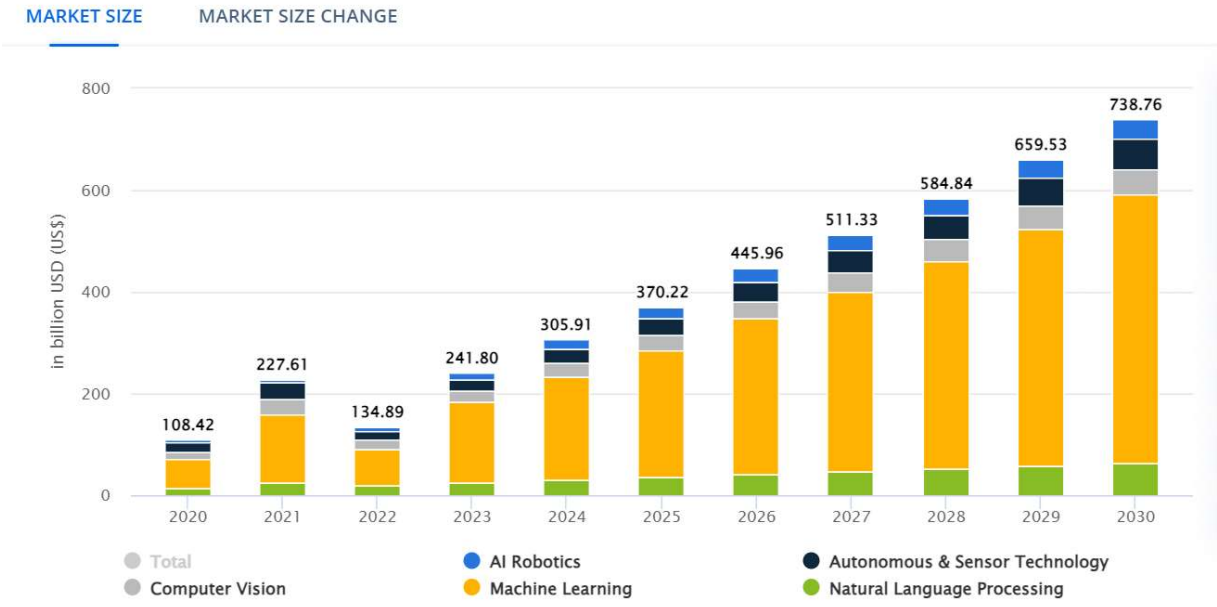


Figure 3 - Market Size of AI (Insights, 2023)

Artificial Intelligence has made a big impact in our lives for past few years now. Machine learning is being used by analysts in the financial services industry to identify fraud, automate trading processes, and offer clients financial advice. There are several ways that machine learning can help businesses increase their efficacy, productivity, and services. For example, chatbots enable companies to offer more responsive, expedient customer care without requiring them to use call centers or put clients on hold until a representative becomes available. (Engineering, n.d.)In numerous airplanes, artificial intelligence has been used to do tasks like as navigation maps, taxing routes, and a fast

inspection of the complete cockpit panel to guarantee that every component is operating correctly. This has been supported a lot because it has been producing really encouraging results. (Pedamkar, 2023)Machine learning can be used in the healthcare industry to assist hospitals and personnel detect and identify infectious diseases more effectively, customize medical treatments, and accelerate administrative procedures. (Engineering, n.d.)

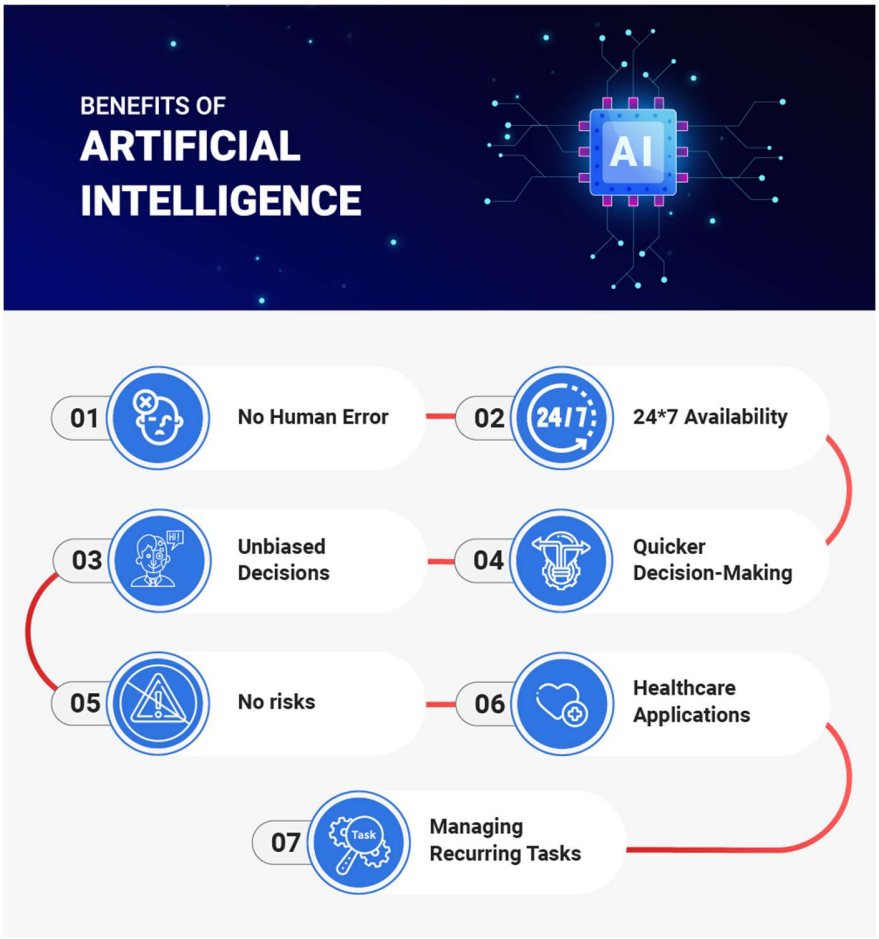


Figure 4 - Importance of AI (Oza, 2021)

1.2 Explanation of chosen topic/domain

For any unwanted message, it is known as spam. Although spam is primarily junk email, it can also be texts, calls, or messages on social media or even can be phone messages. Spam's the end goal is to convince an individual to open a message and purchase a good or service—which may or may not be fraudulent. Alternatively, some

spammers try to trick customers into downloading a virus onto their systems in order to either steal important data or demand money from the victim. Spammers use bulk emailing to carry out phishing frauds. (Pickle, 2022)

Figure 5 - Spam (Gatefy, 2021)

The process of identifying and preventing unsolicited, undesired, and potentially virus-filled emails from getting into email inboxes is known as spam filtering. Spam filtering technology is used by Internet service providers (ISPs) and small- to medium-sized businesses (SMBs) to protect their networks, consumers, and staff against potential dangers. (MailChannels, n.d.)

Importance of Spam filtering

As per the data found in Statista published by Ani Petrosyan and shown below in the diagram, it was found that 48.63% of email sent was a spam mail in 2022. But when compared to the 80.26% spam email of 2011, it is a way better number. However, the email spam percentages are almost half, meaning you are getting only 50% email that are not spam. Therefore, it is still a very large number of spam mails. And this doesn't even count towards other spam sources. (Petrosyan, 2023)

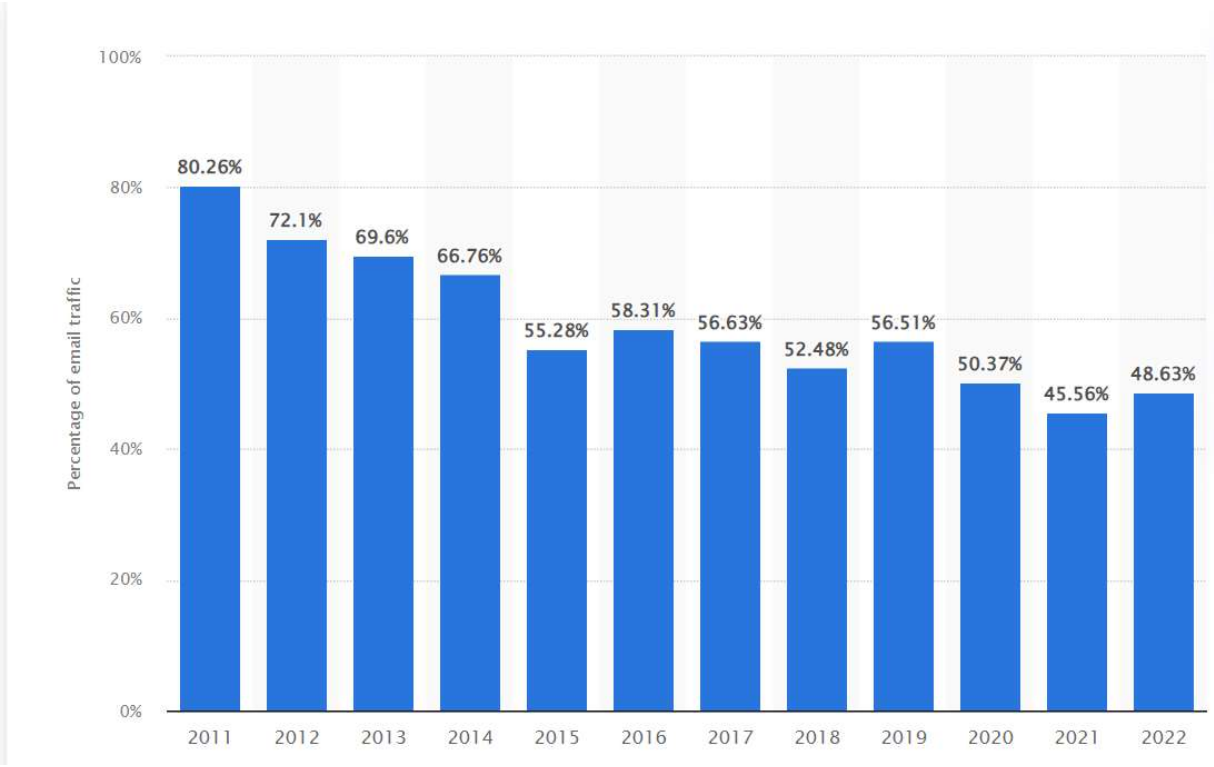


Figure 6 - Email Spam percentage (Petrosyan, 2023)

Since, there is this large percentage of spam mails in our lives every year, without spam filtering, we would potentially be scammed or tricked into buying something that we don't want to. For these, reasons spam filtering is required in our day to day lives.

2. Background

2.1 Research work done on the chosen topic/problem domain.

2.1.1. Spam Filtering

A great protection against unwanted messages (or, in the case of robocalls, phone calls) is known as spam filtering. Spam filtering helps detect and stop unwanted, unsolicited, and virus-filled emails (often referred to as spam) from entering email accounts. (Slavin, 2022)

Figure 7 - Spam Filtering (Slavin, 2022)

Traditional spam filtering without using machine learning.

To determine whether an incoming email is spam or not, non-machine learning techniques are utilized. These techniques primarily consist of a list of email addresses and a list of terms that are used to determine whether a given email is spam or not. List- and content-based spam filters are examples of non-machine techniques. While content-

based filters identify based on the email's content, list-based filters do so based on a specified list. (Sandhi Kranthi Reddy, T Maruthi Padmaja, 2019)

Limitations of spam filtering without using machine learning

This way of spam filtering has significant drawbacks despite their overall effectiveness. These systems are less able to deal with newer and complex types of spam because they are unable to adjust to changing spamming patterns and methods. In addition, as spam increased in quantity and variety, it became more difficult to manage and update the regulations. (Md Rafiqul Islam, Morshed U. Chowdhury, 2015)

2.1.2. Machine Learning in Spam Filtering

The field of spam filtering saw a change in direction with the introduction of machine learning. Machine learning algorithms could be trained to identify patterns and features in data rather than depending on pre-established rules, which would allow them to adapt dynamically to evolving spam strategies. The ability to adapt greatly increased spam detection's effectiveness and accuracy. We can see in the figure below that the spam has decreases significantly after the year 2016 as this is the time when spam filtering software using machine learning has been started.

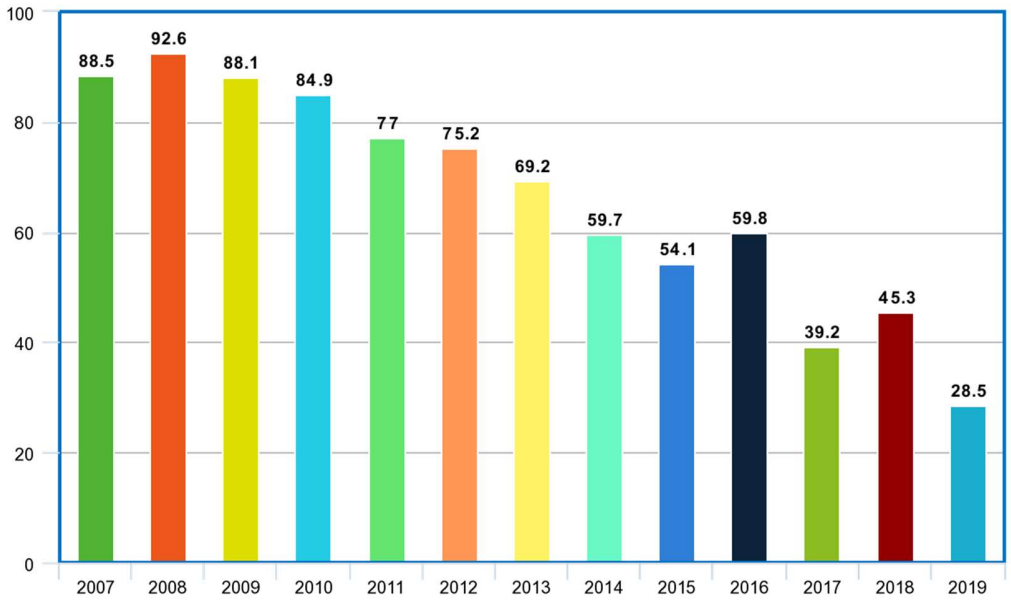


Figure 8 - Machine Learning spam protection (Raj, n.d.)

2.1.3. Advantages of Problem Domain

2.1.3.1. Accuracy

Spam filters can obtain extra information about the sender and their intentions by considering the website name. This aids in separating reputable emails from spam emails that have been cleverly disguised and may have faked well-known sender identities.

2.1.3.2. Reduced clutter

Users can keep their inbox free of unnecessary messages and make it simpler to locate critical emails by filtering spam emails.

2.1.3.3. Security

Spam emails are frequently used by malicious individuals to spread malware and phishing scams. Blocking spam might help in preventing users from becoming victims of these frauds.

2.1.3.4. Reduced Server Load

Server resources might be severely strained by spam emails. Reducing server load and enhancing overall system performance can be achieved by blocking spam.

2.1.3.5. Protects against scams.

Spam filters are essential for protecting people against phishing attempts and other internet scams that try to obtain personal information.

2.1.4. Disadvantages of Problem Domain

2.1.4.1. False Positives and Negatives

Sometimes spam filters can wrongly classify emails that are authentic as spam. Also, sometimes, Spam emails may manage to get past your security measures when spam filters are unable to recognize them.

2.1.4.2. Privacy

Privacy concerns may arise from the fact that spam filters frequently gather and examine your personal data.

2.1.4.3. Technical Challenges

It's challenging to stay on top of the most recent threats because spammers are always coming up with new ways to get around spam filters.

2.1.4.4. Over blocking

Over blocking may occur when spam filters are overly strict, preventing both spam and legitimate emails from getting through, which may affect communication.

2.1.5. Examples of spam filtering used by email service providers.

2.1.5.1. Gmail

Gmail is a popular email service provider by Google. It is one of the email service providers which uses machine learning to detect and block spam.

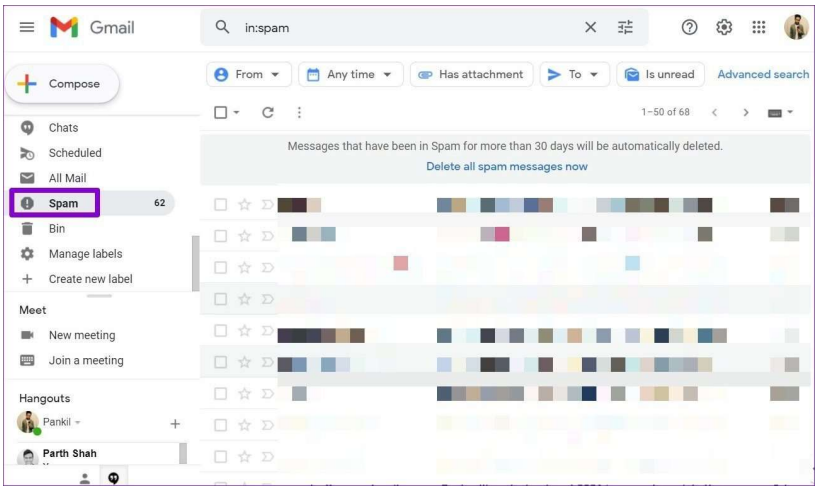


Figure 9 - Gmail Spam (Shah, 2022)

2.1.5.2. Outlook

Outlook is another popular email service provider by Microsoft which also uses machine learning algorithms to detect spams, or they call it junk.

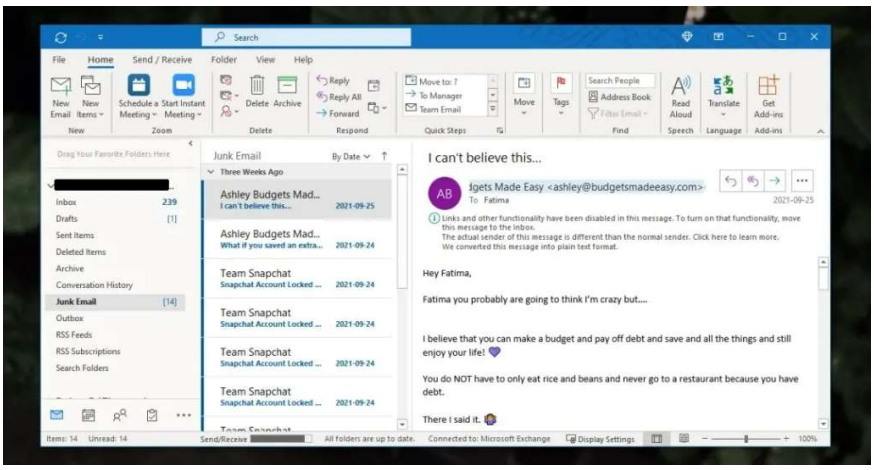


Figure 10 - Outlook Spam (Wahab, 2021)

2.1.5.3. Yahoo Mail

This is another example of email service providers which also uses machine learning to detect and block spams.

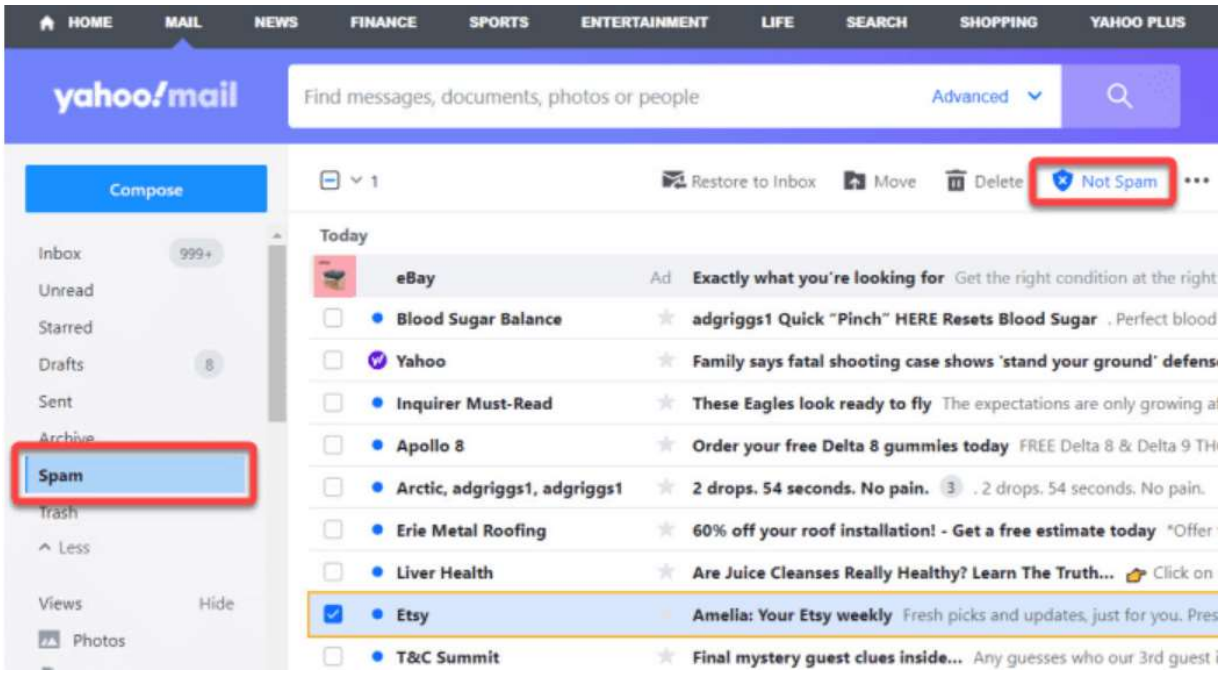


Figure 11 - Yahoo Spam (GRIGGS, 2022)

Despite all the email service provider use machine learning to spam filtering, still they haven't perfected the filtering, as we see that we must sometime, go to spam to find the mails that are not spam, or some mails that are spam can be found in the main email section. There, there is need of constant research to get more effective system that will help detect spam or ham mails.

2.1.6. Dataset

The dataset we are going to use to train the model is taken from Kaggle. This dataset contains three columns which is numbering (unnamed in the dataset), body, and labels. The dataset is not clean dataset which means before doing anything with it, we need to clean or prepare it. There are empty rows and numerous things to be removed before extracting the bag of words from it in order to train models. The dataset is extracted from: https://www.kaggle.com/datasets/nitishabharathi/email-spam-dataset

The main two columns that we will be using are:

- **Body:** All the email contents are found in this column which needs to be converted into bag of words.
- **Label:** The label has two numbers 0 and 1 in which 0 means not a spam mail and 1 means a spam mail.

Name	Description	Datatype
Body	All the contents of emails that are spam and are not spam.	Object
Label	Identification of body whether it is spam or not in 0 and 1.	Integer

Table 1 - Dataset Description

2.2 Review and analysis of work in the problem domain

2.2.1. Research Paper 1

Title: SMS Spam Detection using Machine Learning and Deep Learning Techniques

Authors: Sridevi Gadde, A. Lakshmanarao, S. Satyanarayana

Website Extracted from: https://ieeexplore.ieee.org/document/9441783

Citation: (Sridevi Gadde, A.Lakshmanarao, S.Satyanarayana, 2021)

Summary

This research paper basically explains how in this modern age where the number of phones is increasing day by day in fact it has reached 3.8 billion users of mobile users in just five years from 1 billion users. As per the research, the aim is to see how different machine learning can be used to detect spam and how they compare to each other.

In their research paper, they did their own research where they found out they that deep neural network and Hidden Markov Models (HMM) is the most accurate in finding spams with accuracy of 98%. However, as their aim is to find the effectiveness of ML algorithms themselves, they took a dataset from UCI repository and cleaned it using NLP and trained models for different algorithms and compared them to each other. They found out that Logistic Regression and Decision Tree is the most accurate algorithm to detect spam with the accuracy of 94%. And the SVM was second in their list with 93%. But they saw that the dataset was unbalanced. After balancing the dataset using a sampling technique called SMOTE (Synthetic Minority Oversampling), they achieved that Logistic regression is still at the top with accuracy of 95% whereas the SVM came second with 94%.

2.2.2. Research Paper 2

Title: Detection of Email Spam using Machine Learning Algorithms: A Comparative Study

Authors: Prazwal Thakur, Kartik Joshi, Prateek Thakral, Shruti Jain

Website Extracted from: https://ieeexplore.ieee.org/document/10009149

Citation: (Prazwal Thakur, Kartik Joshi, Prateek Thakral, Shruti Jain, 2022)

Summary

Here, in this paper, they are talking about how internet is a very integral part of your lives in this era and how there are so much internet spam. However, they want to focus on the email spam part of the internet spam. This research paper wants to detect spam using machine learning techniques rather than the knowledge engineering method of spam detection and filtration. This document is more through on the process of training a model with the flowchart and description of the algorithms as well as how algorithms compared to each other. At last, they chose, SVC (Support Vector Classifier), KNN (K-Nearest Neighbours), Logistic Regression, Naïve Bayes, and Decision Tree. In the test, they came to conclusion that SVC or SVM came at the very top with the accuracy of 98.09%. However, we wouldn't know which one of these algorithms came in second as they are describing KNN to be the second with accuracy of 95.3% however, in the table it is shown that Logistic Regression is second with the accuracy of 95.59%. However, we know that the SVM is the most effective in finding the spam in this research paper.

2.2.3. Research Paper 3

Title: A Comprehensive Review on Email Spam Classification using Machine Learning Algorithms

Authors: Mansoor RAZA, Nathali Dilshani Jayasinghe, Muhana Magboul Ali Muslam

Website Extracted from: https://ieeexplore.ieee.org/document/9334020

Citation: (Mansoor RAZA, Nathali Dilshani Jayasinghe, Muhana Magboul Ali Muslam, 2021)

Summary

The research paper is basically about how different algorithms or sets of algorithms can be used to detect spam and how they compare to each other. Here, it is described that nearly 50 percent of emails or unwanted emails among the 40 -50 emails a person gets in a day. They have also highlighted that almost, 3.5 USD million dollars are earned by spammers every year. As per the FBI, in the year 2019, the businesses financial loss due to phishing and spam is over 12.5 USD billion dollars. Here, in the document, they want to see how, and which machine learning are popular to detect spams. Here, unlike the other research papers, they have also done research on unsupervised machine learning approach for spam detection.

In their test, they have checked multiple groups of algorithms for their accuracy. Only two out of twelve are the algorithms that are not in groups to find the accuracy. However, at last they came to conclusion that the two algorithms are highly favoured algorithms for spam detection. They also found out that, multi algorithms are used for better outcome that using a single algorithm.

2.2.4. Research Paper 4

Title: A Study of Machine Learning Classifiers for Spam Detection

Authors: Shrawan Kumar Trivedi

Website Extracted from: https://ieeexplore.ieee.org/document/7743279

Citation: (Trivedi, 2016)

Summary

Here, in this paper different types of machine learning classified have been described thoroughly with tall the formulas and how it works. The classifiers are SVM, Naïve Bayes, Bayesian, J48(Decision Tree), along with the same classifier but using it with adaboost to re-assess the accuracy. The comparison showed that the accuracy of SVM is the highest of them all coming at 93.3%, after that Naive Bayes with boosting came at second with 93.2% of accuracy. It seems that SVM is the best classifier to be used and the research paper also concludes that SVM is the algorithm to be used for spam detection. And the lowest accuracy was of Bayesian with the score of 92.0%. Therefore, the Bayesian should not be used comparatively to others as the research paper shows.

2.2.5. Research Paper 5

Title: Evaluating the Effectiveness of Machine Learning Methods for Spam Detection

Authors: Kingshuk Debnath, Nirmalya Kar

Website Extracted from: https://ieeexplore.ieee.org/document/9850588

Citation: (Kingshuk Debnath, Nirmalya Kar, 2022)

Summary

This research like other research is also about seeing what technique the best is in finding spam, however, here they also focused on deep learning along with machine learning unlike other research papers where the machine learning is only implemented. While researching they found out that, there are 45.1% spam email traffic in March of 2021. They have described that their goal is to make a model using machine learning as well as deep learning to have more precision than the previous models. They found from other research that SVM and Naïve Bayes are the algorithms with a high accuracy. However, to meet their goal they have to tests of their own. Therefore, they took a dataset from Enron, and pre-processed the data and used it to create models of different machine learning algorithms as well deep learning algorithms.

In their test, they found out that the MNB (Multinominal Naïve Bayes) algorithm has the most accuracy with 98.13% while SVM was second on the list with 98.06%, which when they compared to another research that they did was higher. Also, among the deep-learning algorithms, the BERT (Bidirectional Encoder Representations from Transformers) deep-learning technique was the most effective with 99.14% accuracy. As per the result, they did succeed in their goal. However, they didn't compare the precision score which was the main aim.

2.2.6. Summarized Review and analysis

Together, the five studies research machine learning methods for spam recognition. Naïve Bayes, SVM, KNN, Neural Networks, Decision Trees, Random Forests, TF-IDF, BART, Logistic Regression and many others are the algorithms that have been examined among this five research. Every research analyzes the effectiveness of various algorithms and highlights the importance of filtering to reduce spam.

Research 1 shows Logistic Regression is the most accurate in their own tests (95%) and found that SVM was the second most accurate. They determined that SVM and Logistic Regression performed better than other methods.

In Research 2, The best algorithms are found to be SVM in the top and Logistic Regression and KNN are in the second.

The third study compares different multi-algorithms techniques as well as single algorithms techniques for email spam detection. They concluded that SVM and Naïve Bayes are the most popular and wanted algorithms even in the multi algorithms techniques.

Research 4 provides analysis of various classifiers, giving SVM the highest ranking (93.3%) and indicating that it is the recommended algorithm for spam detection.

Research 5 compares both machine learning and deep learning algorithms. The most accurate machine learning algorithms is Multinomial Naïve Bayes with 93.13%, closely followed by SVM (98.06%). In the deep-learning technique, the BERT came in top with 99.14%.

To sum up, studies frequently show that Naïve Bayes and SVM are efficient algorithms for spam identification, beating other approaches and showing high accuracy and precision.

3. Solution

3.1 Proposed Approach to solving the problem.

The proposed approach in solving this problem of spam is to use machine learning algorithms to find the spam and filter them. As per the research we have done above, we can see that Support Vector Machine (SVM) and Naive Bayes are the two of the algorithms that were on top in almost all the research. We will also be these two machine learning algorithms along with k-Nearest Neighbours (KNN) to detect the spams. We will be doing our own test checking which one of the three algorithms are the most accurate in order to find the right algorithm as a solution for spam detection.

3.1.1. Elaboration of the proposed Algorithms

3.1.1.1. Support Vector Machine (SVM)

The goal of SVM supervised machine learning algorithm is to identify the hyperplane that divides the two classes the best. Regression and SVM may seem similar, but they are not the same. whereas both techniques aim to identify the optimal hyperplane, the primary distinction between them is that support vector machines rely on statistical methods, whereas logistic regression takes a probabilistic approach.

Since the margin in SVM is calculated using the points that are closest to the hyperplane (support vectors), we don't need to worry about additional observations; in logistic regression, on the other hand, the classifier is defined over all of the points. As a result, SVM naturally speeds faster. Let's use an example to better understand how SVM functions. Assume we have two classes in our dataset (green and blue). The new data point needs to be categorized as either blue or green. The primary goal of SVM is to find the best hyperplane, which is the plane with the maximum distance from both classes. This is accomplished by identifying many hyperplanes that best classify the labels, after which it selects the hyperplane that is the furthest from the data points or that has the largest margin. We can see a hyper plane that best classify the labels in the below diagram. (Saini, 2023)

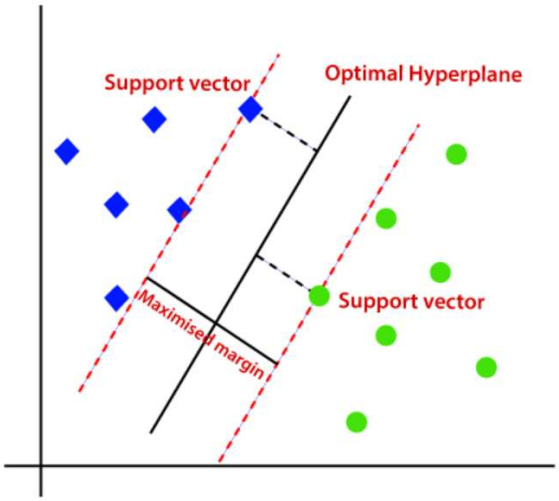


Figure 12 - SVM example (Saini, 2023)

3.1.1.1.1. Hyperparameters to be used

I. C (Regularization parameter)

It manages the trade-off between accurately identifying the training points and having a smooth decision boundary. An increased complex decision boundary will result from higher values of C, which may cause the training data to be overfit.

II. Kernal

By transforming the input data into a higher-dimensional space using kernels, Support Vector Machines (SVM) enable the identification of a hyperplane that divides the data into distinct groups. The shape of the decision border in SVM is determined by the kernel selection, which is very important. We will choose between linear, rbf, polly.

III. Gamma

The hyperparameter "gamma" is a parameter for some kernel functions in the context of Support Vector Machines (SVM), namely the Radial Basis Function (RBF) kernel. A single training example's influence can be determined by its gamma parameter, where low values indicate "far", and high values indicate "close." Put otherwise, a choice boundary that is more generalized will be produced by a small gamma, whereas a decision boundary that is more localized will be produced by a large gamma. We will be either using scale, auto or any numbers.

3.1.1.2. Naïve Bayes

This is a popular supervised machine learning approach for classification applications like text classification. It is a member of the generative learning algorithm family, which implies that it simulates the input distribution for a certain class or category. This method relies on the assumption that the input data's attributes are conditionally independent given the class, which enables the algorithm to produce precise and fast predictions.

Naive Bayes classifiers are regarded as straightforward probabilistic classifiers in statistics that make use of the Bayes theorem. The probability of a hypothesis, given the facts and some prior information, is the basis of this theorem. The naive Bayes classifier assumes that every characteristic in the input data is unrelated to every other feature, which is frequently false in practical applications. The naive Bayes classifier is nevertheless frequently used because to its effectiveness and strong performance in numerous real-world applications. (Ray, 2023)

Using P(c), P(x), and P(x|c), one can calculate the probability P(c|x) using the Bayes theorem. and the equation is as follows:

$$P(c|x) = \frac{P(x|c).P(c)}{P(x)}$$

Equation 1 - Bayes equation

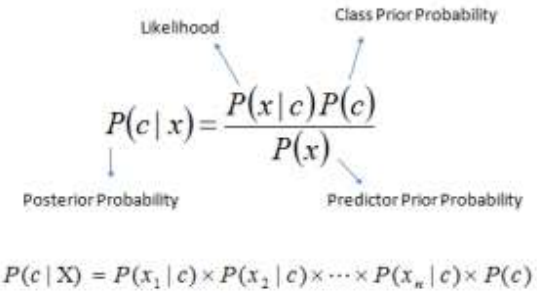


Figure 13 - Bayes equation (Ray, 2023)

where,

- P(c|x) represents the class (c, target) posterior probability given a predictor (x, characteristics).

- The prior probability of the class is P(c).
- The likelihood, or probability of the predictor given class, is expressed as P(x|c).
- The predictor's prior probability is denoted by P(x).

3.1.1.3. K-nearest Neighbours (KNN)

The k-nearest neighbours' algorithm is a non-parametric supervised learning classifier that groups individual data points based on closeness in order to classify or predict data. Although it can be applied to classification or regression issues, it is usually employed as a classification algorithm, based on the idea that comparable points can be located next to each other. The k-nearest neighbour algorithm seeks to locate a query point's closest neighbours so that a class label can be applied to it.

The distance between a query point and the other data points must be computed in order to figure out which data points are closest to a particular query point. The decision boundaries that divide query points into various areas are formed in part by these distance measurements. While there are several distances measures, we will be seeing the equation of Euclidean distance.

Euclidean distance is the most widely used distance metric, although it can only be applied to vectors with real values. It measures a straight line between the query location and the other point being measured using the formula below. (IBM, n.d.)

$$d(x,y) = \sqrt{\sum_{i=1}^{n} (y_i - x_i)^2}$$

Equation 2 - Euclidean distance equation

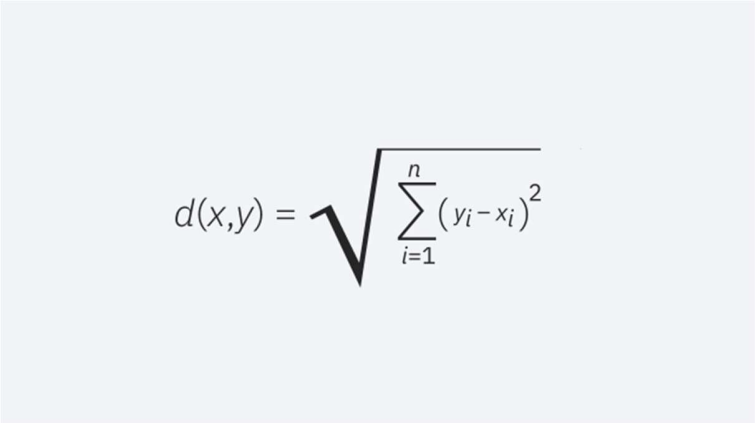


Figure 14 - Euclidean distance equation (IBM, n.d.)

where,

- d represents the distance
- x represents the first point
- y represents the second point
- i represent the index to iterate over coordinates.
- n is number of dimensions

3.1.1.3.1 Hyperparameters to be used

I. N_neighbours

This hyperparameter indicates how many neighbours to consider while generating prediction. It is an important parameter, and the dataset's properties determine what its ideal value should be. Odd numbers are frequently selected in binary classification issues in order to prevent ties.

II. Weights

It determines the contribution of the neighbours to the prediction. Usually, there are two choices:

- **Uniform** means that each neighbour adds the same amount to the prediction.
- **distance**: Weighted by the inverse of their distance, closer neighbours have a bigger impact on the forecast than faraway ones.

III. P

The power parameter for the Minkowski distance metric. When p=1, it corresponds to the Manhattan distance (L1), and when p=2, it corresponds to the Euclidean distance (L2). The default is usually p=2.

3.1.2. Implementation methodology of used Algorithm in the system

3.1.2.1. Data Preparation

- Dataset cleaning and preparation by taking care of missing values and eliminating unnecessary data.
- Feature extraction to make it machine-learning-friendly.

3.1.2.2. Implementation of Algorithm

- Implementing SVM, Naive Bayes, and KNN algorithms to use with a machine learning library (such as Python's scikit-learn).

3.1.2.3. Model Evaluation

- dividing the dataset into sets for testing and training.
- Modelling the training datasets and assessing it using testing datasets

3.1.2.4. Comparison and Analysis

Analysing the accuracy, precision, recall, and F1-score performance of SVM, Naive Bayes, and KNN.

- Accuracy: It indicates the overall rate of correct classification ML models.

$$accuracy = \frac{true\ postives + true\ negatives}{true\ positives + true\ negatives + false\ negatives + false\ negaive}$$

Equation 3 - accuracy equation

- Precision: This metric provides the ratio of actual positive results to the total number of positive results predicted by the model. It provides a response to the query, "How many of our positive predictions came true?"

$$precison = \frac{true\ positives}{true\ positives + false\ positives}$$

Equation 4 - precision

- Recall: This measures the model's ability to identify every positives. The answer to the question "How many of all the data points that should have been predicted as true did we correctly predict as true?" is provided by recall, also known as true positive rate.

$$recall = \frac{true\ positives}{true\ positives + false\ negatives}$$

Equation 5 - Recall

- F1-Score: Recall and precision are combined into one metric called the F1 Score. F1 can be used to measure how well our models make the trade-off between precision and recall, as we have discovered to be necessary.

$$f1 - score = 2.\frac{precison.recall}{precison + recall}$$

Equation 6 - f1-score

3.2 Pseudocode

START

IMPORT libraries

IMPORT datasets

ANALYZE the dataset

REMOVE unwanted columns

IF missing values

    REMOVE missing values

END IF

REMOVE http links

REMOVE unnecessary other characters expect numerical and alphabetical characters

LOWER all the words

SPLIT the words

CREATE a list of the split words

CREATE bag of words

SPLIT the dataset into train and test sets

TRAIN models using train datasets using SVM (Support Vector Machine), Naïve Bayes and KNN (K-Nearest Neighbours) with and without hyperparameters

TEST the model using test datasets

CHECK accuracy, precision, recall, and F1-score of each models

STOP

3.3 Flowchart

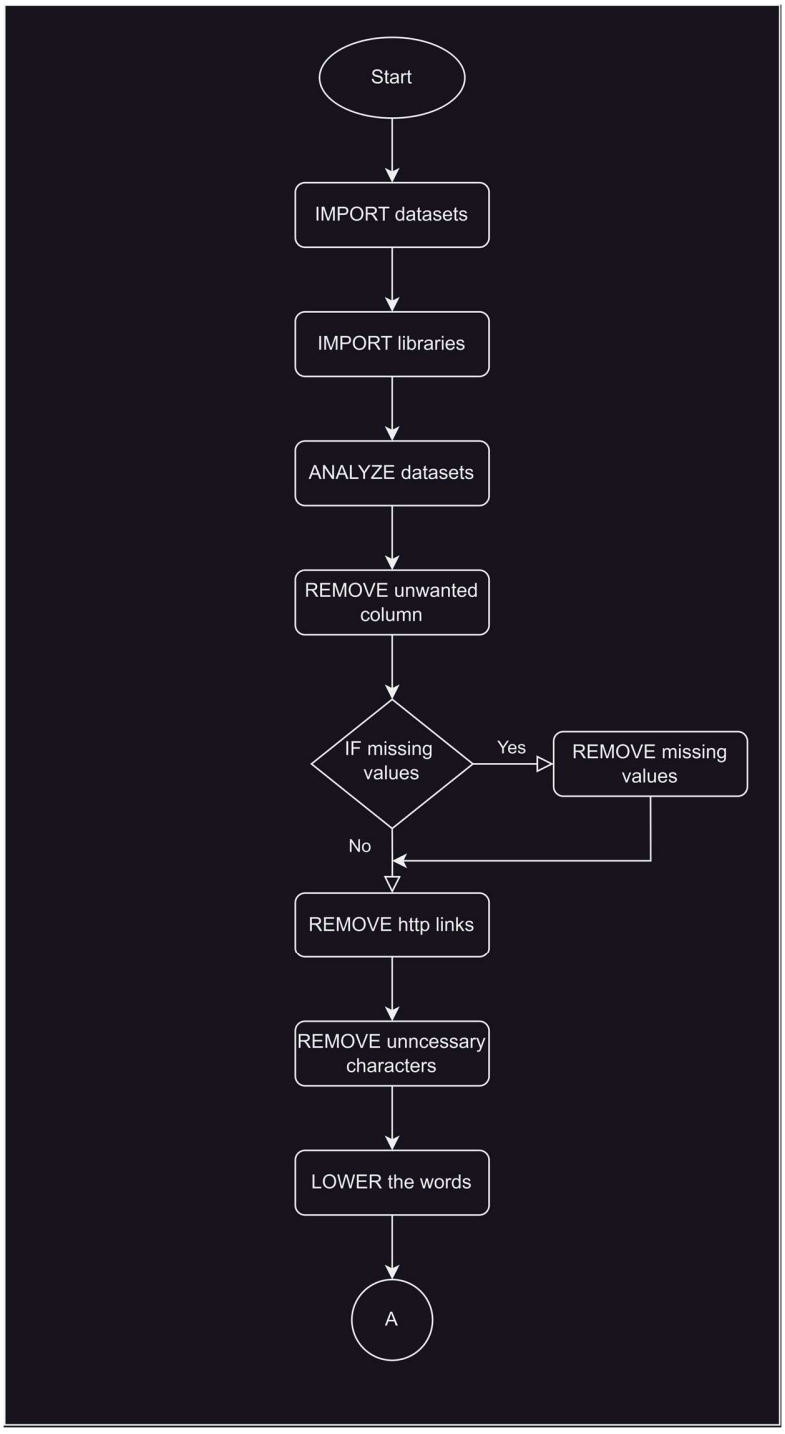


Figure 15 - Flowchart Part 1

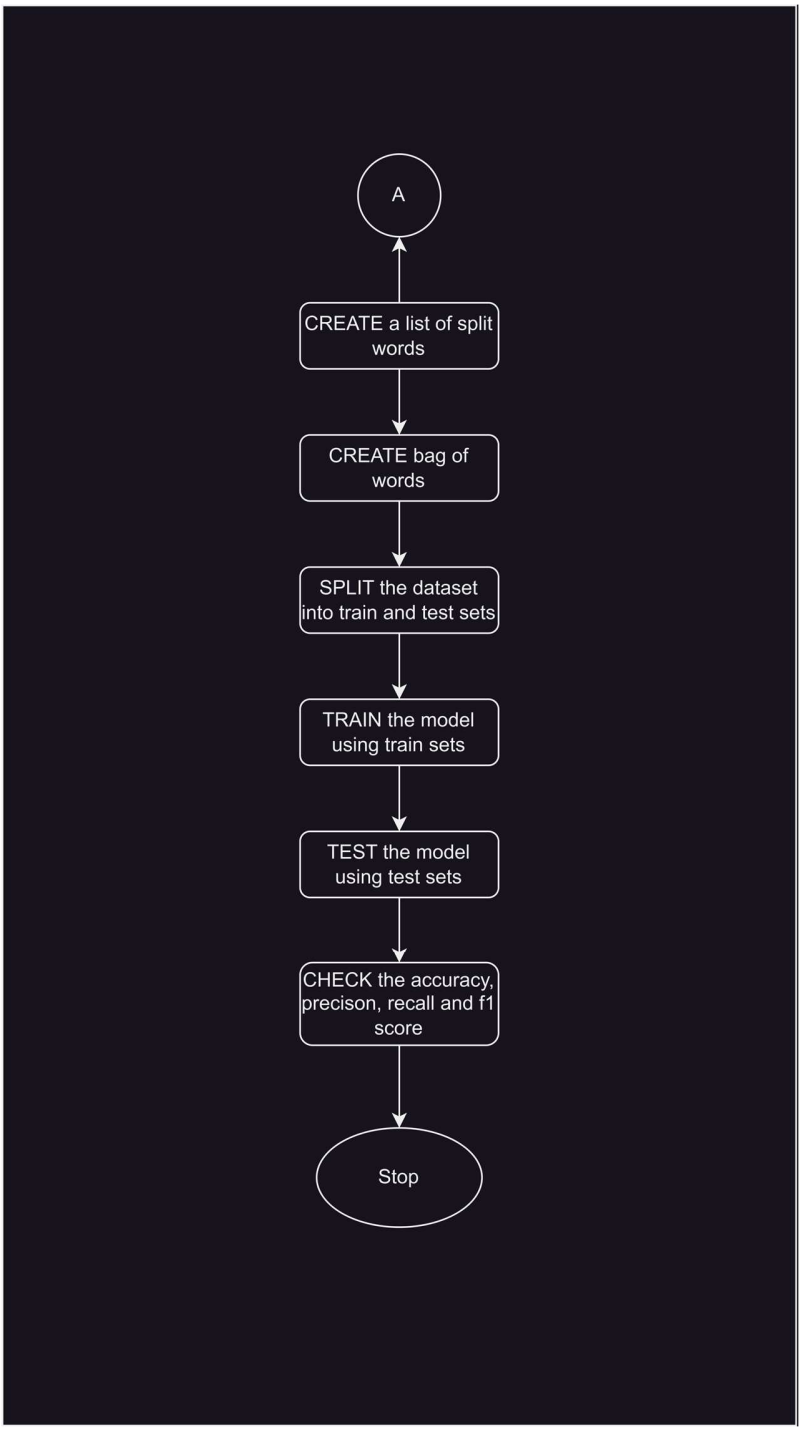


Figure 16 - Flowchart Part 2

3.4 Development Process

3.4.1 Tools Used

3.4.1.1 Anaconda

Anaconda is an open source platform where you can write and run programs with Python programming language. It was created by continuum.io, a Python programming business. The preferred approach for learning and using Python for data science, machine learning, and scientific computing is the Anaconda platform. (Ellis, 2022)

3.4.1.2 Microsoft Excel

Microsoft Excel is a spreadsheet tool that is a part of the Office product group, which consists of business applications. Spreadsheets containing data can be formatted, arranged, and calculated with Microsoft Excel. (Gillis, 2022)

3.4.1.3 Draw.io

Draw.io is a proprietary charting and diagramming software designed by Seibert Media. You can design a custom layout or use the software's automatic layout tool. They offer hundreds of visual elements and a wide variety of shapes to create a unique diagram or chart. The drag-and-drop functionality facilitates the creation of visually appealing charts and diagrams. (Hope, 2020)

3.4.2 Libraries Used

3.4.2.1 Scikit-learn

Within the Python community, Scikit-learn is the go-to library for Machine Learning (ML) and is available as an open source library. We have used different algorithms from it and also, we have used it to get different metrics like accuracy and precisons. (M, 2021)

3.4.2.2 Regular Express(re)

A regular expression is a unique character sequence that, when combined with a certain syntax stored in a pattern, aids in matching or locating other strings or groupings of strings. A string of characters called a regular expression, or regex, specifies a search pattern. Known by its common name, regex or regexp, this string of characters indicates

a match pattern in text. These patterns are typically utilized by string-searching algorithms for input validation or for "find" or "find and replace" operations on strings. (tutorialspoint, 2022)

3.4.2.3 Natural Language Toolkit (NLTK)

Statistical natural language processing (NLP) uses Python programs that operate with human language data and are built on the Natural Language Toolkit (NLTK) framework. For tokenization, parsing, categorization, stemming, tagging, and semantic reasoning, it has text processing libraries. Together with sample data sets and graphical demonstrations, it also comes with a cookbook and a book that teaches the fundamental ideas of the language processing jobs that NLTK offers. (Rouse, 2023)

3.4.2.4 NumPy

One of the most widely used Python packages for scientific computing is called NumPy (Numerical Python). In addition to modifications like masks and matrices, it offers a multidimensional array object (n-dimensional array, abbreviated as ndarray) that may be used for a variety of mathematical operations on numerical datatypes (dtypes). NumPy for Python is compatible with and utilized by numerous other well-known Python packages, such as matplotlib and pandas. (M, 2020)

3.4.2.5 Pandas

The most popular uses for Pandas, an open-source Python module, are in data science/data analysis and machine learning applications. It is constructed upon NumPy, another tool that handles multi-dimensional arrays. One of the most widely used data wrangling programs, Pandas is commonly included in Python distributions and integrates effectively with a wide range of other data science modules within the Python ecosystem. (S, 2020)

3.4.2.6 Matplotlib

Matplotlib is a Python visualization package for two-dimensional array charts. Based on NumPy arrays, Matplotlib is a multi-platform data visualization package intended to be used with the larger SciPy stack. In the year 2002, John Hunter made its debut. The ability to visually access vast volumes of data in a format that is simple to understand is one of

visualization's biggest advantages. Many plot types, including line, bar, scatter, histogram, and more, are available in Matplotlib. (geeksforgeeks, 2023)

3.4.3 Explanation of the development process

3.4.3.1 Importing Libraries

Library Imports

```
#Library for Gaussian Naive Bayes algorithm
from sklearn.naive_bayes import GaussianNB
#Library for Support Vector Machine algorithm
from sklearn import svm
#Library for K-Neigbour Classifier algorithm
from sklearn.neighbors import KNeighborsClassifier
#Library for splitting the data to train and test dataset
from sklearn.model_selection import train_test_split
#Library for checking accuracy,precisonscore, recall, f1score
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix
#Library to count the unique words from the collections of texts
from sklearn.feature_extraction.text import CountVectorizer
#To remove patterns or links
import re
import nltk
nltk.download('omw-1.4')
nltk.download('punkt')
nltk.download('stopwords')
from nltk.stem import WordNetLemmatizer
#For all algebric functions
import numpy as np
#For data manipulation
import pandas as pd
#For plotting tables and graphs
import matplotlib.pyplot as plt
```

```
[nltk_data] Downloading package omw-1.4 to
[nltk_data]     C:\Users\amitb\AppData\Roaming\nltk_data...
[nltk_data]   Package omw-1.4 is already up-to-date!
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\amitb\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\amitb\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Figure 17 - Libraries imports.

In this section required imports of libraries has been made. As you can see in the screenshot above, we have imported all the algorithms that we will be using and along with that we have imported other libraries for data cleaning and manipulation. Also imported libraries to plot different type of figures.

3.4.3.2 Importing Datasets

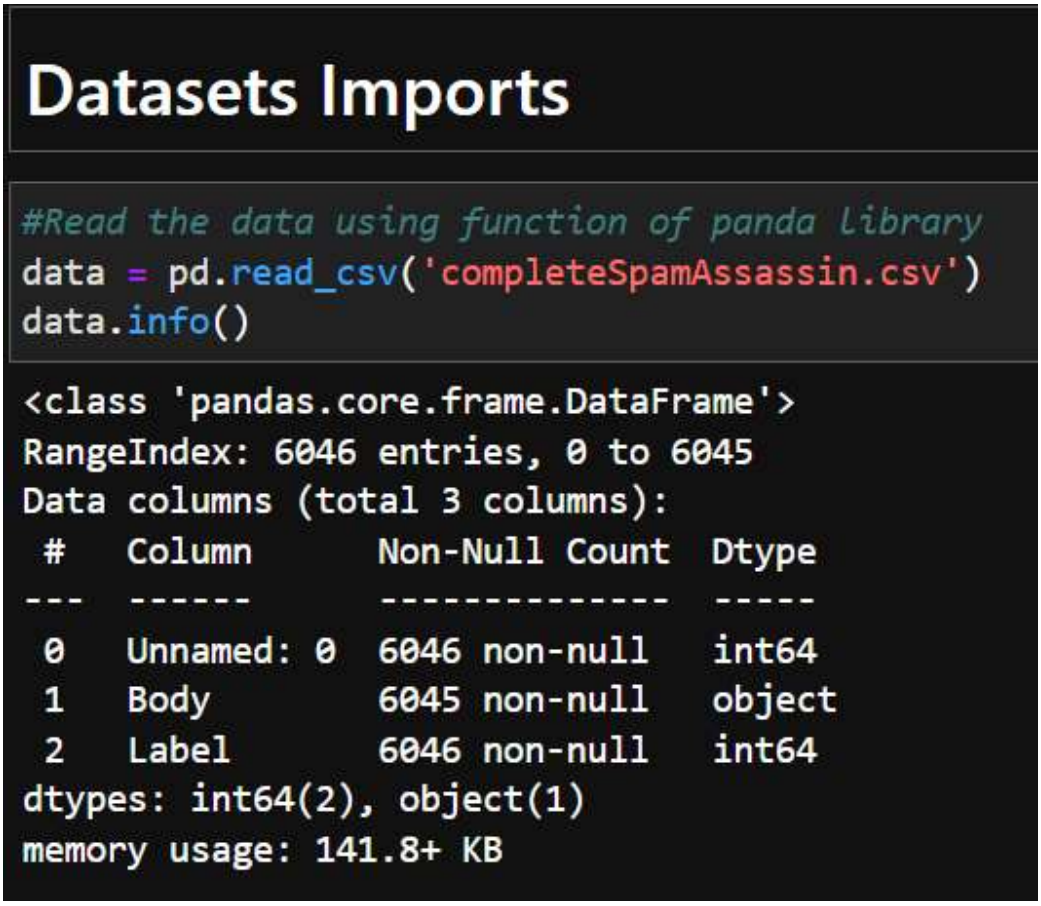


Figure 18 - Importing datasets.

Here, we have imported a dataset which contains three columns which are body, label, and unnamed columns. We will be using body and label column. There are 6046 rows in this dataset. The body contains all the mails which may or may not be spam. To determine if the body is spam or not there is label with each body which shows if the mail is spam or not, 0 means do not spam and 1 means spam.

3.4.3.3 Data Cleaning and Preparation

3.4.3.3.1 Deletion of unwanted column

Deletion of unwanted columns

Here, we will only be using body and label column as body has all the spam and ham mails the other two columns are just numbering therefore we will be removing it.

```
[6]: data.head(3)
```

[6]:

	Unnamed: 0	Body	Label
0	0	\nSave up to 70% on Life Insurance.\nWhy Spend...	1
1	1	1) Fight The Risk of Cancer!\nhttp://www.adcli...	1
2	2	1) Fight The Risk of Cancer!\nhttp://www.adcli...	1

```
[7]: data.drop(["Unnamed: 0"],inplace=True,axis=1)
```

```
[8]: data.head(3)
```

[8]:

	Body	Label
0	\nSave up to 70% on Life Insurance.\nWhy Spend...	1
1	1) Fight The Risk of Cancer!\nhttp://www.adcli...	1
2	1) Fight The Risk of Cancer!\nhttp://www.adcli...	1

Figure 19 - Deletion of unwanted column

Here, we have removed the unwanted column which is the first column ‘unwanted:0’ this column is there just for indexing therefore, we will not be needing this column.

3.4.3.3.2 Missing value row removal

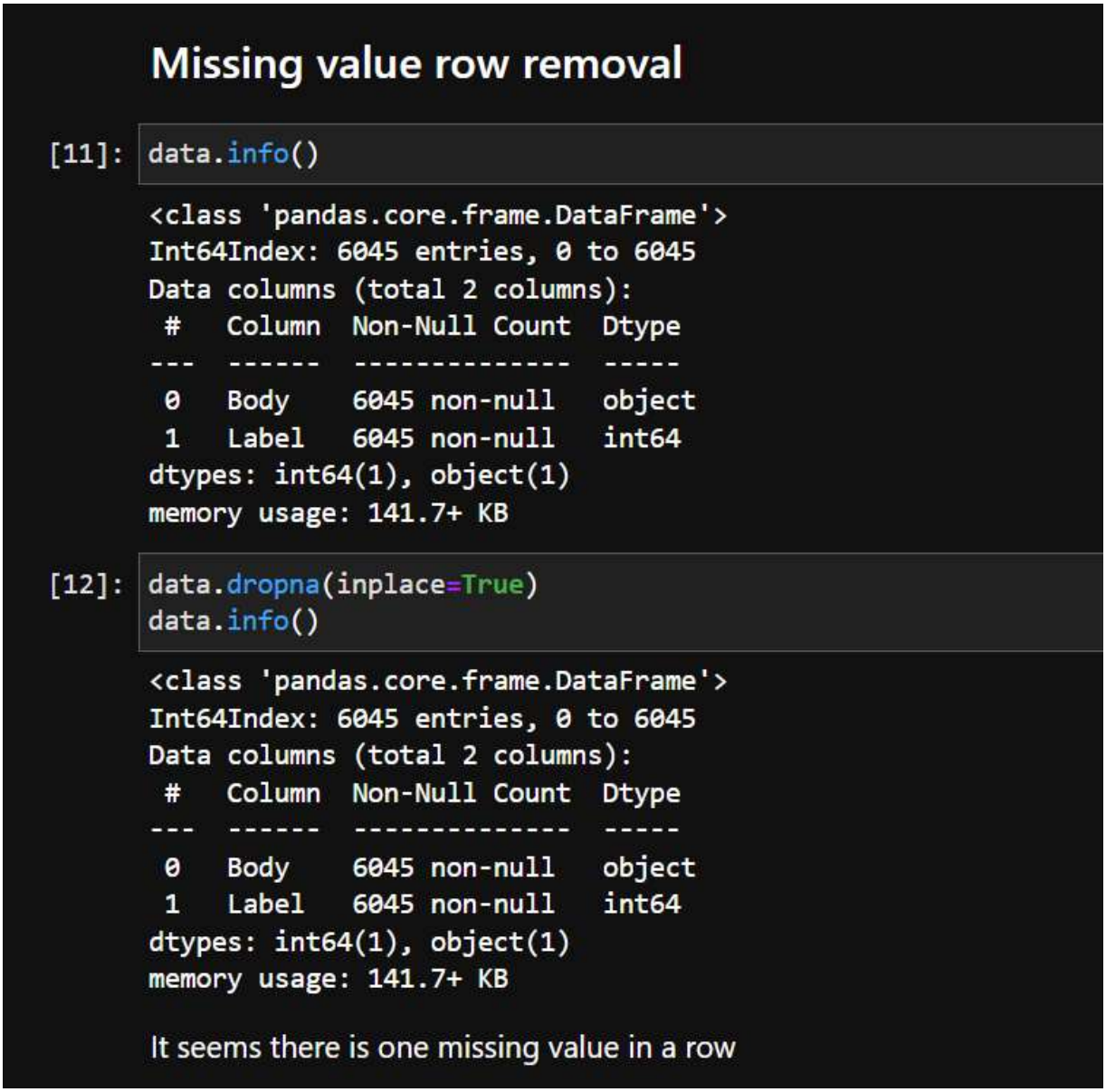


Figure 20 - Missing value row removal

In the above, screenshot you can see that we have removed the row containing the missing value. It seems we only had one row that had missing value in it.

3.4.3.3.3 Removing Links

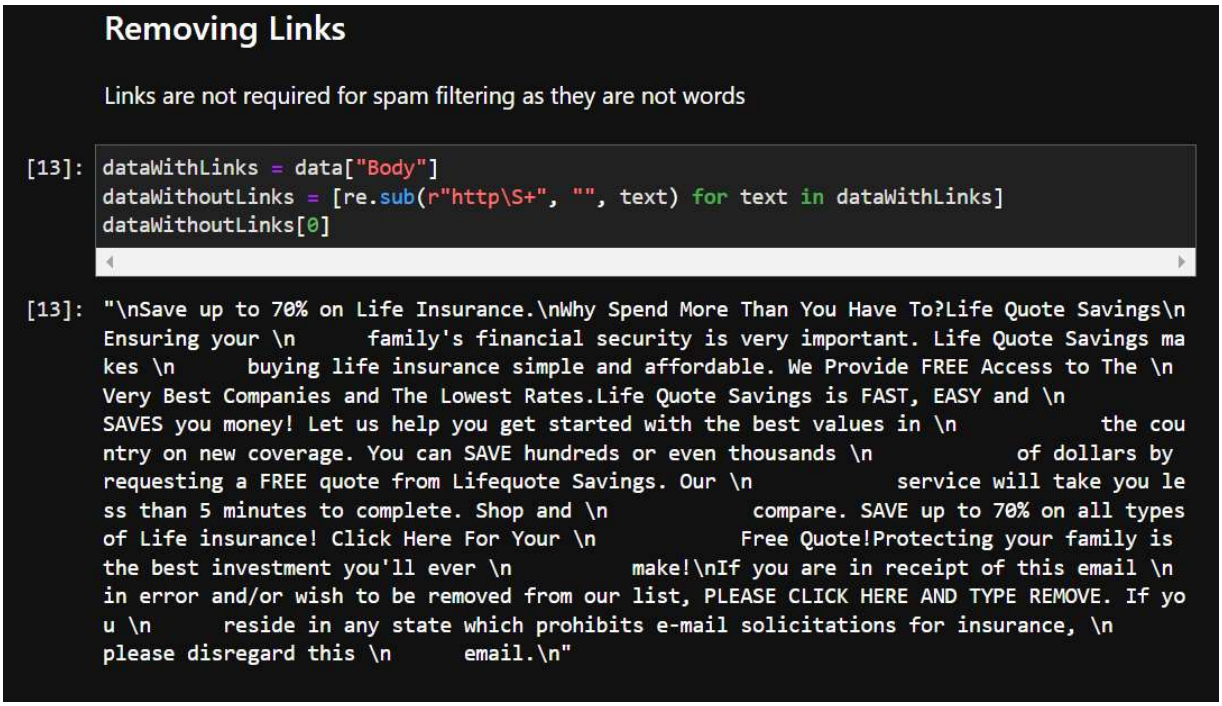


Figure 21 - Removing http links.

Here we have removed all the links to the website. As links are not words and have no meaning, we don't need them.

3.4.3.3.4 Removing character except alphabetical and numerical characters

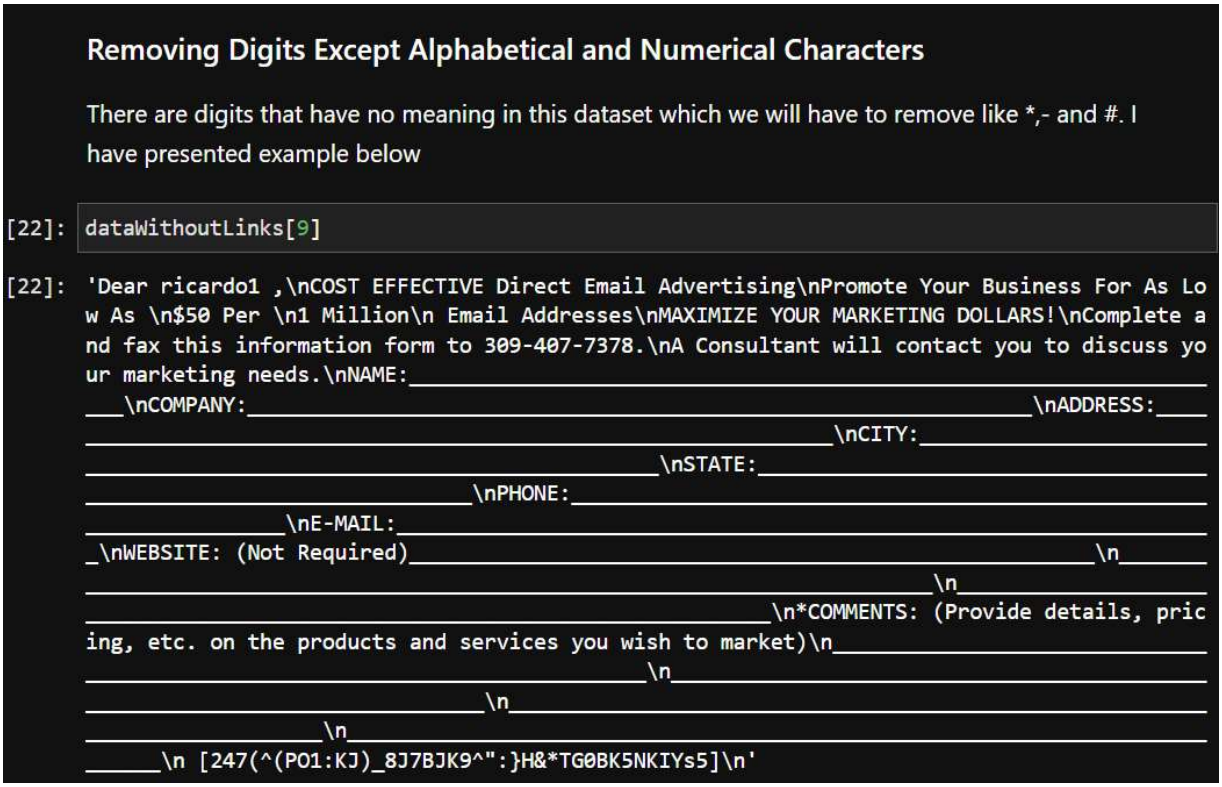


Figure 22 - Unwanted character removal – Before

Here you can see that there are a lot of underscores as well as slashes. These characters are not needed. Therefore, we will be removing them.

```
[23]: #we will be removing all the other except these alphabets and numbers
      pattern = "[^a-zA-Z0-9]"

[24]: uncleanedData = data["Body"]

[25]: cleanedData = [re.sub(pattern," ",text) for text in uncleanedData]
      cleanedData[9]

[25]: 'Dear ricardo1   COST EFFECTIVE Direct Email Advertising Promote Your Business For As Low
      As   50 Per  1 Million  Email Addresses MAXIMIZE YOUR MARKETING DOLLARS  Complete and fax
      this information form to 309 407 7378  A Consultant will contact you to discuss your marke
      ting needs  NAME                                                                    COMPA
      NY                                                                  ADDRESS
      CITY                                                                       STATE
      PHONE                                                                   E MAIL
      WEBSITE   Not Required
      COMMENTS   Provide details  pricing  etc  on the products and services you wish to market
      247   PO1 KJ  8J7BJK9    H  TG0BK5NKIYs5  '
```

Figure 23 - Unwanted character removal – After

As you can see, we have removed all the unwanted characters and replaced it with spaces.

3.4.3.3.5 Lowering all the words

Lower all the words

Now we will lower all the words so that it will be easier to find unique words

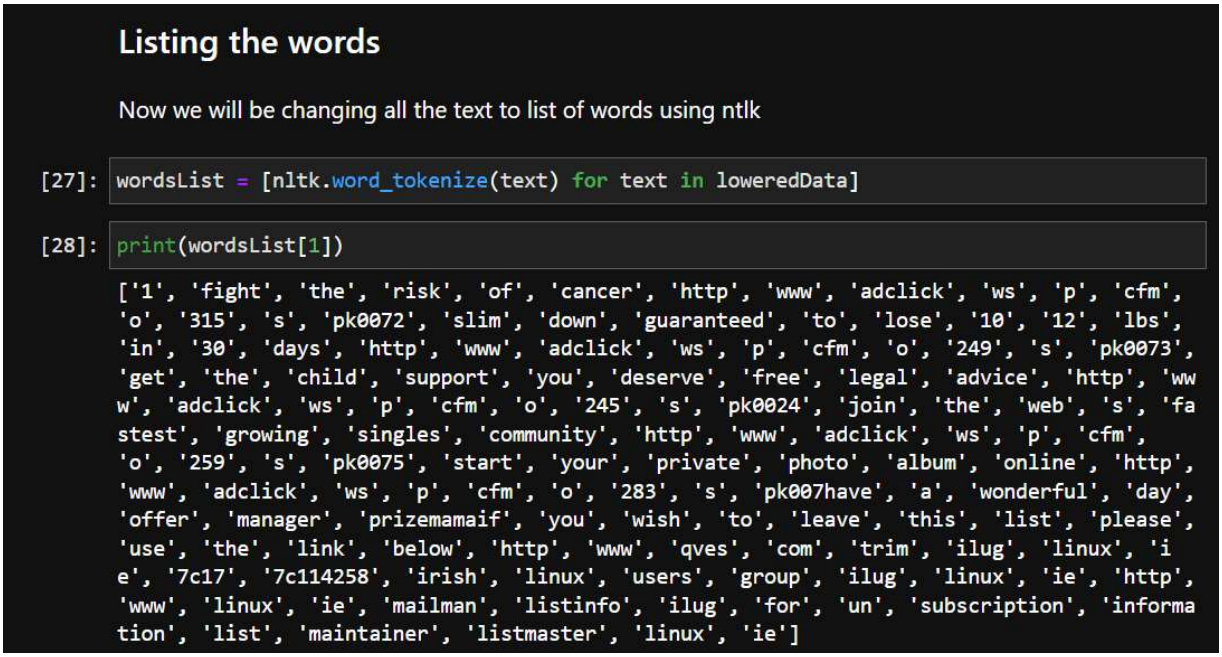
```
[26]: loweredData = [text.lower() for text in cleanedData]
      loweredData[1]

[26]: '1  fight the risk of cancer  http   www adclick ws p cfm o 315 s pk0072  slim down   guar
      anteed to lose 10 12 lbs in 30 days http   www adclick ws p cfm o 249 s pk0073  get the ch
      ild support you deserve   free legal advice http   www adclick ws p cfm o 245 s pk0024  jo
      in the web s fastest growing singles community http   www adclick ws p cfm o 259 s pk0075
      start your private photo album online  http   www adclick ws p cfm o 283 s pk007have a won
      derful day  offer manager prizemamaif you wish to leave this list please use the link belo
      w  http   www qves com trim  ilug linux ie 7c17 7c114258     irish linux users  group  ilu
      g linux ie http   www linux ie mailman listinfo ilug for  un subscription information  lis
      t maintainer  listmaster linux ie'
```

Figure 24 - changing words into lowercase

Here we have changed all the words into lowercase as it will make it easier for same word identification.

3.4.3.3.6 Wordlist Creation



Listing the words

Now we will be changing all the text to list of words using ntlk

```
[27]: wordsList = [nltk.word_tokenize(text) for text in loweredData]

[28]: print(wordsList[1])
```

```
['1', 'fight', 'the', 'risk', 'of', 'cancer', 'http', 'www', 'adclick', 'ws', 'p', 'cfm',
'o', '315', 's', 'pk0072', 'slim', 'down', 'guaranteed', 'to', 'lose', '10', '12', 'lbs',
'in', '30', 'days', 'http', 'www', 'adclick', 'ws', 'p', 'cfm', 'o', '249', 's', 'pk0073',
'get', 'the', 'child', 'support', 'you', 'deserve', 'free', 'legal', 'advice', 'http', 'ww
w', 'adclick', 'ws', 'p', 'cfm', 'o', '245', 's', 'pk0024', 'join', 'the', 'web', 's', 'fa
stest', 'growing', 'singles', 'community', 'http', 'www', 'adclick', 'ws', 'p', 'cfm',
'o', '259', 's', 'pk0075', 'start', 'your', 'private', 'photo', 'album', 'online', 'http',
'www', 'adclick', 'ws', 'p', 'cfm', 'o', '283', 's', 'pk007have', 'a', 'wonderful', 'day',
'offer', 'manager', 'prizemamaif', 'you', 'wish', 'to', 'leave', 'this', 'list', 'please',
'use', 'the', 'link', 'below', 'http', 'www', 'qves', 'com', 'trim', 'ilug', 'linux', 'i
e', '7c17', '7c114258', 'irish', 'linux', 'users', 'group', 'ilug', 'linux', 'ie', 'http',
'www', 'linux', 'ie', 'mailman', 'listinfo', 'ilug', 'for', 'un', 'subscription', 'informa
tion', 'list', 'maintainer', 'listmaster', 'linux', 'ie']
```

In order to create bag of words we will have to create a list of all the words, therefore in this section we have created list of all text using Natural Language Toolkit.

3.4.3.3.7 Removing Similar words

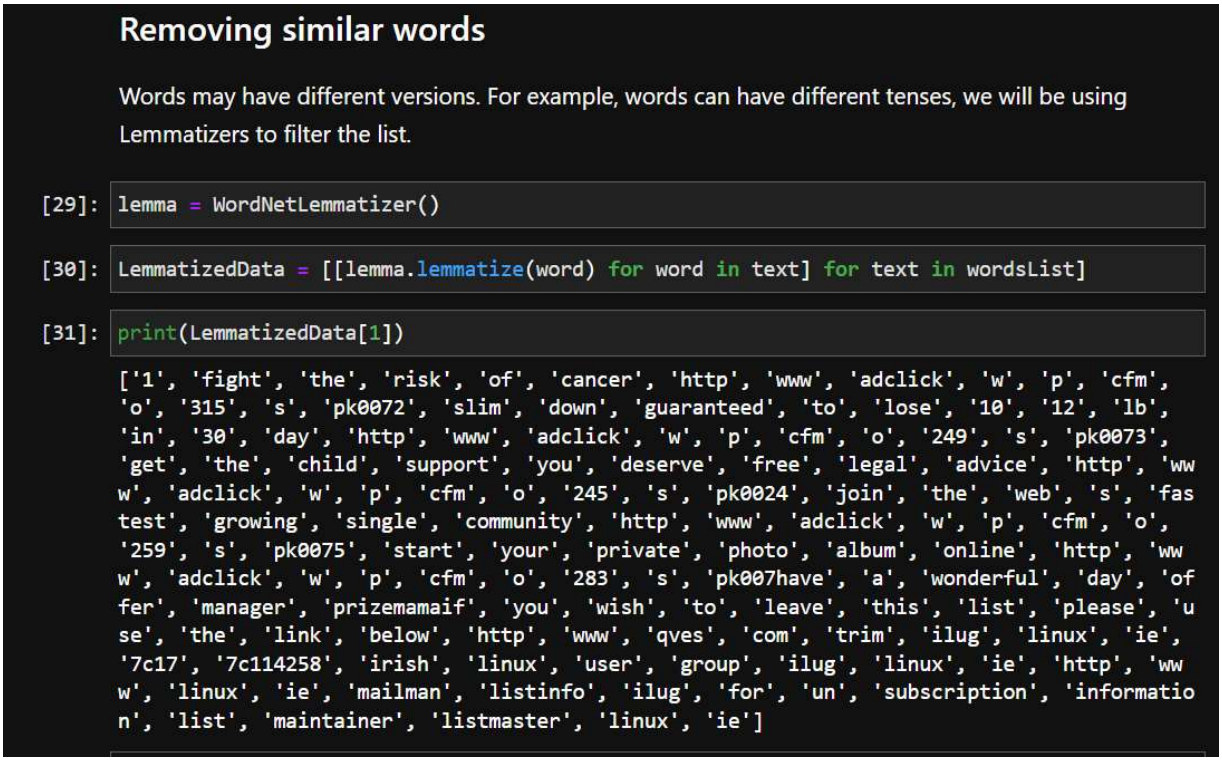


Figure 25 - Removing all the similar words.

Here, we have removed all the similar words. Similar word can be words with different tenses. And words may also have different versions the. After doing this we get the words root form.

3.4.3.3.8 Removing stop words

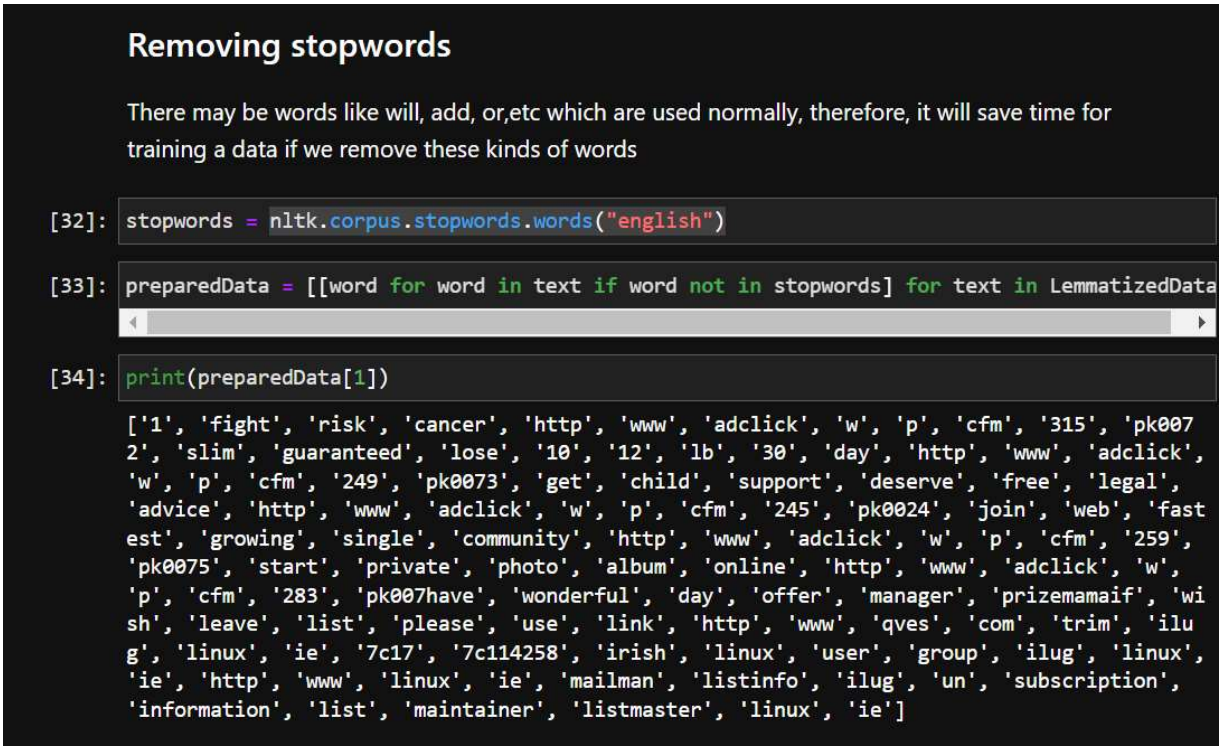


Figure 26 - Stop words removal.

There may be words like will, add, or etc which are used normally. therefore, in order to save time for training a model these kinds of words gave been removed.

3.4.3.3.9 Unique Words Count

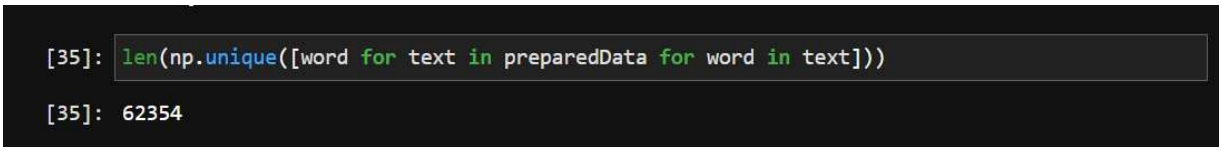


Figure 27 - unique words count.

After the data cleaning process, it seems we have 62354 unique words. Now next process will be created bag of words.

3.4.3.3.10 Creating Bag of words

Bag of words is a way to show text data in numerical form so that machine learning algorithm can easily identify and can train a model using those bags of words.

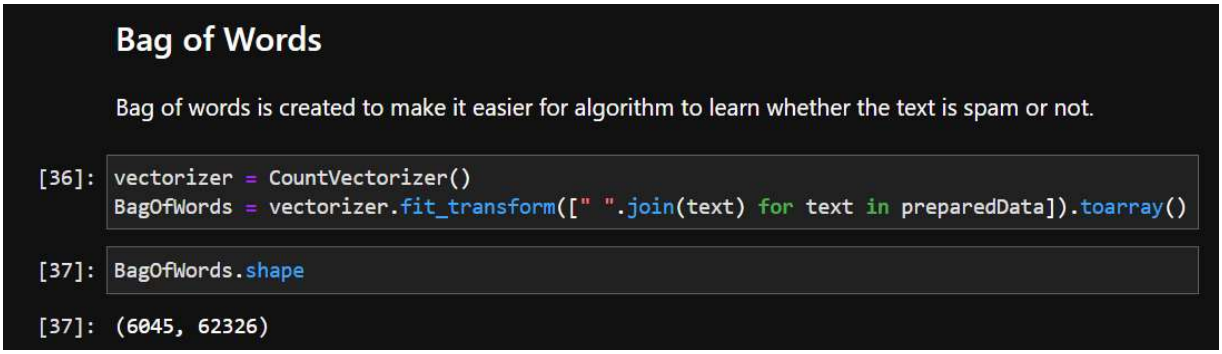


Figure 28 - Bag of words

In the above screenshot, the prepared Data have been converted to bag of words using Count vectorizer library. It seems we have 6045 rows which represents the body and 62326 columns which represents rows.

3.4.3.3.11 Data Splitting into training and testing dataset

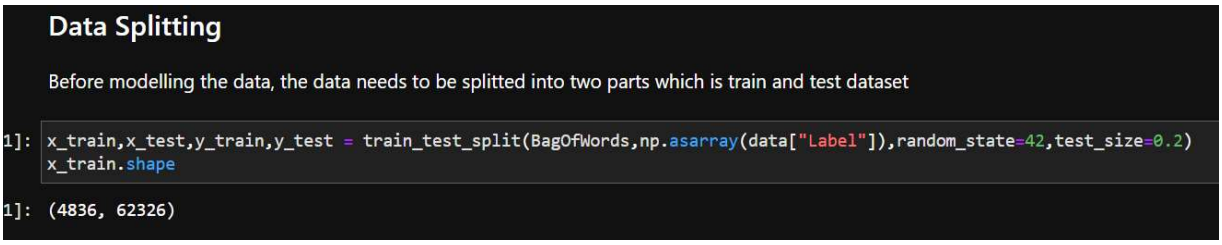


Figure 29 - Data splitting into training and testing datasets

Here we have split data into two parts one for training the model which consist of 80% of all data and another is for testing which consists of 20% of all data.

3.4.3.4 Data Modelling

3.4.3.4.1 SVM (Support Vector Machine)

I. Without Hyperparameter tuning

Support Vector Machine

Without Hyperparameter tuning

```
[25]: normal_svm_model = svm.SVC()
      normal_svm_model.fit(x_train, y_train)
```

```
[25]: ▾ SVC
      SVC()
```

```
[26]: y_pred = normal_svm_model.predict(x_test)

      accuracy = accuracy_score(y_test, y_pred) * 100
      precision = precision_score(y_test, y_pred,zero_division=1) * 100
      recall = recall_score(y_test, y_pred) * 100
      f1 = f1_score(y_test, y_pred) * 100
      conf_matrix = confusion_matrix(y_test, y_pred)
      error_rate = ((conf_matrix[0, 1] + conf_matrix[1, 0]) / float(conf_matrix.sum()))*100

      print("Metrics without Hyperparameter Tuning:")
      print("Accuracy: {:.2f}%".format(accuracy))
      print("Precision:{:.2f}%".format( precision) )
      print("Recall:{:.2f}%".format( recall ))
      print("F1 Score{:.2f}%".format( f1))
      print("Error Rate:{:.2f}%".format( error_rate) )
```

```
Metrics without Hyperparameter Tuning:
Accuracy: 91.15%
Precision:99.66%
Recall:73.63%
F1 Score84.69%
Error Rate:8.85%
```

Figure 30 - SVM - No Hyperparameter tuning.

Here we have trained a model using SVM algorithm and calculated all the required metrics. As you can see the accuracy of the svm is 91.15%, precision is 99.66%, recall is 73.63%, f1score is 84.69% and error rate is 8.85%.

II. With Hyperparameter tuning

In order to train a model with hyperparameter tuning we will be creating a function as we will be hyper tuning 10 times; therefore, we will not have to write all the code again and again.

With Hyperparameter tuning

```
[ ]: def evaluate_svm_model(parameters, x_train, y_train, x_test, y_test):
         svm_model = svm.SVC(**parameters)
         svm_model.fit(x_train, y_train)
         y_pred = svm_model.predict(x_test)

         accuracy = accuracy_score(y_test, y_pred) * 100
         precision = precision_score(y_test, y_pred, zero_division=1) * 100
         recall = recall_score(y_test, y_pred) * 100
         f1 = f1_score(y_test, y_pred) * 100
         conf_matrix = confusion_matrix(y_test, y_pred)
         error_rate = ((conf_matrix[0, 1] + conf_matrix[1, 0]) / float(conf_matrix.sum())) * 100

         return accuracy, precision, recall, f1, error_rate
```

Figure 31 - SVM - Function for model training with hyperparameter tuning.

Here we have created a function which we will be using to train svm models with different hyper parameters this returns accuracy, precision, recall, f1score and error rate.

1. Hyperparameter tuning – 1.

Hyperparameter tuning 1

```
[54]: parameter_1 = {'C': 0.1, 'kernel': 'linear', 'gamma': 'scale'}
      result_1 = evaluate_svm_model(parameter_1, x_train, y_train, x_test, y_test)
      print("Metrics with Hyperparameter Tuning: 1")
      print("Accuracy: {:.2f}%".format(result_1[0]))
      print("Precision: {:.2f}%".format( result_1[1]))
      print("Recall: {:.2f}%".format(result_1[2]))
      print("F1 Score: {:.2f}%".format( result_1[3]))
      print("Error Rate: {:.2f}%".format( result_1[4]))

      Metrics with Hyperparameter Tuning: 1
      Accuracy: 95.29%
      Precision: 89.12%
      Recall: 97.76%
      F1 Score: 93.24%
      Error Rate: 4.71%
```

Figure 32 - SVM – Model with Hyperparameter tuning 1.

Here, we have changed the parameter, and it seems the accuracy is 95.29%.

2. **Hyperparameter tuning – 2.**

Hyperparameter tuning 2

```
[55]: parameter_2 = {'C': 1, 'kernel': 'rbf', 'gamma': 'scale'}
result_2 = evaluate_svm_model(parameter_2, x_train, y_train, x_test, y_test)
print("Metrics with Hyperparameter Tuning: 1")
print("Accuracy: {:.2f}%".format(result_2[0]))
print("Precision: {:.2f}%".format( result_2[1]))
print("Recall: {:.2f}%".format(result_2[2]))
print("F1 Score: {:.2f}%".format( result_2[3]))
print("Error Rate: {:.2f}%".format( result_2[4]))
```

```
Metrics with Hyperparameter Tuning: 1
Accuracy: 91.15%
Precision: 99.66%
Recall: 73.63%
F1 Score: 84.69%
Error Rate: 8.85%
```

Figure 33 - SVM – Model with Hyperparameter tuning 2.

Here, we have changed the parameter, and it seems the accuracy is 91.15%.

3. **Hyperparameter tuning – 3.**

Hyperparameter tuning 3

```
[56]: parameter_3 = {'C': 10, 'kernel': 'poly', 'gamma': 0.1}
result_3 = evaluate_svm_model(parameter_3, x_train, y_train, x_test, y_test)
print("Metrics with Hyperparameter Tuning: 3")
print("Accuracy: {:.2f}%".format(result_3[0]))
print("Precision: {:.2f}%".format( result_3[1]))
print("Recall: {:.2f}%".format(result_3[2]))
print("F1 Score: {:.2f}%".format( result_3[3]))
print("Error Rate: {:.2f}%".format( result_3[4]))
```

```
Metrics with Hyperparameter Tuning: 3
Accuracy: 86.10%
Precision: 74.38%
Recall: 88.81%
F1 Score: 80.95%
Error Rate: 13.90%
```

Figure 34- SVM – Model with Hyperparameter tuning 3.

Here, we have changed the parameter, and it seems the accuracy is 86.10%.

4. Hyperparameter tuning – 4

```
Hyperparameter tuning 4

[57]: parameter_4 = {'C': 0.01, 'kernel': 'linear', 'gamma': 'auto'}
      result_4 = evaluate_svm_model(parameter_4, x_train, y_train, x_test, y_test)
      print("Metrics with Hyperparameter Tuning: 4")
      print("Accuracy: {:.2f}%".format(result_4[0]))
      print("Precision: {:.2f}%".format( result_4[1]))
      print("Recall: {:.2f}%".format(result_4[2]))
      print("F1 Score: {:.2f}%".format( result_4[3]))
      print("Error Rate: {:.2f}%".format( result_4[4]))

      Metrics with Hyperparameter Tuning: 4
      Accuracy: 95.29%
      Precision: 89.47%
      Recall: 97.26%
      F1 Score: 93.21%
      Error Rate: 4.71%
```

Figure 35- SVM – Model with Hyperparameter tuning 4

Here, we have changed the parameter, and it seems the accuracy is 95.29%.

5. Hyperparameter tuning – 5

```
Hyperparameter tuning 5

[58]: parameter_5 = {'C': 0.5, 'kernel': 'rbf', 'gamma': 'auto'}
      result_5 = evaluate_svm_model(parameter_5, x_train, y_train, x_test, y_test)
      print("Metrics with Hyperparameter Tuning: 5")
      print("Accuracy: {:.2f}%".format(result_5[0]))
      print("Precision: {:.2f}%".format( result_5[1]))
      print("Recall: {:.2f}%".format(result_5[2]))
      print("F1 Score: {:.2f}%".format( result_5[3]))
      print("Error Rate: {:.2f}%".format( result_5[4]))

      Metrics with Hyperparameter Tuning: 5
      Accuracy: 68.32%
      Precision: 100.00%
      Recall: 4.73%
      F1 Score: 9.03%
      Error Rate: 31.68%
```

Figure 36- SVM – Model with Hyperparameter tuning 5

Here, we have changed the parameter and it seems the accuracy is 68.32%.

6. Hyperparameter tuning – 6

Hyperparameter tuning 6

```
[59]: parameter_6 = {'C': 5, 'kernel': 'poly', 'gamma': 0.01}
result_6 = evaluate_svm_model(parameter_6, x_train, y_train, x_test, y_test)
print("Metrics with Hyperparameter Tuning: 6")
print("Accuracy: {:.2f}%".format(result_6[0]))
print("Precision: {:.2f}%".format( result_6[1]))
print("Recall: {:.2f}%".format(result_6[2]))
print("F1 Score: {:.2f}%".format( result_6[3]))
print("Error Rate: {:.2f}%".format( result_6[4]))
```

```
Metrics with Hyperparameter Tuning: 6
Accuracy: 86.68%
Precision: 94.14%
Recall: 63.93%
F1 Score: 76.15%
Error Rate: 13.32%
```

Figure 37- SVM – Model with Hyperparameter tuning 6

Here, we have changed the parameter and it seems the accuracy is 86.68%.

7. Hyperparameter tuning – 7

Hyperparameter tuning 7

```
[60]: parameter_7 = {'C': 0.001, 'kernel': 'linear', 'gamma': 'scale'}
result_7 = evaluate_svm_model(parameter_7, x_train, y_train, x_test, y_test)
print("Metrics with Hyperparameter Tuning: 7")
print("Accuracy: {:.2f}%".format(result_7[0]))
print("Precision: {:.2f}%".format( result_7[1]))
print("Recall: {:.2f}%".format(result_7[2]))
print("F1 Score: {:.2f}%".format( result_7[3]))
print("Error Rate: {:.2f}%".format( result_7[4]))
```

```
Metrics with Hyperparameter Tuning: 7
Accuracy: 89.66%
Precision: 99.29%
Recall: 69.40%
F1 Score: 81.70%
Error Rate: 10.34%
```

Figure 38- SVM – Model with Hyperparameter tuning 7

Here, we have changed the parameter and it seems the accuracy is 89.66%.

8. Hyperparameter tuning – 8

Hyperparameter tuning 8

```
[61]: parameter_8 ={'C': 20, 'kernel': 'poly', 'gamma': 'auto'}
result_8 = evaluate_svm_model(parameter_8, x_train, y_train, x_test, y_test)
print("Metrics with Hyperparameter Tuning: 8")
print("Accuracy: {:.2f}%".format(result_8[0]))
print("Precision: {:.2f}%".format( result_8[1]))
print("Recall: {:.2f}%".format(result_8[2]))
print("F1 Score: {:.2f}%".format( result_8[3]))
print("Error Rate: {:.2f}%".format( result_8[4]))
```

```
Metrics with Hyperparameter Tuning: 8
Accuracy: 67.25%
Precision: 87.50%
Recall: 1.74%
F1 Score: 3.41%
Error Rate: 32.75%
```

Figure 39- SVM – Model with Hyperparameter tuning 8

Here, we have changed the parameter and it seems the accuracy is 67.25%.

9. Hyperparameter tuning – 9

```
Hyperparameter tuning 9

[62]: parameter_9 = {'C': 0.005, 'kernel': 'linear', 'gamma': 0.01}
      result_9 = evaluate_svm_model(parameter_9, x_train, y_train, x_test, y_test)
      print("Metrics with Hyperparameter Tuning: 9")
      print("Accuracy: {:.2f}%".format(result_9[0]))
      print("Precision: {:.2f}%".format( result_9[1]))
      print("Recall: {:.2f}%".format(result_9[2]))
      print("F1 Score: {:.2f}%".format( result_9[3]))
      print("Error Rate: {:.2f}%".format( result_9[4]))

      Metrics with Hyperparameter Tuning: 9
      Accuracy: 95.04%
      Precision: 90.14%
      Recall: 95.52%
      F1 Score: 92.75%
      Error Rate: 4.96%
```

Figure 40- SVM – Model with Hyperparameter tuning 9

Here, we have changed the parameter and it seems the accuracy is 95.04%.

10. Hyperparameter tuning – 10

```
      Error Rate: 4.96%

      Hyperparameter tuning 10

[63]: parameter_10 = {'C': 0.2, 'kernel': 'rbf', 'gamma': 'scale'}
      result_10 = evaluate_svm_model(parameter_10, x_train, y_train, x_test, y_test)
      print("Metrics with Hyperparameter Tuning: 9")
      print("Accuracy: {:.2f}%".format(result_10[0]))
      print("Precision: {:.2f}%".format( result_10[1]))
      print("Recall: {:.2f}%".format(result_10[2]))
      print("F1 Score: {:.2f}%".format( result_10[3]))
      print("Error Rate: {:.2f}%".format( result_10[4]))

      Metrics with Hyperparameter Tuning: 9
      Accuracy: 85.19%
      Precision: 99.56%
      Recall: 55.72%
      F1 Score: 71.45%
      Error Rate: 14.81%
```

Figure 41- SVM – Model with Hyperparameter tuning 10

Here, we have changed the parameter and it seems the accuracy is 85.19%.

III. SVM accuracy Evaluation

Evaluation of SVM Accuracy

```
29]: %matplotlib inline


     Headings = ['No-Hyperparameter','Hyperparameter - 1','Hyperparameter - 2',
                 'Hyperparameter - 3','Hyperparameter - 4','Hyperparameter - 5',
                 'Hyperparameter - 6','Hyperparameter - 7','Hyperparameter - 8',
                 'Hyperparameter - 9','Hyperparameter - 10']
     Heading_values = [accuracy, result_1[0],result_2[0],
                       result_3[0],result_4[0],result_5[0],
                       result_6[0],result_7[0],result_8[0],
                       result_9[0],result_10[0]]
     max_value = Heading_values.index(max(Heading_values))
     plt.figure(figsize=(20, 5))
     plt.title('SVM accuracy with different hyperparameters')
     bars = plt.bar(Headings, Heading_values)
     bars[max_value].set_color('purple')
     for i, v in enumerate(Heading_values):
         plt.text(i, v + 1, str(round(v, 2)), ha='center',
                  va='bottom', fontweight='bold', color='black')
     plt.xticks(rotation="vertical")

29]: ([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

Figure 42 - SVM accuracy Evaluation - Code

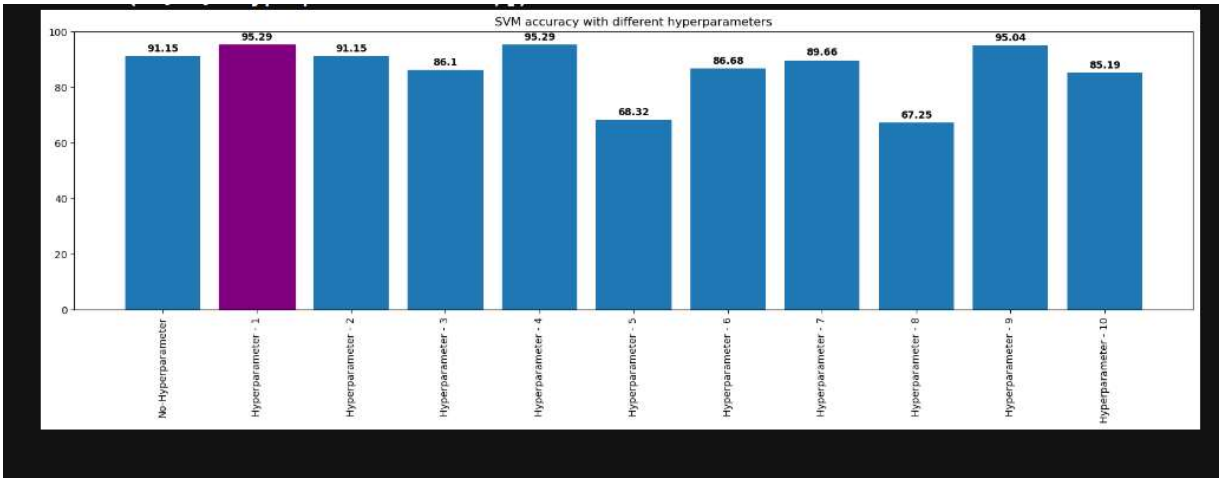


Figure 43 - SVM accuracy Evaluation – Graph

From the comparison we can see that the 2nd model which is with hyperparameter is the one with the highest accuracy

3.4.3.4.2 KNN (K-Nearest Neigbours)

I. Without Hyperparameter tunning

Without Hyperparameter tuning

```
[28]: knn_normal_model = KNeighborsClassifier()

      knn_normal_model.fit(x_train, y_train)
```

```
[28]: ▾ KNeighborsClassifier
      KNeighborsClassifier()
```

```
[29]: knn_y_pred = knn_normal_model.predict(x_test)

      knn_accuracy = accuracy_score(y_test, knn_y_pred) * 100
      knn_precision = precision_score(y_test, knn_y_pred,zero_division=1) * 100
      knn_recall = recall_score(y_test, knn_y_pred) * 100
      knn_f1 = f1_score(y_test, knn_y_pred) * 100
      conf_matrix = confusion_matrix(y_test, knn_y_pred)
      knn_error_rate = ((conf_matrix[0, 1] + conf_matrix[1, 0]) / float(conf_matrix.sum()))*100

      print("Metrics with Hyperparameter Tuning: Normal")
      print("Accuracy: {:.2f}%".format(knn_accuracy))
      print("Precision:{:.2f}%".format(knn_precision))
      print("Recall:{:.2f}%".format(knn_recall))
      print("F1 Score{:.2f}%".format(knn_f1))
      print("Error Rate:{:.2f}%".format(knn_error_rate))
```

```
Metrics with Hyperparameter Tuning: Normal
Accuracy: 83.37%
Precision:67.72%
Recall:95.52%
F1 Score79.26%
Error Rate:16.63%
```

Figure 44 - KNN - Model without hyperparameter tuning

Here we have trained a model using SVM algorithm and calculated all the required metrics. As you can see the accuracy of the knn is 83.37%, precision is 67.72%, recall is 95.52%, f1score is 79.26% and error rate is 16.63%.

II. With Hyperparameter tunning

With Hyperparameter

```
[65]: def evaluate_knn_model(parameters, x_train, y_train, x_test, y_test):
          knn_model = KNeighborsClassifier(**parameters)
          knn_model.fit(x_train, y_train)
          y_pred = svm_model.predict(x_test)

          accuracy = accuracy_score(y_test, y_pred) * 100
          precision = precision_score(y_test, y_pred, zero_division=1) * 100
          recall = recall_score(y_test, y_pred) * 100
          f1 = f1_score(y_test, y_pred) * 100
          conf_matrix = confusion_matrix(y_test, y_pred)
          error_rate = ((conf_matrix[0, 1] + conf_matrix[1, 0]) / float(conf_matrix.sum())) * 100

          return accuracy, precision, recall, f1, error_rate
```

Figure 45 - KNN - Function for model training with hyperparameter tuning

Here we have created a function which we will be using to train knn models with different hyper parameters this returns accuracy, precision, recall, f1score and error rate.

1. Hyperparameter tunning – 1

Hyperparameter tuning 1

```
[71]: knn_parameter_1 = {'n_neighbors': 6, 'weights': 'uniform', 'p': 2}
      knn_result_1 = evaluate_knn_model(knn_parameter_1, x_train, y_train, x_test, y_test)
      print("Metrics with Hyperparameter Tuning: 9")
      print("Accuracy: {:.2f}%".format(knn_result_1[0]))
      print("Precision: {:.2f}%".format( knn_result_1[1]))
      print("Recall: {:.2f}%".format(knn_result_1[2]))
      print("F1 Score: {:.2f}%".format( knn_result_1[3]))
      print("Error Rate: {:.2f}%".format( knn_result_1[4]))
```

```
Metrics with Hyperparameter Tuning: 9
Accuracy: 83.87%
Precision: 69.27%
Recall: 92.54%
F1 Score: 79.23%
Error Rate: 16.13%
```

Figure 46 - KNN - Model with Hyperparameter tuning - 1

Here, we have changed the parameter and it seems the accuracy is 83.87%.

2. Hyperparameter tunning – 2

Hyperparameter tuning 2

```
[72]: knn_parameter_2 ={'n_neighbors': 10, 'weights': 'distance', 'p': 1}
      knn_result_2 = evaluate_knn_model(knn_parameter_2, x_train, y_train, x_test, y_test)
      print("Metrics with Hyperparameter Tuning: 2")
      print("Accuracy: {:.2f}%".format(knn_result_2[0]))
      print("Precision: {:.2f}%".format( knn_result_2[1]))
      print("Recall: {:.2f}%".format(knn_result_2[2]))
      print("F1 Score: {:.2f}%".format( knn_result_2[3]))
      print("Error Rate: {:.2f}%".format( knn_result_2[4]))
```

```
Metrics with Hyperparameter Tuning: 9
Accuracy: 73.61%
Precision: 55.94%
Recall: 97.26%
F1 Score: 71.03%
Error Rate: 26.39%
```

Figure 47 - KNN - Model with Hyperparameter tuning - 2

Here, we have changed the parameter and it seems the accuracy is 73.61%.

3. Hyperparameter tunning – 3

Hyperparameter tuning 3

```
[73]: knn_parameter_3 ={'n_neighbors': 3, 'weights': 'uniform', 'p': 1}
      knn_result_3 = evaluate_knn_model(knn_parameter_3, x_train, y_train, x_test, y_test)
      print("Metrics with Hyperparameter Tuning: 3")
      print("Accuracy: {:.2f}%".format(knn_result_3[0]))
      print("Precision: {:.2f}%".format( knn_result_3[1]))
      print("Recall: {:.2f}%".format(knn_result_3[2]))
      print("F1 Score: {:.2f}%".format( knn_result_3[3]))
      print("Error Rate: {:.2f}%".format( knn_result_3[4]))
```

```
Metrics with Hyperparameter Tuning: 3
Accuracy: 76.84%
Precision: 62.06%
Recall: 78.11%
F1 Score: 69.16%
Error Rate: 23.16%
```

Figure 48 - KNN - Model with Hyperparameter tuning – 3

Here, we have changed the parameter and it seems the accuracy is 76.84%.

4. Hyperparameter tunning – 4

Hyperparameter tuning 4

```
[75]: knn_parameter_4 ={'n_neighbors': 15, 'weights': 'distance', 'p': 2}
knn_result_4 = evaluate_knn_model(knn_parameter_4, x_train, y_train, x_test, y_test)
print("Metrics with Hyperparameter Tuning: 4")
print("Accuracy: {:.2f}%".format(knn_result_4[0]))
print("Precision: {:.2f}%".format( knn_result_4[1]))
print("Recall: {:.2f}%".format(knn_result_4[2]))
print("F1 Score: {:.2f}%".format( knn_result_4[3]))
print("Error Rate: {:.2f}%".format( knn_result_4[4]))
```

```
Metrics with Hyperparameter Tuning: 4
Accuracy: 79.98%
Precision: 63.25%
Recall: 95.02%
F1 Score: 75.94%
Error Rate: 20.02%
```

Figure 49 - KNN - Model with Hyperparameter tuning – 4

Here, we have changed the parameter and it seems the accuracy is 79.98%.

5. Hyperparameter tunning – 5

Hyperparameter tuning 5

```
[76]: knn_parameter_5 ={'n_neighbors': 8, 'weights': 'uniform', 'p': 2}
knn_result_5 = evaluate_knn_model(knn_parameter_5, x_train, y_train, x_test, y_test)
print("Metrics with Hyperparameter Tuning: 5")
print("Accuracy: {:.2f}%".format(knn_result_5[0]))
print("Precision: {:.2f}%".format( knn_result_5[1]))
print("Recall: {:.2f}%".format(knn_result_5[2]))
print("F1 Score: {:.2f}%".format( knn_result_5[3]))
print("Error Rate: {:.2f}%".format( knn_result_5[4]))
```

```
Metrics with Hyperparameter Tuning: 5
Accuracy: 82.80%
Precision: 67.51%
Recall: 93.03%
F1 Score: 78.24%
Error Rate: 17.20%
```

Figure 50 - KNN - Model with Hyperparameter tuning – 5

Here, we have changed the parameter and it seems the accuracy is 82.80%.

6. Hyperparameter tunning – 6

Hyperparameter tuning 6

```
[77]: knn_parameter_6 = {'n_neighbors': 5, 'weights': 'distance', 'p': 1}
knn_result_6 = evaluate_knn_model(knn_parameter_6, x_train, y_train, x_test, y_test)
print("Metrics with Hyperparameter Tuning: 6")
print("Accuracy: {:.2f}%".format(knn_result_6[0]))
print("Precision: {:.2f}%".format( knn_result_6[1]))
print("Recall: {:.2f}%".format(knn_result_6[2]))
print("F1 Score: {:.2f}%".format( knn_result_6[3]))
print("Error Rate: {:.2f}%".format( knn_result_6[4]))
```

```
Metrics with Hyperparameter Tuning: 6
Accuracy: 75.35%
Precision: 57.78%
Recall: 96.02%
F1 Score: 72.15%
Error Rate: 24.65%
```

Figure 51 - KNN - Model with Hyperparameter tuning – 6

Here, we have changed the parameter and it seems the accuracy is 75.35%.

7. Hyperparameter tunning – 7

Hyperparameter tuning 7

```
[78]: knn_parameter_7 ={'n_neighbors': 12, 'weights': 'uniform', 'p': 2}
knn_result_7 = evaluate_knn_model(knn_parameter_7, x_train, y_train, x_test, y_test)
print("Metrics with Hyperparameter Tuning: 7")
print("Accuracy: {:.2f}%".format(knn_result_7[0]))
print("Precision: {:.2f}%".format( knn_result_7[1]))
print("Recall: {:.2f}%".format(knn_result_7[2]))
print("F1 Score: {:.2f}%".format( knn_result_7[3]))
print("Error Rate: {:.2f}%".format( knn_result_7[4]))
```

```
Metrics with Hyperparameter Tuning: 7
Accuracy: 82.05%
Precision: 66.20%
Recall: 94.03%
F1 Score: 77.70%
Error Rate: 17.95%
```

Figure 52 - KNN - Model with Hyperparameter tuning – 7

Here, we have changed the parameter and it seems the accuracy is 82.05%.

8. Hyperparameter tunning – 8

Hyperparameter tuning 8

```
[79]: knn_parameter_8 ={'n_neighbors': 7, 'weights': 'distance', 'p': 1}
knn_result_8 = evaluate_knn_model(knn_parameter_8, x_train, y_train, x_test, y_test)
print("Metrics with Hyperparameter Tuning: 8")
print("Accuracy: {:.2f}%".format(knn_result_8[0]))
print("Precision: {:.2f}%".format( knn_result_8[1]))
print("Recall: {:.2f}%".format(knn_result_8[2]))
print("F1 Score: {:.2f}%".format( knn_result_8[3]))
print("Error Rate: {:.2f}%".format( knn_result_8[4]))
```

```
Metrics with Hyperparameter Tuning: 8
Accuracy: 73.95%
Precision: 56.24%
Recall: 97.51%
F1 Score: 71.34%
Error Rate: 26.05%
```

Figure 53 - KNN - Model with Hyperparameter tuning – 8

Here, we have changed the parameter and it seems the accuracy is 73.95%.

9. Hyperparameter tunning – 9

Hyperparameter tuning 9

```
[80]: knn_parameter_9 ={'n_neighbors': 6, 'weights': 'uniform', 'p': 1}
knn_result_9 = evaluate_knn_model(knn_parameter_9, x_train, y_train, x_test, y_test)
print("Metrics with Hyperparameter Tuning: 9")
print("Accuracy: {:.2f}%".format(knn_result_9[0]))
print("Precision: {:.2f}%".format( knn_result_9[1]))
print("Recall: {:.2f}%".format(knn_result_9[2]))
print("F1 Score: {:.2f}%".format( knn_result_9[3]))
print("Error Rate: {:.2f}%".format( knn_result_9[4]))
```

```
Metrics with Hyperparameter Tuning: 9
Accuracy: 76.34%
Precision: 59.21%
Recall: 92.79%
F1 Score: 72.29%
Error Rate: 23.66%
```

Figure 54- KNN - Model with Hyperparameter tuning – 9

Here, we have changed the parameter and it seems the accuracy is 76.34%.

10. Hyperparameter tunning – 10

```
Hyperparameter tuning 10

[81]: knn_parameter_10 ={'n_neighbors': 20, 'weights': 'distance', 'p': 2}
      knn_result_10 = evaluate_knn_model(knn_parameter_10, x_train, y_train, x_test, y_test)
      print("Metrics with Hyperparameter Tuning: 10")
      print("Accuracy: {:.2f}%".format(knn_result_10[0]))
      print("Precision: {:.2f}%".format( knn_result_10[1]))
      print("Recall: {:.2f}%".format(knn_result_10[2]))
      print("F1 Score: {:.2f}%".format( knn_result_10[3]))
      print("Error Rate: {:.2f}%".format( knn_result_10[4]))

      Metrics with Hyperparameter Tuning: 10
      Accuracy: 78.74%
      Precision: 61.75%
      Recall: 94.78%
      F1 Score: 74.78%
      Error Rate: 21.26%
```

Figure 55- KNN - Model with Hyperparameter tuning - 10

Here, we have changed the parameter and it seems the accuracy is 78.74%.

III. KNN Accuracy Evaluation

```
%matplotlib inline


Headings = ['No-Hyperparameter','Hyperparameter - 1','Hyperparameter - 2',
            'Hyperparameter - 3','Hyperparameter - 4','Hyperparameter - 5',
            'Hyperparameter - 6','Hyperparameter - 7','Hyperparameter - 8',
            'Hyperparameter - 9','Hyperparameter - 10']
Heading_values = [knn_accuracy, knn_result_1[0],knn_result_2[0],
                  knn_result_3[0],knn_result_4[0],knn_result_5[0],
                  knn_result_6[0],knn_result_7[0],knn_result_8[0],
                  knn_result_9[0],knn_result_10[0]]
max_value = Heading_values.index(max(Heading_values))
plt.figure(figsize=(20, 5))
plt.title('KNN accuracy with different hyperparameters')
bars = plt.bar(Headings, Heading_values)
bars[max_value].set_color('purple')
for i, v in enumerate(Heading_values):
    plt.text(i, v + 1, str(round(v, 2)), ha='center',
             va='bottom', fontweight='bold', color='black')
plt.xticks(rotation="vertical")
```

Figure 56 - Knn accuracy Evaluation - Code

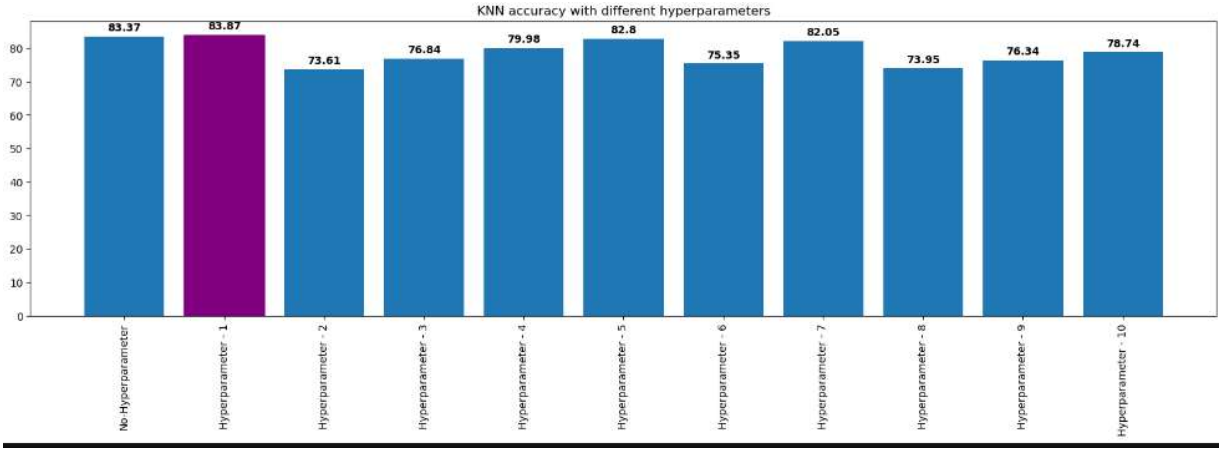


Figure 57 - KNN accuracy Evaluation – Graph

From the comparison we can see that the 2nd model which is with hyperparameter is the one with the highest accuracy

3.4.3.4.3 Naïve Bayes

Naive Bayes

We will not be doing any hyperparameter tuning in this algorithm

[35]:
```
NB = GaussianNB()
NB.fit(x_train,y_train)
```

[35]:
```
GaussianNB()
```

[37]:
```
nb_y_pred = NB.predict(x_test)
nb_accuracy = accuracy_score(y_test, nb_y_pred) * 100
nb_precision = precision_score(y_test, nb_y_pred,zero_division=1) * 100
nb_recall = recall_score(y_test, nb_y_pred) * 100
nb_f1 = f1_score(y_test, nb_y_pred) * 100
conf_matrix = confusion_matrix(y_test, nb_y_pred)
nb_error_rate = ((conf_matrix[0, 1] + conf_matrix[1, 0]) / float(conf_matrix.sum()))*100

print("Metric:")
print("Accuracy: {:.2f}%".format(nb_accuracy))
print("Precision:{:.2f}%".format(nb_precision))
print("Recall:{:.2f}%".format(nb_recall))
print("F1 Score{:.2f}%".format(nb_f1))
print("Error Rate:{:.2f}%".format(nb_error_rate))
```

```
Metric:
Accuracy: 92.80%
Precision:88.89%
Recall:89.55%
F1 Score89.22%
Error Rate:7.20%
```

Figure 58 - Naive Bayes Model Scores

3.4.3.4.4 Comparison of highest accuracy from all Algorithms

```
%matplotlib inline


Headings = ['SVM','KNN','Naive Bayes']
Heading_values = [result_1[0], knn_result_1[0],nb_accuracy]
max_value = Heading_values.index(max(Heading_values))
plt.figure(figsize=(20, 5))
plt.title('Accuracy Comparison of different models')
bars = plt.bar(Headings, Heading_values)
bars[max_value].set_color('purple')
for i, v in enumerate(Heading_values):
    plt.text(i, v + 1, str(round(v, 2)), ha='center',
             va='bottom', fontweight='bold', color='black')
plt.xticks(rotation="vertical")
```

Figure 59 - Comparison of all algorithm's best models - code

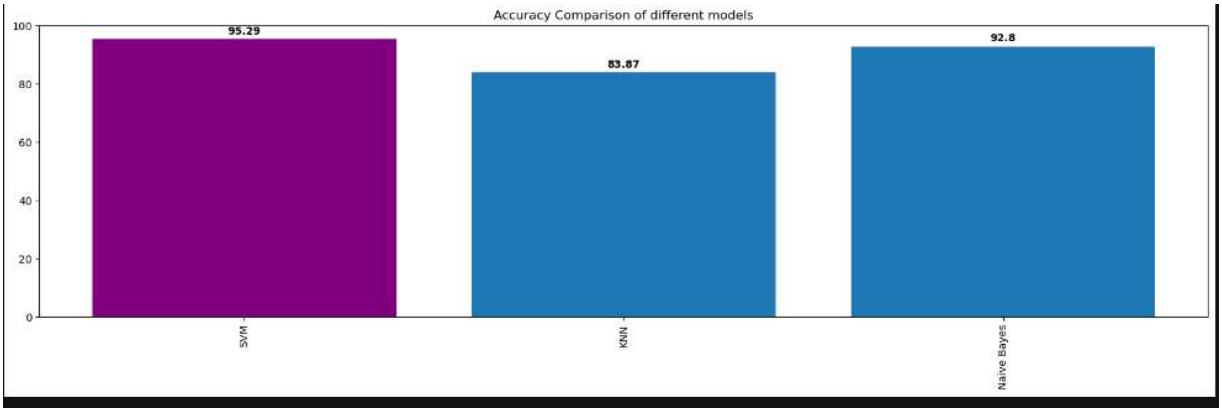


Figure 60 - Comparison of all algorithm's best models - graph

3.4.4 Achieved Results

Now that we have trained and tested the models, we will compare each algorithm's models with each other.

3.4.4.1 SVM (Support Vector Machine)

SVM (Support Vector Machine)								
Scenario				**Accuracy**	**Precision**	**Recall**	**F1 Score**	**Error Rate**
Without Hyperparameter Tuning								
No Hyper parameter				91.15%	99.66	73.63	84.69	8.85
With Hyper parameters								
S. N	**C**	**kernel**	**gamma**	-				
Hyperparameter tuning -1	**0.1**	**linear**	**scale**	**95.29%**	**89.12%**	**97.76%**	**93.24%**	**4.71%**
Hyperparameter tuning -2	1	rbf	scale	91.15%	99.66%	73.63%	84.69%	8.85%
Hyperparameter tuning -3	10	poly	0.1	86.10%	74.38%	88.81%	80.95%	13.90%
Hyperparameter tuning -4	0.01	linear	scale	95.29%	89.47%	97.26%	93.21%	4.71%
Hyperparameter tuning -5	0.5	rbf	auto	68.35%	100.00%	4.73%	9.03%	31.68%
Hyperparameter tuning -6	5	poly	0.01	86.68%	94.14%	63.93%	76.15%	13.32%
Hyperparameter tuning -7	0.001	linear	scale	89.66%	99.29%	69.40%	81.70%	10.34%
Hyperparameter tuning -8	20	poly	auto	67.25%	87.50%	1.74%	3.41%	32.75%
Hyperparameter tuning -9	0.005	linear	0.01	95.04%	90.14%	95.52%	92.75%	4.96%
Hyperparameter tuning -10	0.2	rbf	scale	85.19%	99.56%	55.72%	71.45%	14.81%

Table 2 – SVM – Models Comparison

In the above table, comparison of models has been done. There are two types of models one with hyper parameters and one without hyper parameters. The one without hyper parameter has accuracy of 91.15%, precision of 99.66%, recall of 73.63%, f1-score 84.69%, and error rate of 8.85%. However, it seems that with some hyper parameter tunning we are able to get more accuracy out of the datasets. We have achieved 95.29% accuracy when the C is 0.1, kernel is linear, and gamma is scale.

3.4.4.2 KNN (K-Nearest Neighbours)

KNN (K-Nearest Neighbors)							
Scenario				**Accuracy**	**Precision**	**Recall**	**F1 Score**	**Error Rate**
Without Hyperparameter Tuning								
No Hyper parameter				83.37%	67.72%	95.52%	79.26%	16.63%
With Hyper parameters								
S. N	**n_neigbours**	**weights**	p	-				
Hyperparameter tuning -1	**6**	**uniform**	**2**	**83.87%**	**69.27%**	**32.54%**	**79.23%**	**16.13%**
Hyperparameter tuning -2	10	distance	1	73.61%	55.94%	97.26%	71.03%	26.39%
Hyperparameter tuning -3	3	uniform	1	76.84%	62.06%	78.11%	69.16%	23.16%
Hyperparameter tuning -4	15	distance	2	79.98%	63.25%	95.02%	75.94%	20.02%
Hyperparameter tuning -5	8	uniform	2	82.80%	67.51%	93.03%	78.24%	17.20%
Hyperparameter tuning -6	5	distance	1	75.35%	57.78%	96.02%	72.15%	24.65%
Hyperparameter tuning -7	12	uniform	2	82.05%	66.20%	94.03%	77.70%	17.95%
Hyperparameter tuning -8	7	distance	1	73.95%	56.24%	97.51%	71.34%	26.05%
Hyperparameter tuning -9	6	uniform	1	76.34%	59.21%	92.79%	72.29%	23.66%
Hyperparameter tuning -10	20	distance	2	78.74%	61.75%	94.78%	74.78%	21.26%

Table 3 - KNN - Model Comparison

In the above table, comparison of models has been done. There are two types of models one with hyper parameters and one without hyper parameters. The one without hyper parameter has accuracy of 83.37%, precision of 67.72%, recall of 95.52%, f1-score of 79.26%, and error rate of 16.63%. However, it seems that with some hyper parameter tunning we are able to get more accuracy out of the datasets. We have achieved 83.87% accuracy when the number of neighbours is 6, weights is uniform, and p is 2.

3.4.4.3 Naïve Bayes

For this algorithm, only one modelling has been done. The accuracy for that model is 92.80%, precision is 88.89%, recall is 89.55%, f1-score is 89.22% and error-rate is 7.20%.

3.4.4.4 Comparison of each algorithm most accurate models

Comparison of each algorithm most accurate models					
Algorithms	**Accuracy**	**Precision**	**Recall**	**F1-Score**	**Error Rate**
SVM	**95.29%**	**89.12%**	**97.76%**	**93.24%**	**4.71%**
KNN	83.87%	69.27%	32.54%	79.23%	16.13%
Naïve Bayes	92.80%	88.89%	89.55%	89.22%	7.20%

Table 4 - Comparison of each algorithm most accurate models

As we can see in the table above, the most accurate algorithm of all models is SVM (Support Vector Machine) with the accuracy score of 95.29%. Not only that it seems to be more precise and have more recall and f1-score and also have less error rate than other. Therefore, in conclusion it seems, the SVM is better in every way according to the model trained with the dataset.

11. Conclusion

4.1 Analysis of the work done

Research on spam filtering using AI and machine learning highlights the present and future possibilities of this technology. Analysing various AI types—from simple reactive robots to sophisticated self-awareness—offers a framework for comprehending the evolution of AI. Applications of machine learning, especially deep learning via neural networks, are significant in a number of domains, including spam filters and self-driving automobiles. Studies on spam detection show how ML algorithms, such as Naive Bayes and SVM, are replacing traditional methods because they are more accurate and flexible. These algorithms consistently perform well, as demonstrated by an analysis of five research articles. This makes them promising options in the field spam filtering. In conclusion, we have done our own test of effectiveness of different algorithms for spam detection. After the model training and testing, it seems among the three algorithm that we choose to test the spam detection, it seems SVM came on top not only being the most accurate but also with less error rate as well as with more precision.

4.2 Solution that addresses real world problems

The machine learning techniques SVM, Naive Bayes, and KNN are chosen in order to tackle the common problem of spam. These algorithms were chosen because to their demonstrated effectiveness in previous research articles. Improving the accuracy of spam filtering and reducing the problems caused by spams are the main goals of the solution.

Application in Real-world

Cybersecurity

Cybersecurity measures are strengthened by the application of machine learning algorithms, which offer preventive protection against any risks hidden in spam emails.

Protection against phishing attacks

The application of spam filtering using machine learning will ensure a safer online experience by detecting and blocking phishing emails, which will stop unwanted access to any information that is not supposed to be in the hands of unwanted person.

Business and Individual Benefits

When workers spend less time sorting through spam and more time on important duties, businesses benefit from increased operational efficiency. Furthermore, by protecting companies from email-based dangers and avoiding possible breaches of security and phishing schemes, an efficient spam detection system maintains brand reputation. Also, people feel more confident when interacting with others online because they are aware that their lines of communication are protected from spam, which enhances the online experience.

4.3 Limitations of systems

4.3.1 Computational Resources

Any machine learning project must have computational resources, and the limitations of these resources can have a big impact on the system's development and performance. Limited computational resources can include things like processor power, memory availability, and time limits. These can make it difficult for the implemented solution to scale and operate efficiently. These limitations may have an impact on the system's overall responsiveness, optimization efforts, and model training process. In my case, due to the limitation of these resources, the I had to reduce the size of data to train the model. Also due to this, the training and testing was taking long time.

4.3.2 Data limitations

A crucial component of the study is the data, which has a big impact on how reliable and broadly applicable the machine learning model is. Fundamental factors to consider are the dataset's size and representativeness. A tiny or biased dataset can make it more difficult for the model to identify patterns that broadly apply to a variety of real-world situations. Furthermore, problems with missing or insufficient data present difficulties during training and may have an impact on the model's overall performance.

In my case, the dataset was not clean as well, therefore, it needed a lot of cleaning and preparation before it was ready for modelling. Also, due to computer limitations I wasn't able to put more data into modelling.

4.4 Future works

4.4.1 Group Methods

Exploring on how well group methods which incorporate several machine learning algorithms at the same time to increase the accuracy of spam detection overall.

4.4.2 Deep Learning Approaches

Investigating the use of deep learning methods for higher-level and detailed spam filtering, such as long short-term memory (LSTM) networks and recurrent neural networks (RNNs).

4.4.3 Deployment

Analyse approaches for implementing the spam detection model in real-world situations. This entails looking into deployment frameworks, integrating with current email systems.

4.4.4 User Interface Integration

To give end users a smooth experience, create a simple user interface or integrate with already-existing email services. Think about methods to improve user interaction with the spam detection system and user feedback tools.

4.4.5 Expandability and Interpretability

Investigate ways to improve the interpretability of the spam detection model so that consumers are able to understand the decision-making process. This might include producing human-readable explanations for model predictions or integrating explain ability techniques.

12. References

Brown, S., 2021. *Machine learning, explained.* [Online]
Available at: https://mitsloan.mit.edu/ideas-made-to-matter/machine-learning-explained
[Accessed 9 December 2023].

Copeland, B., 2023. *artificial intelligence.* [Online]
Available at: https://www.britannica.com/technology/artificial-intelligence
[Accessed 6 December 2023].

Ellis, D. R., 2022. *What is Anaconda for Python & Why Should You Learn it?.* [Online]
Available at: https://blog.hubspot.com/website/anaconda-python
[Accessed 16 January 2024].

Engineering, C., n.d. *Top Machine Learning Applications by Industry: 6 Machine Learning Examples.* [Online]
Available at: https://bootcamp.cvn.columbia.edu/blog/machine-learning-applications/
[Accessed 9 December 2023].

Erin Rodrigue,Flori Needle, 2023. *4 Types of Artificial Intelligence & What Marketers Are Using Most (Research).* [Online]
Available at: https://blog.hubspot.com/marketing/types-of-ai
[Accessed 6 December 2023].

Frankenfield, J., 2022. *Weak AI (Artificial Intelligence): Examples and Limitations.* [Online]
Available at: https://www.investopedia.com/terms/w/weak-ai.asp
[Accessed 6 December 2023].

Gatefy, 2021. *7 most common types of email spam.* [Online]
Available at: https://gatefy.com/blog/most-common-types-email-spam/
[Accessed 10 December 2023].

geeksforgeeks, 2023. *Introduction to Matplotlib.* [Online]
Available at: https://www.geeksforgeeks.org/python-introduction-matplotlib/
[Accessed 16 January 2024].

Gillis, A. S., 2022. *Excel.* [Online]
Available at: https://www.techtarget.com/searchenterprisedesktop/definition/Excel
[Accessed 16 January 2023].

GRIGGS, A., 2022. *How to Set up Spam Settings in Yahoo Email.* [Online]
Available at: https://turbofuture.com/internet/How-to-Setup-Spam-Settings-in-Yahoo-Email-Including-Marking-and-Unmarking-Email-Messages-Plus-Understanding-Spam
[Accessed 18 December 2023].

Hope, C., 2020. *Draw.io.* [Online]
Available at: https://www.computerhope.com/jargon/d/drawio.htm
[Accessed 16 January 2024].

IBM, n.d. *What is the k-nearest neighbors algorithm?.* [Online]
Available at: https://www.ibm.com/topics/knn
[Accessed 16 December 2023].

Insights, S. M., 2023. *Artificial Intelligence - Worldwide.* [Online]
Available at: https://www.statista.com/outlook/tmo/artificial-intelligence/worldwide#market-size
[Accessed 9 December 2023].

Jablonski, J., n.d. *Natural Language Processing With Python's NLTK Package.* [Online]
Available at: https://realpython.com/nltk-nlp-python/#:~:text=NLTK%2C%20or%20Natural%20Language%20Toolkit,and%20contains%20human%2Dreadable%20text.
[Accessed 16 January 2024].

Kanade, V., 2022. *What Is Machine Learning? Definition, Types, Applications, and Trends for 2022.* [Online]
Available at: https://www.spiceworks.com/tech/artificial-intelligence/articles/what-is-ml/
[Accessed 6 December 2023].

Kingshuk Debnath, Nirmalya Kar, 2022. Email Spam Detection using Deep Learning Approach. In: *2022 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COM-IT-CON).* Faridabad: IEEE, pp. 37-41.

MailChannels, n.d. *What is Spam Filtering.* [Online]
Available at: https://blog.mailchannels.com/what-is-spam-filtering/
[Accessed 10 December 2023].

Mansoor RAZA, Nathali Dilshani Jayasinghe, Muhana Magboul Ali Muslam, 2021. A Comprehensive Review on Email Spam Classification using Machine Learning Algorithms. In: *2021 International Conference on Information Networking (ICOIN).* Jeju Island: IEEE, pp. 327-332.

Md Rafiqul Islam, Morshed U. Chowdhury, 2015. Spam filtering using Ml algorithms. *Spam filtering using Ml algorithms*, 12 February, p. 425.

Menon, K., 2023. *Different Types of Machine Learning: Exploring AI's Core.* [Online]
Available at: https://www.simplilearn.com/tutorials/machine-learning-tutorial/types-of-machine-learning
[Accessed 18 December 2023].

M, R., 2020. *What Is Numpy Used For In Python?.* [Online]
Available at: https://www.activestate.com/resources/quick-reads/what-is-numpy-used-for-in-python/
[Accessed 16 January 2024].

M, R., 2021. *What is Scikit-Learn in Python?.* [Online]
Available at: https://www.activestate.com/resources/quick-reads/what-is-scikit-learn-in-python/
[Accessed 16 January 2024].

Oza, H., 2021. *Importance And Benefits Of Artificial Intelligence.* [Online]
Available at: https://www.hdatasystems.com/blog/importance-and-benefits-of-artificial-intelligence
[Accessed 9 December 2023].

Pedamkar, P., 2023. *Importance of Artificial Intelligence.* [Online]
Available at: https://www.educba.com/importance-of-artificial-intelligence/
[Accessed 9 December 2023].

Petrosyan, A., 2023. *Global spam volume as percentage of total e-mail traffic from 2011 to 2022.* [Online]
Available at: https://www.statista.com/statistics/420400/spam-email-traffic-share-annual/
[Accessed 10 December 2022].

Pickle, B., 2022. *Spam.* [Online]
Available at: https://techterms.com/definition/spam
[Accessed 10 December 2023].

Prazwal Thakur, Kartik Joshi, Prateek Thakral, Shruti Jain, 2022. Detection of Email Spam using Machine Learning Algorithms: A Comparative Study. In: *2022 8th International Conference on Signal Processing and Communication (ICSC).* Noida: IEEE, pp. 349-352.

Raj, R., n.d. *Email Spam and Non-spam filtering using machine Learning.* [Online]
Available at: https://www.enjoyalgorithms.com/blog/email-spam-and-non-spam-filtering-using-machine-learning
[Accessed 12 December 2023].

Ray, S., 2023. *Naive Bayes Classifier Explained: Applications and Practice Problems of Naive Bayes Classifier.* [Online]
Available at: https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/
[Accessed 16 December 2023].

Rouse, M., 2023. *Natural Language Toolkit.* [Online]
Available at: https://www.techopedia.com/definition/30343/natural-language-toolkit-nltk
[Accessed 16 January 2024].

Saini, A., 2023. *Guide on Support Vector Machine (SVM) Algorithm.* [Online]
Available at: https://www.analyticsvidhya.com/blog/2021/10/support-vector-machinessvm-a-complete-guide-for-beginners/#:~:text=optimize%20in%20SVM.-

,Margin%20in%20Support%20Vector%20Machine,and%20b%20is%20an%20offset.&text=If%20the%20value%20of%20w,it%20is%20a%20negativ
[Accessed 16 December 2023].

Sandhi Kranthi Reddy, T Maruthi Padmaja, 2019. Non Machine and Machine Learning Spam. *International Journal of Recent Technology and Engineering,* 7(5S4), p. 5.

Schroer, A., 2023. *Artificial Intelligence.What Is Artificial Intelligence (AI)? How Does AI Work?.* [Online]
Available at: https://builtin.com/artificial-intelligence
[Accessed 6 December 2023].

Shah, P., 2022. *Top 3 Ways to Stop Emails from Known Senders Going in Gmail's Spam Folder.* [Online]
Available at: https://www.guidingtech.com/ways-to-stop-emails-from-known-senders-going-in-gmails-spam-folder/
[Accessed 18 December 2023].

Slavin, B., 2022. *What Is Spam Filtering And How Does It Work?.* [Online]
Available at: https://www.duocircle.com/email-hosting/what-is-spam-filtering-and-how-does-it-work
[Accessed 12 December 2023].

Sridevi Gadde, A.Lakshmanarao, S.Satyanarayana, 2021. SMS Spam Detection using Machine Learning. In: *2021 7th International Conference on Advanced Computing and Communication Systems (ICACCS).* Coimbatore: IEEE, pp. 358-362.

S, S., 2020. *What Is Pandas in Python? Everything You Need to Know.* [Online]
Available at: https://www.activestate.com/resources/quick-reads/what-is-pandas-in-python-everything-you-need-to-know/
[Accessed 16 January 2024].

Trivedi, S. K., 2016. A study of machine learning classifiers for spam detection. In: *2016 4th International Symposium on Computational and Business Intelligence (ISCBI).* Olten: IEEE, pp. 176-180.

tutorialspoint, 2022. *Python - Regular Expressions.* [Online]
Available at: https://www.tutorialspoint.com/python/python_reg_expressions.htm
[Accessed 16 January 2024].

Wahab, F., 2021. *How to fix Outlook keeps sending emails to Junk or Spam Folder.* [Online]
Available at: https://www.addictivetips.com/windows-tips/outlook-sending-emails-to-junk-folder/
[Accessed 18 December 2023].