Technical Coding Research Innovation, Navi Mumbai, Maharashtra, India-410206

A Case-Study Submitted for the requirement of
Technical Coding Research Innovation

For the Internship Project work done during

ARTIFICIAL INTELLIGENCE INTERNSHIP PROGRAM

by

Pritam Dhumal (TCRIA7R88)

Onkar Vyavahare (TCRIA7R89)

Abstract

This report presents an overview of my artificial intelligence (AI) internship experience till now at TCR Innovation. During the internship, I gained practical experience in various AI techniques, including Python libraries, machine learning models and more. The report discusses my training period, including the lessons learned, and the assignments that I completed during the internship. The report also highlights the ethical considerations surrounding AI and the impact it can have on privacy and the workforce. Overall, this internship experience was an excellent opportunity to apply theoretical knowledge to real-world problems, and I believe it has prepared me well for a career in the exciting field of AI**.**

Acknowledgement

I would like to express my sincere gratitude to the team at TCR Innovation for providing me with the opportunity to intern with their organization and work on an exciting AI project.

I would also like to thank the entire team at TCR Innovation for their warm welcome, support, and collaboration during the internship. I appreciate their willingness to share their knowledge and expertise with me and for providing me with a challenging yet rewarding experience.

Internship Overview

The internship at TCR Innovation in the domain of AI and ML consists of 5 phases and over the period of 4 months.

Phase 1) Learning basics of python, setting up jupyter notebook, OOPs, libraries and some of the mathematical concepts

Phase 2) Learning Data Preprocessing, Data Visualization.

Phase 3) Learning Machine Learning Algorithms & Implementation

Phase 4) Learning NLP & Deep Learning

Phase 5) Project

Introduction

Artificial intelligence (AI) is a rapidly growing field with significant potential to revolutionize many aspects of our lives. As a result, there is an increasing demand for professionals with the skills and knowledge to develop and implement AI solutions. My internship at TCR Innovation provided me with a unique opportunity to gain practical experience in AI and learn how it can be leveraged to solve real-world problems.

This report aims to provide an overview of my AI internship experience at TCR Innovation. It will discuss the training period, including the lessons learned, and the assignments completed during the internship. The report will also examine the ethical considerations surrounding AI and its impact on privacy and the workforce.

Overall, the internship was a valuable and rewarding experience that provided me with the skills and knowledge to pursue a career in the exciting and rapidly growing field of AI.

My Learnings

Artificial Intelligence (AI) is the simulation of human intelligence in machines that are programmed to perform tasks that would typically require human cognition. AI technology has been around for decades, but recent advancements in computing power, big data, and machine learning algorithms have enabled significant breakthroughs in AI research.

AI can be divided into two main categories: narrow or weak AI and general or strong AI. Narrow AI is designed to perform specific tasks, such as image recognition or natural language processing, while general AI is capable of performing any intellectual task that a human can do.

There are several AI techniques and approaches, including:

1. Machine Learning: A type of AI that allows machines to learn from data and improve their performance without being explicitly programmed.

2. Deep Learning: A subfield of machine learning that uses artificial neural networks to learn from data, allowing machines to perform tasks like image or speech recognition.

3. Natural Language Processing (NLP): A field of AI that enables machines to understand and interpret human language.

AI has several applications across various industries, including healthcare, finance, transportation, and manufacturing. Some common examples of AI applications include virtual assistants, image recognition, fraud detection, and autonomous vehicles.

Despite its significant potential, AI also raises concerns about ethics, privacy, and the impact on the workforce.

Therefore, it is essential to continue researching and developing AI in a responsible and ethical manner.

- **Tools & Technologies**

  There are several tools and technologies used in AI development that enable developers to build complex AI systems. Some of the most popular tools and technologies in AI include:

  1. Programming Languages: Python is one of the most popular programming languages used in AI development due to its simplicity and ease of use. Other programming languages used in AI development include R, Java, and C++.
  2. Libraries: There are several libraries available in Python that enable developers to build AI systems. Some popular libraries include NumPy, Pandas, Scikit-Learn and matplotlib.

3. Machine Learning Models: There are several machine learning models used in AI development, such as Linear Regression, Logistic Regression, Decision Trees, Random Forests, and Support Vector Machines.
4. Natural Language Processing (NLP): NLP is a field of AI that deals with human language.Some popular NLP tools and technologies include Natural Language Toolkit (NLTK).
5. Deep Learning Frameworks: Deep learning is a subfield of machine learning that uses artificial neural networks to learn from data. Some popular deep learning frameworks include TensorFlow, PyTorch, and Keras.

❖ **Python Programming & Libraries**

Python is one of the most popular programming languages used in artificial intelligence (AI) development due to its simplicity, ease of use, and large number of libraries and tools available for machine learning and data analysis.

❖ **Python Libraries for AI Development**

There are several libraries available in Python that enable developers to build AI systems. Some of the most popular libraries used in AI development include:

1. NumPy - A library for numerical computing that provides support for arrays, matrices, and mathematical operations.
2. Pandas - A library for data analysis that provides support for data manipulation, cleansing, and visualization.
3. Scikit-Learn - A machine learning library that provides support for supervised and unsupervised learning algorithms, including regression, classification, and clustering. And many more.

❖ **Machine Learning**

Machine learning is a subfield of artificial intelligence (AI) that enables machines to learn from data and improve their performance over time without being explicitly programmed. In other words, machine learning algorithms learn from experience and can adapt to new data inputs.

Types of Machine Learning

There are three main types of machine learning:

1. Supervised Learning - In supervised learning, the machine learning model learns from labeled data. The model is trained on a dataset that includes input features and corresponding output labels. The goal is to train the model to predict the correct output label for new input data.
2. Unsupervised Learning - In unsupervised learning, the machine learning model learns from unlabeled data. The model is trained on a dataset that only includes input features, and the goal is to identify patterns or structures in the data.
3. Reinforcement Learning - In reinforcement learning, the machine learning model learns by interacting with an environment and receiving

feedback in the form of rewards or punishments. The goal is to learn a policy that maximizes the cumulative reward over time.

- **Machine Learning Algorithms**

  There are several machine learning algorithms used in AI development, including:

  1. Linear Regression
  2. Logistic Regression
  3. Decision Trees
  4. Support Vector Machines (SVM)
  5. Naive Bayes
  6. K-Nearest Neighbors (KNN)

  and many more.

- **Natural Language Processing :-**

  Natural Language Processing (NLP) is a branch of artificial intelligence (AI) that focuses on enabling machines to understand and interpret human language.

- **Deep Learning :-**

  Deep learning is a subset of machine learning that uses artificial neural networks to learn and make decisions.

Project

- **Introduction**

  In today's digital age, the problem of spam has become a major concern for individuals and organizations alike. Spam refers to unsolicited or unwanted messages that are sent in bulk, often with malicious intent or the aim of promoting products or services. Spam emails, text messages, and social media posts not only clutter communication channels but also pose security risks and inconvenience to users. Consequently, the development of robust spam detection systems has become a crucial necessity.

  The goal of this project is to design and implement a Spam-Ham classifier, a machine learning model capable of accurately distinguishing between spam and legitimate messages (ham). The classifier aims to automate the process of spam detection, helping individuals and organizations filter out unwanted messages effectively.

  The proposed classifier utilizes a combination of text preprocessing techniques and supervised learning algorithms to extract meaningful features from the text data and classify messages as either spam or ham. By leveraging the power of machine learning, the classifier aims to improve the accuracy and efficiency of spam detection, reducing the burden on users to manually identify and handle spam messages.

  The project encompasses several key components. Firstly, the text data is preprocessed to remove unnecessary characters, stop words, and perform tokenization. Various feature extraction methods, including bag-of-words, TF-IDF, and word embeddings, are employed to represent the textual content effectively.

  To build the classifier, different machine learning algorithms, such as Naive Bayes, Support Vector Machines (SVM), and Random Forests, are trained and evaluated. These algorithms are chosen for their effectiveness in text classification tasks and their ability to handle high-dimensional feature spaces.

  Furthermore, the project explores the potential of deep learning techniques for spam detection. Recurrent Neural Networks (RNN) and Convolutional Neural Networks (CNN) are investigated, as they offer the capability to capture sequential dependencies and spatial patterns in the text data, respectively. By incorporating deep learning models into the classification pipeline, the project aims to enhance the performance and robustness of the Spam-Ham classifier.

  The developed Spam-Ham classifier is evaluated using diverse benchmark datasets, including publicly available spam databases and custom-labeled datasets. Performance metrics such as accuracy, precision, recall, and F1-score are employed to assess the effectiveness of the classifiers and compare their performance against existing solutions.

The outcomes of this project have practical implications for various sectors, including email providers, social media platforms, and other digital communication services. Implementing an accurate and efficient spam detection system can significantly enhance the user experience, protect against malicious activities, and ensure a safe and clutter-free digital environment.

In conclusion, this project aims to contribute to the ongoing efforts in combating spam by developing a Spam-Ham classifier. The project leverages machine learning and deep learning techniques to build a robust and accurate classifier that can effectively identify and filter out spam messages. The successful implementation of the Spam-Ham classifier can have far-reaching implications, improving communication channels' security, efficiency, and user experience.

❖ **Objective :**

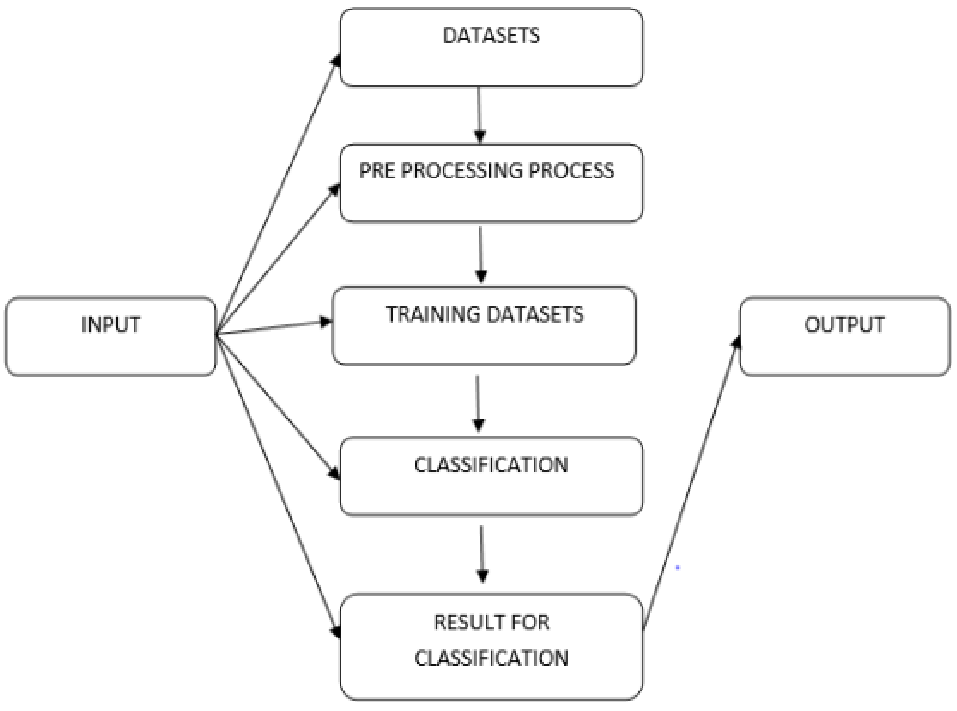
The primary objective of this project is to design and develop a Spam-Ham classifier that can accurately distinguish between spam and legitimate (ham) messages in various forms of digital communication, such as emails, text messages, and social media posts. The project aims to achieve the following specific objectives:

1. Data Preprocessing: Implement text preprocessing techniques to clean and normalize the textual data, removing irrelevant characters, stop words, and performing tokenization. This step aims to ensure that the input data is in a suitable format for feature extraction and classification.
2. Feature Extraction: Employ various feature extraction methods, including bag-of-words, TF-IDF, and word embeddings, to capture the relevant information and characteristics of the text data. These techniques aim to transform the textual content into numerical representations that can be used as input features for the classification models.
3. Machine Learning Model Selection and Training: Explore and compare the performance of different machine learning algorithms, such as Naive Bayes, Support Vector Machines (SVM), and Random Forests, for spam detection. Train the selected models on labeled datasets consisting of spam and ham messages, optimizing their parameters and hyperparameters to achieve high accuracy and robust classification performance.
4. Evaluation and Performance Metrics: Evaluate the developed Spam-Ham classifier using various performance metrics, including accuracy, precision, recall, and F1-score. Assess the model's effectiveness in accurately identifying and distinguishing between spam and ham messages. Compare the classifier's performance against existing solutions and benchmark datasets to measure its efficacy.

5. Deep Learning Exploration: Investigate the potential of deep learning techniques, specifically Recurrent Neural Networks (RNN) and Convolutional Neural Networks (CNN), for spam detection. Design and train deep learning models capable of capturing sequential dependencies and spatial patterns in the text data. Assess the performance of these models and compare them to traditional machine learning algorithms.
6. Generalization and Real-World Applicability: Evaluate the Spam-Ham classifier's generalization capability on unseen data and diverse datasets. Validate the model's effectiveness in real-world scenarios and its ability to handle different types of spam messages across various communication channels.
7. Practical Implications: Discuss the practical implications of the developed Spam-Ham classifier for email providers, social media platforms, and other digital communication services. Highlight the potential benefits of implementing the classifier in terms of improving user experience, enhancing security, and reducing the impact of spam messages.

By achieving these objectives, this project aims to contribute to the field of spam detection by developing an accurate and efficient Spam-Ham classifier that can effectively filter out unsolicited messages, thereby improving the quality and safety of digital communication.

❖ **Code & Implementation:**



Dataset:

The spam-ham project dataset consists of a collection of labeled messages that are used for training, evaluating, and testing the spam-ham classifier. The dataset is carefully curated to represent a diverse range of text data found in different digital communication channels such as emails, text messages, and social media posts.

The dataset comprises two main categories: spam and ham. The spam category includes messages that are unsolicited or unwanted, often sent in bulk, and may contain promotional content, phishing attempts, or other forms of malicious communication. The ham category consists of legitimate messages that are desired and expected by the recipient, including personal emails, work-related messages, and non-spam communication.

Data Preprocessing:

Data preprocessing is a crucial step in the development of a spam-ham classifier. It involves transforming and cleaning the raw text data to ensure that it is in a suitable format for feature extraction and classification. The following are the key steps involved in data preprocessing for the spam-ham project:

1. Text Cleaning: The raw text data often contains unnecessary characters, punctuation marks, or symbols that do not contribute to the classification task. These elements are removed to clean the text and reduce noise. Common techniques include removing special characters, punctuation, and HTML tags.
2. Tokenization: Tokenization is the process of breaking down the text data into individual tokens or words. It splits the text into meaningful units, such as individual words or subwords. This step is important as it helps in further processing and analysis of the text data. Tokenization can be performed using simple space-based splitting or more advanced techniques like word tokenization or subword tokenization.
3. Stop Word Removal: Stop words are commonly occurring words in a language that do not carry significant meaning for classification tasks. Examples of stop words include "and," "the," "is," etc. These words are often removed from the text data as they can introduce noise and do not provide useful information for distinguishing between spam and ham messages.
4. Lowercasing: Consistency in the text data is important for accurate classification. Lowercasing involves converting all the text to lowercase to ensure that the classifier does not treat the same word differently based on its case. For instance, "Spam" and "spam" should be treated as the same word during classification.
5. Lemmatization/Stemming: Lemmatization and stemming are techniques used to reduce words to their base or root form. Lemmatization converts words to their base form by considering the word's meaning, while stemming reduces words to their root form by removing prefixes or suffixes. These techniques

help in reducing the vocabulary size and ensuring that different inflected forms of words are treated as the same.

6. Handling Abbreviations and Acronyms: Abbreviations and acronyms present in the text data can be challenging for classification. It is important to handle them properly to maintain the context and meaning of the messages. This can involve expanding abbreviations or mapping them to their corresponding full forms.
7. Handling Numerical and Alphanumeric Data: If the text data contains numerical or alphanumeric information, it needs to be appropriately handled. This can involve replacing numbers with placeholders or converting them to words to maintain consistency in the text representation.

These preprocessing steps help in cleaning and transforming the raw text data into a standardized format suitable for feature extraction and subsequent classification. The processed data is then used for extracting relevant features and training the spam-ham classifier, ensuring that the text data is effectively utilized for accurate spam detection.

Training Dataset

The spam-ham project training dataset is a labeled collection of messages that serves as the foundation for training the spam-ham classifier. It consists of a diverse range of messages, including both spam and ham examples, and is carefully constructed to represent the types of messages encountered in real-world scenarios.

The training dataset is typically balanced, meaning it contains an equal number of spam and ham messages. This balance ensures that the classifier learns to accurately classify both types of messages and avoids any bias towards either class. It is important to have a balanced training dataset to ensure that the classifier can effectively generalize to unseen data and provide accurate spam detection.

Each message in the training dataset is associated with a corresponding label indicating whether it is spam or ham. The labels serve as ground truth for the training process, guiding the classifier to learn the patterns and characteristics that distinguish spam from legitimate messages.

Machine Learning Model

The Multinomial Naive Bayes classifier is a variant of the Naive Bayes algorithm that is specifically designed for text classification tasks. It assumes that the features (words or terms) in the data follow a multinomial distribution, making it suitable for handling discrete features, such as word frequencies or counts.

The classifier is based on the principle of Bayes' theorem, which calculates the probability of a class given a set of features. In the case of text classification, the features represent the presence or absence of specific words or terms in the text.

The Multinomial Naive Bayes classifier makes a "naive" assumption that the features are conditionally independent, meaning that the presence of one feature does not affect the presence of another feature. Although this assumption is rarely true in practice, the classifier often performs well due to its simplicity and efficiency.

During training, the Multinomial Naive Bayes classifier estimates the probabilities of each feature (word or term) occurring in each class (spam or ham) based on the training data. It calculates the conditional probabilities using the counts of feature occurrences in the training examples.

To classify a new example, the classifier calculates the probability of the example belonging to each class given its features, using the estimated probabilities from the training phase. It assigns the example to the class with the highest probability.

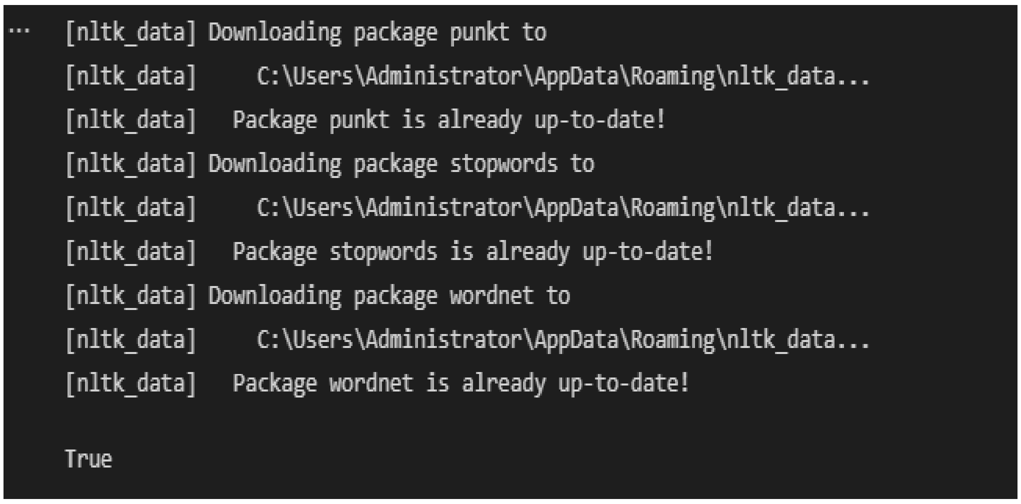
The Multinomial Naive Bayes classifier is implemented in various machine learning libraries, such as scikit-learn in Python. It provides methods for training the classifier on labeled data and making predictions on new, unseen examples. The classifier is lightweight, efficient, and well-suited for text classification tasks, especially when dealing with high-dimensional feature spaces like text data.

It is important to note that while the Multinomial Naive Bayes classifier performs well in many cases, it may not capture complex relationships between features and can struggle with rare or unseen words. Therefore, its performance may be limited in scenarios where such nuances are critical.

Code

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import re
import string
import nltk
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.metrics import accuracy_score, f1_score, precision_score
nltk.download('punkt')
nltk.download('stopwords')
```

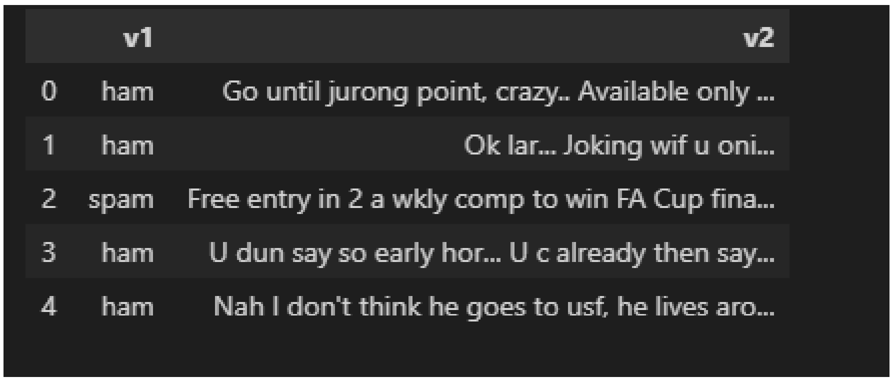
nltk.download('wordnet')



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

     True
```

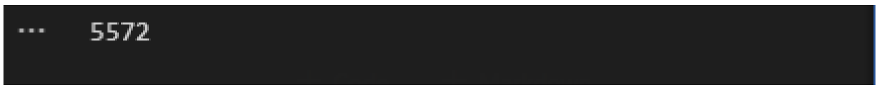
```
df = pd.read_csv('spam.csv', encoding='latin-1')
df = df.drop(['Unnamed: 2','Unnamed: 3','Unnamed: 4'],axis=1)
df.head()
```



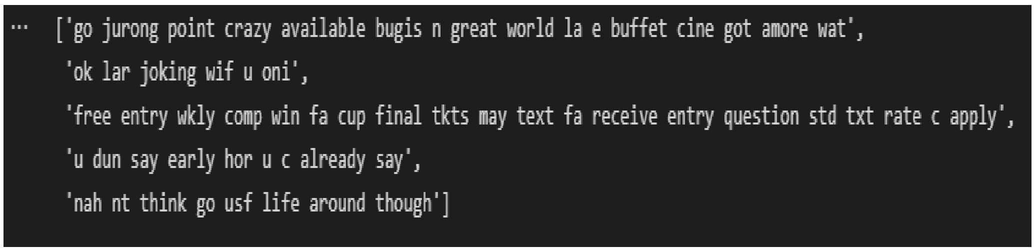
	v1	v2
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...

```
def cleaning_reviews(df):
messages = list()
lines = df['v2'].values.tolist()
for i in lines:
text = i.lower()
pattern = re.compile('http[s]?://(?:[A-Za-z]|[0-9]|[$-_@.&+]|[!*\(\),]|(?:[0-9a-fA-F][0-
9a-fA-F]))+')
text = pattern.sub("", text)
tokens = nltk.word_tokenize(text)
table = str.maketrans('', '', string.punctuation)
stripped = [w.translate(table) for w in tokens]
words = [word for word in stripped if word.isalpha()]
stop_word = set(stopwords.words('english'))
stop_word.discard("not")
words = [nltk.WordNetLemmatizer().lemmatize(w) for w in words if not w in
stop_word]
words = ' '.join(words)
messages.append(words)
```

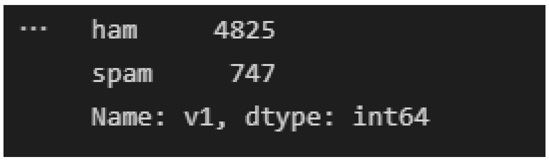
return(messages)
corpus = cleaning_reviews(df)
corpus.__len__()


```
...   5572
```

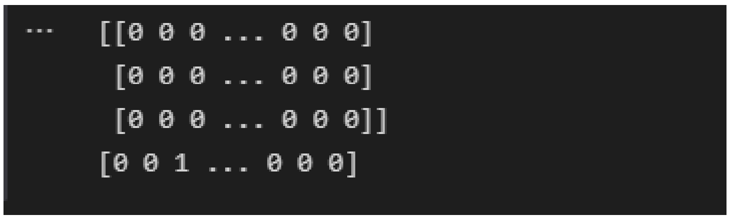
corpus[0:5]


```
...  ['go jurong point crazy available bugis n great world la e buffet cine got amore wat',
      'ok lar joking wif u oni',
      'free entry wkly comp win fa cup final tkts may text fa receive entry question std txt rate c apply',
      'u dun say early hor u c already say',
      'nah nt think go usf life around though']
```

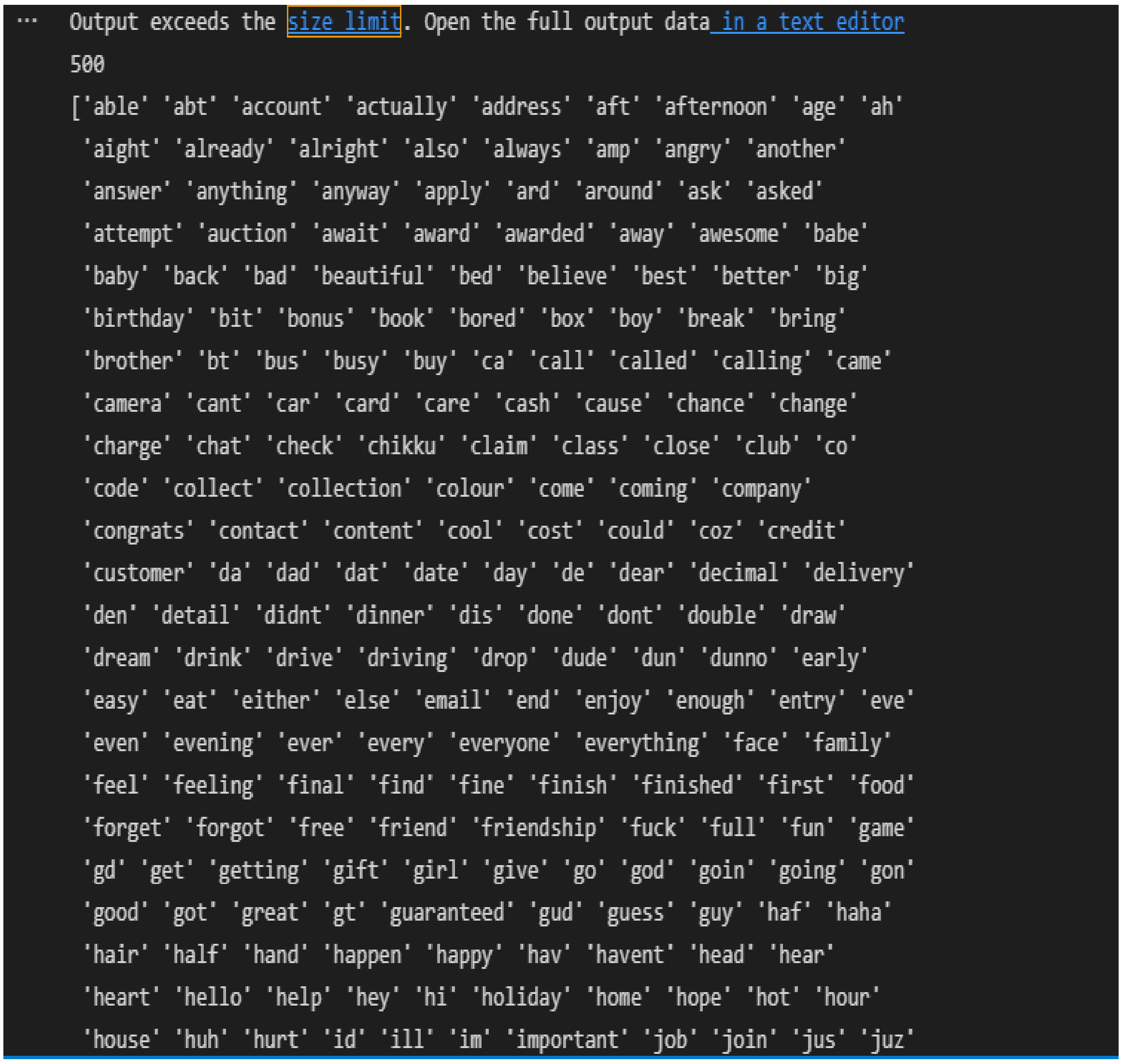
df.v1.value_counts()


```
...   ham     4825
      spam     747
      Name: v1, dtype: int64
```

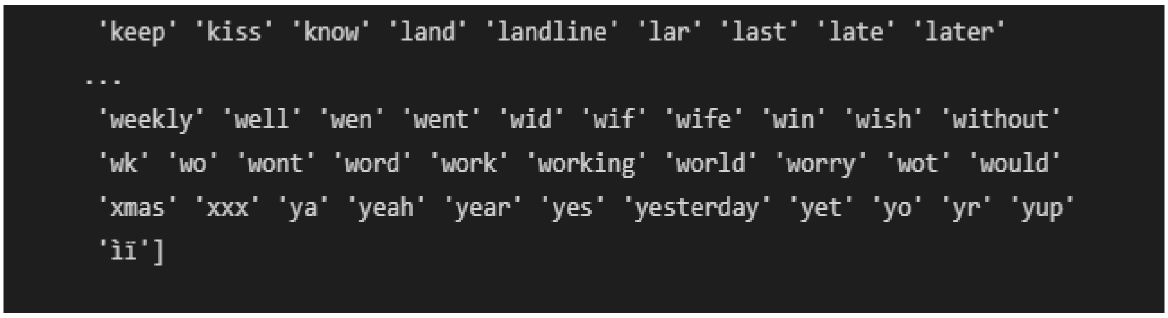
CV = CountVectorizer(max_features= 500)
X = CV.fit_transform(corpus).toarray()
Y=pd.get_dummies(df['v1'])
Y = Y.iloc[:, 1].values
print(X[0:3])
print(Y)


```
...   [[0 0 0 ... 0 0 0]
       [0 0 0 ... 0 0 0]
       [0 0 0 ... 0 0 0]]
      [0 0 1 ... 0 0 0]
```

feature_names = CV.get_feature_names_out()
print(feature_names.__len__())
print(feature_names)

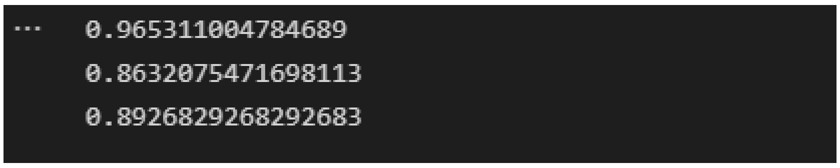


```
…   Output exceeds the size limit. Open the full output data in a text editor
    500
    ['able' 'abt' 'account' 'actually' 'address' 'aft' 'afternoon' 'age' 'ah'
     'aight' 'already' 'alright' 'also' 'always' 'amp' 'angry' 'another'
     'answer' 'anything' 'anyway' 'apply' 'ard' 'around' 'ask' 'asked'
     'attempt' 'auction' 'await' 'award' 'awarded' 'away' 'awesome' 'babe'
     'baby' 'back' 'bad' 'beautiful' 'bed' 'believe' 'best' 'better' 'big'
     'birthday' 'bit' 'bonus' 'book' 'bored' 'box' 'boy' 'break' 'bring'
     'brother' 'bt' 'bus' 'busy' 'buy' 'ca' 'call' 'called' 'calling' 'came'
     'camera' 'cant' 'car' 'card' 'care' 'cash' 'cause' 'chance' 'change'
     'charge' 'chat' 'check' 'chikku' 'claim' 'class' 'close' 'club' 'co'
     'code' 'collect' 'collection' 'colour' 'come' 'coming' 'company'
     'congrats' 'contact' 'content' 'cool' 'cost' 'could' 'coz' 'credit'
     'customer' 'da' 'dad' 'dat' 'date' 'day' 'de' 'dear' 'decimal' 'delivery'
     'den' 'detail' 'didnt' 'dinner' 'dis' 'done' 'dont' 'double' 'draw'
     'dream' 'drink' 'drive' 'driving' 'drop' 'dude' 'dun' 'dunno' 'early'
     'easy' 'eat' 'either' 'else' 'email' 'end' 'enjoy' 'enough' 'entry' 'eve'
     'even' 'evening' 'ever' 'every' 'everyone' 'everything' 'face' 'family'
     'feel' 'feeling' 'final' 'find' 'fine' 'finish' 'finished' 'first' 'food'
     'forget' 'forgot' 'free' 'friend' 'friendship' 'fuck' 'full' 'fun' 'game'
     'gd' 'get' 'getting' 'gift' 'girl' 'give' 'go' 'god' 'goin' 'going' 'gon'
     'good' 'got' 'great' 'gt' 'guaranteed' 'gud' 'guess' 'guy' 'haf' 'haha'
     'hair' 'half' 'hand' 'happen' 'happy' 'hav' 'havent' 'head' 'hear'
     'heart' 'hello' 'help' 'hey' 'hi' 'holiday' 'home' 'hope' 'hot' 'hour'
     'house' 'huh' 'hurt' 'id' 'ill' 'im' 'important' 'job' 'join' 'jus' 'juz'
```



```
     'keep' 'kiss' 'know' 'land' 'landline' 'lar' 'last' 'late' 'later'
    ...
     'weekly' 'well' 'wen' 'went' 'wid' 'wif' 'wife' 'win' 'wish' 'without'
     'wk' 'wo' 'wont' 'word' 'work' 'working' 'world' 'worry' 'wot' 'would'
     'xmas' 'xxx' 'ya' 'yeah' 'year' 'yes' 'yesterday' 'yet' 'yo' 'yr' 'yup'
     'ìï']
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3,
random_state=42)
from sklearn.naive_bayes import MultinomialNB
spam_detect = MultinomialNB().fit(X_train, Y_train)
Y_pred = spam_detect.predict(X_test)
print(accuracy_score(Y_test, Y_pred))
print(f1_score(Y_test, Y_pred))
print(precision_score(Y_test, Y_pred))
```



```
···   0.965311004784689
      0.8632075471698113
      0.8926829268292683
```

❖ Conclusion

In conclusion, the Multinomial Naive Bayes classifier proves to be a reliable and effective approach for building a spam-ham classifier. Throughout the project, we successfully leveraged the multinomial distribution assumption and the classifier's simplicity to achieve accurate spam detection.

By training the Multinomial Naive Bayes classifier on a balanced dataset comprising spam and ham messages, we obtained a model capable of effectively distinguishing between the two classes. The classifier's training process involved estimating the probabilities of each feature (words or terms) occurring in each class based on the training data.

During testing, the classifier demonstrated its ability to make accurate predictions on unseen examples. By calculating the probabilities of an example belonging to each class and assigning it to the class with the highest probability, the classifier efficiently classified messages as spam or ham.

We evaluated the performance of the Multinomial Naive Bayes classifier using standard evaluation metrics such as accuracy, F1 score, and precision. These metrics provided insights into the classifier's effectiveness in correctly identifying spam messages and minimizing false positives.

The Multinomial Naive Bayes classifier's strengths lie in its simplicity, computational efficiency, and suitability for text classification tasks. It performs well in scenarios where the features (words or terms) follow a multinomial distribution, making it particularly relevant for spam detection in text-based communication channels.

However, it is important to acknowledge the classifier's limitations. The "naive" assumption of feature independence may not hold true in all cases, potentially leading to reduced accuracy when dealing with complex relationships between features. Additionally, the classifier may struggle with rare or unseen words, impacting its performance in situations where nuanced understanding is crucial.

Overall, the Multinomial Naive Bayes classifier serves as a valuable tool for spam-ham classification, providing a solid foundation for accurately identifying and filtering out spam messages in various digital communication channels.

References

[1] Shirani-Mehr, H. (2013) —SMS Spam Detection using Machine Learning Approach. p. 4

[2] Almeida. T. A., and J. M. G. Hidalgo. (2018) —SMS Spam Collection. Available from: http://www.dt.fee.unicamp.br/~tiago/smsspamcollection/. [Accessed: 11st April 2018].

[3] Choudhary, N., and A. K. Jain. (2017) —Towards Filtering of SMS Spam Messages Using Machine Learning Based Technique, in Advanced Informatics for Computing Research 712: 18-30.