A MINOR PROJECT REPORT

ON

“CORONA TWEET CLASSIFICATION SYSTEM”

Submitted in partial fulfilment of the

requirement for the award of the degree

of

MASTER OF COMPUTER APPLICATIONS

by

Ahmad Faraz Ansari

(Roll No: 2023073006)

Under the Guidance of

Ms. Pranjal Maurya

Assistant Professor, ITCA Dept.

DEPARTMENT OF INFORMATION TECHNOLOGY AND COMPUTER APPLICATION

MADAN MOHAN MALAVIYA UNIVERSITY OF TECHNOLOGY

GORAKHPUR – 273010

SESSION – 2024-25

Declaration

I, **Ahmad Faraz Ansari,** have completed the project titled "**Corona Tweet Classification System**" under the guidance of **Ms. Pranjal Maurya** in the partial fulfilment of the requirement for the award of Degree of Master of Computer Applications of Madan Mohan Malaviya University of Technology, Gorakhpur. This is an original piece of work, and I have neither copied nor submitted it earlier elsewhere.

Ahmad Faraz Ansari
Roll No: 2023073006
Master of Computer Applications
Madan Mohan Malaviya University of Technology
Gorakhpur

Certificate

This is to certify that the project titled “**Corona Tweet Classification System**” is an academic work done by “**Ahmad Faraz Ansari**” submitted in the partial fulfilment of the requirement for the award of the Degree of “**Master of Computer Applications**” from “**Madan Mohan Malaviya University of Technology, Gorakhpur**” under my guidance & direction. To the best of my knowledge and belief the data & information presented by him in the project has not been submitted earlier.

Ms. Pranjal Maurya

(Project Supervisor)

Acknowledgement

I am extremely grateful and remain indebted to my guide **Ms. Pranjal Maurya** for being a source of inspiration and for their constant support in the Design, Implementation and Evaluation of the project. I am thankful for constant constructive criticism and invaluable suggestions, which benefited me a lot while developing the project on **"Corona Tweet Classification System"**. With candor and pleasure, I take the opportunity to express my sincere thanks and obligation to the faculty members. Through this column, it would be my utmost pleasure to express my warm thanks to them for the encouragement, co-operation and consent without which I might not be able to accomplish this project.

Ahmad Faraz Ansari

(M.C.A. 3rd Semester)

Abstract

Corona Tweet Classification System is a tweet classification system based on Machine-learning. It takes user input as a tweet and then predicts the sentiment of that tweet i.e., whether the tweet is covid-positive or covid-negative. After the prediction, the system displays the predicted sentiment i.e., covid-positive or covid-negative.

This report reflects the idea of taking user's input into consideration and performing classification and establishing conclusion on interested topics using Machine – Learning algorithm. Support Vector Machines in Machine – Learning is tuned up using supervised learning to obtain outputs for classification.

Keywords: Classification, sentiment, Support vector machines, supervised learning.

Table of Contents

1. Introduction

1.1 Overview

The **Corona Tweet Classification System**, also referred to as the **Covid Tweet Classification System**, is an advanced machine learning-based system designed to automatically classify tweets related to COVID-19. The system utilizes the power of the **Support Vector Machine (SVM)** algorithm to categorize tweets according to the emotion they convey, particularly focusing on the sentiment regarding COVID-19. This project is implemented using **Python 3**, ensuring platform independence and compatibility across different operating systems.

The proposed system is built with the ability to classify tweets based on the **severity of the underlying emotion**. The primary categories include **COVID-positive** tweets, which represent tweets with a positive sentiment towards COVID-19, and **COVID-negative** tweets, which indicate negative or cautious sentiments. By analyzing the text and context of tweets, the system determines whether the sentiment is positive, negative, or neutral with respect to the pandemic.

1.2 Objective

The primary objectives for the **Corona Tweet Classification System** are as follows:

- **Develop a Tweet Classification System**: The system should be capable of accurately classifying tweets related to COVID-19, identifying key sentiments such as positivity, negativity, and neutrality.
- **Ensure High Accuracy and Precision**: The system must achieve a high level of accuracy and precision, ensuring that the classifications are reliable and meaningful for further analysis.
- **Emotion Severity Classification**: The classification of tweets should be based on the severity of the emotion conveyed, such as whether the sentiment is strongly negative, mildly negative, neutral, or positive. This enables a deeper understanding of how individuals perceive the pandemic at different emotional levels.

Through the implementation of machine learning algorithms and sentiment analysis, this system offers a powerful approach for real-time classification and sentiment tracking, which can be instrumental in shaping responses to public sentiment surrounding the COVID-19 pandemic.

2. Technology Description

The system has been designed to be efficient and scalable, taking full advantage of modern hardware capabilities while ensuring accessibility across a wide range of devices. This design ensures that the system performs optimally without overburdening the resources of the host machine.

Hardware Requirements:

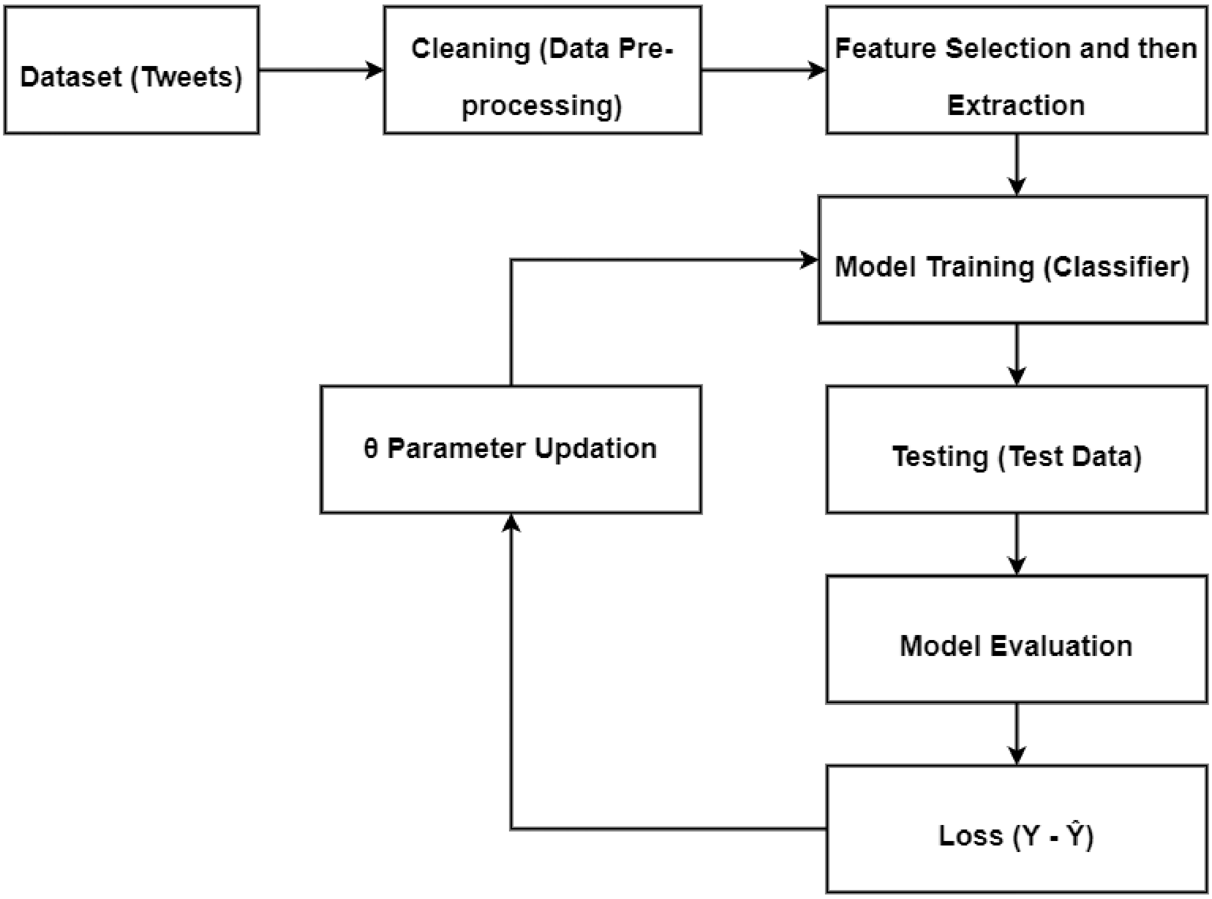
- **RAM:** At least 4GB of RAM is required for basic operations. However, for better performance, especially when handling large datasets or performing model training, 8GB or more of RAM is recommended.
- **Processor:** The system requires at least an Intel Core i3 7th Generation processor or newer. For enhanced performance and faster computation, a higher-end processor like an Intel i5 or i7 or AMD Ryzen 5 or 7 is recommended.
- **GPU:** No dedicated GPU is necessary for the execution of this project, as the project's core functionality is based on CPU-bound computations. However, having a GPU may speed up certain machine learning tasks, particularly in deep learning scenarios, though it is not essential for this project.

Software Requirements:

- **Language Compiler or Interpreter:** Python 3 is the primary programming language used in this project. The Python 3 interpreter must be pre-installed on the system. A recent version (3.6 or later) is recommended to ensure compatibility with libraries and features.
- **Code Editor or IDE:** To write, test, and execute the Python code, you will need a Python-compatible code editor or IDE. PyCharm and Jupyter Notebook are recommended for their robust features and seamless integration with Python. While PyCharm offers advanced features for large-scale Python development, Jupyter Notebook is ideal for interactive coding and data science tasks.
- **Operating System:** The system can run on any 64-bit desktop operating system. Microsoft Windows 10 or later is recommended for compatibility with Python libraries and machine learning frameworks. Linux or macOS can also be used but may require specific adjustments based on the environment.

3. System Design

3.1 Machine Learning Pipeline Diagram


Dataset (Tweets)
Cleaning (Data Pre-processing)
Feature Selection and then Extraction
Model Training (Classifier)
θ Parameter Updation
Testing (Test Data)
Model Evaluation
Loss (Y - Ŷ)

4. Modules

Classification is a supervised learning technique used for categorizing a given set of data into predefined classes. It is an integral part of machine learning and finds applications in numerous domains such as sentiment analysis, spam detection, and fraud detection. For this project, the classification of tweets into specific categories was undertaken through a series of carefully designed modules.

The following modules form the backbone of this project:

1. Importing
2. Processing
3. Feature Extraction
4. Training
5. Testing
6. Evaluation Metrics

4.1 Importing:

Importing is the first and foundational step in any project. It involves loading all the necessary libraries and datasets required to carry out the tasks effectively. By using modular libraries, complex processes are simplified into reusable functions. Importing ensures that tools and datasets required for tasks like data manipulation, cleaning, model building, and evaluation are readily available. It minimizes redundancy and allows focus on core logic.

The following libraries were utilized:

1. pandas – this package is necessary for working on the dataset.
2. numpy – this package is necessary for array conversion.
3. sklearn – this package is responsible for classification of tweets, splitting of dataset and calculation of metric scores.
4. re – this package is used for regular expression operations.
5. string – this package is used for string related operations.
6. nltk – this package is responsible for removing stop-words, Lemmatization of words, Tokenization of words and calculating the polarity scores.

4.2 Processing:

Processing is the second step towards classification of tweets. We require our dataset to be clean i.e., free from all unnecessary things that have no use inour model. All unnecessary columns are dropped from the dataset to save time and resources.

After that, each tweet is cleaned using regular expressions (removing symbols, stickers and hyperlinks) and then tweets are tokenized, and all punctuation and stop-words are removed.

Data processing transforms raw, unstructured data into a clean and structured format suitable for analysis and model training. This step is crucial as the quality of input data directly impacts the performance of the machine learning mode

4.3 Feature Extraction:

We are extracting two features from the tweets that are useful for model training. First, the length of the tweet and second, the polarity score (amount of positivity & negativity of the tweet).

4.4 Training:

Training models means feeding the machine learning algorithm with sufficient training datato learn from. It is very important to provide the best data to the ML algorithm to get the most accurate prediction in return.

ML Algorithm used for this project is called Support Vector Machines (SVM). This is a supervised learning algorithm primarily used for classification.

4.5 Testing:

Testing of model means testing the model that has been trained using unknown data i.e., testing data. The predictions made by the models are then compared in further steps to check model accuracy. If the predictions made by the model are not satisfactory or accurate then the necessary measures must be taken to increase model accuracy.

4.6 Evaluation Metrics:

Evaluation is always good in any field. In the case of machine learning, it is the best practice. There are many metrics used for evaluating machine learning models. In this project, F1 Score and Jaccard Index are used for very specific reasons.

F1 Score: It is used to test the accuracy of model. Its range is [0, 1].

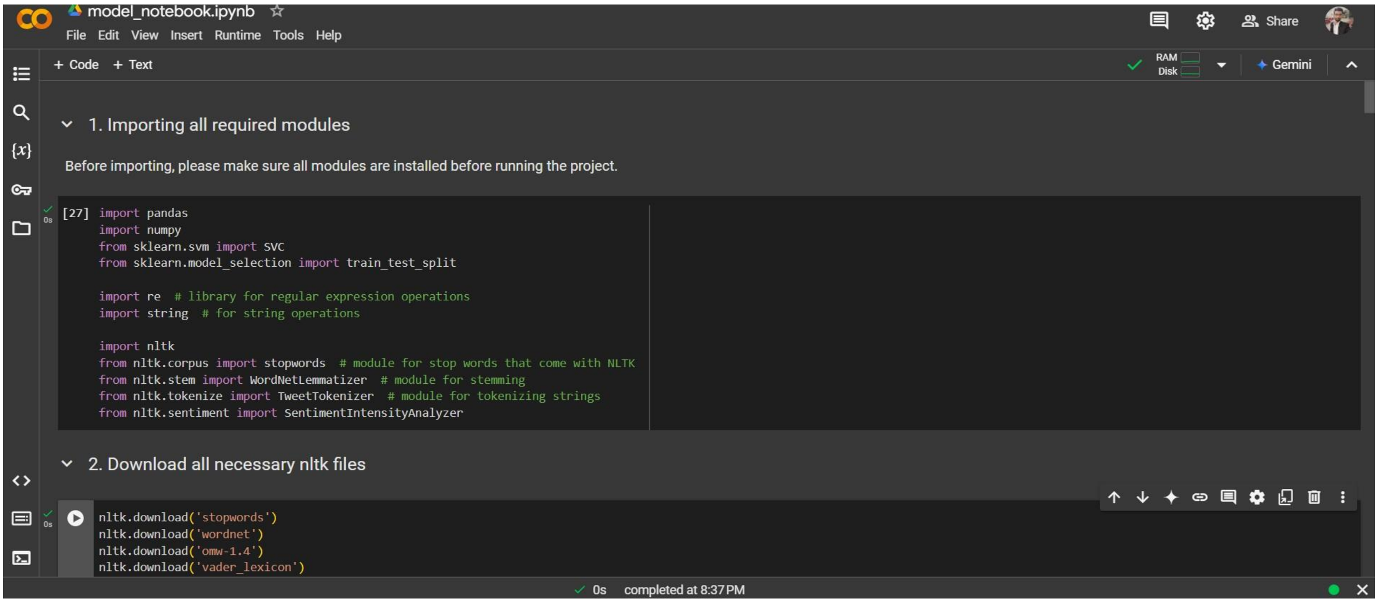
F1 Score = $\mathbf{2} * \frac{(\boldsymbol{Precision} * \boldsymbol{Recall})}{\boldsymbol{Precision} + \boldsymbol{Recall}}$

Jaccard Index: it is used in understanding the similarity between sample sets. The measurement emphasizes similarity between finite sample sets. Its range is [0, 1].

Jaccard Index, J (A, B) = $|A \cap B| / |A \cup B|$

5. Project Screenshot

5.1 Importing packages



1. Importing all required modules

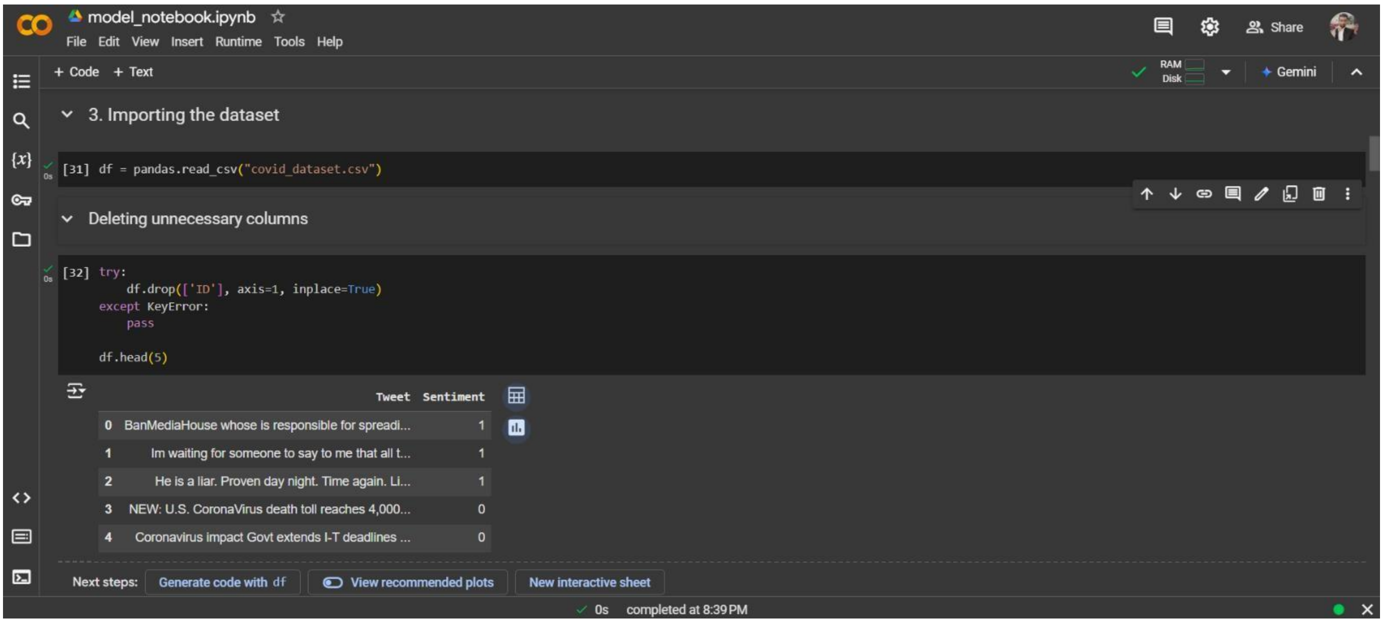
Before importing, please make sure all modules are installed before running the project.

```
[27] import pandas
     import numpy
     from sklearn.svm import SVC
     from sklearn.model_selection import train_test_split

     import re  # library for regular expression operations
     import string  # for string operations

     import nltk
     from nltk.corpus import stopwords  # module for stop words that come with NLTK
     from nltk.stem import WordNetLemmatizer  # module for stemming
     from nltk.tokenize import TweetTokenizer  # module for tokenizing strings
     from nltk.sentiment import SentimentIntensityAnalyzer
```

2. Download all necessary nltk files

```
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('omw-1.4')
nltk.download('vader_lexicon')
```

5.2 Importing dataset



3. Importing the dataset

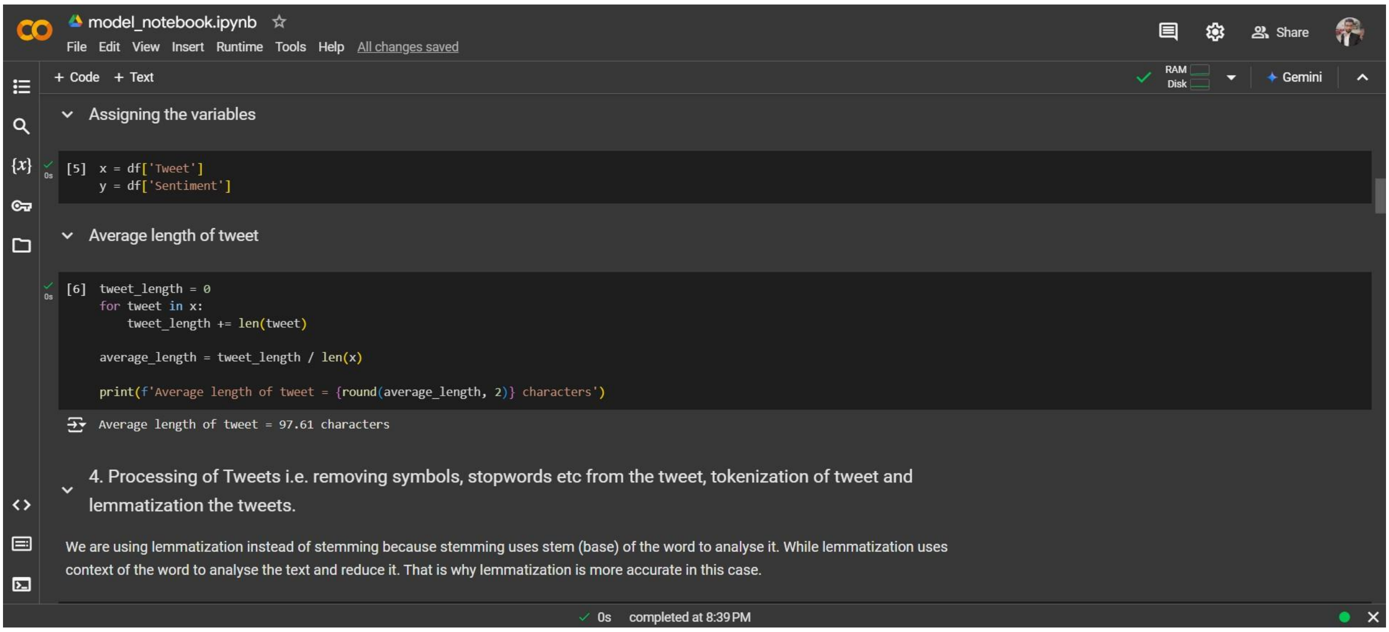
```
[31] df = pandas.read_csv("covid_dataset.csv")
```

Deleting unnecessary columns

```
[32] try:
         df.drop(['ID'], axis=1, inplace=True)
     except KeyError:
         pass

     df.head(5)
```

	Tweet	Sentiment
0	BanMediaHouse whose is responsible for spreadi...	1
1	Im waiting for someone to say to me that all t...	1
2	He is a liar. Proven day night. Time again. Li...	1
3	NEW: U.S. CoronaVirus death toll reaches 4,000...	0
4	Coronavirus impact Govt extends I-T deadlines ...	0

5.3 Variable assignments



Assigning the variables

```
[5] x = df['Tweet']
    y = df['Sentiment']
```

Average length of tweet

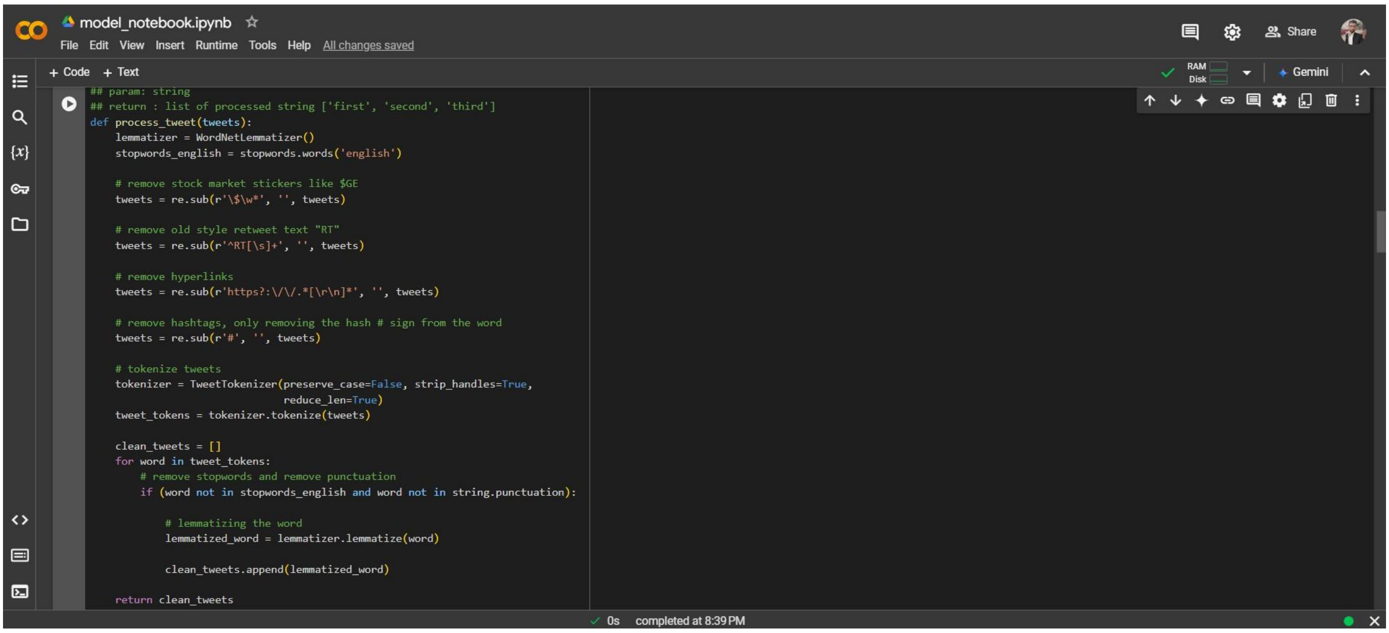
```
[6] tweet_length = 0
    for tweet in x:
        tweet_length += len(tweet)

    average_length = tweet_length / len(x)

    print(f'Average length of tweet = {round(average_length, 2)} characters')
```

```
Average length of tweet = 97.61 characters
```

4. Processing of Tweets i.e. removing symbols, stopwords etc from the tweet, tokenization of tweet and lemmatization the tweets.

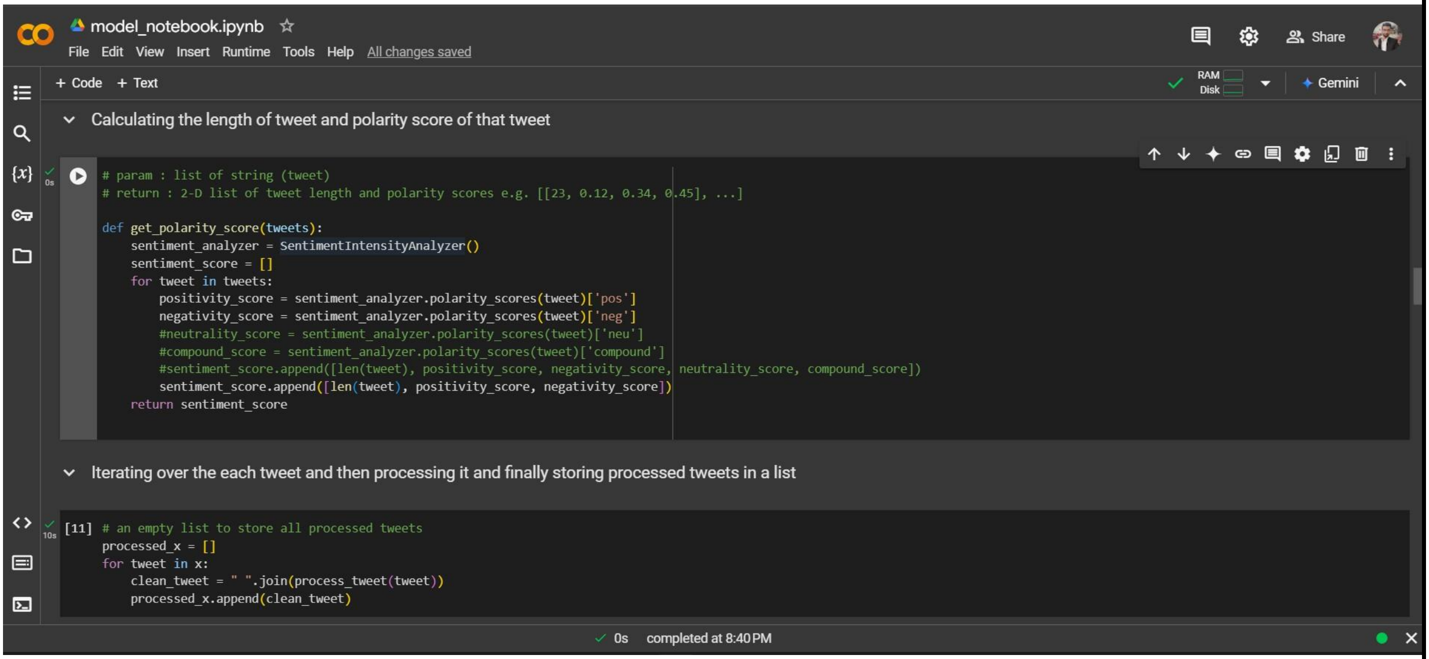
We are using lemmatization instead of stemming because stemming uses stem (base) of the word to analyse it. While lemmatization uses context of the word to analyse the text and reduce it. That is why lemmatization is more accurate in this case.

5.4 Processing



```
## param: string
## return : list of processed string ['first', 'second', 'third']
def process_tweet(tweets):
    lemmatizer = WordNetLemmatizer()
    stopwords_english = stopwords.words('english')

    # remove stock market stickers like $GE
    tweets = re.sub(r'\$\w*', '', tweets)

    # remove old style retweet text "RT"
    tweets = re.sub(r'^RT[\s]+', '', tweets)

    # remove hyperlinks
    tweets = re.sub(r'https?:\/\/.*[\r\n]*', '', tweets)

    # remove hashtags, only removing the hash # sign from the word
    tweets = re.sub(r'#', '', tweets)

    # tokenize tweets
    tokenizer = TweetTokenizer(preserve_case=False, strip_handles=True,
                               reduce_len=True)
    tweet_tokens = tokenizer.tokenize(tweets)

    clean_tweets = []
    for word in tweet_tokens:
        # remove stopwords and remove punctuation
        if (word not in stopwords_english and word not in string.punctuation):

            # lemmatizing the word
            lemmatized_word = lemmatizer.lemmatize(word)

            clean_tweets.append(lemmatized_word)

    return clean_tweets
```

5.5 Polarity score calculation



Calculating the length of tweet and polarity score of that tweet

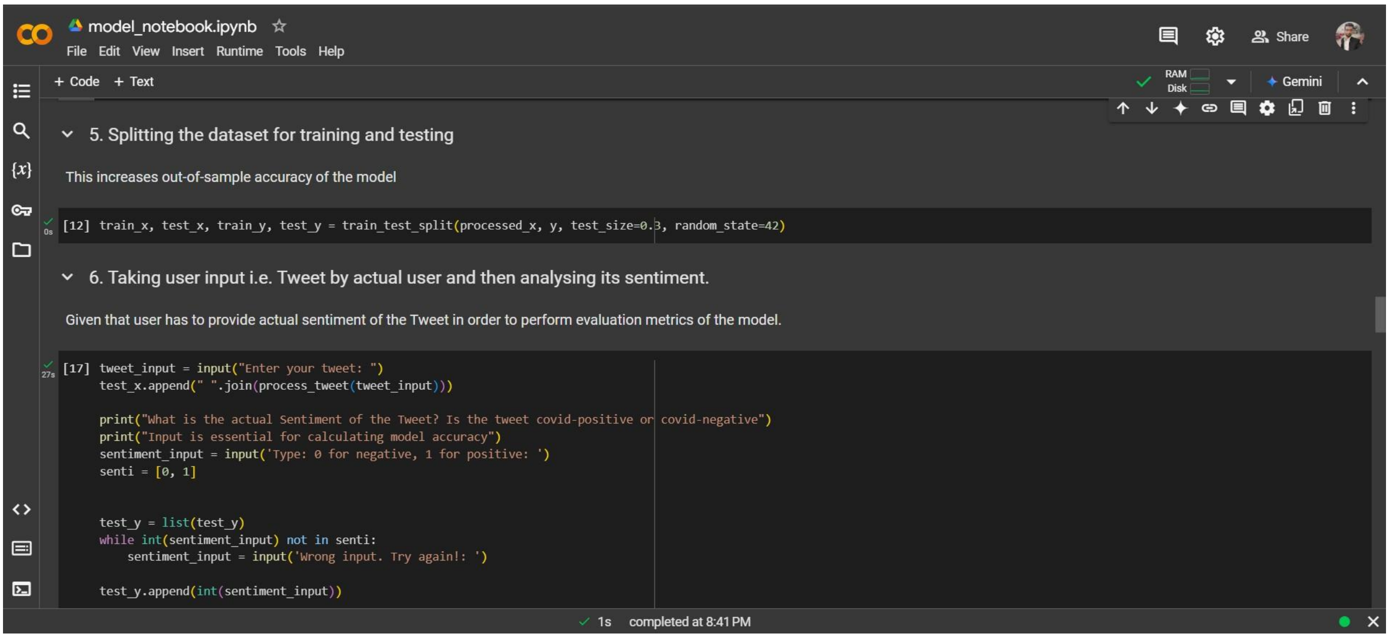
```
# param : list of string (tweet)
# return : 2-D list of tweet length and polarity scores e.g. [[23, 0.12, 0.34, 0.45], ...]

def get_polarity_score(tweets):
    sentiment_analyzer = SentimentIntensityAnalyzer()
    sentiment_score = []
    for tweet in tweets:
        positivity_score = sentiment_analyzer.polarity_scores(tweet)['pos']
        negativity_score = sentiment_analyzer.polarity_scores(tweet)['neg']
        #neutrality_score = sentiment_analyzer.polarity_scores(tweet)['neu']
        #compound_score = sentiment_analyzer.polarity_scores(tweet)['compound']
        #sentiment_score.append([len(tweet), positivity_score, negativity_score, neutrality_score, compound_score])
        sentiment_score.append([len(tweet), positivity_score, negativity_score])
    return sentiment_score
```

Iterating over the each tweet and then processing it and finally storing processed tweets in a list

```
[11] # an empty list to store all processed tweets
     processed_x = []
     for tweet in x:
         clean_tweet = " ".join(process_tweet(tweet))
         processed_x.append(clean_tweet)
```

✓ 0s completed at 8:40 PM

5.6 Dataset split and user input



5. Splitting the dataset for training and testing

This increases out-of-sample accuracy of the model

```
[12] train_x, test_x, train_y, test_y = train_test_split(processed_x, y, test_size=0.3, random_state=42)
```

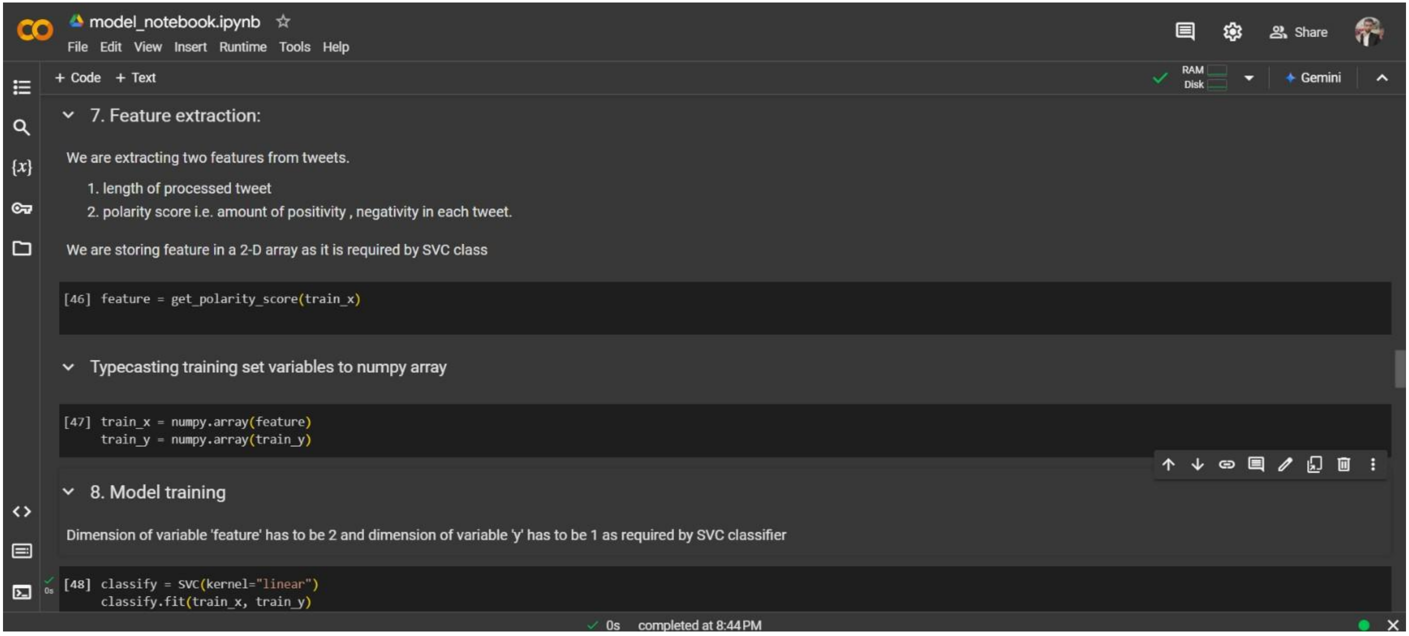
6. Taking user input i.e. Tweet by actual user and then analysing its sentiment.

Given that user has to provide actual sentiment of the Tweet in order to perform evaluation metrics of the model.

```
[17] tweet_input = input("Enter your tweet: ")
     test_x.append(" ".join(process_tweet(tweet_input)))

     print("What is the actual Sentiment of the Tweet? Is the tweet covid-positive or covid-negative")
     print("Input is essential for calculating model accuracy")
     sentiment_input = input('Type: 0 for negative, 1 for positive: ')
     senti = [0, 1]


     test_y = list(test_y)
     while int(sentiment_input) not in senti:
         sentiment_input = input('Wrong input. Try again!: ')

     test_y.append(int(sentiment_input))
```

✓ 1s completed at 8:41 PM

5.7 Feature Extraction and model training



7. Feature extraction:

We are extracting two features from tweets.

1. length of processed tweet
2. polarity score i.e. amount of positivity , negativity in each tweet.

We are storing feature in a 2-D array as it is required by SVC class

```
[46] feature = get_polarity_score(train_x)
```

Typecasting training set variables to numpy array

```
[47] train_x = numpy.array(feature)
     train_y = numpy.array(train_y)
```

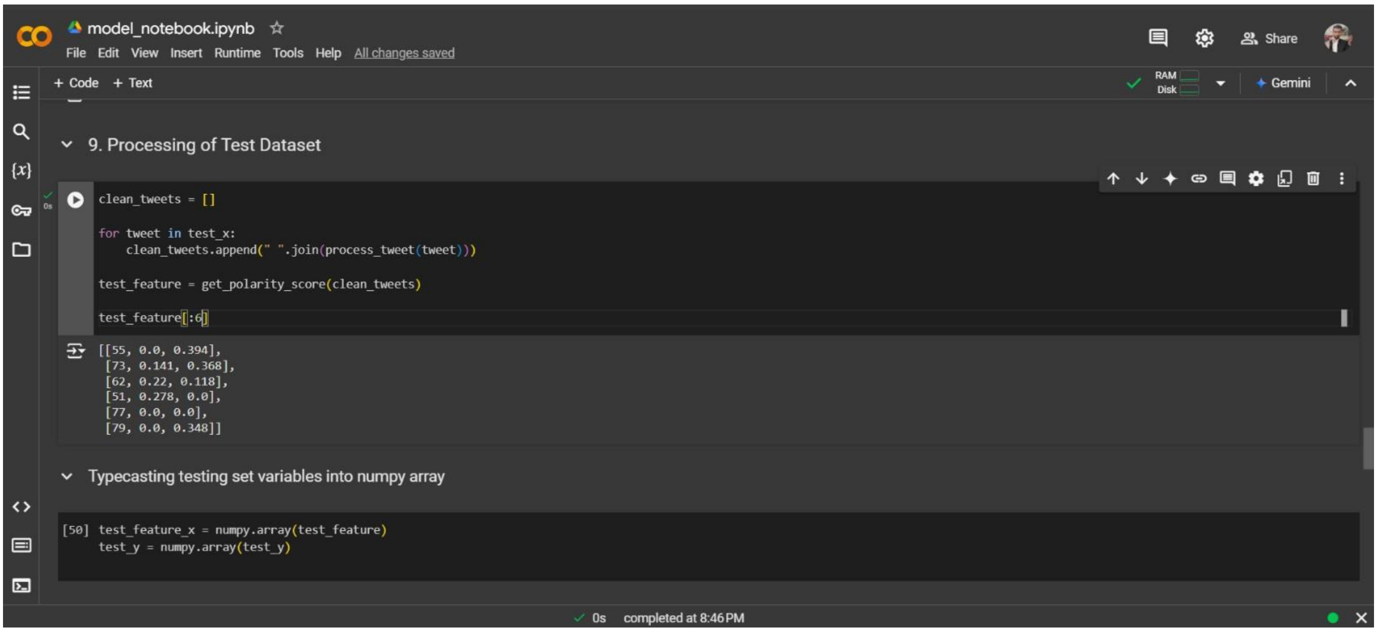
8. Model training

Dimension of variable 'feature' has to be 2 and dimension of variable 'y' has to be 1 as required by SVC classifier

```
[48] classify = SVC(kernel="linear")
     classify.fit(train_x, train_y)
```

✓ 0s completed at 8:44 PM

5.8 Processing of test dataset



9. Processing of Test Dataset

```
clean_tweets = []

for tweet in test_x:
    clean_tweets.append(" ".join(process_tweet(tweet)))

test_feature = get_polarity_score(clean_tweets)

test_feature[:6]
```

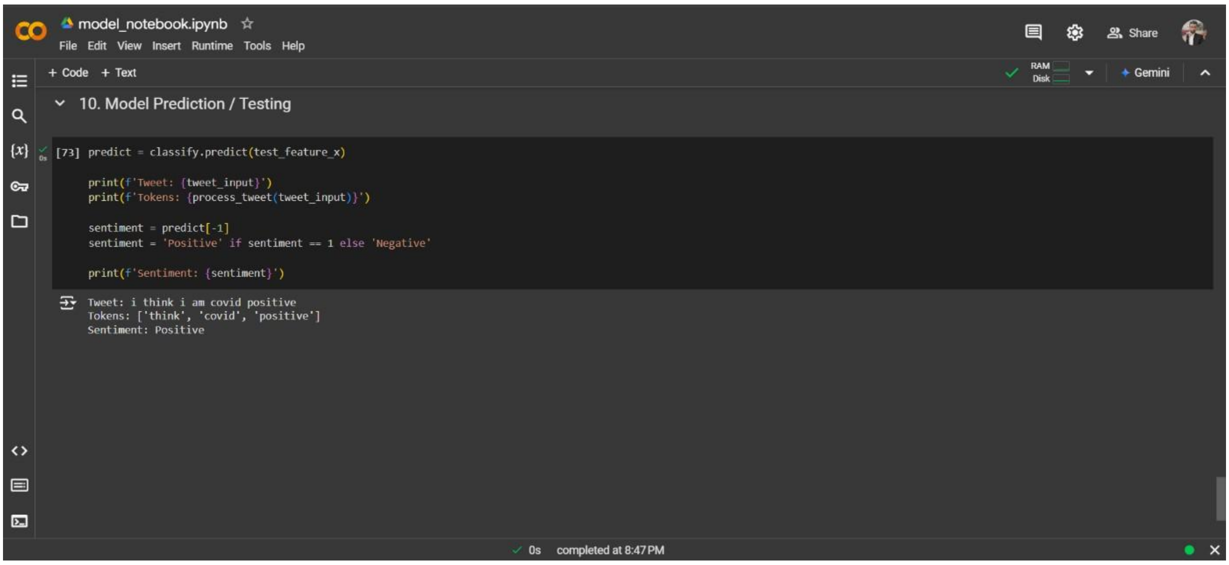
```
[[55, 0.0, 0.394],
 [73, 0.141, 0.368],
 [62, 0.22, 0.118],
 [51, 0.278, 0.0],
 [77, 0.0, 0.0],
 [79, 0.0, 0.348]]
```

Typecasting testing set variables into numpy array

```
[50] test_feature_x = numpy.array(test_feature)
     test_y = numpy.array(test_y)
```

✓ 0s completed at 8:46 PM

5.9 Model prediction



10. Model Prediction / Testing

```
[73] predict = classify.predict(test_feature_x)

     print(f'Tweet: {tweet_input}')
     print(f'Tokens: {process_tweet(tweet_input)}')

     sentiment = predict[-1]
     sentiment = 'Positive' if sentiment == 1 else 'Negative'

     print(f'Sentiment: {sentiment}')
```

```
Tweet: i think i am covid positive
Tokens: ['think', 'covid', 'positive']
Sentiment: Positive
```

6. Future Scope

In the dynamic field of machine learning and natural language processing, projects can evolve continuously to meet changing demands and incorporate new technologies. While the current project demonstrates robust functionality and achieves its primary objectives, there are several potential enhancements that could make it more dynamic, user-friendly, and impactful over time.

Potential Enhancements:

- Transitioning the project from an interactive Jupyter Notebook (.ipynb) to an executable file (.exe) would eliminate the need for end-users to have Python or Jupyter Notebook installed on their systems.
- Creating a user-friendly interface would make the project more interactive and visually appealing. Implementing concepts of **UI/UX design**, such as proper use of color theory, typography, and layout design, can significantly enhance the user experience.
- Implementing advanced optimization techniques to update model parameters after each loss during training can enhance accuracy and thereby make the model more accurate.
- Expanding the project's capabilities to handle tweets in multiple languages by leveraging libraries like **Google Translate API** can widen its applicability.

The future scope of this project is vast, with ample opportunities to enhance its functionality, scalability, and user experience. By integrating advanced technologies, improving usability, and ensuring efficient maintainability, the project can evolve into a comprehensive and impactful solution for real-world applications. Continuous updates and innovation will keep the project relevant and adaptable to changing needs.

Challenges can be overcome by refactoring the Project from time to time to increase Code Maintainability engaging platform, offering comprehensive experience for both book enthusiasts and potential buyers. These additions not only enhance the platform's functionality but also contribute to its sustainability and competitiveness in the online bookstore market.

7. Conclusion

The primary objective of this project was to develop a robust and efficient **Tweet Classification System** capable of delivering high accuracy while maintaining resource efficiency and enhanced user experience (UX). In today's data-driven era, where information from social media platforms like Twitter plays a significant role in shaping opinions and decisions, the need for a reliable classification system cannot be overstated. This project aims to address that need by building a streamlined solution for classifying tweets effectively.

To achieve this, extensive research was undertaken to explore the existing systems and methodologies employed in tweet classification. A comprehensive review of literature and market products provided insights into the strengths and weaknesses of existing systems. This exploration uncovered inefficiencies in handling diverse datasets, limited scalability, and suboptimal accuracy, especially in resource-constrained environments. These findings formed the foundation of this project, driving the design and development of a system that addresses these limitations. The incorporation of modern machine learning algorithms and preprocessing techniques ensured that the project not only achieves its objectives but also sets a benchmark for resource optimization and precision in tweet classification.

Throughout this journey, extensive literature reviews were conducted to understand existing systems, their strengths, and their limitations. These reviews provided valuable insights into the state of the art and exposed several challenges faced by current systems, including inefficiencies in processing, suboptimal accuracy, and user accessibility issues. By addressing these gaps, this project has been tailored to overcome such limitations, offering a solution that is both practical and impactful.

8. References

1. Géron, A. **Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow**. O'Reilly Media, 2019.
2. Ng, A. **Machine Learning Yearning**. DeepLearning.ai, 2018.
3. Deitel, P., & Deitel, H. **Intro to Python for Computer Science and Data Science**. Pearson, 2020.
4. Kaggle.com: Datasets and Resources for Machine Learning and Natural Language Processing.