Software Requirements Specification

for

Machine Learning Based Spam Detection

Version 2.0 approved

Prepared by
Benny Berba, Syam Jason Bonela,
Leonard Garcia, Saiyang Liu,
Mary Semerdjian

California State University, Los Angeles

September 12, 2022

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Benny Berba	09/12/22	Initial draft, added to 1.1-1.5	1.0
Mary Semerdjian	12/11/22	Final draft, revised 1.1-Appendix	2.0

1. Introduction

1.1 Purpose

The purpose of this document is to explain the functions of Machine Learning Spam Detection software (Version 2.0). This document will cover all aspects of the software, which include: an overall description of the ML Spam Detection software, the external interface requirements, other nonfunctional requirements, and the legal and ethical considerations.

1.2 Intended Audience and Reading Suggestions

The intended audience of this document includes developers, machine learning researchers, users, computer science students, marketing staff, and testers. This SRD is organized as follows: introduction, overall description, external interface requirements, requirements specification, other nonfunctional requirements, and legal and ethical considerations. Developers and ML researchers may be interested in reading the entirety of the document and following each section in sequential order. Marketing staff might be interested in the high-level descriptions of the project, such as the Introduction and Overall Description.

1.3 Product Scope

Machine Learning-Based Spam Detection (Version 2.0) is the product name.

The software functions-software will take emails and text messages as input. It will classify the input using different classification ML models and output whether the emails/text messages are spam or ham.

The benefits of the software are improving spam detection accuracy, precision, and recall will increase user experience and satisfaction.

The object is to improve spam detection metrics: accuracy, precision, and recall.

Lastly, the goal is to train and test ML models that achieve at least >80% accuracy, precision, and recall and improve spam detection metrics: accuracy, precision, and recall.

1.4 Definitions, Acronyms, and Abbreviations

Refer to Appendix: A

1.5 References

Refer to Appendix: C

2. Overall Description

2.1 System Analysis

Machine Learning Based Spam Detection (Version 2.0) is a software designed and intended to improve the ability to detect spam using Machine Learning models. This program aims to Achieve high accuracy, recall, and precision (e.g.,> 90%). And this product can be used in a professional environment as it will help data scientists in Spam-Based detection.
Project Goals:

- Create a simple and methodical straightforward approach to using Graphical User Interface (GUI)
- Create a bar graph, and graph of a mathematical model where the model can specify the Machine Learning classifiers

Analysis of Technical Hurdles:

- Higher ethernet speed and Wi-Fi for testing data
- Need a powerful PC to test large dataset

Solutions to Overcome Technical Hurdles:

- Finish testing the GUI with a Machine Learning classifier/s as an output
- Conversing with the group to discuss a more methodological approach on how to succeed the Spam Texts/Emails and improve accuracy if possible
- Learning and understanding the Machine Learning classifiers as well as the algorithms
- Dedicate and put time into group-work-project
- Discuss work-documentation-slides each week

2.2 Product Perspective

Google has a colaboratory or "colab" which is a product from Google Research. Colab is a Jupyter Notebook-like product. It allows you to combine executable code and rich text in a single document alongside images, etc. Google Colab was used to test-train the data.

Jupyter Notebook is a web application for creating and sharing computational documents. Jupyter notebook supports programming languages such as Python and more. Jupyter Notebook was used to test-train the data.

The primary language that the software was coded in was Python, and Python is an object-orientated, high-level programming language that was used to create the graphical user interface and user interface.

2.3 Product Functions

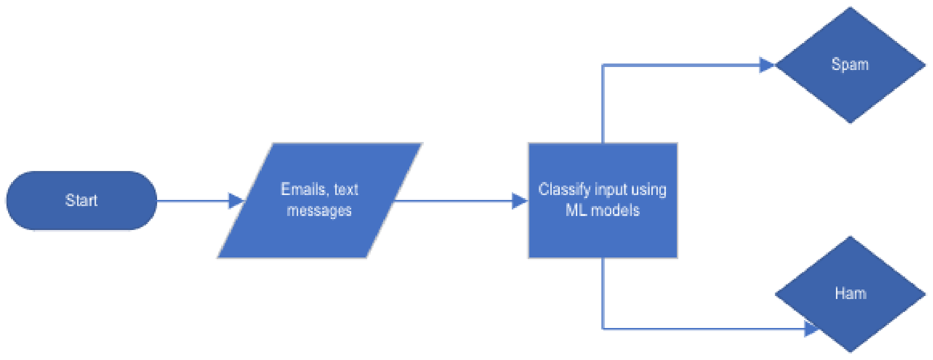
2.3.1 Python

2.3.1.1 Needed to code the GUI-Software.

2.3.2 Google Colab-Jupyter Notebook

2.3.2.1 Test and train the data of the Spam Texts/Emails.

2.3.2.2 Create bars, graphs, plots, and data visualizations.



2.4 User Classes and Characteristics

This product has only one level of functionality. The features inside the product are accessible to data scientists, teachers, professors, and university students.

2.5 Operating Environment

The software should work on any OS such as Windows, macOS, or Linux distributions. The team mainly used Windows, macOS, or Linux operating systems, as a result. The software takes emails and text messages as input. It classifies the input using different classification Machine Learning models and outputs whether the emails and text messages are spam or ham.

2.6 Design and Implementation Constraints

2.6.1 Dataset

2.6.1.1 Massive amount of Spam Texts and Emails dataset.

2.6.1.2 Testing and training the data with a high-low speed ethernet-Wi-Fi.

The team encountered many difficulties with needing a powerful PC to test and train the data in a timely manner.

2.7 User Documentation

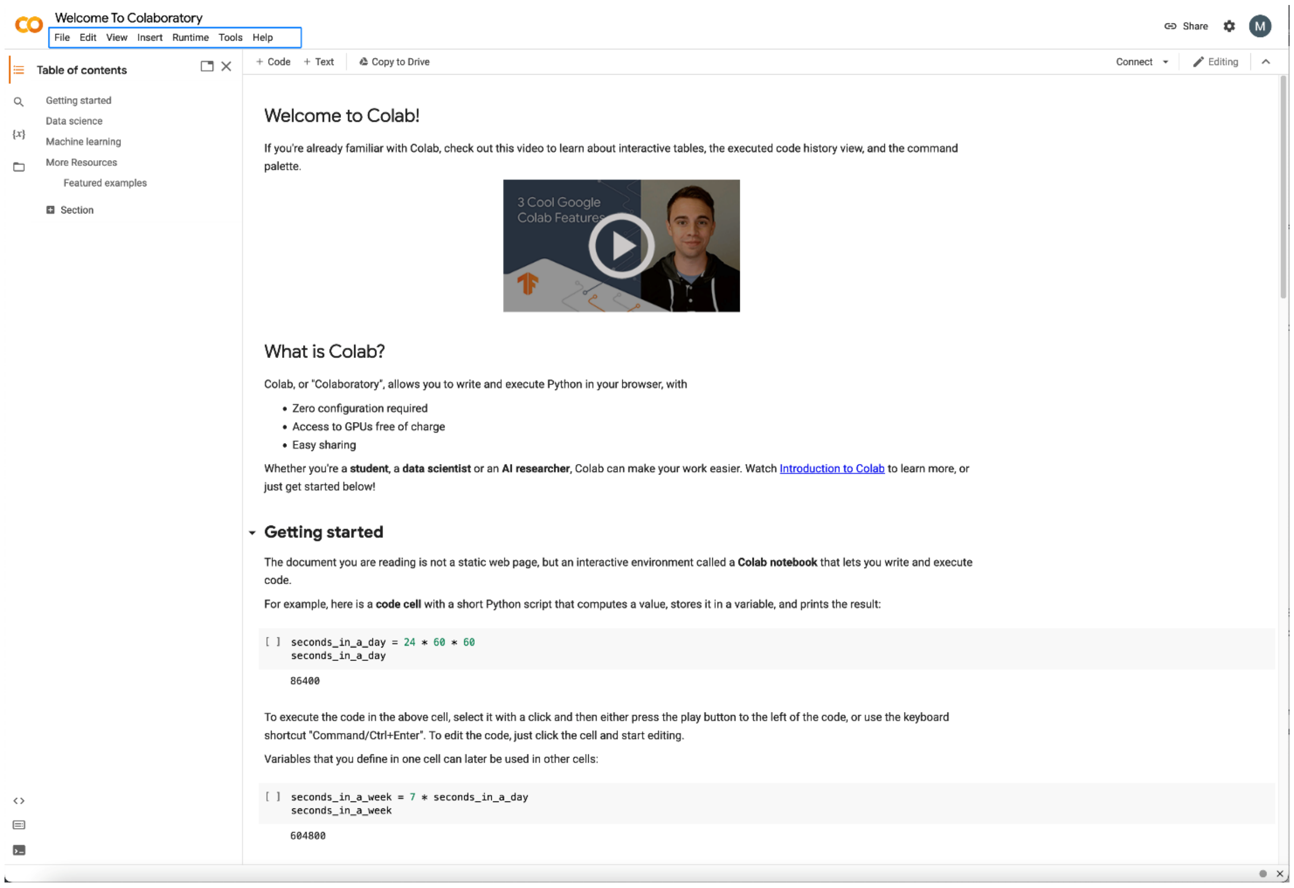
Running required software/environment on Google Colab, Jupyter Notebook, and Python:

Part A)

I. Sign into your Google account

II. GO TO https://colab.research.google.com/

  a. Screenshots of running Google Colab



Welcome to Colab!

If you're already familiar with Colab, check out this video to learn about interactive tables, the executed code history view, and the command palette.

What is Colab?

Colab, or "Colaboratory", allows you to write and execute Python in your browser, with

- Zero configuration required
- Access to GPUs free of charge
- Easy sharing

Whether you're a **student**, a **data scientist** or an **AI researcher**, Colab can make your work easier. Watch Introduction to Colab to learn more, or just get started below!

Getting started

The document you are reading is not a static web page, but an interactive environment called a **Colab notebook** that lets you write and execute code.

For example, here is a **code cell** with a short Python script that computes a value, stores it in a variable, and prints the result:

```
[ ] seconds_in_a_day = 24 * 60 * 60
    seconds_in_a_day

    86400
```

To execute the code in the above cell, select it with a click and then either press the play button to the left of the code, or use the keyboard shortcut "Command/Ctrl+Enter". To edit the code, just click the cell and start editing.

Variables that you define in one cell can later be used in other cells:

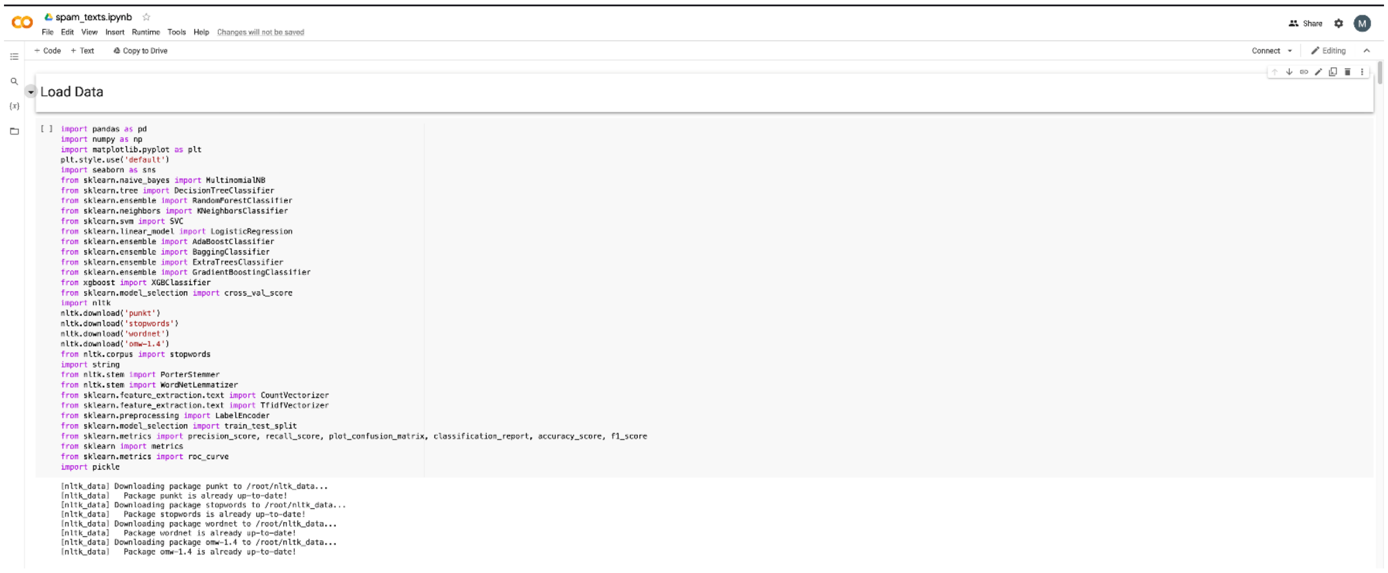
```
[ ] seconds_in_a_week = 7 * seconds_in_a_day
    seconds_in_a_week

    604800
```

III. Next GO TO File

  a. New Notebook

  b. Screenshot of New Notebook from Google Colab, up and running.


spam_texts.ipynb
File Edit View Insert Runtime Tools Help Changes will not be saved
+ Code + Text Copy to Drive
Share
Connect
Editing
Load Data

```
[ ] import pandas as pd
    import numpy as np
    import matplotlib.pyplot as plt
    plt.style.use('default')
    import seaborn as sns
    from sklearn.naive_bayes import MultinomialNB
    from sklearn.tree import DecisionTreeClassifier
    from sklearn.ensemble import RandomForestClassifier
    from sklearn.neighbors import KNeighborsClassifier
    from sklearn.svm import SVC
    from sklearn.linear_model import LogisticRegression
    from sklearn.ensemble import AdaBoostClassifier
    from sklearn.ensemble import BaggingClassifier
    from sklearn.ensemble import ExtraTreesClassifier
    from sklearn.ensemble import GradientBoostingClassifier
    from xgboost import XGBClassifier
    from sklearn.model_selection import cross_val_score
    import nltk
    nltk.download('punkt')
    nltk.download('stopwords')
    nltk.download('wordnet')
    nltk.download('omw-1.4')
    from nltk.corpus import stopwords
    import string
    from nltk.stem import PorterStemmer
    from nltk.stem import WordNetLemmatizer
    from sklearn.feature_extraction.text import CountVectorizer
    from sklearn.feature_extraction.text import TfidfVectorizer
    from sklearn.preprocessing import LabelEncoder
    from sklearn.model_selection import train_test_split
    from sklearn.metrics import precision_score, recall_score, plot_confusion_matrix, classification_report, accuracy_score, f1_score
    from sklearn import metrics
    from sklearn.metrics import roc_curve
    import pickle

    [nltk_data] Downloading package punkt to /root/nltk_data...
    [nltk_data]   Package punkt is already up-to-date!
    [nltk_data] Downloading package stopwords to /root/nltk_data...
    [nltk_data]   Package stopwords is already up-to-date!
    [nltk_data] Downloading package wordnet to /root/nltk_data...
    [nltk_data]   Package wordnet is already up-to-date!
    [nltk_data] Downloading package omw-1.4 to /root/nltk_data...
    [nltk_data]   Package omw-1.4 is already up-to-date!
```

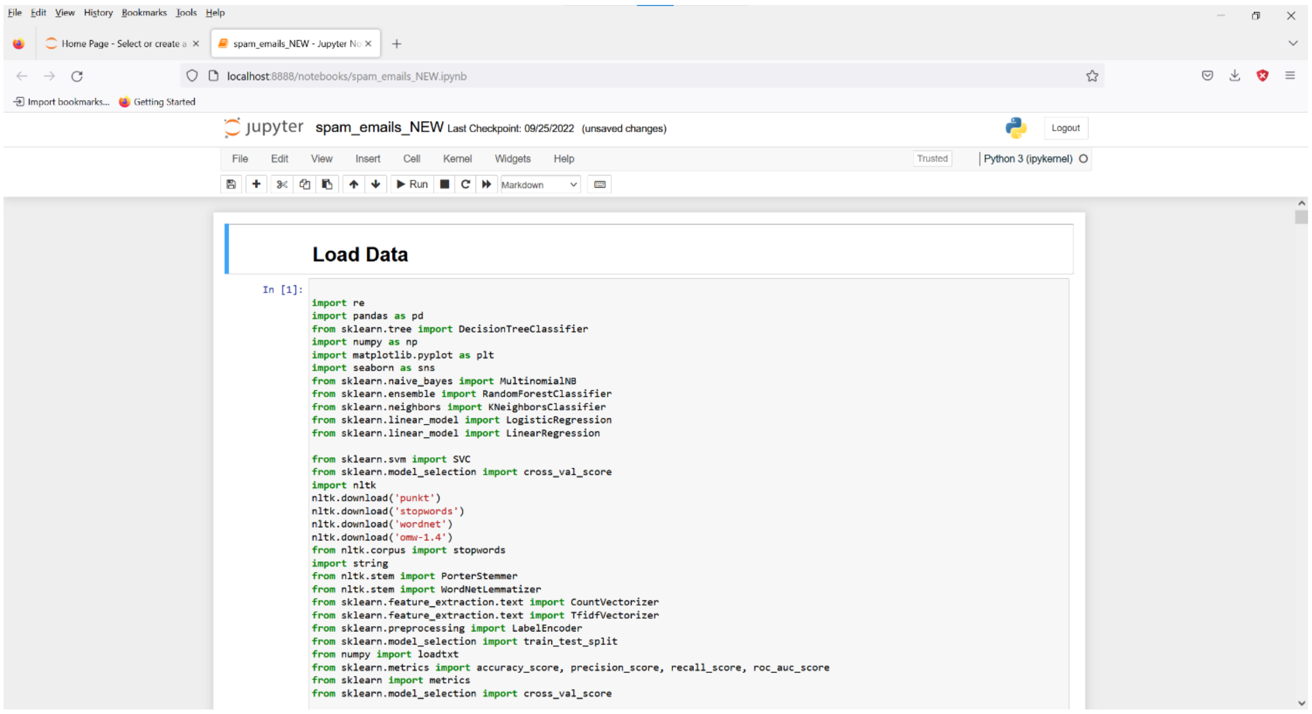
Part B)

IV. Running Jupyter Notebook

a. Start and type in Jupyter Notebook (Anaconda3)

i. Next click New Notebook

ii. Screenshot of running Notebook in Jupyter


File Edit View History Bookmarks Tools Help
Home Page - Select or create a
spam_emails_NEW - Jupyter No
localhost:8888/notebooks/spam_emails_NEW.ipynb
Import bookmarks...
Getting Started
jupyter spam_emails_NEW Last Checkpoint: 09/25/2022 (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Markdown
Load Data

```
In [1]:
        import re
        import pandas as pd
        from sklearn.tree import DecisionTreeClassifier
        import numpy as np
        import matplotlib.pyplot as plt
        import seaborn as sns
        from sklearn.naive_bayes import MultinomialNB
        from sklearn.ensemble import RandomForestClassifier
        from sklearn.neighbors import KNeighborsClassifier
        from sklearn.linear_model import LogisticRegression
        from sklearn.linear_model import LinearRegression

        from sklearn.svm import SVC
        from sklearn.model_selection import cross_val_score
        import nltk
        nltk.download('punkt')
        nltk.download('stopwords')
        nltk.download('wordnet')
        nltk.download('omw-1.4')
        from nltk.corpus import stopwords
        import string
        from nltk.stem import PorterStemmer
        from nltk.stem import WordNetLemmatizer
        from sklearn.feature_extraction.text import CountVectorizer
        from sklearn.feature_extraction.text import TfidfVectorizer
        from sklearn.preprocessing import LabelEncoder
        from sklearn.model_selection import train_test_split
        from numpy import loadtxt
        from sklearn.metrics import accuracy_score, precision_score, recall_score, roc_auc_score
        from sklearn import metrics
        from sklearn.model_selection import cross_val_score
```

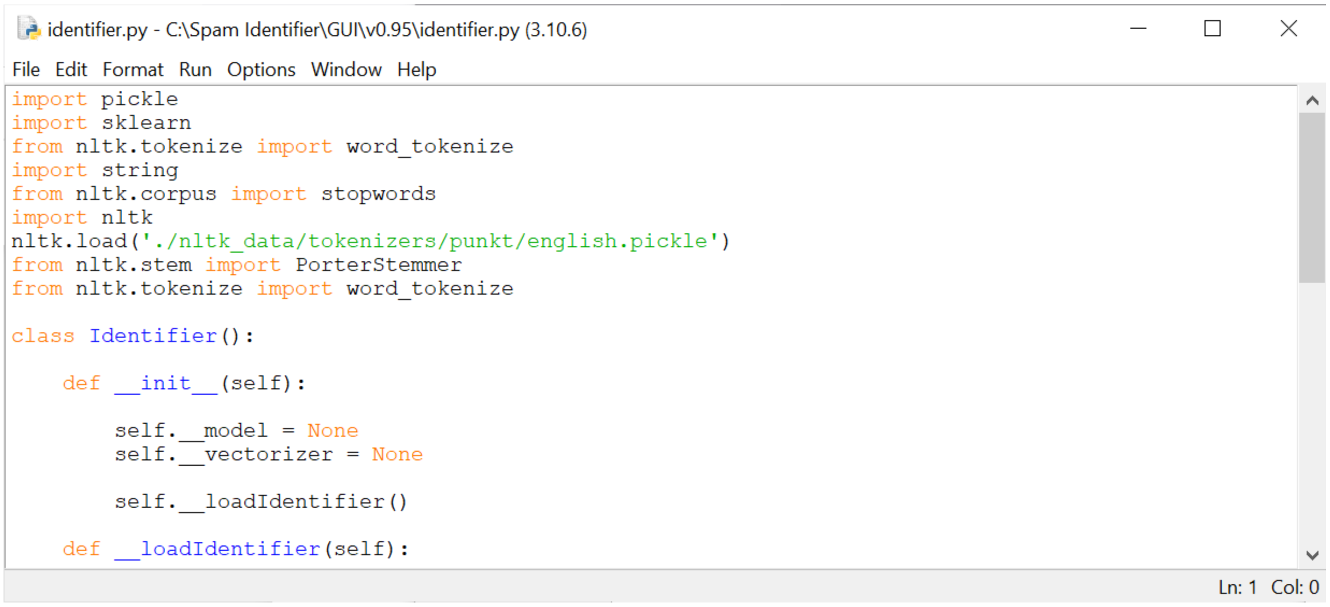
Part C)

V. Running Python

a. Start and type in Python

b. Run IDLE Shell in Python

i. Screenshot of running Python


identifier.py - C:\Spam Identifier\GUI\v0.95\identifier.py (3.10.6)
File Edit Format Run Options Window Help
Ln: 1 Col: 0

```
import pickle
import sklearn
from nltk.tokenize import word_tokenize
import string
from nltk.corpus import stopwords
import nltk
nltk.load('./nltk_data/tokenizers/punkt/english.pickle')
from nltk.stem import PorterStemmer
from nltk.tokenize import word_tokenize

class Identifier():

    def __init__(self):

        self.__model = None
        self.__vectorizer = None

        self.__loadIdentifier()

    def __loadIdentifier(self):
```

2.8 Assumptions and Dependencies

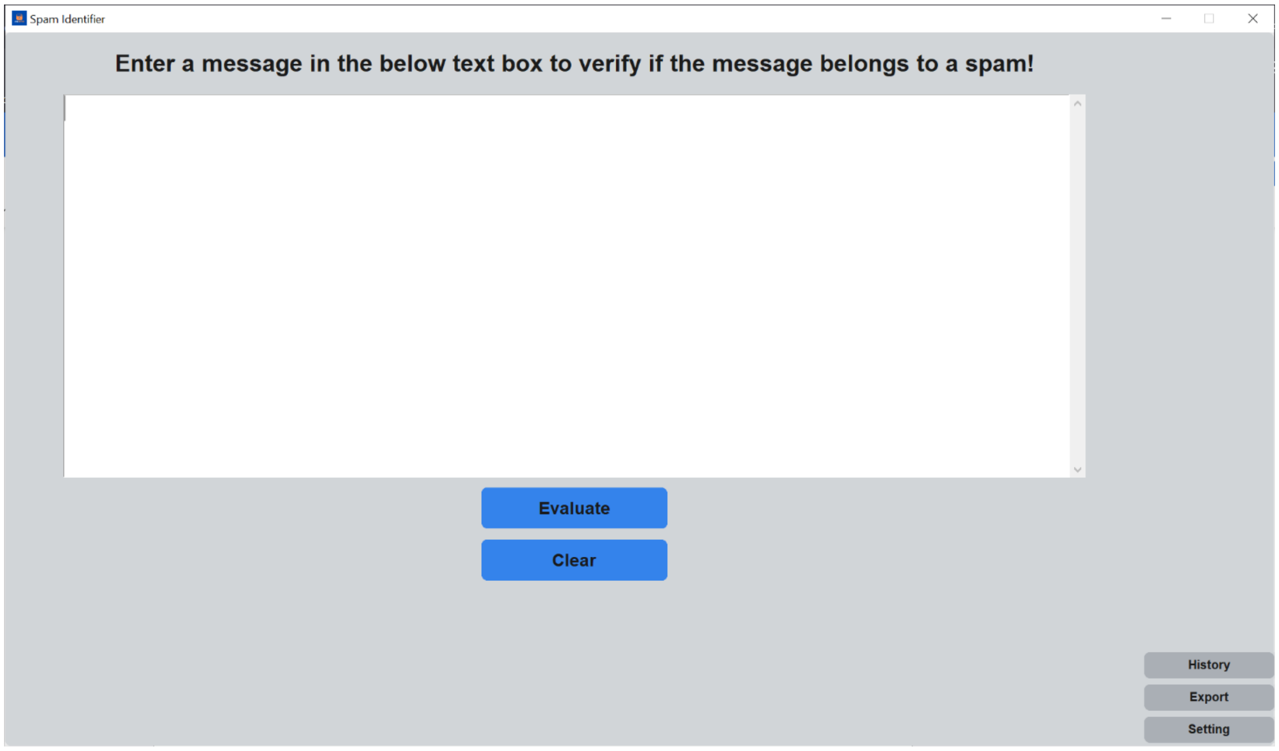
It has been assumed that developers will have all the machine requirements, the appropriate system, and a sufficient Operating System. A developer will need a powerful GPU for testing and training the data and a good CPU.

2.9 Apportioning of Requirements

The future requirement of this project is to create Machine Learning Based Spam Detection software for data scientists and a Graphical User Interface (GUI). Also, the future goal is to implement/Test more Machine Learning models with Spam Based Texts/Emails.

3. External Interface Requirements

3.1 User Interfaces


Spam Identifier
Enter a message in the below text box to verify if the message belongs to a spam!
Evaluate
Clear
History
Export
Setting

Above is the User Interface.

The UI typically starts on a pop window when the main.py Python file has been called. After that, a UI is shown; see above. Note that the top left-hand corner displays our logo of the Spam Identifier.

Next, below, we have a text message that states what the user should type in the text box, which is the white part. The user has two options; evaluate and clear. What the clear does is, delete the message in the text. And the evaluate will determine the text/message to specify if the message belongs to a spam or ham.

The user sees three UI buttons in the lower right-hand corner: history, export, and setting. History specifies the history of the messages that are spam or ham. The export will export your result/message, and the setting will change the resolution of the User Interface and Export Directory, if necessary.

In addition, to the UI, there is a back button that goes back to the display and an apply button to make changes to the User Interface, which is done through Setting.

3.2 Hardware Interfaces

This software does not have a hardware interface.

3.3 Software Interfaces

3.3.1 Google Colab, Colab Python Version Number 3.x

3.3.1.1 https://colab.research.google.com

3.3.2 Jupyter Notebook, version number 5. x

3.3.2.1 https://jupyter.org/

3.3.3 Python, version number 3.10.x

3.3.3.1 https://www.python.org/downloads/

3.4 Communications Interfaces

3.4.1 Collaborative tools

3.4.1.1 The team used Zoom for meetings through voice calls and Discord to communicate with each other.

3.4.1.2 Project work-planning was done through Google Drive, Google slides, and Microsoft docs.

3.4.1.3 Project testing was done through Google Colab, Jupyter Notebook, Python.

3.4.1.4 Google credentials access.

4. Requirements Specification

4.1 Functional Requirements

4.1.1 Sequence of Operations

4.1.1.1 The system shall show the main user interface page and allow the user to choose which type of input they want to use as an evaluation tool:

- Email
- Text message

4.1.1.2 The system evaluation method shall take and evaluate the input and return the result to the user.

4.1.1.3 The system shall have a History UI page that will allow the user to show all prior input evaluation results, including:

- Input message
- Result
- Types of evaluation

4.1.1.4 The system shall have a settings page that allows the user to adjust different settings including:

- Window resolution
- Export directory

4.1.1.5 The system shall have an export function that allows the user to export all prior evaluation viewable through the History UI to the target location.

4.2 External Interface Requirements

This does not apply.

4.3 Logical Database Requirements

This does not apply.

4.4 Design Constraints

4.4.1 Hardware

4.4.1.1 Google Colab utilizes GPU acceleration

4.4.1.2 Jupyter Notebook utilizes GPU acceleration

4.4.1.3 If the minimum requirement of the hardware capacity is not met, the system will most likely fail.

4.4.2 Software

4.4.2.1 User can be in Mac-Windows-Linux environments to obtain results

4.4.3 Software application user requirements

4.4.3.1 Access to Ethernet or Wi-fi

4.4.3.2 Usable CSV files

4.4.3.3 Functioning computer

5. Other Nonfunctional Requirements

5.1 Performance Requirements

5.1.1 The software should perform-take no more than x seconds of opening the browsers, web application, and software application.

5.2 Safety Requirements

5.2.1 The user should not share the data visualization with other companies/users.

5.2.2 The user can only use data that they have permission to use.

5.3 Security Requirements

5.3.1 The user must have credentials for access to Google Colab.

5.3.2 Only the user has access to the outputted results of the system.

5.3.3 The developers will not have access to any data that is run through the machine learning models.

5.4 Software Quality Attributes

5.4.1 Usability

5.4.1.1 It is important that the software be usable for the user, and that the user is able to complete the desired functions after:

- Reading the instructions
- Using the context clues provided by the icons and button descriptions

5.4.2 Correctness

5.4.2.1 The software should achieve a level of correctness and accuracy when outputting the results.

5.4.3 Availability

5.4.3.1 The software should be available at all times except for the following occurrences:

- Scheduled service downtime
- Google Colab downtime
- Jupyter Notebook downtime

5.5 Business Rules

5.5.1 Any changes involving the machine learning models or the GUI should only be completed by the original developers.

6. Legal and Ethical Considerations

There are no legal and/or ethical issues involved in this project.

Appendix A: Glossary

- Accuracy
  - Accuracy is one metric for evaluating classification models
- Anaconda
  - Anaconda is an open-source distribution of the Python and R programming languages for data science that aims to simplify package management and deployment
- Binary Label Maps
  -
- Classification model
  - A type of model that distinguishes between two or more discrete classes
- Comma-Separated Values (CSV)
  - A comma-separated values (CSV) file is a delimited text file that uses a comma to separate values
- Data pre-processing
  - A component of data preparation, describes any type of processing performed on raw data to prepare it for another data processing procedure
- Exploratory Data Analysis (EDA)
  - Refers to the critical process of performing initial investigations on data so as to discover patterns, to spot anomalies, to test hypothesis and to check assumptions with the help of summary statistics and graphical representations
- Emails
  - Messages distributed by electronic means from one computer user to one or more recipients via a network
- Ensemble learning
  - Process by which multiple models, such as classifiers or experts, are strategically generated and combined to solve a particular computational intelligence problem
- Feature extraction
  - Feature extraction refers to the process of transforming raw data into numerical features that can be processed while preserving the information in the original data set
- Google Colab
  - Colaboratory, or "Colab" for short, is a product from Google Research and allows anybody to write and execute arbitrary python code through the browser, and is especially well suited to machine learning, data analysis and education
- Graphical User Interface (GUI)
  - A graphical user interface (GUI) is an interface through which a user interacts with electronic devices such as computers and smartphones through the use of icons, menus and other visual indicators or representations (graphics)
- IPYNB
  - Notebook document file type for Jupyter Notebook
- Jupyter Notebook
  - Original web application for creating and sharing computational documents
- Local host

    - A hostname that refers to the local machine currently making the request
- Machine Learning (ML):
    - A branch of artificial intelligence (AI) and computer science which focuses on the use of data and algorithms to imitate the way that humans learn, gradually improving its accuracy
- Natural Language Processing (NLP):
    - A component of AI in the field of linguistics that deals with interpretation and manipulation of human speech or text using software. It enables the computer to understand the natural way of human communication by combining machine learning, deep learning and statistical models
- Precision
    - The quality, condition, or fact of being exact and accurate
- Python
    - A high-level general-purpose programming language.
- Recall
    - Calculated as the ratio between the number of Positive samples correctly classified as Positive to the total number of Positive samples
- Spam
    - Unnecessary, unwanted, or repetitive content that clogs inboxes and clutters social media feeds
- User Interface (UI)
    - The point of human-computer interaction and communication in a device.
- Word stemming
    - Process of reducing a word to its word stem that affixes to suffixes and prefixes or to the roots of words known as a lemma
- Word lemmatization
    - Grouping together of different forms of the same word

Appendix B: Analysis Models

Appendix C: References To Be Determined List

1. "Ham." *Ham - SPAMASSASSIN - Apache Software Foundation*, https://cwiki.apache.org/confluence/display/spamassassin/Ham#:~:text=%22Ham%22%20is%20e%2Dmail,non%2Dspam%22%2C%20instead.
2. "Machine Learning Glossary | Google Developers." *Google*, Google, https://developers.google.com/machine-learning/glossary.
3. Technical Updates November 2022 | CDC. https://www.cdc.gov/nssp/news/2022/11-november/technical-updates.html
4. "Consultation Number: 210435012_amo_irbe The Project Management Assistance Mission For The Supply Of Supply And Installation Of Electric Bus Charging Infrastructure Party Load System. [Tender Documents : T462602691]." MENA Report, Albawaba (London) Ltd., May 2021.