

**Spam email Classification Using ML Models**

# **AIE241 || Natural Language Processing**

**Project Team members**

|  |  |  |  |
| --- | --- | --- | --- |
| **Student ID** | **Student Name** | **Project Evaluation** | |
| 222102487 | Aser Mohamed Ali Mahmoud | Report |  |
| Presentation |  |
| Code |  |
| Total |  |
| 222100794 | Seif El-Deen Samer Samy | Report |  |
| Presentation |  |
| Code |  |
| Total |  |

**Under supervision of**

Prof. Manar El-Shazly

TA || Dr. Radwa Taher

**Summer-2024**

**Introduction:**

A good text classifier is a classifier that efficiently categorizes large sets of text documents in a reasonable time frame and with acceptable accuracy, and that provides classification rules that are humanly readable for possible fine-tuning. If the training of the classifier is also quick, this could become in some application domains a good asset for the classifier. Many techniques and algorithms for automatic text categorization have been devised.

The text classification task can be defined as assigning category labels to new documents based on the knowledge gained in a classification system at the training stage. In the training phase, we are given a set of documents with class labels attached, and a classification system is built using a learning method. Classification is an important task in both data mining and machine learning communities, however, most of the learning approaches in text categorization are coming from machine learning research.

**Context:**

The email Spam Collection is a set of email tagged messages that have been collected for SMS Spam research. It contains one set of email messages in English of 33,857 email, tagged according being ham (legitimate) or spam.

**Project Overview:**

* Classifying email messages as spam or not **(often called “ham”)** is a common task in **natural language processing (NLP).**
* Created a machine learning model that **detects and classifies a SMS into SPAM or HAM (normal) based on the textual data using Natural Language Processing.**
* **Engineered features like word count, contains currency symbol, and contains number** from the text SMS.

**How will this project help?**

* This project helps in filtering and cleaning the SMS from the phone.

**Aim:**

The email Spam Classifier project aims to develop a machine learning model capable of distinguishing between spam and non-spam (ham) SMS emails. This project addresses the issue of unwanted email spam and provides a solution for filtering out spam messages.

**Objectives and Techniques used:**

1. **Developing a Machine Learning Model:** Creating an effective and accurate classifier to distinguish between spam and ham (legitimate) emails . Ensuring that the model is not only accurate but also optimized for speed and resource usage during real-time classification.
2. **Data Collection and Preprocessing:** Gathering a comprehensive dataset of emails labeled as spam or ham. Preprocessing the data by meticulously removing noise (e.g., punctuation, stopwords) and tokenizing the text into meaningful units. This step includes normalizing text, stemming/lemmatizing words, and converting text into numerical features (e.g., using TF-IDF or word embeddings) to enhance model accuracy.
3. **Model Development**: Implementing and fine-tuning machine learning models using algorithms such as Naive Bayes or Support Vector Machines (SVM), which are known for their effectiveness in text classification tasks. Carefully selecting hyperparameters, optimizing feature selection, and possibly employing techniques like cross-validation to ensure that the model generalizes well to new data.
4. **Model Evaluation:** Rigorously evaluating the classifier's performance using standard metrics like accuracy, precision, recall, and F1-score. Analyzing these metrics to identify and address any weaknesses in the model, such as imbalanced classes or overfitting, and iterating on the model to improve its predictive power.
5. **Deployment:** Deploying the trained model in a production environment where it can classify incoming SMS messages in real-time. Ensuring that the deployment pipeline is robust, scalable, and capable of handling high volumes of messages with minimal latency, while maintaining the model's accuracy and efficiency in filtering spam.

**Expected Deliverables:**

1. Trained machine learning model for email spam classification.
2. Evaluation report detailing the performance of the classifier on test data.
3. Documentation and codebase for replicating the project, including data preprocessing steps, model implementation, and deployment instructions.

**Resources Used:**

* Packages: **pandas, numpy, sklearn, matplotlib, seaborn, nltk.**
* **Dataset:** The project utilizes a dataset consisting of labeled email messages, with each message categorized as either spam or ham. The dataset is split into training and testing sets to train the classifier and evaluate its performance.
* Dataset by **UCI Machine Learning on Kaggle**:

[SMS Spam Collection Dataset (kaggle.com)](https://www.kaggle.com/datasets/uciml/sms-spam-collection-dataset)

* **The original dataset:**

[SMS Spam Collection - UCI Machine Learning Repository](https://archive.ics.uci.edu/dataset/228/sms+spam+collection)

**Exploratory Data Analysis:**

**•** Exploring values in dataset.

• Plotted count plot for SMS labels Spam vs. Ham.

**Model Architecture:**

The email spam classifier employs a variety of natural language processing (NLP) techniques, including tokenization, text preprocessing, and feature extraction. It utilizes machine learning algorithms such as Naive Bayes, Support Vector Machines (SVM), or Recurrent Neural Networks (RNNs) to learn patterns and classify messages accurately.

**Evaluation Metrics:**

The performance of the classifier is evaluated using standard metrics such as accuracy, precision, recall, and F1-score. These metrics provide insights into how well the classifier distinguishes between spam and ham messages and its overall effectiveness in spam detection.

**Usage:**

To use the email spam classifier:

1. Clone the repository to your local machine.
2. Install the required dependencies listed in the requirements.txt file.
3. Run the provided Jupyter notebook or Python script to train the classifier and make predictions on new SMS messages.

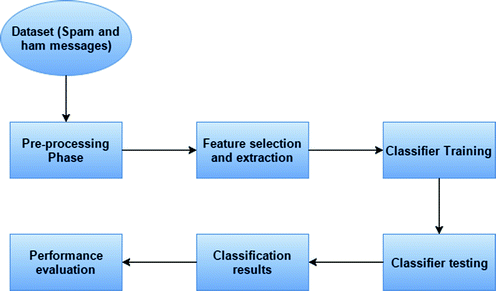
**Future Improvements:**

Potential enhancements to the project include:

* Experimenting with different machine learning algorithms to improve classification accuracy.
* Fine-tuning hyperparameters and text preprocessing techniques for better performance.
* Exploring advanced NLP models or ensemble methods to handle complex text data more effectively.

**Conclusion:**

The email Spam Classifier project offers a practical solution for identifying and filtering out unwanted spam emails. By leveraging machine learning techniques, we aim to develop an effective classifier. capable of improving user experience and reducing the impact of SMS spam within a one-month timeframe.

**Work Flow:**

**Implementation and Outputs** (Approach of NLP used):-

1. **Reading the Data:**



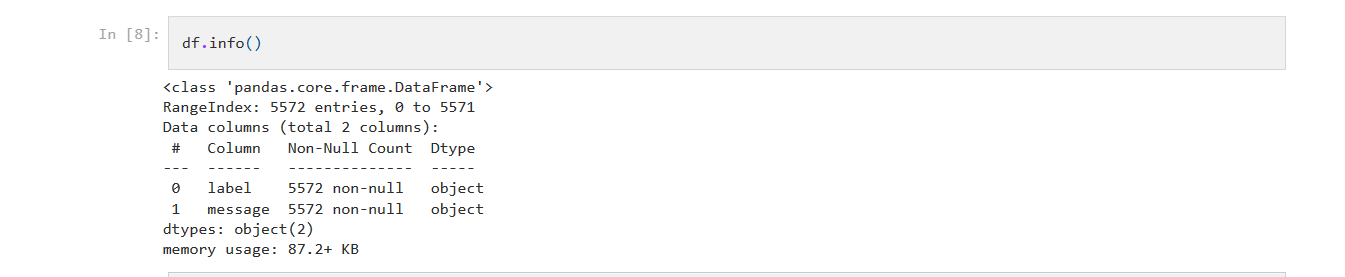
To read the data, we should import the Pandas library and read the data using *pd.read\_csv()*.

Here the data in the file is Tab (\t) separated, so we must provide the “*sep*” (separate) parameter. Also, the file does not contain any column names, so we should provide column names using “*names*” parameter.

Data is getting stored in a data frame named “*messages*”.

To view the first 5 rows of data, we should use *messages.head()*

We can see here that the first column contains the **Labels (dependent variable)**i.e., a message is Spam or Ham and the second column contains the actual **Messages (independent variable)**

1. **Exploratory Data Analysis:**

We can see here that there are 33,857 rows and 2 columns. It means that there are 33,857 messages and 2 columns named “*Label*” and “*text*”.

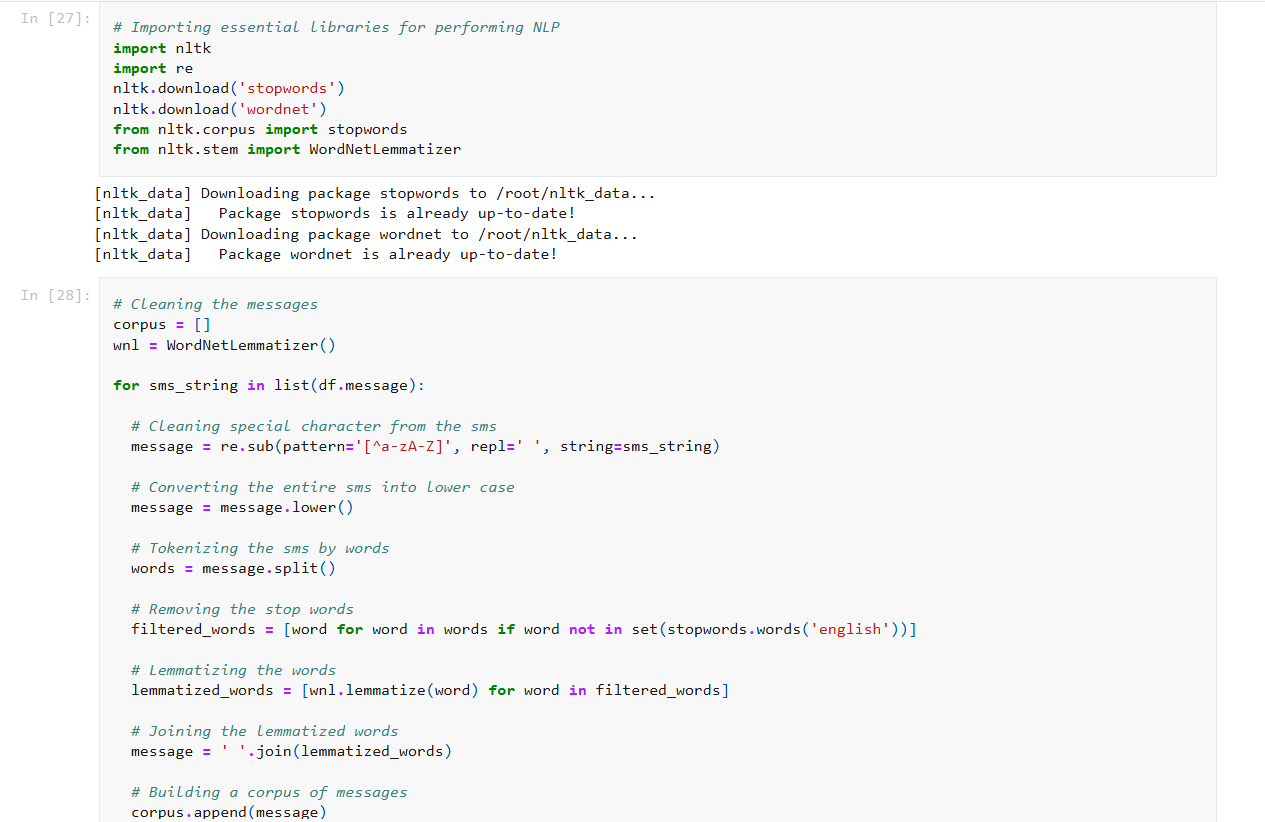
1. **Text (data) Preprocessing:**

* **Tokenization:** Breaking down the SMS text into individual words or tokens.
* **Lowercasing:** Converting all text to lowercase to maintain consistency.
* **Eliminating punctuation marks** as they typically do not contribute to meaning.
* **Stop-word Removal:** Removing common words like "and," "the," etc., which may not be helpful for classification.
* **Lemmatization:** Reducing words to their base or root form (e.g., "running" to "run") to treat similar words uniformly.

**The Data Cleaning performed:**

* Removing special character and numbers using regular expression
* Converting the entire SMS into lower case
* Tokenizing the SMS by words
* Removing the stop words
* Lemmatizing the words
* Joining the lemmatized words
* Building a corpus of messages

Here, we are calculating the Length and Punctuation of each message for further analysis, and this is added to data frame (messages) as Column:

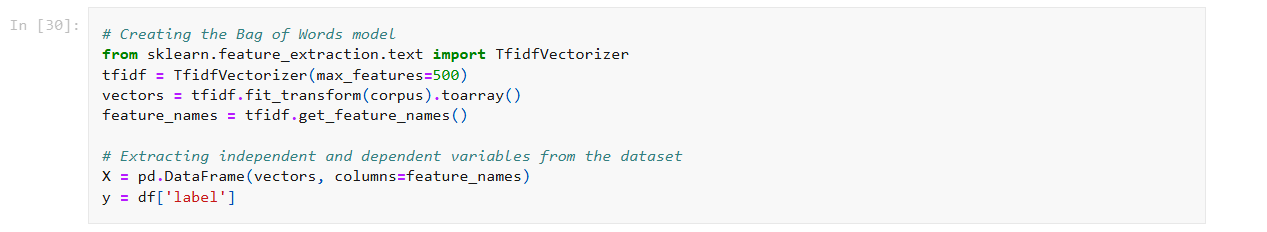
***Text Cleaning>>***

Messages contain text but with many punctuation, Stop-words (These are the words in any language which does not add much meaning to a sentence. They can safely be ignored without sacrificing the meaning of the sentence), special characters, and many forms of verb.

* Now we will clean messages by removing the unnecessary things.
* We will import **re** (regex library) and from nltk library, we will import **stopwords** and **wordnet Lemmatizer** (One such method of Lemmitization) and create its object.
* Now we will iterate through every message, and using regex (substitute method), we will take everything from the message except small alphabets(a-z) and capital alphabets(A-Z) and substitute it with blank space. Next, we will lowercase the message (as abc is not the same as ABC) to make learning easy for the machine and then split it into words. Will pass split words in a list comprehension where we will check each word, that whether it exists in the stopwords collection by nltk , and if the word is not in stopwords, it will be lemmatized using wordnet Lemmatizer object. After each word is lemmatized, we will again join the words and form a sentence and append in the corpus list of sentences.

1. **Feature Extraction:-**

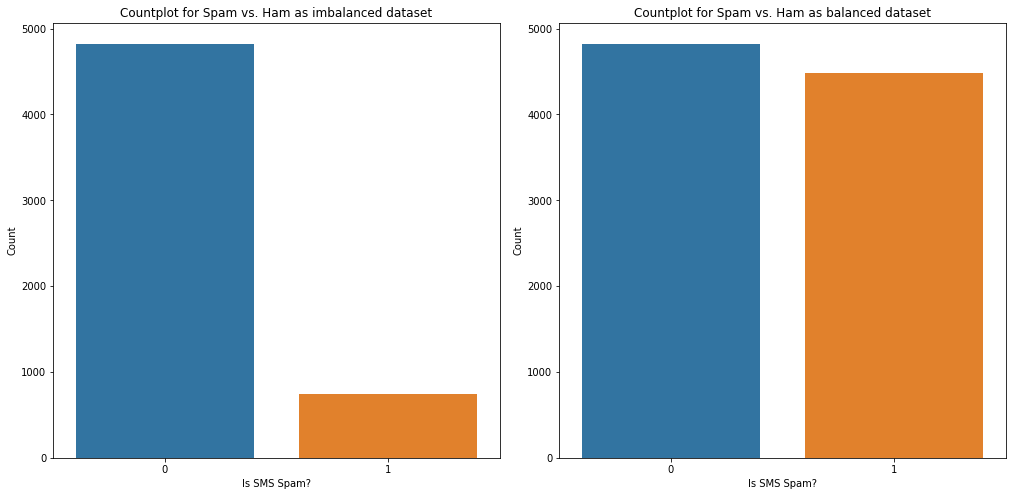
**Dealing with Text (Natural Language data):**

* **Bag of Words (BOW):** Represents the text as a vector of word frequencies, where each word corresponds to a feature.
* **Term Frequency-Inverse Document Frequency (TF-IDF):** Weighs the frequency of words by their importance across the dataset, giving more weight to rare but important words.

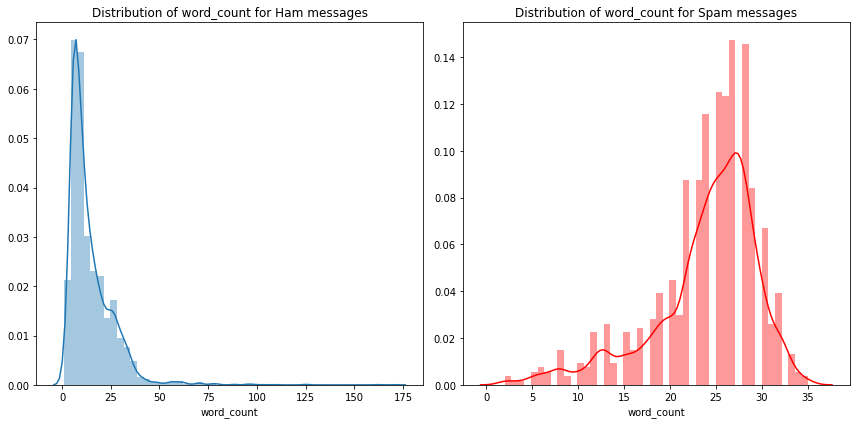
1. **Feature Engineering:-**

**• Handling imbalanced dataset using Over sampling:**

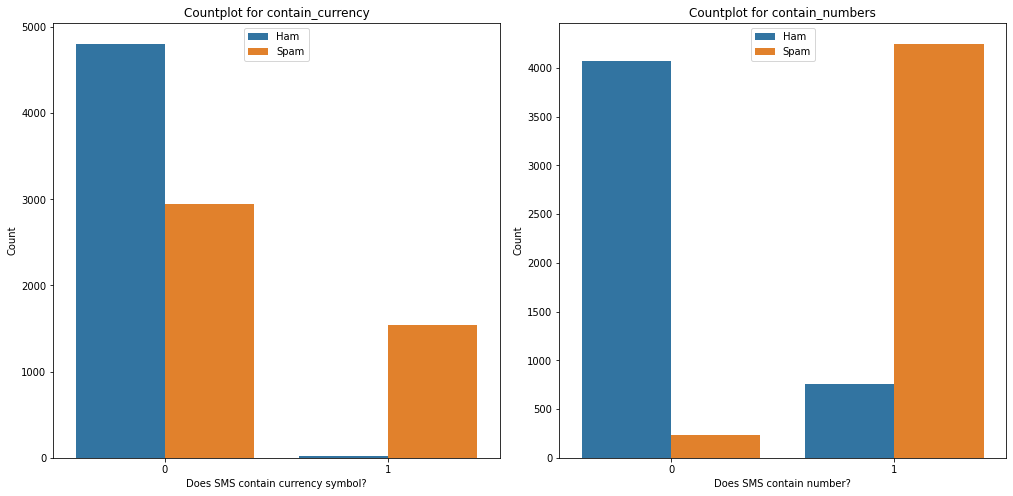
**Spam Vs Ham:**

****

**• Creating new features from existing features:** word count, contains currency symbol, contains numbers.

**word count:**

**Currency numbers:**

****

1. **Model Training:**

* **Choosing an Algorithm:** Three ML algorithms used for spam SMS classification include:
* **Naive Bayes:** Particularly effective for text classification tasks.
* **Decision Tree:** splits the data based on features (words or n-grams) to separate different classes (spam or ham).
* **Random Forest:** An ensemble method that can capture complex patterns.
* **Training:** The model is trained using labeled data (spam or ham) to learn patterns in the text.

1. **Model Building and Evaluation:-**

**Metric: F1-Score:**

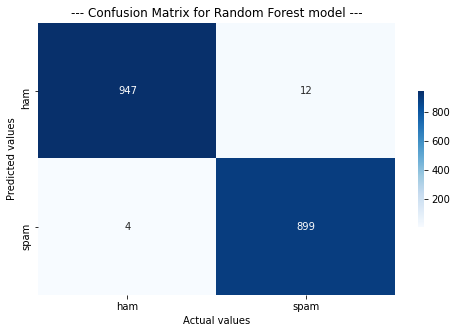
• Multinomial Naive Bayes: **0.943**

• Decision Tree: **0.98**

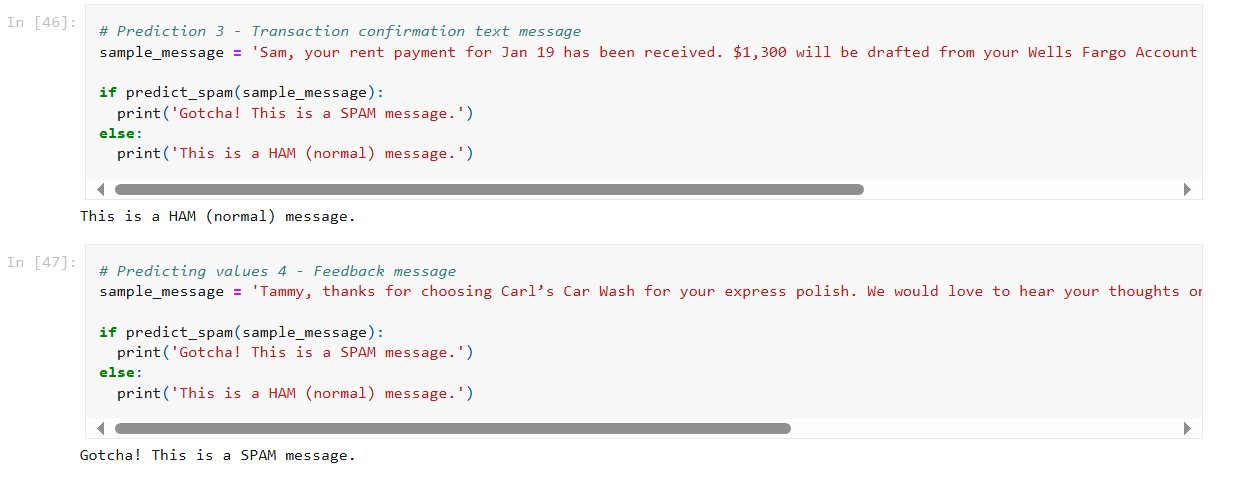
•Random Forest: **0.994**

• Voting (Decision Tree + Multinomial Naive Bayes): **0.98**

• LSTM : **0.95**



**Note:** Evaluation scores are obtained using cross validation.

1. **Model Prediction:**

**References:**

* <http://www.dt.fee.unicamp.br/~tiago/smsspamcollection/>
* [SMS Spam Collection Dataset (kaggle.com)](https://www.kaggle.com/datasets/uciml/sms-spam-collection-dataset)
* [SMS Spam Collection - UCI Machine Learning Repository](https://archive.ics.uci.edu/dataset/228/sms+spam+collection)
* [Build SMS Spam Classification Model using Naive Bayes & Random Forest | by Dhaval Thakur | Towards Data Science](https://towardsdatascience.com/build-sms-spam-classification-model-using-naive-bayes-random-forest-43465d6617ed)
* [Natural-Language-Processing-Projects/Spam SMS Classification/Spam SMS Classication.ipynb at master · anujvyas/Natural-Language-Processing-Projects · GitHub](https://github.com/anujvyas/Natural-Language-Processing-Projects/blob/master/Spam%20SMS%20Classification/Spam%20SMS%20Classication.ipynb)
* Text Preprocessing (Tokenization, Stopword Removal, Lemmatization, etc.), Book: *"Speech and Language Processing"* by Daniel Jurafsky and James H. Martin, from: [The Stanford Natural Language Processing Group](https://nlp.stanford.edu/)
* Regular Expressions (regex) in Python, Official Documentation: [re — Regular expression operations — Python 3.12.5 documentation](https://docs.python.org/3/library/re.html)
* NLTK (Natural Language Toolkit), Online book: "Natural Language Processing with Python", The official website: [NLTK :: Natural Language Toolkit](https://www.nltk.org/)
* Lemmatization, Research Paper: "A Survey of Stemming Algorithms" by M.F, From: [EJ1020841.pdf (ed.gov)](https://files.eric.ed.gov/fulltext/EJ1020841.pdf), Article: [2— Stemming & Lemmatization in NLP: Text Preprocessing Techniques | by Aysel Aydin | Medium](https://ayselaydin.medium.com/2-stemming-lemmatization-in-nlp-text-preprocessing-techniques-adfe4d84ceee#:~:text=Stemming%20involves%20removing%20suffixes%20from%20words%20to%20obtain,and%20obtain%20the%20base%20form%20of%20a%20word.), - A concise explanation of lemmatization, with examples and differences from stemming.