# Email Spam Classification

## A Micro Project Report

### Submitted by

**D.Venkat Abhilash 99220040498**

## B.Tech - [CSE], Specialization[AIML]



**Kalasalingam Academy of Research and Education**

#### (Deemed to be University)

**Anand Nagar, Krishnankoil - 626 126**

## [February] [2024]



**SCHOOL OF COMPUTING**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

# BONAFIDE CERTIFICATE

Bonafide record of the work done by D. Venkat Abhilash - 99220040498 in partial fulfill- ment of the requirements for the award of the degree of Bachelor of Technology in Specializa- tion of the Computer Science and Engineering, during the Academic Year Even Semester (2023-24).

**[Project Guide Name] [Faculty Incharge Name]**

**Project Guide Faculty Incharge**

**[Designation] [Designation]**

**[Department Name] [Department Name]**

**Kalasalingam Academy of Kalasalingam Academy of**

**Research and Education Research and Education**

**Krishnan kovil - 626126 Krishnan kovil - 626126**

**[Evaluator Name] Evaluator [Designation] Department Name**

**Kalasalingam Academy of Research and Education Krishnan kovil - 626126**

# Abstract

Modern life cannot function without email communication, but it also serves as a conduct for spam, or unsolicited emails that may include viruses or other hazardous content. A clutter-free mailbox and user protection from numerous cyber risks depend heavily on the detection and filtering of spam emails. The Natural Language Toolkit (NLTK), a Python toolkit popular for tasks involving natural language processing, is used in this research to describe a method for classifying email messages as spam or non-spam.

The suggested approach makes use of NLTK's broad text processing, feature extraction, and machine learning capabilities to create a powerful email spam classification system.

The outcomes of our tests show how well NLTK-based email spam classification works, underscoring its potential to lessen mailbox clutter and improve email security. To address the enduring issue of email spam, this strategy gives email providers, companies, and individual users a flexible and adaptable solution.

**Keywords:**

Email spam classification, Natural Language Processing (NLP), NLTK, Machine Learning, Feature Extraction, Text Preprocessing.

# Contents

1. Chapter 1: Introduction 1
   1. Natural language processing 1
   2. What is Nlp used for? 1
2. Chapter 2: System Study 3
   1. Implementation 3
3. Experimental Analysis 6
   1. Dataset
   2. Packages imported…………………………………………………………………………………………………………
   3. Model………………………………………………………………………………………………………………………………….
   4. performance
   5. python code ……………………………………………………………………………………………………………………
4. [Conclusion and Future Work](#_bookmark23)

CONTENTS CONTENTS

1. [References](#_bookmark26) 10
2. [Certification](#_bookmark27) 11

**Chapter 1**

# Chapter 1 Introduction

Email has developed into a crucial tool for communication in the current digital era, enabling the global flow of data, concepts, and business transactions. The constant influx of uninvited and potentially destructive messages known as spam is a drawback to this convenience. In addition to clogging inboxes, spam emails can present security issues such as phishing scams, malware dissemination, and fraud attempts. Therefore, it is crucial to create efficient systems for classifying and removing spam emails.

It takes more than just pattern matching or rule-based filtering to classify email spam. Given that spammers are constantly changing their strategies to avoid detection, it is essential to have a thorough comprehension of the textual content contained in emails. By providing tools for text preprocessing, feature extraction, and machine learning model creation, all within a Python context, NLTK offers an excellent option.

This research project includes several critical stages, including:

* Data Preprocessing: Raw email data is preprocessed to remove noise, including HTML tags, special characters, and irrelevant metadata. NLTK's tokenization and stemming functions are employed to break down text into meaningful units.
* Feature Extraction: The processed text is transformed into numerical features that machine learning algorithms can work with. NLTK's tools for feature extraction, such as TF-IDF (Term Frequency-Inverse Document Frequency), are used to represent the text data.
* Model Selection: Various machine learning models available in NLTK, including Naive Bayes, Support Vector Machines (SVM), and decision trees, are evaluated for their effectiveness in classifying spam emails. Cross-validation techniques are employed to assess model performance.
* Training and Testing: The selected model is trained on a labeled dataset containing both spam and non-spam emails. Performance metrics like accuracy, precision, recall, and F1-score are calculated to gauge the model's effectiveness
* Deployment: Once a satisfactory model is trained, it can be deployed in an email filtering system to automatically categorize incoming emails as spam or legitimate. NLTK's ease of integration with Python makes it suitable for real-time email classification.

In conclusion, this work emphasizes the synergy between natural language processing and machine learning approaches while presenting a thorough method for classifying email spam using NLTK. By utilizing NLTK's capabilities, we hope to provide a flexible solution that can change along with the always evolving email spam landscape, improve inbox organization and email security in the process.

## Natural language processing:

* Natural language processing (NLP) refers to the branch of artificial intelligence .
* This deals by allowing computers to understand the text and spoken words in much the same way human beings can.
* In simple - it is the application for the analysis and synthesis of natural language and speech.
* In NLP, the interaction, understanding, and response are made by a computer instead of a human.
  1. **WHAT IS NLP USED FOR?**

NLP is used to understand the structure and meaning of human language by analyzing different aspects like syntax, semantics, pragmatics, etc.

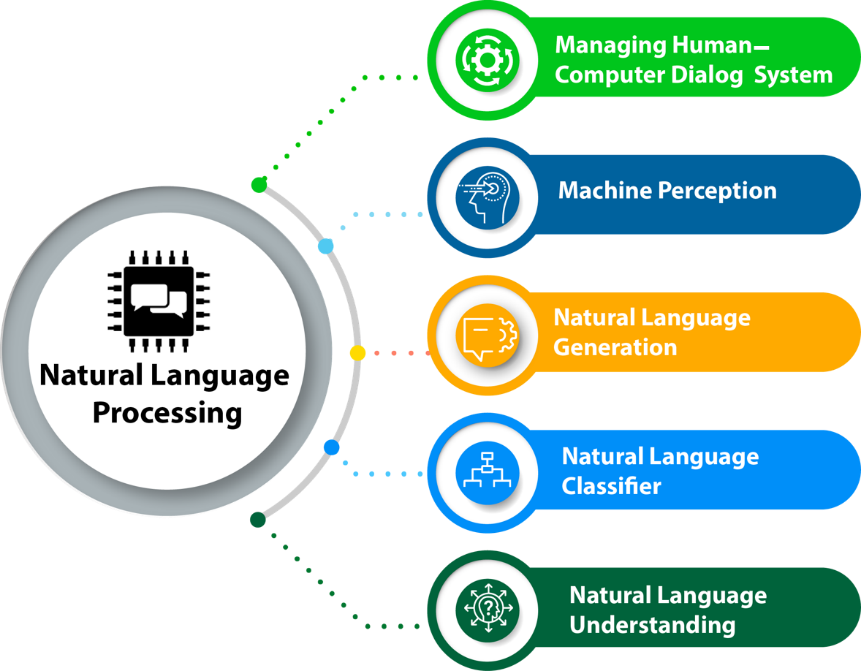
****

Figure 1.1: natural language processing

**Approaches**:

* + NLP employs various techniques, including:
    - **Rule-based approaches**: These involve predefined rules (similar to a phrasebook) that guide the computer’s understanding of language.
    - **Probabilistic approaches**: These use statistical and machine learning methods to process language data.
    - **Neural network-based approaches**: These have gained prominence recently due to their effectiveness in handling complex language tasks.

**Historical Roots**:

* + In the 1940s, Alan Turing proposed the **Turing test**, which involved automated interpretation and generation of natural language.
  + Notable systems in the 1960s:
    - **SHRDLU**: A natural language system working in restricted “blocks worlds.”
    - **ELIZA**: A simulation of a Rogerian psychotherapist, providing surprisingly human-like interactions.

**Chapter 2**

# Chapter 2 System Study

A system study of "Email Spam Classification Using NLTK: A Natural Language Processing Approach" involves a detailed examination of the components, processes, and functionalities that make up the email spam classification system based on NLTK. This study aims to provide an in-depth understanding of how the system operates, its architecture.

* + - 1. Model Selection and Training: Machine Learning Models: Detail the various machine learning models explored, such as Naive Bayes, Support Vector Machines (SVM), and decision trees, and explain their suitability for the task. Cross-Validation: Explain how cross-validation techniques are used to assess the performance of different models and select the most effective one.
      2. Performance Evaluation: Metrics: Describe the performance metrics used to evaluate the spam classification model, including accuracy, precision, recall, and F1-score.
      3. Deployment: Integration: Explain how the trained model is integrated into an email filtering system for real-time classification of incoming emails.
      4. Future Enhancements: Scalability: Suggest potential improvements for scalability, especially as email traffic grows. Advanced Techniques: Consider the integration of more advanced NLP techniques or machine learning algorithms for even better spam classification.

5. Conclusion: Summarize the findings and contributions of the system study, highlighting the effectiveness of NLTK in email spam classification.

2.2. Heading 2 Chapter 2 System Study

**2.2 IMPLEMENTATION**

Implementing an email spam classification system using NLTK involves several steps, from data preprocessing to model training and deployment. Here is a high-level outline of the implementation process:

1. Data Collection: Gather a dataset of labeled emails, where each email is tagged as either spam or legitimate (ham). You can find such datasets online or create your own.
2. Data Preprocessing: Clean the email data by removing HTML tags, special characters, and irrelevant metadata. NLTK's text processing functions can be helpful for this step. Tokenize the text into words or tokens. Apply stemming or lemmatization to reduce words to their root form.
3. Feature Extraction: Calculate TF-IDF (Term Frequency-Inverse Document Frequency) values for each word or token in the email corpus. This step transforms the text data into numerical features. Split the dataset into training and testing sets for model evaluation.
4. Model Selection and Training: Experiment with different machine learning algorithms available in NLTK, such as Naive Bayes, Support Vector Machines (SVM), and decision trees. Train multiple models on the training data and evaluate their performance using metrics like accuracy, precision, recall, and F1-score. Select the best-performing model based on evaluation results.

5. Model Deployment: Integrate the selected model into an email filtering system. This system can be a standalone application or part of an email client or server.

Configure the system to automatically classify incoming emails as spam or legitimate based on the model's predictions.

6. Real-time Classification: In the email filtering system, implement a mechanism to process incoming emails. Preprocess each incoming email in real-time, following the same steps as in the training phase (cleaning, tokenization, stemming, etc.). Use the trained model to classify the email as spam or legitimate. Move spam emails to a spam folder or take appropriate action (e.g., blocking or quarantining).

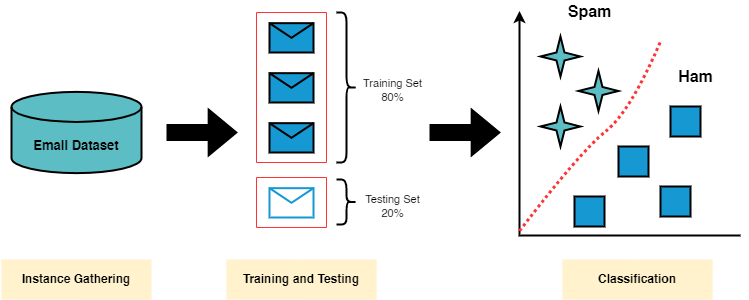
7. User Interface (Optional): If your system is user-facing, create a user-friendly interface for users to interact with the spam classification system. This interface can include settings for adjusting spam filters and monitoring classifications.

8. Monitoring and Maintenance: Regularly monitor the system's performance to ensure it continues to effectively classify spam. Retrain the model periodically with new data to adapt to evolving spam patterns. Address false positives and false negatives by fine-tuning the system.

9. Security Considerations: Implement security measures to protect against potential attacks, such as adversarial inputs or data poisoning.

10. Documentation: Document the entire implementation process, including data sources, preprocessing steps, model selection, and deployment procedures.

This documentation will be valuable for future maintenance and troubleshooting. Remember that email spam classification is an ongoing process, and the system may need adjustments over time to keep up with changing spam tactics. Regularly updating and fine-tuning the system is essential for maintaining its effectiveness in filtering spam emails.



**Chapter 3**

# Chapter 3 Experimental Analysis

**3.1.DATASET:**

This dataset is designed for the task of email spam classification. It consists of a collection of email messages that are labeled as either "spam" or "non-spam" (ham). Email spam classification is a common application of machine learning and natural language processing to identify and filter out unwanted or malicious emails.

Each row in the dataset represents an email message, with the following columns:

Email Text: The text content of the email.

Label: The label indicating whether the email is spam (1) or non-spam (0).

**3.2.PACKAGES IMPORTED:**

* Scikit-learn
* Natural language Toolkit (NLTK)
* TextBlob
* Quepy
* SpaCy
* Gensim

**3.3.MODEL:**

We use the modules “pickle” to save the trained model in the binary format.

We save the final model as “spam\_classifier.pkl” for saving the model and use in the streamlit as a binary format from given spam and non-spam emails.

**3.4.PERFORMANCE:**

* Using Naïve Bayes we get 96% of accuracy on the model.
* Using Support Vector Machine algorithm we get 95% accuracy on the model trained.

**3.5.PYTHON CODE:**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import nltk

from nltk.corpus import stopwords

import string

from nltk.tokenize import word\_tokenize

nltk.download('punkt')

emails = pd.read\_csv('/content/sample\_data/emails.csv')

emails

emails.info()

emails = emails.drop\_duplicates(keep = 'last') #remove all duplicate emails from the dataframe

emails

emails.spam.value\_counts().plot(kind='pie',

explode=[0,.1],

figsize=(6,6),

autopct='%.2f%%')

plt.title('Normal Mails vs Spam mails')

plt.legend(['Normal','Spam'])

plt.show()

emails.spam.value\_counts()

spam\_messages = emails[emails['spam']==1]['text']

notspam\_messages = emails[emails['spam']==0]['text']

spam\_words = []

notspam\_words = []

def tokenize\_spam\_words(text):

words = [w.lower() for w in word\_tokenize(text) if w.lower() not in stopwords.words('english') and w.lower().isalpha()]

spam\_words.extend(words)

def tokenize\_notspam\_words(text):

words = [w.lower() for w in word\_tokenize(text) if w.lower() not in stopwords.words('english') and w.lower().isalpha()]

notspam\_words.extend(words)

spam\_messages.apply(tokenize\_spam\_words)

print(spam\_words[:100])

import nltk

nltk.download('stopwords')

notspam\_messages.apply(tokenize\_notspam\_words)

print(notspam\_words[:100])

from nltk.stem import PorterStemmer

stemmer = PorterStemmer()

message = message.translate(str.maketrans('','',string.punctuation))

words = [stemmer.stem(w) for w in message.split() if w.lower() not in stopwords.words('english') ]

return ' '.join(words)

emails.text = emails.text.apply(cleanup\_text)

emails.head()

from sklearn.feature\_extraction.text import CountVectorizer

vect = CountVectorizer(stop\_words = 'english')

import pickle

with open('count\_vectorizer.pkl','wb') as f:

pickle.dump(vect,f)

print('done')

from sklearn.preprocessing import LabelEncoder

from sklearn.model\_selection import train\_test\_split

category = LabelEncoder()

emails.spam = category.fit\_transform(emails.spam)

emails.head()

from sklearn.model\_selection import cross\_val\_score

from sklearn.metrics import confusion\_matrix

model = GaussianNB()

model.fit(x\_train,y\_train)

y\_pred = model.predict(x\_test)

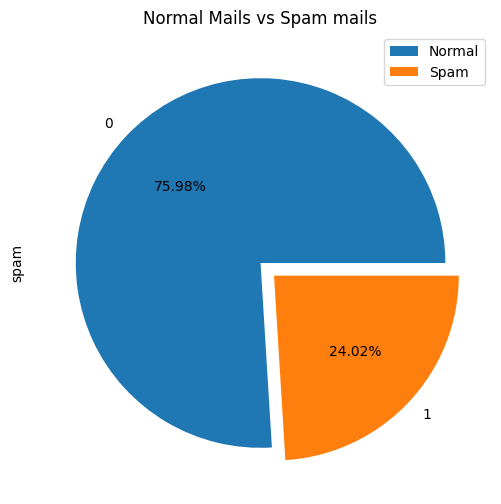
confusion\_matrix(y\_test,y\_pred)

import pickle

with open('spam\_classifier.pkl','wb') as f:

pickle.dump(model,f)

print('done')



**Chapter 4**

# Conclusion and Future Work

In the conclusion we can conclude that the spam emails and non-spam email as the messages are divided by the tokenization and preprocessed the final website using the streamlit.

If we enter the spam messages it detects the messages as spam or non-spam

**METHODOLOGY AND IMPLEMENTATION:**

Going to implement it in the website using “streamlit” using the saved models using pickle to the spam and non-spam which are converted to vector forms

**SYSTEM SPECIFICATIONS**

* This system specification is more accuracy than the previous models and
* Also performs well on the vast datasets
* Best training

**Chapter 5**

# References

1. D. K. Renuka, T. Hamsapriya, M. R. Chakkaravarthi and P. L. Surya, "Spam Classification Based on Supervised Learning Using Machine Learning Techniques", in proc. IEEE-International Conference on Process Automation Control and Computing, vol. 5, pp. 1-7, 2011.
2. M. Rathi and V. Pareek, "Spam Email Detection through Data Mining-A Comparative Performance Analysis", in International Journal of Modern Education and Computer Science, vol. 12, pp. 31-39, 2013.
3. A. Harisinghaney, A. Dixit, S. Gupta and A. Arora, "Text and image based spam email classification using KNN Naïve Bayes and Reverse DBSCAN algorithm", In 2014 International Conference on Reliability Optimization and Information Technology (ICROIT), pp. 153-155, 2014, February.
4. S. P. Teli and S. K. Biradar, "Effective Email Classification for Spam and Non-spam", in International Journal of Advanced Research in Computer and software Engineering, vol. 4, 2014.
5. M. Habib, H. Faris, M. A. Hassonah, J. F. Alqatawna, A. F. Sheta and A. Z. Ala’M, "Automatic email spam detection using genetic programming with SMOTE", In 2018 Fifth HCT Information Technology Trends (ITT), pp. 185-190, November 2018.

**Chapter 6**

# Certification

****

Figure 6.1: Certification details