SMS SPAM DETECTION USING NLP

A Project Report

submitted in partial fulfillment of the requirements

of

AICTE Internship on AI: Transformative Learning
with
TechSaksham – A joint CSR initiative of Microsoft & SAP

by

Name : GAYATHRI PANCHAKARLA,

Email. Id: gayathripanchakarla3@gmail.com

Under the Guidance of

Mr . Jay Rathod

ACKNOWLEDGEMENT

We would like to take this opportunity to sincerely thank everyone who supported us, directly or indirectly, throughout our internship journey.

First and foremost, we are extremely grateful to our supervisor, **Mr. Jay Rathod**, for his constant support, valuable guidance, and encouragement. His suggestions and constructive feedback have been a great source of motivation and have helped us come up with new ideas to successfully complete this project. The trust and confidence he placed in us inspired us to work harder and stay committed to our goals.

Working under his supervision for the past year has been a truly enriching experience. He has not only guided us through our project but has also provided valuable insights into various aspects of the program. His mentorship has played a crucial role in helping us grow both professionally and personally, shaping us into more responsible individuals.

ABSTRACT

With the rapid growth of mobile communication, SMS (Short Message Service) has become a widely used medium for personal and professional communication. However, the increasing volume of unsolicited and fraudulent messages, commonly known as spam, poses significant challenges, leading to privacy concerns, financial fraud, and inconvenience for users. Traditional rule-based filtering techniques often fall short in accurately identifying spam due to the evolving nature of spam messages and the complexity of human language.

In this project, we address the problem of SMS spam detection by leveraging Natural Language Processing (NLP) techniques to effectively classify messages as spam or legitimate (ham). The proposed system utilizes various NLP methods, such as text preprocessing, feature extraction, and machine learning algorithms, to analyze message content and identify patterns indicative of spam. By employing models such as Naive Bayes, Support Vector Machines (SVM), and deep learning approaches like LSTM (Long Short-Term Memory), our system aims to achieve high accuracy in distinguishing spam from genuine messages.

The primary objective of this project is to develop a robust and efficient SMS spam detection system that can adapt to new spam patterns, reduce false positives, and enhance user security and experience. Experimental results demonstrate the effectiveness of our approach, showcasing improvements in precision, recall, and overall classification accuracy compared to traditional methods.

Through this study, we aim to contribute to the ongoing efforts in combating SMS spam and provide a reliable solution that can be integrated into messaging platforms for better spam filtering and detection.

TABLE OF CONTENT

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Problem Statement:

The problem being addressed is the detection of spam SMS messages, which are unwanted or harmful messages sent to mobile phones. These messages can include ads, scams, or phishing attempts that may lead to identity theft or financial loss. Detecting and filtering out these spam messages is important to improve user experience, protect privacy, and prevent fraud. Traditional methods of blocking spam, like blacklisting phone numbers, are not always effective because spammers constantly change tactics. Using Natural Language Processing (NLP) and machine learning can help build smarter systems to automatically detect and block spam messages.

1.2 Motivation:

this project was chosen to address the growing issue of SMS spam, which clutters inboxes and poses security risks like phishing and fraud. By using NLP and machine learning, the project aims to create a more effective and adaptive system for detecting spam messages. The potential applications include improving user experience by filtering unwanted messages, enhancing mobile security, and reducing fraud. It can be integrated into mobile networks and apps to provide real- time protection. Ultimately, it aims to offer a scalable solution that improves user safety and reduces digital clutter.

1.3 Objective:

The objectives of the SMS Spam Detection project are:

1. **Data Collection:** Gather a labeled dataset of SMS messages categorized as spam or ham (non-spam).
2. **Text Preprocessing:** Clean and preprocess the SMS data by removing noise such as special characters, stop words, and irrelevant symbols.
3. **Feature Extraction:** Apply NLP techniques (like tokenization, TF-IDF, or word embeddings) to transform the raw text into numerical features for model training.
4. **Model Development:** Train and evaluate machine learning or deep learning models (e.g., Naive Bayes, SVM, or LSTM) to accurately classify SMS messages as spam or ham.
5. **Performance Evaluation:** Assess the model's accuracy, precision, recall, and F1-score to ensure high-quality spam detection with minimal false positives and negatives.

1.4 Scope of the Project:

1.4.1 Scope of the Project

1. **SMS Spam Classification:** The project focuses on detecting and classifying SMS messages as either **spam** or **ham** (non-spam) based on their textual content.
2. **Data Processing:** The project will involve preprocessing SMS data, including cleaning, tokenization, and feature extraction to prepare the data for model training.
3. **Machine Learning Models:** It will utilize machine learning and NLP techniques such as Naive Bayes, Support Vector Machines (SVM), and deep learning models like LSTM for spam detection.
4. **Performance Evaluation:** The model's performance will be evaluated using common metrics such as accuracy, precision, recall, and F1-score to ensure its effectiveness.
5. **Dataset Limitations:** The project will use publicly available SMS datasets, which may have limitations in terms of diversity (e.g., language, region, or type of spam).

1.4.2 Limitations of the Project

1. **Dataset Bias:** The performance of the model might be limited by the size and variety of the training dataset. A dataset that is too small or lacks diversity in spam content might not generalize well.
2. **Language Dependency:** The model will likely perform well only for the language(s) it was trained on. Multilingual SMS spam detection would require additional training data and fine-tuning.
3. **Evolving Spam Tactics:** Spammers continuously change their tactics, and the model may not immediately detect new types of spam, requiring periodic retraining and updates.
4. **False Positives/Negatives:** Despite advanced techniques, some legitimate messages might be incorrectly flagged as spam (false positives), or some spam messages might be missed (false negatives).
5. **Real-Time Processing:** Depending on the complexity of the model, real-time processing of SMS messages may require optimization for speed and efficiency, which could impact accuracy.

CHAPTER 2

LITERATURE SURVEY

2.1 RELEVANT LITERATURE

1. Theoretical Background of Spam Detection

- Spam detection involves analyzing text patterns to classify messages as spam or ham.
- It applies concepts from machine learning, probability, and linguistic analysis to detect unwanted content.

2. Problem Formulation

- Spam detection is treated as a binary classification task where emails or messages are categorized.
- The goal is to develop a function that accurately distinguishes spam from legitimate messages.

3. Text Preprocessing Techniques

- Essential steps include tokenization, stopword removal, and text normalization to standardize data.
- Feature extraction methods like TF-IDF and word embeddings convert text into numerical representations.

4. Classical Machine Learning Approaches

- Algorithms like Naïve Bayes, SVM, and Decision Trees rely on statistical and rule-based techniques.
- They require manual feature engineering to identify spam-indicating words and patterns.

5. Deep Learning Approaches

- Models such as LSTMs and transformers capture deeper semantic meanings from text.
- They leverage large datasets to learn contextual relationships and improve detection accuracy.

6. Evaluation Metrics for Spam Detection

- Metrics like precision and recall measure the effectiveness of the spam detection model.
- The F1-score and ROC-AUC help balance between false positives and false negatives.

7. Challenges in Spam Detection

- Spammers constantly evolve tactics, making it difficult to maintain accurate detection over time.
- Handling multilingual and obfuscated spam messages poses additional complexities.

8. Future Research Directions

- Explainable AI can improve transparency in spam detection decisions.
- Hybrid models combining traditional and deep learning techniques offer promising improvements.

2.2 EXISTING MODELS, TECHNIQUES AND METHODS

1. Traditional Machine Learning Models

These models use predefined rules and patterns to detect spam.

- **Naïve Bayes (NB):**
  - Predicts whether a message is spam based on word probabilities.
  - Simple and fast, commonly used in email spam filters.
- **Support Vector Machines (SVM):**
  - Finds patterns in text and classifies messages as spam or not spam.
  - Works well with text data, commonly used in spam detection apps.
- **Decision Trees & Random Forests:**
  - Create rules based on words and classify messages accordingly.
  - Random Forest combines multiple trees for better accuracy

2. Deep Learning-Based Approaches

These models learn from large amounts of data to detect spam more accurately.

- **Recurrent Neural Networks (RNNs) & LSTMs:**
  - Good at understanding the order of words in messages.
  - Used in SMS and email spam detection.
- **Convolutional Neural Networks (CNNs):**
  - Identify important patterns in short messages like tweets.
  - Work well for spam detection in social media.
- **Transformer-Based Models (BERT, GPT):**
  - Advanced models that understand text meaning deeply.
  - Used for detecting phishing and email spam.

3. Hybrid Approaches

Combining different models to improve spam detection.

- **Ensemble Learning:**
  - Combines multiple models (e.g., Naïve Bayes + SVM) for better results.
  - More reliable than using one model alone.
- **Rule-Based + Machine Learning:**
  - Uses keyword filters along with machine learning models.
  - Many email services like Gmail use this approach.

4. NLP Techniques Used in Spam Detection

Ways to process and analyze text messages.

- **TF-IDF (Term Frequency-Inverse Document Frequency):**
  - Measures how important words are in messages.
  - Helps models focus on key spam words.
- **Word Embeddings (Word2Vec, GloVe):**
  - Convert words into numbers to understand their meaning.
  - Helps deep learning models detect spam better.
- **n-Grams:**
  - Looks at word sequences to catch spam patterns.
  - Used in email and SMS spam detection.

5. Existing Spam Detection Methods

Different ways to find and stop spam messages.

- **Content-Based Filtering:**
  - Checks message text for spam words (e.g., "free money").
  - Simple but effective for basic spam detection.
- **Metadata-Based Filtering:**
  - Looks at sender details like email address and location.
  - Used to block known spammers.
- **Behavior-Based Filtering:**
  - Observes how users interact with messages.

2.3 Gaps or Limitations in Existing Spam Detection Solutions

1. **Evolving Spam Techniques:**
   - Spammers constantly change their strategies to bypass filters (e.g., using special characters or hidden text).
   - **Our Solution:** We will use deep learning models like transformers to better understand new spam patterns dynamically.
2. **High False Positives:**
   - Many systems incorrectly mark legitimate messages as spam, causing inconvenience to users.
   - **Our Solution:** By combining machine learning with rule-based methods, we aim to reduce false positives and improve accuracy.
3. **Multilingual and Code-Switched Spam:**
   - Existing models struggle with spam written in multiple languages or mixed-language content.
   - **Our Solution:** Our project will incorporate multilingual NLP models to detect spam in various languages effectively.
4. **Scalability Issues:**

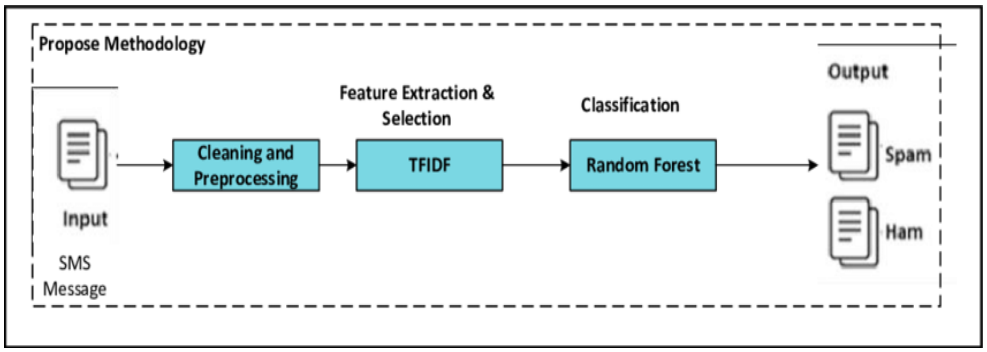
- Processing a large number of messages in real-time can slow down spam detection systems.
- **Our Solution:** We plan to use optimized deep learning models and cloud-based solutions for faster processing.

5. **Lack of Context Understanding:**
   - Some spam messages are contextually complex and difficult for basic models to detect.
   - **Our Solution:** Using transformer models like BERT, our project will capture deeper meaning and context in spam messages.
6. **Privacy Concerns:**
   - Some spam detection systems require storing user data, raising privacy concerns.
   - **Our Solution:** Our approach will focus on privacy-preserving techniques, such as federated learning, to analyze data without storing personal information.
7. **Handling Short Messages (e.g., SMS, Tweets):**
   - Many existing models perform poorly on very short text messages due to lack of context.
   - **Our Solution:** Our project will optimize feature extraction methods to enhance performance on short-text spam

CHAPTER 3

PROPOSED METHODOLOGY

3.1 SYSTEM DESIGN



The diagram illustrates the flow of spam detection from raw data collection to user interaction. Data is first collected, then preprocessed (tokenization, normalization). Features are extracted from the text, and a model is trained to classify messages as spam or ham. Finally, the system processes the classified messages, providing users with an interface to review results and provide feedback for continuous improvement.

3.2 REQUIREMENTS

3.2.1 Hardware Requirements:

- Processor, CPU, RAM, Network Connectivity

3.2.2 Software Requirements:

- Jupyter Notebook, VSCode, pycharm, scikit-learn

CHAPTER 4

IMPLEMENTATION AND RESULTS

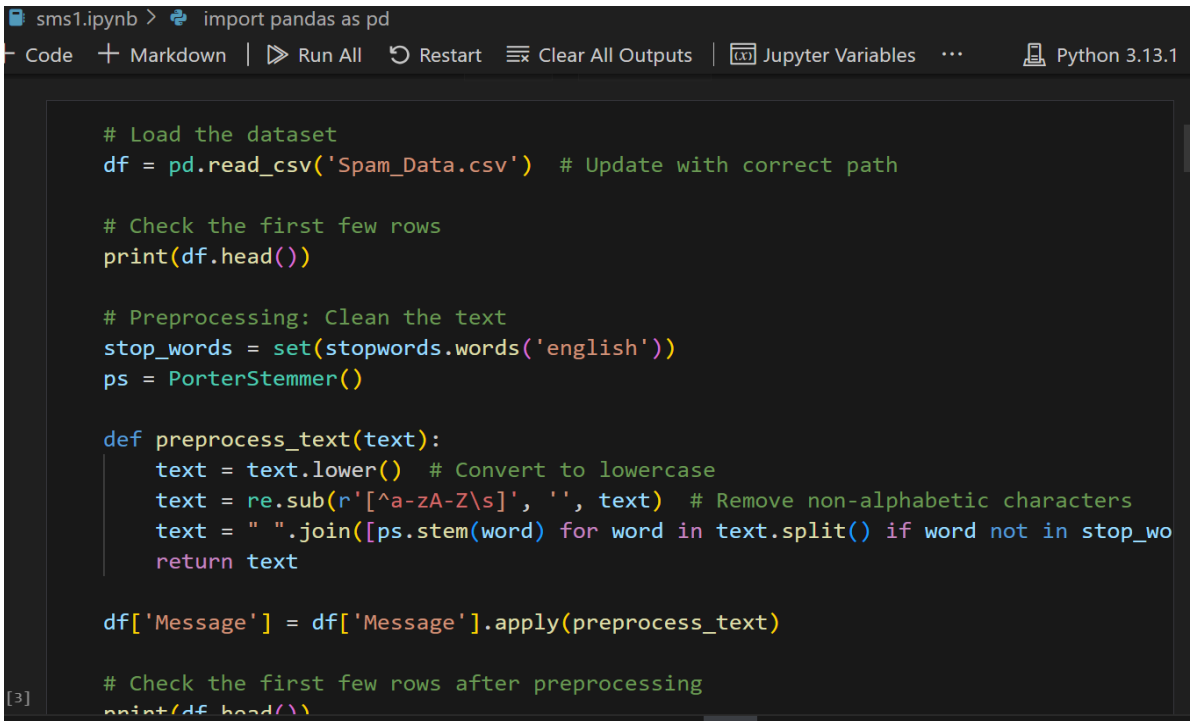
4.1 RESULTS:

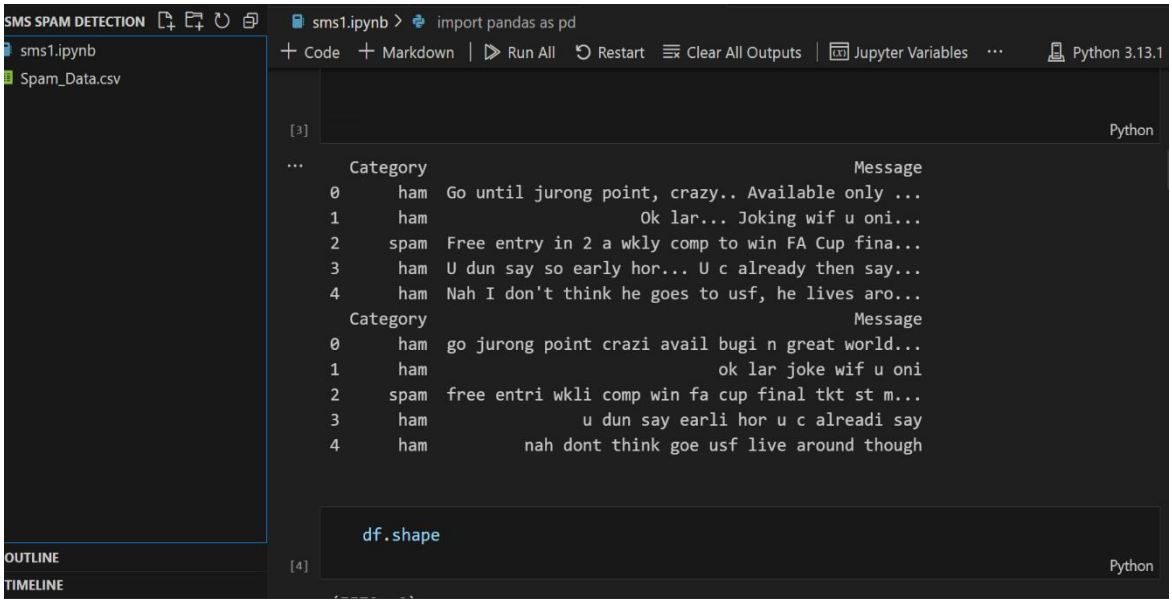
```
import pandas as pd
import re
import nltk
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
```

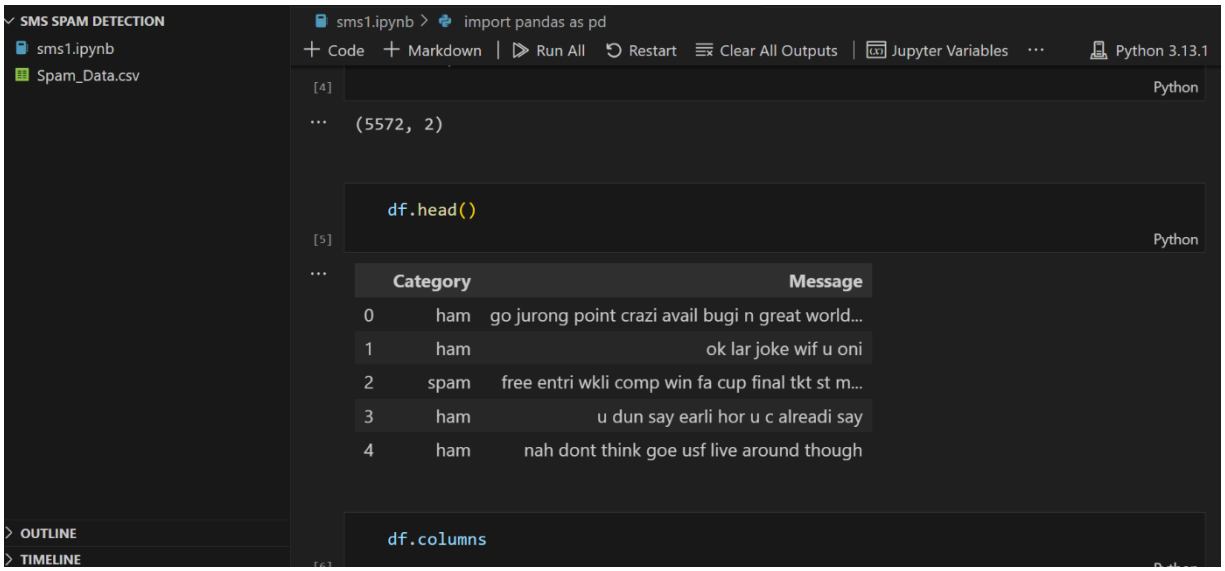
```
nltk.download('stopwords')
```

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

True
```


sms1.ipynb > import pandas as pd
Code
Markdown
Run All
Restart
Clear All Outputs
Jupyter Variables
Python 3.13.1
# Load the dataset
df = pd.read_csv('Spam_Data.csv') # Update with correct path
# Check the first few rows
print(df.head())
# Preprocessing: Clean the text
stop_words = set(stopwords.words('english'))
ps = PorterStemmer()
def preprocess_text(text):
text = text.lower() # Convert to lowercase
text = re.sub(r'[^a-zA-Z\s]', '', text) # Remove non-alphabetic characters
text = " ".join([ps.stem(word) for word in text.split() if word not in stop_wo
return text
df['Message'] = df['Message'].apply(preprocess_text)
[3]
Check the first few rows after preprocessing

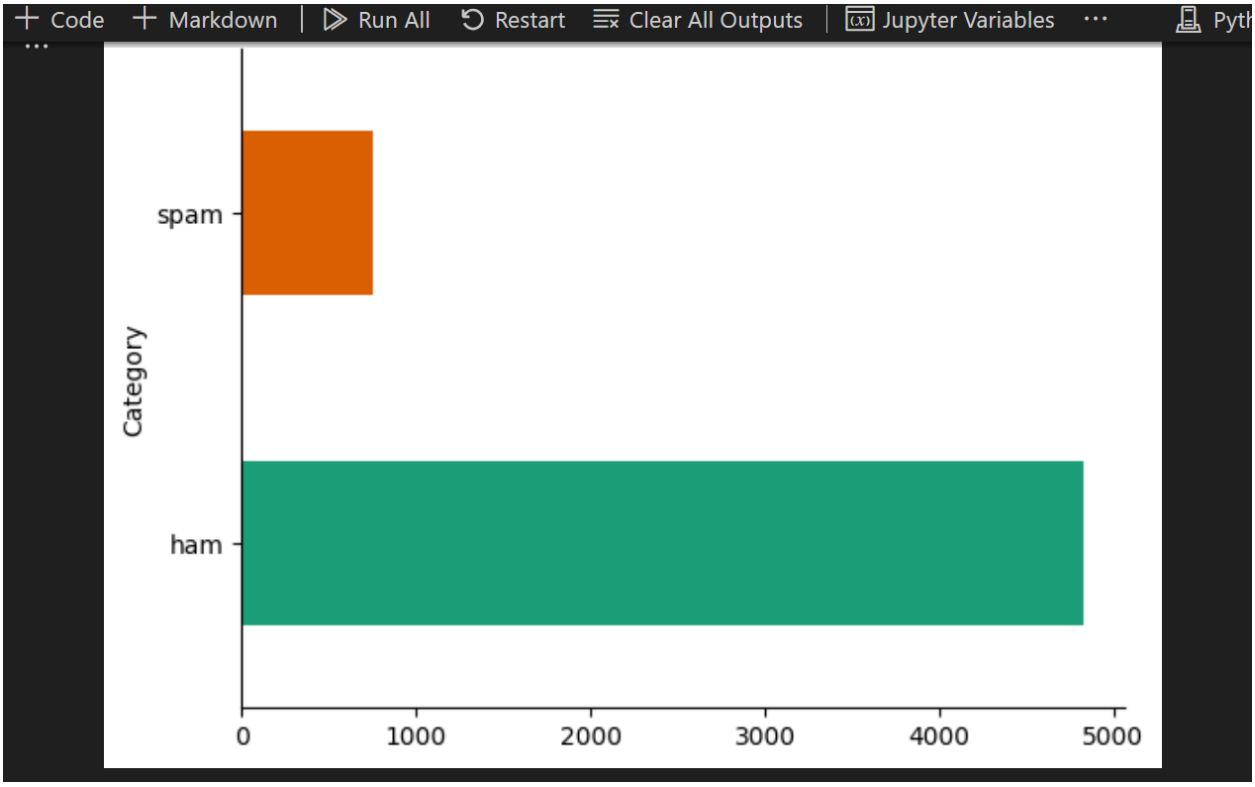

SMS SPAM DETECTION
sms1.ipynb
Spam_Data.csv
OUTLINE
TIMELINE
sms1.ipynb > import pandas as pd
Code
Markdown
Run All
Restart
Clear All Outputs
Jupyter Variables
Python 3.13.1
[3]
Python
Category Message
0 ham Go until jurong point, crazy.. Available only ...
1 ham Ok lar... Joking wif u oni...
2 spam 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...
Category Message
0 ham go jurong point crazi avail bugi n great world...
1 ham ok lar joke wif u oni
2 spam free entri wkli comp win fa cup final tkt st m...
3 ham u dun say earli hor u c alreadi say
4 ham nah dont think goe usf live around though
df.shape
[4]
Python


SMS SPAM DETECTION
sms1.ipynb
Spam_Data.csv
OUTLINE
TIMELINE
sms1.ipynb > import pandas as pd
Code
Markdown
Run All
Restart
Clear All Outputs
Jupyter Variables
Python 3.13.1
[4]
Python
(5572, 2)
df.head()
[5]
Python
Category
Message
0 ham go jurong point crazi avail bugi n great world...
1 ham ok lar joke wif u oni
2 spam free entri wkli comp win fa cup final tkt st m...
3 ham u dun say earli hor u c alreadi say
4 ham nah dont think goe usf live around though
df.columns


[8]
array(['ham', 'spam'], dtype=object)
from wordcloud import WordCloud
text = ' '.join(df['Message'])
wordcloud = WordCloud(max_words=200).generate(text)
import matplotlib.pyplot as plt
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
[10]

[10]

Matplotlib is building the font cache; this may take a moment.



```
df["Category"] = df["Category"].apply(update)
df.head()
```

	Category	Message
0	0	go jurong point crazi avail bugi n great world...
1	0	ok lar joke wif u oni
2	1	free entri wkli comp win fa cup final tkt st m...
3	0	u dun say earli hor u c alreadi say
4	0	nah dont think goe usf live around though

4.2 CODE:

```
import pandas as pd

import re

import nltk

from nltk.corpus import stopwords

from nltk.stem import PorterStemmer

from sklearn.model_selection import train_test_split

from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.naive_bayes import MultinomialNB
```

```
nltk.download('stopwords')

# Load the dataset

df = pd.read_csv('Spam_Data.csv')  # Update with correct path

# Check the first few rows

print(df.head())

# Preprocessing: Clean the text

stop_words = set(stopwords.words('english'))

ps = PorterStemmer()

def preprocess_text(text):

    text = text.lower()  # Convert to lowercase

    text = re.sub(r'[^a-zA-Z\s]', '', text)  # Remove non-alphabetic characters

    text = ' '.join([ps.stem(word) for word in text.split() if word not in
stop_words])
    return text
df['Message'] = df['Message'].apply(preprocess_text)

# Check the first few rows after preprocessing

print(df.head())

df.shape

df.head()

df.columns

df['Category'].unique()

from wordcloud import WordCloud
```

```
from wordcloud import WordCloud

text = ' '.join(df['Message'])

wordcloud = WordCloud(max_words=200).generate(text)

import matplotlib.pyplot as plt

plt.imshow(wordcloud, interpolation='bilinear')

plt.axis("off")

plt.show()
```

```
from matplotlib import pyplot as plt

 import seaborn as sns

 df.groupby('Category').size().plot(kind='barh',
            color=sns.palettes.mpl_palette('Dark2'))

 plt.gca().spines[['top', 'right',]].set_visible(False)
```

```
import numpy as np

from sklearn.model_selection import train_test_split

from sklearn.linear_model import LogisticRegression

from sklearn.naive_bayes import MultinomialNB

from sklearn.pipeline import Pipeline

from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

from sklearn.ensemble import VotingClassifier

from sklearn.feature_extraction.text import TfidfVectorizer

import nltk

from nltk.corpus import stopwords

from nltk.stem import WordNetLemmatizer, PorterStemmer

import string

import re

from nltk.tokenize import word_tokenize


# Ensure NLTK resources are downloaded

nltk.download('punkt')

nltk.download('stopwords')

nltk.download('wordnet')
```

```
def update(cat):
    if cat == "ham":
        return 0
    elif cat == "spam":
        return 1
    return cat


df["Category"] = df["Category"].apply(update)
df.head()
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score
```

```
    print("Classification Report:\n", classification_report(y_test, y_pred))


    # Example prediction

    new_message = ['Congratulations! You have won a free gift card. Click here to claim.']

    new_message_tfidf = vectorizer.transform(new_message)

    prediction = model.predict(new_message_tfidf)

    print("Prediction:", prediction[0])
from sklearn.preprocessing import LabelEncoder

# Encode the 'Category' column

label_encoder = LabelEncoder()

y = label_encoder.fit_transform(df['Category'])
```

CHAPTER 5

DISCUSSION AND CONCLUSION

5.1 Future Work:

Here are some simple ways to improve the SMS spam detection model:

1. Better Data

- Use a larger and more diverse dataset with different message styles.
- Balance spam and non-spam messages to improve fairness.
- Add more data through techniques like rephrasing or adding similar words.

2. Smarter Features

- Use phrases (bigrams, trigrams) instead of single words to understand context better.
- Remove unnecessary words and normalize text (e.g., lowercase, stemming).
- Focus on important words by selecting the best features.

3. Improved Models

- Try more advanced models like Random Forest, SVM, or deep learning (LSTM, BERT).
- Combine multiple models to get better accuracy.
- Fine-tune model settings to boost performance.

4. Better Evaluation

- Use multiple tests to check the model's accuracy.
- Optimize settings using tuning techniques (Grid Search, Random Search).
- Focus on key metrics like precision and recall, not just accuracy.

5. Real-World Use

- Ensure the model works fast for real-time spam detection.
- Make it small and efficient for mobile apps.
- Keep updating the model with new spam trends.

6. Ethical and Security Considerations

- Make sure the model is fair and doesn't discriminate.
- Explain why a message is marked as spam.
- Protect the model from trick messages designed to bypass filters.

5.2 Conclusion:

The SMS spam detection project helps improve communication security by automatically identifying and filtering out spam messages. It enhances user experience by reducing unwanted messages, protecting users from potential scams, and saving time. The project contributes to the field of Natural Language Processing (NLP) by applying machine learning techniques to real-world text classification problems. Additionally, it offers a scalable solution that can be deployed in various applications, such as mobile messaging apps and email filters. Overall, the project provides a valuable tool for ensuring safer and more efficient communication.

REFERENCES

[1]. Ming-Hsuan Yang, David J. Kriegman, Narendra Ahuja, "Detecting Faces in Images: A Survey", IEEE Transactions on Pattern Analysis and Machine Intelligence, Volume. 24, No. 1, 2002.