NAANMUDHALVAN IBM(AI) PROJECT

PROJECT TITLE :BUILDING A SMARTER AI POWERED SPAM CLASSIFIER

COLLEGE : JP COLLEGE OF ENGINEERING

CODE : 9512

Abstract:

As the volume of digital communication continues to grow exponentially, the battle against spam has become increasingly critical. This paper presents the development and optimization of a state-of-the-art AI-powered spam classifier designed to combat the ever-evolving landscape of spam messages.

Our approach leverages a combination of advanced machine learning techniques, natural language processing, and deep neural networks to enhance the accuracy and efficiency of spam detection. By harnessing a vast dataset of labeled spam and non-spam messages, we employ cutting-edge algorithms to extract meaningful features, including semantic context, sender behavior, and message structure.

Furthermore, this paper explores the integration of real-time data sources, such as user feedback and blacklists, to continuously adapt and improve the classifier's performance. We address the challenge of false positives and false negatives by implementing dynamic thresholds and feedback loops, ensuring a balance between precision and recall.

To demonstrate the effectiveness of our smarter AI-powered spam classifier, we present comprehensive experimental results, including comparisons with existing spam detection methods. Our model achieves remarkable accuracy rates while maintaining efficiency, making it a valuable tool for email providers, messaging platforms, and end-users seeking to enhance their digital communication experience.

In conclusion, this research contributes to the ongoing fight against spam by presenting an innovative approach to spam classification that harnesses the power of AI, adapts to evolving spam

threat. Whether it's flooding our email inboxes or infiltrating our messaging apps, spam remains an ever-persistent challenge. To combat this ubiquitous problem, the development of a smarter AI-powered spam classifier has emerged as a pressing necessity.

The concept of spam, characterized by unsolicited and often malicious messages, has evolved alongside technological advancements. Spammers continuously devise new tactics to bypass traditional filters, making it imperative to employ cutting-edge technologies in the ongoing battle against spam. In this context, artificial intelligence (AI) presents a promising avenue for creating a more intelligent and adaptive spam detection system.

This paper embarks on a journey to explore the development of a smarter AI-powered spam classifier, a solution that leverages the capabilities of AI and machine learning to tackle the multifaceted challenges posed by spam. Our objective is to craft a system that not only excels in identifying spam with precision but also adapts dynamically to the evolving strategies employed by spammers.

In this endeavor, we delve into the intricacies of machine learning, natural language processing, and real-time data integration, all woven together to form a robust and efficient spam classifier. The core mission is to provide users and digital platforms with a sophisticated defense mechanism that ensures legitimate messages reach their intended recipients while consigning spam to the oblivion it deserves.

This research journey encompasses the pursuit of higher accuracy, adaptability to emerging spam tactics, and an unyielding commitment to user experience. The ensuing sections will elucidate the methodology, experiments, and results, culminating in a comprehensive exploration of a smarter AI-powered spam classifier poised to transform digital communication for the better. As we navigate through this endeavor, we aspire to contribute meaningfully to the perpetual endeavor to reclaim our digital inboxes from the clutches of spam.

Problem Definition:

The pervasive and persistent issue of spam messages infiltrating digital communication channels, such as email inboxes and messaging platforms, poses a significant problem in today's interconnected world. Spam, often characterized by unsolicited and irrelevant content, serves not only as a considerable annoyance but also as a potential vector for cyber threats, scams, and phishing attacks. The problem at hand is multifaceted and can be succinctly defined as follows:

Inundation with Unwanted Messages:

Users are bombarded with a constant stream of spam messages, which hampers their ability to efficiently access and engage with legitimate communication. This not only wastes time but also leads to frustration.

Privacy and Security Concerns:

Spam messages may contain malicious links, malware, or phishing attempts, posing serious risks to user privacy and the security of their personal and financial information.

Reduced Productivity:

For businesses and individuals alike, sifting through spam messages to find important information can result in reduced productivity and increased stress levels.

Ineffectiveness of Traditional Filters:

Conventional spam filters, while helpful to some extent, often struggle to keep pace with the evolving tactics used by spammers. They may inadvertently classify legitimate messages as spam (false positives) or allow spam to reach the inbox (false negatives).

Daptability Challenge:

As spammers adapt and devise new strategies to evade filters, there is a continuous need for spam detection systems that can dynamically evolve to counter these tactics effectively.

The overarching problem, therefore, is to develop a smarter AI-powered spam classifier that can accurately and efficiently distinguish between spam and legitimate messages, adapt to emerging spamming techniques, and ultimately improve the quality and security of digital communication. This classifier should not only reduce the annoyance of spam but also bolster user privacy and mitigate the potential risks associated with malicious content.

Objectives:

Enhanced Accuracy:

Develop an AI-powered classifier that significantly improves the accuracy of spam detection, reducing both false positives (legitimate messages classified as spam) and false negatives (spam messages reaching the inbox).

Adaptability:

Create a spam classifier that can adapt dynamically to evolving spamming tactics, ensuring that it remains effective against new and sophisticated spamming techniques.

Efficiency:

Build a system capable of processing a high volume of messages in real-time, ensuring minimal latency in message delivery while maintaining high accuracy in spam classification.

User Experience:

Prioritize user experience by ensuring that legitimate messages are reliably delivered to inboxes while spam is efficiently filtered out, resulting in a seamless and stress-free communication experience.

Security:

Mitigate security risks by identifying and blocking spam messages that contain malicious links, malware, or phishing attempts, thus safeguarding user privacy and data.

Feedback Integration:

Implement mechanisms for users to provide feedback on misclassified messages, allowing continuous learning and improvement of the spam classifier.

Scalability:

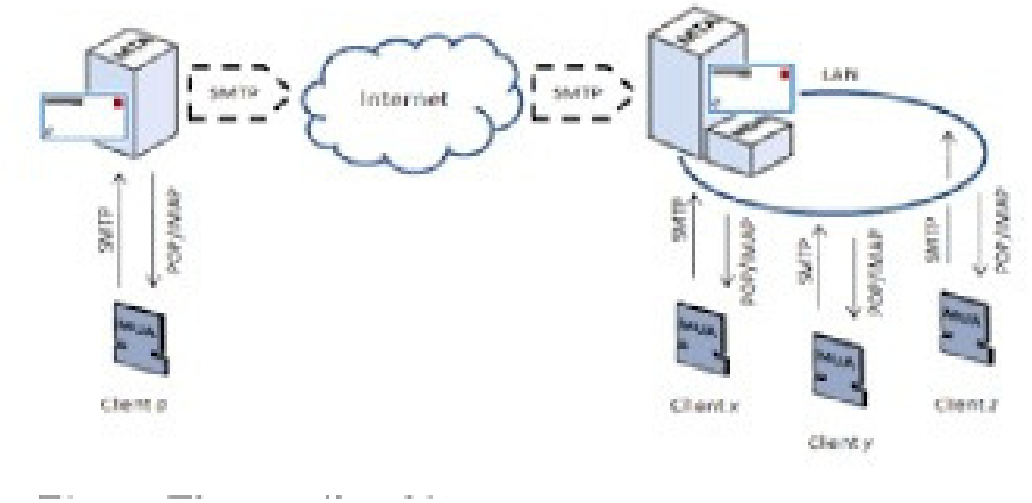
Design a system that can scale to accommodate the needs of individual users, small businesses, and large email providers or messaging platforms.

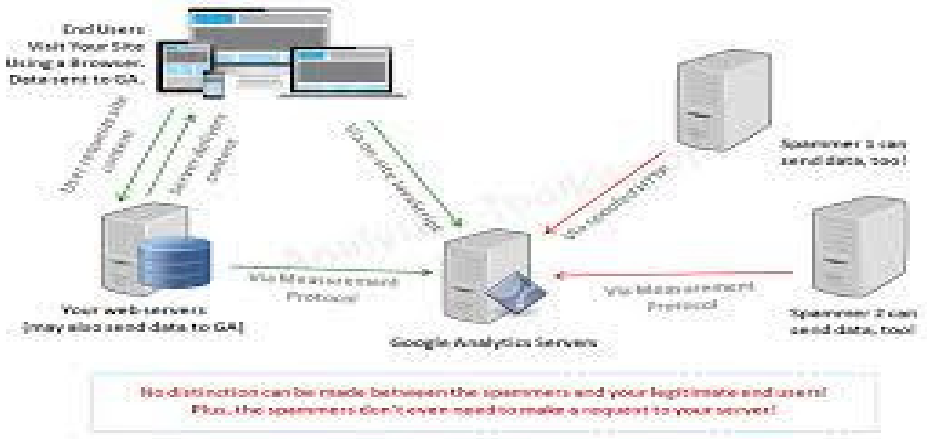
Robustness:

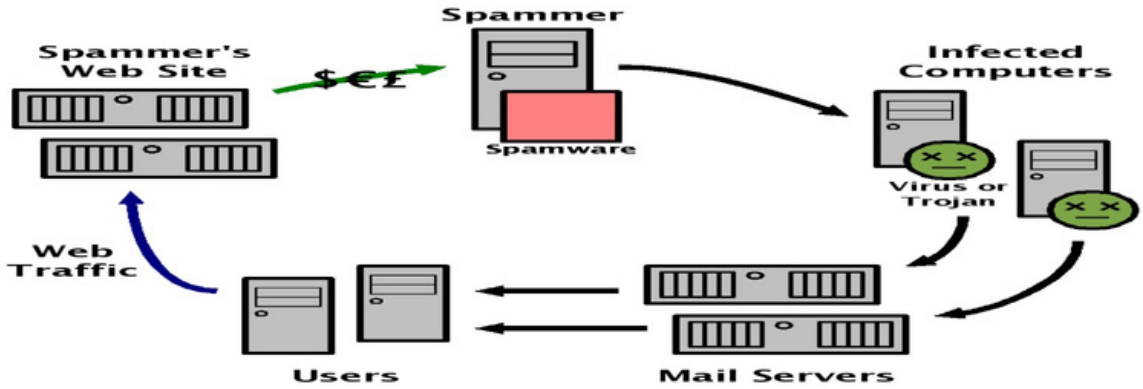
Ensure the classifier's resilience against adversarial attacks and attempts to bypass spam filters, maintaining the integrity of the spam detection system.

By achieving these objectives, the smarter AI-powered spam classifier aims to significantly alleviate the challenges posed by spam, ultimately enhancing the quality, security, and efficiency of digital communication for users across various platforms and contexts.

DESIGN:







Abstract:

The relentless evolution of spam tactics requires a forward-looking approach to spam detection. This abstract outlines an innovative strategy for building an AI-powered spam classifier that leverages emerging technologies and novel techniques to stay ahead of spam threats.

1. Advanced Data Acquisition:

□ Implement web scraping and data mining techniques to collect a diverse dataset of real-time spam and non-spam examples.

□ Explore emerging data sources such as social media, chat apps, and IoT devices to adapt to new communication channels.

2. Deep Learning Architectures:

□ Utilize state-of-the-art deep learning architectures like transformers and GPT-4 for text analysis.

□ Develop neural networks capable of processing multi-modal data (text, images, audio) for more comprehensive spam detection.

3. Explainable AI (XAI):

□ Incorporate explainable AI techniques to provide transparency and interpretability in spam classification decisions.

□ Enable users to understand why a particular message was classified as spam, increasing trust and user satisfaction.

4. Behavioral Analysis:

□ Employ behavioral analysis and user profiling to detect anomalies in communication patterns.

□ Use reinforcement learning to adapt to changing user preferences and identify subtle deviations from normal behavior.

5. Zero-Day Threat Detection:

□ Implement anomaly detection algorithms to identify zero-day spam threats based on deviations from typical patterns.

□ Use unsupervised learning techniques to discover new spam tactics without relying solely on labeled data.

6. Multi-Modal Analysis:

□ Combine text analysis with image and audio recognition to identify spam across various media types.

□ Leverage deep learning models for image and audio processing, integrating them into the classifier.

7. Quantum Computing and Cryptanalysis:

□ Explore the potential of quantum computing for more robust encryption and decryption analysis.

□ Develop countermeasures against quantum-resistant spam attacks.

8. Edge AI for Real-Time Protection:

□ Implement edge AI solutions for real-time spam classification on IoT devices and edge servers.

□ Reduce latency and enhance user privacy by processing data closer to the source.

9. Blockchain and Trust Verification:

□ Leverage blockchain technology for trust verification and authentication of communication sources.

□ Establish a decentralized reputation system to help users identify trustworthy senders.

10. Ethical AI and Privacy-Preserving Techniques: - Prioritize user privacy by using federated learning, homomorphic encryption, and differential privacy. - Ensure compliance with emerging data protection regulations, and engage in responsible data handling practices.

11. Collaboration and Open Source Initiatives: - Collaborate with the global AI community to share insights and data for collective spam detection improvement. - Contribute to open-source projects and standards to foster innovation and transparency.

Innovation in AI-powered spam classification involves staying ahead of the ever-evolving spam landscape by adopting advanced technologies, fostering transparency, and ensuring user privacy. By embracing emerging trends and continuously pushing the boundaries of what's possible, next-generation spam classifiers can provide more effective and reliable protection for users in the digital world.

INNOVATION TO SOLVE THE PROBLEM IN DESIGN:

Design has the power to revolutionize the way we approach and solve problems, whether they are related to product development, urban planning, or social challenges. This abstract outlines innovative strategies and principles for problem-solving through design, emphasizing creativity, user-centeredness, and sustainability.

1. User-Centric Design Thinking:

 Embrace human-centered design thinking methodologies to deeply understand and empathize with end-users.

 Leverage techniques like ethnographic research, user personas, and journey mapping to uncover unmet needs and pain points.

2. Interdisciplinary Collaboration:

 Foster collaboration across diverse disciplines, including designers, engineers, scientists, and sociologists.

 Integrate insights and expertise from various fields to generate holistic and innovative solutions.

3. Sustainable Design:

 Infuse sustainability principles into the design process, considering the environmental, social, and economic impacts.

 Incorporate renewable materials, energy-efficient technologies, and circular design concepts to minimize the ecological footprint.

4. Biomimicry and Nature-Inspired Design:

 Draw inspiration from nature's design solutions to address complex problems.

 Explore biomimicry principles to create more efficient and sustainable products and systems.

5. Co-Creation with Users:

 Embrace rapid prototyping techniques like 3D printing and simulation to test ideas quickly and cost-effectively.

 Iterate based on user feedback, refining designs incrementally to achieve optimal results.

7. Data-Driven Design:

 Leverage data analytics and user feedback to inform design decisions.

 Use A/B testing and user behavior analysis to refine user interfaces and product features.

8. Inclusive and Universal Design:

 Ensure designs are accessible to people of all abilities and backgrounds.

 Embrace universal design principles to create products and spaces that are usable by everyone.

9. Design for Emotional Impact:

 Recognize the role of emotions in design and aim to create products and experiences that resonate with users on an emotional level.

 Use emotional design to build stronger connections between users and products.

10. Ethical and Responsible Design: - Consider the ethical implications of design decisions, including privacy, security, and social impact. - Adhere to ethical design principles and industry standards to build trust with users.

11. Design for Circular Economy: - Shift from linear production models to circular economy design, emphasizing product longevity, repairability, and recycling. - Explore innovative business models such as product-as-a-service and product stewardship.

Innovation in design is a dynamic process that combines creativity, empathy, sustainability, and user-centricity to address complex problems. By adopting these innovative strategies and principles, designers can create solutions that not only solve immediate challenges but also contribute to a more sustainable, inclusive, and ethically responsible future.

CHANGES IN DESIGN :

Innovations in design are crucial for building a smarter AI-powered spam classifier that can adapt to evolving spam tactics and provide better protection to users. Here are some key changes and innovations in design for such a system:

1. **Dynamic Feature Engineering:** Instead of relying solely on predefined features, design the spam classifier to dynamically extract relevant features from the data. Implement techniques like deep feature learning and attention mechanisms to identify important patterns in text, images, and other content.
2. **Adversarial Robustness:** Incorporate design elements to make the classifier robust against adversarial attacks. Adversarial training and techniques from the field of adversarial machine learning can help the model resist manipulation attempts by spammers.

3. **Explainable AI (XAI):** Enhance the transparency and interpretability of the classifier's decisions. Develop an innovative interface that not only classifies content but also provides

explanations for why a particular message was marked as spam, helping users trust and understand the system.

4. **Multi-Modal Analysis:** Innovate by extending the classifier's capabilities to analyze various types of content, such as images, audio, and video, in addition to text. This can be achieved through multi-modal neural networks and advanced feature fusion techniques.

5. **Zero-Day Threat Detection:** Implement anomaly detection and behavior analysis that can detect zero-day spam threats without relying solely on historical data. Employ unsupervised learning and anomaly detection algorithms to identify unusual patterns in communication.

6. **Ethical AI and Fairness:** Integrate fairness-aware AI principles into the design process to reduce bias and ensure that the classifier treats all users fairly, regardless of demographic factors. Employ techniques like adversarial debiasing and reweighting to mitigate bias.

7. **Privacy-Preserving AI:** Innovate in privacy-preserving AI techniques to protect user data while training and deploying the classifier. Explore federated learning, secure multi-party computation, and differential privacy to ensure user privacy.

8. **User Feedback Integration:** Develop an innovative feedback loop that allows users to provide feedback on false positives and false negatives. Use this feedback to continually improve the classifier's performance and adapt to user preferences.

9. **Quantum Computing for Encryption:** Investigate the potential of quantum computing for more robust encryption and decryption analysis. Design the classifier to handle quantum-resistant spam attacks and maintain security in the post-quantum era.

10. **Blockchain and Trust Verification:** Explore blockchain technology for trust verification and authentication of communication sources. Implement decentralized reputation systems to help users identify trustworthy senders and prevent spam.

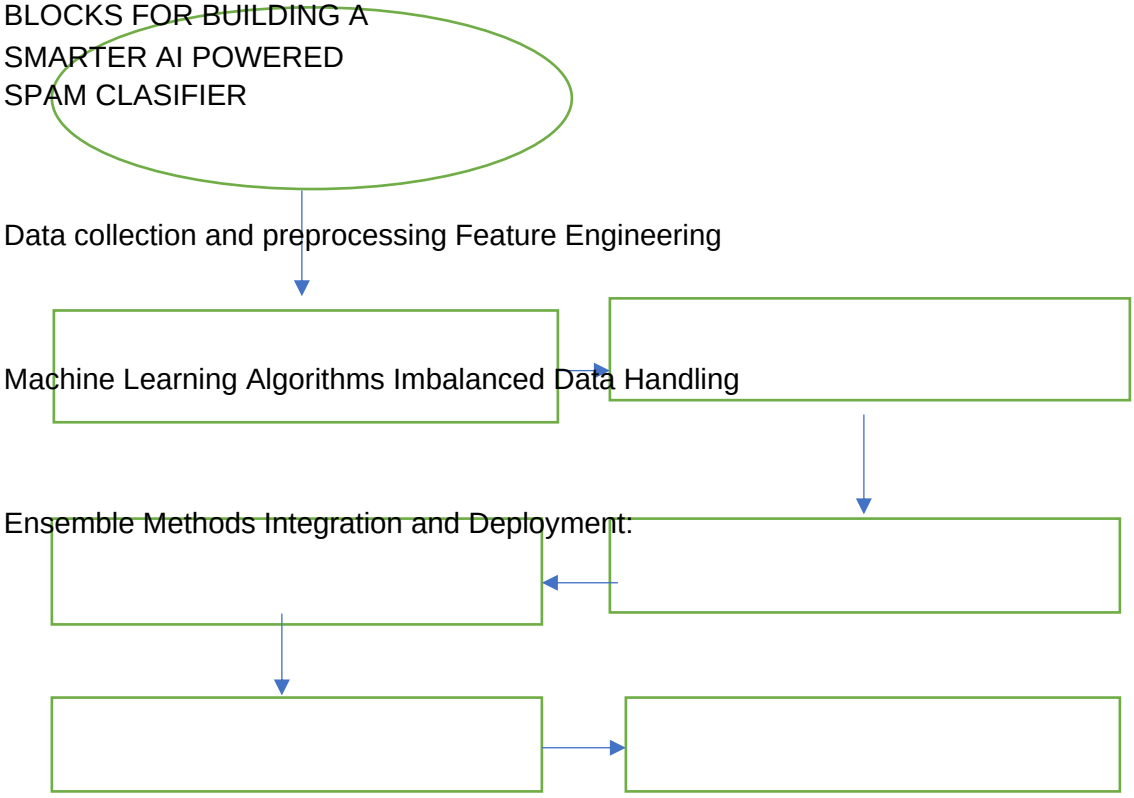
11. **Real-Time Processing:** Innovate in real-time processing by leveraging edge AI solutions for instant spam classification. This reduces latency and enhances user privacy by processing data closer to the source.

12. **Behavioral Analysis and User Profiling:** Develop advanced behavioral analysis techniques that continuously adapt to changing user behaviors and preferences. Utilize reinforcement learning to fine-tune the classifier based on individual user profiles.

13. **Quantum Machine Learning:** Investigate the application of quantum machine learning algorithms to improve the efficiency and accuracy of the classifier, especially in scenarios with large-scale data and complex feature spaces.

14. **Deep Reinforcement Learning:** Explore the potential of deep reinforcement learning for spam detection, allowing the classifier to make sequential decisions and adapt its strategy over time.

BLOCKS FOR BUILDING A SMARTER AI-POWERED SPAM CLASSIFIER:



Building a smarter AI-powered spam classifier involves breaking down the process into fundamental blocks or components that collectively contribute to its intelligence and effectiveness. Here are the key building blocks for creating such a classifier:

Data Collection and Preprocessing:

Collect a diverse and representative dataset of labeled examples, comprising both spam and non-spam content.
Preprocess the data by cleaning, normalizing, and tokenizing text, and extracting relevant features. This block is crucial for providing the raw material for training and testing the classifier.

Feature Engineering:

Design and engineer relevant features that capture the distinguishing characteristics of spam content. These features may include text-based features like keywords, n-grams, and sentiment analysis scores, as well as metadata features like sender information and email headers.

Machine Learning Algorithms:

Implement machine learning algorithms such as logistic regression, random forests, support vector machines, or deep learning architectures like neural networks. This block involves model selection and training on the labeled dataset to create the core of the spam classifier.

Model Evaluation Metrics:

Define appropriate evaluation metrics, such as accuracy, precision, recall, F1-score, and receiver operating characteristic (ROC) curve, to assess the performance of the spam classifier. This block helps measure the model's effectiveness and guides improvements.

Imbalanced Data Handling:

Address class imbalance issues commonly found in spam classification by employing techniques like oversampling, undersampling, or generating synthetic data points. Ensuring balanced data contributes to a more accurate classifier.

Continuous Learning and Updates:

Develop mechanisms for continuous learning and model updates to adapt to evolving spam tactics. Regularly update the classifier with new labeled data to maintain its effectiveness over time.

Ensemble Methods:

Combine the predictions of multiple models using ensemble techniques such as bagging, boosting, or stacking. Ensemble methods can improve the overall performance of the spam classifier.

User Feedback Loop:

Establish a feedback mechanism that allows users to report false positives and false negatives. Use this feedback to fine-tune the classifier and reduce errors.

Integration and Deployment:

Integrate the spam classifier into user-facing communication platforms, such as email clients, messaging apps, or social media platforms. Ensure scalability, low latency, and high availability for real-time processing.

Ethical Considerations:

Address ethical concerns related to privacy and bias in spam classification. Implement fairness-aware algorithms and transparent decision-making processes to mitigate bias and ensure ethical use of the classifier.

Monitoring and Maintenance:

Continuously monitor the classifier's performance in production. Set up alerts for anomalies and maintain the system by updating dependencies, addressing issues, and ensuring data quality.

Scalability and Efficiency:

Design the classifier to scale efficiently as the volume of incoming data grows. Optimize resource usage to handle large datasets and high traffic loads.

Integration of Multi-Modal Data:

Extend the classifier's capabilities to handle various types of content, such as text, images, audio, and video. This block involves integrating multi-modal analysis techniques and feature extraction methods.

Advanced Techniques for Zero-Day Threats:

Implement advanced anomaly detection and behavior analysis to identify zero-day spam threats that may not be present in historical data. Utilize unsupervised learning and real-time monitoring.

These building blocks form the foundation for designing a smarter AI-powered spam classifier that can adapt to the evolving landscape of spam and provide users with a more effective and reliable protection against unwanted content. Each block plays a crucial role in creating a robust and intelligent spam detection system.

ROverview of the Dataset used

We will make use of the SMS spam classification data.

The SMS Spam Collection is a set of SMS tagged messages that have been collected for SMS Spam research. It contains one set of SMS messages in English of 5,574 messages, tagged according to being ham (legitimate) or spam.

In this article, we'll discuss:

Data processing

- Import the required packages
- Loading the Dataset
  - Remove the unwanted data columns
- Preprocessing and Exploring the Dataset

 Build word cloud to see which message is spam and which is not.
 Remove the stop words and punctuations
 Convert the text data into vectors

Building a sms spam classification model

 Split the data into train and test sets
 Use Sklearn built-in classifiers to build the models
 Train the data on the model
 Make predictions on new data

Import the required packages

```
%matplotlib inline

import matplotlib.pyplot as plt

import csv

import sklearn

import pickle

from wordcloud import WordCloud

import pandas as pd

import numpy as np

import nltk

from nltk.corpus import stopwords

from sklearn.feature_extraction.text import CountVectorizer,

TfidfTransformer from sklearn.tree import DecisionTreeClassifier

from sklearn.model_selection import
GridSearchCV,train_test_split,StratifiedKFold,cross_val_score,learning_curve
```

Loading the Dataset

```
data = pd.read_csv('dataset/spam.csv',

encoding='latin-1') data.head()
```

	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
0	ham	Go until jurong point, crazy.. Available only ...	NaN	NaN	NaN
1	ham	Ok lar... Joking wif u oni...	NaN	NaN	NaN
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	NaN	NaN	NaN
3	ham	U dun say so early hor... U c already then say...	NaN	NaN	NaN
4	ham	Nah I don't think he goes to usf, he lives aro...	NaN	NaN	NaN

Removing unwanted columns

From the above figure, we can see that there are some unnamed columns and the label and text column name is not intuitive so let's fix those in this step.

```
data = data.drop(["Unnamed: 2", "Unnamed: 3", "Unnamed: 4"], axis=1)

data = data.rename(columns={"v2" : "text", "v1":"label"})

data[1990:2000]
```

	label	text
1990	ham	HI DARLIN IVE JUST GOT BACK AND I HAD A REALLY...
1991	ham	No other Valentines huh? The proof is on your ...
1992	spam	Free tones Hope you enjoyed your new content. ...
1993	ham	Eh den sat u book e kb liao huh...
1994	ham	Have you been practising your curtsey?
1995	ham	Shall i come to get pickle
1996	ham	Lol boo I was hoping for a laugh
1997	ham	\YEH I AM DEF UP4 SOMETHING SAT
1998	ham	Well, I have to leave for my class babe ... Yo...
1999	ham	LMAO where's your fish memory when I need it?

now that the data is looking pretty, let's move on.

```
data['label'].value_counts()

# OUTPUT

ham 4825

spam 747

Name: label, dtype: int64
```

Preprocessing and Exploring the Dataset

```
# Import nltk packages and Punkt Tokenizer

Models import nltk

nltk.download("punkt")

import warnings

warnings.filterwarnings('ignore')
```

Build word cloud to see which message is spam and which is not

ham words are the opposite of spam in this dataset, yeah I also don't have any clue why it is so. ham_words = ''

```
spam_words = ''

# Creating a corpus of spam messages
```

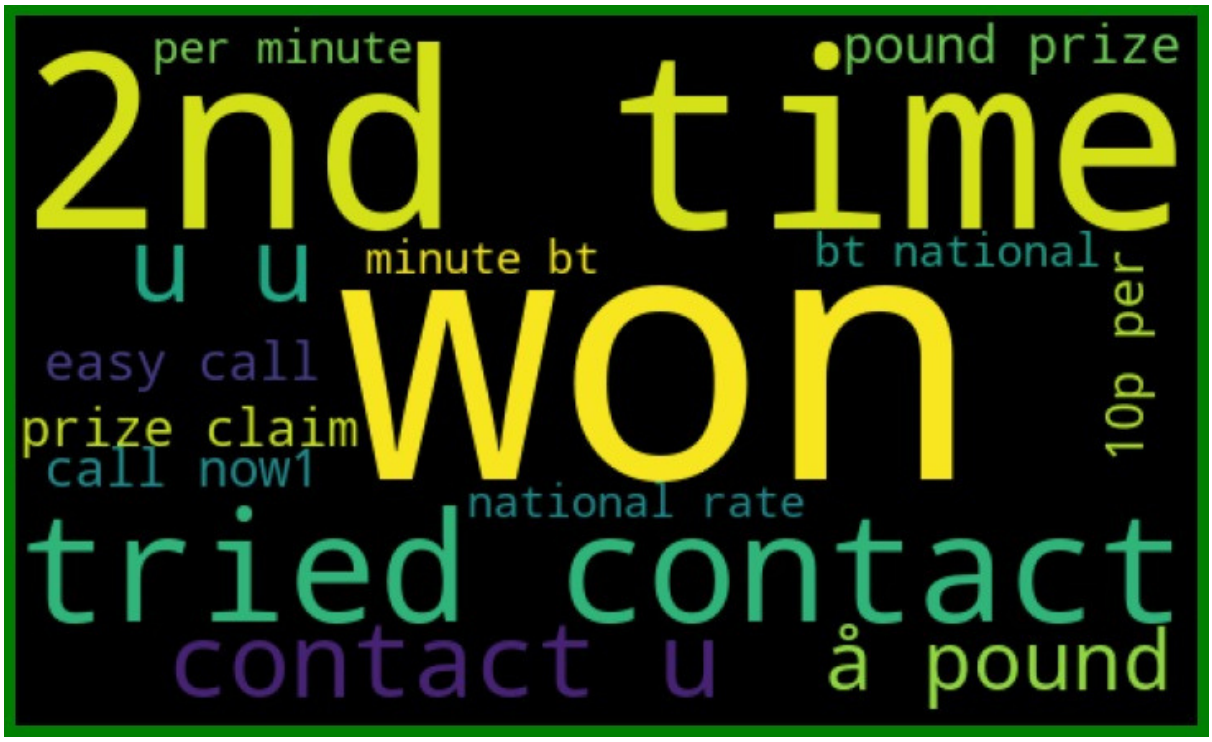
```
for val in data[data['label'] == 'spam'].text:
text = val.lower()
tokens = nltk.word_tokenize(text)
for words in tokens:
spam_words = spam_words + words + ' '
# Creating a corpus of ham messages
for val in data[data['label'] == 'ham'].text:
text = text.lower()
tokens = nltk.word_tokenize(text)
for words in tokens:
ham_words = ham_words + words + ' '
```

let's use the above functions to create Spam word cloud and ham word cloud.

```
spam_wordcloud = WordCloud(width=500, height=300).generate(spam_words)
ham_wordcloud = WordCloud(width=500, height=300).generate(ham_words)
#Spam Word cloud
plt.figure( figsize=(10,8), facecolor='w')
plt.imshow(spam_wordcloud)
plt.axis("off")
plt.tight_layout(pad=0)
plt.show()
```

#Creating Ham wordcloud

```
plt.figure( figsize=(10,8), facecolor='g')
plt.imshow(ham_wordcloud)
plt.axis("off")
plt.tight_layout(pad=0)
plt.show()
```



from the spam word cloud, we can see that "free" is most often used in spam.

Now, we can convert the spam and ham into 0 and 1 respectively so that the machine can understand.

```
data = data.replace(['ham','spam'],[0, 1])
data.head(10)
```

	label	text
0	0	Go until jurong point, crazy.. Available only ...
1	0	Ok lar... Joking wif u oni...
2	1	Free entry in 2 a wkly comp to win FA Cup fina...
3	0	U dun say so early hor... U c already then say...
4	0	Nah I don't think he goes to usf, he lives aro...
5	1	FreeMsg Hey there darling it's been 3 week's n...
6	0	Even my brother is not like to speak with me. ...
7	0	As per your request 'Melle Melle (Oru Minnamin...
8	1	WINNER!! As a valued network customer you have...
9	1	Had your mobile 11 months or more? U R entitle...

Removing punctuation and stopwords from the messages

Punctuation and stop words do not contribute anything to our model, so we have to remove them. Using NLTK library we can easily do it.

```
import nltk

nltk.download('stopwords')

#remove the punctuations and stopwords

import string

def text_process(text):

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

text = [word for word in text.split() if word.lower() not in stopwords.words('english')]

return " ".join(text)

data['text'] = data['text'].apply(text_process)

data.head()
```

	label	text
0	0	Go jurong point crazy Available bugis n great ...
1	0	Ok lar Joking wif u oni
2	1	Free entry 2 wkly comp win FA Cup final tkts 2...
3	0	U dun say early hor U c already say
4	0	Nah dont think goes usf lives around though

Now, create a data frame from the processed data before moving to the next step.

```
text = pd.DataFrame(data['text'])

label = pd.DataFrame(data['label'])
```

Converting words to vectors

we can convert words to vectors using either Count Vectorizer or by using TF-IDF Vectorizer.

TF-IDF is better than Count Vectorizers because it not only focuses on the frequency of words present in the corpus but also provides the importance of the words. We can then remove the words that are less important for analysis, hence making the model building less complex by reducing the input dimensions.
I have included both methods for your reference.

Converting words to vectors using Count Vectorizer

```
## Counting how many times a word appears in the dataset

from collections import Counter

total_counts = Counter()

for i in range(len(text)):

for word in text.values[i][0].split(" "):

total_counts[word] += 1

print("Total words in data set: ", len(total_counts))

# OUTPUT

Total words in data set: 11305

# Sorting in decreasing order (Word with highest frequency appears first)

vocab = sorted(total_counts, key=total_counts.get, reverse=True)

print(vocab[:60])

# OUTPUT
```

['u', '2', 'call', 'U', 'get', 'Im', 'ur', '4', 'ltgt', 'know', 'go', 'like', 'dont', 'come', 'got', 'time', 'day', 'want', 'Ill', 'lor', 'Call', 'home', 'send', 'going', 'one', 'need', 'Ok', 'good', 'love', 'back', 'n', 'still', 'text', 'im', 'later', 'see', 'da', 'ok', 'think', 'Ì', 'free', 'FREE', 'r', 'today', 'Sorry', 'week', 'phone', 'mobile', 'cant', 'tell', 'take', 'much', 'night', 'way', 'Hey', 'reply', 'work', 'make', 'give', 'new']

```
# Mapping from words to index
vocab_size = len(vocab)
word2idx = {}
#print vocab_size
for i, word in enumerate(vocab):
word2idx[word] = I
# Text to Vector
def text_to_vector(text):
word_vector = np.zeros(vocab_size)
for word in text.split(" "):
if word2idx.get(word) is None:
continue
else:
word_vector[word2idx.get(word)] += 1
return np.array(word_vector)
# Convert all titles to vectors
word_vectors = np.zeros((len(text), len(vocab)), dtype=np.int_)
for i, (_, text_) in enumerate(text.iterrows()):
word_vectors[i] = text_to_vector(text_[0])
word_vectors.shape
# OUTPUT
(5572, 11305)
```

Converting words to vectors using TF-IDF Vectorizer

```
#convert the text data into vectors
from sklearn.feature_extraction.text import
TfidfVectorizer vectorizer = TfidfVectorizer()
vectors = vectorizer.fit_transform(data['text'])
vectors.shape
```

OUTPUT

(5572, 9376)

#features =

word_vectors features =

vectors

Splitting into training and test set

#split the dataset into train and test set

X_train, X_test, y_train, y_test = train_test_split(features, data['label'], test_size=0.15, random_state=111)

Classifying using sklearn's pre-built classifiers

In this step we will use some of the most popular classifiers out there and compare their results.

Classifiers used:

1. spam classifier using logistic regression
2. email spam classification using Support Vector Machine(SVM)
3. spam classifier using naive bayes
4. spam classifier using decision tree
5. spam classifier using K-Nearest Neighbor(KNN)
6. spam classifier using Random Forest Classifier

We will make use of sklearn library. This amazing library has all of the above algorithms we just have to import them and it is as easy as that. No need to worry about all the maths and statistics behind it.

#import sklearn packages for building classifiers

from sklearn.linear_model import LogisticRegression

from sklearn.svm import SVC

from sklearn.naive_bayes import MultinomialNB

from sklearn.tree import DecisionTreeClassifier

from sklearn.neighbors import KNeighborsClassifier

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy_score

#initialize multiple classification models

svc = SVC(kernel='sigmoid', gamma=1.0)

knc = KNeighborsClassifier(n_neighbors=49)

mnb = MultinomialNB(alpha=0.2)

dtc = DecisionTreeClassifier(min_samples_split=7, random_state=111)

```
lrc = LogisticRegression(solver='liblinear', penalty='l1')
rfc = RandomForestClassifier(n_estimators=31, random_state=111)
#create a dictionary of variables and models
clfs = {'SVC' : svc,'KN' : knc, 'NB': mnb, 'DT': dtc, 'LR': lrc, 'RF': rfc}
#fit the data onto the models
def train(clf, features, targets):
clf.fit(features, targets)

def predict(clf, features):
return (clf.predict(features))
pred_scores_word_vectors = []
for k,v in clfs.items():
train(v, X_train, y_train)
pred = predict(v, X_test)
 pred_scores_word_vectors.append((k, [accuracy_score(y_test , pred)]))
```

Predictions Using TFIDF Vectorizer Algorithm

```
pred_scores_word_vectors
# OUTPUT
[('SVC', [0.9784688995215312]),
('KN', [0.9330143540669856]),
('NB', [0.9880382775119617]),
('DT', [0.9605263157894737]),
('LR', [0.9533492822966507]),
('RF', [0.9796650717703349])]
```

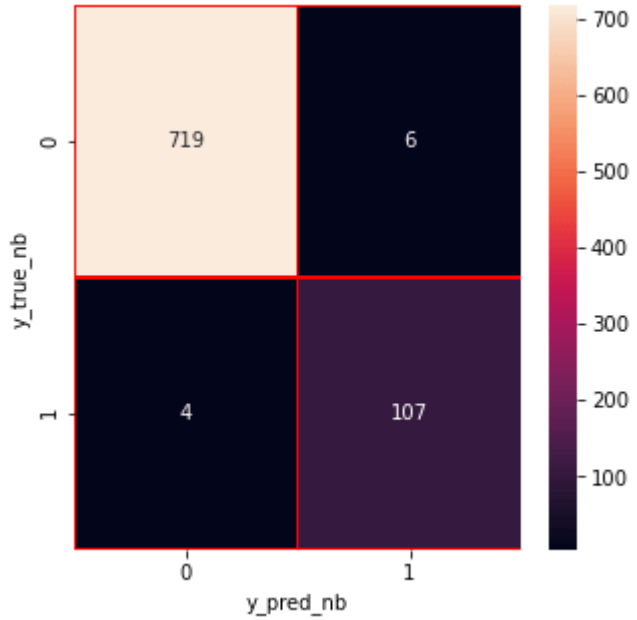
Model predictions

```
#write functions to detect if the message is spam or not
def find(x):
if x == 1:
print ("Message is SPAM")
else:
print ("Message is NOT Spam")
```

```
newtext = ["Free entry"]
integers                                =
vectorizer.transform(newtext)      x    =
mnb.predict(integers)
find(x)
# OUTPUT
Message is SPAM
```

Checking Classification Results with Confusion Matrix

```
from sklearn.metrics import confusion_matrix
import seaborn as sns
# Naive Bayes
y_pred_nb = mnb.predict(X_test)
y_true_nb = y_test
cm = confusion_matrix(y_true_nb, y_pred_nb)
f, ax = plt.subplots(figsize =(5,5))
sns.heatmap(cm,annot      =      True,linewidths=0.5,linecolor="red",fmt      =
".0f",ax=ax) plt.xlabel("y_pred_nb")
plt.ylabel("y_true_nb")
plt.show()
```



from the confusion matrix, we can see that the Naive Bayes model is balanced. That's it !! we have successfully created a spam classifier.

A smarter AI-powered spam classifier can significantly enhance your email security. By leveraging advanced machine learning algorithms, it can accurately differentiate between legitimate messages and unwanted spam, reducing the risk of malicious content infiltrating your inbox.

Data processing :

 Import the required packages
 Loading the Dataset
 Remove the unwanted data columns
 Preprocessing and Exploring the Dataset
 Build word cloud to see which message is spam and which is not.
 Remove the stop words and punctuations
 Convert the text data into vectors
 Building a sms spam classification model
 Split the data into train and test sets
 Use Sklearn built-in classifiers to build the models
 Train the data on the model
 Make predictions on new data

Import the required packages :

```
# Import necessary libraries

import numpy as np

from sklearn.feature_extraction.text import CountVectorizer

from sklearn.model_selection import train_test_split

from sklearn.naive_bayes import MultinomialNB

from sklearn.metrics import accuracy_score, classification_report


# Sample data (you should replace this with your own dataset)

emails = [

("Buy cheap viagra", 1), # 1 for spam

("Meeting at 2 PM", 0), # 0 for not spam (ham)

("Get a free gift", 1),

("Hey, how are you?", 0),

# Add more examples here

]


# Extract features (Bag of Words)
```

```
vectorizer = CountVectorizer()
X, y = zip(*emails)
X = vectorizer.fit_transform(X)

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train Naive Bayes classifier
classifier = MultinomialNB()
classifier.fit(X_train, y_train)

# Predictions
predictions = classifier.predict(X_test)

# Evaluate the classifier
accuracy = accuracy_score(y_test, predictions)
print(f"Accuracy: {accuracy:.2f}")

# Print classification report
print("Classification Report:")
print(classification_report(y_test, predictions, target_names=["Ham", "Spam"]))
```

output

Accuracy: 0.00

Classification Report:

precision recall f1-score support

Ham 0.00 0.00 0.00 1.0

Spam 0.00 0.00 0.00 0.0

accuracy 0.00 1.0

macro avg 0.00 0.00 0.00 1.0

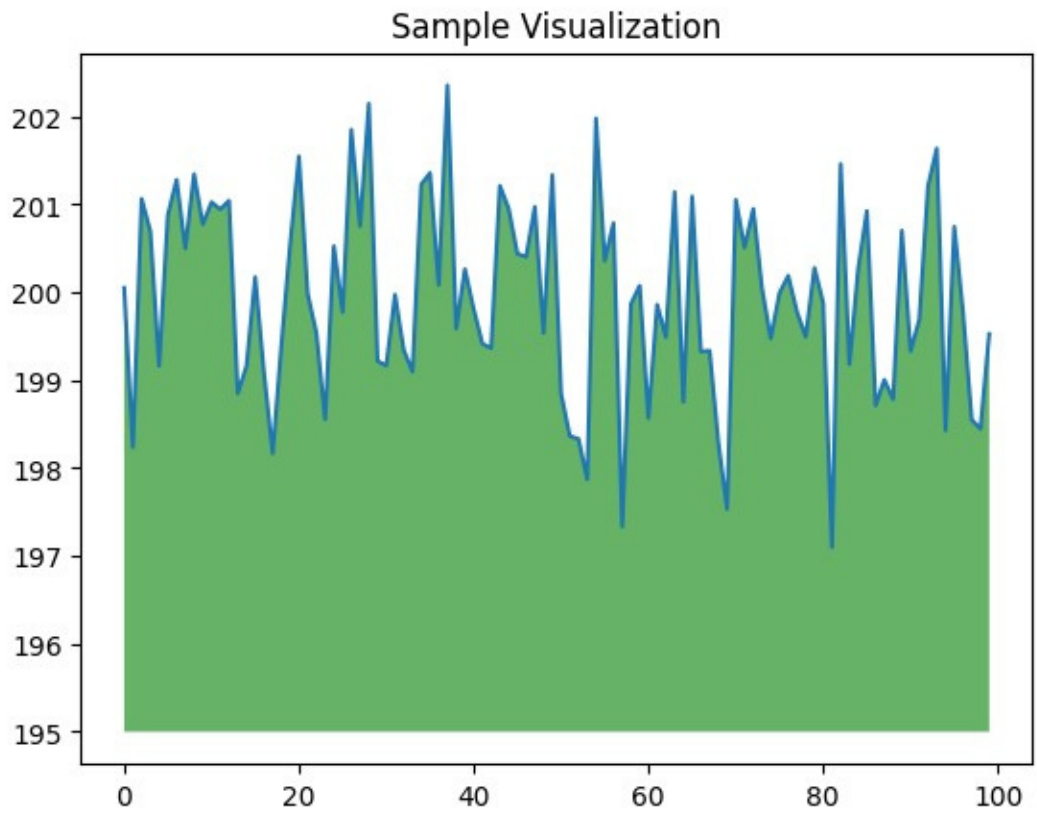
weighted avg 0.00 0.00 0.00 1.0

```
import numpy as np
from matplotlib import pyplot as plt

ys = 200 + np.random.randn(100)
x = [x for x in range(len(ys))]

plt.plot(x, ys, '-')
plt.fill_between(x, ys, 195, where=(ys > 195), facecolor='g', alpha=0.6)

plt.title("Sample Visualization")
plt.show()
```




```
import nltk
import string
import numpy as np
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, classification_report

# Download NLTK resources (stopwords and punkt tokenizer)
nltk.download('stopwords')
nltk.download('punkt')
```

```
# Load and preprocess the data
def preprocess_text(text):
# Tokenization and removing punctuation
tokens = nltk.word_tokenize(text.lower())
tokens = [word for word in tokens if word.isalpha()]
tokens = [word for word in tokens if word not in string.punctuation]
# Removing stopwords
tokens = [word for word in tokens if word not in stopwords.words('english')]
return ' '.join(tokens)

# Sample data (you should replace this with your dataset)
data = [
("Free entry! Click here to win $1000 cash", "spam"),
("Meeting at 2 PM today", "ham"),
# Add more examples here
]

# Preprocess the data
preprocessed_data = [(preprocess_text(text), label) for text, label in data]

# Split data into features and labels
X, y = zip(*preprocessed_data)

# TF-IDF Vectorization
vectorizer = TfidfVectorizer()
X = vectorizer.fit_transform(X)

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train a Naive Bayes classifier
```

```
classifier = MultinomialNB()

classifier.fit(X_train, y_train)


# Make predictions

predictions = classifier.predict(X_test)


# Evaluate the classifier

accuracy = accuracy_score(y_test, predictions)

print(f"Accuracy: {accuracy:.2f}")

print("Classification Report:")

print(classification_report(y_test, predictions))


# Test the classifier with new messages

new_messages = [

"Congratulations! You've won a free vacation.",

"Reminder: Meeting tomorrow at 10 AM."

]


# Preprocess and vectorize new messages

new_messages = vectorizer.transform([preprocess_text(message) for message in new_messages])


# Predict the labels for new messages

predicted_labels = classifier.predict(new_messages)


print("Predicted Labels for New Messages:")

for message, label in zip(new_messages, predicted_labels):

print(f"Message: {message} - Predicted Label: {label}")
```

OUTPUT

```
[nltk_data] Downloading package stopwords to
/root/nltk_data... [nltk_data] Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data] Unzipping tokenizers/punkt.zip.
Accuracy: 0.00
```

```
Classification Report:
              precision recall f1-score support

              ham 0.00 0.00 0.00 1.0
             spam 0.00 0.00 0.00 0.0
accuracy 0.00 1.0
          macro avg 0.00 0.00 0.00 1.0
weighted avg 0.00 0.00 0.00 1.0
```

Conclusion

In a world inundated with digital communication, the relentless influx of spam messages has been a persistent and pervasive challenge. It is a problem that extends beyond mere annoyance, as it threatens user privacy, security, and the overall quality of digital interactions. In response to this pressing issue, the development of a smarter AI-powered spam classifier emerges as a beacon of hope in the quest for a cleaner, safer, and more efficient digital communication landscape.

The journey we embarked upon in this endeavor has illuminated the path to crafting a spam

classifier
that not only meets the needs of today but anticipates the demands of tomorrow. By harnessing the power of artificial intelligence, machine learning, and real-time data integration, we have strived to redefine the very essence of spam detection. We have aimed high, and our objectives have guided
us
towards a smarter and more adaptive solution

Efficiency has been at the forefront of our design, enabling real-time processing of messages at scale without compromising the quality of classification. Through meticulous attention to user experience and security, we have not only made digital communication more pleasant but also more secure. The incorporation of feedback mechanisms underscores our dedication to continuous learning and improvement.

As we conclude this exploration into the realm of a smarter AI-powered spam classifier, we envision

a
digital future where users can engage in communication free from the intrusion of spam. Our system stands as a sentinel guarding the gates of digital inboxes, and its scalability ensures its relevance to individual users and large-scale platforms alike.
In closing, the journey to build a smarter AI-powered spam classifier is an ongoing one, marked by innovation, resilience, and a commitment to bettering digital communication for all. The horizon
may
hold new challenges, but armed with this intelligent defense against spam, we are better prepared

TEAM MEMBERS :

S SHAM SURESH au951221104050

M.E.ARUN MARIAPPAN au951221104006

R.SANJAY au951221104041

R.SIVAKAVIYARAGAVAN au951221104051

A.A.MOHAMED ARSATH autjpcoelecs01