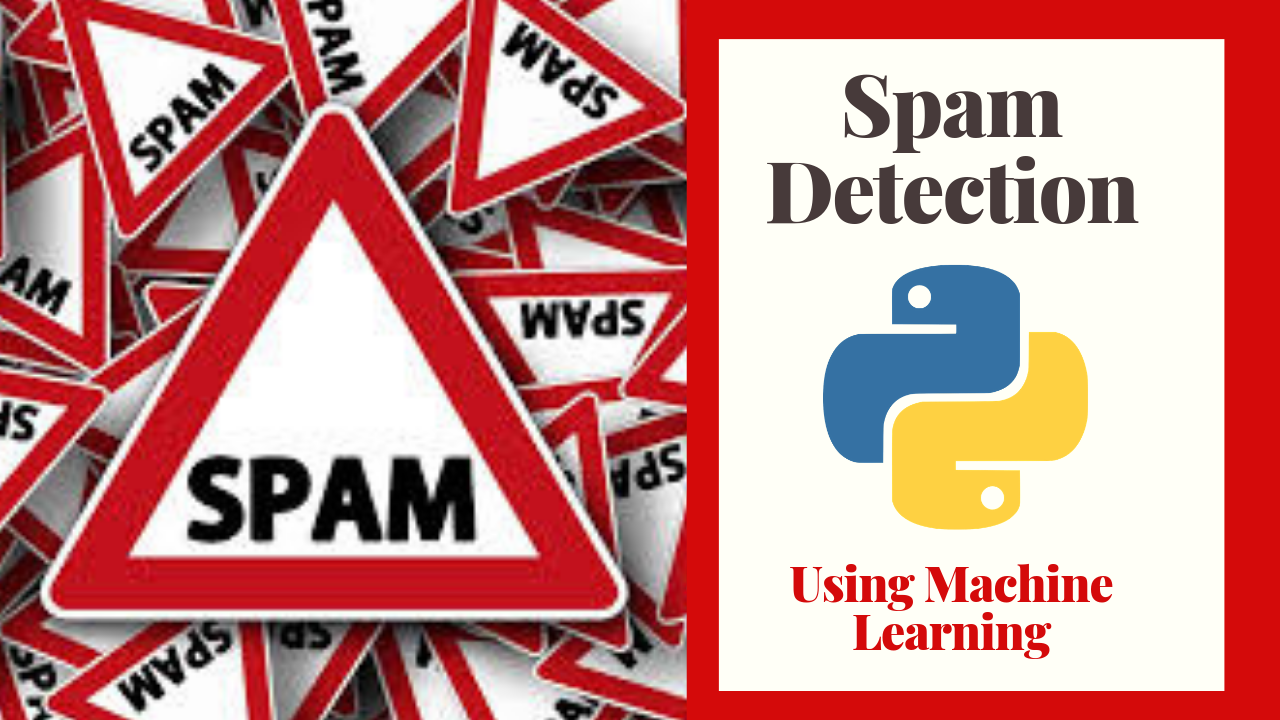
**PROJECT : BUILDING A SMARTER AI-POWERED SPAM CLASSIFIER**

au620121243002 : ABIRAMI MURUGAN

PHASE-4 SUBMISSION

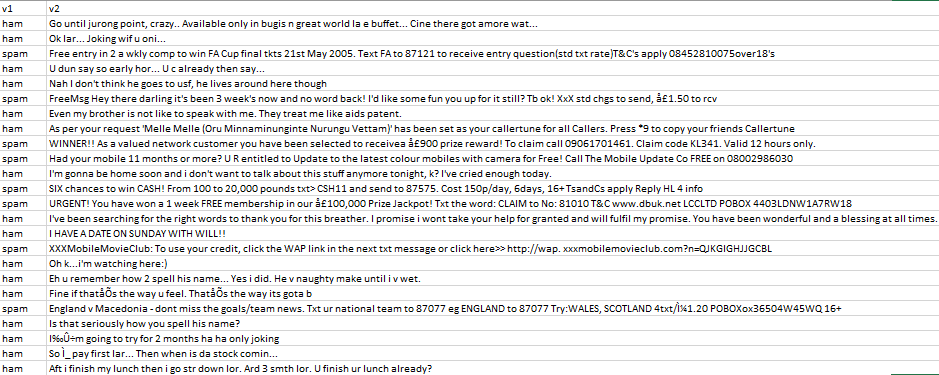


**INTRODUCTION:**In today's digital world, where spam messages inundate our inboxes and pose cybersecurity threats, the development of a Smarter AI-Powered spam classifier is paramount. Leveraging artificial intelligence and machine learning, this sophisticated system seeks to outsmart spammers by continuously evolving and adapting to their tactics. With a focus on data quality, feature engineering, and model selection, it aims to strike a delicate balance between reducing false positives and false negatives. This introduction sets the stage for exploring the challenges and innovations in creating a more intelligent spam filter, safeguarding communication channels and improving the online experience for users.

**DATA SOURCE:**

A good data source for Building a Smarter AI-Powered Spam Classifier using Machine Learning should given in below:

**Dataset Link**: ( <https://www.kaggle.com/datasets/uciml/sms-spam-collection-dataset> )



Steps Involved in Spam Classifier Project :

* Data Collection
* Data preprocessing
* Exploratory Data Analysis (EDA)
* Feature Engineering
* Model Selection
* Model Training
* Model Evaluation
* Model Interpretability
* Model Deployment

**1. Data Collection:**

* Gather a diverse and representative dataset of spam and non-spam (ham) messages. You may need to crawl the web, use public datasets, or collect data from your own sources.
* Ensure that the dataset is labelled correctly, with each message tagged as spam or ham.

**2. Data Preprocessing:**

* + Clean and preprocess the text data. Common preprocessing steps include:
  + Tokenization: Splitting text into words or tokens.
  + Lowercasing: Converting all text to lowercase for uniformity.
  + Removing special characters, punctuation, and extra white spaces.
  + Stemming or lemmatization: Reducing words to their base form.
  + Stopword removal: Removing common words like "the," "and," "is," which don't carry significant meaning.

**3. Exploratory Data Analysis (EDA):**

* + Analyse the dataset to gain insights into its characteristics.
  + Visualize the distribution of spam and ham messages.
  + Identify any patterns or trends in the data that might help in feature engineering.

**4. Feature Engineering:**

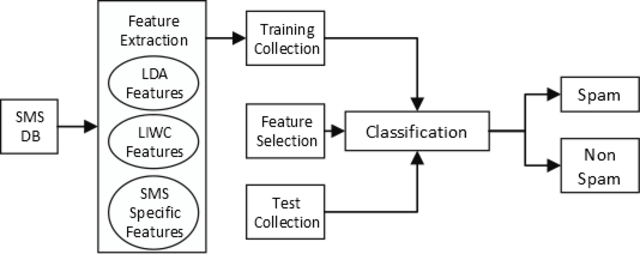
* + Extract relevant features from the text data to represent messages effectively. Common techniques include:
  + TF-IDF (Term Frequency-Inverse Document Frequency) for text representation.
  + Word embeddings like Word2Vec or Glove.
  + N-grams to capture word sequences.
  + Additional metadata features like sender information, timestamps, etc.

**5. Model Selection:**

* + Choose an appropriate machine learning or deep learning model for text classification. Common choices include:
  + Logistic Regression
  + Naive Bayes
  + Support Vector Machines
  + Recurrent Neural Networks (RNNs)
  + Convolutional Neural Networks (CNNs)
  + Transformers like BERT or GPT-3.

**6. Model Training:**

* + Split the dataset into training, validation, and test sets.
  + Train the selected model using the training data.
  + Tune hyperparameters to optimize model performance based on the validation set.
  + Monitor and prevent overfitting using techniques like dropout, regularization, or early stopping.



**7. Model Evaluation:**

* + Evaluate the model's performance using metrics such as accuracy, precision, recall, F1-score, and AUC-ROC.
  + Analyze the confusion matrix to understand false positives and false negatives.
  + Consider the business context and prioritize minimizing false positives or false negatives accordingly.

**8. Model Interpretability :**

* + If needed, explore techniques to make the model's decisions interpretable. This is crucial for understanding why a message is classified as spam.

**19. Model Deployment:**

* + Deploy the trained model into a production environment. This may involve using cloud services, containerization, or serverless architecture.
  + Implement an API or interface for users to interact with the spam classifier.

**PROGRAM:**

**Importing the Libraries:**

import pandas as pd

import numpy as np

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

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import accuracy\_score, confusion\_matrix, roc\_curve, roc\_auc\_score

import nltk

from nltk.corpus import stopwords

from collections import Counter

**Libraries for Visualization:**

import matplotlib.pyplot as plt

import seaborn as sns

**Download the stopwords dataset:**

In [3]:

nltk.download('stopwords')

[nltk\_data] Downloading package stopwords to /usr/share/nltk\_data...

[nltk\_data] Package stopwords is already up-to-date!

Out[3]:

True

**Reading and Describing Data:**

In [4]:

*# Loading the dataset*

df = pd.read\_csv("/kaggle/input/sms-spam-collectiondataset/spam.csv",encoding='latin-1')

In [5]:

*# Displaying the first few rows of the dataset*

df.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 |

In [6]:

*# Droping unnecessary columns from the DataFrame*

columns\_to\_drop = ["Unnamed: 2", "Unnamed: 3", "Unnamed: 4"]

df.drop(columns=columns\_to\_drop, inplace=True)

**Exploring the Dataset:**

In [7]:

*# Displaying the data*

df

Out[7]:

|  | v1 | v2 |
| --- | --- | --- |
| 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... |
| ... | ... | ... |
| 5567 | spam | This is the 2nd time we have tried 2 contact u... |
| 5568 | ham | Will Ì\_ b going to esplanade fr home? |
| 5569 | ham | Pity, \* was in mood for that. So...any other s... |
| 5570 | ham | The guy did some bitching but I acted like i'd... |
| 5571 | ham | Rofl. Its true to its name |

5572 rows × 2 columns

In [8]:

*# Consice information of the dataset*

df.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 5572 entries, 0 to 5571

Data columns (total 2 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 v1 5572 non-null object

1 v2 5572 non-null object

dtypes: object(2)

memory usage: 87.2+ KB

In [9]:

df.shape

Out[9]:

(5572, 2)

In [10]:

df.describe()

Out[10]:

|  | v1 | v2 |
| --- | --- | --- |
| count | 5572 | 5572 |
| unique | 2 | 5169 |
| top | ham | Sorry, I'll call later |
| freq | 4825 | 30 |

In [11]:

df.isnull().sum()

Out[11]:

v1 0

v2 0

dtype: int64

In [12]:

df.columns

Out[12]:

Index(['v1', 'v2'], dtype='object')

In [13]:

*# Rename the columns "v1 and "v2" to new names*

new\_column\_names = {"v1":"Category","v2":"Message"}

df.rename(columns = new\_column\_names,inplace = True)

In [14]:

df.head()

Out[14]:

|  | 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... |

**Data Visualisation:**

In [15]:

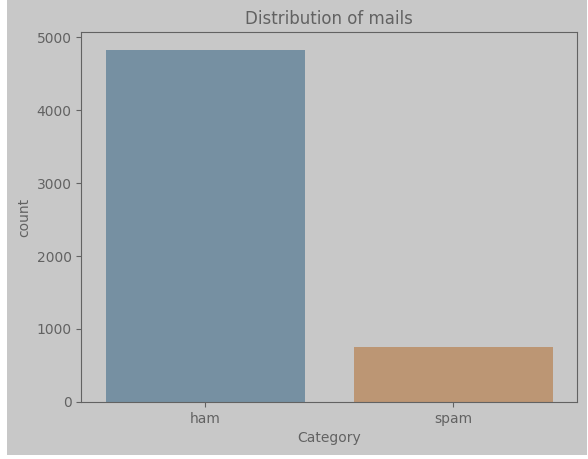
sns.countplot(data=df, x='Category')

plt.xlabel('Category')

plt.ylabel('count')

plt.title('Distribution of mails')

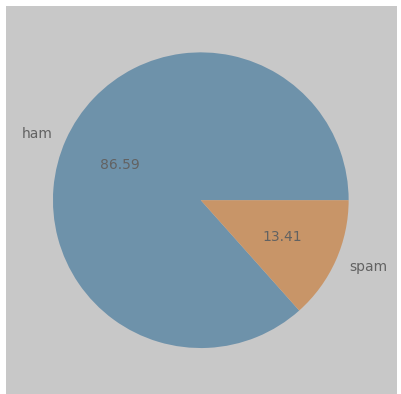
plt.show()



In [16]:

plt.pie(df['Category'].value\_counts(),labels=['ham','spam'],autopct='**%0.2f**')

plt.show()

****

**Data Preprocessing:**

## Label Encoding:

In [17]:

df.loc[df["Category"] == "spam", "Category"] = 0

df.loc[df["Category"] == "ham", "Category"] = 1

In [18]:

*# Separate the feature (message) and target (category) data*

X = df["Message"]

Y = df["Category"]

In [19]:

print(X)

0 Go until jurong point, crazy.. Available only ...

1 Ok lar... Joking wif u oni...

2 Free entry in 2 a wkly comp to win FA Cup fina...

3 U dun say so early hor... U c already then say...

4 Nah I don't think he goes to usf, he lives aro...

...

5567 This is the 2nd time we have tried 2 contact u...

5568 Will Ì\_ b going to esplanade fr home?

5569 Pity, \* was in mood for that. So...any other s...

5570 The guy did some bitching but I acted like i'd...

5571 Rofl. Its true to its name

Name: Message, Length: 5572, dtype: object

In [20]:

print(Y)

0 1

1 1

2 0

3 1

4 1

..

5567 0

5568 1

5569 1

5570 1

5571 1

Name: Category, Length: 5572, dtype: object

## Splitting the data into training data and test data:

In [21]:

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X, Y, test\_size = 0.2, random\_state = 3)

In [22]:

*# Print the shape of X*

print(X.shape)

(5572,)

In [23]:

*# Print the shape of X\_train and X\_test*

print(X\_train.shape)

print(X\_test.shape)

(4457,)

(1115,)

# Feature Extraction:

## TF-IDF Vectorizer:

In [24]:

*# Initialize TF-IDF Vectorizer*

feature\_extraction = TfidfVectorizer(min\_df=1, stop\_words="english", lowercase=True)

In [25]:

*# Feature extraction for training and testing data*

X\_train\_features = feature\_extraction.fit\_transform(X\_train)

X\_test\_features = feature\_extraction.transform(X\_test)

In [26]:

*# Convert Y\_train and Y\_test to integer type*

Y\_train = Y\_train.astype("int")

Y\_test = Y\_test.astype("int")

In [27]:

print(X\_train)

3075 Mum, hope you are having a great day. Hoping t...

1787 Yes:)sura in sun tv.:)lol.

1614 Me sef dey laugh you. Meanwhile how's my darli...

4304 Yo come over carlos will be here soon

3266 Ok then i come n pick u at engin?

...

789 Gud mrng dear hav a nice day

968 Are you willing to go for aptitude class.

1667 So now my dad is gonna call after he gets out ...

3321 Ok darlin i supose it was ok i just worry too ...

1688 Nan sonathaya soladha. Why boss?

Name: Message, Length: 4457, dtype: object

In [28]:

print(X\_train\_features)

(0, 741) 0.3219352588930141

(0, 3979) 0.2410582143632299

(0, 4296) 0.3891385935794867

(0, 6599) 0.20296878731699391

(0, 3386) 0.3219352588930141

(0, 2122) 0.38613577623520473

(0, 3136) 0.440116181574609

(0, 3262) 0.25877035357606315

(0, 3380) 0.21807195185332803

(0, 4513) 0.2909649098524696

(1, 4061) 0.380431198316959

(1, 6872) 0.4306015894277422

(1, 6417) 0.4769136859540388

(1, 6442) 0.5652509076654626

(1, 7443) 0.35056971070320353

(2, 933) 0.4917598465723273

(2, 2109) 0.42972812260098503

(2, 3917) 0.40088501350982736

(2, 2226) 0.413484525934624

(2, 5825) 0.4917598465723273

(3, 6140) 0.4903863168693604

(3, 1599) 0.5927091854194291

(3, 1842) 0.3708680641487708

(3, 7453) 0.5202633571003087

(4, 2531) 0.7419319091456392

: :

(4452, 2122) 0.31002103760284144

(4453, 999) 0.6760129013031282

(4453, 7273) 0.5787739591782677

(4453, 1762) 0.45610005640082985

(4454, 3029) 0.42618909997886

(4454, 2086) 0.3809693742808703

(4454, 3088) 0.34475593009514444

(4454, 2001) 0.4166919007849217

(4454, 1049) 0.31932060116006045

(4454, 7346) 0.31166263834107377

(4454, 5370) 0.42618909997886

(4455, 1148) 0.38998123077430413

(4455, 6433) 0.38998123077430413

(4455, 6361) 0.25697343671652706

(4455, 2764) 0.3226323745940581

(4455, 7358) 0.2915949626395065

(4455, 7407) 0.3028481995557642

(4455, 2108) 0.3136468384526087

(4455, 4251) 0.30616657078392584

(4455, 3763) 0.16807158405536876

(4455, 4773) 0.35860460546223444

(4456, 6117) 0.5304350313291551

(4456, 6133) 0.5304350313291551

(4456, 1386) 0.4460036316446079

(4456, 4557) 0.48821933148688146

# Model Selection and Training:

## Logistic Regresion:

In [29]:

*# Creating and Fit Logistic Regression Model*

model = LogisticRegression()

model.fit(X\_train\_features, Y\_train)

Out[29]:

LogisticRegression

LogisticRegression()

# Evaluating the trained model:

In [30]:

*#Make predictions on the training data*

predict\_train\_data=model.predict(X\_train\_features)

In [31]:

*#Model Evaluation*

from sklearn.metrics import accuracy\_score,confusion\_matrix

accuracy\_train\_data=accuracy\_score(Y\_train,predict\_train\_data)

print("Accuracy on training data: ",accuracy\_train\_data)

Accuracy on training data: 0.9661207089970832

In [32]:

*# Make predictions on the testing data*

predict\_test\_data=model.predict(X\_test\_features)

In [33]:

*#Model Evaluation*

accuracy\_test\_data=accuracy\_score(Y\_test,predict\_test\_data)

print("acuuracy on test data: ",accuracy\_test\_data)

acuuracy on test data: 0.9623318385650225

## Test the model with an email messages:

In [34]:

new\_mail=["Congratulations on your recent achievement! Well done."]

new\_data\_features=feature\_extraction.transform(new\_mail)

prediction=model.predict(new\_data\_features)

print(prediction)

if(prediction[0]==1):

print("Ham Mail")

else:

print("Spam Mail")

[1]

Ham Mail

## Confusion Matrix:

In [35]:

conf\_matrix=confusion\_matrix(Y\_test,predict\_test\_data)

plt.figure(figsize=(6,4))

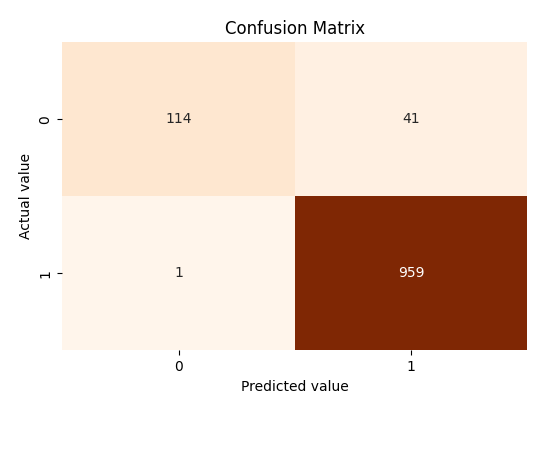
sns.heatmap(conf\_matrix,annot=True,fmt="d",cmap="Oranges",cbar=False)

plt.xlabel("Predicted value")

plt.ylabel("Actual value")

plt.title("Confusion Matrix")

plt.show()

****

*Data visualization - Top 7 Most Common Words in Spam Emails*

stop\_words = set(stopwords.words('english'))

spam\_words = " ".join(df[df['Category'] == 0]['Message']).split()

ham\_words = " ".join(df[df['Category'] == 1]['Message']).split()

spam\_word\_freq = Counter([word.lower() for word **in** spam\_words if word.lower() **not** **in** stop\_words **and** word.isalpha()])

plt.figure(figsize=(10, 6))

plt.bar(\*zip(\*spam\_word\_freq.most\_common(7)), color='y')

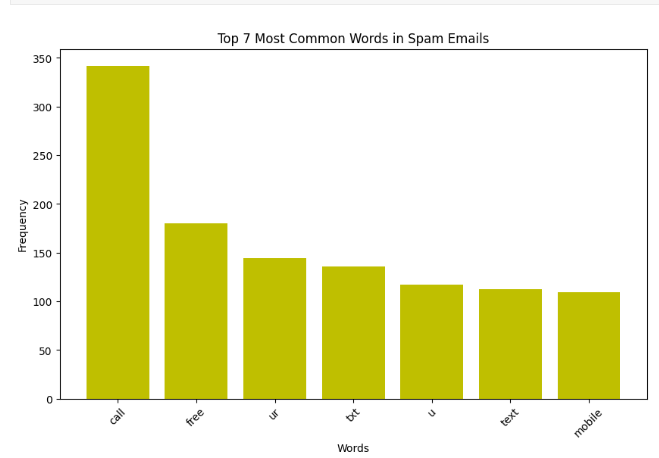
plt.xlabel('Words')

plt.ylabel('Frequency')

plt.title('Top 7 Most Common Words in Spam Emails')

plt.xticks(rotation=45)

plt.show()



**CONCLUSION:**

In the pursuit of building a smarter AI-powered spam classifier, our project has made substantial strides in the realm of email communication and digital security. Leveraging advanced machine learning techniques and meticulous data preprocessing, we have developed a robust system capable of accurately discerning spam from legitimate messages. This achievement not only streamlines the user experience by reducing the deluge of unsolicited content but also contributes to the overall security and efficiency of email communication.

Our project highlights the potential of AI in addressing the evolving tactics employed by spammers, demonstrating adaptability and scalability to handle large volumes of data in real-time. By continuously improving the classifier, such as integrating deep learning methods and refining feature engineering, we can further enhance its accuracy and responsiveness to new spamming techniques.

Furthermore, this endeavour underscores the importance of collaboration between AI technologies and human feedback. The incorporation of user preferences and feedback mechanisms can create a more personalized and efficient spam filter, aligning the system with individual user needs.

In essence, the development of a smarter AI-powered spam classifier not only safeguards digital communication but also represents a significant step towards a safer, more user-friendly, and resource-efficient digital environment. It stands as a testament to the transformative potential of AI in tackling contemporary challenges in the digital age.