Problem Definition:
The problem is to build an AI-powered spam classifier that can accurately distinguish between spam and non-spam messages in emails or text messages. The goal is to reduce the number of false positives (classifying legitimate messages as spam) and false negatives (missing actual spam messages) while achieving a high level of accuracy.

Design Thinking:

Data Collection:
We will need a dataset containing labeled examples of spam and nonspam messages. We can use a Kaggle dataset for this purpose.

Data Preprocessing:
The text data needs to be cleaned and preprocessed. This involves removing special characters, converting text to lowercase, and tokenizing the text into individual words.

Feature Extraction:
We will convert the tokenized words into numerical features using techniques like TF-IDF (Term Frequency-Inverse Document Frequency).

Model Selection:
We can experiment with various machine learning algorithms such as Naive Bayes, Support Vector Machines, and more advanced techniques like deep learning using neural networks.

Building a smarter AI spam classifier involves using advanced machine learning techniques and a rich dataset. Here are some key steps:

1.**Data Collection**: Gather a diverse and extensive dataset of email or message content,including both spam and non-spam examples.
2. **Feature Engineering**: Extract relevant features from the messages, such as text content,sender information, metadata, and more.
3. **Preprocessing**: Clean and preprocess the data, including tasks like text normalization, tokenization, and removing stop words.
4.**Selecting a Model**: Choose an appropriate machine learning model, such as a neural network, support vector machine, or decision tree, and consider deep learning models like recurrent neural networks (RNNs) or transformers.
5.**Training**: Train the model on your dataset, using appropriate loss functions and evaluation metrics like precision, recall, and F1-score.
6.**Data Augmentation**: Augment the dataset by generating more synthetic spam examples tohelp the model learn different variations of spam.
7.**Regularization**: Apply techniques like dropout and weight decay to prevent overfitting.
8. **Hyperparameter Tuning**: Experiment with different hyperparameters to optimize the model's performance.
9.**Ensemble Methods**: Consider using ensemble techniques like bagging or boosting toimprove classification accuracy.
10.**Anomaly Detection**: Implement anomaly detection methods to identify unusual patternsor behaviors that could indicate spam.
11.**Feedback Loop**: Create a feedback loop where user interactions with the classifier help continuously improve its accuracy.
12.**Monitoring and Updating**: Regularly monitor the classifier's performance and update it with new data and techniques.
13.**User Feedback Integration**: Allow users to report false positives and false negatives, and use this feedback to improve the model.
14.**Explainability**: Implement techniques to make the model's decisions more interpretable and understandable
15. **Scalability**: Ensure that the classifier can handle a large volume of messages efficiently.
16. **Security**: Implement security measures to protect against adversarial attacks and ensure the privacy of users' messages.

Text Classification for Spam Detection using SVM

Import necessary libraries

```
import nltk
from nltk.corpus import stopwords
import re
import subprocess
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt
```

Example to test functions

```
test_text = "Hi, Check our website http://example.com for daily coupons! Don't miss promotions."
```

Function to remove stop words

```
stop_words = stopwords.words('english') # Outside the function for code optimization

def stop_words_removal(text):
   text = text.lower()
   tokens = [token for token in text.split() if token not in stop_words]
   text_no_sw = " ".join(tokens)
   return text_no_sw

# Test
print("input:",test_text)
test_text = stop_words_removal(test_text)
print("output:",test_text)
```

Function to remove urls, punctuations and numbers

```
url_pattern = re.compile(r'http\S+|www\S+')
cleaning_pattern = re.compile(r'[^\w\s]|[\d]')

def text_no_urls_puncs_nums(text):
   text = text.lower()
   # Remove URLs
   text = url_pattern.sub('', text)
   # Remove punctuations and numbers
   text = cleaning_pattern.sub('', text)
   return text

# Test
```

```
print("input:",test_text)
test_text = text_no_urls_puncs_nums(test_text)
print("output:",test_text)
```

Function to lemmatize words

```
# Download and unzip wordnet
try:
    nltk.data.find('wordnet.zip')
except:
    nltk.download('wordnet', download_dir='/kaggle/working/')
    command = "unzip /kaggle/working/corpora/wordnet.zip -d /kaggle/working/corpora"
    subprocess.run(command.split())
    nltk.data.path.append('/kaggle/working/')

# Now you can import the NLTK resources as usual
from nltk.corpus import wordnet


from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()

# Lemmatization
def lemmatize_text(text):
    text = text.lower()
    lemmatized_tokens = [lemmatizer.lemmatize(token) for token in nltk.word_tokenize(text)]
    lemmatized_text = ' '.join(lemmatized_tokens)
    return lemmatized_text

# Test
print("input:",test_text)
test_text = lemmatize_text(test_text)
print("output:",test_text)
```

Function to clean text using previous functions

```
# Merge the previous functions
def clean_text(text):
    clean_text = lemmatize_text(text_no_urls_puncs_nums(stop_words_removal(text)))
    # Remove one letter tokens
    clean_text = " ".join([token for token in clean_text.split() if len(token)>1])
    return clean_text

# Test
test_text_2 = "Hi, Check our website http://example.com for daily coupons! Don't miss
promotions."
print("input:",test_text_2)
test_text_2 = clean_text(test_text_2)
print("output:",test_text_2)
```

Loading Dataset

```
# Read Dataset
```

```
data_orig =
pd.read_csv('/kaggle/input/sms-spam-collection-dataset/spam.csv',encoding='latin1')
```
Rename columns
```
data.columns = ['class', 'text']
data
```
Convert class column to spam=1 ham=0
```
data['class'] = data['class'].apply(lambda x: 1 if x == 'spam' else 0)
data
```
Clean Text
```
data['text'] = data['text'].apply(lambda x: clean_text(x))
data.head()
```
Split data to train and test parts
```
x_train, x_test, y_train, y_test = train_test_split(data['text'], data['class'], test_size=0.2,
random_state=20)
# Check Test samples percent
len(x_test)/len(data['text']
```
Transform text to numerical data that SVM can work with
```
cv = CountVectorizer()
x_train = cv.fit_transform(x_train)
x_test = cv.transform(x_test)
```
Train the SVM classifier using the training data
```
model = SVC(random_state = 20)
model.fit(x_train, y_train)
```
Calculate the accuracy of the SVM model on the test data
```
model.score(x_test,y_test)
```
Confusion Matrix
```
y_pred = model.predict(x_test)
y_pred
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt="d")
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()


# Verify Confusion matrix output

check_res = pd.DataFrame(y_test)
check_res['y_pred'] = y_pred

count = 0
for index, row in check_res.iterrows():
   if row['class']==1 and row['y_pred']==0:
      count += 1

print(count)
```

Import necessary libraries

```
import nltk
from nltk.corpus import stopwords
import re
import subprocess
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

test_text = "Hi, Check our website http://example.com for daily coupons! Don't miss promotions."
```

Function to remove stop words

Input(3)

```
stop_words = stopwords.words('english') # Outside the function for code optimization

def stop_words_removal(text):
    text = text.lower()
    tokens = [token for token in text.split() if token not in stop_words]
    text_no_sw = " ".join(tokens)
    return text_no_sw

# Test
print("input:",test_text)
test_text = stop_words_removal(test_text)
print("output:",test_text)
```

output: hi, check website http://example.com daily coupons! miss promotions.

Function to remove urls, punctuations and numbers

```
url_pattern = re.compile(r'http\S+|www\S+')
cleaning_pattern = re.compile(r'[^\w\s]|[\d]')

def text_no_urls_puncs_nums(text):
    text = text.lower()
    # Remove URLs
    text = url_pattern.sub('', text)
    # Remove punctuations and numbers
    text = cleaning_pattern.sub('', text)
    return text

# Test
print("input:",test_text)
test_text = text_no_urls_puncs_nums(test_text)
print("output:",test_text)
```

output: hi check website daily coupons miss promotions

Function to lemmatize words

```
try:
    nltk.data.find('wordnet.zip')
except:
    nltk.download('wordnet', download_dir='/kaggle/working/')
    command = "unzip /kaggle/working/corpora/wordnet.zip -d /kaggle/working/corpora"
    subprocess.run(command.split())
    nltk.data.path.append('/kaggle/working/')

# Now you can import the NLTK resources as usual
from nltk.corpus import wordnet



from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()

# Lemmatization
def lemmatize_text(text):
    text = text.lower()
    lemmatized_tokens = [lemmatizer.lemmatize(token) for token in nltk.word_tokenize(text)]
    lemmatized_text = ' '.join(lemmatized_tokens)
    return lemmatized_text

# Test
print("input:",test_text)
test_text = lemmatize_text(test_text)
print("output:",test_text)
```

Downloading package

```
[nltk_data] Downloading package wordnet to /kaggle/working/...
Archive:  /kaggle/working/corpora/wordnet.zip
   creating: /kaggle/working/corpora/wordnet/
  inflating: /kaggle/working/corpora/wordnet/lexnames
  inflating: /kaggle/working/corpora/wordnet/data.verb
  inflating: /kaggle/working/corpora/wordnet/index.adv
  inflating: /kaggle/working/corpora/wordnet/adv.exc
  inflating: /kaggle/working/corpora/wordnet/index.verb
  inflating: /kaggle/working/corpora/wordnet/cntlist.rev
  inflating: /kaggle/working/corpora/wordnet/data.adj
  inflating: /kaggle/working/corpora/wordnet/index.adj
  inflating: /kaggle/working/corpora/wordnet/LICENSE
  inflating: /kaggle/working/corpora/wordnet/citation.bib
  inflating: /kaggle/working/corpora/wordnet/noun.exc
  inflating: /kaggle/working/corpora/wordnet/verb.exc
  inflating: /kaggle/working/corpora/wordnet/README
  inflating: /kaggle/working/corpora/wordnet/index.sense
  inflating: /kaggle/working/corpora/wordnet/data.noun
  inflating: /kaggle/working/corpora/wordnet/data.adv
  inflating: /kaggle/working/corpora/wordnet/index.noun
  inflating: /kaggle/working/corpora/wordnet/adj.exc
input: hi check website  daily coupons miss promotions
output: hi check website daily coupon miss promotion
```

Function to clean text using previous functions

```
def clean_text(text):
    clean_text = lemmatize_text(text_no_urls_puncs_nums(stop_words_removal(text)))
    # Remove one letter tokens
    clean_text = " ".join([token for token in clean_text.split() if len(token)>1])
    return clean_text

# Test
test_text_2 = "Hi, Check our website http://example.com for daily coupons! Don't miss promotions."
print("input:",test_text_2)
test_text_2 = clean_text(test_text_2)
print("output:",test_text_2)
```

output: hi check website daily coupon miss promotion

Loading Dataset

```
data_orig=pd.read_csv('/kaggle/input/sms-spam-collection-dataset/spam.csv',encoding='latin 1')

data_orig.head()
```

Output

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	NaN	NaN	NaN

		win FA Cup fina...			
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

Output

v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4	
count	5572	5572	50	12	6
unique	2	5169	43	10	5
top	ham	Sorry, I'll call later	bt not his girlfrnd... G o o d n i g h t . . .@"	MK17 92H. 450Ppw 16"	GNT:-)"
freq	4825	30	3	2	2

Output

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...

Rename columns

```
data.columns = ['class', 'text']
data
```

Output

class	text	
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

Convert class column to spam=1 ham=0

```
data['class'] = data['class'].apply(lambda x: 1 if x == 'spam' else 0)
data
```

Output

class	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...
...	...	...
5567	1	This is the 2nd time we have tried 2 contact u...
5568	0	Will Ì_ b going to esplanade fr home?
5569	0	Pity, * was in mood for that. So...any other s...
5570	0	The guy did some bitching but I acted like i'd...
5571	0	Rofl. Its true to its name

Clean Text
data['text'] = data['text'].apply(lambda x: clean_text(x))
data.head()
Output

class	text	

0	0	go jurong point crazy available bugis great wo...
1	0	ok lar joking wif oni
2	1	free entry wkly comp win fa cup final tkts st ...
3	0	dun say early hor already say
4	0	nah think go usf life around though

Split data to train and test parts

```
x_train, x_test, y_train, y_test = train_test_split(data['text'], data['class'], test_size=0.2, random_state=20)

len(x_test)/len(data['text'])
```

Output

```
0.20010768126346015
```

Transform text to numerical data that SVM can work with

```
cv = CountVectorizer()
x_train = cv.fit_transform(x_train)
x_test = cv.transform(x_test
```

Train the SVM classifier using the training data

```
model = SVC(random_state = 20)
model.fit(x_train, y_train)
```

Output

```
SVC
SVC(random_state=20)
```

Calculate the accuracy of the SVM model on the test data

```
model.score(x_test,y_test)
```

Output

```
0.9829596412556054
```

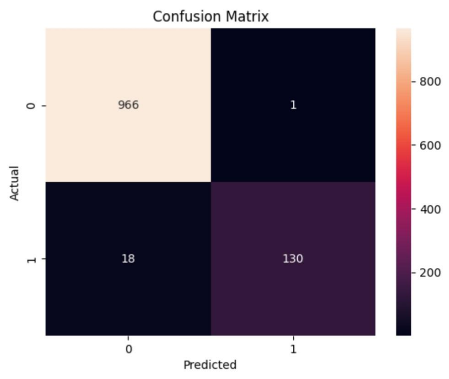
Confusion Matrix

```
y_pred = model.predict(x_test)
y_pred
```

Output
array([0, 0, 0, ..., 0, 0, 1])

```
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt="d")
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```

Output



Verify Confusion matrix output

```
Verify Confusion matrix output

check_res = pd.DataFrame(y_test)
check_res['y_pred'] = y_pred

count = 0
for index, row in check_res.iterrows():
    if row['class']==1 and row['y_pred']==0:
        count += 1

print(count)
18
```