- **TO BUILD A SMART AI POWERED SPAM CLASSIFIER,CONSIDER THESE KEY MODULES :**

Data Collection:

Gather a diverse dataset of both spam and non-spam emails for training.

1. Preprocessing:

Clean and preprocess the data by removing irrelevant information, handling missing values, and normalizing text.

2. Feature Extraction:

Extract relevant features from emails, such as word frequency, sender information, and email structure.

3. Model Selection:

Choose a suitable machine learning model, like a Naïve Bayes classifier or a neural network, based on your dataset and requirements.

4. Training:

Train the selected model using the preprocessed data, adjusting parameters for optimal performance.

5. Evaluation:

Assess the model's performance using metrics like accuracy, precision, recall, and F1 score.

6. Integration with Email System:

Implement the classifier into the email system, ensuring seamless integration and real-time spam detection.

7. Feedback Mechanism:

Develop a feedback loop to continuously improve the model by incorporating user feedback on misclassifications.

8. Regular Updates:

Periodically update the model with new data to adapt to evolving spam patterns.

9. User Interface:

Create a user-friendly interface to allow users to manage and customize spam filtering settings.

10. Source Code :

Building a spam classifier involves using machine learning. Here's a simplified example in Python using a popular library like scikit-learn:

```python
# Import necessary libraries
From sklearn.model_selection import train_test_split
From sklearn.feature_extraction.text import CountVectorizer
From sklearn.naive_bayes import MultinomialNB
From sklearn.metrics import accuracy_score, classification_report

# Sample data (replace with your dataset)
Data = [("Spam message 1", 1), ("Non-spam message 2", 0), …]

# Separate data into features and labels
Messages, labels = zip(*data)

# Split the data into training and testing sets
Msg_train, msg_test, label_train, label_test = train_test_split(messages, labels, test_size=0.2, random_state=42)

# Convert messages to a bag-of-words representation
Vectorizer = CountVectorizer()
Msg_train = vectorizer.fit_transform(msg_train)
Msg_test = vectorizer.transform(msg_test)
```

```
# Train a Naïve Bayes classifier
Classifier = MultinomialNB()
Classifier.fit(msg_train, label_train)

# Make predictions on the test set
Predictions = classifier.predict(msg_test)

# Evaluate the classifier
Accuracy = accuracy_score(label_test, predictions)
Report = classification_report(label_test, predictions)

Print(f”Accuracy: {accuracy}”)
Print(“Classification Report:\n”, report)
```