Building a smarter AI powered spam classifier

Phase-3

Loading and preprocessing a dataset for building a smarter AI-powered spam classifier involves several steps. Here's a high-level overview of the process:

Data Collection: Gather a labeled dataset that contains examples of both spam and non-spam (ham) messages. This dataset should be diverse and representative of the messages your classifier will encounter.

Data Cleaning: Remove any irrelevant or duplicate data, as well as any outliers. Ensure that your dataset is well-structured and consistent.

- Text Preprocessing:
- Text Tokenization: Split the text into individual words or tokens.
- Lowercasing: Convert all text to lowercase to ensure consistency.
- Removing Punctuation: Eliminate punctuation marks that don't carry significant meaning.
- Stopword Removal: Exclude common words (e.g., "and," "the," "in") that are unlikely to help classify spam.
- Stemming or Lemmatization: Reduce words to their base or root form to handle variations (e.g., "running" to "run").
- Word Embeddings: Use pre-trained word vectors like Word2Vec or GloVe to capture semantic meaning.Split the Dataset into training, validation, and test sets to evaluate your model's performance.

PROGRAM:

```
# Import necessary libraries
import pandas as pd
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

# Load and preprocess the dataset
data = pd.read_csv('spam_data.csv')  # Replace 'spam_data.csv' with your dataset

# Perform text preprocessing as described in the previous response

# Split the dataset into training and testing sets
X = data['message']
y = data['label']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Feature extraction using TF-IDF
tfidf_vectorizer = TfidfVectorizer(max_features=5000)  # You can adjust the max_features
X_train_tfidf = tfidf_vectorizer.fit_transform(X_train)
X_test_tfidf = tfidf_vectorizer.transform(X_test)
```

```
# Build a classification model (Naive Bayes in this example)
classifier = MultinomialNB()
classifier.fit(X_train_tfidf, y_train)

# Make predictions
y_pred = classifier.predict(X_test_tfidf)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred)

print("Accuracy:", accuracy)
print("Classification Report:\n", report)

# Save the trained model for future use
import joblib
joblib.dump(classifier, 'spam_classifier_model.pkl')

# Deployment: Integrate this model into your application or service to classify spam messages
```

CONCLUSION:

Thus,building a smarter AI-powered spam classifier involves several steps,including data preprocessing,model development,and deployment.