NAAN MUDHALVAN

Phase 5

Name : S. David Ephraim

Register number : 210821243009

College : Kings Engineering College

Domain : Artificial Intelligence

Project : Building a Smarter AI-Powered Spam Classifier

Program :

```
import pandas as pd
from sklearn.model_selection import train_test_split
df = pd.read_csv('spam.csv', encoding='ISO-8859-1')
df.rename(columns={'v1': 'class_label', 'v2': 'message'}, inplace=True)
df.drop(['Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4'], axis=1, inplace=True
X_train, X_test, y_train, y_test = train_test_split(df['message'], df['class_label'],
test_size=0.15, random_state=111)
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.preprocessing import LabelEncoder
vectorizer = TfidfVectorizer()
X_train = vectorizer.fit_transform(X_train)
X_test = vectorizer.transform(X_test)
encoder = LabelEncoder()
y_train = encoder.fit_transform(y_train)
y_test = encoder.transform(y_test)
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
naive_bayes = MultinomialNB()
naive_bayes.fit(X_train, y_train)
y_pred = naive_bayes.predict(X_test)
print('Accuracy:', accuracy_score(y_test, y_pred))
print('Precision:', precision_score(y_test, y_pred))
```

```
print('Recall:', recall_score(y_test, y_pred))
print('F1 Score:', f1_score(y_test, y_pred))
```

Design Thinking Process:

Understand the Problem:

- Identify the characteristics of spam messages.
- Analyze existing spam filters and their limitations.

Define Objectives:

- Improve spam classification accuracy.
- Minimize false positives (legitimate messages classified as spam) and false negatives (spam messages not detected).

Data Collection:

- Gather a labeled dataset of messages, including both spam and non-spam messages.

Data Preprocessing:

- Data Cleaning:
  - Remove duplicate messages.
  - Handle missing or noisy data.
- Text Preprocessing:
  - Tokenization: Split messages into words or phrases.
  - Stopword removal: Eliminate common and uninformative words.
  - Lemmatization or stemming: Reduce words to their base forms.
  - Handle special characters, URLs, and numbers.

Feature Extraction:

- Bag of Words (BoW): Create a vector representation of messages.
- TF-IDF (Term Frequency-Inverse Document Frequency): Assign importance scores to words based on their frequency.
- Word Embeddings (e.g., Word2Vec, GloVe): Represent words as dense vectors.
- N-grams: Capture word sequences and context.

Model Selection:

- Choose machine learning algorithms suitable for text classification:
  - Logistic Regression, Naive Bayes, Support Vector Machines, Decision Trees, Random Forest, Gradient Boosting, Neural Networks, etc.

Model Training:

- Split the dataset into training, validation, and test sets.
- Train the chosen model(s) using the training data.
- Tune hyperparameters using the validation set to optimize model performance.

Evaluation Metrics:

- Use appropriate metrics to evaluate the model:
  - Accuracy, Precision, Recall, F1-score, ROC-AUC, Confusion Matrix.
- Optimize for precision and recall, depending on the specific requirements (e.g., minimize false positives or false negatives).

Innovative Techniques:

a. Ensemble Methods:
- Combine multiple models (e.g., Random Forest, Gradient Boosting) to improve performance and reduce overfitting.

b. Deep Learning:
- Experiment with neural networks, such as LSTM or CNN, to capture complex patterns in text data.

c. Active Learning:
- Implement an active learning approach to iteratively improve the model by selecting the most informative samples for labeling.

d. Explainability:
- Utilize techniques like SHAP values or LIME to explain model predictions and build user trust.

Phases of Development:

Data Collection and Preprocessing:
- Collect and clean the dataset.
- Preprocess the text data.

Feature Extraction:
- Extract relevant features from text.

Model Selection:
- Choose appropriate machine learning algorithms or deep learning architectures.

Model Training and Tuning:
- Train the model on the preprocessed data.
- Tune hyperparameters for optimal performance.

Evaluation:
- Evaluate the model's performance using appropriate metrics.

Iteration:
- Implement innovative techniques and iterate on the model for improvements.

Deployment:
- Deploy the spam classifier as a service or integrate it into an existing system.

Monitoring and Maintenance:
- Continuously monitor the model's performance and retrain as needed.