Building a Smarter AI-Powered Spam Classifier

Phase – II

Team Name : Proj_208227_Team_1

INNOVATION:

Building an AI-powered spam classifier using innovative techniques and approaches.

1. Data Preprocessing Module:

○ Objective: Clean and prepare the raw data for model training.

○ Innovation: Use advanced text preprocessing techniques like word embeddings, lemmatization, and custom feature extraction to capture nuanced patterns in spam messages.

○ Sample Code:

```
from nltk.corpus import stopwords

from nltk.stem import WordNetLemmatizer

import re

    def preprocess_text(text):

    text = re.sub(r'\W', ' ', text)

    text = re.sub(r'\s+', ' ', text)

    text = text.lower()

    lemmatizer = WordNetLemmatizer()

    text = ' '.join(lemmatizer.lemmatize(word) for word in text.split() if word not in set(stopwords.words('english')))
```

```
        return text
```

2. Machine Learning Model Training Module:

○ Objective: Train a machine learning model on the preprocessed data.

○ Innovation: Experiment with ensemble learning techniques or transfer learning approaches for improved model accuracy.

○ Sample Code:

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

predictions = model.predict(X_test)
accuracy = accuracy_score(y_test, predictions)
print(f'Model Accuracy: {accuracy}')
```

3. Natural Language Processing (NLP) Module:

○ Objective: Leverage NLP techniques for better feature extraction and understanding of text data.

○ Innovation: Use pre-trained language models like BERT or GPT for contextual understanding of messages.

○ Sample Code:

```
Using Hugging Face Transformers library for BERT

from transformers import BertTokenizer, BertModel


tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')

model = BertModel.from_pretrained('bert-base-uncased')


text = "Example spam message"

tokens = tokenizer(text, return_tensors='pt')

outputs = model(**tokens)
```

4.Deep Learning Module:

○ Objective: Explore deep learning architectures for improved performance.

○ Innovation: Try using recurrent neural networks (RNNs) or attention mechanisms for capturing sequential patterns in text.

○ Sample Code:

```
Simple LSTM model with Keras

from keras.models import Sequential

from keras.layers import Embedding, LSTM, Dense
```

```
model = Sequential()

model.add(Embedding(input_dim=vocab_size, output_dim=embedding_dim, input_length=max_sequence_length))

model.add(LSTM(units=100))

model.add(Dense(units=1, activation='sigmoid'))


model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

model.fit(X_train, y_train, epochs=10, batch_size=32, validation_data=(X_test, y_test))
```

5. Adaptive Learning Module:

○ Objective: Implement mechanisms for the model to adapt and learn from changing spam patterns.

○ Innovation: Integrate online learning techniques to update the model in real-time based on user feedback.

○ Sample Code: Not as straightforward as others; may involve updating the model weights based on new data periodically.

6. Explainability and Interpretability Module:

○ Objective: Ensure transparency and interpretability of the model's decisions.

○ Innovation: Use SHAP (SHapley Additive exPlanations) values or LIME (Local Interpretable Model-agnostic Explanations) for explaining model predictions.

○ Sample Code:

```
SHAP values with a trained model
import shap

explainer = shap.TreeExplainer(model)
shap_values = explainer.shap_values(X_test)

shap.summary_plot(shap_values, X_test)
```

7. User Interface Module:

○ Objective: Provide a user-friendly interface for users to interact with the spam classifier.

○ Innovation: Develop a web-based interface using frameworks like Flask or Django, or create a chatbot interface for seamless communication.

○ Sample Code:

```
Flask web application
from flask import Flask, render_template, request

app = Flask(__name__)

@app.route('/')
def home():
    return render_template('index.html')
```

```
@app.route('/predict', methods=['POST'])
def predict():
    user_input = request.form['user_input']
    processed_input = preprocess_text(user_input)
    prediction = model.predict([processed_input])[0]

    return render_template('result.html', prediction=prediction)

if __name__ == '__main__':
    app.run(debug=True)
```