PHASE 5

Building a smarter AI-Powered spam classifier

Project Description:

it is a binary classification problem. The reason to do this is simple: by detecting unsolicited and unwanted emails, we can prevent spam messages from creeping into the user's inbox, thereby improving user experience.

Understanding the Problem:

Today, learning-based classifiers are commonly used for spam detection. In learning-based classification, the detection process assumes that spam emails have a specific set of features that differentiate them from legitimate emails .

Approach:

Creating a smarter AI-based spam classifier involves a combination of techniques and technologies. Here are some steps to consider:

1. Data collection:

Collect a diverse and comprehensive dataset of spam and non-spam emails. The data set must be clearly labeled.

2. Data preprocessing:

Data cleaning and preprocessing. This includes removing duplicates, handling missing values, and encoding text.

3. Technical features:

Extract relevant features from text, such as word frequency, character patterns, and sender information. Consider using techniques like TF-IDF (Term Frequency Inverse Document Frequency) or word embeddings like Word2Vec or GloVe.

4. Select model:

Choose the appropriate machine learning or deep learning model for classification. Popular choices include Naive Bayes, Support Vector Machines, Random Forests, and neural networks like LSTM or CNN.

5. Training:

Split the dataset into training set and validation set. Train the selected model on the training data and fine-tune the hyperparameters to optimize performance.

6. Review:

Evaluate model performance using metrics such as accuracy, precision, recall, F1 score, and ROC-AUC. Adjust models and features based on evaluationresults.

7. Overall method:

Consider using aggregation methods such as stacking or boosting to improve classification accuracy.

8. Cross validation:

Perform cross-validation to ensure generalizability of the model.

9. Real-time scoring:

Deploy the trained model in a real-time environment where the modelcan classify incoming emails or messages as spam or not.

10. Feedback loop:

Continuously monitor the classifier's performance and periodicallyretrain it with new data to adapt to changing

```
# -*-
coding: utf-
8 -*-#
coding: utf-
8
#NaiveBaye
s
import numpy
from pandas import DataFrame
from sklearn.feature_extraction.text
import CountVectorizerfrom
sklearn.naive_bayes import MultinomialNB

#Function to read files (emails) from
the local directorydef readFiles(path):
  for root, dirnames, filenames in
    os.walk(path):for filename in
    filenames:
      path = os.path.join(root, filename)

      inBod
      y =
      False
      lines =
      []
```

```
        f = io.open(path, 'r',
        encoding='latin1')for line
        in f:
          if inBody:
            lines.app
          end(line)
          elif line ==
          '\n':
            inBod
        y = True
        f.close()
        message =
        '\n'.join(lines)
        yield path,
        message
def dataFrameFromDirectory(path, classification):
    rows.append({'message': message,
    'class': classification})
    index.append(filename)

  return DataFrame(rows, index=index)

#An empty dataframe with 'message' and
'class' headersdata =
DataFrame({'message': [], 'class': []})
```

```
#Including the email details with the spam/ham classification in the dataframe

data =
data.append(dataFrameFromDirectory('C:/Users/surya/Desktop/DecemberBre
ak/emails/spam','spam'))

data =
data.append(dataFrameFromDirectory('C:/Users/surya/Desktop/DecemberBre
ak/emails/ham','ham'))


#Head and the

Tail of 'data'

data.head()

print(data.tail())


vectoriser = CountVectorizer()

count =

vectoriser.fit_transform(data['message'].val

ues)print(count)


target = data['class'].values
print(target)


classifier =

MultinomialNB()

classifier.fit(cou

nt, target)

print(classifier)
```

exampleInput = ["Hey. This is John Cena. You can't see me", "Free Viagra boys!!", "Please reply to get thisoffer"]

excount =

vectoriser.transform(exampleInput)

print(excount)

prediction =

classifier.predict(excount

)print(prediction)

DatasetLink:

https://www.kaggle.com/datasets/uciml/sms-spam-collection-dataset