***Building a smarter AI-Powered spam classifier(phase 4)***

```
# -*- coding: utf-8 -*-
# coding: utf-8
#Naive Bayes
import os
import io
import numpy
from pandas import DataFrame
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB

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

            inBody = False
            lines = []
            f = io.open(path, 'r', encoding='latin1')
            for line in f:
                if inBody:
                    lines.append(line)
                elif line == '\n':
                    inBody = True
            f.close()
            message = '\n'.join(lines)
            yield path, message
```

```
def dataFrameFromDirectory(path, classification):
    rows = []
    index = []
    for filename, message in readFiles(path):
        rows.append({'message': message, 'class': classification})
        index.append(filename)

    return DataFrame(rows, index=index)

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

#Including the email details with the spam/ham classification in the dataframe
data = data.append(dataFrameFromDirectory('C:/Users/surya/Desktop/DecemberBreak/emails/spam',
'spam'))
data = data.append(dataFrameFromDirectory('C:/Users/surya/Desktop/DecemberBreak/emails/ham',
'ham'))

#Head and the Tail of 'data'
data.head()
print(data.tail())

vectoriser = CountVectorizer()
count = vectoriser.fit_transform(data['message'].values)
print(count)

target = data['class'].values
```

```
print(target)


classifier = MultinomialNB()

classifier.fit(count, target)

print(classifier)


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

excount = vectoriser.transform(exampleInput)

print(excount)


prediction = classifier.predict(excount)

print(prediction)
```

Implementing a smarter AI-powered spam classifier in your project involves several steps across different technologies. Here's a more detailed guide on how to go about it:

1. **AI (Machine Learning):**

   - Start by selecting a machine learning algorithm for spam classification. For instance, you can begin with a simple algorithm like Naive Bayes.

   - Collect a dataset of labeled emails or messages, distinguishing between spam and non-spam.

   - Preprocess the text data, including tasks like tokenization, lowercasing, and removing stop words.

   - Split your dataset into training and testing sets.

   - Train your initial model and evaluate its performance using metrics like accuracy and F1-score.

   - Experiment with different techniques, algorithms, and hyperparameters to improve model performance.

2. **ADS (Advanced Data Science):**

- Dive deeper into feature engineering by creating new features from the text data. This could involve techniques like word embeddings, topic modeling, or sentiment analysis.

- Explore more advanced machine learning algorithms such as Random Forest, Gradient Boosting, or deep learning (e.g., LSTM or CNN for text classification).

- Address the issue of imbalanced data by applying techniques like oversampling (SMOTE) or undersampling.

- Conduct cross-validation to assess the model's generalization performance.

- Analyze the confusion matrix to understand the trade-offs between precision and recall.

3. **DAC (Data Analytics and Visualization):**

- Use IBM Cognos to analyze your data and create visualizations.

- Develop visualizations that show the distribution of spam and non-spam messages and their characteristics.

- Create interactive dashboards to allow users to explore the data and the model's performance.

- Use filters and drill-through capabilities to enable in-depth analysis.

- Create a comprehensive report that explains the insights and findings from the analysis.

4. **IOT (Internet of Things):**

- Develop a web-based platform that integrates your AI-powered spam classifier.

- Implement user authentication for secure access to the platform.

- Consider incorporating real-time data streams or event triggers if needed for monitoring incoming messages.

- Ensure the platform is scalable and responsive.

- Document the development process and provide clear instructions for deploying the platform.

5. **CAD (Cloud Application Development):**

- Utilize IBM Cloud Foundry for building and hosting your application.

- Create a user-friendly interface for users to input messages and receive spam classification results.

- Focus on security by encrypting data and implementing access control mechanisms.

- Include error handling and logging to ensure the application's reliability.
- Document the development, codebase, and deployment steps thoroughly.

Throughout all these phases, keep in mind the importance of documentation. Document your code, methodologies, and the rationale behind your choices. This documentation will not only help with assessment but also with maintaining and evolving your project in the future.