Title of the Project: Email Spam Detection Using Machine Learning

Introduction and Objectives of the Project: The objective of this project is to develop a machine learning model that can accurately detect spam emails and filter them out of a user's inbox. The goal is to create a more efficient and effective way to manage email communication, especially for businesses that receive a large volume of emails on a daily basis.

Project Category: Artificial Intelligence and Machine Learning along with data Analysis

Tools/Platform, Hardware and Software Requirements:

- Programming Language: Python
- Machine Learning Libraries: NumPy, Pandas, and scikit-learn
- Database: MySQL (csv file)
- Hardware Requirements: A computer with minimum 8GB RAM and 500GB storage
- Software Requirements: Python IDE (such as PyCharm or Anaconda or jupyter notebook), MySQL Workbench

Problem Definition, Requirement Specifications, Project Planning and Scheduling: The problem is to identify spam emails accurately while minimizing false positives (i.e., legitimate emails mistakenly identified as spam). The project will be completed in several phases:

- Data Collection: Gather a large dataset of emails, including both spam and legitimate emails.
- Data Cleaning and Pre-processing: Process the data to remove any noise or irrelevant information.
- Feature Extraction: Extract relevant features from the emails, such as the sender's email address, subject line, and body text.
- Model Development: Train and test a machine learning model on the extracted features to accurately classify emails as spam or legitimate.
- Model Deployment: Integrate the model into an email client to filter spam emails in real-time.

Scope of the Solution: The solution will focus on developing a machine learning model that can accurately identify spam emails. The model will be integrated into an email client to filter spam emails in real-time. The solution will not address other email-related problems, such as email security or email encryption.

Analysis:

- Data Models: 0, 1, and 2 level DFDs will be created to illustrate the flow of data in the system.
- ER Diagrams: ER diagrams will be created to show the relationships between different entities in the system.
- Class Diagrams: Class diagrams will be created to illustrate the different classes and their relationships in the system.

A complete database and tables detail with primary and foreign keys, and appropriate limitations in the fields (according to extend necessities) will be determined during the project development phase.

This code performs email spam detection using machine learning. Here's a brief explanation of each section of the code:

1. Imports: The code imports the necessary libraries for data processing and machine learning such as NumPy, Pandas, and scikit-learn.
2. Data Reading: The code reads the data from a CSV file using Pandas.
3. Data Processing: The code processes the data by creating a new column called "Spam" which is assigned a value of 1 if the email is spam and 0 if it is not. It then splits the data into training and testing sets using the train_test_split function from scikit-learn.
4. Model Creation: The code creates a pipeline that consists of two parts: a CountVectorizer and a Multinomial Naive Bayes classifier. CountVectorizer is used to convert the email text into a matrix of word counts, and MultinomialNB is used as the classifier.
5. Training the Model: The code trains the pipeline on the training set using the fit function.
6. Prediction: The code predicts the classification of two sample emails using the trained model.
7. Model Evaluation: The code evaluates the performance of the model on the testing set using the score function, which returns the mean accuracy of the classifier.

Overall, the code performs email spam detection using a simple machine learning model and achieves a high accuracy of 0.9777458722182341 on the testing set.

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the inp
ut directory

import os
for dirname, _, filenames in os.walk('D:\DNO\minor'):
    for filename in spam:
        print(os.path.join(dirname, filename))

data=pd.read_csv('D:\DNO\minor\spam.csv')
data
```

	Category	Message
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...
...	...	...
5567	spam	This is the 2nd time we have tried 2 contact u...
5568	ham	Will ü b going to esplanade fr home?
5569	ham	Pity, * was in mood for that. So...any other s...
5570	ham	The guy did some bitching but I acted like i'd...
5571	ham	Rofl. Its true to its name

```
data.columns
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5572 entries, 0 to 5571
Data columns (total 2 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   Category  5572 non-null   object
 1   Message   5572 non-null   object
dtypes: object(2)
memory usage: 87.2+ KB
```

```
data['Spam']=data['Category'].apply(lambda x:1 if x=='spam' else 0)
data.head(5)
```

	Category	Message	Spam
0	ham	Go until jurong point, crazy.. Available only ...	0
1	ham	Ok lar... Joking wif u oni...	0
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	1
3	ham	U dun say so early hor... U c already then say...	0
4	ham	Nah I don't think he goes to usf, he lives aro...	0

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(data.Message,data.Spam,test_size=0.25)
#CounterVectorizer Convert the text into matrics
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.pipeline import Pipeline
clf=Pipeline([
    ('vectorizer',CountVectorizer()),
    ('nb',MultinomialNB())
])
```

Tarining The Model

```
clf.fit(X_train,y_train)
```

Here I given Two email Two detect 1st One is looking good and the other one looking spam

```
emails=[
    'Sounds great! Are you home now?',
    'Will u meet ur dream partner soon? Is ur career off 2 a flyng start? 2 find out free, txt HORO fo
llowed by ur star sign, e. g. HORO ARIES'
]
```

Predict Email

```
clf.predict(emails)
```

Prediction Of Model

```
clf.score(X_test,y_test)
```

0.9777458722182341