```
print("CS ASS 3")
```

```
CS ASS 3
```

```
import pandas as pd

spam_mail =
pd.read_csv("/home/comp53/Documents/archive/spam_ham_dataset.csv")

spam_mail.head()
```

```
   Unnamed: 0 label                                                text
\
0         605   ham  Subject: enron methanol ; meter # : 988291\r\n...

1        2349   ham  Subject: hpl nom for january 9 , 2001\r\n( see...

2        3624   ham  Subject: neon retreat\r\nho ho ho , we ' re ar...

3        4685  spam  Subject: photoshop , windows , office . cheap ...

4        2030   ham  Subject: re : indian springs\r\nthis deal is t...


   label_num
0          0
1          0
2          0
3          1
4          0
```

```
spam_mail.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5171 entries, 0 to 5170
Data columns (total 4 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Unnamed: 0  5171 non-null   int64
 1   label       5171 non-null   object
 2   text        5171 non-null   object
 3   label_num   5171 non-null   int64
dtypes: int64(2), object(2)
memory usage: 161.7+ KB
```

```
# Remove the Unnecessary Column
spam_mail = spam_mail.drop(columns=['Unnamed: 0'])

# Check for Missing Values
spam_mail.isnull().sum()
```

```
label        0
text         0
label_num    0
dtype: int64
```

```
# Check spam vs ham distribution
spam_mail['label'].value_counts()
```

```
label
ham     3672
spam    1499
Name: count, dtype: int64
```

```
# Undersampling (Reduce Ham Emails)
from imblearn.under_sampling import RandomUnderSampler

X = spam_mail['text']  # Features (Email Text)
y = spam_mail['label_num']  # Target (0 = Ham, 1 = Spam)

undersampler = RandomUnderSampler(random_state=42)
X_resampled, y_resampled =
undersampler.fit_resample(X.values.reshape(-1, 1), y)

# Convert back to DataFrame
balanced_data = pd.DataFrame({'text': X_resampled.flatten(),
'label_num': y_resampled})
print(balanced_data['label_num'].value_counts())  # Now spam = ham
```

```
label_num
0    1499
1    1499
Name: count, dtype: int64
```

```
# Convert Text into Numerical Features (TF-IDF)
from sklearn.feature_extraction.text import TfidfVectorizer

# Convert email text into numerical form
vectorizer = TfidfVectorizer(max_features=5000)  # Keep top 5000 words
X_transformed = vectorizer.fit_transform(spam_mail['text']).toarray()
y = spam_mail['label_num']  # Use the numeric labels

# Split Data into Training & Testing Sets
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X_transformed, y,
test_size=0.2, random_state=42)

print(f"Training set size: {len(X_train)}, Testing set size:
{len(X_test)}")
```

```
Training set size: 4136, Testing set size: 1035
```

```python
# Train an ML Model
# Naïve Bayes (Best for Text Data)
from sklearn.naive_bayes import MultinomialNB

nb_model = MultinomialNB()
nb_model.fit(X_train, y_train)
# Predict using Naïve Bayes
y_pred_nb = nb_model.predict(X_test)

# Evaluate the Model's Performance
# Evaluate Naïve Bayes Model
from sklearn.metrics import accuracy_score, classification_report

# Evaluate Naïve Bayes Model
print("Naïve Bayes Accuracy:", accuracy_score(y_test, y_pred_nb) *
100)
print(classification_report(y_test, y_pred_nb))
```

```
Naïve Bayes Accuracy: 95.7487922705314
              precision    recall  f1-score   support

           0       0.97      0.97      0.97       742
           1       0.92      0.92      0.92       293

    accuracy                           0.96      1035
   macro avg       0.95      0.95      0.95      1035
weighted avg       0.96      0.96      0.96      1035
```

```python
# Function to Predict Spam or Ham
def predict_email(text, model):
    # Convert the input text into numerical form using the trained
vectorizer
    text_transformed = vectorizer.transform([text]).toarray()

    # Predict using the trained model
    prediction = model.predict(text_transformed)

    # Map numeric label back to text
    return "Spam" if prediction[0] == 1 else "Ham"

# Example: Test Custom Emails
custom_email_1 = "Congratulations! You won a free iPhone. Click here
to claim now."
custom_email_2 = "Hey, let's meet for coffee tomorrow."

# Predict using Naïve Bayes Model
print(f"Custom Email 1: {predict_email(custom_email_1, nb_model)}")
print(f"Custom Email 2: {predict_email(custom_email_2, nb_model)}")
```

```
Custom Email 1: Spam
Custom Email 2: Ham
```

```
custom_email = "Siddhi, you almost missed this opportunity!"
print(f"Prediction: {predict_email(custom_email, nb_model)}")
```

```
Prediction: Spam
```

```
custom_email = "I am confirming my participation to this bootcamp"
print(f"Prediction: {predict_email(custom_email, nb_model)}")
```

```
Prediction: Ham
```