```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.feature_extraction.text import CountVectorizer, TfidfTransformer
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```

```
data = pd.read_csv('email_spam (1)-IR.csv')
data.head()
```

	title	text	type
0	?? the secrets to SUCCESS	Hi James,\n\nHave you claim your complimentary...	spam
1	?? You Earned 500 GCLoot Points	\nalt_text\nCongratulations, you just earned\n...	not spam
2	?? Your GitHub launch code	Here's your GitHub launch code, @Mortyj420!\n ...	not spam
3	[The Virtual Reward Center] Re: ** Clarifications	Hello,\n \nThank you for contacting the Virtua...	not spam
4	10-1 MLB Expert Inside, Plus Everything You Ne...	Hey Prachanda Rawal,\n\nToday's newsletter is ...	spam

Next steps: Generate code with data | View recommended plots | New interactive sheet

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 84 entries, 0 to 83
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   title   84 non-null     object
 1   text    84 non-null     object
 2   type    84 non-null     object
dtypes: object(3)
memory usage: 2.1+ KB
```

```
data.describe()
```

	title	text	type
count	84	84	84
unique	78	82	2
top	English	Model Casting Call\nThank you for taking the t...	not spam
freq	3	2	58

```
data.dropna(inplace=True)
```

```
X_train, X_test, y_train, y_test = train_test_split(data['text'],data['type'],test_size=0.2,random_state=42)
```

```
vectorizer = CountVectorizer()
tfidf_transformer = TfidfTransformer()
X_train_count = vectorizer.fit_transform(X_train)
X_train_tfidf = tfidf_transformer.fit_transform(X_train_count)
```

```
clf = MultinomialNB()
clf.fit(X_train_tfidf,y_train)
```

```
▾ MultinomialNB ⓘ ?
MultinomialNB()
```

```
X_test_count = vectorizer.transform(X_test)
X_test_tfidf = tfidf_transformer.transform(X_test_count)
y_pred = clf.predict(X_test_tfidf)
```

```
conf_matrix = confusion_matrix(y_test,y_pred)
print('Confusion Matrix:\n',conf_matrix)

cr = classification_report(y_test,y_pred)
print('Classification Report:\n',cr)

acc = accuracy_score(y_test,y_pred)
print('Accuracy:',acc*100,'%')
```

```
Confusion Matrix:
 [[11  0]
 [ 6  0]]
Classification Report:
               precision    recall  f1-score   support

    not spam       0.65      1.00      0.79        11
        spam       0.00      0.00      0.00         6

    accuracy                           0.65        17
   macro avg       0.32      0.50      0.39        17
weighted avg       0.42      0.65      0.51        17

Accuracy: 64.70588235294117 %
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1531: UndefinedMetricWarning: Precision is ill-defined ar
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1531: UndefinedMetricWarning: Precision is ill-defined ar
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1531: UndefinedMetricWarning: Precision is ill-defined ar
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

```
e = ['Kindly acknowledge the email.']
e_count = vectorizer.transform(e)
e_tf = tfidf_transformer.transform(e_count)
p = clf.predict(e_tf)

if p[0] == 'spam':
  print('Spam Email')
elif p[0] == 'not spam':
  print('Ham Email')
else:
  print(p[0])
```

```
Ham Email
```

Start coding or generate with AI.