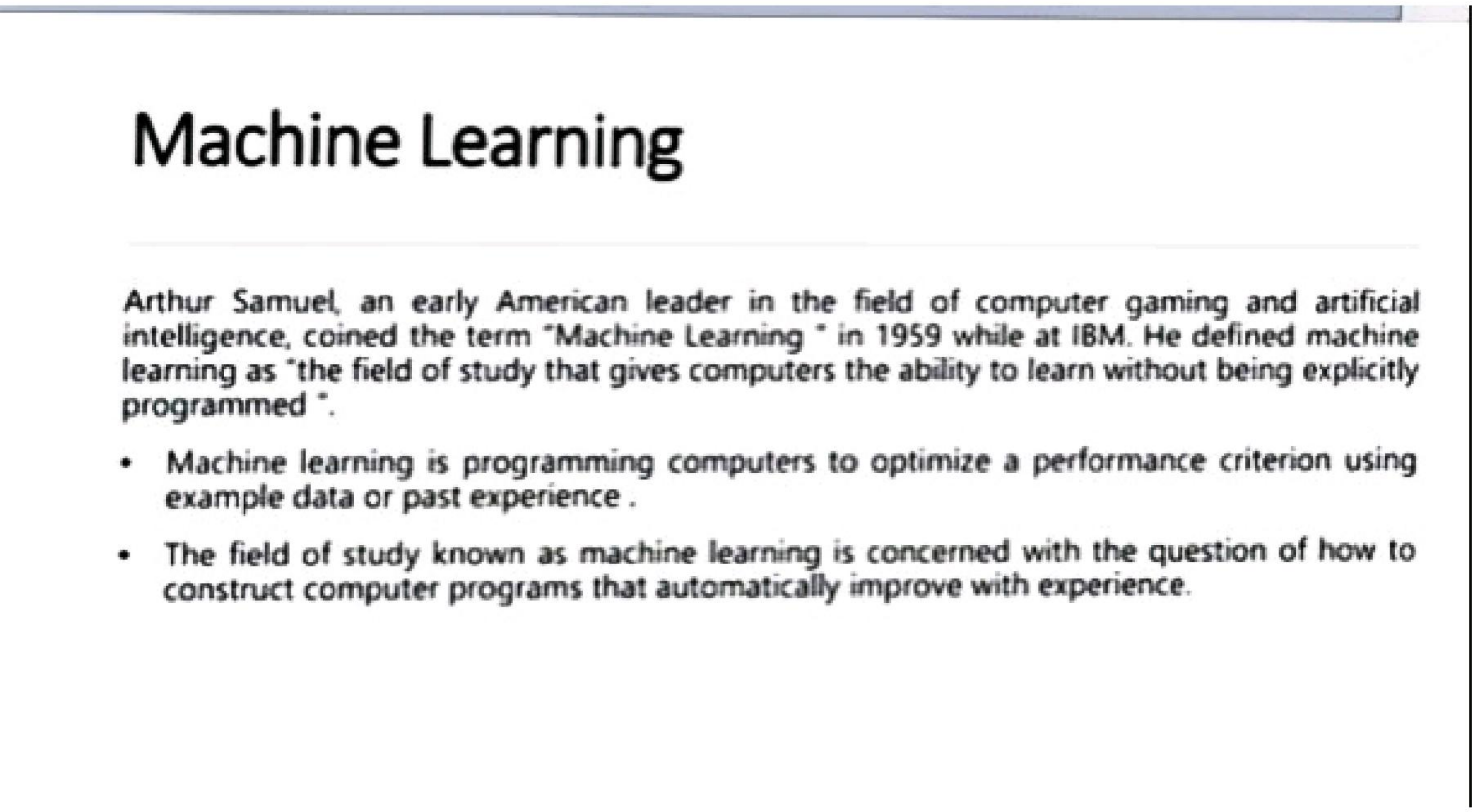
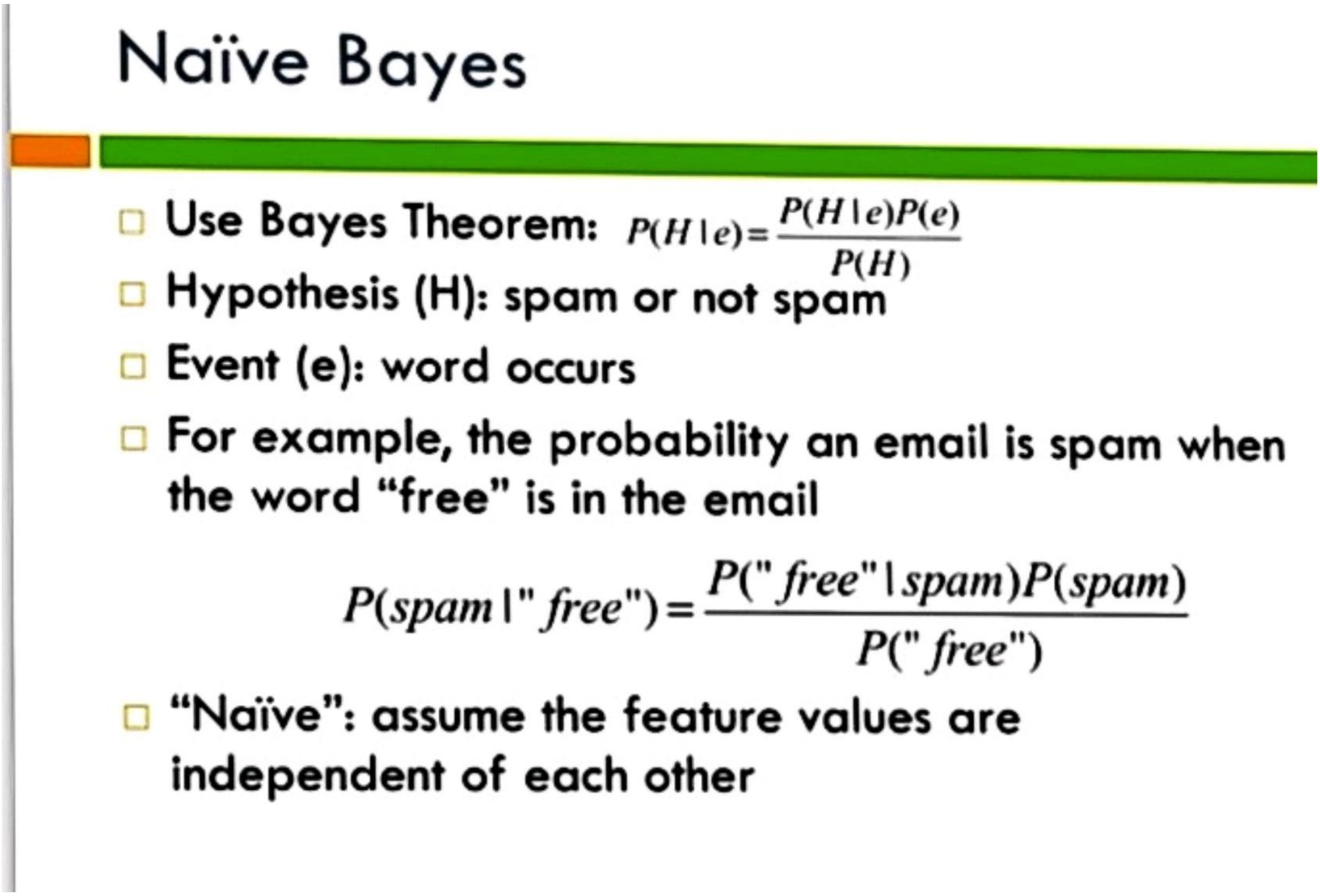
Spam Detection with Machine Learning



Machine Learning

Arthur Samuel, an early American leader in the field of computer gaming and artificial intelligence, coined the term "Machine Learning " in 1959 while at IBM. He defined machine learning as "the field of study that gives computers the ability to learn without being explicitly programmed ".

- Machine learning is programming computers to optimize a performance criterion using example data or past experience .
- The field of study known as machine learning is concerned with the question of how to construct computer programs that automatically improve with experience.

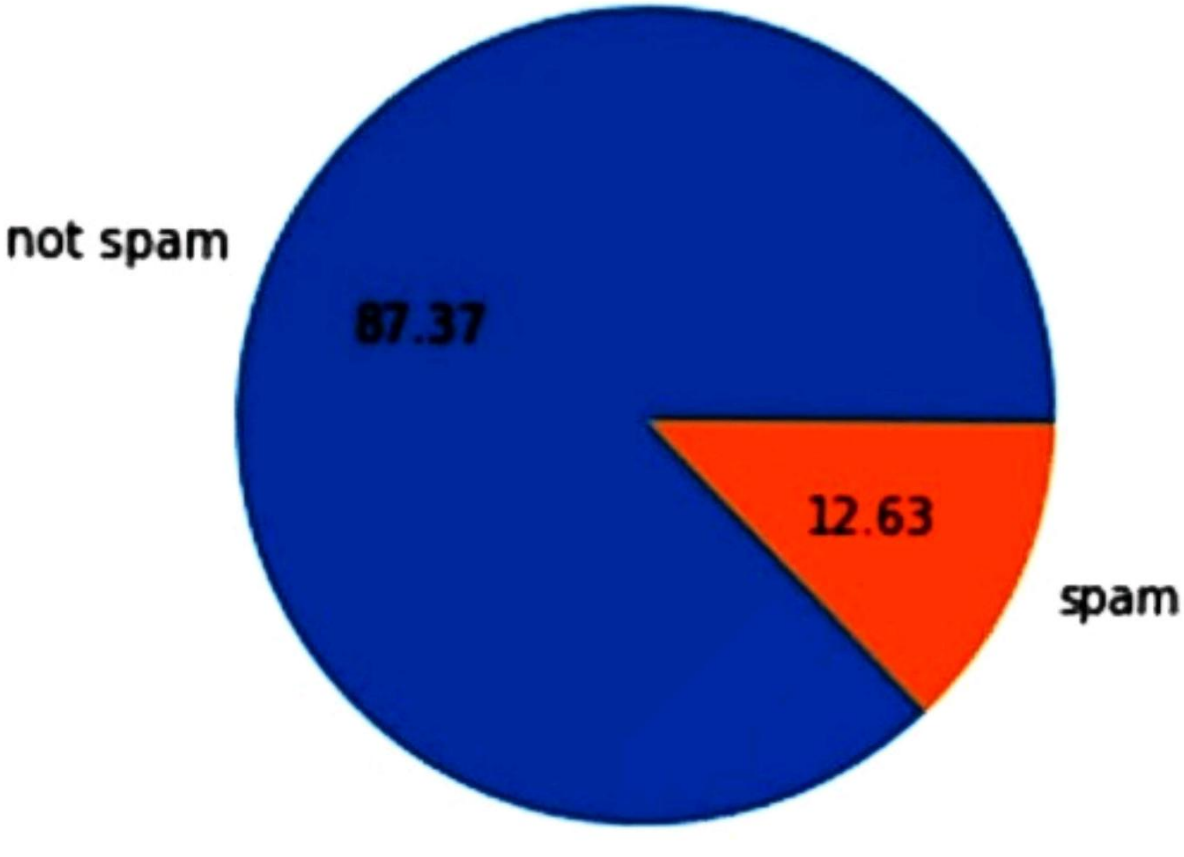


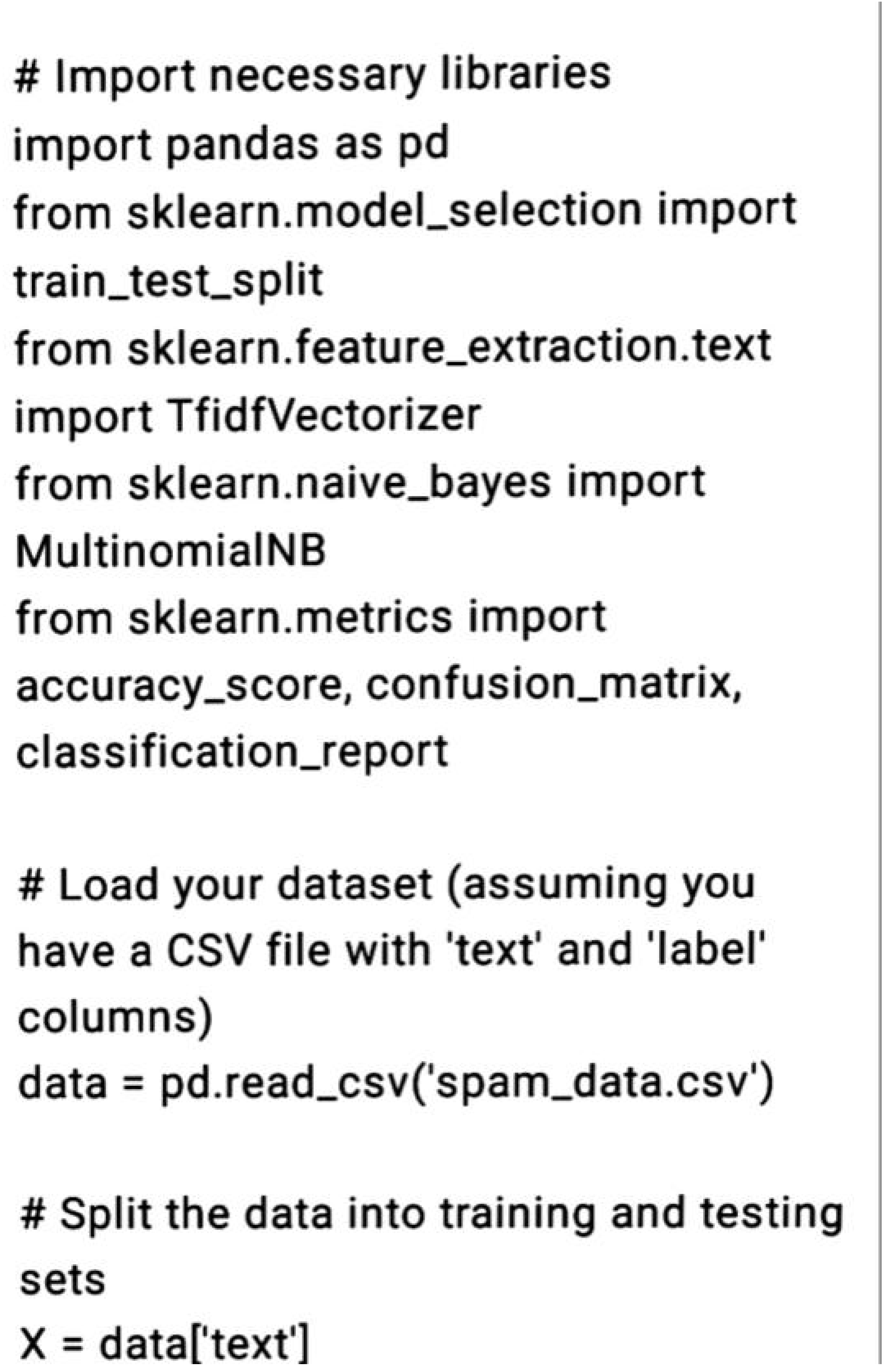
Naïve Bayes

- □ Use Bayes Theorem: $P(H \mid e) = \frac{P(H \mid e)P(e)}{P(H)}$
- □ Hypothesis (H): spam or not spam
- □ Event (e): word occurs
- □ For example, the probability an email is spam when the word “free” is in the email

$$P(spam \mid \text{"} free \text{"}) = \frac{P(\text{"} free \text{"} \mid spam)P(spam)}{P(\text{"} free \text{"})}$$

- □ “Naïve”: assume the feature values are independent of each other

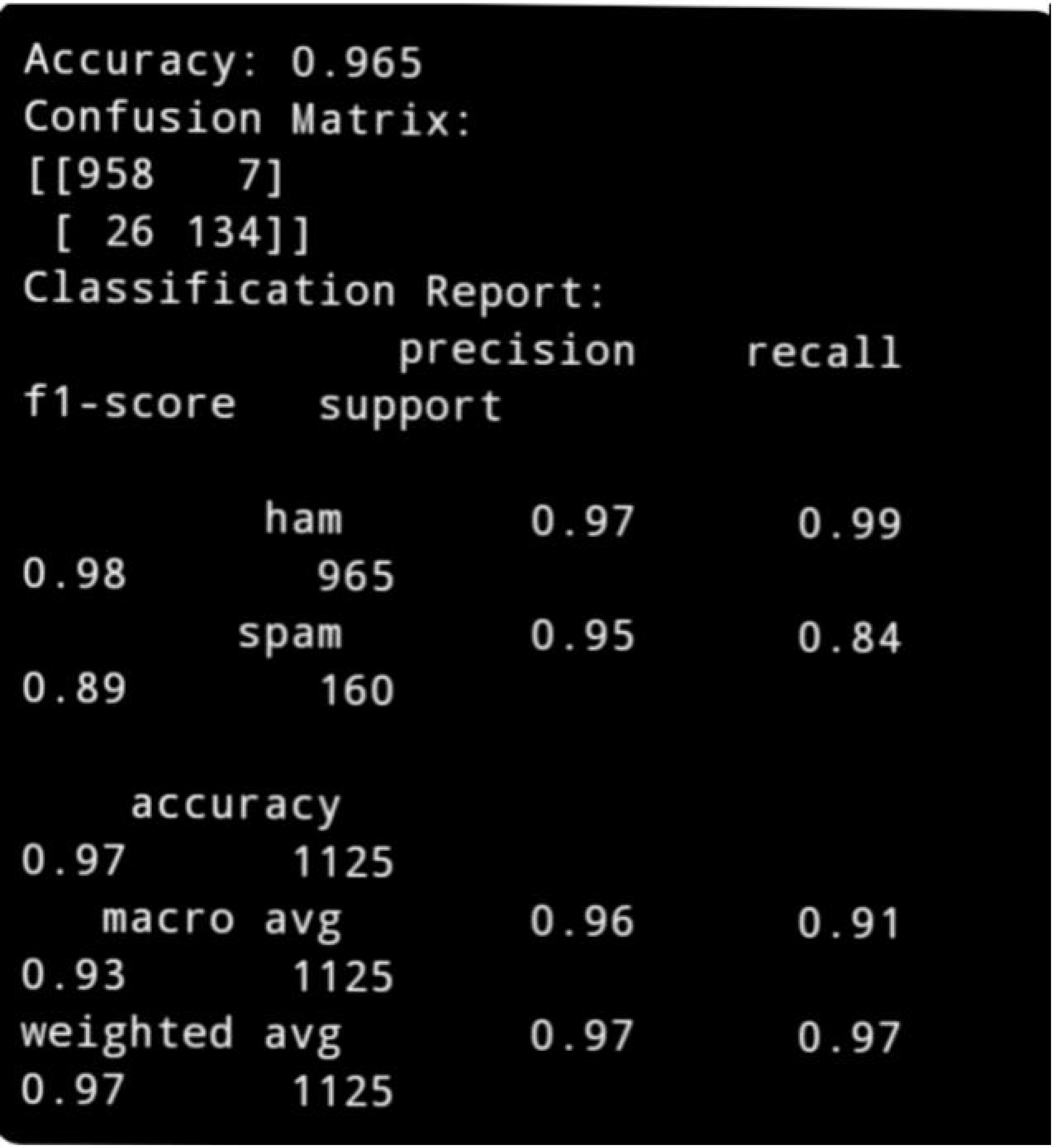

not spam
87.37
12.63
spam



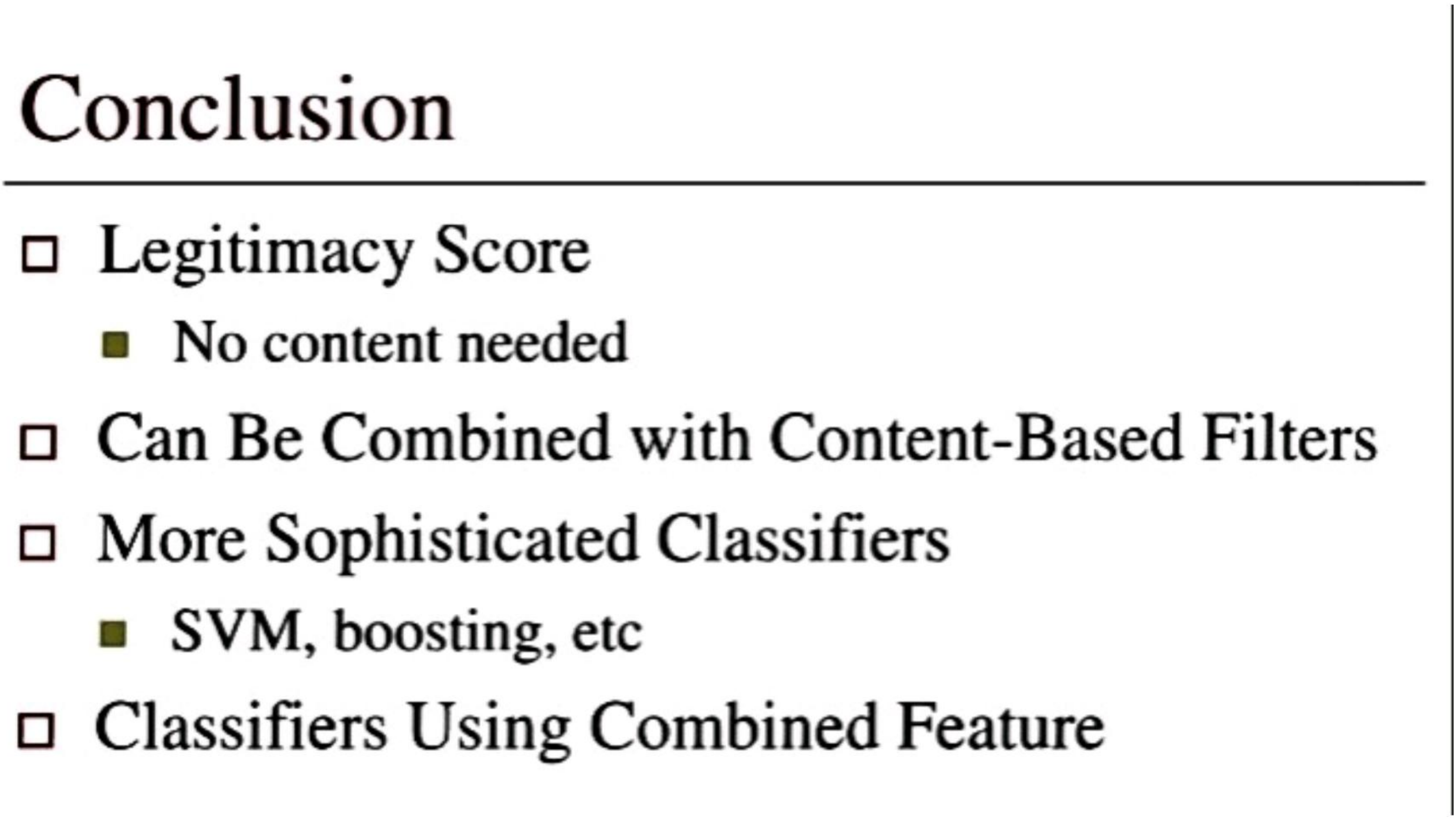
```
# Import necessary libraries
import pandas as pd
from sklearn.model_selection import
train_test_split
from sklearn.feature_extraction.text
import TfidfVectorizer
from sklearn.naive_bayes import
MultinomialNB
from sklearn.metrics import
accuracy_score, confusion_matrix,
classification_report

# Load your dataset (assuming you
have a CSV file with 'text' and 'label'
columns)
data = pd.read_csv('spam_data.csv')

# Split the data into training and testing
sets
X = data['text']
```

```
y = data['label']
X_train, X_test, y_train, y_test =
train_test_split(X, y, test_size=0.2,
random_state=42)

# Text vectorization using TF-IDF
tfidf_vectorizer = TfidfVectorizer()
X_train_tfidf =
tfidf_vectorizer.fit_transform(X_train)
X_test_tfidf =
tfidf_vectorizer.transform(X_test)

# Train a Naive Bayes classifier
spam_classifier = MultinomialNB()
spam_classifier.fit(X_train_tfidf, y_train)

# Make predictions on the test data
y_pred =
spam_classifier.predict(X_test_tfidf)
```

~~~
# Evaluate the classifier
accuracy = accuracy_score(y_test,
y_pred)
confusion = confusion_matrix(y_test,
y_pred)
report = classification_report(y_test,
y_pred)

print(f'Accuracy: {accuracy}')
print('Confusion Matrix:')
print(confusion)
print('Classification Report:')
print(report)
```
~~~



```
Accuracy: 0.965
Confusion Matrix:
[[958   7]
 [ 26 134]]
Classification Report:
              precision    recall
f1-score   support

         ham       0.97      0.99
0.98       965
        spam       0.95      0.84
0.89       160

    accuracy
0.97      1125
   macro avg       0.96      0.91
0.93      1125
weighted avg       0.97      0.97
0.97      1125
```



Conclusion

- □ Legitimacy Score
  - ■ No content needed
- □ Can Be Combined with Content-Based Filters
- □ More Sophisticated Classifiers
  - ■ SVM, boosting, etc
- □ Classifiers Using Combined Feature