Start coding or generate with AI.

```
import pandas as pd
import zipfile
import os
```

```
with zipfile.ZipFile("smsspamcollection.zip", "r") as zip_ref:
    zip_ref.extractall(".")

print("Files extracted:", os.listdir("."))
```

```
Files extracted: ['.config', 'readme', 'SMSSpamCollection', 'smsspamcollecti
```

```
data = pd.read_csv("SMSSpamCollection", sep="\t", header=None, names=
data.head()
```

	label	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...

```
from sklearn.model_selection import train_test_split

X = data["message"]   # features (the SMS text)
y = data["label"]     # target (ham/spam)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0
```

```
from sklearn.feature_extraction.text import TfidfVectorizer

vectorizer = TfidfVectorizer()
X_train_vec = vectorizer.fit_transform(X_train)
X_test_vec = vectorizer.transform(X_test)
```

```
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, classification_report

model = MultinomialNB()
model.fit(X_train_vec, y_train)
```

```
y_pred = model.predict(X_test_vec)

print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nReport:\n", classification_report(y_test, y_pred))
```

```
Accuracy: 0.9668161434977578

Report:
               precision    recall  f1-score   support

         ham       0.96      1.00      0.98       966
        spam       1.00      0.75      0.86       149

    accuracy                           0.97      1115
   macro avg       0.98      0.88      0.92      1115
weighted avg       0.97      0.97      0.96      1115
```

```
# Test your model with your own examples
examples = [
    "Congratulations! You have won $1000. Claim now!",
    "Hey, are we still meeting for lunch tomorrow?",
    "Free entry to win an iPhone, click here!",
    "Good morning, just checking in."
]

examples_vec = vectorizer.transform(examples)
predictions = model.predict(examples_vec)

for text, label in zip(examples, predictions):
    print(f"Message: {text}\nPredicted as: {label}\n")
```

```
Message: Congratulations! You have won $1000. Claim now!
Predicted as: spam

Message: Hey, are we still meeting for lunch tomorrow?
Predicted as: ham

Message: Free entry to win an iPhone, click here!
Predicted as: ham

Message: Good morning, just checking in.
Predicted as: ham
```

```
import joblib

# Save trained model & vectorizer
joblib.dump(model, "spam_model.pkl")
joblib.dump(vectorizer, "vectorizer.pkl")
```

```
['vectorizer.pkl']
```

```
from google.colab import files
files.download("spam_model.pkl")
files.download("vectorizer.pkl")
```

Spam Message Classifier (AI Project)

📌 Introduction

Spam messages are unwanted texts that can be annoying or harmful. This project builds a simple AI model to automatically classify SMS messages as **spam** or **ham** (not spam).

📊 Dataset

We used the **SMS Spam Collection Dataset** (5,574 messages).

- Labeled as either "ham" or "spam".
- Preprocessed with **TF-IDF vectorization**.

🧠 Model

- Algorithm: **Multinomial Naive Bayes**
- Training/Test Split: 80/20
- Tools: Python, scikit-learn, Google Colab

✅ Results

- Accuracy: **96%**
- Spam detection worked well, but one example ("Free entry to win an iPhone") was misclassified as ham.

🔮 Conclusion

This project shows how AI can detect spam effectively. Future improvements:

- Try deep learning (e.g., LSTM or BERT)
- Use larger, updated datasets
- Deploy the model in a simple web or mobile app

👨‍💻 Author

First AI Project Idea: *Spam Message Classifier*