```
# -------------- 1) IMPORTS & DATA CLEANING --------------
import pandas as pd
import string, re, nltk
from nltk.corpus import stopwords

# Download NLTK stop-word list (one-time per environment)
nltk.download('stopwords')

# Load the CSV that contains the raw text and sentiment labels
df = pd.read_csv("legal_sentiment_dataset.csv")  # Columns: Text, Sentiment

# Create a reusable cleaning function
stop_words = set(stopwords.words('english'))

def clean_text(text: str) -> str:
    text = text.lower()                                         # a) lowercase
    text = re.sub(r"\d+", "", text)                             # b) remove numbers
    text = text.translate(str.maketrans("", "", string.punctuation))  # c) strip punctuation
    text = " ".join(w for w in text.split() if w not in stop_words)   # d) remove stop-words
    return text

# Apply cleaning and store in new column
df["clean_text"] = df["Text"].apply(clean_text)

# Show a small sample
print(df.head(3))
```

```
                                                Text Sentiment  \
0  [Case 3708] The appeal is hereby dismissed for...  Negative
1  [Case 6994] The plaintiff's motion for summary...  Positive
2  [Case 2793] The legal claim is barred by the s...  Negative

                                        clean_text
0          case appeal hereby dismissed lack merit
1  case plaintiffs motion summary judgment granted
2       case legal claim barred statute limitations
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Pl
fc
ou
bl
tc
se
m
in
al
ne
fe
ti
ar
tr
ar
fe
nc
su
as
A
a
B
F
W
C

2
0
1

```python
# -------------- 2) TRAIN–TEST SPLIT & TF-IDF --------------
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer

# Separate features & labels
X = df["clean_text"]
y = df["Sentiment"]

# 80 % training / 20 % test
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.20, random_state=42, stratify=y
)

# TF-IDF converts text to numeric vectors
vectorizer = TfidfVectorizer(max_features=5000)   # top 5 000 terms
X_train_vec = vectorizer.fit_transform(X_train)   # fit + transform (train)
X_test_vec  = vectorizer.transform(X_test)        # transform only (test)

print("Training matrix shape:", X_train_vec.shape)
print("Example feature names:", vectorizer.get_feature_names_out()[:10])
```

```
Training matrix shape: (80, 72)
Example feature names: ['acquitted' 'agreement' 'amicable' 'appeal' 'barred' 'beyond' 'breach'
 'case' 'caused' 'charges']
```

```python
# -------------- 3) MODEL, METRICS, PREDICTION, SAVE --------------
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report
import pickle

# a) Train a binary classifier
model = LogisticRegression(max_iter=1000)   # more iterations for convergence
model.fit(X_train_vec, y_train)

# b) Evaluate on the held-out test set
y_pred = model.predict(X_test_vec)
```

```
print("Accuracy on test set:", accuracy_score(y_test, y_pred))
print("\nDetailed classification report:\n", classification_report(y_test, y_pred))

# c) Helper for predictions on new sentences
def predict_sentiment(raw_text: str) -> str:
    cleaned = clean_text(raw_text)
    vec = vectorizer.transform([cleaned])
    return model.predict(vec)[0]

print("\nSample inference:",
      predict_sentiment("The injunction is granted to prevent further harm."))

# d) Persist model + vectorizer for later reuse
with open("legal_sentiment_model.pkl", "wb") as f:
    pickle.dump((model, vectorizer), f)
print("\nModel saved as legal_sentiment_model.pkl")
```

```
Accuracy on test set: 1.0

Detailed classification report:
               precision    recall  f1-score   support

    Negative       1.00      1.00      1.00         9
    Positive       1.00      1.00      1.00        11

    accuracy                           1.00        20
   macro avg       1.00      1.00      1.00        20
weighted avg       1.00      1.00      1.00        20


Sample inference: Positive

Model saved as legal_sentiment_model.pkl
```

2
0
1

2
0
0

Py
pa
up

Py
pa

in

2
0
1

Py
pa
up

Py
pa
in

2
1
1

Py
pa

u

Py
pa
in

Li
ar
dr
up

2
0
2

Py
pa
up

Py
pa
in

**2
0
2**

2
0

2

2
0
1

2
0
1

2
0
1

2
0
1

2
0
2

2
0
2

2
0
0

2
1
1

2
1
2

2
1
0

2
1
2

2
0
2

2
0
1

2
0
2

2
0
2

2
0
0

2
0
0

2
0
1

2
0
3

2
0
1

2
0
1

2
0
0

2
0
1

2
1
0

2
1
1

2
1
2

2
0
3

2
0
1

2
0
2

2
0
1

2
0
2

2
0
0

2
0
1

2
0
2

2
0
2

2
0
1

2
0
2

2
0
1

2
0
2

2
0
1

2
0
0

2
0
2