```python
import re
import string
import nltk
import pandas as pd
from nltk.corpus import stopwords
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report
```

```python
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
True
```

```python
tweets_data = {
    "text": [
        "Absolutely loved the new phone, it's amazing!",
        "Worst customer service experience ever!",
        "Feeling happy and excited about the concert tonight",
        "I am frustrated with the slow delivery",
        "The restaurant had excellent food and friendly staff",
        "Horrible app update, keeps crashing!",
        "Such a delightful day, everything went perfectly",
        "The product quality is terrible, very disappointed",
        "Great performance by the team, very proud!",
        "I am unhappy with this purchase, waste of money"
    ],
    "label": [
        "positive", "negative", "positive", "negative", "positive",
        "negative", "positive", "negative", "positive", "negative"
    ]
}
```

```python
tweets_df = pd.DataFrame(tweets_data)
print("Initial Data:\n", tweets_df.head(), "\n")
```

```
Initial Data:
                                                 text     label
0      Absolutely loved the new phone, it's amazing!  positive
1            Worst customer service experience ever!  negative
2  Feeling happy and excited about the concert to...  positive
3             I am frustrated with the slow delivery  negative
4  The restaurant had excellent food and friendly...  positive
```

```python
def preprocess_text(sentence):
    sentence = sentence.lower()
    sentence = re.sub(r"http\S+|www\S+", "", sentence)
    sentence = re.sub(r"@\w+", "", sentence)
    sentence = re.sub(r"#\w+", "", sentence)
    sentence = sentence.translate(str.maketrans('', '', string.punctuation))
    sentence = re.sub(r"\d+", "", sentence)
    words = [word for word in sentence.split() if word not in stopwords.words('english')]
    return " ".join(words)
```

```python
tweets_df["processed"] = tweets_df["text"].apply(preprocess_text)
print("Cleaned Text Samples:\n", tweets_df[["text", "processed"]].head(), "\n")
```

```
Cleaned Text Samples:
                                                 text  \
0      Absolutely loved the new phone, it's amazing!
1            Worst customer service experience ever!
2  Feeling happy and excited about the concert to...
3             I am frustrated with the slow delivery
4  The restaurant had excellent food and friendly...

                              processed
0       absolutely loved new phone amazing
```

```
                              processed
0        absolutely loved new phone amazing
1    worst customer service experience ever
2     feeling happy excited concert tonight
3                  frustrated slow delivery
4  restaurant excellent food friendly staff
```

```python
X_train, X_test, y_train, y_test = train_test_split(
    tweets_df["processed"], tweets_df["label"], test_size=0.2, random_state=42
)
```

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

```python
classifier = LogisticRegression(max_iter=1000)
classifier.fit(X_train_vec, y_train)
```

```
▾ LogisticRegression
LogisticRegression(max_iter=1000)
```

```python
y_pred = classifier.predict(X_test_vec)
print("Model Accuracy:", round(accuracy_score(y_test, y_pred) * 100, 2), "%\n")
print("Detailed Report:\n", classification_report(y_test, y_pred))
```

```
Model Accuracy: 50.0 %

Detailed Report:
               precision    recall  f1-score   support

    negative       0.50      1.00      0.67         1
    positive       0.00      0.00      0.00         1

    accuracy                           0.50         2
   macro avg       0.25      0.50      0.33         2
weighted avg       0.25      0.50      0.33         2

/usr/local/lib/python3.12/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedM
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.12/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedM
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.12/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedM
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

```python
new_samples = [
    "The service at the cafe was fantastic!",
    "I am really annoyed with the delayed shipment",
    "What a beautiful day, feeling great!",
    "The software update ruined my phone, very upset"
]
```

```python
new_processed = [preprocess_text(txt) for txt in new_samples]
new_features = tfidf.transform(new_processed)
new_results = classifier.predict(new_features)
```

```python
print("\nSentiment Predictions:")
for original, sentiment in zip(new_samples, new_results):
    print(f"'{original}' → {sentiment}")
```

```

Sentiment Predictions:
'The service at the cafe was fantastic!' → negative
'I am really annoyed with the delayed shipment' → negative
'What a beautiful day, feeling great!' → positive
'The software update ruined my phone, very upset' → negative
```