```
import pandas as pd
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 classification_report
import nltk
import re
```

```
nltk.download('stopwords')
from nltk.corpus import stopwords
```

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

```
data = {
    "tweet": [
        "I love this product! It's amazing 😍",
        "Worst experience ever. Totally disappointed.",
        "Not bad, could be better but okay overall.",
        "I am extremely happy with the service.",
        "This is terrible! I will never buy again."
    ],
    "label": ["positive", "negative", "neutral", "positive", "negative"]
}
df = pd.DataFrame(data)
```

```
df = pd.DataFrame(data)
```

```
# 2) Preprocessing function
def clean_text(text):
    text = re.sub(r'http\S+', '', text)        # remove links
    text = re.sub(r'@\w+', '', text)           # remove mentions
    text = re.sub(r'[^A-Za-z\s]', '', text)    # remove special chars
    text = text.lower()
    return text

df['clean_tweet'] = df['tweet'].apply(clean_text)
```

```
# 3) Train-test split
X = df['clean_tweet']
y = df['label']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

```
# 4) Text vectorization (TF-IDF)
vectorizer = TfidfVectorizer(stop_words=stopwords.words('english'))
X_train_vec = vectorizer.fit_transform(X_train)
X_test_vec = vectorizer.transform(X_test)
```

```
# 5) Train ML model (Logistic Regression)
model = LogisticRegression()
model.fit(X_train_vec, y_train)
```

```
▾ LogisticRegression
LogisticRegression()
```

```
# 6) Predictions
y_pred = model.predict(X_test_vec)
```

```
# 7) Evaluation
print("Classification Report:\n")
print(classification_report(y_test, y_pred))
```

```
Classification Report:

              precision    recall  f1-score   support

    negative       0.00      0.00      0.00       2.0
    positive       0.00      0.00      0.00       0.0

    accuracy                           0.00       2.0
   macro avg       0.00      0.00      0.00       2.0
weighted avg       0.00      0.00      0.00       2.0
```

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