```
import pandas as pd
import re
import nltk
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
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

# Download stopwords
nltk.download('stopwords')
stop_words = set(stopwords.words('english'))
stemmer = PorterStemmer()

def preprocess_text(text):
    text = re.sub(r'[^\w\s]', '', text)  # Remove punctuation
    text = text.lower()  # Convert to lowercase
    text = ' '.join([stemmer.stem(word) for word in text.split() if word not in stop_words])
    return text

# Sample dataset with neutral sentiment included
data = {'review': [
    'This product is amazing!',
    'I hate this product.',
    'It is okay.',
    'The best thing ever!',
    'Terrible experience.',
    'Not bad but not great either.',
    'Absolutely fantastic!',
    'Worst purchase I have ever made.',
    'Just average, nothing special.',
    'Could be better, could be worse.'
    ],
    'sentiment': [1, 0, 2, 1, 0, 2, 1, 0, 2, 2]
}  # 1: Positive, 0: Negative, 2: Neutral

df = pd.DataFrame(data)
df['review'] = df['review'].apply(preprocess_text)

# Split data
X = df['review']
y = df['sentiment']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

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

# Logistic Regression model
model = LogisticRegression(multi_class='multinomial', solver='lbfgs')
model.fit(X_train_tfidf, y_train)

# Predictions
y_pred = model.predict(X_test_tfidf)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy}")
print(classification_report(y_test, y_pred))

# Example prediction
new_review = "This is a decent product, not the best but okay."
new_review_processed = preprocess_text(new_review)
new_review_tfidf = tfidf_vectorizer.transform([new_review_processed])
sentiment = model.predict(new_review_tfidf)
sentiment_labels = {0: 'Negative', 1: 'Positive', 2: 'Neutral'}
print(f"Sentiment of '{new_review}': {sentiment_labels[sentiment[0]]}")
```

```
Accuracy: 0.5
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         1
           1       0.00      0.00      0.00         0
           2       1.00      1.00      1.00         1
```

```
    accuracy                           0.50         2
   macro avg       0.33      0.33      0.33         2
weighted avg       0.50      0.50      0.50         2

Sentiment of 'This is a decent product, not the best but okay.': Positive
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
/usr/local/lib/python3.11/dist-packages/sklearn/linear_model/_logistic.py:1247: FutureWarning:
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetr
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetr
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetr
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetr
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetr
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetr
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```