```
import pandas as pd
import numpy as np
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split,
GridSearchCV
from sklearn.linear_model import Perceptron,
LogisticRegression
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.pipeline import Pipeline
from sklearn.metrics import accuracy_score,
classification_report
import nltk
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
import re

# Ensure NLTK resources are downloaded
nltk.download('stopwords', quiet=True)

# Data cleaning and preprocessing
porter = PorterStemmer()

def preprocessor(text):
    """
    Clean the text by removing HTML tags and non-word
characters,
    convert to lowercase
    """
    # Remove HTML tags
    text = re.sub('<[^>]*>', '', text)
    # Remove non-word characters and convert to lowercase
    text = re.sub('[\W]+', ' ', text.lower())
    return text

def tokenizer_porter(text):
    """
    Tokenize and stem the text using Porter Stemmer
    """
    return [porter.stem(word) for word in text.split()]
```

```
# Get English stopwords
stop = stopwords.words('english')

# Read the IMDb dataset
df = pd.read_csv(r'C:\Users\HUANG\Desktop\movie_data.csv')

# Preprocess the reviews
df['review'] = df['review'].apply(preprocessor)

# Separate features and target
X = df['review']
y = df['sentiment']

# Split the dataset (50/50 train/test with stratification)
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.5, random_state=1, stratify=y)

# Define TF-IDF Vectorizer
tfidf = TfidfVectorizer(
    strip_accents=None,
    lowercase=False,
    preprocessor=None,
    tokenizer=tokenizer_porter,
    stop_words=stop
)

# Define models and their parameter grids
models = {
    "Perceptron": (
        Perceptron(),
        {
            'clf__penalty': [None, 'l1', 'l2'],
            'clf__alpha': [0.0001, 0.001, 0.01]
        }
    ),
    "Logistic Regression": (
        LogisticRegression(solver='liblinear'),
        {
            'clf__penalty': ['l1', 'l2'],
            'clf__C': [1.0, 10.0, 100.0]
        }
    ),
```

```
    "Support Vector Machine": (
        SVC(),
        {
            'clf__kernel': ['linear', 'rbf'],
            'clf__C': [1.0, 10.0, 100.0]
        }
    ),
    "Decision Tree": (
        DecisionTreeClassifier(),
        {
            'clf__criterion': ['gini', 'entropy'],
            'clf__max_depth': [None, 10, 50]
        }
    ),
    "Random Forest": (
        RandomForestClassifier(),
        {
            'clf__n_estimators': [10, 50, 100],
            'clf__max_depth': [None, 10, 50]
        }
    ),
    "K-Nearest Neighbors": (
        KNeighborsClassifier(),
        {
            'clf__n_neighbors': [3, 5, 7],
            'clf__p': [1, 2]
        }
    )
}

# Store results
results = []

# Evaluate each model
for model_name, (model, param_grid) in models.items():
    print(f"Processing: {model_name}")

    # Create pipeline
    pipeline = Pipeline([('vect', tfidf), ('clf', model)])

    # Perform Grid Search
    grid_search = GridSearchCV(
        pipeline,
```

```
            param_grid,
            scoring='accuracy',
            cv=5,
            n_jobs=-1
        )

        # Fit the model
        grid_search.fit(X_train, y_train)

        # Get best parameters
        best_params = grid_search.best_params_

        # Calculate accuracies
        train_acc = grid_search.best_score_
        best_model = grid_search.best_estimator_
        y_pred = best_model.predict(X_test)
        test_acc = accuracy_score(y_test, y_pred)

        # Print detailed classification report
        print(f"\nClassification Report for {model_name}:")
        print(classification_report(y_test, y_pred))

        # Store results
        results.append([
            model_name,
            train_acc,
            test_acc,
            best_params
        ])

        print(f"{model_name} completed!")

    # Convert results to DataFrame
    results_df = pd.DataFrame(
        results,
        columns=['Algorithm', 'Train Accuracy', 'Test
    Accuracy', 'Best Parameters']
    )

    # Display results
    print("\nFinal Results:")
Processing: Perceptron
```

```
C:\Anaconda3\envs\pyml-book\lib\site-packages\sklearn\feature_e
xtraction\text.py:396: UserWarning: Your stop_words may be inco
nsistent with your preprocessing. Tokenizing the stop words gen
erated tokens ['abov', 'ani', 'becaus', 'befor', 'doe', 'dure',
'ha', 'hi', "it'", 'onc', 'onli', 'ourselv', "she'", "shoul
d'v", 'themselv', 'thi', 'veri', 'wa', 'whi', "you'r", "you'v",
'yourselv'] not in stop_words.
  warnings.warn(
Classification Report for Perceptron:
              precision    recall  f1-score   support

           0       0.86      0.87      0.86     12500
           1       0.87      0.86      0.86     12500

    accuracy                           0.86     25000
   macro avg       0.86      0.86      0.86     25000
weighted avg       0.86      0.86      0.86     25000

Perceptron completed!
Processing: Logistic Regression
C:\Anaconda3\envs\pyml-book\lib\site-packages\sklearn\feature_e
xtraction\text.py:396: UserWarning: Your stop_words may be inco
nsistent with your preprocessing. Tokenizing the stop words gen
erated tokens ['abov', 'ani', 'becaus', 'befor', 'doe', 'dure',
'ha', 'hi', "it'", 'onc', 'onli', 'ourselv', "she'", "shoul
d'v", 'themselv', 'thi', 'veri', 'wa', 'whi', "you'r", "you'v",
'yourselv'] not in stop_words.
  warnings.warn(
Classification Report for Logistic Regression:
              precision    recall  f1-score   support

           0       0.90      0.88      0.89     12500
           1       0.89      0.90      0.89     12500

    accuracy                           0.89     25000
   macro avg       0.89      0.89      0.89     25000
weighted avg       0.89      0.89      0.89     25000

Logistic Regression completed!
Processing: Support Vector Machine
C:\Anaconda3\envs\pyml-book\lib\site-packages\sklearn\feature_e
xtraction\text.py:396: UserWarning: Your stop_words may be inco
nsistent with your preprocessing. Tokenizing the stop words gen
```

```
erated tokens ['abov', 'ani', 'becaus', 'befor', 'doe', 'dure',
'ha', 'hi', "it'", 'onc', 'onli', 'ourselv', "she'", "shoul
d'v", 'themselv', 'thi', 'veri', 'wa', 'whi', "you'r", "you'v",
'yourselv'] not in stop_words.
  warnings.warn(
Classification Report for Support Vector Machine:
              precision    recall  f1-score   support

           0       0.91      0.88      0.90     12500
           1       0.89      0.91      0.90     12500

    accuracy                           0.90     25000
   macro avg       0.90      0.90      0.90     25000
weighted avg       0.90      0.90      0.90     25000

Support Vector Machine completed!
Processing: Decision Tree
C:\Anaconda3\envs\pyml-book\lib\site-packages\sklearn\feature_e
xtraction\text.py:396: UserWarning: Your stop_words may be inco
nsistent with your preprocessing. Tokenizing the stop words gen
erated tokens ['abov', 'ani', 'becaus', 'befor', 'doe', 'dure',
'ha', 'hi', "it'", 'onc', 'onli', 'ourselv', "she'", "shoul
d'v", 'themselv', 'thi', 'veri', 'wa', 'whi', "you'r", "you'v",
'yourselv'] not in stop_words.
  warnings.warn(
Classification Report for Decision Tree:
              precision    recall  f1-score   support

           0       0.80      0.62      0.70     12500
           1       0.69      0.85      0.76     12500

    accuracy                           0.73     25000
   macro avg       0.75      0.73      0.73     25000
weighted avg       0.75      0.73      0.73     25000

Decision Tree completed!
Processing: Random Forest
C:\Anaconda3\envs\pyml-book\lib\site-packages\sklearn\feature_e
xtraction\text.py:396: UserWarning: Your stop_words may be inco
nsistent with your preprocessing. Tokenizing the stop words gen
erated tokens ['abov', 'ani', 'becaus', 'befor', 'doe', 'dure',
'ha', 'hi', "it'", 'onc', 'onli', 'ourselv', "she'", "shoul
d'v", 'themselv', 'thi', 'veri', 'wa', 'whi', "you'r", "you'v",
```

```
'yourselv'] not in stop_words.
  warnings.warn(
Classification Report for Random Forest:
              precision    recall  f1-score   support

           0       0.85      0.86      0.85     12500
           1       0.86      0.85      0.85     12500

    accuracy                           0.85     25000
   macro avg       0.85      0.85      0.85     25000
weighted avg       0.85      0.85      0.85     25000

Random Forest completed!
Processing: K-Nearest Neighbors
C:\Anaconda3\envs\pyml-book\lib\site-packages\sklearn\feature_e
xtraction\text.py:396: UserWarning: Your stop_words may be inco
nsistent with your preprocessing. Tokenizing the stop words gen
erated tokens ['abov', 'ani', 'becaus', 'befor', 'doe', 'dure',
'ha', 'hi', "it'", 'onc', 'onli', 'ourselv', "she'", "shoul
d'v", 'themselv', 'thi', 'veri', 'wa', 'whi', "you'r", "you'v",
'yourselv'] not in stop_words.
  warnings.warn(
C:\Anaconda3\envs\pyml-book\lib\site-packages\sklearn\neighbors
\_classification.py:228: FutureWarning: Unlike other reduction
functions (e.g. `skew`, `kurtosis`), the default behavior of `m
ode` typically preserves the axis it acts along. In SciPy 1.11.
0, this behavior will change: the default value of `keepdims` w
ill become False, the `axis` over which the statistic is taken
will be eliminated, and the value None will no longer be accept
ed. Set `keepdims` to True or False to avoid this warning.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
Classification Report for K-Nearest Neighbors:
              precision    recall  f1-score   support

           0       0.80      0.72      0.76     12500
           1       0.75      0.82      0.78     12500

    accuracy                           0.77     25000
   macro avg       0.77      0.77      0.77     25000
weighted avg       0.77      0.77      0.77     25000

K-Nearest Neighbors completed!
```

```
Final Results:
                Algorithm  Train Accuracy  Test Accuracy  \
0              Perceptron         0.85996        0.86304
1     Logistic Regression         0.88636        0.89224
2  Support Vector Machine         0.89228        0.89664
3           Decision Tree         0.72524        0.73312
4           Random Forest         0.84960        0.85284
5     K-Nearest Neighbors         0.77556        0.77076

                                     Best Parameters
0       {'clf__alpha': 0.0001, 'clf__penalty': None}
1             {'clf__C': 10.0, 'clf__penalty': 'l2'}
2              {'clf__C': 1.0, 'clf__kernel': 'rbf'}
3   {'clf__criterion': 'gini', 'clf__max_depth': 10}
4  {'clf__max_depth': None, 'clf__n_estimators': ...
5              {'clf__n_neighbors': 7, 'clf__p': 2}
```

In [ ]: