```
import pandas as pd
import numpy as np
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, confusion_matrix
import seaborn as sns
import pickle

# Load dataset (replace with actual dataset path)
df = pd.read_csv('amazon_cell_phone_reviews.csv')  # Replace with your dataset path

# Assuming the dataset has 'review' column for text data and 'label' column for sentiment (positive/negative)
X = df['review']  # Features (text data)
y = df['label']  # Labels (positive/negative sentiment)

# Split dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Apply CountVectorizer (Bag of Words model)
count_vectorizer = CountVectorizer(stop_words='english', max_features=5000)  # Limit features for performance
count_train = count_vectorizer.fit_transform(X_train)
count_test = count_vectorizer.transform(X_test)

# Apply TF-IDF Vectorizer
tfidf_vectorizer = TfidfVectorizer(stop_words='english', max_features=5000)  # Limit features for performance
tfidf_train = tfidf_vectorizer.fit_transform(X_train)
tfidf_test = tfidf_vectorizer.transform(X_test)

# Train Naive Bayes classifier with Count Vectorizer data
nb_classifier = MultinomialNB()  # Using default alpha=1.0
nb_classifier.fit(count_train, y_train)

# Predict on test data (Count Vectorizer)
pred = nb_classifier.predict(count_test)
score = accuracy_score(y_test, pred)  # Calculate accuracy
print(f'Accuracy (Count Vectorizer): {score:.4f}')  # Print accuracy score

# Confusion matrix (Count Vectorizer)
cm = confusion_matrix(y_test, pred, labels=['positive', 'negative'])
print(f'Confusion Matrix (Count Vectorizer):\n{cm}')  # Print confusion matrix
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')  # Visualize confusion matrix as a heatmap
sns.plt.show()

# Train Naive Bayes classifier with TF-IDF data
nb_classifier.fit(tfidf_train, y_train)

# Predict on test data (TF-IDF Vectorizer)
pred = nb_classifier.predict(tfidf_test)
score = accuracy_score(y_test, pred)  # Calculate accuracy
print(f'Accuracy (TF-IDF Vectorizer): {score:.4f}')  # Print accuracy score

# Confusion matrix (TF-IDF Vectorizer)
cm = confusion_matrix(y_test, pred, labels=['positive', 'negative'])
print(f'Confusion Matrix (TF-IDF Vectorizer):\n{cm}')  # Print confusion matrix
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')  # Visualize confusion matrix as a heatmap
sns.plt.show()

# Test different alpha values for Naive Bayes with TF-IDF data
alphas = [0.1, 0.5, 1.0, 2.0, 5.0]  # Different alpha values
for alpha in alphas:
    nb_classifier = MultinomialNB(alpha=alpha)  # Create model with different alpha
    nb_classifier.fit(tfidf_train, y_train)  # Fit the model
    pred = nb_classifier.predict(tfidf_test)  # Predict
    score = accuracy_score(y_test, pred)  # Calculate accuracy
    print(f'Alpha: {alpha}, Accuracy (TF-IDF): {score:.4f}')  # Print results

# Optional: Save the trained model and vectorizer for later use
with open('nb_classifier.pkl', 'wb') as model_file:
    pickle.dump(nb_classifier, model_file)

with open('tfidf_vectorizer.pkl', 'wb') as vectorizer_file:
    pickle.dump(tfidf_vectorizer, vectorizer_file)
```

```
Start coding or generate with AI.
```

```
Start coding or generate with AI.
```

```
Start coding or generate with AI.
```

```
Start coding or generate with AI.
```

```
Start coding or generate with AI.
```