```
X_train = ["This was really awesome an awesome movie",
           "Great movie! Ilikes it a lot",
           "Happy Ending! Awesome Acting by hero",
           "loved it!",
           "Bad not upto the mark",
           "Could have been better",
           "really Dissapointed by the movie"]
# X_test = "it was really awesome and really disspntd"

y_train = ["positive","positive","positive","positive","negative","negative","negative"] # 1- Positive class, 0- negative cla



X_train # Reviews
```

```
['This was awesome an awesome movie',
 'Great movie! Ilikes it a lot',
 'Happy Ending! Awesome Acting by hero',
 'loved it!',
 'Bad not upto the mark',
 'Could have been better',
 'Dissapointed by the movie']
```

Cleaning of the data

```
# Tokenize
# "I am a python dev" -> ["I", "am", "a", "python", "dev"]


from nltk.tokenize import RegexpTokenizer
# NLTK -> Tokenize -> RegexpTokenizer


# Stemming
# "Playing" -> "Play"
# "Working" -> "Work"


from nltk.stem.porter import PorterStemmer
# NLTK -> Stem -> Porter -> PorterStemmer

from nltk.corpus import stopwords
# NLTK -> Corpus -> stopwords


# Downloading the stopwords
import nltk
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
True
```

```
tokenizer = RegexpTokenizer(r"\w+")
en_stopwords = set(stopwords.words('english'))
ps = PorterStemmer()


def getCleanedText(text):
  text = text.lower()

  # tokenizing
  tokens = tokenizer.tokenize(text)
  new_tokens = [token for token in tokens if token not in en_stopwords]
  stemmed_tokens = [ps.stem(tokens) for tokens in new_tokens]
  clean_text = " ".join(stemmed_tokens)
  return clean_text
```

Input from the user

```
X_test = ["it was bad"]


X_clean = [getCleanedText(i) for i in X_train]
xt_clean = [getCleanedText(i) for i in X_test]
```

```
X_clean
```

```
['awesom awesom movi',
 'great movi ilik lot',
 'happi end awesom act hero',
 'love',
 'bad upto mark',
 'could better',
 'dissapoint movi']
```

```
# Data before cleaning
'''
X_train = ["This was awesome an awesome movie",
           "Great movie! Ilikes it a lot",
           "Happy Ending! Awesome Acting by hero",
           "loved it!",
           "Bad not upto the mark",
           "Could have been better",
           "Dissapointed by the movie"]
'''
```

```
'\nX_train = ["This was awesome an awesome movie",\n           "Great movie!
Ilikes it a lot",\n           "Happy Ending! Awesome Acting by hero",\n
"loved it!",\n           "Bad not upto the mark",\n           "Could have bee
```

Vectorize

```
from sklearn.feature_extraction.text import CountVectorizer
```

```
cv = CountVectorizer(ngram_range = (1,2))
# "I am PyDev" -> "i am", "am Pydev"
```

```
X_vec = cv.fit_transform(X_clean).toarray()
```

```
X_vec
```

```
array([[0, 0, 2, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 1, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1,
        1, 0, 0, 1, 1, 0, 0],
       [1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0,
        0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 1, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 1, 0, 0, 1, 1],
       [0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 1, 0, 0, 0]])
```

```
print(cv.get_feature_names_out())
```

```
['act' 'act hero' 'awesom' 'awesom act' 'awesom awesom' 'awesom movi'
 'bad' 'bad upto' 'better' 'could' 'could better' 'dissapoint'
 'dissapoint movi' 'end' 'end awesom' 'great' 'great movi' 'happi'
 'happi end' 'hero' 'ilik' 'ilik lot' 'lot' 'love' 'mark' 'movi'
 'movi ilik' 'realli' 'realli awesom' 'realli dissapoint' 'upto'
 'upto mark']
```

```
Xt_vect = cv.transform(xt_clean).toarray()
```

```
Xt_vect
```

```
array([[0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0]])
```

Multinomial Naive Bayes

```
from sklearn.naive_bayes import MultinomialNB
```

```
mn = MultinomialNB()
```

```
mn.fit(X_vec, y_train)
```

```
MultinomialNB()
```

```
y_pred = mn.predict(Xt_vect)
```

```
y_pred
```

```
array(['negative'], dtype='<U8')
```

```
import pandas as pd
import numpy as np
import csv
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, confusion_matrix

# Data
X_train = ["This was really awesome an awesome movie",
            "Great movie! I likes it a lot",
            "Happy Ending! Awesome Acting by hero",
            "loved it!",
            "Bad not upto the mark",
            "Could have been better",
            "really Dissapointed by the movie"]

y_train = ["positive", "positive", "positive", "positive", "negative", "negative", "negative"]

# Create DataFrame
df_train = pd.DataFrame({"Review": X_train, "Sentiment": y_train})

# Use CountVectorizer
cv = CountVectorizer(ngram_range=(1, 2))
X_vec_train = cv.fit_transform(df_train["Review"]).toarray()

# Train a Naive Bayes classifier
mn = MultinomialNB()
mn.fit(X_vec_train, df_train["Sentiment"])

# Predict on the training data
Y_pred_train = mn.predict(X_vec_train)

# Calculate and print the accuracy on the training data
accuracy_train = accuracy_score(df_train["Sentiment"], Y_pred_train)
print("Accuracy on training data:", accuracy_train)

# Confusion Matrix
conf_matrix = confusion_matrix(df_train["Sentiment"], Y_pred_train, labels=["positive", "negative"])

# Plot the confusion matrix
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues", xticklabels=["positive", "negative"], yticklabels=["positive", "n
plt.xlabel("Predicted")
plt.ylabel("True")
plt.title("Confusion Matrix")

# Move plt.show() outside the indented block
plt.show()
```

```
Accuracy on training data: 1.0
```