```
X_train = ["This was really an amazing movie",
           "Great movie! I liked 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 class
```

```
X_train # Reviews
```

```
['This was really an amazing movie',
 'Great movie! I liked 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']
```

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]   Unzipping corpora/stopwords.zip.
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
```

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

```
xt_clean
```

```
['bad']
```

```
# 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 Actin
g by hero",\n           "loved it!",\n           "Bad not upto the mark",\n           "Could have been better",\n           "Dissapoint
ed by the movie"]\n'
```

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

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

```
X_clean
```

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

```
xt_clean
```

```
['good']
```

```
# Data before cleaning
'''
X_train = ["This was really an amazing movie",
           "Great movie! I liked 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"]
'''
```

```
'\nX_train = ["This was really an amazing movie",\n           "Great movie! I liked it
a lot",\n           "Happy Ending! Awesome Acting by hero",\n           "Loved it!",\n
"Bad not upto the mark",\n           "Could have been better",\n           "really Diss
```

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, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 1, 0, 1, 1, 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, 0, 0, 0],
       [1, 1, 0, 0, 1, 1, 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, 0, 0,
        0, 1, 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, 0, 0,
        0, 0, 1, 0, 0, 0, 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, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 1, 0, 1, 0, 1, 0, 0]])
```

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

```
Xt_vect
```

```
array([[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]])
```

Multinomial Naive Bayes

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

```
mn = MultinomialNB()
```

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

```
▾ MultinomialNB
MultinomialNB()
```

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

```
y_pred
```

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