In [1]:

```
import pandas as pd
```

In [2]:

```
df = pd.read_csv('Downloads/amazon_alexa.csv')
df.head(10)
```

Out[2]:

	reviews	sentiment
0	Love my Echo!	1
1	Loved it!	1
2	Sometimes while playing a game, you can answer...	1
3	I have had a lot of fun with this thing. My 4 ...	1
4	Music	1
5	I received the echo as a gift. I needed anothe...	1
6	Without having a cellphone, I cannot use many ...	1
7	I think this is the 5th one I've purchased. I'...	1
8	looks great	1
9	Love it! I've listened to songs I haven't hear...	1

In [3]:

```
df.isnull().sum()
df.dropna(inplace=True)
```

In [4]:

```
import nltk
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
from nltk.stem import WordNetLemmatizer
import string
```

In [5]:

```
# Tokenizing words
nltk.download('punkt')
df['reviews'] = df['reviews'].apply(nltk.word_tokenize)
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\jagap\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

In [6]:

```
df['reviews'] = df['reviews'].apply(lambda x: [word.lower() for word in x])
```

In [7]:

```
# Removing Punctuations
nltk.download('stopwords')
punct = string.punctuation
df['reviews'] = df['reviews'].apply(lambda x: [word for word in x if word not
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\jagap\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

In [8]:
```
# Removing Stop words
stop_words = set(stopwords.words('english'))
df['reviews'] = df['reviews'].apply(lambda x: [word for word in x if word not
```

In [9]:
```
# Stemming or Lemmatizing the words
nltk.download('wordnet')
porter_stemmer = PorterStemmer()
wordnet_lemmatizer = WordNetLemmatizer()
df['reviews'] = df['reviews'].apply(lambda x: [porter_stemmer.stem(word) for w
```

```
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\jagap\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

In [10]:
```
from sklearn.feature_extraction.text import CountVectorizer
```

In [11]:
```
# Using Count Vectorizer
count_vectorizer = CountVectorizer()
X = count_vectorizer.fit_transform(df['reviews'].apply(lambda x: ' '.join(x)))
```

In [12]:
```
from sklearn.model_selection import train_test_split
y = df['sentiment']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, rando
```

In [13]:
```
from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix, classification_report
```

In [14]:
```
nb = MultinomialNB()
lr = LogisticRegression()
knn = KNeighborsClassifier()
```

In [15]:

```
models = {'MultinomialNB': nb, 'Logistic Regression': lr, 'KNN': knn}
for name, model in models.items():
    print(name)
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    print(confusion_matrix(y_test, y_pred))
    print(classification_report(y_test, y_pred))
```

```
MultinomialNB
[[ 17  41]
 [ 11 561]]
              precision    recall  f1-score   support

           0       0.61      0.29      0.40        58
           1       0.93      0.98      0.96       572

    accuracy                           0.92       630
   macro avg       0.77      0.64      0.68       630
weighted avg       0.90      0.92      0.90       630

Logistic Regression
[[ 19  39]
 [  3 569]]
              precision    recall  f1-score   support

           0       0.86      0.33      0.48        58
           1       0.94      0.99      0.96       572

    accuracy                           0.93       630
   macro avg       0.90      0.66      0.72       630
weighted avg       0.93      0.93      0.92       630

KNN
[[  2  56]
 [  0 572]]
              precision    recall  f1-score   support

           0       1.00      0.03      0.07        58
           1       0.91      1.00      0.95       572

    accuracy                           0.91       630
   macro avg       0.96      0.52      0.51       630
weighted avg       0.92      0.91      0.87       630


C:\Users\jagap\anaconda3\lib\site-packages\sklearn\neighbors\_classification.
py:228: FutureWarning: Unlike other reduction functions (e.g. `skew`, `kurtos
is`), the default behavior of `mode` typically preserves the axis it acts alo
ng. In SciPy 1.11.0, this behavior will change: the default value of `keepdim
s` will become False, the `axis` over which the statistic is taken will be el
iminated, and the value None will no longer be accepted. Set `keepdims` to Tr
ue or False to avoid this warning.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
```