In [19]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import re
import string
import nltk
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer, WordNetLemmatizer
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
```

In [20]:
```
df = pd.read_csv("AmazonAlexa_Reviews.csv")
```

In [21]:
```
df
```

Out[21]:

	rating	date	variation	verified_reviews	feedback
0	5	31-Jul-18	Charcoal Fabric	Love my Echo!	1
1	5	31-Jul-18	Charcoal Fabric	Loved it!	1
2	4	31-Jul-18	Walnut Finish	Sometimes while playing a game, you can answer...	1
3	5	31-Jul-18	Charcoal Fabric	I have had a lot of fun with this thing. My 4 ...	1
4	5	31-Jul-18	Charcoal Fabric	Music	1
...	...	...	...	...	...
3145	5	30-Jul-18	Black Dot	Perfect for kids, adults and everyone in betwe...	1
3146	5	30-Jul-18	Black Dot	Listening to music, searching locations, check...	1
3147	5	30-Jul-18	Black Dot	I do love these things, i have them running my...	1
3148	5	30-Jul-18	White Dot	Only complaint I have is that the sound qualit...	1
3149	4	29-Jul-18	Black Dot	Good	1

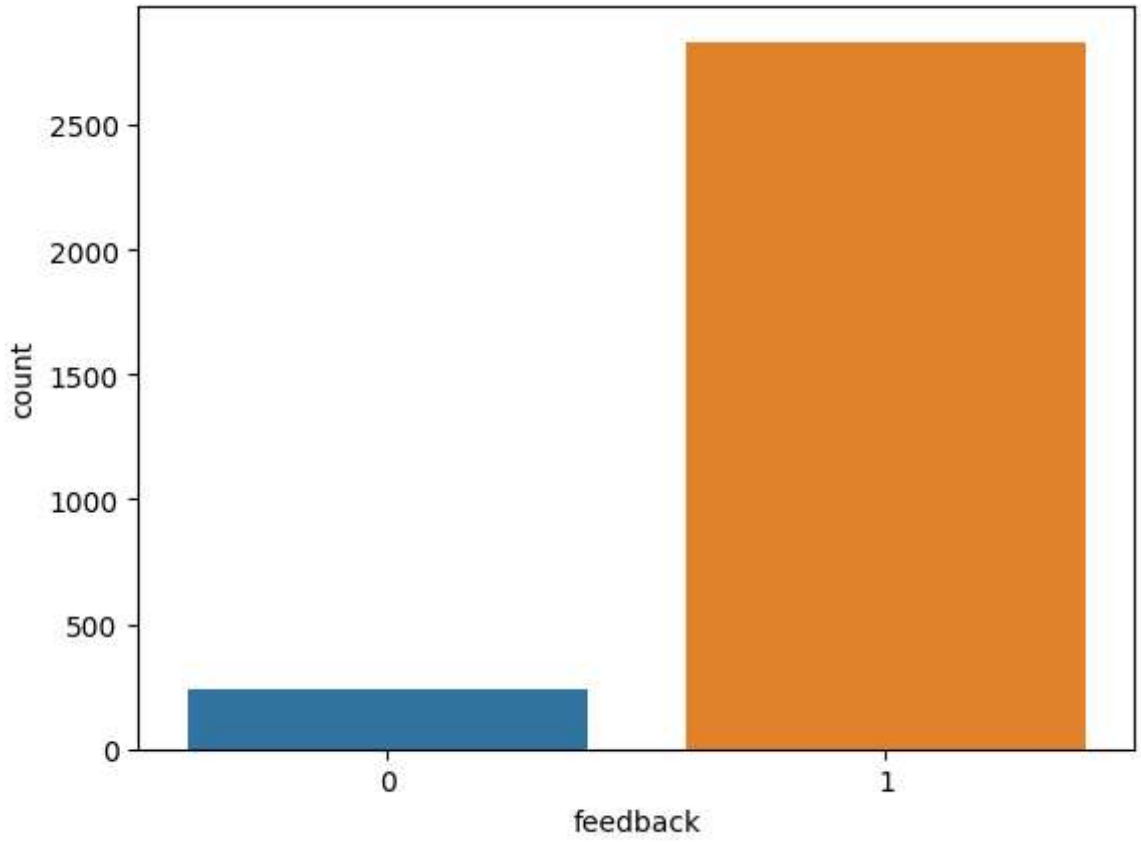
3150 rows × 5 columns

In [22]:
```
print(df.shape)
print(df.isnull().sum())
```

```
(3150, 5)
rating               0
date                 0
variation            0
verified_reviews    79
feedback             0
dtype: int64
```

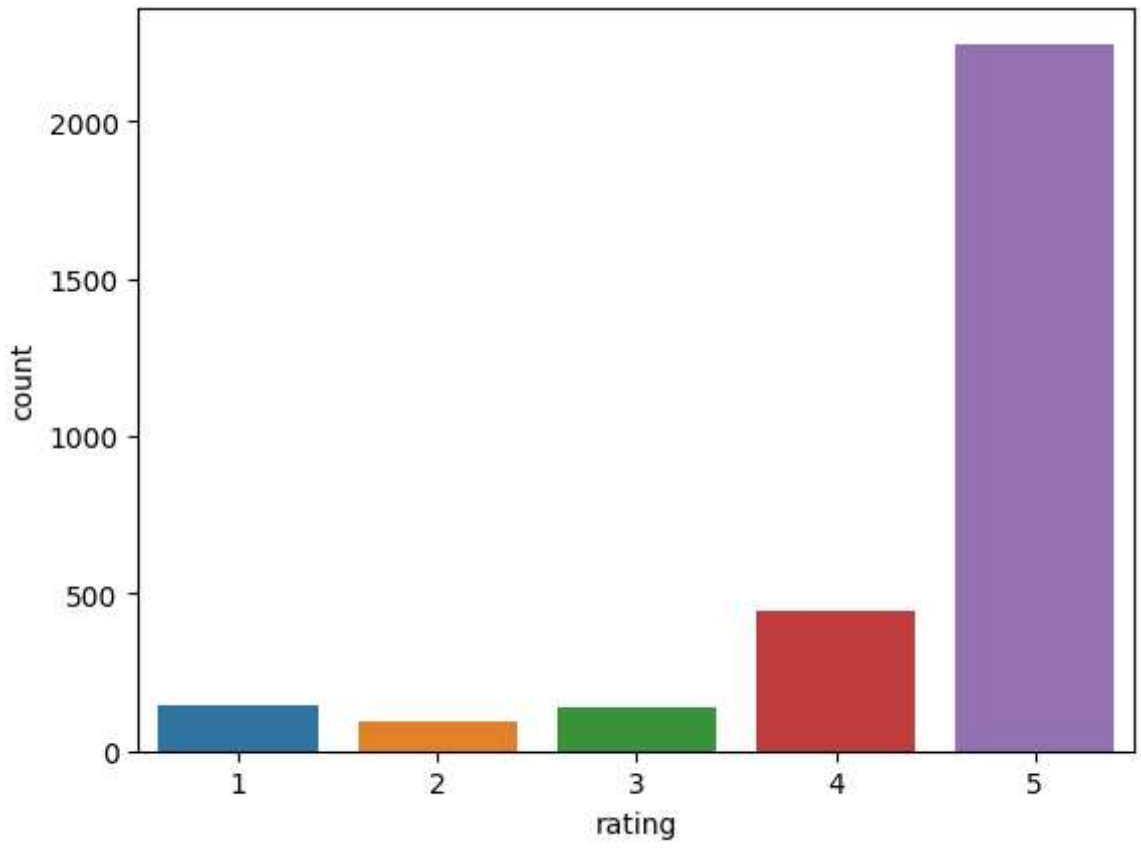
In [23]:
```
df.dropna(subset=['verified_reviews'], inplace=True)
```

In [24]:
```
sns.countplot(x="feedback", data=df)
plt.figure(figsize=(6,2))
plt.show()
```



<Figure size 600x200 with 0 Axes>

In [25]:
```
sns.countplot(x="rating", data=df)
plt.show()
```



In [26]:
```
df['verified_reviews'] = df['verified_reviews'].str.lower()
```

In [27]:
```
# Define a function to remove punctuation
def remove_punctuation(text):
    if isinstance(text, str):
        translator = str.maketrans("", "", string.punctuation)
        return text.translate(translator)
    return ""
```

```
# Apply the function to the "verified_reviews" column
df["verified_reviews"] = df["verified_reviews"].apply(remove_punctuation)
df.head()
```

Out[27]:

	rating	date	variation	verified_reviews	feedback
0	5	31-Jul-18	Charcoal Fabric	love my echo	1
1	5	31-Jul-18	Charcoal Fabric	loved it	1
2	4	31-Jul-18	Walnut Finish	sometimes while playing a game you can answer ...	1
3	5	31-Jul-18	Charcoal Fabric	i have had a lot of fun with this thing my 4 y...	1
4	5	31-Jul-18	Charcoal Fabric	music	1

In [28]:

```
def remove_emojis(text):
    emoji_pattern = re.compile("["
        u"\U0001F600-\U0001F64F"  # emoticons
        u"\U0001F300-\U0001F5FF"  # symbols & pictographs
        u"\U0001F680-\U0001F6FF"  # transport & map symbols
        u"\U0001F1E0-\U0001F1FF"  # flags (iOS)
                           "]+", flags=re.UNICODE)
    return emoji_pattern.sub(r"", text)

df["verified_reviews"] = df["verified_reviews"].apply(lambda x: remove_emojis(x))
df.head()
```

Out[28]:

	rating	date	variation	verified_reviews	feedback
0	5	31-Jul-18	Charcoal Fabric	love my echo	1
1	5	31-Jul-18	Charcoal Fabric	loved it	1
2	4	31-Jul-18	Walnut Finish	sometimes while playing a game you can answer ...	1
3	5	31-Jul-18	Charcoal Fabric	i have had a lot of fun with this thing my 4 y...	1
4	5	31-Jul-18	Charcoal Fabric	music	1

In [29]:

```
def tokenize_text(text):
    return text.split()

df['verified_reviews'] = df['verified_reviews'].apply(tokenize_text)
```

In [30]:

```
df.head()
```

Out[30]:

	rating	date	variation	verified_reviews	feedback
0	5	31-Jul-18	Charcoal Fabric	[love, my, echo]	1
1	5	31-Jul-18	Charcoal Fabric	[loved, it]	1
2	4	31-Jul-18	Walnut Finish	[sometimes, while, playing, a, game, you, can,...	1
3	5	31-Jul-18	Charcoal Fabric	[i, have, had, a, lot, of, fun, with, this, th...	1
4	5	31-Jul-18	Charcoal Fabric	[music]	1

In [31]:

```
def remove_stopwords(tokens):
    stop_words = set(stopwords.words('english'))
    return [word for word in tokens if word not in stop_words]

df['verified_reviews'] = df['verified_reviews'].apply(remove_stopwords)
```

In [32]:

```
df.head()
```

Out[32]:

	rating	date	variation	verified_reviews	feedback
0	5	31-Jul-18	Charcoal Fabric	[love, echo]	1
1	5	31-Jul-18	Charcoal Fabric	[loved]	1
2	4	31-Jul-18	Walnut Finish	[sometimes, playing, game, answer, question, c...	1
3	5	31-Jul-18	Charcoal Fabric	[lot, fun, thing, 4, yr, old, learns, dinosaur...	1
4	5	31-Jul-18	Charcoal Fabric	[music]	1

In [33]:
```
from nltk.tokenize import word_tokenize

# Convert the list of tokens to a string
df["verified_reviews"] = df["verified_reviews"].apply(lambda x: " ".join(x))

# Tokenize the "verified_reviews" column
df["verified_reviews"] = df["verified_reviews"].apply(word_tokenize)
df.head()
```

Out[33]:

	rating	date	variation	verified_reviews	feedback
0	5	31-Jul-18	Charcoal Fabric	[love, echo]	1
1	5	31-Jul-18	Charcoal Fabric	[loved]	1
2	4	31-Jul-18	Walnut Finish	[sometimes, playing, game, answer, question, c...	1
3	5	31-Jul-18	Charcoal Fabric	[lot, fun, thing, 4, yr, old, learns, dinosaur...	1
4	5	31-Jul-18	Charcoal Fabric	[music]	1

In [34]:
```
nltk.download('wordnet')
```

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

Out[34]:
```
True
```

In [35]:
```
stemmer = PorterStemmer()
lemmatizer = WordNetLemmatizer()
def stem_and_lemmatize(tokens):
    lemmatized = [lemmatizer.lemmatize(token) for token in tokens]
    stemmed = [stemmer.stem(token) for token in lemmatized]
    return stemmed
df['verified_reviews'] = df['verified_reviews'].apply(stem_and_lemmatize)
```

In [38]:
```
df.head()
```

Out[38]:

	rating	date	variation	verified_reviews	feedback
0	5	31-Jul-18	Charcoal Fabric	['love', 'echo']	1
1	5	31-Jul-18	Charcoal Fabric	['love']	1
2	4	31-Jul-18	Walnut Finish	['sometim', 'play', 'game', 'answer', 'questio...	1
3	5	31-Jul-18	Charcoal Fabric	['lot', 'fun', 'thing', '4', 'yr', 'old', 'lea...	1
4	5	31-Jul-18	Charcoal Fabric	['music']	1

In [41]:
```
df['verified_reviews'] = df['verified_reviews'].astype(str)
```

In [44]:
```
corpus=df['verified_reviews']
vector=CountVectorizer()
```

```
tt=vector.fit_transform(corpus)
print(tt.toarray())
```

```
[[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 0]]
```

In [52]:

```
from sklearn.feature_extraction.text import TfidfVectorizer

# Create an instance of the TfidfVectorizer
tfidf = TfidfVectorizer()

# Fit and transform the "verified_reviews" column
tfidf_matrix = tfidf.fit_transform(df["verified_reviews"])

# Get the feature names
feature_names = tfidf.get_feature_names_out()

# Print the feature names
print(feature_names[200:300])
```

```
['againthi' 'age' 'agent' 'ago' 'agoyesterday' 'agre' 'agreement' 'ahead'
 'ai' 'aid' 'aint' 'air' 'aka' 'al' 'alabama' 'alarm' 'alarmb' 'alarmcom'
 'alarmthi' 'albeit' 'alcohol' 'alert' 'alex' 'alexa' 'alexa34'
 'alexaalso' 'alexaechocomput' 'alexathi' 'alexi' 'alexia' 'alexu' 'algo'
 'aliv' 'allevi' 'alli' 'allinon' 'alloveral' 'allow' 'allrecip' 'almost'
 'alon' 'along' 'alongsid' 'alot' 'aloud' 'alread' 'alreadi' 'alright'
 'also' 'alter' 'altern' 'although' 'alway' 'amaonmaz' 'amax' 'amaz'
 'amazin' 'amazingli' 'amazon' 'amazonalexa' 'amazonia' 'amazonmark'
 'amazons' 'amazont' 'amazonzigbe' 'ambient' 'american' 'among' 'amount'
 'amozon' 'amplifi' 'amus' 'analog' 'and' 'android' 'angl' 'annoy' 'anoth'
 'answer' 'ant' 'anticip' 'antitechnolog' 'anybodi' 'anyhow' 'anylist'
 'anymor' 'anyon' 'anypod' 'anyth' 'anythingi' 'anytim' 'anyway' 'anywher'
 'apart' 'app' 'app34' 'appar' 'apparentlylong' 'appeal' 'appear']
```