In [1]:
```python
# Create a sentiment analysis model using Amazon reviews of a wireless xbox controller
# to predict positive, negative, or neutral sentiments.

# Create another model that can predict the rating of Amazon reviews of a wireless xbox
# controller (1-5 stars).
```

In [2]:
```python
# Import required libraries.

import requests
import pandas as pd
from bs4 import BeautifulSoup
from matplotlib import pyplot as plt
from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.naive_bayes import MultinomialNB
from sklearn.svm import LinearSVC
```

In [3]:
```python
# Scrape wireless xbox controller reviews from Amazon.com.

product_link = "https://www.amazon.com/Xbox-Wireless-Controller-Pulse-Red-Windows-Devices/product-reviews" \
               "/B0859XT328/ref=cm_cr_getr_d_paging_btm_prev_1?ie=UTF8&reviewerType=all_reviews&pageNumber=1"

HEADERS = ({'User-Agent':
            'Mozilla/5.0 (Windows NT 10.0; Win64; x64) \
            AppleWebKit/537.36 (KHTML, like Gecko) \
            Chrome/90.0.4430.212 Safari/537.36',
            'Accept-Language': 'en-US, en;q=0.5'})

# Gather the first 100 pages of reviews.
html_data = ""
for i in range(200):
    current_page = str("=" + str(int((i + 1))))
    next_page = str("=" + str(int((i + 2))))
    html_data += requests.get(product_link, headers=HEADERS).text
    product_link = product_link.replace(current_page, next_page)

soup = BeautifulSoup(html_data, "html.parser")
```

In [4]:
```python
# Gather the titles, ratings (1-5 stars), and descriptions of each review. Create a new
# category called "sentiment" that classifies the reviews as positive, negative, or neutral
# depending on the amount of stars it has (1-2 = negative, 3 = neutral, 4-5 = positive).

html_data = soup.find_all("div", {"data-hook": "review"})

# Store each review in a list.
reviews = []
for review in html_data:
    title = ""
    rating = ""
    description = ""
    sentiment = ""
    try:
        title = review.find("a", {"data-hook": "review-title"}).text.strip()
        rating = float(review.find("i", {"data-hook": "review-star-rating"}).text.replace("out of 5 stars", "")
                       .strip())
        description = review.find("span", {"data-hook": "review-body"}).text.replace("Read more", "").strip()
        if rating < 3:
            sentiment = "negative"
        elif rating < 4:
            sentiment = "neutral"
        else:
            sentiment = "positive"
    except AttributeError:
        pass

    # Ignore reviews without descriptions.
    if title != "" and rating != "" and description != "" and "The media could not be loaded." not in description \
            and "a " not in description:
        review_dict = {
            "title": title,
            "rating": rating,
            "description": description,
            "sentiment": sentiment
        }
        reviews.append(review_dict)
```

In [5]:
```python
# Create a dataframe.
df = pd.DataFrame(reviews)
```

In [6]:
```python
# Print 5 rows of the dataframe.
print(df.head(5))
```

```
                                            title  rating  \
0                          Great controller for PC     5.0
1      Finally playing pc games with a controller (:     5.0
2                    A more refined Xbox controller     5.0
3                          not what I was expecting     1.0
4  PC Users: You have to buy the wireless adapter...     4.0

                                        description sentiment
0  The controller works great with PC. However, o...  positive
1  For PC players, you DO NOT need to buy the wir...  positive
2  This review is for the new "Shock Blue" varian...  positive
3  for the price I did not expect to get a wirele...  negative
4  Pictured: The Robot White controller (it looks...  positive
```

In [7]:
```python
# Create 4 models.

# The first two will use a support vector classifier (SVC). One will use ratings and the other sentiments.
# The second will use naive bayes (NB). One will use ratings and the other sentiments.
```

In [8]:
```python
# (SVC) Review ratings model - create a model that predicts the RATING of reviews (1-5 stars).
```

In [9]:
```python
# (SVC) Review ratings model - vectorize the reviews.

tfidf = TfidfVectorizer(max_features=20000, ngram_range=(1, 5), analyzer="char")
X = tfidf.fit_transform(df["description"])
y = df["rating"]
```

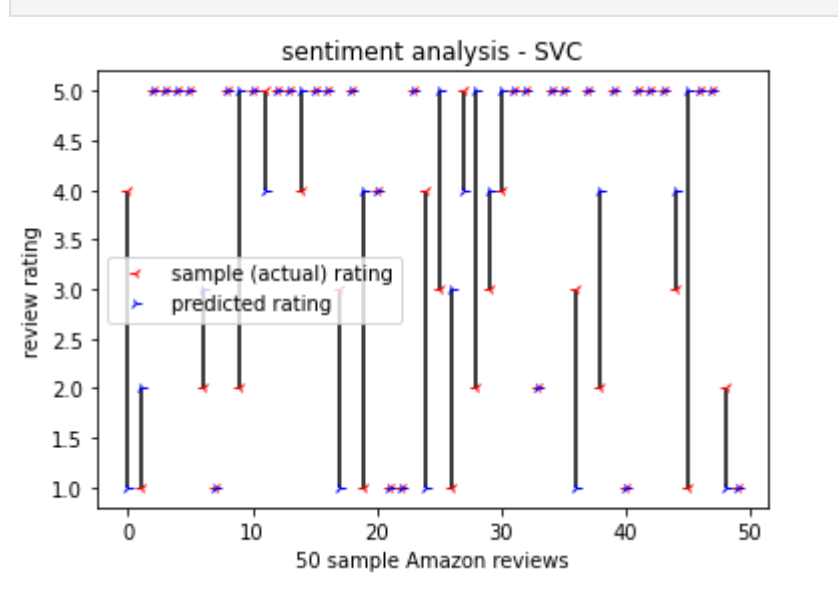
In [10]:
```python
# (SVC) Review ratings model - train the data.

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
```

In [11]:
```python
# (SVC) Review ratings model - fit the data.

clf = LinearSVC(C=20, class_weight="balanced", max_iter=20000)
clf.fit(X_train, y_train)

y_pred = clf.predict(X_test)
```

In [12]:
```python
# (SVC) Review ratings model - display a classification report. Show the accuracy of the model.
# being able to predict a rating (1-5 stars).

print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

         1.0       0.58      0.74      0.65       101
         2.0       0.29      0.12      0.17        41
         3.0       0.43      0.17      0.24        36
         4.0       0.31      0.23      0.27        43
         5.0       0.77      0.89      0.83       177

    accuracy                           0.64       398
   macro avg       0.48      0.43      0.43       398
weighted avg       0.59      0.64      0.60       398
```

In [13]:
```python
# (SVC) Review ratings model - plot some sample data. Show 50 samples.

# The red star is a sample category. It represents the ACTUAL category. The blue star
# is the predicted category. When a red and blue star overlap it means the prediction
# is correct. If they don't overlap it means the prediction is wrong and a line is drawn
# connecting the stars to visually indicate an incorrect prediction. There are thousands
# of questions, but only 50 are plotted because it is more visually pleasing and easy
# to read.

y_test = y_test.tolist()
y_pred = y_pred.tolist()

fig1 = plt.figure()
ax1 = plt.axes()

ax1.plot(y_test[0:50], "3", color="red", label="sample (actual) rating")
ax1.plot(y_pred[0:50], "4", color="blue", label="predicted rating")

for i in range(50):  # len(y_test)
    ax1.vlines(x=i, ymin=y_test[i], ymax=y_pred[i], color="black")

ax1.legend(numpoints=1)
ax1.set_title("sentiment analysis - SVC")
ax1.set_xlabel("50 sample Amazon reviews")
ax1.set_ylabel("review rating")

plt.show()
```



In [14]:
```python
# (SVC) Review sentiment model - create a model that predicts the SENTIMENT of reviews (pos, neg, neut).
```

In [15]:
```python
# (SVC) Review sentiment model - vectorize the reviews.

tfidf = TfidfVectorizer(max_features=20000, ngram_range=(1, 5), analyzer="char")
X = tfidf.fit_transform(df["description"])
y = df["sentiment"]
```

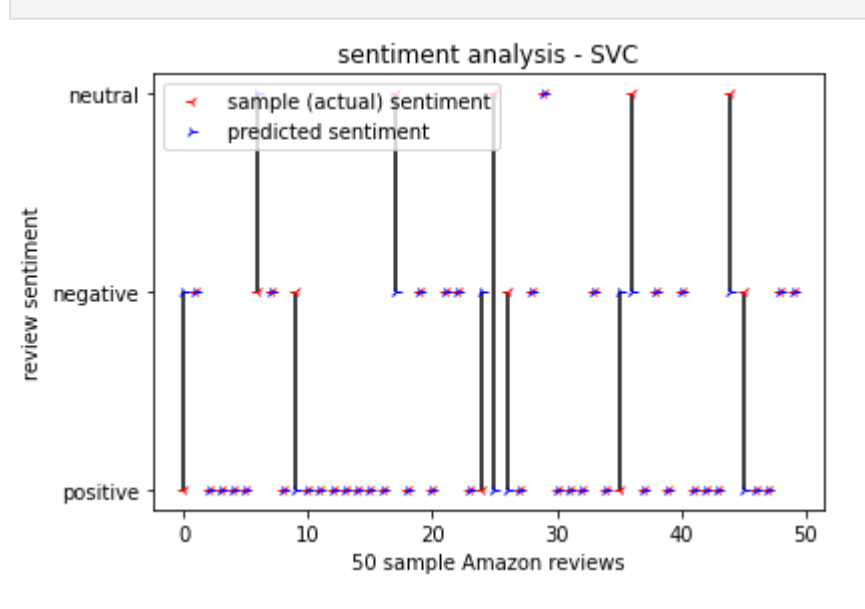
In [16]:
```python
# (SVC) Review sentiment model - train the data.

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
```

In [17]:
```python
# (SVC) Review sentiment model - fit the data.

clf = LinearSVC(C=20, class_weight="balanced", max_iter=20000)
clf.fit(X_train, y_train)

y_pred = clf.predict(X_test)
```

In [18]:
```python
# (SVC) Review sentiment model - display a classification report. Show the accuracy of the model.
# being able to predict a sentiment (pos, neg, neut).

print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

    negative       0.75      0.80      0.77       142
     neutral       0.39      0.19      0.26        36
    positive       0.85      0.89      0.87       220

    accuracy                           0.79       398
   macro avg       0.66      0.63      0.63       398
weighted avg       0.78      0.79      0.78       398
```

In [19]:
```python
# (SVC) Review sentiment model - plot some sample data. Show 50 samples.

# The red star is a sample category. It represents the ACTUAL category. The blue star
# is the predicted category. When a red and blue star overlap it means the prediction
# is correct. If they don't overlap it means the prediction is wrong and a line is drawn
# connecting the stars to visually indicate an incorrect prediction. There are thousands
# of questions, but only 50 are plotted because it is more visually pleasing and easy
# to read.

y_test = y_test.tolist()
y_pred = y_pred.tolist()

fig2 = plt.figure()
ax2 = plt.axes()

ax2.plot(y_test[0:50], "3", color="red", label="sample (actual) sentiment")
ax2.plot(y_pred[0:50], "4", color="blue", label="predicted sentiment")

for i in range(50):
    ax2.vlines(x=i, ymin=y_test[i], ymax=y_pred[i], color="black")

ax2.legend(numpoints=1)
ax2.set_title("sentiment analysis - SVC")
ax2.set_xlabel("50 sample Amazon reviews")
ax2.set_ylabel("review sentiment")

plt.show()
```



In [20]:
```python
# (NB) Review ratings model - create a model that predicts the RATING of reviews (1-5 stars).
```

In [21]:
```python
# (NB) Review ratings model - use review description and rating.

x = df["description"]
y = df["rating"]
```

In [22]:
```python
# (NB) Review ratings model - train the data.

x, x_test, y, y_test = train_test_split(x, y, stratify=y, test_size=0.25, random_state=42)
```

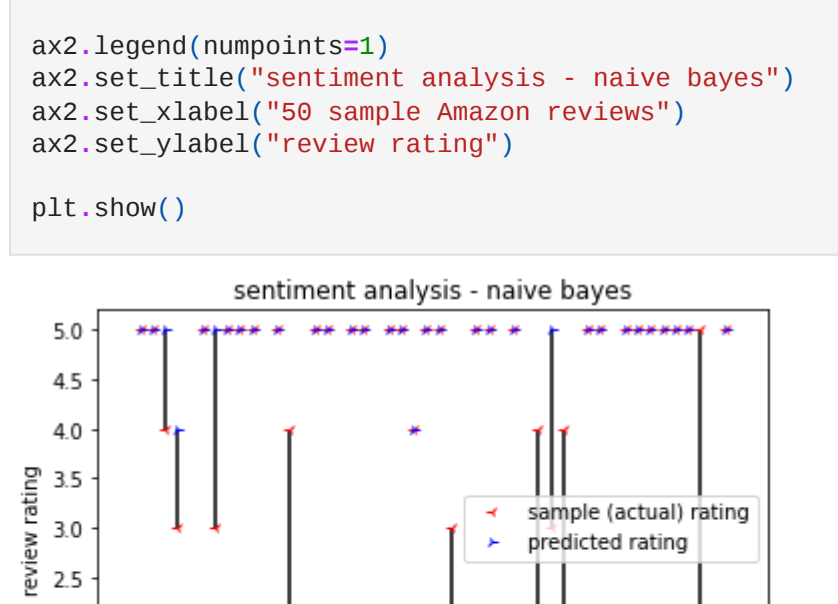
In [23]:
```python
# (NB) Review ratings model - transform the data.

vec = CountVectorizer(stop_words='english')
x = vec.fit_transform(x).toarray()
x_test = vec.transform(x_test).toarray()
```

In [24]:
```python
# (NB) Review ratings model - fit the data.

model = MultinomialNB()
model.fit(x, y)

y_pred = model.predict(x_test)
```

In [25]:
```python
# (NB) Review ratings model - display a classification report. Show the accuracy of the model.
# being able to predict a rating (1-5 stars).

print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

         1.0       0.56      0.87      0.68       127
         2.0       0.00      0.00      0.00        42
         3.0       0.00      0.00      0.00        42
         4.0       0.33      0.02      0.04        52
         5.0       0.75      0.93      0.83       235

    accuracy                           0.66       498
   macro avg       0.33      0.36      0.31       498
weighted avg       0.53      0.66      0.57       498
```

In [26]:
```python
# (NB) Review ratings model - plot some sample data. Show 50 samples.

# The red star is a sample category. It represents the ACTUAL category. The blue star
# is the predicted category. When a red and blue star overlap it means the prediction
# is correct. If they don't overlap it means the prediction is wrong and a line is drawn
# connecting the stars to visually indicate an incorrect prediction. There are thousands
# of questions, but only 50 are plotted because it is more visually pleasing and easy
# to read.

y_test = y_test.tolist()
y_pred = y_pred.tolist()

fig2 = plt.figure()
ax2 = plt.axes()

ax2.plot(y_test[0:50], "3", color="red", label="sample (actual) rating")
ax2.plot(y_pred[0:50], "4", color="blue", label="predicted rating")

for i in range(50):  # len(y_test)
    ax2.vlines(x=i, ymin=y_test[i], ymax=y_pred[i], color="black")

ax2.legend(numpoints=1)
ax2.set_title("sentiment analysis - naive bayes")
ax2.set_xlabel("50 sample Amazon reviews")
ax2.set_ylabel("review rating")

plt.show()
```



In [27]:
```python
# (NB) Review sentiment model - create a model that predicts the SENTIMENT of reviews (pos, neg, neut).
```

In [28]:
```python
# (NB) Review sentiment model - use review description and rating.

x = df["description"]
y = df["sentiment"]
```

In [29]:
```python
# (NB) Review sentiment model - train the data.

x, x_test, y, y_test = train_test_split(x, y, stratify=y, test_size=0.25, random_state=42)
```

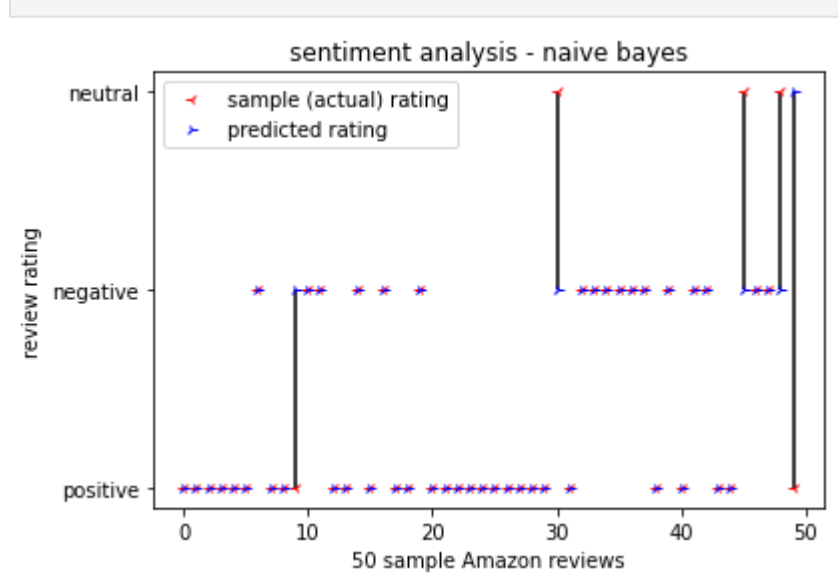
In [30]:
```python
# (NB) Review sentiment model - transform the data.

vec = CountVectorizer(stop_words='english')
x = vec.fit_transform(x).toarray()
x_test = vec.transform(x_test).toarray()
```

In [31]:
```python
# (NB) Review sentiment model - fit the data.

model = MultinomialNB()
model.fit(x, y)

y_pred = model.predict(x_test)
```

In [32]:
```python
# (NB) Review sentiment model - display a classification report. Show the accuracy of the model.
# being able to predict a sentiment (pos, neg, neut).

print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

    negative       0.76      0.85      0.80       169
     neutral       0.00      0.00      0.00        43
    positive       0.86      0.92      0.89       286

    accuracy                           0.81       498
   macro avg       0.54      0.59      0.56       498
weighted avg       0.75      0.81      0.78       498
```

In [33]:
```python
# (NB) Review sentiment model - plot some sample data. Show 50 samples.

# The red star is a sample category. It represents the ACTUAL category. The blue star
# is the predicted category. When a red and blue star overlap it means the prediction
# is correct. If they don't overlap it means the prediction is wrong and a line is drawn
# connecting the stars to visually indicate an incorrect prediction. There are thousands
# of questions, but only 50 are plotted because it is more visually pleasing and easy
# to read.

y_test = y_test.tolist()
y_pred = y_pred.tolist()

fig2 = plt.figure()
ax2 = plt.axes()

ax2.plot(y_test[0:50], "3", color="red", label="sample (actual) rating")
ax2.plot(y_pred[0:50], "4", color="blue", label="predicted rating")

for i in range(50):  # len(y_test)
    ax2.vlines(x=i, ymin=y_test[i], ymax=y_pred[i], color="black")

ax2.legend(numpoints=1)
ax2.set_title("sentiment analysis - naive bayes")
ax2.set_xlabel("50 sample Amazon reviews")
ax2.set_ylabel("review rating")

plt.show()
```



In [34]:
```python
# ANALYSIS
#
#
# The classification model was able to more accurately predict ratings (1-5 stars)
# than the naive bayes model by 1%. It was able to much more accurately predict
# 3 star ratings than the naive bayes one without additional tinkering.
#
# The naive bayes model was able to more accurately predict sentiments (pos, neg,
# and neutr) than the classification model by 3%. It wasn't able to predict
# neutral sentiments well at all, but gave strong results for positive and negative
# sentiments. This could partly be due to the fact there aren't many 3 star reviews
# using language that well depicts something that isn't too high or too low - which
# makes it extremely difficult to make middle-of-the-road predictions.
#
# The takeaway is naive bayes is the best option to use in this example. While it
# doesn't outperform the model using a support vector classifier, with further
# optimization for 3-star reviews, it can potentially be much more accurate.
```