OBJECTIVE

To predict whether a women's clothing review is positive or negative based on the text of the review using the Multinomial Naive Bayes algorithm.

Step-by-step implementation:

1.Import libraries

```
import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

# Text processing libraries

from sklearn.feature_extraction.text import CountVectorizer

from sklearn.model_selection import train_test_split

from sklearn.naive_bayes import MultinomialNB

# Evaluation

from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```

2.Import Data

```
# Assuming the dataset is downloaded as 'Womens_Clothing_Reviews.csv'

df = pd.read_csv('Womens_Clothing_Reviews.csv')
```

3.Describe Data

```
# Show the first few rows of the dataset

print(df.head())

# Summary statistics

print(df.describe())

# Information about the dataset

print(df.info())

# Check for null values

print(df.isnull().sum())
```

4.Data Visualization

```
# Visualize the distribution of ratings
sns.countplot(df['Rating'])
plt.title('Distribution of Ratings')
plt.show()
# Checking the distribution of review lengths
df['Review Length'] = df['Review Text'].apply(lambda x: len(str(x)))
sns.histplot(df['Review Length'], kde=True)
plt.title('Distribution of Review Lengths')
plt.show()
```

5. Data Preprocessing:

- Remove rows with missing values in the Review Text.
- Label reviews as positive (rating 4 or 5) or negative (rating 1, 2, 3).
- Convert text data into a format suitable for modeling (using CountVectorizer).

```
# Drop missing values in 'Review Text'
df = df.dropna(subset=['Review Text'])
# Define target variable - binary classification based on rating
df['Sentiment'] = df['Rating'].apply(lambda x: 1 if x > 3 else 0)
# Feature (X) and target (y)
X = df['Review Text']
y = df['Sentiment']
# Convert text data to numerical using CountVectorizer
vectorizer = CountVectorizer(stop_words='english')
X = vectorizer.fit_transform(X)
```

6.Define Target Variable (y) and Feature Variables (X):

- X: Processed text reviews.

- y: The binary target variable representing sentiment (positive or negative).

We have already defined X and y in the previous section

7. Train-Test Split:

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

8. Modeling:

Instantiate the Multinomial Naive Bayes model

model = MultinomialNB()

Train the model

model.fit(X_train, y_train)

9. Model Evaluation:

Evaluate the performance of the model using accuracy, confusion matrix, and classification report.

Predict on the test set

y_pred = model.predict(X_test)

Accuracy score

print(f'Accuracy: {accuracy_score(y_test, y_pred)}')

Confusion matrix

print(confusion_matrix(y_test, y_pred))

Classification report

print(classification_report(y_test, y_pred))

10. Prediction and Explanation:

To predict the sentiment of new reviews:

New review example

new_reviews = ['This dress is amazing!', 'I did not like the fit at all.']

```
# Convert to numerical format using the same vectorizer
new_reviews_vectorized = vectorizer.transform(new_reviews)

# Make predictions
predictions = model.predict(new_reviews_vectorized)

# Display results
for review, sentiment in zip(new_reviews, predictions):
    print(f'Review: "{review}" -> Sentiment: {"Positive" if sentiment == 1 else "Negative"}')
```

CONCLUSION:

After evaluation,you can fine -tune the model by adjusting the parameters, using different preprocessing techniques or experimenting with different feature extraction methods.