Product Review Prediction with Multinomial Naive Bayes

The multinomial Naive Bayes is suitable for classification with discrete features(e.g., word counts for text classification). The multinomial distribution typically requires integer feature counts. However, in practice, fractional counts such as term frequency-inverse document frequency may also work.

Data Source : GitHub

Import Library

```
import pandas as pd
```

```
import numpy as np
```

```
import matplotlib.pyplot as plt
```

```
import seaborn as sns
```

Import Dataset

```
df=pd.read_csv('https://raw.githubusercontent.com/YBIFoundation/ProjectHub-MachineLearning/main/Women%20Clothing%20E-Commerce%20Review.csv')
```

Descibe Data

```
df.head()
```

	Clothing ID	Age	Title	Review	Rating	Recommended	Positive Feedback	Division	Departmen
0	767	33	NaN	Absolutely wonderful - silky and sexy and comf...	4	1	0	Initmates	Intimat
1	1080	34	NaN	Love this dress! it's sooo pretty. i happene...	5	1	4	General	Dresse

Next steps: Generate code with df View recommended plots

```
df.tail()
```

	Clothing ID	Age	Title	Review	Rating	Recommended	Positive Feedback	Division	Department	Category
23481	1104	34	Great dress for many occasions	I was very happy to snag this dress at such a ...	5	1	0	General Petite	Dresses	Dresses
23482	862	48	Wish it was made of cotton	It reminds me of maternity clothes. soft, stre...	3	1	0	General Petite	Tops	Knits
23483	1104	31	Cute, but see through	This fit well, but the top was very see throug...	3	0	1	General Petite	Dresses	Dresses
23484	1084	28	Very cute dress, perfect for summer parties an...	I bought this dress for a wedding i have this ...	3	1	2	General	Dresses	Dresses

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 23486 entries, 0 to 23485
Data columns (total 10 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Clothing ID        23486 non-null  int64
 1   Age                23486 non-null  int64
 2   Title              19676 non-null  object
 3   Review             22641 non-null  object
 4   Rating             23486 non-null  int64
 5   Recommended        23486 non-null  int64
 6   Positive Feedback  23486 non-null  int64
 7   Division           23472 non-null  object
 8   Department         23472 non-null  object
 9   Category           23472 non-null  object
dtypes: int64(5), object(5)
memory usage: 1.8+ MB
```

```
df.describe()
```

	Clothing ID	Age	Rating	Recommended	Positive Feedback
count	23486.000000	23486.000000	23486.000000	23486.000000	23486.000000
mean	918.118709	43.198544	4.196032	0.822362	2.535936
std	203.298980	12.279544	1.110031	0.382216	5.702202
min	0.000000	18.000000	1.000000	0.000000	0.000000
25%	861.000000	34.000000	4.000000	1.000000	0.000000
50%	936.000000	41.000000	5.000000	1.000000	1.000000
75%	1078.000000	52.000000	5.000000	1.000000	3.000000
max	1205.000000	99.000000	5.000000	1.000000	122.000000

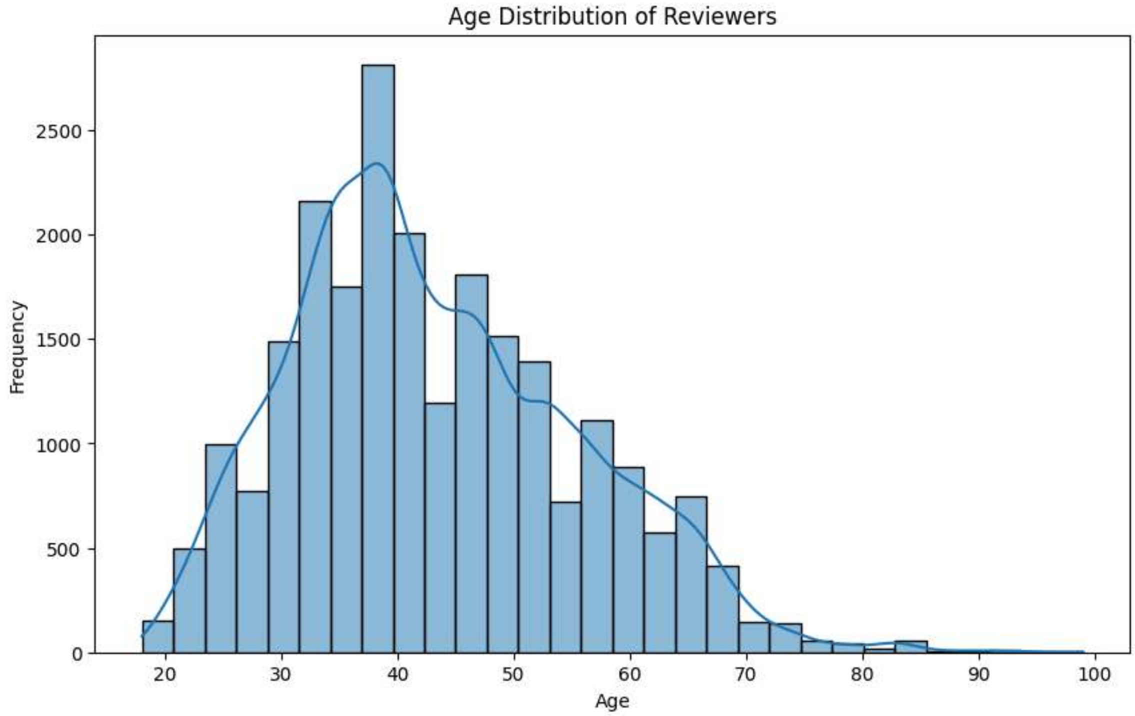
```
df.shape
```

```
(23486, 10)
```

Data Visualization

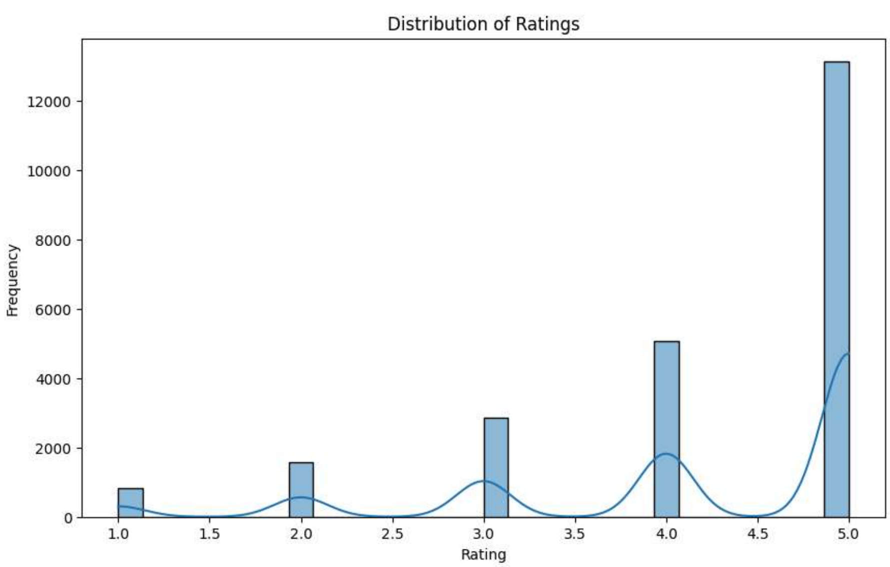
Histogram of Age

```
plt.figure(figsize=(10, 6))
sns.histplot(df['Age'], bins=30, kde=True)
plt.title('Age Distribution of Reviewers')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.show()
```



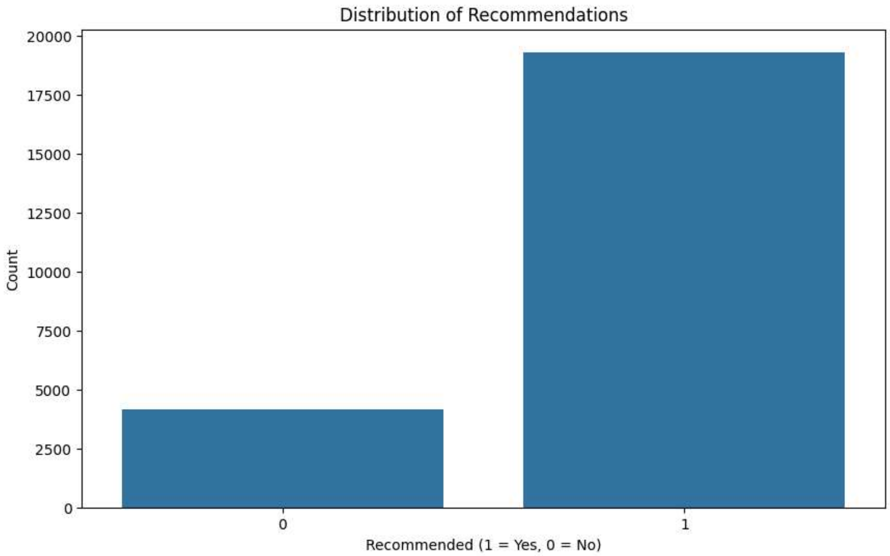
Bar Plot of Ratings

```
plt.figure(figsize=(10, 6))
sns.histplot(df['Rating'], bins=30, kde=True)
plt.title('Distribution of Ratings')
plt.xlabel('Rating')
plt.ylabel('Frequency')
plt.show()
```



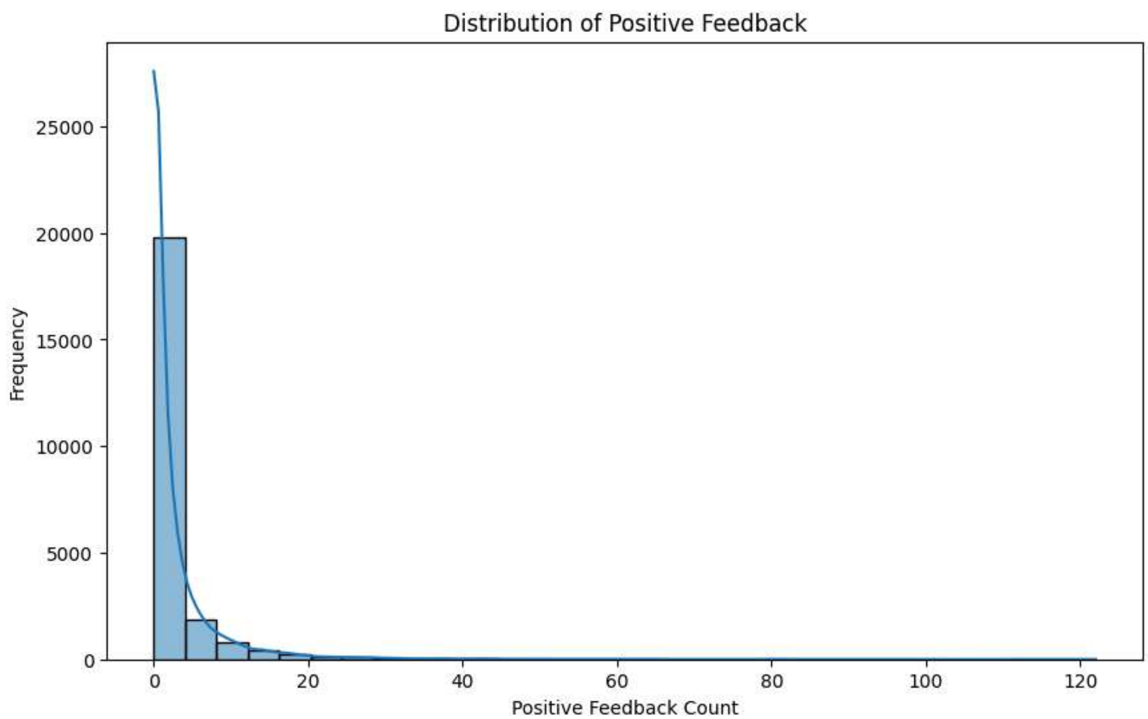
Bar Plot of Recommendations

```
plt.figure(figsize=(10, 6))
sns.countplot(x='Recommended', data=df)
plt.title('Distribution of Recommendations')
plt.xlabel('Recommended (1 = Yes, 0 = No)')
plt.ylabel('Count')
plt.show()
```



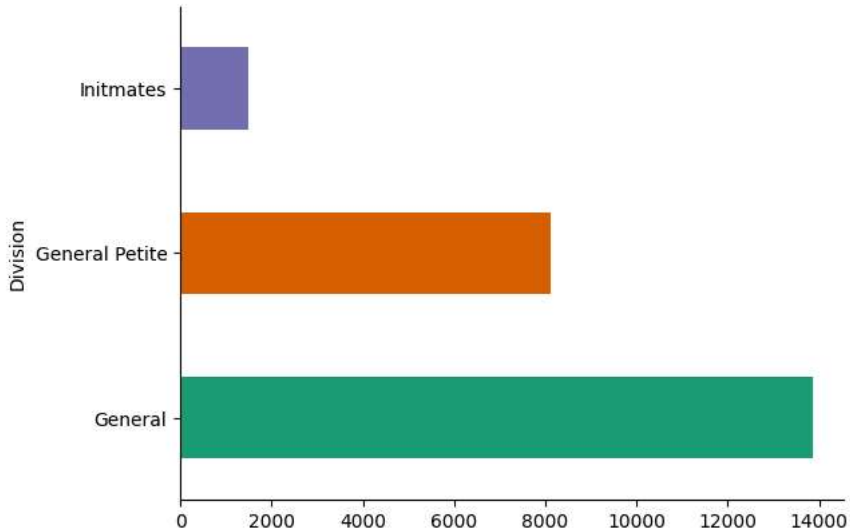
Histogram of Positive Feedback

```
plt.figure(figsize=(10, 6))
sns.histplot(df['Positive Feedback'], bins=30, kde=True)
plt.title('Distribution of Positive Feedback')
plt.xlabel('Positive Feedback Count')
plt.ylabel('Frequency')
plt.show()
```



Division

```
# @title Division
df.groupby('Division').size().plot(kind='barh', color=sns.palettes.mpl_palette('Dark2'))
plt.gca().spines[['top', 'right',]].set_visible(False)
```



Missing Values/Data Preprocessing

```
df.isna().sum()
```

```
Clothing ID            0
Age                    0
Title               3810
Review               845
Rating                 0
Recommended            0
Positive Feedback      0
Division              14
Department            14
Category              14
dtype: int64
```

```
df[df['Review']==""]=np.NaN
```

```
df['Review'].fillna("No Review",inplace=True)
```

```
df.isna().sum()
```

```
Clothing ID            0
Age                    0
Title               3810
Review                 0
Rating                 0
Recommended            0
Positive Feedback      0
Division              14
Department            14
Category              14
dtype: int64
```

```
df['Review']
```

```
0        Absolutely wonderful - silky and sexy and comf...
1        Love this dress!  it's sooo pretty.  i happene...
2        I had such high hopes for this dress and reall...
3        I love, love, love this jumpsuit. it's fun, fl...
4        This shirt is very flattering to all due to th...
                               ...
23481    I was very happy to snag this dress at such a ...
23482    It reminds me of maternity clothes. soft, stre...
```

```
23483    This fit well, but the top was very see throug...
23484    I bought this dress for a wedding i have this ...
23485    This dress in a lovely platinum is feminine an...
Name: Review, Length: 23486, dtype: object
```

Define Target Variable (y) and Feature Variables (X)

```
df.columns
```

```
Index(['Clothing ID', 'Age', 'Title', 'Review', 'Rating', 'Recommended',
       'Positive Feedback', 'Division', 'Department', 'Category'],
      dtype='object')
```

```
x=df['Review']
```

```
y=df['Rating']
```

```
df['Rating'].value_counts()
```

```
Rating
5    13131
4     5077
3     2871
2     1565
1      842
Name: count, dtype: int64
```

Train Test Split

```
from sklearn.model_selection import train_test_split
```

```
x_train,x_test,y_train,y_test=train_test_split(x,y,train_size=0.7,stratify=y,random_state=2529)
```

```
x_train.shape,x_test.shape,y_train.shape,y_test.shape
```

```
((16440,), (7046,), (16440,), (7046,))
```

Get Features Text Conversion to Tokens

```
from sklearn.feature_extraction.text import CountVectorizer
```

```
cv=CountVectorizer(lowercase=True,analyzer='word',ngram_range=(2,3), stop_words='english', max_features=5000)
```

```
x_train=cv.fit_transform(x_train)
```

```
cv.get_feature_names_out()
```

```
array(['10 12', '10 bought', '10 fit', ..., 'yellow color', 'yoga pants',
       'zipper little'], dtype=object)
```

```
x_train.toarray()
```

```
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, 0, ..., 0, 0, 0]])
```

```
x_test=cv.fit_transform(x_test)
```

```
cv.get_feature_names_out()
```

```
array(['10 12', '10 dress', '10 fit', ..., 'years come', 'years old',
       'yoga pants'], dtype=object)
```

```
x_test.toarray()
```

```
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, 0, ..., 0, 0, 0]])
```

Get Model Train

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

```
model=MultinomialNB()
```

```
model.fit(x_train,y_train)
```

```
▾ MultinomialNB
MultinomialNB()
```

Get Model Prediction

```
y_pred=model.predict(x_test)
```

```
y_pred.shape
```

```
(7046,)
```

```
y_pred
```

```
array([1, 5, 5, ..., 5, 5, 5])
```

Get Probability of Each Predicted Class

```
model.predict_proba(x_test)
```

```
array([[0.71118473, 0.02625165, 0.15465118, 0.01496876, 0.09294369],
       [0.02416867, 0.04769471, 0.35268622, 0.16185007, 0.41360034],
       [0.03582725, 0.06660584, 0.12226277, 0.21618005, 0.55912409],
       ...,
       [0.02320281, 0.08950939, 0.08962183, 0.16719203, 0.63047394],
       [0.01167675, 0.00202714, 0.08539004, 0.34347398, 0.55743209],
       [0.03959824, 0.05612822, 0.00688869, 0.1560574 , 0.74132745]])
```

Get Model Evaluation

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

```
print(confusion_matrix(y_test, y_pred))
```

```
[[  15   13   45   36  144]
 [  43   43   86   85  213]
 [ 116   78  113  166  388]
 [ 166  108  194  336  719]
 [ 371  272  349  722 2225]]
```

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

```
              precision    recall  f1-score   support

           1       0.02      0.06      0.03       253
           2       0.08      0.09      0.09       470
           3       0.14      0.13      0.14       861
           4       0.25      0.22      0.23      1523
           5       0.60      0.56      0.58      3939

    accuracy                           0.39      7046
   macro avg       0.22      0.21      0.21      7046
weighted avg       0.42      0.39      0.40      7046
```

Recategories Ratings as Poor(0) and Good(1)

```
df['Rating'].value_counts()
```

```
Rating
5    13131
4     5077
3     2871
2     1565
1      842
Name: count, dtype: int64
```

Re-Rating as 1,2,3 as 0 and 4,5 as 1

```
df.replace({'Rating' : {1:0,2:0,3:0,4:1,5:1}},inplace=True)
```

```
y=df['Rating']
```

```
x=df['Review']
```

Train Test Split

```
from sklearn.model_selection import train_test_split
```

```
x_train,x_test,y_train,y_test=train_test_split(x,y,train_size=0.7,stratify=y,random_state=2529)
```

```
x_train.shape,x_test.shape,y_train.shape,y_test.shape
```

```
((16440,), (7046,), (16440,), (7046,))
```

Get Feature Text Conversion to Tokens

```
from sklearn.feature_extraction.text import CountVectorizer
```

```
cv=CountVectorizer(lowercase=True,analyzer='word',ngram_range=(2,3),stop_words='english',max_features=5000)
```

```
x_train=cv.fit_transform(x_train)
```

```
x_test=cv.fit_transform(x_test)
```

Get Model Re-Train

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

```
model=MultinomialNB()
```

```
model.fit(x_train,y_train)
```

```
▾ MultinomialNB
MultinomialNB()
```

Get Model Prediction

```
y_pred=model.predict(x_test)
```

```
y_pred.shape
```

```
(7046,)
```

```
y_pred
```

```
array([1, 1, 1, ..., 1, 1, 1])
```

Get Model Evaluation

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

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

```
              precision    recall  f1-score   support

           0       0.31      0.28      0.30      1583
           1       0.80      0.82      0.81      5463

    accuracy                           0.70      7046
   macro avg       0.56      0.55      0.55      7046
weighted avg       0.69      0.70      0.69      7046
```

```
print(accuracy_score(y_test,y_pred))
```

```
0.698694294635254
```

```
confusion_matrix(y_test,y_pred)
```

```
array([[ 449, 1134],
       [ 989, 4474]])
```

Explanation

The project predicts product reviews' sentiments using Multinomial Naive Bayes (MNB), starting with data preprocessing and feature extraction. Initially, the model predicts detailed ratings (1 to 5), but due to low accuracy, the task is simplified by categorizing ratings into binary classes: poor (0) and good (1). This retraining focuses on distinguishing between positive and negative reviews, significantly improving the model's performance. The final model achieves around 70% accuracy, effectively classifying reviews as positive or negative. The binary classification task reduces complexity, leading to better performance metrics. Retraining with this simplified target improves accuracy and provides more reliable predictions for review sentiment.