In [2]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

# Step 1: Load and preprocess the dataset
# Replace 'consumer_complaints.csv' with the actual file path to your dataset
data = pd.read_csv('C:/Users/491583/Downloads/complaints/complaints.csv')
```

```
In [29]: data
```

Out[29]:

	Date received	Product	Sub-product	Issue	Sub-issue	Consumer complaint narrative	Company public response	Company	State	ZIP code	Tags	Cons co prov
0	2023-08-24	Credit reporting, credit repair services, or o...	Credit reporting	Problem with a credit reporting company's inve...	Was not notified of investigation status or re...	nan	NaN	Experian Information Solutions Inc.	NJ	07024	NaN	
1	2023-08-25	Credit reporting or other personal consumer re...	Credit reporting	Improper use of your report	Reporting company used your report improperly	nan	NaN	SANTANDER HOLDINGS USA, INC.	FL	33972	NaN	
2	2023-07-13	Checking or savings account	Checking account	Problem caused by your funds being low	Overdrafts and overdraft fees	Citibank allowed debit card transactions to ov...	Company has responded to the consumer and the ...	CITIBANK, N.A.	TX	XXXXX	NaN	Cc prc
3	2023-09-04	Money transfer, virtual currency, or money ser...	Mobile or digital wallet	Trouble accessing funds in your mobile or digi...	NaN	nan	NaN	Paypal Holdings, Inc	NC	27587	NaN	
4	2023-09-13	Credit reporting or other personal consumer re...	Credit reporting	Incorrect information on your report	Old information reappears or never goes away	nan	NaN	EQUIFAX, INC.	FL	33805	NaN	
...	...	...	...	...	...	...	...	...	...	...	...	
4069696	2015-02-21	Credit card	NaN	Sale of account	NaN	nan	NaN	JPMORGAN CHASE & CO.	AL	36695	NaN	
4069697	2015-07-19	Credit reporting	NaN	Incorrect information on credit report	Account status	nan	NaN	EQUIFAX, INC.	IL	60614	NaN	Cc prc

	Date received	Product	Sub-product	Issue	Sub-issue	Consumer complaint narrative	Company public response	Company	State	ZIP code	Tags	Cons co prov
4069698	2022-06-16	Mortgage	Conventional home mortgage	Applying for a mortgage or refinancing an exis...	NaN	nan	NaN	Mr. Cooper Group Inc.	IA	52205	NaN	
4069699	2022-04-26	Debt collection	I do not know	Attempts to collect debt not owed	Debt is not yours	nan	NaN	V and H Portfolio	SC	29624	NaN	Cc prc
4069700	2022-08-15	Checking or savings account	Checking account	Managing an account	Deposits and withdrawals	nan	NaN	JPMORGAN CHASE & CO.	PA	17972	Older American	

4069701 rows × 18 columns

In [20]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4069701 entries, 0 to 4069700
Data columns (total 18 columns):
 #   Column                        Dtype 
---  ------                        ----- 
 0   Date received                 object
 1   Product                       object
 2   Sub-product                   object
 3   Issue                         object
 4   Sub-issue                     object
 5   Consumer complaint narrative  object
 6   Company public response       object
 7   Company                       object
 8   State                         object
 9   ZIP code                      object
 10  Tags                          object
 11  Consumer consent provided?    object
 12  Submitted via                 object
 13  Date sent to company          object
 14  Company response to consumer  object
 15  Timely response?              object
 16  Consumer disputed?            object
 17  Complaint ID                  int64 
dtypes: int64(1), object(17)
memory usage: 558.9+ MB
```

In [21]:
```
data.describe()
```

Out[21]:

	Complaint ID
count	4.069701e+06
mean	4.325604e+06
std	2.029052e+06
min	1.000000e+00
25%	2.873374e+06
50%	4.262868e+06
75%	6.150424e+06
max	7.552997e+06

In [ ]:

In [3]:
```
# Data Cleaning and Feature Engineering can be done here

# Step 2: Text Pre-Processing
# Assuming the text data is in a column named 'complaint_text'
data['Consumer complaint narrative'] = data['Consumer complaint narrative'].astype(str)  # Convert to strings
data['Consumer complaint narrative'].fillna('', inplace=True)  # Replace NaN with empty strings

# Text Cleaning and Tokenization can be performed here
```

In [4]:
```
# Step 3: Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(data['Consumer complaint narrative'], data['Product'], te
```

In [5]:
```
# Step 4: Text Vectorization
tfidf_vectorizer = TfidfVectorizer(max_features=50)  # You can adjust max_features as needed
X_train_tfidf = tfidf_vectorizer.fit_transform(X_train)
X_test_tfidf = tfidf_vectorizer.transform(X_test)
```

In [6]:
```
# Step 5: Model Selection
# Choose a classification model, e.g., Multinomial Naive Bayes
model = MultinomialNB()
```

In [ ]:
```
from sklearn.metrics import precision_score, f1_score

# Set zero_division to "warn" (default behavior)
precision = precision_score(y_true, y_pred, zero_division="warn")
f1 = f1_score(y_true, y_pred, zero_division="warn")

# Set zero_division to "raise" (raises an exception if there are zero predicted samples)
precision = precision_score(y_true, y_pred, zero_division="raise")
f1 = f1_score(y_true, y_pred, zero_division="raise")

# Set zero_division to a numeric value (e.g., 1.0)
precision = precision_score(y_true, y_pred, zero_division=1.0)
f1 = f1_score(y_true, y_pred, zero_division=1.0)
```

In [17]:
```python
# Step 6: Model Training and Evaluation
model.fit(X_train_tfidf, y_train)
y_pred = model.predict(X_test_tfidf)



from sklearn.metrics import precision_score, recall_score, f1_score

# Assuming y_true and y_pred are your true labels and predicted labels for a multiclass problem
"""
# Calculate precision, recall, and F1-score using 'macro' averaging
precision_macro = precision_score(y_test, y_pred, average='macro', zero_division="warn")
recall_macro = recall_score(y_test, y_pred, average='macro')
f1_macro = f1_score(y_test, y_pred, average='macro')
"""
# Calculate precision, recall, and F1-score using 'micro' averaging
precision_micro = precision_score(y_test, y_pred, average='micro', zero_division="warn")
recall_micro = recall_score(y_test, y_pred, average='micro')
f1_micro = f1_score(y_test, y_pred, average='micro')



# Model Evaluation
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred)

print(f"Accuracy: {accuracy}")
print("Confusion Matrix:")
print(conf_matrix)
print("Classification Report:")
print(class_report)
print(f"Precision: {precision_micro}")
print("F1 Score")
print(f1_micro)
```

```
0.00      9235
Credit reporting, credit repair services, or other personal consumer reports       0.54      1.00
0.70    433179
                                                            Debt collection       0.78      0.07
0.13    101623
                                                  Debt or credit management       0.00      0.00
0.00        11
                          Money transfer, virtual currency, or money service       0.00      0.00
0.00     11660
                                                            Money transfers       0.00      0.00
0.00      1068
                                                                   Mortgage       0.00      0.00
0.00     76923
                                                    Other financial service       0.00      0.00
0.00       194
                                                                Payday loan       0.00      0.00
0.00      1114
                                         Payday loan, title loan, or personal loan       0.00      0.00
0.00      6091
                              Payday loan, title loan, personal loan, or advance loan       0.00      0.00
```

In [18]:
```python
# Step 7: Prediction
# You can use the trained model to make predictions on new data
new_complaints = ["This company keeps calling me for debts I don't owe.",
                  "My mortgage interest rate is too high."]
new_complaints_tfidf = tfidf_vectorizer.transform(new_complaints)
predicted_categories = model.predict(new_complaints_tfidf)

print("Prediction for New Complaints:")
for complaint, category in zip(new_complaints, predicted_categories):
    print(f"Complaint: {complaint}\nPrediction: {category}")
```

```
Prediction for New Complaints:
Complaint: This company keeps calling me for debts I don't owe.
Prediction: Credit reporting, credit repair services, or other personal consumer reports
Complaint: My mortgage interest rate is too high.
Prediction: Credit reporting, credit repair services, or other personal consumer reports
```

In [31]:
```python
# Step 7: Prediction
# You can use the trained model to make predictions on new data
new_complaints = ["Citibank allowed debit card transactions."]
new_complaints_tfidf = tfidf_vectorizer.transform(new_complaints)
predicted_categories = model.predict(new_complaints_tfidf)

print("Prediction for New Complaints:")
for complaint, category in zip(new_complaints, predicted_categories):
    print(f"Complaint: {complaint}\nPrediction: {category}")
```

```
Prediction for New Complaints:
Complaint: Citibank allowed debit card transactions.
Prediction: Credit reporting, credit repair services, or other personal consumer reports
```

In [ ]: