Sentiment Analysis for Product Reviews

```
import pandas as pd
import nltk
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, classification_report
```

1.Load text data and preprocess

In this step, we load the raw customer reviews from a dataset like Amazon product reviews. The text is cleaned to remove unnecessary elements such as punctuation, extra spaces, stop-words (like the, is, and), and special characters. We then tokenize the text, meaning we split sentences into individual words. Lemmatization is applied to convert words to their base form (e.g., running → run). This step ensures the text becomes clean and structured before applying machine learning models.

```
#About the datasetThis is a list of over 34,000 consumer reviews for Amazon products like the Kindle, Fire TV Stick, and more pr
df = pd.read_csv("/content/Datafiniti_Amazon_Consumer_Reviews_of_Amazon_Products_May19.csv")
```

```
nltk.download('stopwords')
from nltk.corpus import stopwords

stopwords = stopwords.words('english')

print("Available columns in the DataFrame:", df.columns)

# Based on common dataset structures for Amazon reviews, the column might be 'reviews.text'.
df['cleaned'] = df['reviews.text'].str.lower()
```

```
Available columns in the DataFrame: Index(['id', 'dateAdded', 'dateUpdated', 'name', 'asins', 'brand',
       'categories', 'primaryCategories', 'imageURLs', 'keys', 'manufacturer',
       'manufacturerNumber', 'reviews.date', 'reviews.dateSeen',
       'reviews.didPurchase', 'reviews.doRecommend', 'reviews.id',
       'reviews.numHelpful', 'reviews.rating', 'reviews.sourceURLs',
       'reviews.text', 'reviews.title', 'reviews.username', 'sourceURLs'],
      dtype='object')
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

2.Convert text to numerical features (TF-IDF / Embeddings)

Machine learning models cannot understand raw text, so we convert words into numeric form. TF-IDF represents words based on how important they are in a document, while word embeddings like Word2Vec convert words into vectors that capture meaning and relationships. These feature vectors help the model understand text patterns, similarities, and sentiment.

```
vectorizer = TfidfVectorizer(stop_words='english')
X = vectorizer.fit_transform(df['cleaned'])

# Create a 'sentiment' column based on 'reviews.rating'
# Assuming ratings > 3 are positive (1) and ratings <= 3 are negative (0)
df['sentiment'] = df['reviews.rating'].apply(lambda x: 1 if x > 3 else 0)
y = df['sentiment']
```

3.Train a classification model

Once text is converted into numerical features, we train a machine learning model to classify whether a review is positive, negative, or neutral. Traditional models like Naive Bayes work well on TF-IDF features, while deep-learning models ,it understand sentence context and long sequences. The model learns from labeled examples so it can accurately classify new reviews.

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

model = MultinomialNB()
model.fit(X_train, y_train)
```

```
▾ MultinomialNB
MultinomialNB()
```

4.Evaluate model performance

After training, we test the model using unseen data. Metrics like accuracy show overall correctness of 91% , while F1-score balances precision and recall—especially useful for imbalanced datasets. We also analyze misclassified reviews to understand where the model struggles, helping to fine-tune preprocessing or improve the model.

```
pred = model.predict(X_test)
print("Accuracy =", accuracy_score(y_test, pred))
print(classification_report(y_test, pred))
```

```
Accuracy = 0.9066213921901528
              precision    recall  f1-score   support

           0       1.00      0.02      0.04        56
           1       0.91      1.00      0.95       533

    accuracy                           0.91       589
   macro avg       0.95      0.51      0.49       589
weighted avg       0.92      0.91      0.86       589
```

5.NLP-Based Sentiment Classification

In this project, I performed sentiment analysis on product reviews using both machine learning and NLP-based AI. First, I trained a traditional ML model (such as Naive Bayes / TF-IDF) to classify reviews into positive, negative, or neutral categories. To enhance accuracy and add natural language reasoning, I also used the Gemini API, which analyzes user-entered multiple reviews and returns structured JSON sentiment results with the model "models/gemini-flash-latest". This hybrid approach combines statistical text classification with advanced generative reasoning. Overall, the system delivers more accurate, context-aware sentiment predictions suitable for real-time applications.

```
!pip install -U google-generativeai
import google.generativeai as genai
from google.colab import userdata

api_key = userdata.get("GEMINI_KEY")   # or GOOGLE_API_KEY based on your secret
genai.configure(api_key=api_key)
```

```
Requirement already satisfied: google-generativeai in /usr/local/lib/python3.12/dist-packages (0.8.5)
Requirement already satisfied: google-ai-generativelanguage==0.6.15 in /usr/local/lib/python3.12/dist-packages (from google-gene
Requirement already satisfied: google-api-core in /usr/local/lib/python3.12/dist-packages (from google-generativeai) (2.28.1)
Requirement already satisfied: google-api-python-client in /usr/local/lib/python3.12/dist-packages (from google-generativeai) (2
Requirement already satisfied: google-auth>=2.15.0 in /usr/local/lib/python3.12/dist-packages (from google-generativeai) (2.38.0
Requirement already satisfied: protobuf in /usr/local/lib/python3.12/dist-packages (from google-generativeai) (5.29.5)
Requirement already satisfied: pydantic in /usr/local/lib/python3.12/dist-packages (from google-generativeai) (2.11.10)
Requirement already satisfied: tqdm in /usr/local/lib/python3.12/dist-packages (from google-generativeai) (4.67.1)
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.12/dist-packages (from google-generativeai) (4.15.0)
Requirement already satisfied: proto-plus<2.0.0dev,>=1.22.3 in /usr/local/lib/python3.12/dist-packages (from google-ai-generativ
Requirement already satisfied: googleapis-common-protos<2.0.0,>=1.56.2 in /usr/local/lib/python3.12/dist-packages (from google-a
Requirement already satisfied: requests<3.0.0,>=2.18.0 in /usr/local/lib/python3.12/dist-packages (from google-api-core->google-
Requirement already satisfied: cachetools<6.0,>=2.0.0 in /usr/local/lib/python3.12/dist-packages (from google-auth>=2.15.0->goog
Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/python3.12/dist-packages (from google-auth>=2.15.0->googl
Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.12/dist-packages (from google-auth>=2.15.0->google-genera
Requirement already satisfied: httplib2<1.0.0,>=0.19.0 in /usr/local/lib/python3.12/dist-packages (from google-api-python-client
Requirement already satisfied: google-auth-httplib2<1.0.0,>=0.2.0 in /usr/local/lib/python3.12/dist-packages (from google-api-py
Requirement already satisfied: uritemplate<5,>=3.0.1 in /usr/local/lib/python3.12/dist-packages (from google-api-python-client->
Requirement already satisfied: annotated-types>=0.6.0 in /usr/local/lib/python3.12/dist-packages (from pydantic->google-generati
Requirement already satisfied: pydantic-core==2.33.2 in /usr/local/lib/python3.12/dist-packages (from pydantic->google-generativ
Requirement already satisfied: typing-inspection>=0.4.0 in /usr/local/lib/python3.12/dist-packages (from pydantic->google-genera
Requirement already satisfied: grpcio<2.0.0,>=1.33.2 in /usr/local/lib/python3.12/dist-packages (from google-api-core[grpc]!=2.0
Requirement already satisfied: grpcio-status<2.0.0,>=1.33.2 in /usr/local/lib/python3.12/dist-packages (from google-api-core[grp
Requirement already satisfied: pyparsing<4,>=3.0.4 in /usr/local/lib/python3.12/dist-packages (from httplib2<1.0.0,>=0.19.0->goo
Requirement already satisfied: pyasn1<0.7.0,>=0.6.1 in /usr/local/lib/python3.12/dist-packages (from pyasn1-modules>=0.2.1->goog
Requirement already satisfied: charset_normalizer<4,>=2 in /usr/local/lib/python3.12/dist-packages (from requests<3.0.0,>=2.18.0
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.12/dist-packages (from requests<3.0.0,>=2.18.0->google-api
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.12/dist-packages (from requests<3.0.0,>=2.18.0->goog
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.12/dist-packages (from requests<3.0.0,>=2.18.0->goog
```

```
model = genai.GenerativeModel("models/gemini-flash-latest")
```

```
text = input("Enter your review: ")

prompt = f"""
Analyze the sentiment of the following product review.

Return response in this format:

Sentiment: <positive/negative/neutral>
Reason: <short explanation>

Review: "{text}"
"""

response = model.generate_content(prompt)
print("\nOutput:")
print(response.text)
```

```
Enter your review: i love the red color bowl and the white bowl is not good

Output:
Sentiment: neutral
Reason: The review contains mixed sentiment, explicitly praising one part of the product ("love the red color bowl") while expli
```

```
input_text = input("Enter multiple reviews separated by commas:\n")

# Convert to list
reviews = [r.strip() for r in input_text.split(",")]

# ---- PROCESS REVIEWS ----
results = []

for review in reviews:
    prompt = f"""
    Analyze the sentiment of this review and return valid JSON ONLY.

    {{
      "review": "{review}",
      "sentiment": "",
      "reason": "",
      "confidence": ""
    }}
    """

    response = model.generate_content(prompt)
    results.append(response.text)

# ---- PRINT RESULTS ----
print("\n---- OUTPUT ----\n")
for r in results:
    print(r, "\n")
```

```
Enter multiple reviews separated by commas:
i like the cup due to it's beautiful design ,where as the bottle looks unsatisfied due to it's bad smell

---- OUTPUT ----

```json
{
  "review": "i like the cup due to it's beautiful design",
  "sentiment": "Positive",
  "reason": "The review uses the strongly positive words 'like' and 'beautiful design' to describe the cup.",
  "confidence": "High"
}
```

```json
{
  "review": "where as the bottle looks unsatisfied due to it's bad smell",
  "sentiment": "Negative",
  "reason": "The review explicitly uses negative descriptors ('unsatisfied,' 'bad smell') to criticize the product's appearance
  "confidence": "High"
}
```
```