**AIM:** Build a simple sentiment analysis classifier using bag-of-words or TF-IDF and Naïve Bayes.

**DESCRIPTION:**

The objective of this project is to develop a simple text classification system that can automatically identify the **sentiment** (positive or negative) expressed in textual data such as reviews, feedback, or comments using **Natural Language Processing (NLP)** techniques.

Sentiment analysis is one of the most common applications of NLP and machine learning.  
In this project, we build a **sentiment classifier** that determines whether a given text expresses a **positive** or **negative** opinion.

The process involves the following steps:

1. **Data Collection:**
   * Create or load a small dataset of sentences or product reviews labeled as *positive* or *negative*.
2. **Text Preprocessing:**
   * Convert text to lowercase.
   * Remove punctuation, special symbols, or stopwords (optional).
3. **Feature Extraction (Bag-of-Words or TF-IDF):**
   * Represent each sentence as a numerical vector.
   * **Bag-of-Words (BoW):** Counts the frequency of each word in the text.
   * **TF-IDF (Term Frequency–Inverse Document Frequency):** Gives higher weight to unique words that are more informative.
4. **Model Training (Naïve Bayes):**
   * Use the **Multinomial Naïve Bayes** algorithm, which works well for text classification.
   * The model learns from the word patterns to predict sentiment labels.
5. **Model Evaluation:**
   * Evaluate model performance using metrics like **accuracy**, **precision**, **recall**, and **F1-score**.
6. **Prediction:**
   * Input new text or sentences and let the model predict whether the sentiment is *positive* or *negative*.

**Algorithm Used:**

**Naïve Bayes Classifier**

* Based on Bayes’ Theorem and the assumption that features (words) are independent.
* Efficient and effective for text classification tasks.

**Technologies and Libraries:**

* **Python 3.x**
* **scikit-learn** (for TF-IDF/BoW, model training, and evaluation)
* **NumPy** (optional for handling arrays)
* **pandas** (optional for datasets)

**Expected Output:**

* The model prints:
  + Training and testing accuracy.
  + A classification report.
  + Predictions for new sentences such as:
    - *“I love this product!” → Positive*
    - *“This is a terrible experience.” → Negative*

**Outcome:**

By the end of this project, you will have a simple yet complete **sentiment analysis system** that:

* Converts raw text into numerical features.
* Trains a machine learning classifier.
* Evaluates sentiment polarity of unseen text data.

**PROGRAM:**

# ---------------- Sentiment Analysis ----------------  
try:  
 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  
except ModuleNotFoundError:  
 import os  
 os.system("pip install scikit-learn")  
 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  
  
# ---------------- SAMPLE DATASET ----------------  
# Example dataset: text + sentiment (1=positive, 0=negative)  
texts = [  
 "I love this product, it is amazing!",  
 "This is the worst experience I have ever had.",  
 "Absolutely fantastic, I am very happy!",  
 "I hate it, very disappointing.",  
 "Best purchase ever, highly recommend it.",  
 "Not good, I will never buy this again.",  
 "Excellent quality and great service.",  
 "Terrible, completely useless.",  
]  
  
labels = [1, 0, 1, 0, 1, 0, 1, 0]  
  
# ---------------- VECTORIZE TEXT ----------------  
vectorizer = TfidfVectorizer()  
X = vectorizer.fit\_transform(texts)  
y = labels  
  
# ---------------- SPLIT DATA ----------------  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.25, random\_state=42)  
  
# ---------------- TRAIN NAIVE BAYES ----------------  
model = MultinomialNB()  
model.fit(X\_train, y\_train)  
  
# ---------------- PREDICT & EVALUATE ----------------  
y\_pred = model.predict(X\_test)  
  
print("Accuracy:", accuracy\_score(y\_test, y\_pred))  
print("\nClassification Report:\n", classification\_report(y\_test, y\_pred))  
  
# ---------------- TEST WITH NEW TEXT ----------------  
new\_texts = [  
 "I am extremely satisfied with this product!",  
 "This is a horrible product, very bad experience."  
]  
  
new\_X = vectorizer.transform(new\_texts)  
predictions = model.predict(new\_X)  
  
for text, pred in zip(new\_texts, predictions):  
 sentiment = "Positive" if pred == 1 else "Negative"  
 print(f"Text: {text}\nPredicted Sentiment: {sentiment}\n")

**OUTPUT:**

"C:\Users\Surekha Swarna\PycharmProjects\NLP3RDYEARPROJ\.venv\Scripts\python.exe" "C:\Users\Surekha Swarna\PycharmProjects\NLP3RDYEARPROJ\app.py"

Accuracy: 0.0

C:\Users\Surekha Swarna\PycharmProjects\NLP3RDYEARPROJ\.venv\Lib\site-packages\sklearn\metrics\\_classification.py:1731: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

\_warn\_prf(average, modifier, f"{metric.capitalize()} is", result.shape[0])

C:\Users\Surekha Swarna\PycharmProjects\NLP3RDYEARPROJ\.venv\Lib\site-packages\sklearn\metrics\\_classification.py:1731: UndefinedMetricWarning: Recall is ill-defined and being set to 0.0 in labels with no true samples. Use `zero\_division` parameter to control this behavior.

\_warn\_prf(average, modifier, f"{metric.capitalize()} is", result.shape[0])

C:\Users\Surekha Swarna\PycharmProjects\NLP3RDYEARPROJ\.venv\Lib\site-packages\sklearn\metrics\\_classification.py:1731: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

\_warn\_prf(average, modifier, f"{metric.capitalize()} is", result.shape[0])

C:\Users\Surekha Swarna\PycharmProjects\NLP3RDYEARPROJ\.venv\Lib\site-packages\sklearn\metrics\\_classification.py:1731: UndefinedMetricWarning: Recall is ill-defined and being set to 0.0 in labels with no true samples. Use `zero\_division` parameter to control this behavior.

\_warn\_prf(average, modifier, f"{metric.capitalize()} is", result.shape[0])

C:\Users\Surekha Swarna\PycharmProjects\NLP3RDYEARPROJ\.venv\Lib\site-packages\sklearn\metrics\\_classification.py:1731: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.

\_warn\_prf(average, modifier, f"{metric.capitalize()} is", result.shape[0])

C:\Users\Surekha Swarna\PycharmProjects\NLP3RDYEARPROJ\.venv\Lib\site-packages\sklearn\metrics\\_classification.py:1731: UndefinedMetricWarning: Recall is ill-defined and being set to 0.0 in labels with no true samples. Use `zero\_division` parameter to control this behavior.

\_warn\_prf(average, modifier, f"{metric.capitalize()} is", result.shape[0])

Classification Report:

precision recall f1-score support

0 0.00 0.00 0.00 2.0

1 0.00 0.00 0.00 0.0

accuracy 0.00 2.0

macro avg 0.00 0.00 0.00 2.0

weighted avg 0.00 0.00 0.00 2.0

Text: I am extremely satisfied with this product!

Predicted Sentiment: Positive

Text: This is a horrible product, very bad experience.

Predicted Sentiment: Positive

Process finished with exit code 0

**RESULT:** Hence, building a simple sentiment analysis classifier using bag-of-words or TF-IDF and Naïve Bayes has been successfully implemented.