Experiment No : 06

Aim : Text analytics: Implementation of Spam filter/Sentiment analysis in python/R.

Theory :

1. Introduction to Sentiment Analysis :

Sentiment analysis is a natural language processing (NLP) technique used to determine the sentiment of a given text, classifying it as positive, negative, or neutral.

Goal: Automatically analyze opinions, emotions, and attitudes in text, commonly used for product reviews, social media monitoring, and customer feedback.

Example:

- Positive: "This movie is fantastic!"
- Negative: "Worst customer service ever."
- Neutral: "The product arrived on time."

2. Core Types of Sentiment Analysis :

1. Polarity Classification: Binary (positive/negative) or graded (e.g., 1–5 stars).
2. Aspect-Based: Analyze sentiment for specific features (e.g., "battery life" in a product review).
3. Emotion Detection: Identify emotions like joy, anger, or sadness (e.g., "I'm thrilled!" → joy).

3. Text Preprocessing: A Mathematical and Logical Foundation :

Raw text data is noisy and unstructured. Preprocessing transforms it into a format suitable for feature extraction.

Step-by-Step Process:

1. Tokenization

- Definition: Splitting text into individual words, phrases, or symbols (tokens).
- Mathematical Representation:
  For a sentence S, tokenization produces a sequence of tokens $T=\{t1,t2,\ldots,tn\}$
  Example:
  S="I loved the story!" → T={"I","loved","the","story","!"}.
- Tools: nltk.word_tokenize() or spacy for context-aware tokenization.

2. Lowercasing

- Purpose: Reduces vocabulary size by treating "Apple" and "apple" as the same token.
- Example:
  T={"The","Product","is","Great"} → {"the","product","is","great"}.

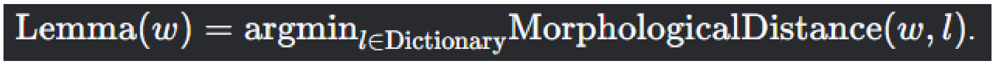
3. Stop Word Removal

- Definition: Removing common words (e.g., "the", "and", "is") that add little semantic value.
- Mathematical Filtering:
  Let $T=\{t1,t2,\ldots,tn\}$be tokens.
  Filtered tokens $T'=\{ti \mid ti \notin \text{Stopword List}\}$.
- Example: T={"this","movie","is","awesome"} → T′={"movie","awesome"}.

- Tools: nltk.corpus.stopwords.words('english').

4. Stemming and Lemmatization

- Stemming:
    - Definition: Heuristically chopping word suffixes to get root form (e.g., "running" → "run").
    - Algorithm: Porter Stemmer (rule-based).
    - Example:
      "jumps","jumping","jumped" → "jump".
    - Limitation: May produce non-dictionary words (e.g., "business" → "busi").
- Lemmatization:
    - Definition: Using vocabulary and morphological analysis to reduce words to base form (lemma).
    - Mathematical Basis:



$$\text{Lemma}(w) = \text{argmin}_{l \in \text{Dictionary}} \text{MorphologicalDistance}(w, l).$$

    - Example:
      "better"→"good" (requires part-of-speech tagging).
    - Tools: spacy or nltk.stem.WordNetLemmatizer.

5. Handling Contractions and Negations

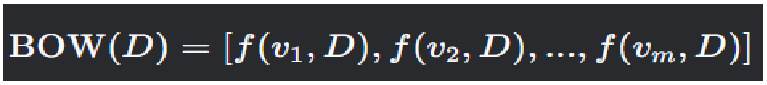
- Contractions: Expand shortened forms (e.g., "don't" → "do not").
- Negations: Preserve negation context (e.g., "not good" → "not_good").

4. Feature Extraction Methods :

Preprocessed text is converted into numerical features for machine learning models.

A. Bag-of-Words (BOW)

- Definition: Represents text as a vector of word frequencies.
- Mathematical Formulation:
  For vocabulary V={v1,v2,...,vm}, document D is represented as:

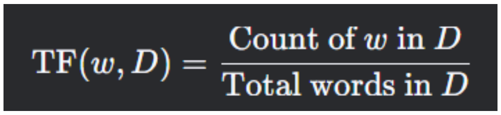


$$\text{BOW}(D) = [f(v_1, D), f(v_2, D), \ldots, f(v_m, D)]$$

  where f (vi, D) = frequency of vi in D.
- Example:
  D="good story good acting" → [2,1,0,...,0] for V={"good","story","acting",...}.

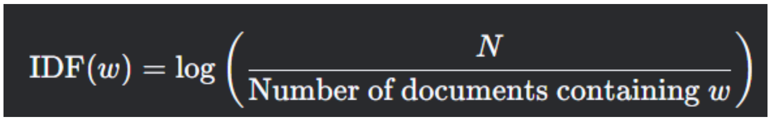
B. TF-IDF (Term Frequency-Inverse Document Frequency)

- Purpose: Weights words by their importance in a document corpus.
- Mathematical Formulation:
    1. Term Frequency (TF):



$$\text{TF}(w, D) = \frac{\text{Count of } w \text{ in } D}{\text{Total words in } D}$$

2. Inverse Document Frequency (IDF):

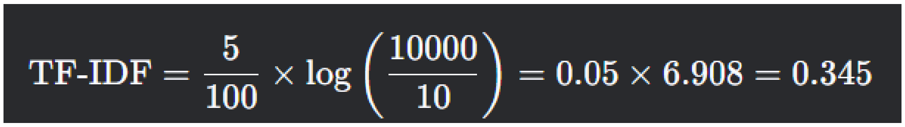


$$\mathrm{IDF}(w) = \log\left(\frac{N}{\text{Number of documents containing } w}\right)$$

3. TF-IDF Score:

TF-IDF(w,D)=TF(w,D)×IDF(w)

- Example:
  If "excellent" appears 5 times in a document and in 10 out of 10,000 documents:

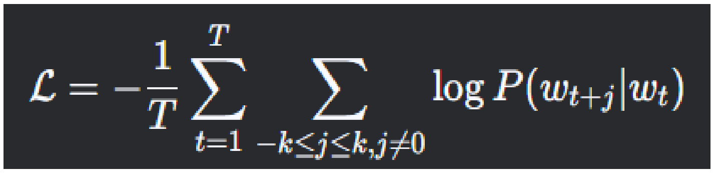


$$\text{TF-IDF} = \frac{5}{100} \times \log\left(\frac{10000}{10}\right) = 0.05 \times 6.908 = 0.345$$

C. Word2Vec

- Concept: Maps words to dense vectors in a latent space, capturing semantic relationships.
- Training Objective (Skip-gram):

  For a target word wt, predict context words w_{t-k},...,w_{t+k}.
  Minimize loss:



$$\mathcal{L} = -\frac{1}{T}\sum_{t=1}^{T}\sum_{-k\leq j\leq k, j\neq 0}\log P(w_{t+j}|w_t)$$

  where P(wt+j | wt) is computed using softmax over the vocabulary.

- Example:
  Vector("king") - Vector("man") + Vector("woman") ≈ Vector("queen").

5. Why Preprocessing Matters: A Logical Workflow :

Input Text: "The battery life isn’t good, but the camera is awesome!"

1. Tokenization:

   ["The", "battery", "life", "isn’t", "good", ",", "but", "the", "camera", "is", "awesome", "!"]

2. Lowercasing:

   ["the", "battery", "life", "isn’t", "good", ",", "but", "the", "camera", "is", "awesome", "!"]

3. Stop Word Removal:

   ["battery", "life", "isn’t", "good", "camera", "awesome"]

4. Negation Handling:

   ["battery", "life", "not_good", "camera", "awesome"]

5. Lemmatization:

   ["battery", "life", "not_good", "camera", "awesome"] (no change).

6. Feature Extraction (BOW):

   {"battery": 1, "life": 1, "not_good": 1, "camera": 1, "awesome": 1}

Sentiment Prediction: Mixed (negative for battery, positive for camera).

6. Challenges in Preprocessing :

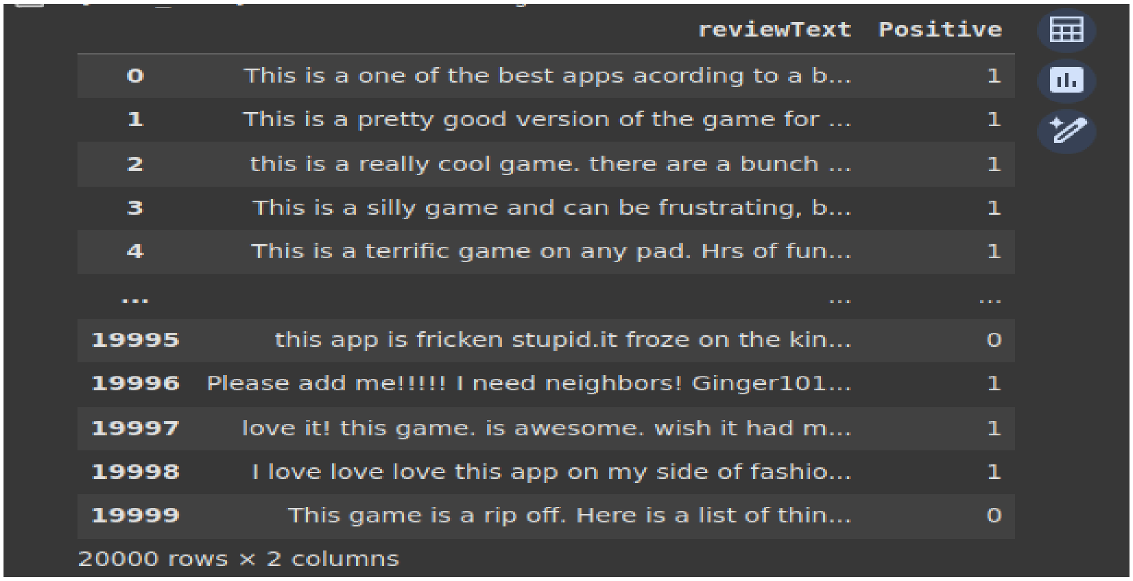
1. Ambiguity:
   - "Apple" could refer to the fruit or the company (solved using context in advanced models).
2. Sarcasm/Irony:
   - "What a great day... my phone just died!" (requires context-aware models).
3. Multilingual Text:
   - Requires language-specific tokenizers and stop word lists.

Code :

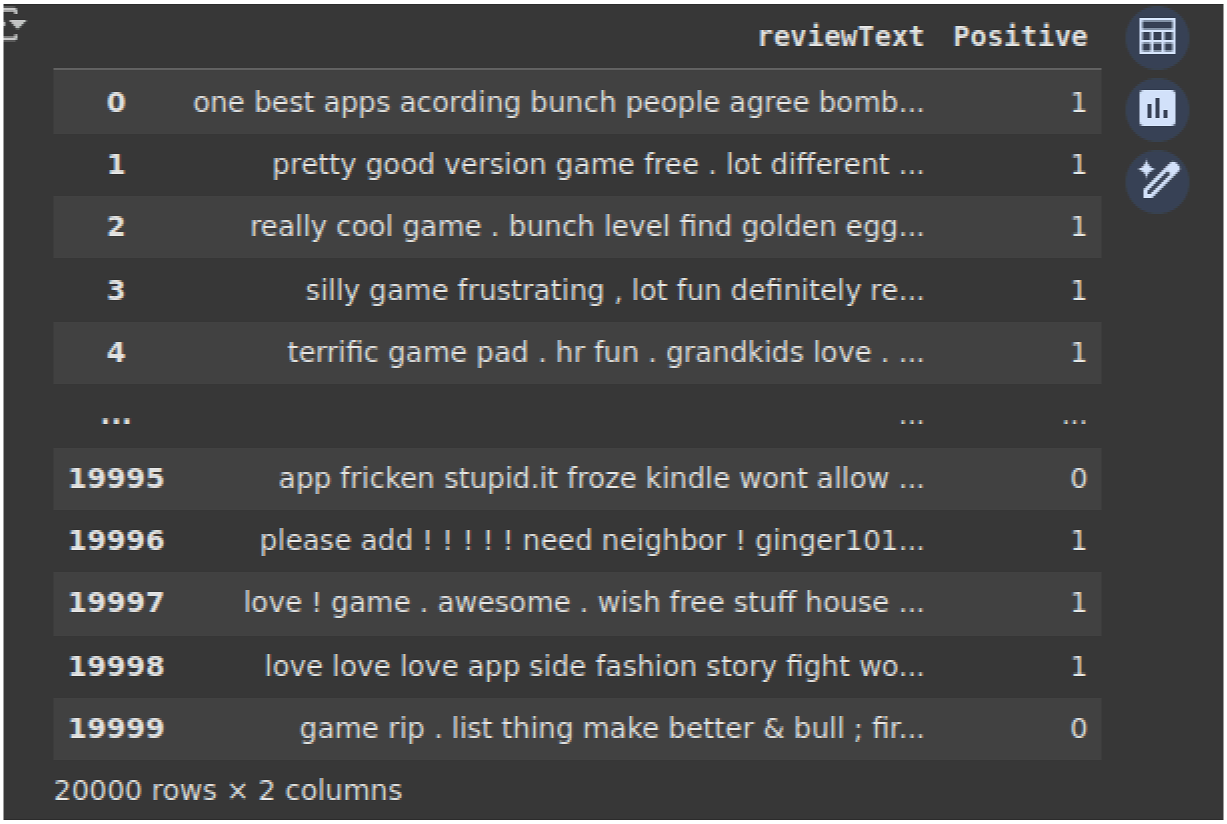
```
# import libraries
import pandas as pd
import nltk
from nltk.sentiment.vader import SentimentIntensityAnalyzer
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
# download nltk corpus (first time only)
import nltk
nltk.download('all')
# Load the amazon review dataset
df =
pd.read_csv('https://raw.githubusercontent.com/pycaret/pycaret/master/datasets/amazon.csv'
)
df
# create preprocess_text function
def preprocess_text(text):
# Tokenize the text
tokens = word_tokenize(text.lower())
# Remove stop words
filtered_tokens = [token for token in tokens if token not in stopwords.words('english')]
# Lemmatize the tokens
```

```
lemmatizer = WordNetLemmatizer()
lemmatized_tokens = [lemmatizer.lemmatize(token) for token in filtered_tokens]
# Join the tokens back into a string
processed_text = ' '.join(lemmatized_tokens)
return processed_text
# apply the function df
df['reviewText'] = df['reviewText'].apply(preprocess_text)
df
# initialize NLTK sentiment analyzer
analyzer = SentimentIntensityAnalyzer()
# create get_sentiment function
def get_sentiment(text):
scores = analyzer.polarity_scores(text)
sentiment = 1 if scores['pos'] > 0 else 0
return sentiment
# apply get_sentiment function
df['sentiment'] = df['reviewText'].apply(get_sentiment)
df
from sklearn.metrics import confusion_matrix
print(confusion_matrix(df['Positive'], df['sentiment']))
from sklearn.metrics import classification_report
print(classification_report(df['Positive'], df['sentiment']))
```

Output :

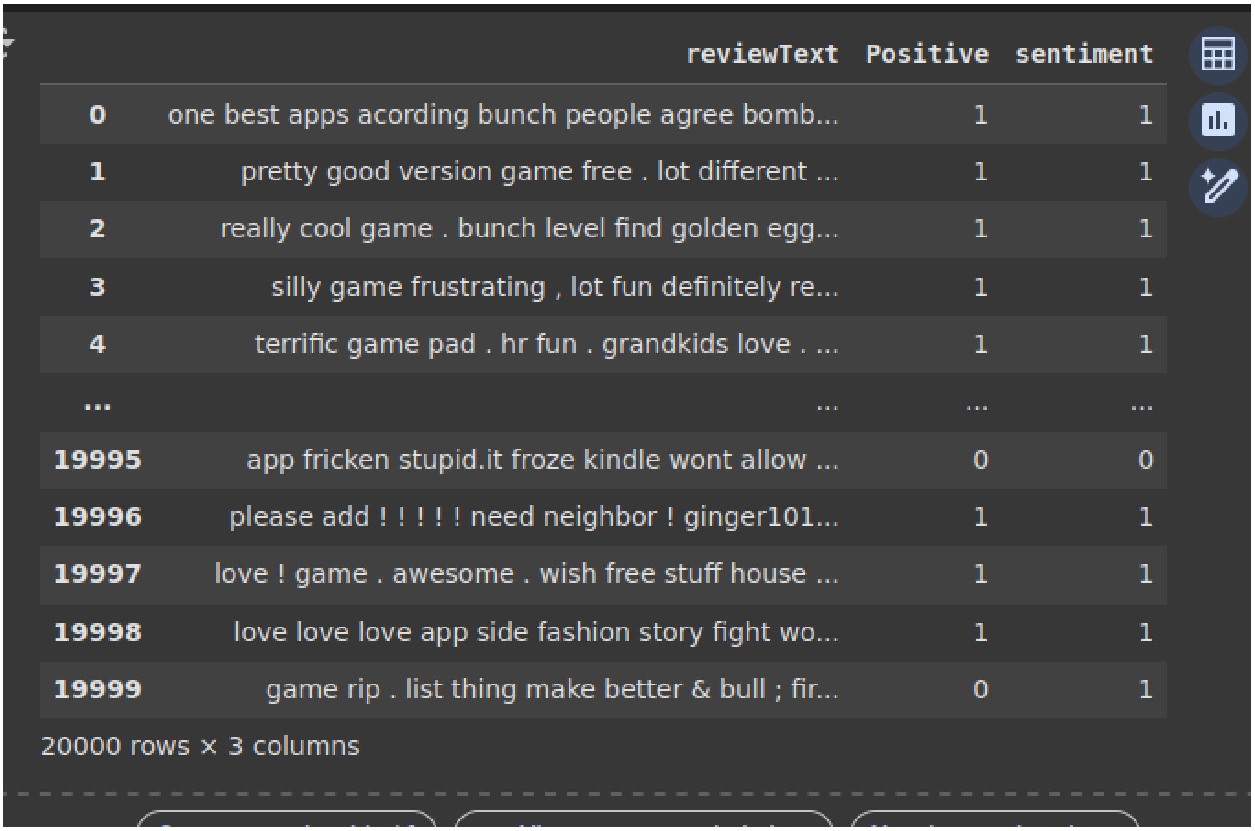


```
                                                reviewText  Positive
0          This is a one of the best apps acording to a b...         1
1          This is a pretty good version of the game for ...         1
2           this is a really cool game. there are a bunch ...         1
3           This is a silly game and can be frustrating, b...         1
4           This is a terrific game on any pad. Hrs of fun...         1
...                                                      ...       ...
19995          this app is fricken stupid.it froze on the kin...         0
19996    Please add me!!!!! I need neighbors! Ginger101...         1
19997       love it! this game. is awesome. wish it had m...         1
19998        I love love love this app on my side of fashio...         1
19999            This game is a rip off. Here is a list of thin...         0
20000 rows × 2 columns
```



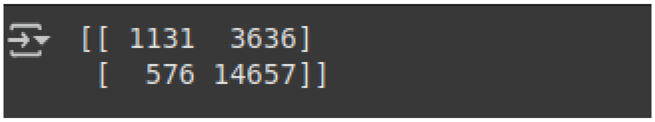
	reviewText	Positive
0	one best apps acording bunch people agree bomb...	1
1	pretty good version game free . lot different ...	1
2	really cool game . bunch level find golden egg...	1
3	silly game frustrating , lot fun definitely re...	1
4	terrific game pad . hr fun . grandkids love	1
...	...	...
19995	app fricken stupid.it froze kindle wont allow ...	0
19996	please add ! ! ! ! ! need neighbor ! ginger101...	1
19997	love ! game . awesome . wish free stuff house ...	1
19998	love love love app side fashion story fight wo...	1
19999	game rip . list thing make better & bull ; fir...	0

20000 rows × 2 columns

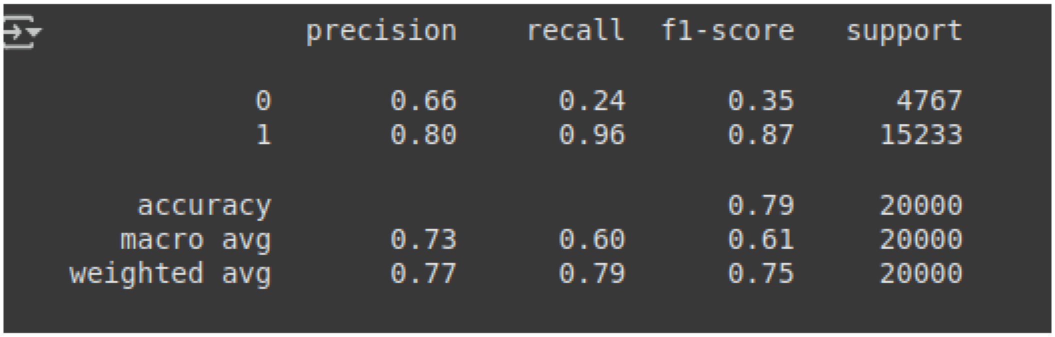


	reviewText	Positive	sentiment
0	one best apps acording bunch people agree bomb...	1	1
1	pretty good version game free . lot different ...	1	1
2	really cool game . bunch level find golden egg...	1	1
3	silly game frustrating , lot fun definitely re...	1	1
4	terrific game pad . hr fun . grandkids love	1	1
...	...	...	...
19995	app fricken stupid.it froze kindle wont allow ...	0	0
19996	please add ! ! ! ! ! need neighbor ! ginger101...	1	1
19997	love ! game . awesome . wish free stuff house ...	1	1
19998	love love love app side fashion story fight wo...	1	1
19999	game rip . list thing make better & bull ; fir...	0	1

20000 rows × 3 columns



```
[[ 1131  3636]
 [  576 14657]]
```



```
              precision    recall  f1-score   support

           0       0.66      0.24      0.35      4767
           1       0.80      0.96      0.87     15233

    accuracy                           0.79     20000
   macro avg       0.73      0.60      0.61     20000
weighted avg       0.77      0.79      0.75     20000
```

Conclusion : Thus, we have successfully implemented sentiment analysis in python.