EX.NO.: 15
DATE: 03.03.2025

NAIVE BAYES ON NLTK MOVIE REVIEW DATASET

To implement a **Naïve Bayes classifier** using the **NLTK** library for text classification on two datasets:

1. **Movie Reviews Dataset** from NLTK.
2. **20 Newsgroups Dataset** from scikit-learn.

PROCEDURE:

Part 1: Movie Reviews Classification

1. Import necessary libraries (nltk, random).
2. Download and load the movie_reviews dataset from NLTK.
3. Store reviews as (word_list, category) tuples and shuffle them.
4. Define a document_features() function to extract word features.
5. Convert data into feature sets using document_features().
6. Split the dataset into **training (80%)** and **testing (20%)** sets.
7. Train a **Naïve Bayes Classifier** using the training set.
8. Evaluate model performance using **accuracy** on the test set.
9. Display the **most informative features** used by the classifier.

Part 2: 20 Newsgroups Classification

1. Import required libraries (sklearn.datasets, nltk).
2. Load the **20 Newsgroups** dataset (subset of 4 categories).
3. Remove **headers, footers, and quotes** to clean the text.
4. Convert text data into numerical features using CountVectorizer (bag-of-words).
5. Apply TfidfTransformer to normalize feature importance.
6. Convert feature matrices into **NLTK-compatible dictionaries**.
7. Split the dataset into **training and testing** sets.
8. Train a **Naïve Bayes Classifier** using the processed feature sets.
9. Evaluate the model using **accuracy** on the test set.
10. Display the **most informative words** influencing classification.

CODE AND OUTPUT

```
import nltk
from nltk.corpus import movie_reviews
import random
from nltk.classify import NaiveBayesClassifier
from nltk.classify.util import accuracy

# Download dataset if not available
nltk.download('movie_reviews')

# Load dataset
documents = [(list(movie_reviews.words(fileid)), category)
             for category in movie_reviews.categories()
             for fileid in movie_reviews.fileids(category)]

# Shuffle the data
random.shuffle(documents)

# Feature extraction
```

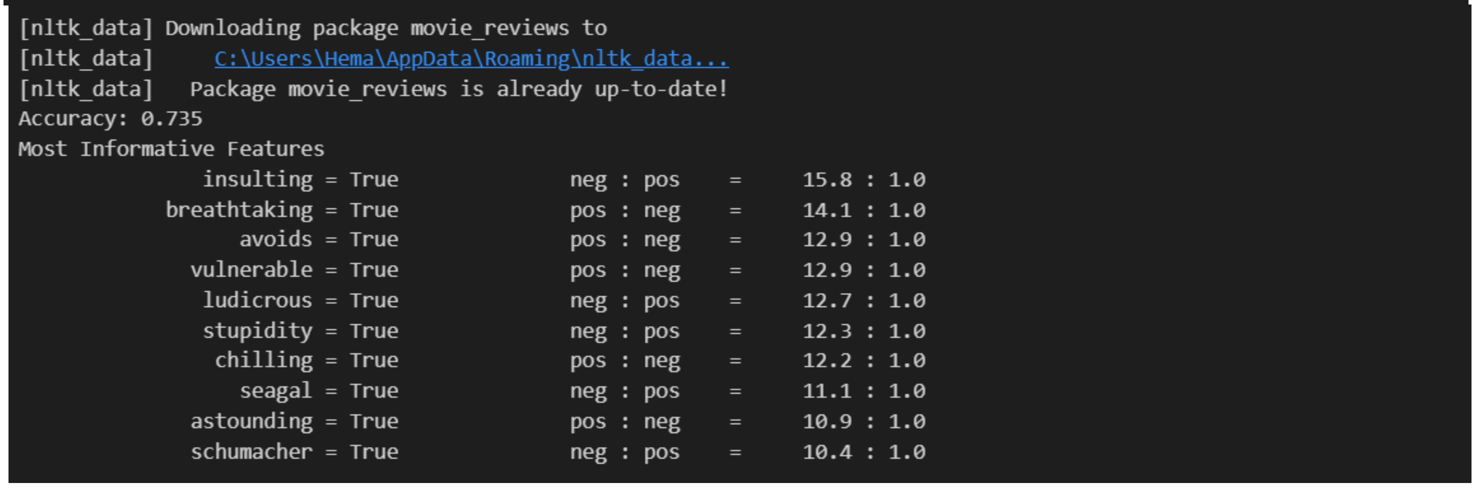
```
def document_features(words):
    return {word: True for word in words}

# Split dataset into training and testing
feature_sets = [(document_features(doc), category) for (doc, category) in documents]
train_set, test_set = feature_sets[:1600], feature_sets[1600:]

# Train Naive Bayes Classifier
classifier = NaiveBayesClassifier.train(train_set)

# Evaluate the classifier
print("Accuracy:", accuracy(classifier, test_set))

# Show the most informative features
classifier.show_most_informative_features(10)
```



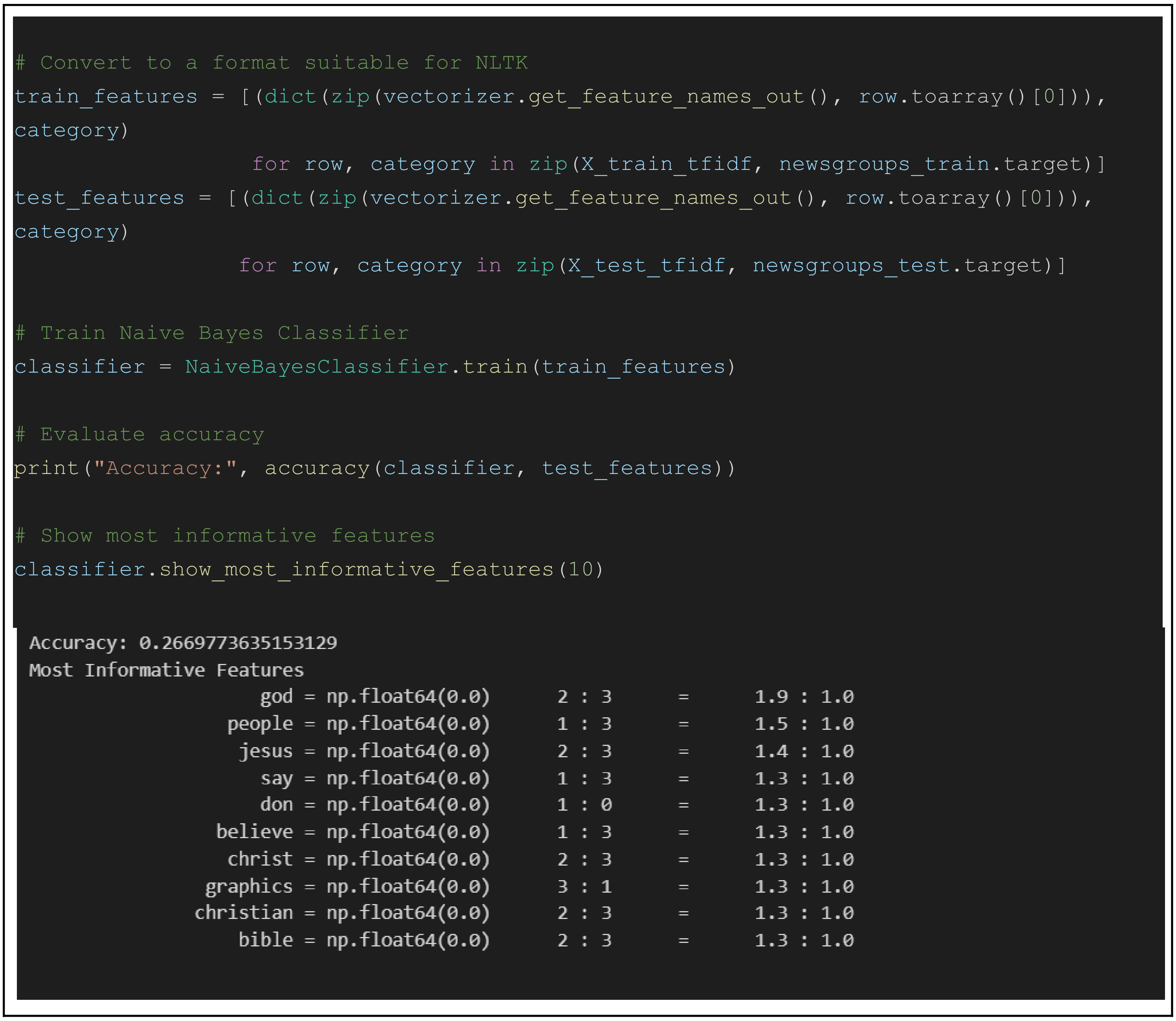
```
[nltk_data] Downloading package movie_reviews to
[nltk_data]     C:\Users\Hema\AppData\Roaming\nltk_data...
[nltk_data]   Package movie_reviews is already up-to-date!
Accuracy: 0.735
Most Informative Features
               insulting = True              neg : pos    =     15.8 : 1.0
            breathtaking = True              pos : neg    =     14.1 : 1.0
                  avoids = True              pos : neg    =     12.9 : 1.0
              vulnerable = True              pos : neg    =     12.9 : 1.0
               ludicrous = True              neg : pos    =     12.7 : 1.0
               stupidity = True              neg : pos    =     12.3 : 1.0
                chilling = True              pos : neg    =     12.2 : 1.0
                  seagal = True              neg : pos    =     11.1 : 1.0
              astounding = True              pos : neg    =     10.9 : 1.0
              schumacher = True              neg : pos    =     10.4 : 1.0
```

```
from sklearn.datasets import fetch_20newsgroups
from sklearn.feature_extraction.text import CountVectorizer, TfidfTransformer
from nltk.classify import NaiveBayesClassifier
from nltk.classify.util import accuracy
import nltk

# Load dataset
categories = ['alt.atheism', 'comp.graphics', 'sci.med', 'soc.religion.christian']
newsgroups_train = fetch_20newsgroups(subset='train', categories=categories,
remove=('headers', 'footers', 'quotes'))
newsgroups_test = fetch_20newsgroups(subset='test', categories=categories,
remove=('headers', 'footers', 'quotes'))

# Convert text to feature vectors
vectorizer = CountVectorizer(stop_words='english', max_features=3000)
X_train_counts = vectorizer.fit_transform(newsgroups_train.data)
X_test_counts = vectorizer.transform(newsgroups_test.data)

# Convert counts to TF-IDF features
tfidf_transformer = TfidfTransformer()
X_train_tfidf = tfidf_transformer.fit_transform(X_train_counts)
X_test_tfidf = tfidf_transformer.transform(X_test_counts)
```



```
# Convert to a format suitable for NLTK
train_features = [(dict(zip(vectorizer.get_feature_names_out(), row.toarray()[0])),
category)
                  for row, category in zip(X_train_tfidf, newsgroups_train.target)]
test_features = [(dict(zip(vectorizer.get_feature_names_out(), row.toarray()[0])),
category)
                 for row, category in zip(X_test_tfidf, newsgroups_test.target)]

# Train Naive Bayes Classifier
classifier = NaiveBayesClassifier.train(train_features)

# Evaluate accuracy
print("Accuracy:", accuracy(classifier, test_features))

# Show most informative features
classifier.show_most_informative_features(10)
```

```
Accuracy: 0.2669773635153129
Most Informative Features
                     god = np.float64(0.0)      2 : 3      =      1.9 : 1.0
                  people = np.float64(0.0)      1 : 3      =      1.5 : 1.0
                   jesus = np.float64(0.0)      2 : 3      =      1.4 : 1.0
                     say = np.float64(0.0)      1 : 3      =      1.3 : 1.0
                     don = np.float64(0.0)      1 : 0      =      1.3 : 1.0
                 believe = np.float64(0.0)      1 : 3      =      1.3 : 1.0
                  christ = np.float64(0.0)      2 : 3      =      1.3 : 1.0
                graphics = np.float64(0.0)      3 : 1      =      1.3 : 1.0
               christian = np.float64(0.0)      2 : 3      =      1.3 : 1.0
                   bible = np.float64(0.0)      2 : 3      =      1.3 : 1.0
```