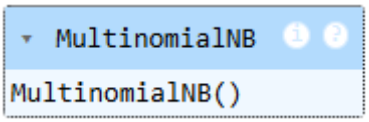
Practical No.5

Aim: Text Categorization

- **Implement a text classification algorithm (e.g., Naive Bayes or Support Vector Machines).**
- **Train the classifier on a labelled dataset and evaluate its performance.**

```
#Step 1 Import Necessary Libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, classification_report

#Step 2: Load the training dataset
#Read the CSV file that contains training data.
df = pd.read_csv("/content/Dataset - Dataset.csv")

#Step 3: Combine text columns
data = df["covid"] + " " + df["fever"]
X = data.astype(str)   # Features (text)
y = df["flu"]          # Labels

#Step 4: Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42)
#80% data for learning, 20% data for testing accuracy.

#Step 5: Convert text into numbers (Bag of Words)
vectorizer = CountVectorizer()
X_train_counts = vectorizer.fit_transform(X_train)
X_test_counts = vectorizer.transform(X_test)

#Step 6: Train Naive Bayes Classifier
classifier = MultinomialNB()
classifier.fit(X_train_counts, y_train)
```




```
#Step 7: Evaluate the model
y_pred = classifier.predict(X_test_counts)
#Predicts output for test data.
accuracy = accuracy_score(y_test, y_pred)
#Predicts output for test data.
print(f"Accuracy: {accuracy:.2f}")
#Displays accuracy value.
```

```
print("Classification Report:")
print(classification_report(y_test, y_pred)) #Shows precision, recall, and F1-score.
```

```
Accuracy: 1.00
Classification Report:
              precision    recall  f1-score   support

          No       1.00      1.00      1.00         1
         Yes       1.00      1.00      1.00         2

    accuracy                           1.00         3
   macro avg       1.00      1.00      1.00         3
weighted avg       1.00      1.00      1.00         3
```

```
#Step 8: Test on new (unseen) dataset
data1 = pd.read_csv("/content/Test - Test.csv")
#Loads new unseen data for prediction.
new_data = data1["covid"] + " " + data1["fever"]
#Combines symptoms into single text.
new_data_counts = vectorizer.transform(new_data.astype(str))
#Converts new text into numeric form.
predictions = classifier.predict(new_data_counts)
#Predicts whether flu = Yes or No.

#Step 9: Save predictions to CSV
predictions_df = pd.DataFrame(predictions, columns=["flu_prediction"])
#Converts predictions into table format.
data1 = pd.concat([data1, predictions_df], axis=1)
#Adds predictions as a new column.
data1.to_csv("Downloads\Test_output.csv", index=False)
#Saves results to a new CSV file.
print("Predictions saved successfully!")
```

```
Predictions saved successfully!
<>:9: SyntaxWarning: invalid escape sequence '\T'
<>:9: SyntaxWarning: invalid escape sequence '\T'
/tmp/ipython-input-4185696482.py:9: SyntaxWarning: invalid escape sequence '\T'
  data1.to_csv("Downloads\Test_output.csv", index=False)
```