**Experiment3.2**

**Student Name:-** SANJIV GUPTA

**Branch:-** B.E(CSE)

**Semester:-** Vth

**UID:-** 21BCS3478

**Section/Group:-**602/B

**Date of Performance:-** 01/11/23

**Subject Name:-** Artificial Intelligence **Subject Code:-** 21CSH-316

& Machine Learning

**Aim:-** *Implement Naive Bayes theorem to classify the English text.*

**Objective:-**

*How to calculate the probabilities required by the Naive Bayes algorithm.*

*How to implement the Naive Bayes algorithm from scratch.*

*How to apply Naive Bayes to a real-world predictive modeling problem.*

# Procedure/Algorithm/Code:-

*First we will develop each piece of the algorithm, then we will tie all of the elements together into a working implementation applied to a real dataset.*

*This Naive Bayes tutorial is broken down into 5 parts:*

*Step 1: Separate By Class. Step 2: Summarize Dataset.*

*Step 3: Summarize Data By Class.*

*Step 4: Gaussian Probability Density Function. Step 5: Class Probabilities.*

*These steps will provide the foundation that you need to implement Naive Bayes from scratch and apply it to your own predictive modeling problems.*

# Source Code:-

*from collections import defaultdict import math*

# *Sample training data (text samples with corresponding labels) training\_data = [*

*("I love programming", "positive"), ("I hate bugs", "negative"), ("Coding is fun", "positive"), ("Bugs are annoying", "negative")*

*]*

# *Step 1: Data Preparation vocabulary = set()*

*class\_word\_counts = defaultdict(lambda: defaultdict(int)) class\_counts = defaultdict(int)*

*for text, label in training\_data: words = text.lower().split() vocabulary.update(words) class\_counts[label] += 1 for word in words:*

*class\_word\_counts[label][word] += 1 # Step 2: Calculate Class Priors*

total\_samples = len(training\_data)

class\_priors = {label: count / total\_samples for label, count in class\_counts.items()}

# Step 3: Calculate Likelihood Probabilities word\_likelihoods = defaultdict(lambda: defaultdict(float)) for label, word\_counts in class\_word\_counts.items():

total\_words\_in\_class = sum(word\_counts.values()) for word in vocabulary:

word\_likelihoods[label][word] = (word\_counts[word] + 1) / (total\_words\_in\_class + len(vocabulary))

# Step 4: Text Classification (Example for a new text) new\_text = "I love coding bugs" tokenized\_text = new\_text.lower().split()

class\_scores = defaultdict(float) for label in class\_priors:

class\_scores[label] = math.log(class\_priors[label]) # Prior probability for word in tokenized\_text:

if word in vocabulary:

class\_scores[label] += math.log(word\_likelihoods[label][word]) #

Likelihood probability

predicted\_class = max(class\_scores, key=class\_scores.get) print("Predicted Class:", predicted\_class)

**OUTPUT:**

****

**Learning Outcomes:-**

* *Learnt about Data Summary.*
* *Learnt about Dara Cleaning.*
* *Implementation of Data Visualization.*