Name: **Harsh Pramod Padyal**

Roll No : **40**

Div: **D20B**

Experiment 04

AIM: To build an adaptive and contextual Cognitive based Customer service application/ Insurance/ Healthcare Application/ Smarter Cities/ Government etc.

Theory

A **cognitive-based application** is a system that uses artificial intelligence to understand data in a human-like way and provide adaptive solutions. Unlike traditional systems that only follow fixed rules, cognitive systems can **learn from data, adjust to context, and give meaningful outputs**.

The aim of this experiment is to build an **adaptive and contextual application** that can be applied to different domains like **Customer Service, Insurance, Healthcare, Smarter Cities, Government, and Kitchen-based applications**. Here, "adaptive" means the system can improve or change its response based on data, and "contextual" means it understands the situation before giving an output.

---

1. Aspect-Based Feedback Classification

Customer or user feedback is very important for decision-making. Instead of just knowing whether feedback is *positive* or *negative*, organizations also need to know **which aspect the feedback talks about**.

For example, in a kitchen-based application:

- *"The kitchen was dirty"* → Aspect: **Hygiene**
- *"The waiter was slow"* → Aspect: **Service**
- *"The curry was delicious"* → Aspect: **Taste**
- *"The dining area was not cleaned"* → Aspect: **Cleanliness**

By classifying feedback into such aspects, businesses can focus on solving the exact problem (improving hygiene, service speed, taste quality, or cleanliness). This makes the system **context-aware** rather than just sentiment-based.

---

2. Contextual Recommendation System

Another part of cognitive applications is the ability to **recommend** solutions or actions based on **context**. In our case, a **kitchen recommendation system** suggests food items or recipes depending on the **time of day** (morning, evening, night) and **user preference** (healthy, spicy, kids, diet).

For example:

- *Morning + Healthy* → Oats with milk, Fruit salad.
- *Evening + Kids* → Cheese sandwich, Milkshake.
- *Night + Spicy* → Paneer tikka, Masala dosa.

This makes the system **adaptive (different users get different recommendations)** and **contextual (depends on time and preference)**.

---

3. Why Adaptive and Contextual Systems?

- In **Customer Service** → Helps classify customer complaints and recommend quick solutions.
- In **Insurance** → Can classify claim issues and suggest the next steps.
- In **Healthcare** → Can classify patient feedback and recommend treatment options.
- In **Smarter Cities** → Can analyze complaints (traffic, cleanliness) and recommend actions.
- In **Government** → Can organize citizen feedback and recommend policies or improvements.

Thus, such systems are **practical and useful in real life** where both *understanding feedback* and *recommending actions* are equally important.

Install Required Libraries

```
!pip install -q sklearn
```

Show hidden output

Import Libraries

```
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn import metrics
import random
```

Part A: Aspect-Based Feedback Classification

Prepare Sample Kitchen Reviews Dataset

```
# Each review is tagged with an aspect: Hygiene, Taste, Service, Cleanliness
data = {
    "review": [
        "The kitchen was very clean and hygienic",
        "The utensils were dirty and unhygienic",
        "The food taste was delicious and fresh",
        "The meal was too salty and badly cooked",
        "The staff was very friendly and helpful",
        "The service was extremely slow",
        "The floor of the kitchen was not cleaned",
```

```
        "The waiter delivered food very late",
        "The curry was tasty but a bit oily",
        "The kitchen smelled bad and was not maintained",
        "The food was spicy but tasty",
        "I loved the freshness of the vegetables",
        "The plates were not washed properly",
        "The waiter forgot my order",
        "The service was quick and efficient",
        "The soup was bland and tasteless",
        "The kitchen environment was very hygienic",
        "The staff behaved rudely",
        "The rice was cooked perfectly",
        "The dining area was not cleaned"
    ],
    "aspect": [
        "Hygiene",
        "Hygiene",
        "Taste",
        "Taste",
        "Service",
        "Service",
        "Cleanliness",
        "Service",
        "Taste",
        "Cleanliness",
        "Taste",
        "Taste",
        "Hygiene",
        "Service",
        "Service",
        "Taste",
        "Hygiene",
        "Service",
        "Taste",
        "Cleanliness"
    ]
}

df = pd.DataFrame(data)
print(df.head())
```

```
                                    review   aspect
0  The kitchen was very clean and hygienic  Hygiene
1   The utensils were dirty and unhygienic  Hygiene
2   The food taste was delicious and fresh    Taste
3  The meal was too salty and badly cooked    Taste
4  The staff was very friendly and helpful  Service
```

Convert Text Reviews into Features

```
# Convert text into numerical features using Bag of Words
cv = CountVectorizer(stop_words='english')
X = cv.fit_transform(df['review'])
y = df['aspect']
```

Train-Test Split

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=40)
```

Train Naive Bayes Classifier

```
model = MultinomialNB()
model.fit(X_train, y_train)
```

```
MultinomialNB()
```

Evaluate Model

```
y_pred = model.predict(X_test)
print("Predictions:", y_pred)
print("Actual:", y_test.values)
print("Accuracy:", metrics.accuracy_score(y_test, y_pred))
```

```
Predictions: ['Hygiene' 'Taste' 'Service' 'Service']
Actual: ['Cleanliness' 'Cleanliness' 'Service' 'Service']
Accuracy: 0.5
```

Test Model with New Feedback

```
new_feedback = [
    "The food was very tasty",
    "The kitchen was dirty",
    "The waiter was polite but service was slow",
    "The utensils were shining clean"
]

X_new = cv.transform(new_feedback)
predictions = model.predict(X_new)

for text, pred in zip(new_feedback, predictions):
    print(f"Feedback: {text} --> Aspect Predicted: {pred}")
```

```
Feedback: The food was very tasty --> Aspect Predicted: Taste
Feedback: The kitchen was dirty --> Aspect Predicted: Hygiene
Feedback: The waiter was polite but service was slow --> Aspect Predicted: Service
Feedback: The utensils were shining clean --> Aspect Predicted: Hygiene
```

Part B: Contextual Recommendation System

Define Recommendation Knowledge Base

```
recommendations = {
    "morning": ["Oats with milk", "Idli with chutney", "Poha", "Paratha", "Fruit salad"],
    "afternoon": ["Dal with rice", "Paneer curry", "Vegetable biryani", "Curd rice"],
    "evening": ["Tea with biscuits", "Samosa", "Sandwich", "Pakora"],
    "night": ["Roti with sabji", "Khichdi", "Soup", "Grilled vegetables"],

    "healthy": ["Sprouts salad", "Green smoothie", "Steamed vegetables", "Ragi dosa"],
    "spicy": ["Chole bhature", "Spicy noodles", "Paneer tikka", "Masala dosa"],
    "kids": ["Mango milkshake", "Cheese sandwich", "French fries", "Pasta"],
    "diet": ["Boiled eggs", "Chicken salad", "Low-oil upma", "Brown rice with dal"]
}
```

Define Recommendation Function

```
def recommend_food(time=None, preference=None):
    results = []

    if time and time in recommendations:
        results.extend(recommendations[time])
    if preference and preference in recommendations:
        results.extend(recommendations[preference])

    if not results:
        return "No recommendation found for given context."

    return random.choice(results)
```

Generate Recommendations

```
print("Morning Recommendation:", recommend_food(time="morning"))
```

```
Morning Recommendation: Paratha
```

```
print("Healthy Morning Recommendation:", recommend_food(time="morning", preference="healthy"))
```

```
Healthy Morning Recommendation: Idli with chutney
```

```
print("Kids Evening Recommendation:", recommend_food(time="evening", preference="kids"))
```

```
Kids Evening Recommendation: Cheese sandwich
```

```
print("Spicy Night Recommendation:", recommend_food(time="night", preference="spicy"))
```

```
Spicy Night Recommendation: Khichdi
```

Conclusion

In this experiment, we studied how cognitive applications can be made more **powerful** by combining two approaches:

1. **Aspect-Based Feedback Classification** → to understand the *context* of user reviews.
2. **Contextual Recommendation System** → to provide *adaptive suggestions* based on situation and preference.

This shows how cognitive systems can move beyond basic tasks to become **intelligent, adaptive, and context-aware tools**, useful across domains like kitchens, customer service, healthcare, and government services.
