Name: **Harsh Pramod Padyal**

Roll No : **40**

Div: **D20B**

Experiment 05

AIM: To build a Cognitive Analytics for personalization of Customer service application/ Insurance/ Healthcare Application/ Smarter Cities/ Government etc.

Theory

Introduction

Cognitive analytics refers to the use of AI techniques like natural language processing (NLP), sentiment analysis, and machine learning to understand data in a human-like way and provide actionable insights. Unlike traditional analytics that only shows trends, cognitive analytics focuses on context, meaning, and personalization.

In this experiment, the goal is to use **cognitive analytics for personalization**. Personalization means giving tailored responses, services, or recommendations to individual users based on their feedback, preferences, or behavior. This is widely used in customer service systems, insurance claim handling, healthcare advice, and even in smart city services where citizen feedback is analyzed for improvement.

Concepts Involved

1. **Sentiment Analysis**
    - Helps in identifying the **emotions behind feedback** (Positive, Negative, Neutral).
    - Multiple approaches can be used:
        - **Lexicon-based methods** like **VADER** that use dictionaries of positive/negative words.
        - **Polarity and Subjectivity** scores using **TextBlob**.
        - **Deep learning models (Transformers)** like BERT that provide more accurate results.
2. **Analytics for Insights**
    - Once sentiments are identified, they can be analyzed using graphs, charts, and distributions.
    - For example: percentage of positive vs negative reviews, or most common complaints.
    - Visualization techniques (bar charts, heatmaps, word clouds) help in **understanding customer priorities**.

3. **Topic Detection**
    - Beyond general sentiment, cognitive systems must know *what the feedback is about*.
    - This is achieved through:
        - **Rule-based keyword matching** (e.g., words like “dirty” → Hygiene).
        - **Zero-shot classification using Transformers**, which can classify topics without needing training.
    - In a kitchen feedback scenario, topics may include **Hygiene, Taste, Service, Cleanliness, Price, Ambience**.
4. **Personalization**
    - Based on both **sentiment** and **topic**, the system provides **tailored actions**.
    - Example:
        - If feedback is **Negative + Hygiene** → “Increase cleaning checks, sanitize utensils.”
        - If feedback is **Positive + Taste** → “Introduce new recipes and keep the same chef style.”
    - This shows how analytics can **adapt to individual customer needs**.
5. **Summary & Decision Support**
    - The final step is to combine all insights into a **summary report**.
    - This helps managers or decision makers to see:
        - Common complaints.
        - Overall satisfaction level.
        - Personalized recommendations for future improvement.

Relevance of This Experiment

- **Customer Service** → Improves satisfaction by acting on personalized suggestions.
- **Insurance** → Speeds up claim handling by understanding the sentiment of customers.
- **Healthcare** → Provides patient-specific advice and identifies negative experiences.
- **Smarter Cities** → Analyzes citizen complaints and prioritizes key improvement areas.
- **Government** → Helps in policymaking by analyzing feedback from the public.

Install Libraries

```
# Core NLP + ML + Viz
!pip install -q nltk textblob vaderSentiment transformers wordcloud scikit-learn matplotlib seaborn
```

```
                                             126.0/126.0 kB 2.1 MB/s eta 0:00:00
```

Imports & Setup

```
import re
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from textblob import TextBlob
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
from transformers import pipeline

from wordcloud import WordCloud, STOPWORDS
import nltk
nltk.download('vader_lexicon')

# Plot settings
plt.rcParams["figure.figsize"] = (7,4)
sns.set_theme(style="whitegrid")
```

```
[nltk_data] Downloading package vader_lexicon to /root/nltk_data...
```

Load Data (Use Sample)

```
from google.colab import files

reviews = [
        "The kitchen was very clean and well organized.",
        "The utensils were dirty and unhygienic.",
        "Food was delicious and fresh!",
        "The service was too slow and disappointing.",
        "The curry was spicy but tasty.",
        "The floor of the kitchen was not cleaned properly.",
        "The staff was polite and friendly.",
        "I am not happy with the hygiene of the kitchen.",
        "The soup was bland and tasteless.",
        "Excellent food quality and hygiene maintained.",
        "The waiter forgot my order and served very late.",
        "Great taste and balanced spices.",
        "Dining area needs better cleaning schedule.",
]
df = pd.DataFrame({"review": reviews})

df = df.dropna(subset=["review"]).reset_index(drop=True)
df.head()
```

	review
0	The kitchen was very clean and well organized.
1	The utensils were dirty and unhygienic.
2	Food was delicious and fresh!
3	The service was too slow and disappointing.
4	The curry was spicy but tasty.

Next steps: Generate code with df | View recommended plots | New interactive sheet

Basic Cleaning (Lowercase, Remove Extra Symbols)

```
def clean_text(text: str) -> str:
    text = text.lower()
    text = re.sub(r"http\S+|www\S+|https\S+", " ", text)
    text = re.sub(r"[^a-zA-Z0-9\s]", " ", text)  # keep alphanumerics and space
    text = re.sub(r"\s+", " ", text).strip()
    return text

df["clean_review"] = df["review"].astype(str).apply(clean_text)
df.head(10)
```

	review	clean_review
0	The kitchen was very clean and well organized.	the kitchen was very clean and well organized
1	The utensils were dirty and unhygienic.	the utensils were dirty and unhygienic
2	Food was delicious and fresh!	food was delicious and fresh
3	The service was too slow and disappointing.	the service was too slow and disappointing
4	The curry was spicy but tasty.	the curry was spicy but tasty
5	The floor of the kitchen was not cleaned prope...	the floor of the kitchen was not cleaned properly
6	The staff was polite and friendly.	the staff was polite and friendly
7	I am not happy with the hygiene of the kitchen.	i am not happy with the hygiene of the kitchen
8	The soup was bland and tasteless.	the soup was bland and tasteless
9	Excellent food quality and hygiene maintained.	excellent food quality and hygiene maintained

Next steps: Generate code with df | View recommended plots | New interactive sheet

Sentiment with VADER

```
analyzer = SentimentIntensityAnalyzer()

def vader_sentiment_label(text):
    score = analyzer.polarity_scores(text)["compound"]
    if score > 0.05:
        return "Positive"
    elif score < -0.05:
        return "Negative"
    else:
        return "Neutral"

df["vader_score"] = df["clean_review"].apply(lambda t: analyzer.polarity_scores(t)["compound"])
df["vader_label"] = df["clean_review"].apply(vader_sentiment_label)

df[["review","vader_score","vader_label"]].head(10)
```

	review	vader_score	vader_label
0	The kitchen was very clean and well organized.	0.6549	Positive
1	The utensils were dirty and unhygienic.	-0.4404	Negative
2	Food was delicious and fresh!	0.7184	Positive
3	The service was too slow and disappointing.	-0.4939	Negative
4	The curry was spicy but tasty.	0.0000	Neutral
5	The floor of the kitchen was not cleaned prope...	0.0000	Neutral
6	The staff was polite and friendly.	0.4939	Positive
7	I am not happy with the hygiene of the kitchen.	-0.4585	Negative
8	The soup was bland and tasteless.	0.0000	Neutral
9	Excellent food quality and hygiene maintained.	0.5719	Positive

Sentiment with TextBlob (Polarity & Subjectivity)

```
df["tb_polarity"] = df["clean_review"].apply(lambda t: TextBlob(t).sentiment.polarity)
df["tb_subjectivity"] = df["clean_review"].apply(lambda t: TextBlob(t).sentiment.subjectivity)

def tb_label(p):
    return "Positive" if p > 0.05 else ("Negative" if p < -0.05 else "Neutral")

df["tb_label"] = df["tb_polarity"].apply(tb_label)
df[["review","tb_polarity","tb_subjectivity","tb_label"]].head(10)
```

	review	tb_polarity	tb_subjectivity	tb_label
0	The kitchen was very clean and well organized.	0.476667	0.910000	Positive
1	The utensils were dirty and unhygienic.	-0.600000	0.800000	Negative
2	Food was delicious and fresh!	0.650000	0.750000	Positive
3	The service was too slow and disappointing.	-0.450000	0.550000	Negative
4	The curry was spicy but tasty.	0.000000	0.000000	Neutral
5	The floor of the kitchen was not cleaned prope...	0.000000	0.100000	Neutral
6	The staff was polite and friendly.	0.375000	0.500000	Positive
7	I am not happy with the hygiene of the kitchen.	-0.400000	1.000000	Negative
8	The soup was bland and tasteless.	-0.383333	0.866667	Negative
9	Excellent food quality and hygiene maintained.	1.000000	1.000000	Positive

Transformer Sentiment (HuggingFace Pipeline)

```
hf_sa = pipeline("sentiment-analysis")  # default SST-2 style pipeline

hf_results = hf_sa(df["review"].tolist())
df["hf_label"] = [r["label"] for r in hf_results]
df["hf_score"] = [r["score"] for r in hf_results]

# Normalize labels to Positive/Negative/Neutral (if model returns only POS/NEG, keep as is)
df["hf_label_norm"] = df["hf_label"].replace({"POSITIVE":"Positive","NEGATIVE":"Negative"})

df[["review","hf_label_norm","hf_score"]].head(10)
```

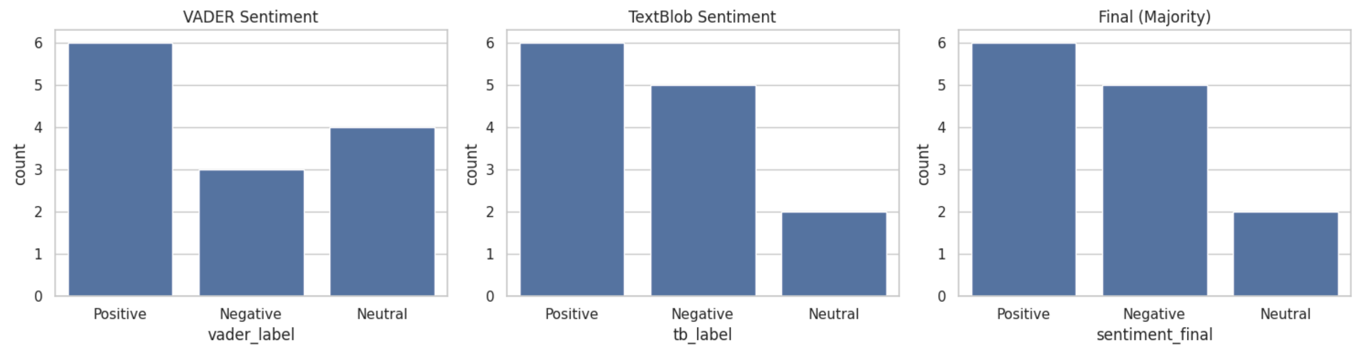
```
No model was supplied, defaulted to distilbert/distilbert-base-uncased-finetuned-sst-2-english
Using a pipeline without specifying a model name and revision in production is not recommended.
/usr/local/lib/python3.12/dist-packages/huggingface_hub/utils/_auth.py:94: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://hugging
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or da
  warnings.warn(
config.json: 100%                                          629/629 [00:00<00:00, 57.3kB/s]

model.safetensors: 100%                                     268M/268M [00:03<00:00, 96.0MB/s]

tokenizer_config.json: 100%                                 48.0/48.0 [00:00<00:00, 5.47kB/s]

vocab.txt:      232k/? [00:00<00:00, 1.53MB/s]
Device set to use cpu
```

	review	hf_label_norm	hf_score
0	The kitchen was very clean and well organized.	Positive	0.999780
1	The utensils were dirty and unhygienic.	Negative	0.999546
2	Food was delicious and fresh!	Positive	0.999887
3	The service was too slow and disappointing.	Negative	0.999705
4	The curry was spicy but tasty.	Positive	0.996216
5	The floor of the kitchen was not cleaned prope...	Negative	0.999787
6	The staff was polite and friendly.	Positive	0.999821
7	I am not happy with the hygiene of the kitchen.	Negative	0.999608
8	The soup was bland and tasteless.	Negative	0.999653
9	Excellent food quality and hygiene maintained.	Positive	0.999837

Compare Methods (Agreement & Distribution)

```
# Majority vote sentiment (simple)
def majority_vote(row):
    labels = [row["vader_label"], row["tb_label"], row["hf_label_norm"]]
    return pd.Series(labels).mode().iloc[0]

df["sentiment_final"] = df.apply(majority_vote, axis=1)

# Distribution plots
fig, axes = plt.subplots(1, 3, figsize=(15,4))
sns.countplot(x="vader_label", data=df, ax=axes[0])
axes[0].set_title("VADER Sentiment")
sns.countplot(x="tb_label", data=df, ax=axes[1])
axes[1].set_title("TextBlob Sentiment")
sns.countplot(x="sentiment_final", data=df, ax=axes[2])
axes[2].set_title("Final (Majority)")
```

```
plt.tight_layout()
plt.show()
```



```
df[["review","vader_label","tb_label","hf_label_norm","sentiment_final"]].head(10)
```

	review	vader_label	tb_label	hf_label_norm	sentiment_final
0	The kitchen was very clean and well organized.	Positive	Positive	Positive	Positive
1	The utensils were dirty and unhygienic.	Negative	Negative	Negative	Negative
2	Food was delicious and fresh!	Positive	Positive	Positive	Positive
3	The service was too slow and disappointing.	Negative	Negative	Negative	Negative
4	The curry was spicy but tasty.	Neutral	Neutral	Positive	Neutral
5	The floor of the kitchen was not cleaned prope...	Neutral	Neutral	Negative	Neutral
6	The staff was polite and friendly.	Positive	Positive	Positive	Positive
7	I am not happy with the hygiene of the kitchen	Negative	Negative	Negative	Negative

Topic Detection (Rule-based Keywords)

```
# Kitchen-domain topics
TOPIC_KEYWORDS = {
    "Hygiene": ["hygiene","hygienic","dirty","unclean","clean","washed","sanitized","smell","odor",
    "Service": ["service","waiter","late","slow","quick","staff","rude","friendly","polite","forgot
    "Taste":   ["taste","tasty","delicious","bland","spicy","salty","sweet","oily","flavor","flavou
    "Cleanliness": ["cleaning","cleaned","floor","dining","maintained","messy","stain","wipe"]
}

def detect_topic(text):
    text = text.lower()
    scores = {k:0 for k in TOPIC_KEYWORDS}
    for topic, kws in TOPIC_KEYWORDS.items():
        for kw in kws:
            if kw in text:
                scores[topic] += 1
    # pick the topic with highest score; if all zero -> "General"
    best_topic = max(scores, key=scores.get)
    return best_topic if scores[best_topic] > 0 else "General"

df["topic_rule"] = df["clean_review"].apply(detect_topic)
df[["review","topic_rule"]].head(10)
```

	review	topic_rule
0	The kitchen was very clean and well organized.	Hygiene
1	The utensils were dirty and unhygienic.	Hygiene
2	Food was delicious and fresh!	Taste
3	The service was too slow and disappointing.	Service
4	The curry was spicy but tasty.	Taste
5	The floor of the kitchen was not cleaned prope...	Cleanliness
6	The staff was polite and friendly.	Service
7	I am not happy with the hygiene of the kitchen.	Hygiene
8	The soup was bland and tasteless.	Taste
9	Excellent food quality and hygiene maintained.	Hygiene

Topic Detection (Zero-Shot Classification)

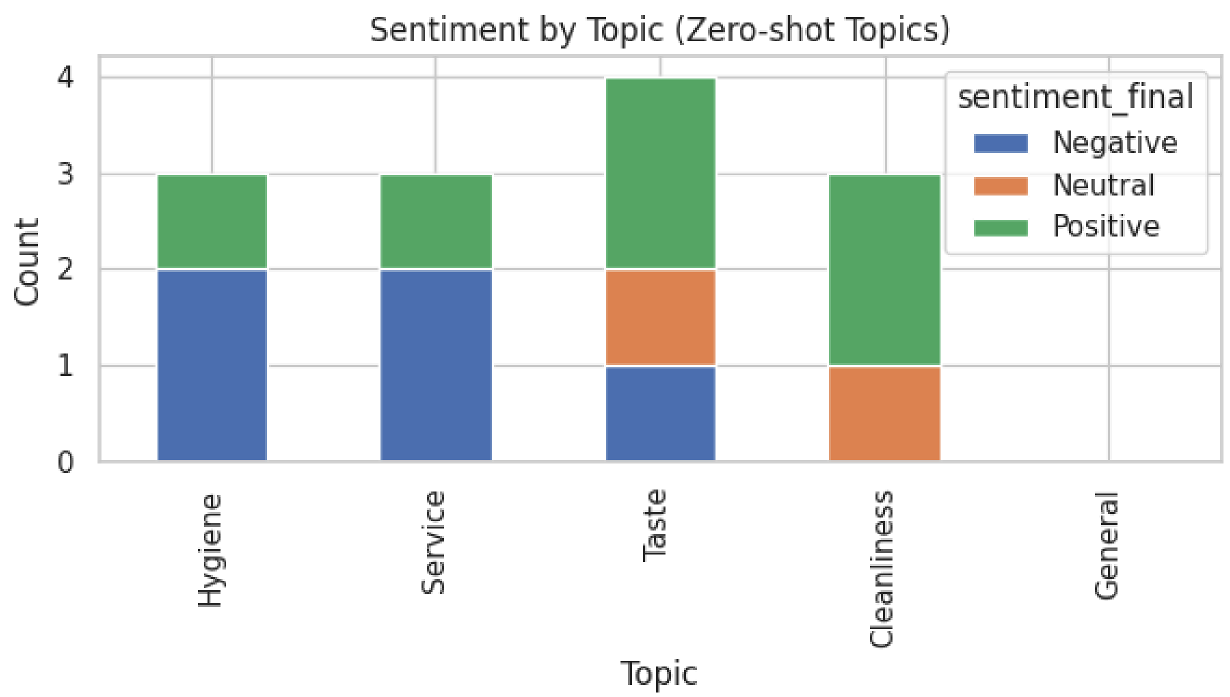
```
# Zero-shot classification with candidate labels = topics
candidate_labels = ["Hygiene", "Service", "Taste", "Cleanliness", "General"]
zsc = pipeline("zero-shot-classification", model="facebook/bart-large-mnli")

zsc_results = zsc(df["review"].tolist(), candidate_labels=candidate_labels, multi_label=False)
df["topic_zsc"] = [r["labels"][0] for r in zsc_results]

df[["review","topic_rule","topic_zsc"]].head(10)
```

```
config.json:          1.15k/?  [00:00<00:00,  123kB/s]
model.safetensors:  100%                                        1.63G/1.63G  [00:25<00:00,  51.2MB/s]
tokenizer_config.json:  100%                                     26.0/26.0  [00:00<00:00,  2.29kB/s]
vocab.json:          899k/?  [00:00<00:00,  20.7MB/s]
merges.txt:          456k/?  [00:00<00:00,  19.5MB/s]
tokenizer.json:          1.36M/?  [00:00<00:00,  52.9MB/s]
Device set to use cpu
```

	review	topic_rule	topic_zsc
0	The kitchen was very clean and well organized.	Hygiene	Cleanliness
1	The utensils were dirty and unhygienic.	Hygiene	Hygiene
2	Food was delicious and fresh!	Taste	Taste
3	The service was too slow and disappointing.	Service	Service
4	The curry was spicy but tasty.	Taste	Taste
5	The floor of the kitchen was not cleaned prope...	Cleanliness	Cleanliness
6	The staff was polite and friendly.	Service	Service
7	I am not happy with the hygiene of the kitchen.	Hygiene	Hygiene
8	The soup was bland and tasteless.	Taste	Taste
9	Excellent food quality and hygiene maintained.	Hygiene	Hygiene

Analytics: Sentiment by Topic

```
# Use final sentiment and zero-shot topic for analytics
pivot = pd.crosstab(df["topic_zsc"], df["sentiment_final"]).reindex(candidate_labels, fill_value=0)

pivot.plot(kind="bar", stacked=True)
plt.title("Sentiment by Topic (Zero-shot Topics)")
plt.xlabel("Topic")
plt.ylabel("Count")
plt.tight_layout()
plt.show()

print("Counts by Topic (Zero-shot):")
display(df["topic_zsc"].value_counts())
```



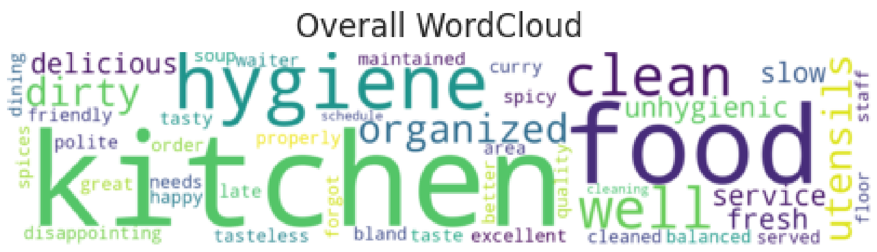
```
Counts by Topic (Zero-shot):
```

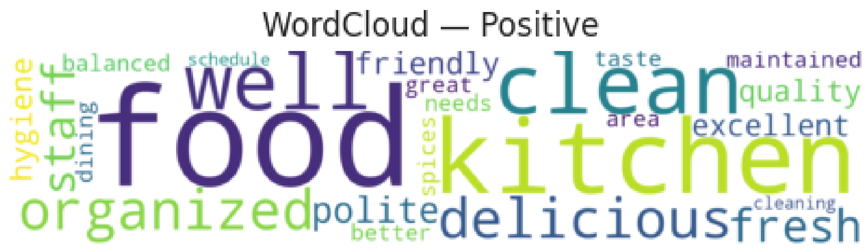
topic_zsc	count
Taste	4
Cleanliness	3
Hygiene	3
Service	3

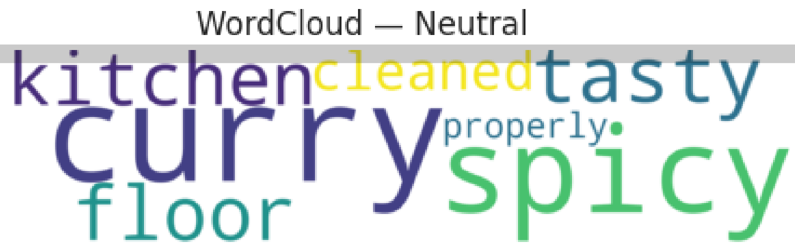
dtype: int64

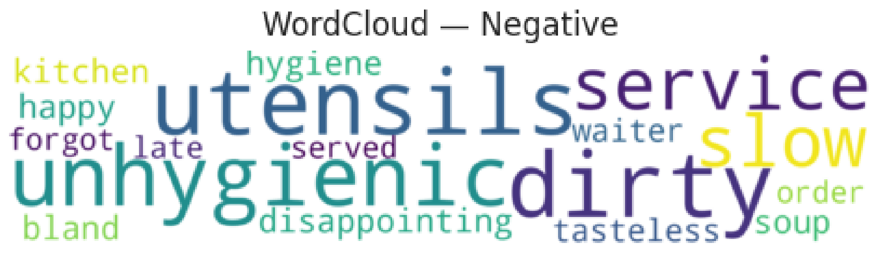
WordClouds (Overall & By Sentiment)

```
def make_wordcloud(texts, title="WordCloud"):
    if len(texts) == 0:
        print(f"No texts for {title}")
        return
    wc_text = " ".join(texts)
    wc = WordCloud(width=1800, height=400, stopwords=STOPWORDS, background_color="white").generate(
    plt.figure(figsize=(6,3))
    plt.imshow(wc, interpolation="bilinear")
    plt.axis("off")
    plt.title(title)
    plt.show()

# Overall
make_wordcloud(df["clean_review"].tolist(), title="Overall WordCloud")

# By sentiment
for s in ["Positive","Neutral","Negative"]:
    subset = df[df["sentiment_final"]==s]["clean_review"].tolist()
    make_wordcloud(subset, title=f"WordCloud — {s}")
```









Personalization Rules (Actions & Recommendations)

```
# Simple action rules combining topic + sentiment
def personalized_action(topic, sentiment):
    if topic == "Hygiene":
        return "Increase sanitization checks, ensure utensils/storage are cleaned frequently."
    if topic == "Cleanliness":
        return "Tighten cleaning schedule for kitchen & dining area; add closing-time audit."
    if topic == "Service":
        return "Train staff for faster response, add order tracking, set service-time SLAs."
    if topic == "Taste":
        return "Refine recipes; offer spice-level options; collect taste preference at ordering."
    # General fallback
    return "Collect more detailed feedback; offer quick survey for preferences."

def sentiment_tone(sentiment):
    if sentiment == "Positive":
        return "Maintain quality; consider loyalty rewards or upsell add-ons."
    if sentiment == "Neutral":
        return "Encourage detailed feedback; provide small incentives to share preferences."
    if sentiment == "Negative":
        return "Immediate remediation; acknowledge issue and provide make-good coupon."
    return "Monitor feedback trend."

df["action_topic"] = df["topic_zsc"].apply(lambda t: personalized_action(t, None))
df["action_sentiment"] = df["sentiment_final"].apply(sentiment_tone)

df[["review","topic_zsc","sentiment_final","action_topic","action_sentiment"]].head(10)
```

	review	topic_zsc	sentiment_final	action_topic	action_sentiment
0	The kitchen was very clean and well organized.	Cleanliness	Positive	Tighten cleaning schedule for kitchen & dining...	Maintain quality; consider loyalty rewards or ...
1	The utensils were dirty and unhygienic.	Hygiene	Negative	Increase sanitization checks, ensure utensils/...	Immediate remediation; acknowledge issue and p...
2	Food was delicious and fresh!	Taste	Positive	Refine recipes; offer spice-level options; col...	Maintain quality; consider loyalty rewards or ...
3	The service was too slow and disappointing.	Service	Negative	Train staff for faster response, add order tra...	Immediate remediation; acknowledge issue and p...
4	The curry was spicy but tasty.	Taste	Neutral	Refine recipes; offer spice-level options; col...	Encourage detailed feedback; provide small inc...

Simple Personalization Engine (Per User Profile)

```
# Example: create a tiny user profile table (simulate)
# Columns: user_id, last_preference (spicy/healthy/kids), typical_time (morning/evening)
profiles = pd.DataFrame({
    "user_id":[101,102,103,104],
    "last_preference":["spicy","healthy","kids","healthy"],
    "typical_time":["evening","morning","evening","night"]
})

# Contextual menu knowledge base (from Exp 04 idea)
MENU = {
    "morning": ["Oats with milk","Idli","Poha","Paratha","Fruit salad"],
    "evening": ["Tea & biscuits","Samosa","Sandwich","Pakora"],
    "night": ["Roti & sabji","Khichdi","Soup","Grilled vegetables"],
    "healthy": ["Sprouts salad","Green smoothie","Steamed vegetables","Ragi dosa"],
    "spicy": ["Chole bhature","Spicy noodles","Paneer tikka","Masala dosa"],
    "kids": ["Milkshake","Cheese sandwich","French fries","Pasta"]
}

import random
def contextual_reco(time=None, pref=None):
    opts = []
    if time in MENU: opts += MENU[time]
    if pref in MENU: opts += MENU[pref]
    return random.choice(opts) if opts else "Chef's Special of the day"

# Generate a personalized suggestion per user using their profile + global sentiment trend
trend = df["sentiment_final"].value_counts(normalize=True).to_dict()
overall_mood = max(trend, key=trend.get) if len(trend)>0 else "Neutral"

print("Overall Sentiment Mood (majority):", overall_mood)
print()

for _, row in profiles.iterrows():
    suggestion = contextual_reco(time=row["typical_time"], pref=row["last_preference"])
```

```
Overall Sentiment Mood (majority): Positive

User 101 → Recommendation: Samosa (time=evening, pref=spicy)
User 102 → Recommendation: Green smoothie (time=morning, pref=healthy)
User 103 → Recommendation: Pasta (time=evening, pref=kids)
User 104 → Recommendation: Sprouts salad (time=night, pref=healthy)
```

Conclusion

In this experiment, we used cognitive analytics to study customer feedback and provide personalized suggestions. By combining sentiment analysis, topic detection, and visual analytics, the system was able to understand both emotions and key issues in the feedback. This helps in improving services by taking the right actions for each type of customer response.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.