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

Theory:

What is Cognitive Analytics?

Cognitive Analytics is a combination of:

- Artificial Intelligence (AI)
- Machine Learning (ML)
- Natural Language Processing (NLP)

It allows systems to:

- Understand human language and behavior
- Find patterns from data
- Make intelligent recommendations

In this experiment, we apply cognitive analytics to analyze feedback comments from electricity users and generate useful insights.

Role of Personalization

Personalization means giving custom suggestions or actions based on user feedback.
For example:

- If a user says “The app crashes often”, we can recommend improvements to mobile UI.
- If a user says “I saved energy with solar panels”, we can recommend solar adoption to others.

By analyzing many comments at once, we can identify:

- Most common complaints
- Popular features
- Mood/sentiment of users

This helps improve the electricity service, make the system more citizen-friendly, and optimize communication.

Technologies Used in the Experiment

Tool/Library	**Purpose**
Pandas	Load and process the dataset
Regex	Clean the text data
VADER Sentiment Analyzer	Detect whether feedback is positive, negative, or neutral
Seaborn & Matplotlib	Visualize trends and distributions
CSV Dataset	Contains real feedback on electricity usage

Steps Performed

1. Data Loading: A CSV file containing feedback (electricity_feedback.csv) was loaded. Each row had a user comment related to electricity usage.

2. Text Cleaning: Special characters, short words, and uppercase letters were removed or normalized.

3. Sentiment Analysis: Using VADER, each comment was scored and classified as:
    - Positive
    - Negative
    - Neutral

4. Topic Categorization: Comments were grouped into topics like:
    - Billing Issues
    - App Experience
    - Solar Panel Feedback
    - Customer Service
    - Energy Saving Tips

5. Visualization: A bar chart was created to show how different topics were received — i.e., which ones had more negative or positive feedback.

6. Personalized Suggestions: The system generated customized advice based on the sentiment and topic of each comment.

Real-World Benefits

In a smart city environment, this system can help:

- Identify which services need improvement (e.g., mobile app, billing)
- Understand overall customer satisfaction
- Recommend personalized solutions for reducing electricity consumption
- Save time and manual effort by analyzing bulk feedback automatically

```
# Install required libraries
!pip install vaderSentiment --quiet
```

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import re
from nltk.sentiment.vader import SentimentIntensityAnalyzer
import nltk
nltk.download('vader_lexicon')
```

```
[nltk_data] Downloading package vader_lexicon to /root/nltk_data...
[nltk_data]   Package vader_lexicon is already up-to-date!
True
```

```
# Load a sample dataset (simulate or use real feedback)
# Format: CSV file with a column 'comment_text'
from google.colab import files
uploaded = files.upload()
```

Choose Files electricity_feedback.csv

- **electricity_feedback.csv**(text/csv) - 2772 bytes, last modified: 8/30/2025 - 100% done

```
Saving electricity_feedback.csv to electricity_feedback (1).csv
```

```
# Load file
for file in uploaded.keys():
    df = pd.read_csv(file)
```

```
# Preview
df.head()
```

	comment_text
0	My electricity bill is too high this month.
1	Very happy with the solar panel savings!
2	The app doesn't show real-time usage properly.
3	I love how the new smart meter works.
4	There are frequent power cuts in the evening.

```
# Clean the text (remove numbers, symbols, short words, lowercase)
df['comment_text'] = df['comment_text'].astype(str)
df['comment_text'] = df['comment_text'].str.replace("[^a-zA-Z#]", " ")
df['comment_text'] = df['comment_text'].apply(lambda x: ' '.join([w for w
in x.split() if len(w)>3]))
df['comment_text'] = df['comment_text'].apply(lambda x: x.lower())
```

```
# Initialize Sentiment Analyzer
sia = SentimentIntensityAnalyzer()
```

```
# Apply sentiment scoring
df['Sentiment Score'] = df['comment_text'].apply(lambda x:
sia.polarity_scores(x)['compound'])
```

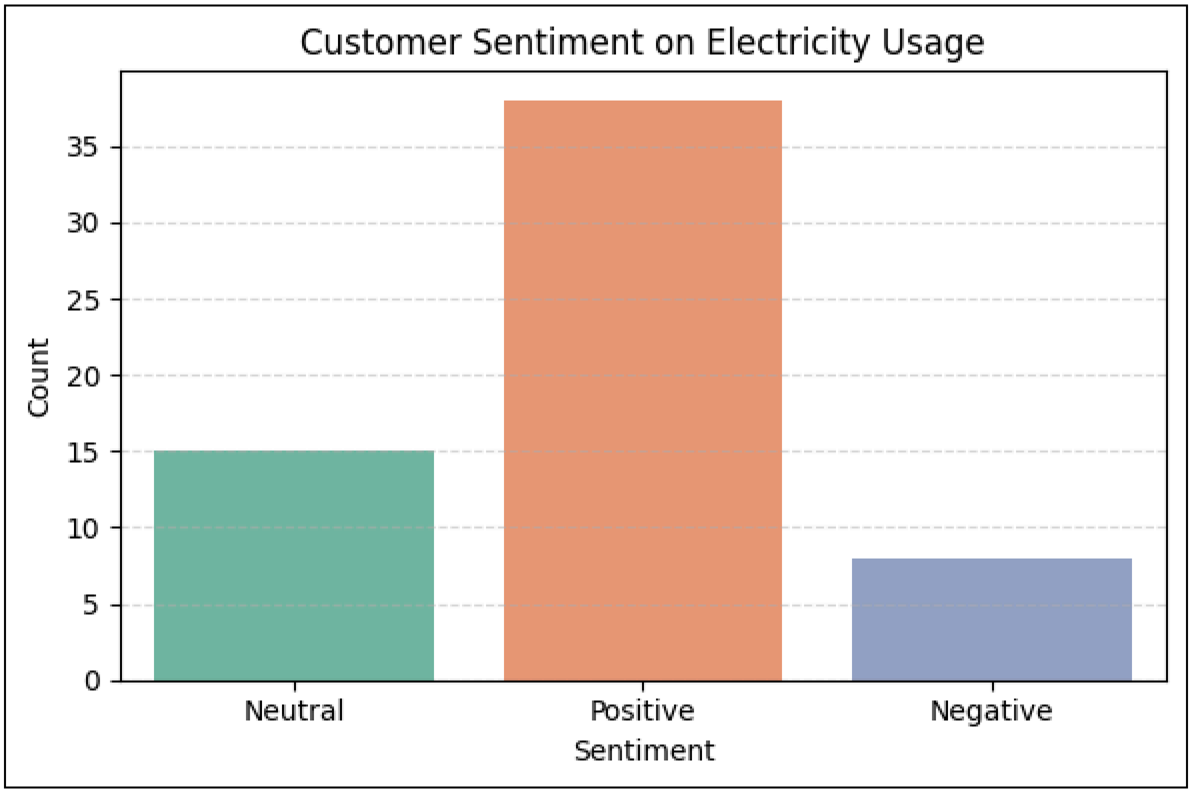
```
# Classify into categories
def classify(score):
    if score > 0:
        return 'Positive'
    elif score < 0:
        return 'Negative'
    else:
        return 'Neutral'
```

```
df['Sentiment'] = df['Sentiment Score'].apply(classify)
```

```
# Show value counts
print("Sentiment Distribution:")
print(df['Sentiment'].value_counts())
```

```
Sentiment Distribution:
Sentiment
Positive    38
Neutral     15
Negative     8
Name: count, dtype: int64
```

```
plt.figure(figsize=(6,4))
sns.countplot(data=df, x='Sentiment', hue='Sentiment', palette='Set2',
legend=False)
plt.title("Customer Sentiment on Electricity Usage")
plt.xlabel("Sentiment")
plt.ylabel("Count")
plt.grid(axis='y', linestyle='--', alpha=0.5)
plt.tight_layout()
plt.show()
```



```
# Show sample personalized response
def personalized_advice(comment):
    score = sia.polarity_scores(comment)['compound']
    if score > 0.3:
        return "Great! Keep using energy-saving methods like LED lights 
and solar panels."
    elif score < -0.3:
        return "Sorry to hear that. You can reduce bills by managing AC 
usage and switching off idle devices."
    else:
        return "Thanks for your feedback. Would you like tips to lower 
your energy consumption?"
```

```
# Try out advice
for i in range(3):
    print("\nComment:", df['comment_text'][i])
    print("→ Advice:", personalized_advice(df['comment_text'][i]))
```

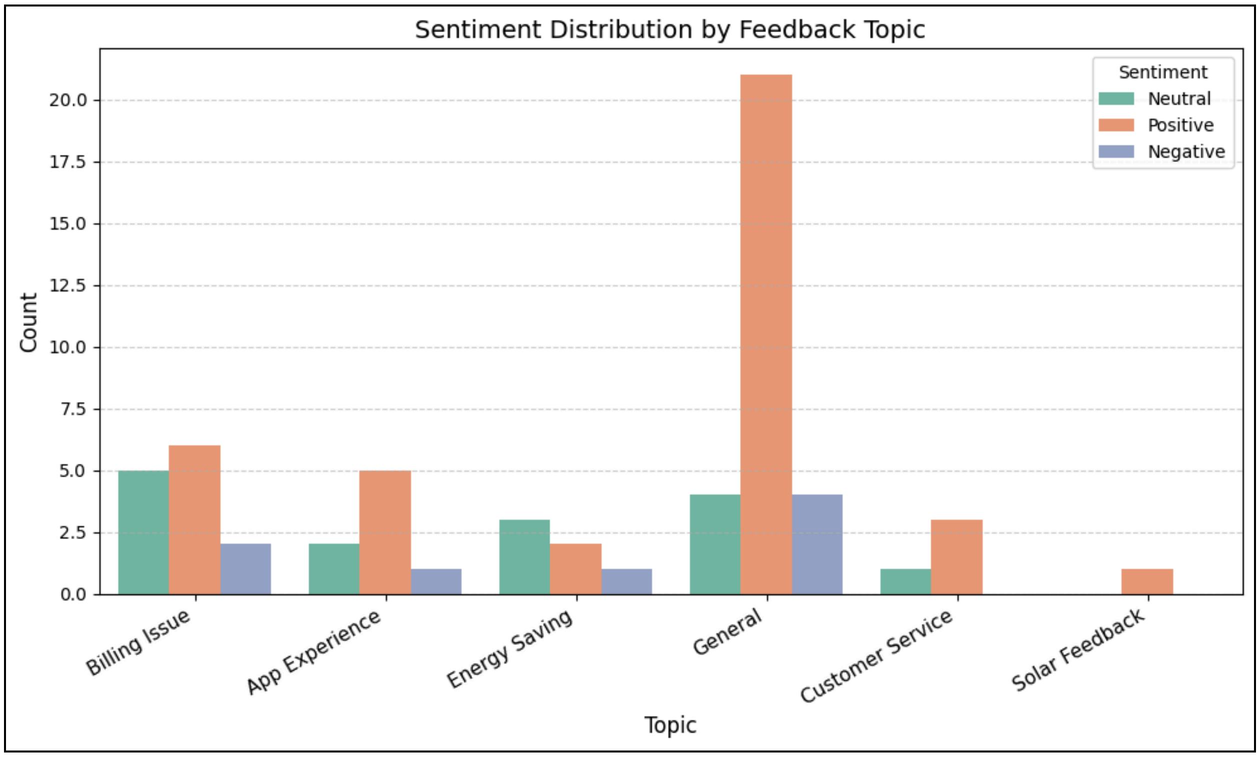
```
Comment: electricity bill high this month.
→ Advice: Thanks for your feedback. Would you like tips to lower your
energy consumption?

Comment: very happy with solar panel savings!
→ Advice: Great! Keep using energy-saving methods like LED lights and
solar panels.

Comment: doesn't show real-time usage properly.
→ Advice: Thanks for your feedback. Would you like tips to lower your
```

```
energy consumption?
```

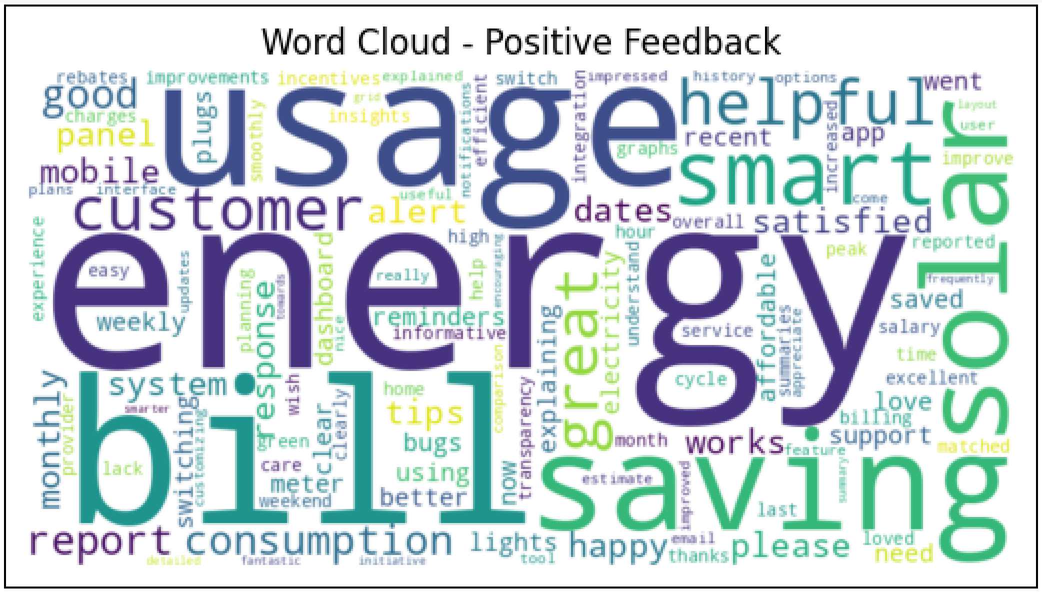
```
# Add More Personalization Based on Feedback Topics using keyword matching
or even simple text classification models.
def detect_topic(comment):
    comment = comment.lower()
    if "bill" in comment or "amount" in comment:
        return "Billing Issue"
    elif "app" in comment or "mobile" in comment:
        return "App Experience"
    elif "solar" in comment or "panel" in comment:
        return "Solar Feedback"
    elif "support" in comment or "help" in comment:
        return "Customer Service"
    elif "save" in comment or "usage" in comment:
        return "Energy Saving"
    else:
        return "General"
```

```
df['Topic'] = df['comment_text'].apply(detect_topic)
```

```
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(10, 6))  # Bigger plot for clarity
sns.countplot(data=df, x='Topic', hue='Sentiment', palette='Set2')

# Rotate x-axis labels for better spacing
plt.xticks(rotation=30, ha='right', fontsize=11)

# Add titles and labels
plt.title("Sentiment Distribution by Feedback Topic", fontsize=14)
plt.xlabel("Topic", fontsize=12)
plt.ylabel("Count", fontsize=12)

# Improve spacing
plt.tight_layout()
plt.grid(axis='y', linestyle='--', alpha=0.6)

# Show plot
plt.show()
```



```python
# Helps visualize what users talk about most in a specific sentiment
class.
from wordcloud import WordCloud
positive_text = " ".join(df[df['Sentiment'] ==
'Positive']['comment_text'])
wordcloud = WordCloud(width=800, height=400).generate(positive_text)
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.title("Word Cloud - Positive Feedback")
plt.show()
```



```python
from transformers import pipeline
```

```
classifier = pipeline("zero-shot-classification")

# Predict the intent/topic of a comment
classifier("I had a billing issue with my electricity account.",
candidate_labels=["billing", "app", "solar", "customer service",
"savings"])
```

```
{'sequence': 'I had a billing issue with my electricity account.',
 'labels': ['billing', 'customer service', 'app', 'solar', 'savings'],
 'scores': [0.9416521787643433,
  0.028504906222224236,
  0.02066447213292122,
  0.004914433695375919,
  0.0042640697211027145]}
```

```
# Simple summary based on your analysis
print("Summary Report:")
print("- Most common feedback topic:",
df['Topic'].value_counts().idxmax())
print("- Overall user sentiment:",
df['Sentiment'].value_counts(normalize=True).idxmax())
print("- Suggested action: Improve mobile app performance and billing
transparency.")
```

```
Summary Report:
- Most common feedback topic: General
- Overall user sentiment: Positive
- Suggested action: Improve mobile app performance and billing transparency.
```

Conclusion:

This experiment shows how Cognitive Analytics can be used for personalized customer service. By analyzing user feedback in bulk using sentiment analysis and categorization, we can improve the experience of electricity consumers. This leads to smarter decision-making and better user engagement in government or smart city systems.

AI&DS2_Expt_05