UNIVERSITY COLLEGE OF ENGINEERING VILLUPURAM

NAAN MUDHALVAN

IBM-ARTIFICIAL INTELLIGENCE

GOKUL R

422521104010

Measure Energy Consumption

Problem Definition:

The problem at hand is to create an automated system that measures energy consumption, analyzes the data, and provides visualizations for informed decision-making. This solution aims to enhance efficiency, accuracy, and ease of understanding in managing energy consumption across various sectors.

1. Data Source Identification:

- Identify a suitable dataset that contains energy consumption measurements. Some potential sources include government agencies, utility companies, research institutions, or open data portals. You can search for datasets on websites like data.gov, Kaggle, or the U.S. Energy Information Administration (EIA) website.

2. Data Preprocessing:

- Clean the dataset by addressing missing values, outliers, and data inconsistencies.

- Convert data types as needed (e.g., date/time conversion).

- Handle duplicates and remove irrelevant columns.

- Normalize or scale data if necessary.

- Ensure data quality and consistency.

3. Feature Extraction:

- Identify relevant features and metrics that can provide insights into energy consumption. These might include:

- Time-based features (hourly, daily, monthly, seasonal patterns).

- Weather data (temperature, humidity) if available, as it can impact energy usage.

- Demographic data (population, building types) if relevant.

- Calculate aggregate statistics, such as mean, median, or standard deviation, for different time periods.

4. Model Development:

- Utilize statistical analysis techniques to uncover trends, patterns, and anomalies in the data. This can involve time series analysis, regression, clustering, or anomaly detection methods.

- Develop predictive models if you want to forecast future energy consumption based on historical data.

- Use tools like Python's Pandas, NumPy, scikit-learn, and libraries specific to time series analysis for this step.

5. Visualization:

- Create visualizations to present energy consumption trends and insights. This can include:

- Time series plots to visualize changes over time.

- Histograms or bar charts to show distribution of energy consumption.

- Heatmaps to display correlations between features.

- Box plots or violin plots for outlier detection.

- Geographic maps to visualize regional variations.

- Use visualization libraries like Matplotlib, Seaborn, Plotly, or geographic mapping tools like Folium (for maps).

6. Automation:

- Build a script or pipeline to automate the entire process, from data collection to visualization.

- Use scripting languages like Python to create a workflow that:

- Downloads and updates the dataset at regular intervals.

- Performs data preprocessing and feature extraction.

- Runs statistical analyses and generates insights.

- Creates and saves visualizations in a shareable format (e.g., PDFs or interactive web dashboards).

- Schedules the script to run automatically at specified intervals (e.g., daily, weekly).

SAMPLE PROGRAM:

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

# Step 1: Data Preprocessing

# Load the dataset

data = pd.read_csv('energy_consumption_data.csv')

# Data cleaning (handle missing values and outliers)

data.dropna(inplace=True)

data = data[data['EnergyConsumption'] > 0]  # Remove zero or negative values

# Convert the 'Date' column to datetime if not already

data['Date'] = pd.to_datetime(data['Date'])

# Step 2: Feature Extraction

# Extract relevant time-based features

data['Year'] = data['Date'].dt.year

data['Month'] = data['Date'].dt.month

data['DayOfWeek'] = data['Date'].dt.dayofweek
```

```
# Step 3: Model Development (Simple Visualization)

# Plot monthly energy consumption

monthly_avg = data.groupby('Month')['EnergyConsumption'].mean()

plt.figure(figsize=(10, 6))

plt.plot(monthly_avg.index, monthly_avg.values, marker='o')

plt.xlabel('Month')

plt.ylabel('Average Energy Consumption')

plt.title('Monthly Energy Consumption Trends')

plt.xticks(np.arange(1, 13), ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul',
'Aug', 'Sep', 'Oct', 'Nov', 'Dec'])

plt.grid(True)

plt.show()

# Step 4: Visualization (Additional Visualization)

# Create a histogram of energy consumption

plt.figure(figsize=(10, 6))

plt.hist(data['EnergyConsumption'], bins=30, edgecolor='k')

plt.xlabel('Energy Consumption')

plt.ylabel('Frequency')

plt.title('Energy Consumption Distribution')

plt.grid(True)
```

```
plt.show()

# Step 5: Automation

# You can create functions for these steps and schedule them to run periodically.
```