KINGS ENGINEERING COLLEGE

**PROJECT TITLE: MEASURE OF ENERGY CONSUMPTION**

TEAM MEMBERS: R. RAJA LAKSHMI, S. SUDHARSHANA DEVI, R. SHANTHINI

DEPARTMENT: ELECTRONICS AND COMMUNICATION ENGINEERING

MENTOR: BESTLEY JOE

TO LOAD AND PREPROCESSING THE DATASET

**Here's a detailed step-by-step guide for loading and preprocessing a dataset for measuring energy consumption in a project:**

**Define Objectives**: Clearly define the objectives of your project regarding energy consumption. What specific aspects are you aiming to measure or analyze? This will guide your data collection and preprocessing efforts.

**Identify Data Sources:** Determine where you'll get the energy consumption data. This could be from smart meters, IoT devices, utility bills, or any other relevant source.

**Data Gathering:** Collect raw data from the identified sources. Ensure that the data includes relevant information such as timestamps, power readings, and any additional contextual data.

**Data Format and Storage:** Choose a suitable format for your data (CSV, Excel, database, etc.) and store it in a structured manner. This facilitates easy loading and preprocessing.

**Data Cleaning**: Address any missing or erroneous values in the dataset. This may involve imputation or removal of incomplete records to ensure data quality.

**Handling Outliers**: Identify and handle outliers in the energy consumption data. Extreme values can skew the analysis, so consider strategies such as trimming or transforming data points.

**Data Normalization/Scaling:** Normalize or scale the data if necessary, especially if the dataset contains variables with different scales. This ensures that each feature contributes proportionally to the analysis

**Feature Engineering:** Create new features that might provide additional insights into energy consumption. For example, you might derive features like daily averages or identify peak usage periods.

**Time Series Handling:** If your dataset involves a time component, ensure that timestamps are in the correct format. Consider resampling or aggregating data to align with the analysis frequency (e.g., hourly, daily).

**Exploratory Data Analysis (EDA):** Perform exploratory data analysis to understand the distribution of energy consumption, trends, and patterns. Visualization tools can help in this stage.

**Correlation Analysis:** Examine correlations between different variables in the dataset. This can reveal relationships that might influence energy consumption.

**Data Splitting:** Split the dataset into training and testing sets. This ensures that you can train your models on one subset and validate their performance on another.

**Save Preprocessed Data:** Save the preprocessed dataset in a separate file or database. This allows you to easily access and use the cleaned data for further analysis or model training without repeating preprocessing steps.

**Here we use python and some common libraries for loading and preprocessing a dataset and also we assume a csv file with energy consumption data.**

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn.ensemble import RandomForestRegressor

from sklearn.metrics import mean\_squared\_error

import matplotlib.pyplot as plt

**# Step 1: Load the Dataset**

file\_path = 'energy\_consumption\_data.csv'

df = pd.read\_csv(file\_path)

**# Step 2: Data Cleaning**

**# Handle missing values if any**

df = df.dropna()

**# Step 3: Feature Engineering**

**# For simplicity, let's assume there's a 'timestamp' column and we extract features from it**

df['timestamp'] = pd.to\_datetime(df['timestamp'])

df['hour'] = df['timestamp'].dt.hour

df['day\_of\_week'] = df['timestamp'].dt.dayofweek

**# Step 4: Splitting Data**

features = df[['hour', 'day\_of\_week', 'other\_features']]

target = df['energy\_consumption']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(features, target, test\_size=0.2, random\_state=42)

**# Step 5: Preprocessing - Scaling**

scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train)

X\_test\_scaled = scaler.transform(X\_test)

**# Step 6: Model Training**

model = RandomForestRegressor(n\_estimators=100, random\_state=42)

model.fit(X\_train\_scaled, y\_train)

**# Step 7: Model Evaluation**

predictions = model.predict(X\_test\_scaled)

mse = mean\_squared\_error(y\_test, predictions)

print(f'Mean Squared Error: {mse}')

**# Step 8: Visualize Results**

plt.scatter(y\_test, predictions)

plt.xlabel('True Values')

plt.ylabel('Predictions')

plt.show()