MEASUREMENT OF ENERGY CONSUMPTION

Phase 3 Submission

Gowsalya D (411521104038)

**Project Goal and Scope:**

Define the specific objectives of your project. Are you monitoring energy consumption in a home, a building, or an industrial facility? What aspects of energy consumption do you want to measure (e.g., electricity, gas, water)?

**Hardware Selection:**

Choose the sensors and hardware components needed for measurement. For electricity, you might need current sensors or smart meters. For other resources, appropriate sensors should be selected.

**Data Acquisition System:**

Set up a data acquisition system to collect data from the sensors. This could involve microcontrollers (e.g., Arduino, Raspberry Pi) and data logging devices.

**Data Visualization and Storage:**

Create a database or data storage solution to store the collected data. Develop a user-friendly interface for data visualization, which can be a website or a mobile app.

**Real-time Monitoring:**

If needed, implement real-time monitoring and alerts. For example, you could set up alerts for unusual spikes in energy consumption.

**Analytics and Reports:**

Develop analytics to gain insights from the data. You might want to create reports or dashboards to track usage trends.

**Energy Efficiency Recommendations:**

Implement recommendations or suggestions for optimizing energy consumption based on the data collected. Machine learning models can help in this regard.

**Testing and Calibration:**

Ensure the accuracy of your measurements through testing and calibration of the sensors and system.

**Privacy and Security:**

Address privacy concerns, especially if this project involves personal or sensitive data. Implement security measures to protect the data and system.

**Scaling and Maintenance:**

Consider how the system can be scaled if needed. Regular maintenance is crucial to keep the system running efficiently.

**Data Integration:**

If relevant, integrate your system with other home automation or industrial control systems.

**Documentation:**

Document the project thoroughly, including schematics, code, and user manuals.

**Community and Feedback:**

Share your project with the community, gather feedback, and improve it over time.

**Objective:**

To create a system that measures and monitors the electricity consumption in a household and provides insights to help reduce energy usage and save on utility bills.

**Components:**

1.Raspberry Pi (or similar single-board computer)

2.Current sensor (e.g., SCT-013-030)

3.Voltage sensor (e.g., ACS712)

4.Wi-Fi adapter (for Raspberry Pi)

5.Relay (optional for controlling appliances)

6.Power strip (for easier connection)

7.Database and web server (you can use SQLite and Flask)

8.Mobile app or web interface for data visualization

**Hardware Setup:**

1.Connect the current sensor to monitor the main electricity supply line.

2.Attach the voltage sensor to measure the line voltage.

3.Connect the Raspberry Pi to these sensors and set up the Wi-Fi connection.

**Exploratory Analysis**

To begin this exploratory analysis, first use matplotlib to import libraries and define functions for plotting the data. Depending on the data, not all plots will be made.

From mpl\_toolkits.mplot3d import Axes3D

From sklearn.preprocessing import StandardScaler

Import matplotlib.pyplot as plt # plotting

Import numpy as np # linear algebra

Import os # accessing directory structure

Import pandas as pd # data processing, CSV file I/O (e.g. pd.read\_csv)

There is 0 csv file in the current version of the dataset:

For dirname, \_, filenames in os.walk(‘/kaggle/input’):

For filename in filenames:

Print(os.path.join(dirname, filename))

The next hidden code cells define functions for plotting data. Click on the “Code” button in the published kernel to reveal the hidden code.

# Distribution graphs (histogram/bar graph) of column data

Def plotPerColumnDistribution(df, nGraphShown, nGraphPerRow):

Nunique = df.nunique()

Df = df[[col for col in df if nunique[col] > 1 and nunique[col] < 50]] # For displaying purposes, pick columns that have between 1 and 50 unique values

nRow, nCol = df.shape

columnNames = list(df)

nGraphRow = (nCol + nGraphPerRow – 1) / nGraphPerRow

plt.figure(num = None, figsize = (6 \* nGraphPerRow, 8 \* nGraphRow), dpi = 80, facecolor = ‘w’, edgecolor = ‘k’)

for I in range(min(nCol, nGraphShown)):

plt.subplot(nGraphRow, nGraphPerRow, I + 1)

columnDf = df.iloc[:, i]

if (not np.issubdtype(type(columnDf.iloc[0]), np.number)):

valueCounts = columnDf.value\_counts()

valueCounts.plot.bar()

else:

columnDf.hist()

plt.ylabel(‘counts’)

plt.xticks(rotation = 90)

plt.title(f’{columnNames[i]} (column {i})’)

plt.tight\_layout(pad = 1.0, w\_pad = 1.0, h\_pad = 1.0)

plt.show()

# Correlation matrix

Def plotCorrelationMatrix(df, graphWidth):

Filename = df.dataframeName

Df = df.dropna(‘columns’) # drop columns with NaN

Df = df[[col for col in df if df[col].nunique() > 1]] # keep columns where there are more than 1 unique values

If df.shape[1] < 2:

Print(f’No correlation plots shown: The number of non-NaN or constant columns ({df.shape[1]}) is less than 2’)

Return

Corr = df.corr()

Plt.figure(num=None, figsize=(graphWidth, graphWidth), dpi=80, facecolor=’w’, edgecolor=’k’)

corrMat = plt.matshow(corr, fignum = 1)

plt.xticks(range(len(corr.columns)), corr.columns, rotation=90)

plt.yticks(range(len(corr.columns)), corr.columns)

plt.gca().xaxis.tick\_bottom()

plt.colorbar(corrMat)

plt.title(f’Correlation Matrix for {filename}’, fontsize=15)

plt.show()

# Scatter and density plots

Def plotScatterMatrix(df, plotSize, textSize):

Df = df.select\_dtypes(include =[np.number]) # keep only numerical columns

# Remove rows and columns that would lead to df being singular

Df = df.dropna(‘columns’)

Df = df[[col for col in df if df[col].nunique() > 1]] # keep columns where there are more than 1 unique values

columnNames = list(df)

if len(columnNames) > 10: # reduce the number of columns for matrix inversion of kernel density plots

columnNames = columnNames[:10]

df = df[columnNames]

ax = pd.plotting.scatter\_matrix(df, alpha=0.75, figsize=[plotSize, plotSize], diagonal=’kde’)

corrs = df.corr().values

for I, j in zip(\*plt.np.triu\_indices\_from(ax, k = 1)):

ax[I, j].annotate(‘Corr. Coef = %.3f’ % corrs[I, j], (0.8, 0.2), xycoords=’axes fraction’, ha=’center’, va=’center’, size=textSize)

plt.suptitle(‘Scatter and Density Plot’)

plt.show()

**Attribute Information:**

1.date: Date in format dd/mm/yyyy

2.time: time in format hh:mm:ss

3.global\_active\_power: household global minute-averaged active power (in kilowatt)

4.global\_reactive\_power: household global minute-averaged reactive power (in kilowatt)

5.voltage: minute-averaged voltage (in volt)

6.global\_intensity: household global minute-averaged current intensity (in ampere)

7.sub\_metering\_1: energy sub-metering No. 1 (in watt-hour of active energy). It corresponds to the kitchen, containing mainly a dishwasher, an oven and a microwave (hot plates are not electric but gas powered).

8.sub\_metering\_2: energy sub-metering No. 2 (in watt-hour of active energy). It corresponds to the laundry room, containing a washing-machine, a tumble-drier, a refrigerator and a light.

9.sub\_metering\_3: energy sub-metering No. 3 (in watt-hour of active energy). It corresponds to an electric water-heater and an air-conditioner.

**PROGRAMS FOR DATA SET EXAMPLE:**

1.Import matplotlib.pyplot as plt

I = 1

Cols=[0, 1, 3, 4, 5, 6]

Plt.figure(figsize=(20, 10))

For col in cols:

Plt.subplot(len(cols), 1, i)

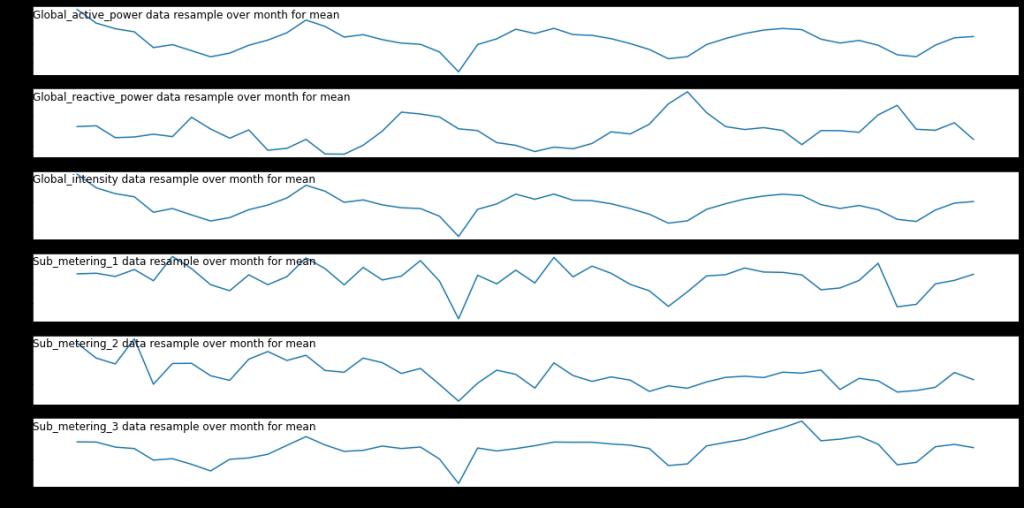
Plt.plot(df.resample(‘M’).mean().values[:, col])

Plt.title(df.columns[col] + ‘ data resample over month for mean’, y=0.75, loc=’left’)

I += 1

Plt.show()

OUTPUT:



2.i = 1

Cols=[0, 1, 3, 4, 5, 6]

Plt.figure(figsize=(20, 10))

For col in cols:

Plt.subplot(len(cols), 1, i)

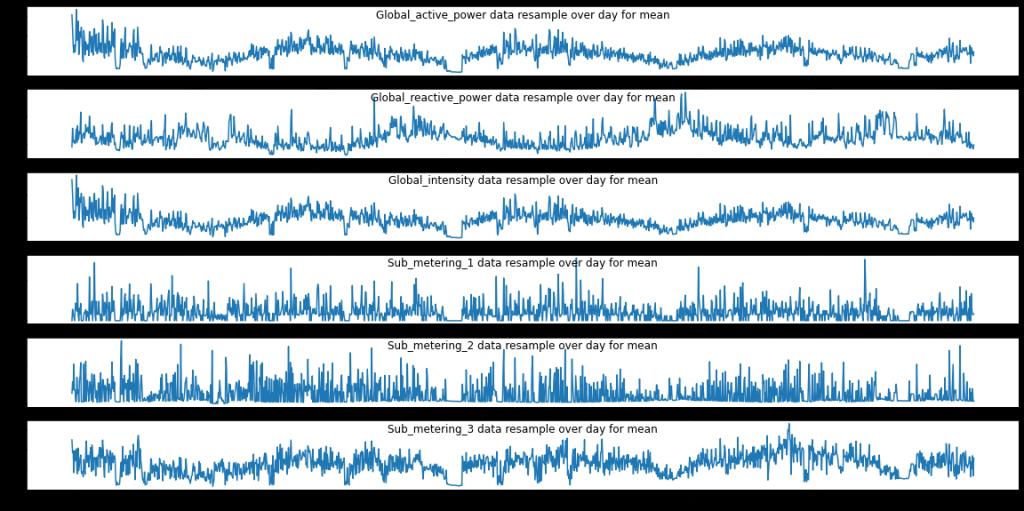
Plt.plot(df.resample(‘D’).mean().values[:, col])

Plt.title(df.columns[col] + ‘ data resample over day for mean’, y=0.75, loc=’center’)

I += 1

Plt.show()

OUTPUT:



3.i = 1

Cols=[0, 1, 3, 4, 5, 6]

Plt.figure(figsize=(20, 10))

For col in cols:

Plt.subplot(len(cols), 1, i)

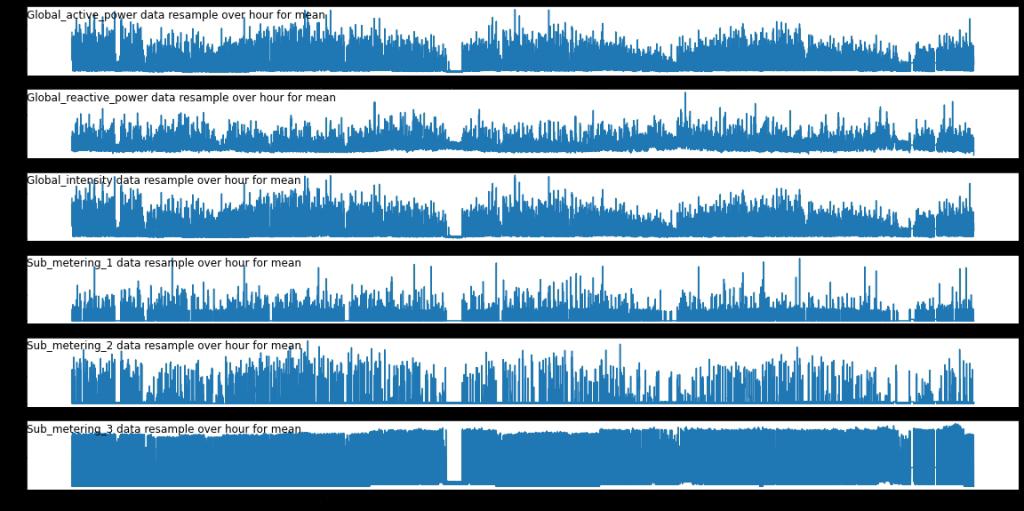
Plt.plot(df.resample(‘H’).mean().values[:, col])

Plt.title(df.columns[col] + ‘ data resample over hour for mean’, y=0.75, loc=’left’)

I += 1

Plt.show()

OUTPUT:



**Observations from the above visualizations:**

Resampling by month, date or time is very important because it has a great interaction as expected (changing the periodicity of the system).

Therefore, if you process all the original data, the run time will be very expensive, but if you process data with large timescale samples (e.g., monthly), it will affect the predictability of the model.

From observation, we can see it is relatively reasonable to resample the data per hour.

**Conclusion:**

Conservation of energy is an action made to decrease energy consumption by any means. This can be done by using less energy. It is important for saving our natural resources.