*The following codes in Phyton used to conduct the analysis based on the research objectives:*

*Specific objective 1.* ***Identify Key Data Types Collected from Smart Homes in two parts: 1.*** *Investigate the variety of data generated by smart home devices, including sensor readings, energy usage, and user interactions. 2. Categorize and evaluate the relevance of these data types to automation, security, and energy efficiency.*

*import pandas as pd*

*import matplotlib.pyplot as plt*

*import seaborn as sns*

*# Load the dataset*

*file\_path = '/mnt/data/10000 Mahdi\_Houshangi\_sampled\_smart\_home\_dataset.csv'*

*smart\_home\_data = pd.read\_csv(file\_path)*

*# Display the first few rows and basic information of the dataset to understand its structure*

*smart\_home\_data\_info = smart\_home\_data.info()*

*smart\_home\_data\_head = smart\_home\_data.head()*

*smart\_home\_data\_info, smart\_home\_data\_head*

*STDOUT/STDERR*

*<class 'pandas.core.frame.DataFrame'>*

*RangeIndex: 10000 entries, 0 to 9999*

*Data columns (total 18 columns):*

*# Column Non-Null Count Dtype*

*--- ------ -------------- -----*

*0 Datetime 10000 non-null object*

*1 Transaction\_ID 10000 non-null int64*

*2 Television 10000 non-null int64*

*3 Dryer 10000 non-null int64*

*4 Oven 10000 non-null int64*

*5 Refrigerator 10000 non-null int64*

*6 Microwave 10000 non-null int64*

*7 Line Voltage 10000 non-null int64*

*8 Voltage 10000 non-null int64*

*9 Apparent Power 10000 non-null int64*

*10 Energy Consumption (kWh) 10000 non-null float64*

*11 Month 10000 non-null object*

*12 Day of the Week 10000 non-null object*

*13 Hour of the Day 10000 non-null int64*

*14 Offloading Decision 10000 non-null object*

*15 Unnamed: 15 0 non-null float64*

*16 Unnamed: 16 0 non-null float64*

*17 Unnamed: 17 0 non-null float64*

*dtypes: float64(4), int64(10), object(4)*

*memory usage: 1.4+ MB*

*Result*

*(None,*

*Datetime Transaction\_ID Television Dryer Oven Refrigerator \*

*0 7/26/2022 17:56 32115 1 0 0 0*

*1 2/15/2021 5:33 14084 1 1 0 0*

*2 2/15/2022 8:32 26588 1 1 1 0*

*3 1/12/2022 2:29 25415 0 1 0 1*

*4 1/24/2021 7:15 13333 0 0 1 0*

*Microwave Line Voltage Voltage Apparent Power Energy Consumption (kWh) \*

*0 1 236 231 1917 38.824861*

*1 1 220 214 1695 96.218052*

*2 1 233 232 1942 34.065614*

*3 0 229 223 1757 11.898820*

*4 1 228 227 1656 24.689123*

*Month Day of the Week Hour of the Day Offloading Decision Unnamed: 15 \*

*0 July Tuesday 17 Local NaN*

*1 February Monday 5 Local NaN*

*2 February Tuesday 8 Local NaN*

*3 January Wednesday 2 Local NaN*

*4 January Sunday 7 Local NaN*

*Unnamed: 16 Unnamed: 17*

*0 NaN NaN*

*1 NaN NaN*

*2 NaN NaN*

*3 NaN NaN*

*4 NaN NaN )*

*Then the dataset cleaned up and generated descriptive statistics. Then visualized distributions (e.g., histograms, boxplots). Finaly calculate correlations among impactful variables using the following codes:*

*# Clean the dataset by dropping unnecessary columns*

*smart\_home\_data\_cleaned = smart\_home\_data.drop(columns=['Unnamed: 15', 'Unnamed: 16', 'Unnamed: 17'])*

*# Generate descriptive statistics for numerical variables*

*descriptive\_stats = smart\_home\_data\_cleaned.describe()*

*# Plot histograms for key numeric variables*

*numeric\_columns = ['Line Voltage', 'Voltage', 'Apparent Power', 'Energy Consumption (kWh)']*

*smart\_home\_data\_cleaned[numeric\_columns].hist(bins=15, figsize=(12, 8))*

*plt.suptitle('Histograms of Key Numeric Variables', fontsize=16)*

*plt.tight\_layout()*

*# Generate a correlation heatmap*

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

*correlation\_matrix = smart\_home\_data\_cleaned[numeric\_columns].corr()*

*sns.heatmap(correlation\_matrix, annot=True, cmap='coolwarm', fmt=".2f")*

*plt.title('Correlation Heatmap of Energy Metrics', fontsize=16)*

*plt.show()*

*descriptive\_stats*

*Result*

*Transaction\_ID Television Dryer Oven Refrigerator \*

*count 10000.000000 10000.000000 10000.000000 10000.000000 10000.000000*

*mean 24466.067300 0.499800 0.498900 0.507600 0.504100*

*std 14202.332348 0.500025 0.500024 0.499967 0.500008*

*min 32.000000 0.000000 0.000000 0.000000 0.000000*

*25% 12075.750000 0.000000 0.000000 0.000000 0.000000*

*50% 24544.500000 0.000000 0.000000 1.000000 1.000000*

*75% 36800.750000 1.000000 1.000000 1.000000 1.000000*

*max 48972.000000 1.000000 1.000000 1.000000 1.000000*

*Microwave Line Voltage Voltage Apparent Power \*

*count 10000.000000 10000.000000 10000.000000 10000.000000*

*mean 0.495100 229.523600 225.038200 1751.121200*

*std 0.500001 5.802449 6.476259 144.432146*

*min 0.000000 220.000000 211.000000 1500.000000*

*25% 0.000000 224.000000 220.000000 1627.000000*

*50% 0.000000 230.000000 225.000000 1751.000000*

*75% 1.000000 235.000000 230.000000 1876.000000*

*max 1.000000 239.000000 239.000000 1999.000000*

*Energy Consumption (kWh) Hour of the Day*

*count 10000.000000 10000.000000*

*mean 55.297014 11.435500*

*std 25.773634 6.930458*

*min 10.002715 0.000000*

*25% 33.275391 5.000000*

*50% 55.618034 11.000000*

*75% 77.454878 17.000000*

*max 99.974912 23.000000*

*Specific objective 2****. Analyze Patterns in Device Usage and Energy Consumption*** *in the following parts: 1. Examine usage trends and behavioral patterns across different devices and household contexts. 2.* *Identify inefficiencies in energy consumption and propose strategies for optimization.*

*# Re-import necessary libraries after reset*

*import pandas as pd*

*import matplotlib.pyplot as plt*

*import seaborn as sns*

*# Reload the dataset*

*file\_path = '/mnt/data/10000 Mahdi\_Houshangi\_sampled\_smart\_home\_dataset.csv'*

*smart\_home\_data = pd.read\_csv(file\_path)*

*# Clean the dataset by dropping unnecessary columns*

*smart\_home\_data\_cleaned = smart\_home\_data.drop(columns=['Unnamed: 15', 'Unnamed: 16', 'Unnamed: 17'])*

*# Convert Datetime column to datetime format for time-based analysis*

*smart\_home\_data\_cleaned['Datetime'] = pd.to\_datetime(smart\_home\_data\_cleaned['Datetime'])*

*# Part 1: Examine usage trends and behavioral patterns across different devices and household contexts*

*# Group by hour to analyze hourly energy consumption trends*

*hourly\_energy\_consumption = smart\_home\_data\_cleaned.groupby('Hour of the Day')['Energy Consumption (kWh)'].mean()*

*# Group by day of the week for behavioral analysis*

*daily\_energy\_consumption = smart\_home\_data\_cleaned.groupby('Day of the Week')['Energy Consumption (kWh)'].mean()*

*# Plot hourly energy consumption trends*

*plt.figure(figsize=(10, 5))*

*hourly\_energy\_consumption.plot(kind='line', marker='o', color='b')*

*plt.title('Hourly Energy Consumption Trend')*

*plt.xlabel('Hour of the Day')*

*plt.ylabel('Average Energy Consumption (kWh)')*

*plt.grid(True)*

*plt.show()*

*# Plot daily energy consumption trends*

*plt.figure(figsize=(10, 5))*

*daily\_energy\_consumption.plot(kind='bar', color='g')*

*plt.title('Daily Energy Consumption Trend')*

*plt.xlabel('Day of the Week')*

*plt.ylabel('Average Energy Consumption (kWh)')*

*plt.xticks(rotation=45)*

*plt.grid(True)*

*plt.show()*

*# Analyze appliance-specific trends by aggregating their usage*

*appliance\_usage = smart\_home\_data\_cleaned[['Television', 'Dryer', 'Oven', 'Refrigerator', 'Microwave']].sum()*

*# Plot appliance usage frequency*

*plt.figure(figsize=(10, 5))*

*appliance\_usage.plot(kind='bar', color='purple')*

*plt.title('Total Appliance Usage Frequency')*

*plt.xlabel('Appliance')*

*plt.ylabel('Total Usage Count')*

*plt.xticks(rotation=45)*

*plt.grid(True)*

*plt.show()*

*# Part 2: Identify inefficiencies in energy consumption*

*# Compare energy consumption during peak vs. off-peak hours*

*peak\_hours = smart\_home\_data\_cleaned[(smart\_home\_data\_cleaned['Hour of the Day'] >= 18) | (smart\_home\_data\_cleaned['Hour of the Day'] <= 7)]*

*off\_peak\_hours = smart\_home\_data\_cleaned[(smart\_home\_data\_cleaned['Hour of the Day'] > 7) & (smart\_home\_data\_cleaned['Hour of the Day'] < 18)]*

*# Calculate average consumption*

*peak\_consumption = peak\_hours['Energy Consumption (kWh)'].mean()*

*off\_peak\_consumption = off\_peak\_hours['Energy Consumption (kWh)'].mean()*

*# Visualize peak vs. off-peak energy consumption*

*plt.figure(figsize=(8, 5))*

*plt.bar(['Peak Hours (6PM-7AM)', 'Off-Peak Hours (8AM-5PM)'], [peak\_consumption, off\_peak\_consumption], color=['red', 'blue'])*

*plt.title('Energy Consumption: Peak vs. Off-Peak Hours')*

*plt.ylabel('Average Energy Consumption (kWh)')*

*plt.grid(True)*

*plt.show()*

*peak\_consumption, off\_peak\_consumption*

*Result*

*(55.63319748398872, 54.82331314173935)*

*# Re-import necessary libraries after reset*

*import pandas as pd*

*import matplotlib.pyplot as plt*

*import seaborn as sns*

*from sklearn.cluster import KMeans*

*from sklearn.preprocessing import StandardScaler*

*# Reload the dataset*

*file\_path = '/mnt/data/10000 Mahdi\_Houshangi\_sampled\_smart\_home\_dataset.csv'*

*smart\_home\_data = pd.read\_csv(file\_path)*

*# Clean the dataset by dropping unnecessary columns*

*smart\_home\_data\_cleaned = smart\_home\_data.drop(columns=['Unnamed: 15', 'Unnamed: 16', 'Unnamed: 17'], errors='ignore')*

*# Convert 'Datetime' column to datetime format for time-series analysis*

*smart\_home\_data\_cleaned['Datetime'] = pd.to\_datetime(smart\_home\_data\_cleaned['Datetime'])*

*# Set datetime as index for easier time-series analysis*

*smart\_home\_data\_cleaned.set\_index('Datetime', inplace=True)*

*# Resample data to analyze energy consumption trends by hour and day*

*hourly\_energy\_consumption = smart\_home\_data\_cleaned['Energy Consumption (kWh)'].resample('H').mean()*

*daily\_energy\_consumption = smart\_home\_data\_cleaned['Energy Consumption (kWh)'].resample('D').mean()*

*# Plot hourly and daily energy consumption trends*

*plt.figure(figsize=(14, 6))*

*hourly\_energy\_consumption.plot(title='Hourly Energy Consumption Trends', color='blue')*

*plt.xlabel('Time')*

*plt.ylabel('Energy Consumption (kWh)')*

*plt.grid(True)*

*plt.show()*

*plt.figure(figsize=(14, 6))*

*daily\_energy\_consumption.plot(title='Daily Energy Consumption Trends', color='green')*

*plt.xlabel('Time')*

*plt.ylabel('Energy Consumption (kWh)')*

*plt.grid(True)*

*plt.show()*

*# Prepare data for clustering by extracting relevant features*

*clustering\_features = smart\_home\_data\_cleaned[['Energy Consumption (kWh)', 'Hour of the Day', 'Apparent Power']]*

*scaler = StandardScaler()*

*scaled\_features = scaler.fit\_transform(clustering\_features)*

*# Apply K-means clustering with an optimal number of clusters (e.g., 3)*

*kmeans = KMeans(n\_clusters=3, random\_state=42, n\_init=10)*

*smart\_home\_data\_cleaned['Cluster'] = kmeans.fit\_predict(scaled\_features)*

*# Visualize clusters using a scatter plot*

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

*sns.scatterplot(x=smart\_home\_data\_cleaned['Hour of the Day'], y=smart\_home\_data\_cleaned['Energy Consumption (kWh)'],*

*hue=smart\_home\_data\_cleaned['Cluster'], palette='viridis')*

*plt.title('Clustering of Energy Consumption Patterns')*

*plt.xlabel('Hour of the Day')*

*plt.ylabel('Energy Consumption (kWh)')*

*plt.show()*

*# Correlation analysis*

*correlation\_matrix = smart\_home\_data\_cleaned[['Energy Consumption (kWh)', 'Hour of the Day', 'Apparent Power']].corr()*

*plt.figure(figsize=(8, 5))*

*sns.heatmap(correlation\_matrix, annot=True, cmap='coolwarm', fmt=".2f")*

*plt.title('Correlation Analysis of Key Factors')*

*plt.show()*

*smart\_home\_data\_cleaned['Cluster'].value\_counts()*

*Specific objective 3.* ***Develop Intelligent Automation Rules and Recommendations i****n the following parts: 1. Utilize machine learning and data analysis techniques to create adaptive automation rules tailored to user preferences and behaviors. 2. Recommend improvements to existing systems to enhance usability, interoperability, and overall efficiency.*

*# Re-import necessary libraries after reset*

*import pandas as pd*

*import matplotlib.pyplot as plt*

*import seaborn as sns*

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

*from sklearn.tree import DecisionTreeClassifier*

*from sklearn.metrics import accuracy\_score, classification\_report*

*# Reload the dataset*

*file\_path = '/mnt/data/10000 Mahdi\_Houshangi\_sampled\_smart\_home\_dataset.csv'*

*smart\_home\_data = pd.read\_csv(file\_path)*

*# Clean the dataset by dropping unnecessary columns*

*smart\_home\_data\_cleaned = smart\_home\_data.drop(columns=['Unnamed: 15', 'Unnamed: 16', 'Unnamed: 17'], errors='ignore')*

*# Convert 'Datetime' column to datetime format*

*smart\_home\_data\_cleaned['Datetime'] = pd.to\_datetime(smart\_home\_data\_cleaned['Datetime'])*

*# Encode categorical variable 'Offloading Decision'*

*smart\_home\_data\_cleaned['Offloading Decision'] = smart\_home\_data\_cleaned['Offloading Decision'].map({'Local': 0, 'Remote': 1})*

*# Define feature set and target variable for machine learning*

*features = ['Energy Consumption (kWh)', 'Hour of the Day', 'Apparent Power', 'Offloading Decision']*

*target = 'Refrigerator' # Example of predicting refrigerator usage*

*X = smart\_home\_data\_cleaned[features]*

*y = smart\_home\_data\_cleaned[target]*

*# Split data into training and test sets*

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

*# Train a decision tree classifier to predict appliance usage*

*model = DecisionTreeClassifier(max\_depth=5, random\_state=42)*

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

*# Predict on test set*

*y\_pred = model.predict(X\_test)*

*# Evaluate model performance*

*accuracy = accuracy\_score(y\_test, y\_pred)*

*classification\_rep = classification\_report(y\_test, y\_pred)*

*# Feature importance analysis*

*feature\_importances = pd.Series(model.feature\_importances\_, index=features).sort\_values(ascending=False)*

*# Visualize feature importance*

*plt.figure(figsize=(8, 5))*

*sns.barplot(x=feature\_importances.values, y=feature\_importances.index)*

*plt.title('Feature Importance for Appliance Usage Prediction')*

*plt.xlabel('Importance Score')*

*plt.ylabel('Feature')*

*plt.show()*

*accuracy, classification\_rep*

*Result*

*(0.5085,*

*' precision recall f1-score support\n\n 0 0.51 0.57 0.54 998\n 1 0.51 0.45 0.48 1002\n\n accuracy 0.51 2000\n macro avg 0.51 0.51 0.51 2000\nweighted avg 0.51 0.51 0.51 2000\n')*