MEASURE ENERGY CONSUMPTION

NAME: LINGESH.G

NM ID: aut2221l302

Phase 5 document

Project Documentation & Submission

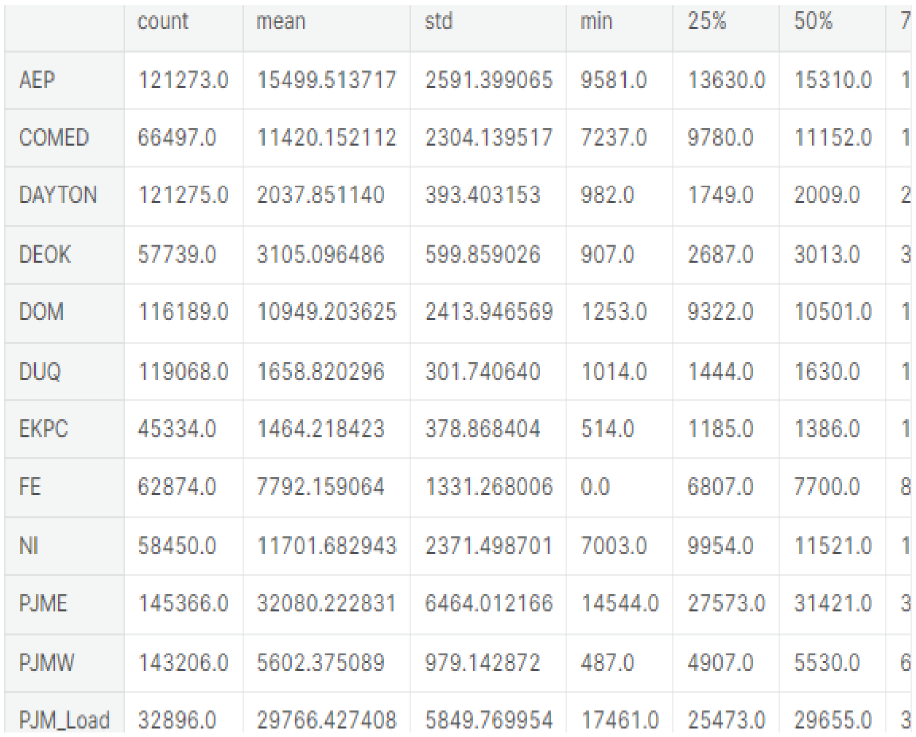
INTRODUCTION:-

- Energy consumption is a critical aspect of modern society, impacting not only our daily lives but also the environment and global sustainability.
- The ability to accurately measure and predict energy consumption is essential for optimizing resource allocation, reducing waste, and supporting the transition to more sustainable energy sources.
- In this context, machine learning plays a pivotal role in enhancing our ability to monitor and manage energy consumption effectively.
- Machine learning, a subfield of artificial intelligence, leverages algorithms and statistical techniques to extract valuable insights from data.
- This Introduction outlines the significance of using machine learning for measuring energy consumption and highlights its potential benefits for both consumers and the environment.
- Data Model training helps create a simplified, logical database that eliminates redundancy, reduces storage requirements, and enables efficient retrieval.

❖ This introduction will guide you through the initial steps of the process. We'll explore how to import essential libraries, load the energy dataset, and perform critical preprocessing steps.

GIVEN DATASET:

A good data source for hourly energy consumption using machine learning should be accurate and complete.



	count	mean	std	min	25%	50%	7
AEP	121273.0	15499.513717	2591.399065	9581.0	13630.0	15310.0	1
COMED	66497.0	11420.152112	2304.139517	7237.0	9780.0	11152.0	1
DAYTON	121275.0	2037.851140	393.403153	982.0	1749.0	2009.0	2
DEOK	57739.0	3105.096486	599.859026	907.0	2687.0	3013.0	3
DOM	116189.0	10949.203625	2413.946569	1253.0	9322.0	10501.0	1
DUQ	119068.0	1658.820296	301.740640	1014.0	1444.0	1630.0	1
EKPC	45334.0	1464.218423	378.868404	514.0	1185.0	1386.0	1
FE	62874.0	7792.159064	1331.268006	0.0	6807.0	7700.0	8
NI	58450.0	11701.682943	2371.498701	7003.0	9954.0	11521.0	1
PJME	145366.0	32080.222831	6464.012166	14544.0	27573.0	31421.0	3
PJMW	143206.0	5602.375089	979.142872	487.0	4907.0	5530.0	6
PJM_Load	32896.0	29766.427408	5849.769954	17461.0	25473.0	29655.0	3

List of tools and software used:

Programming Language: Python is the most popular language for machine learning due to its extensive libraries and frameworks. You can use libraries like NumPy, pandas, scikit-learn, and more.

Energy Management Software: Software solutions like Energy CAP and energy plus are used for monitoring, managing, and optimizing energy consumption data.

Smart Meters: Devices for collecting real-time energy consumption data.

IoT Devices: Sensors and IoT platforms (e.g., Arduino, Raspberry Pi) for monitoring energy usage and environmental factors.

SCADA (Supervisory Control and Data Acquisition) Systems: Used for real-time data acquisition and control in industrial settings.

Microsoft Excel: Often used for basic data analysis and visualization.

Python: Libraries like pandas, NumPy, and matplotlib for data analysis and visualization.

Scikit-Learn: A popular Python library for machine learning tasks.

TensorFlow and PyTorch: Frameworks for building deep learning models.

AutoML Tools: Automated machine learning tools like Google AutoML and H2O.ai for simplified model development.

Energy Auditing Tools: Retroficiency, EnergyCAP, EnergyAudit: Tools for conducting energy audits to identify opportunities for energy efficiency improvements.

Optimization Software: Tools for optimizing energy usage, such as GAMS and AMPL.

Energy Management Software: Some solutions, like EnergyCAP, also provide utility billing and cost tracking features.

1. design thinking and present in form of document:

1. Empathize:

Understand the needs and challenges of all stakeholders involved in the energy consumption process, including homebuyers, sellers, real estate professionals, appraisers, and investors. Conduct interviews and surveys to gather insights on what users value in property valuation and what information is most critical for their decision-making.

2. Define:

Clearly articulate the problem statement, such as "How might we predict house energy prices more accurately and transparently using machine learning?" Identify the key goals and success criteria for the project, such as increasing prediction accuracy, reducing bias, or improving user trust in the valuation process.

3. Ideate:

Brainstorm creative solutions and data sources that can enhance the accuracy and transparency of house price predictions. Encourage interdisciplinary collaboration to generate a wide range of ideas, including the use of alternative data, new algorithms, or improved visualization techniques.

4. Prototype:

Create prototype machine learning models based on the ideas generated during the ideation phase. Test and iterate on these prototypes to determine which approaches are most promising in terms of accuracy and usability.

5. Test:

Gather feedback from users and stakeholders by testing the machine learning models with real-world data and scenarios. Assess how well the models meet the defined goals and success criteria, and make adjustments based on user feedback.

6. Implement:

Develop a production-ready machine learning solution for predicting house energy consumption prices, integrating the best-performing algorithms and data sources. Implement transparency measures, such as model interpretability tools, to ensure users understand how predictions are generated.

7. Evaluate:

Continuously monitor the performance of the machine learning model after implementation to ensure it remains accurate and relevant in a changing real estate market.

8. Iterate:

Apply an iterative approach to refine the machine learning model based on ongoing feedback and changing user needs. Continuously seek ways to enhance prediction accuracy, transparency, and user satisfaction.

9. Scale and Deploy:

Once the machine learning model has been optimized and validated, deploy it at scale to serve a broader audience, such as real estate professionals, investors, and homeowners. Ensure the model is accessible through user-friendly interfaces and integrates seamlessly into real estate workflows.

10.Educate and Train:

Provide training and educational resources to help users understand how the machine learning model works, what factors it considers, and its limitations. Foster a culture of data literacy among stakeholders to enhance trust in the technology.

2. DESIGN INTO INNOVATION

1. **Data Collection:** Gather a comprehensive dataset that includes features such as location, size, age, amenities, nearby schools, crime rates, and other relevant variables.

2. **Data Preprocessing:** Clean the data by handling missing values, outliers, and encoding categorical variables. Standardize or normalize

PYTHON PROGRAM:

# import libraries

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns


RED="\033[91m"

GREEN="\033[92m"

YELLOW="\033[93m"

BLUE="\033[94m"

RESET="\033[0m"
```

#LOAD DATASET

```
df=pd.read_csv("C:\\Users\\VIJAY\\Desktop\\PJMW_hourly.csv")

df["Datetime"]=pd.to_datetime(df["Datetime"])
```

#DATA CLEANING

```
print(BLUE +"\n DATA CLEANING" +RESET)

#Check for missing values

missing_values=df.isnull().sum()
```

```
print(GREEN + "Missing values:" +RESET)

print(missing_values)


# HANDLE MISSING VALUES

df.dropna(inplace=True)


# CHECK FOR DUPLICATE VALUES

duplicate_values=df.duplicated().sum()

print(GREEN +"Duplicate Values :"+RESET)

print(duplicate_values)


# DROP DUPLICATE VALUES

df.drop_duplicates(inplace=True)


# DATA ANALYSIS

print(BLUE +"\n DATA ANALYSIS"+RESET)

#SUMMARY STATISTICS

summary_stats=df.describe()

print(GREEN +"Summary statistics :"+RESET)

print(summary_stats)
```

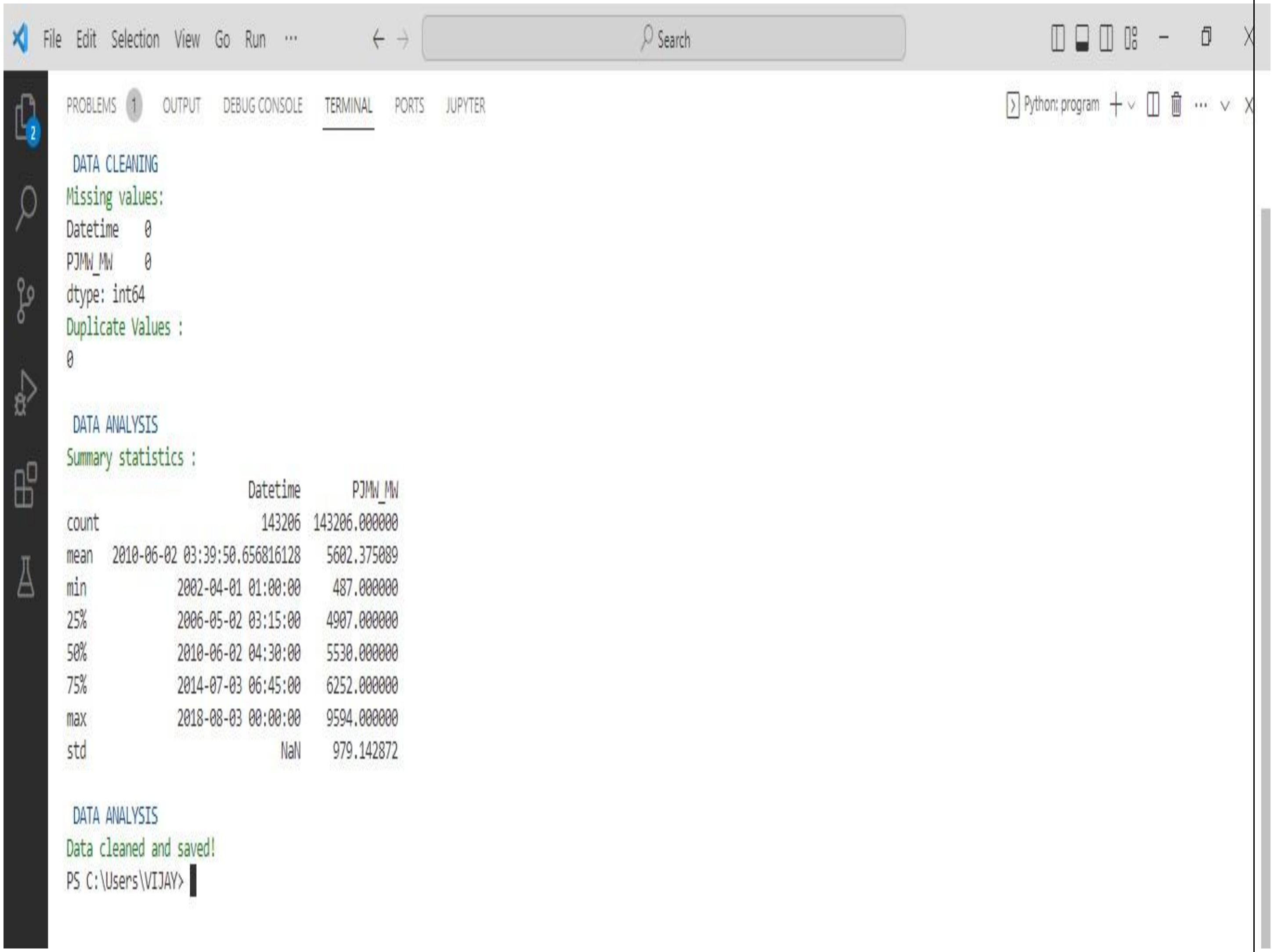
#DATA VISUALIZATION

```
#line plot for energy consumption over time

plt.figure(figsize=(12,6))

plt.plot(df.index,df["PJMW_MW"],label="Energy consumption(PJMW_MW)")

plt.xlabel("Datetime")

plt.ylabel("Energy consumption(MW)")

plt.title("Energy consumption Over Time")

plt.grid()

plt.legend()

plt.show()
```

#SAVING THE FILE

```
df.to_csv("cleaned_PJMW_hourly.csv",index=False)

print(BLUE+"\n DATA ANALYSIS" +RESET)

PRINT(GREEN +"Data cleaned and saved!"+RESET
```

Output:


File Edit Selection View Go Run ···
Search
PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS JUPYTER
Python: program

```
 DATA CLEANING
Missing values:
Datetime    0
PJMW_MW     0
dtype: int64
Duplicate Values :
0

 DATA ANALYSIS
Summary statistics :
                             Datetime        PJMW_MW
count                          143206  143206.000000
mean    2010-06-02 03:39:50.656816128    5602.375089
min               2002-04-01 01:00:00     487.000000
25%               2006-05-02 03:15:00    4907.000000
50%               2010-06-02 04:30:00    5530.000000
75%               2014-07-03 06:45:00    6252.000000
max               2018-08-03 00:00:00    9594.000000
std                               NaN     979.142872

 DATA ANALYSIS
Data cleaned and saved!
PS C:\Users\VIJAY>
```

1. Feature Engineering:

Create new features or transform existing ones to extract more valuable information. For example, you can calculate the distance to the nearest public transportation, or create a feature for the overall condition of the house.

2. **Model Selection:**

Choose the appropriate machine learning model for the task. Common models for regression problems like house price prediction include Linear Regression, Decision Trees, Random Forest, Gradient Boosting, and Neural Networks.

3. **Training:**

Split the dataset into training and testing sets to evaluate the model's performance. Consider techniques like cross-validation to prevent over fitting.

4. **Hyperparameter Tuning:**

Optimize the model's hyperparameters to improve its predictive accuracy. Techniques like grid search or random search can help with this.

5. Evaluation Metrics:

Select appropriate evaluation metrics for regression tasks, such as Mean Absolute Error (MAE), Mean Squared Error (MSE), or Root Mean Squared Error (RMSE). Choose the metric that aligns with the specific objectives of your project.

6. Regularization:

Apply regularization techniques like L1 (Lasso) or L2 (Ridge) regularization to prevent over fitting.

7. Feature Selection:

Use techniques like feature importance scores or recursive feature elimination to identify the most relevant features for the prediction.

8. Interpretability:

Ensure that the model's predictions are interpretable and explainable. This is especially important for real estate applications where stakeholders want to understand the factors affecting predictions.

9. Deployment:

Develop a user-friendly interface or API for end-users to input property details and receive price predictions

10. Model Improvement

Implement a feedback loop for continuous model improvement based on user feedback and new data.

11. Ethical Considerations:

Be mindful of potential biases in the data and model. Ensure fairness and transparency in your predictions.

12. Monitoring and Maintenance:

Regularly monitor the model's performance in the real world and update it as needed.

13. Innovation:

Consider innovative approaches such as using satellite imagery or IoT data for real-time property condition monitoring, or integrating natural language processing for textual property descriptions.

3. BUILD LOADING AND PREPROCESSING THE DATASET

Clean, well structured data is essential for training and testing. Proper data preprocessing can have a significant impact on the quality and effectiveness.

1. Data Collection:

Obtain a dataset that contains information about energy consumption and their corresponding prices. This dataset can be obtained from sources like real buildings, homes, or other electricity devices.

2. Load the Dataset:

Import relevant libraries, such as pandas for data manipulation and numpy for numerical operations. Load the dataset into a pandas Data Frame for easy data handling. You can use pd.read_csv() for CSV files or other appropriate functions for different file formats.

```
# Program
import pandas as pd
import numpy as np

# Load the dataset
dataset = pd.read_csv("energy_consumption.csv")

# Display basic information about the dataset
```

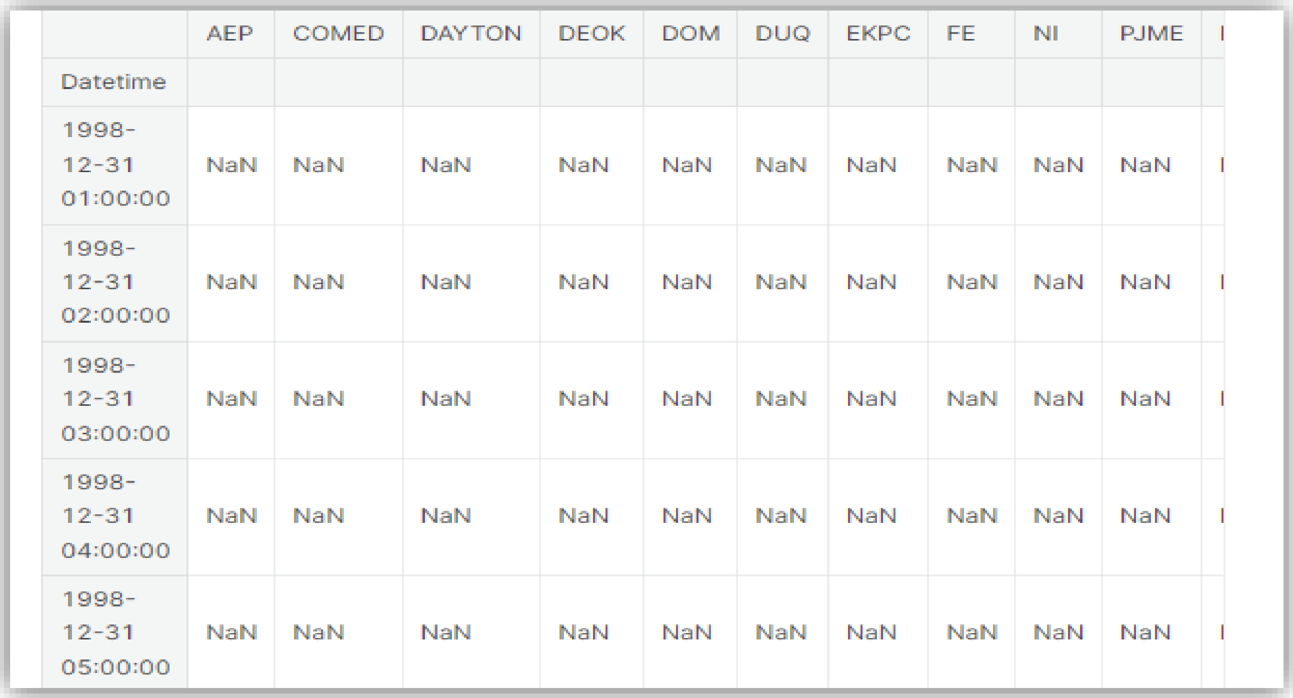
```
print("Dataset Info:")
print(dataset.info())

# Display the first few rows of the dataset
print("\nFirst 5 Rows of the Dataset:")
print(dataset.head())

# Check for missing values
missing_values = dataset.isnull().sum()
print("\nMissing Values:")
print(missing_values)

# Handle missing values (if any)
# Example: Replace missing values with the mean of the column
dataset.fillna(dataset.mean(), inplace=True)

# Check for duplicate rows
duplicate_rows = dataset.duplicated().sum()
print("\nDuplicate Rows:", duplicate_rows)

# Remove duplicate rows (if any)
```

dataset = dataset.drop_duplicates()

# Perform additional preprocessing steps (e.g., data transformation, feature engineering)

# Save the preprocessed dataset to a new CSV file

dataset.to_csv("preprocessed_energy_consumption.csv", index=False)



	AEP	COMED	DAYTON	DEOK	DOM	DUQ	EKPC	FE	NI	PJME	I
Datetime											
1998-12-31 01:00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	I
1998-12-31 02:00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	I
1998-12-31 03:00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	I
1998-12-31 04:00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	I
1998-12-31 05:00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	I

3. Data Exploration:

Explore the dataset to understand its structure and contents.

Check for the presence of missing values, outliers, and data types of each feature.

4. Data Cleaning:

Handle missing values by either removing rows with missing data or imputing values based on the nature of the data.

5. Feature Selection:

Identify relevant features for measuring energy consumption. Features like the number of bedrooms, square footage, location, and amenities are often important.

4. PERFORMING DIFFERENT ACTIVITIES LIKE FEATURE ENGINEERING, MODEL TRAINING, EVALUATION

1. Feature Engineering:

As mentioned earlier, feature engineering is crucial. It involves creating new features or transforming existing ones to provide meaningful information for your model.

Extracting information from textual descriptions (e.g., presence of keywords like "pool" or "granite countertops").

Calculating distances to key locations (e.g., schools, parks)

2. Data Preprocessing & Visualization:

Continue data preprocessing by handling any remaining

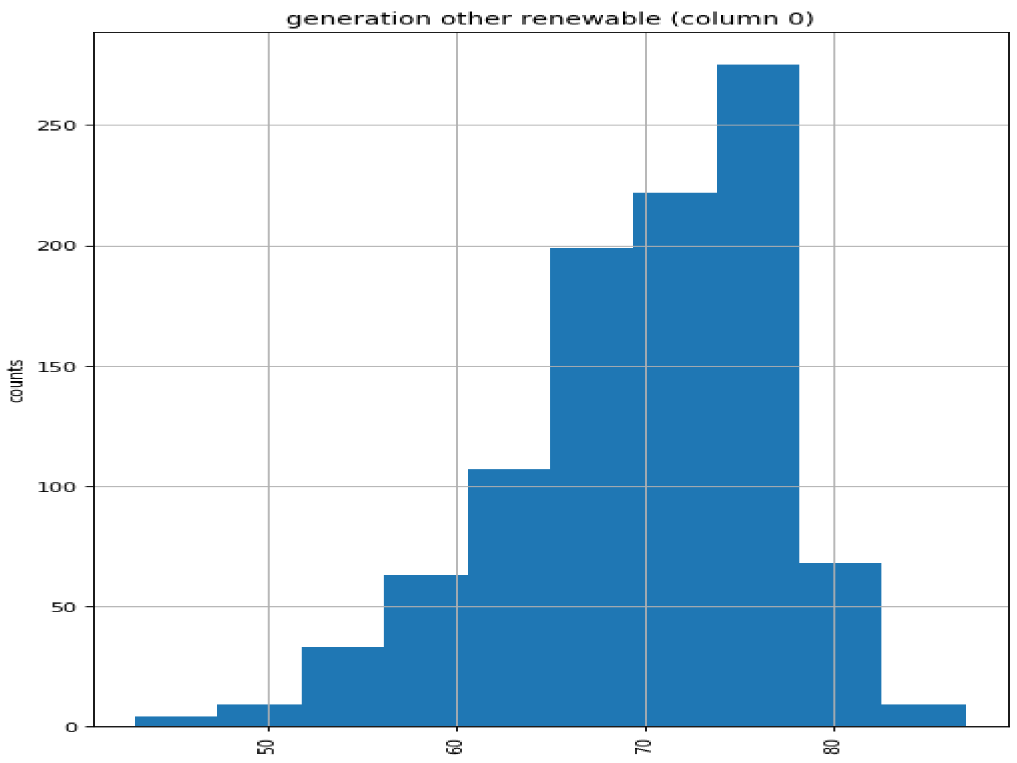
missing values or outliers based on insights from your data exploration.

```
#data visualization

#line plot for energy consumption over time

plt.figure(figsize=(12,6))

plt.plot(df.index,df["PJMW_MW"],label="Energy
consumption(PJMW_MW)")

plt.xlabel("Datetime")

plt.ylabel("Energy consumption(MW)")

plt.title("Energy consumption Over Time")

plt.grid()

plt.legend()

plt.show()
```

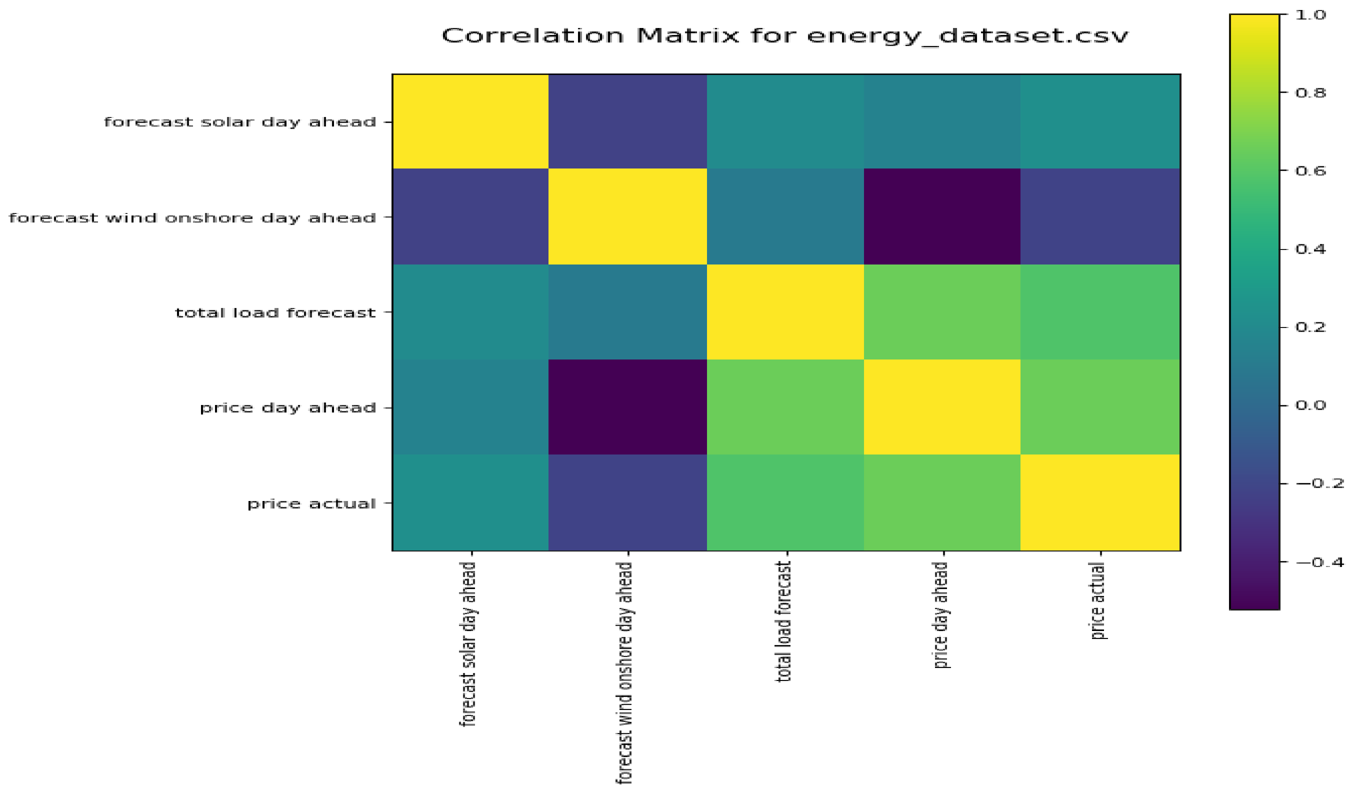
Program:

```
In [1]:
 plotPerColumnDistribution(df1, 10, 5)
```

Out [1]:



```
In [2]

PlotCorrelationMatrix(df1, 8)
```



In [3]:

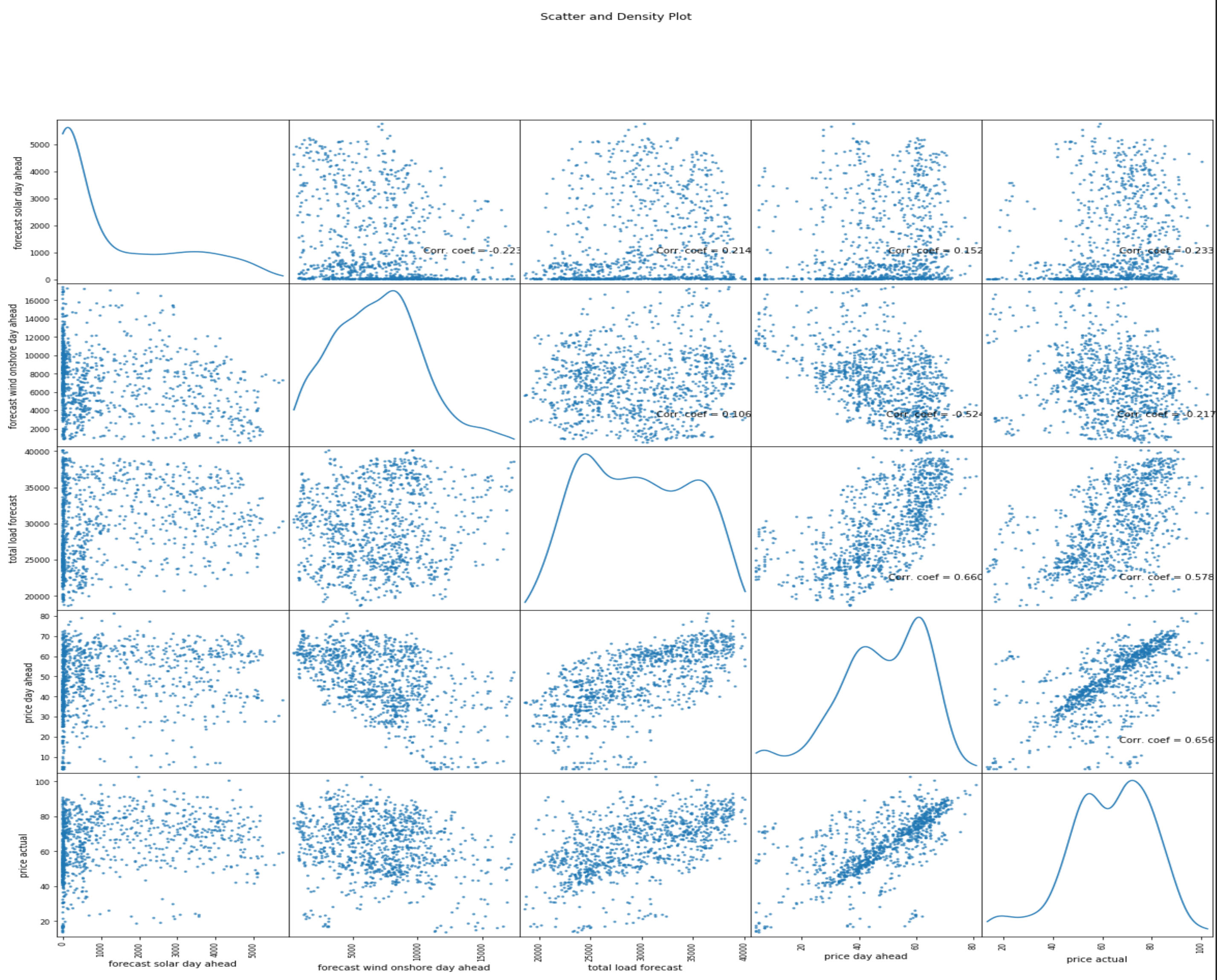
```
plotScatterMatrix(df1, 20, 10)
```

#Scatter plots are useful for visualizing the relationship between two continuous variables, such as energy consumption and time.

X-axis: Time (e.g., days, months, or years).
Y-axis: Energy consumption.

# Density plots, also known as kernel density plots, provide a smoothed representation of the data's distribution.

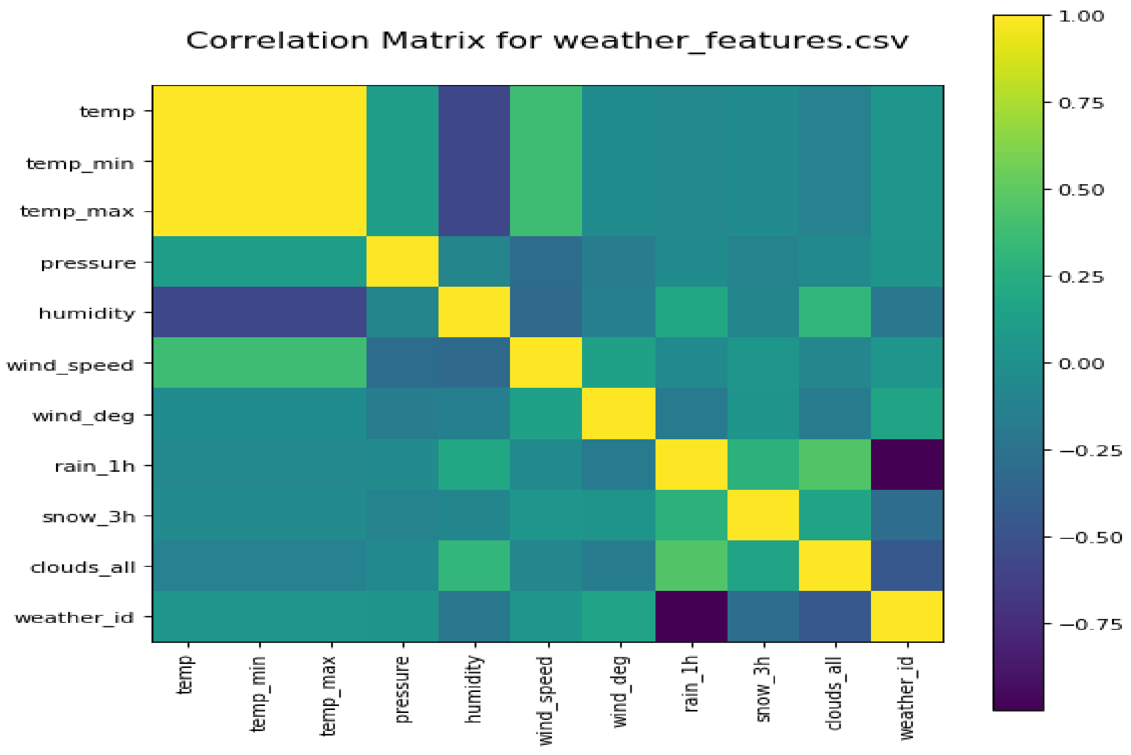
Out [3]:



In [4]:

```
plotCorrelationMatrix(df2, 8)
```

Out [4]:



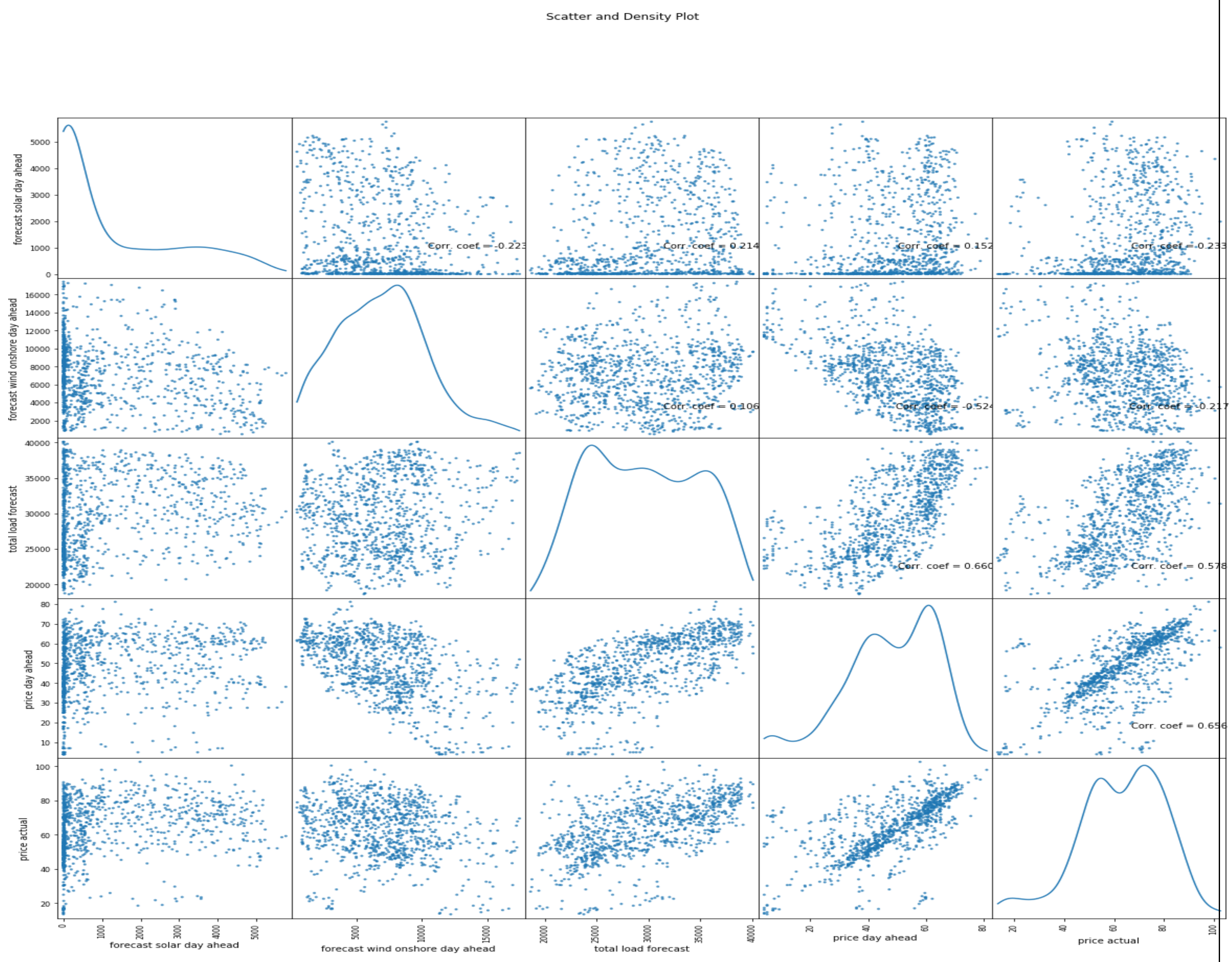
In [5]:

```
plotScatterMatrix(df1, 20, 10)
```

X-axis: Time (e.g., days, months, or years).

Y-axis: Energy consumption

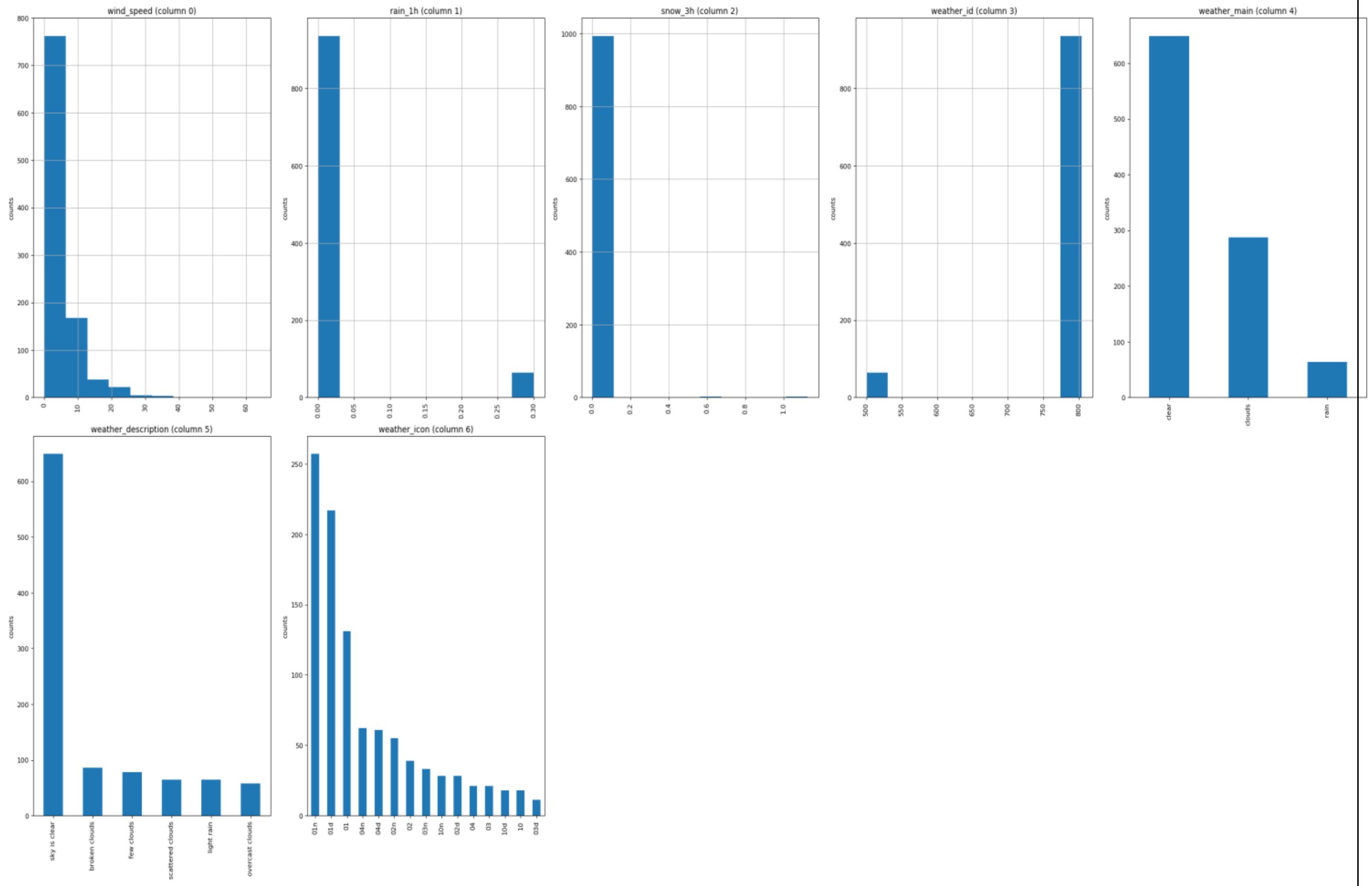
Out [5]:



In [6]:

```
plotPerColumnDistribution(df2, 10, 5)
```

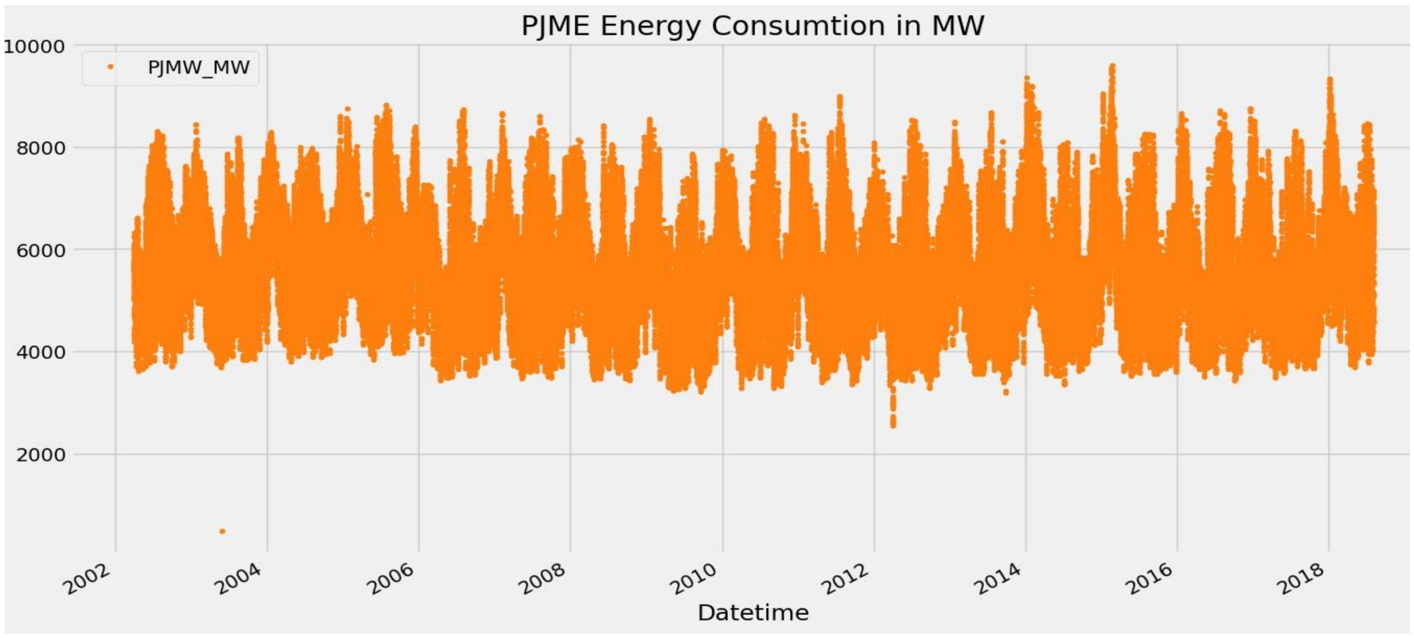
Out [6]:



In [7]:

```
energy_df.index = pd.to_datetime(energy_df.index)

energy_df.plot(style='.', figsize=(15, 8), color=color_pal[1],
          title='PJME Energy Consumtion in MW');
```

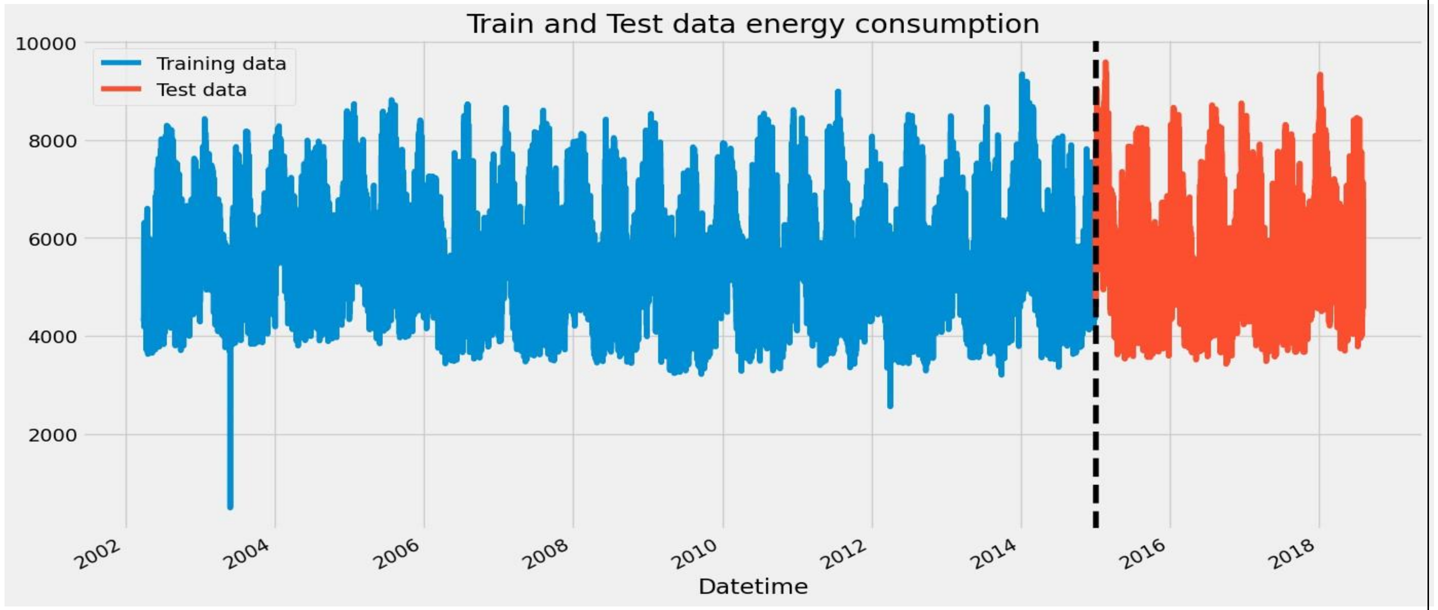
Out [7]:



In [8]:

```
 fig, ax = plt.subplots(figsize=(15,8))

train.plot(ax=ax, label='Training data')

test.plot(ax=ax, label='Test data')

ax.axvline('01-01-2015', color='black', ls='--')

ax.legend(['Training data', 'Test data'])

plt.title('Train and Test data energy consumption')

plt.show()
```

Out [8]:



In [9]:

```
energy_df.loc[(energy_df.index > '01-01-2010') & (energy_df.index < '01-08-2010')].plot(figsize=(15,8), title='Week of data')

plt.show()
```

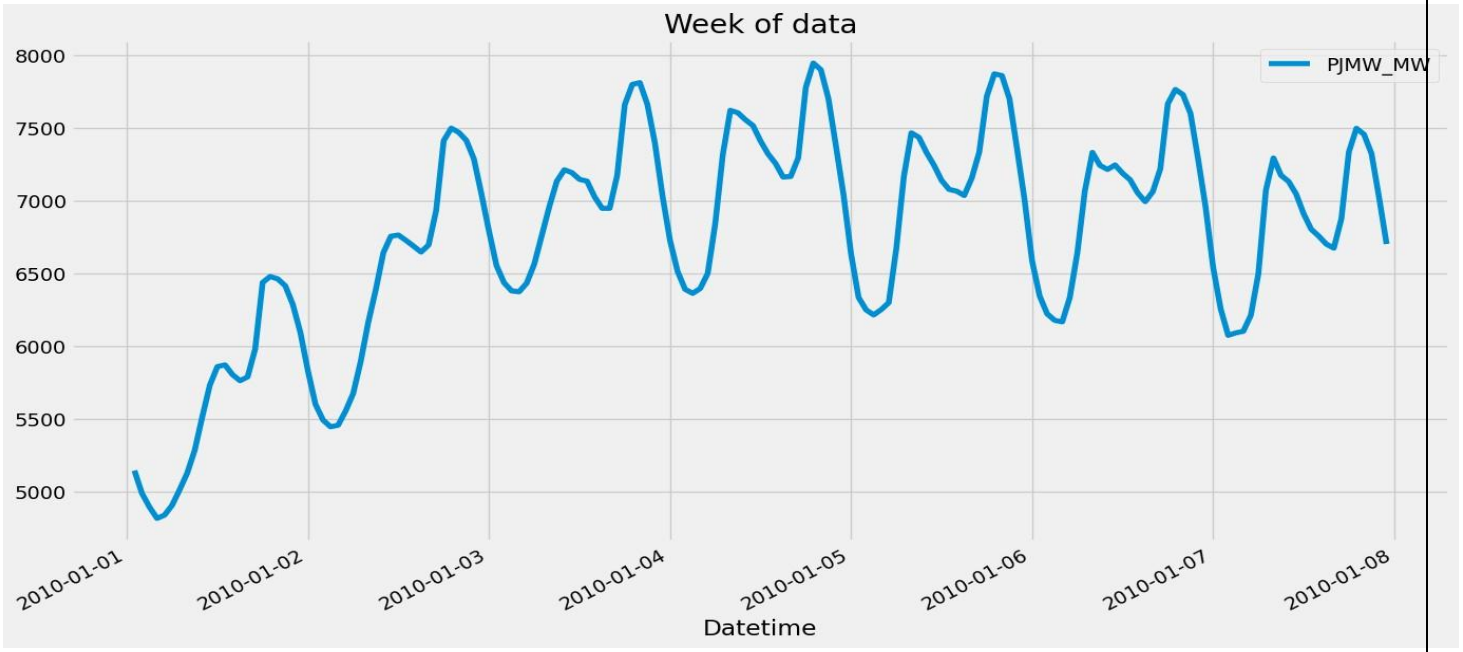
Hour of the day: Encode the hour as a categorical variable or as a numeric feature (0-23).

Day of the week: Encode the day of the week as a categorical variable (e.g., Monday, Tuesday, etc.).

Month of the year: Encode the month as a categorical variable (e.g., January, February, etc.).

Year: If you have data spanning multiple years, encode the year as a feature.

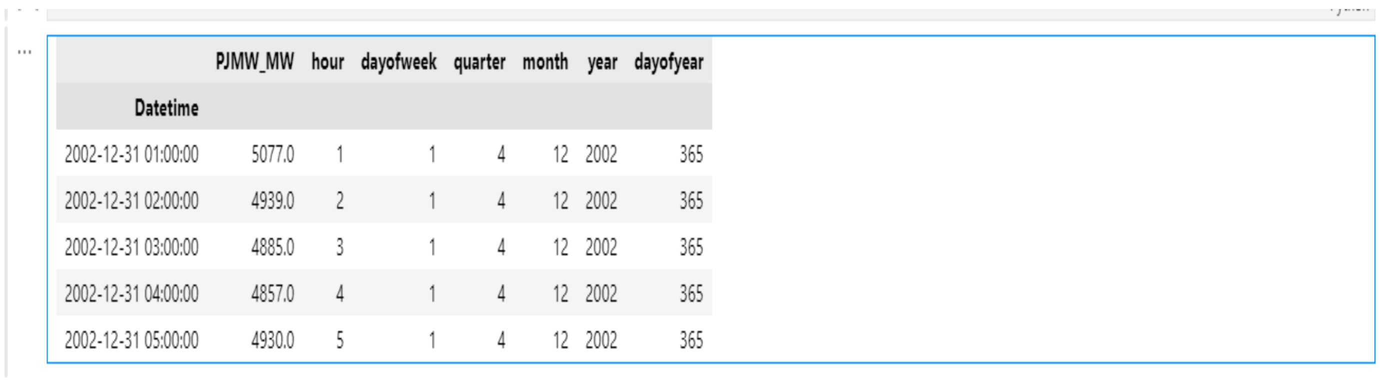
Out [9]:



In [10]:

energy_df = create_features(energy_df)

energy_df.



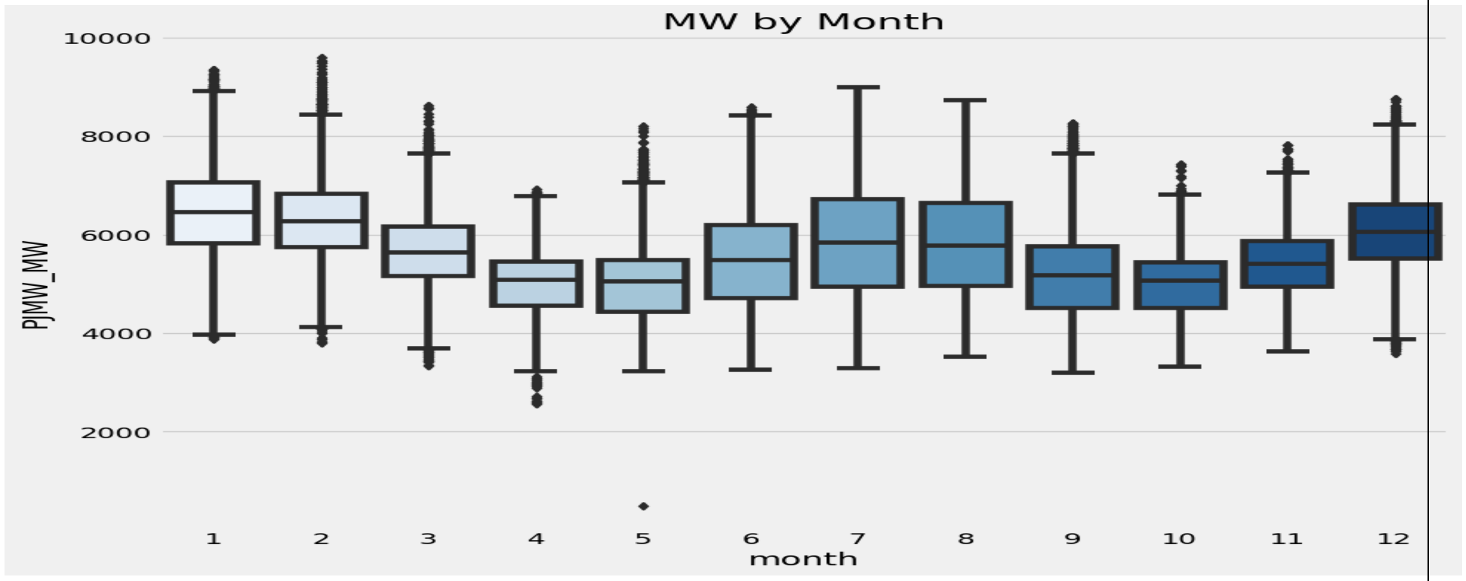
	PJMW_MW	hour	dayofweek	quarter	month	year	dayofyear
Datetime							
2002-12-31 01:00:00	5077.0	1	1	4	12	2002	365
2002-12-31 02:00:00	4939.0	2	1	4	12	2002	365
2002-12-31 03:00:00	4885.0	3	1	4	12	2002	365
2002-12-31 04:00:00	4857.0	4	1	4	12	2002	365
2002-12-31 05:00:00	4930.0	5	1	4	12	2002	365

In [11]:

fig, ax = plt.subplots(figsize=(10,8))

```
sns.boxplot(data=energy_df, x='month', y='PJMW_MW', palette='Blues')
ax.set_title('MW by Month')
plt.show()
```

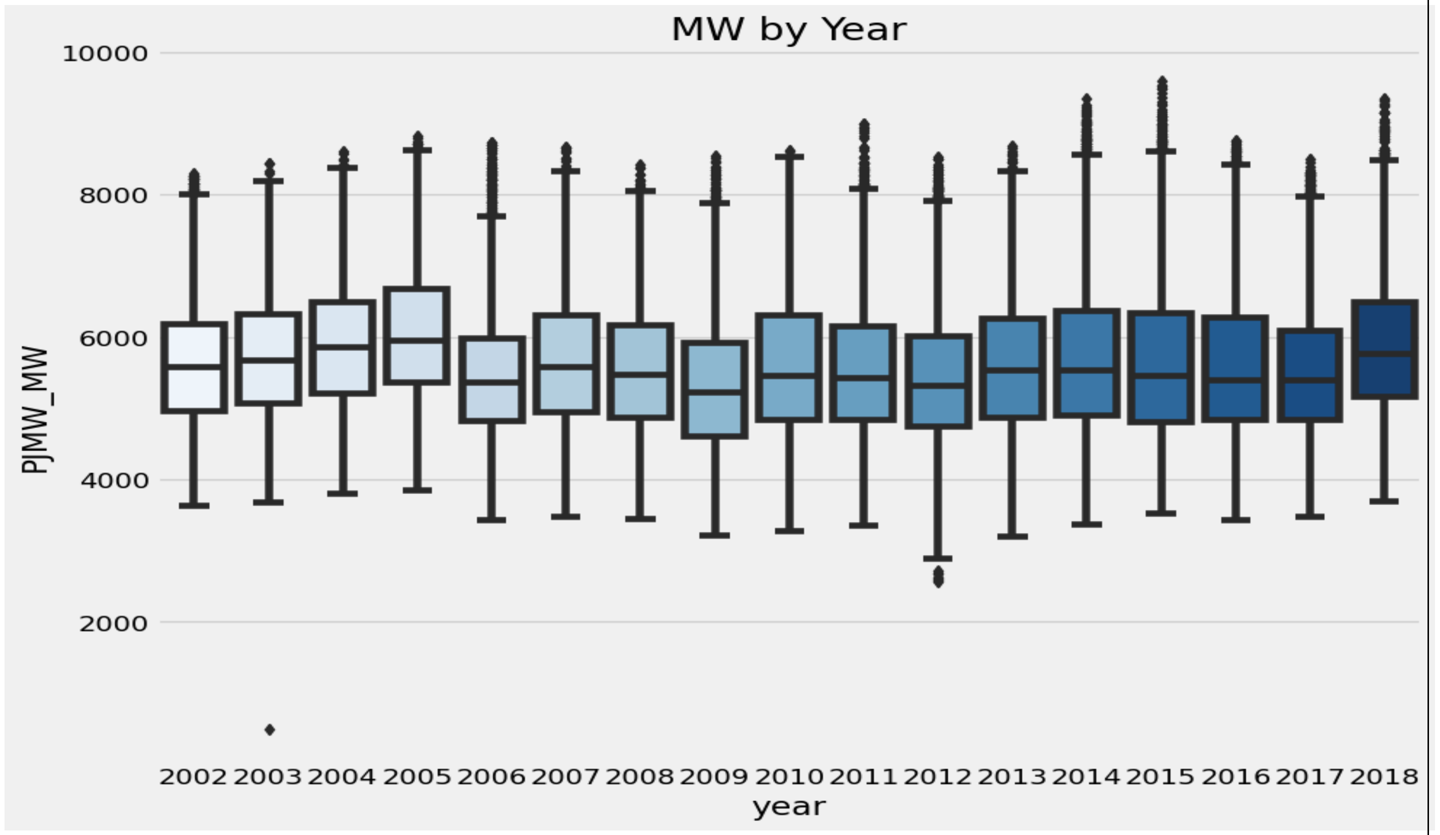
Out [11]:



In [12]:

```
fig, ax = plt.subplots(figsize=(10,8))
sns.boxplot(data=energy_df, x='year', y='PJMW_MW', palette='Blues')
ax.set_title('MW by Year')
plt.show()
```

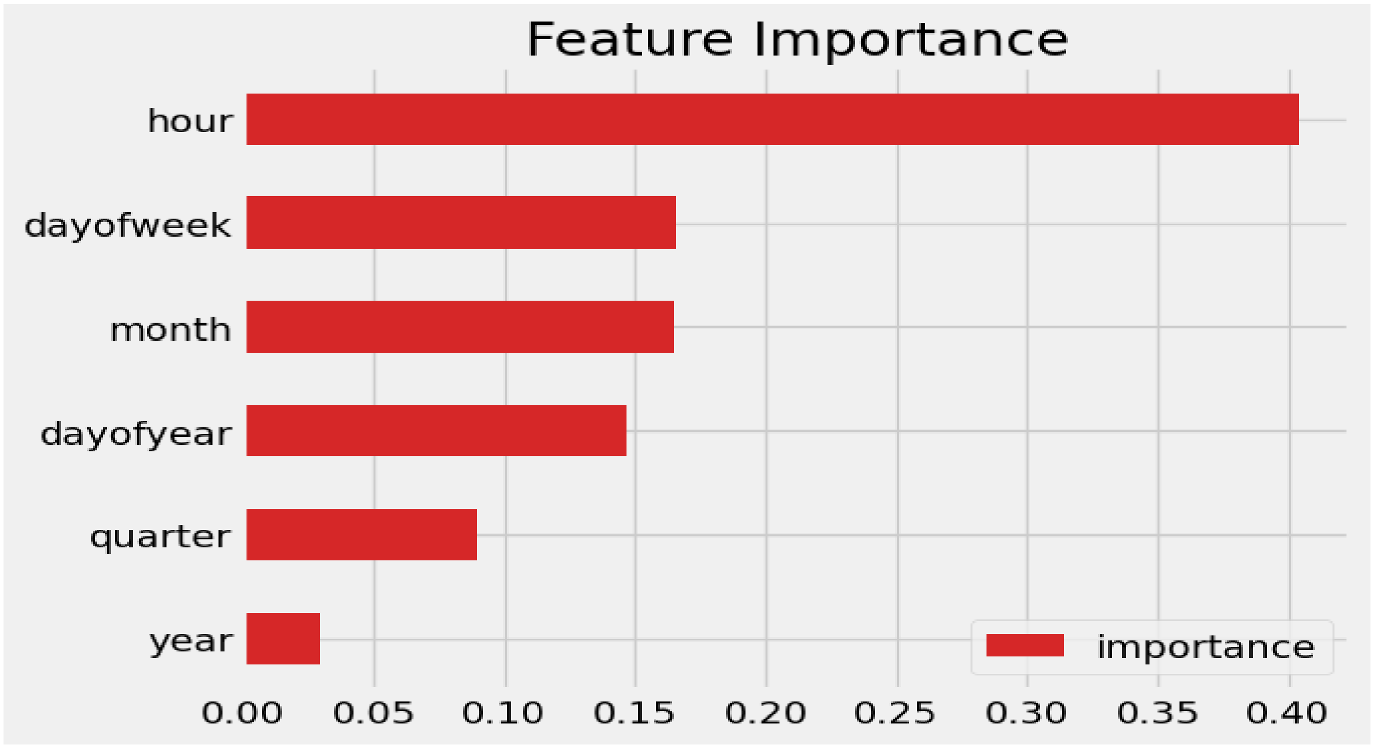
Out [12]:



In [13]:

```
feature_imp.sort_values('importance').plot(kind='barh', title='Feature
Importance',   color=color_pal[3])
plt.show()
```

Out [13]:



Model Training:

XGBoost is good and reliable model for regression and time series analysis as well. Also, for the metrics, we'll use mean squared error.

Prepare the data:

In [2]:

```
# preprocessing
train = create_features(train)
test = create_features(test)

features = ['dayofyear', 'hour', 'dayofweek', 'quarter', 'month',
'year']
target = 'PJME_MW'

X_train = train[features]
```

```
y_train = train[target]

X_test = test[features]
y_test = test[target]
```

Build the model:

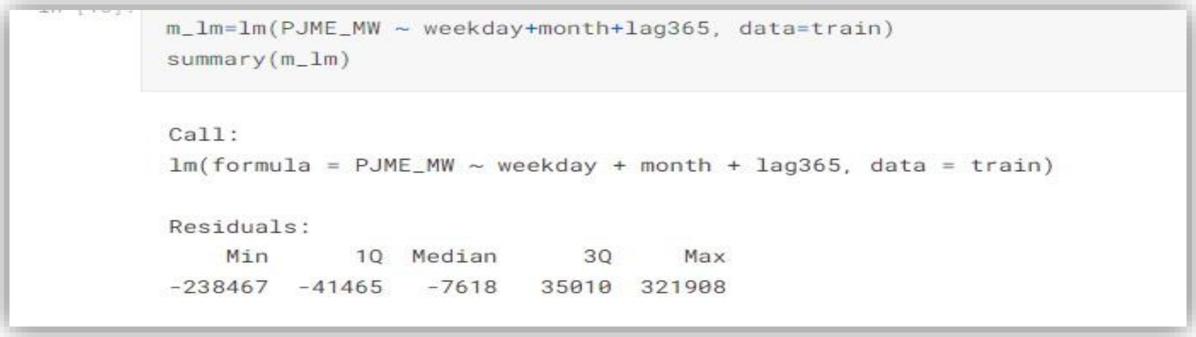
In [3]:

```
linkcode
import xgboost as xgb
from sklearn.metrics import mean_squared_error

reg = xgb.XGBRegressor(base_score=0.5, booster='gbtree',
       n_estimators=1000,
       early_stopping_rounds=50,
       objective='reg:linear',
       max_depth=3,
       learning_rate=0.01)
reg.fit(X_train, y_train,
        eval_set=[(X_train, y_train), (X_test, y_test)],
        verbose=100)
```

Linear Regression:

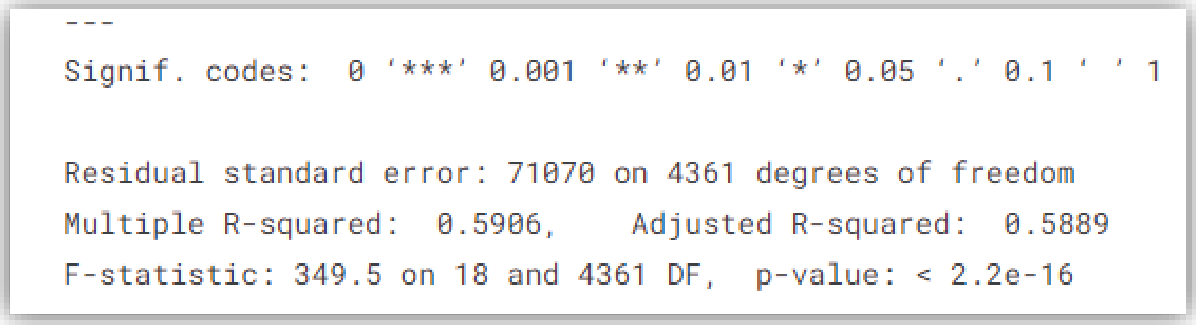
Linear regression is a simple and widely used technique. You would use it to model the relationship between independent variables (such as temperature, time of day, day of the week) and the dependent variable (energy consumption) as a linear equation.

In [4]:



```
m_lm=lm(PJME_MW ~ weekday+month+lag365, data=train)
summary(m_lm)

Call:
lm(formula = PJME_MW ~ weekday + month + lag365, data = train)

Residuals:
    Min      1Q  Median      3Q     Max
-238467  -41465   -7618   35010  321908
```

Out [4]:



```
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 71070 on 4361 degrees of freedom
Multiple R-squared:  0.5906,    Adjusted R-squared:  0.5889
F-statistic: 349.5 on 18 and 4361 DF,  p-value: < 2.2e-16
```

```
lr = LinearRegression() lr.fit(X_train, y_train)
```

Ridge Regression:

Ridge regression is used to select the variables which are related to energy consumption significantly. Since we got five variables, we could select some significant variables, which can improve the accuracy of prediction.

```
ridge = Ridge(alpha=1.0)
ridge.fit(X_train, y_train)
```

Lasso Regression:

group lasso was first utilized to identify the key drivers of energy consumption from both physical and human factors, and then quantile regression was applied to investigate the effects of these drivers at different levels of energy consumption

```
lasso = Lasso(alpha=1.0)
lasso.fit(X_train, y_train)
```

Decision Tree Regressor:

The decision tree and neural network models appear to be viable alternatives to the stepwise regression model in understanding energy consumption patterns and predicting energy consumption levels.

```
dt = DecisionTreeRegressor(max_depth=10)
dt.fit(X_train, y_train)
```

K Nearest Neighbors:-

KNN, or K-nearest Neighbor, is a supervised machine learning technique for classification and regression problems. In KNN regression, the K value you choose is crucial since it has a big impact on how well the algorithm works. The model may overfit if K is too small because it may be very sensitive to data noise. However, if K is too high, the model can be oversimplified and fail to recognize the underlying trends in the data.

XGBoost Regressor:

The XGBRegressor in Python is the regression-specific implementation of XGBoost and is used for regression problems where the intent is to predict continuous numerical values. Objective is a required parameter representing objective function to use for regression.

```
reg = xgb.XGBRegressor(n_estimators=1000)
reg.fit(X_train, y_train,
        eval_set=[(X_train, y_train), (X_test, y_test)],
        early_stopping_rounds=50,
       verbose=False) # Change verbose to True if you want to see it train
```

```
XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
       colsample_bytree=1, gamma=0, learning_rate=0.1, max_delta_step=0,
       max_depth=3, min_child_weight=1, missing=None, n_estimators=1000,
       n_jobs=1, nthread=None, objective='reg:linear', random_state=0,
       reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,silent=True, subsample=1)
```

Long Short Term Memory (LSTM):-

An artificial neural network with Long Short-Term Memory (LSTM) is us ed in deep learning and artificial intelligence. Because LSTM contains fe edback connections, they differ from traditional feed forward neural ne tworks. In addition to analysing single data points (like photos), such as audio or video, this kind of RNN can also evaluate whole data s equences Networked, unsegmented handwriting identification, speech recognition, machine translation, robot control, video gaming, and heal thcare are a few examples of LSTM applications.

- ❖ Model evaluation is the process of assessing the performance of a machine learning model on unseen data.

- ❖ This is important to ensure that the model will generalize well to new data. There are a number of different metrics that can be use d to evaluate the performance of a house price prediction model.

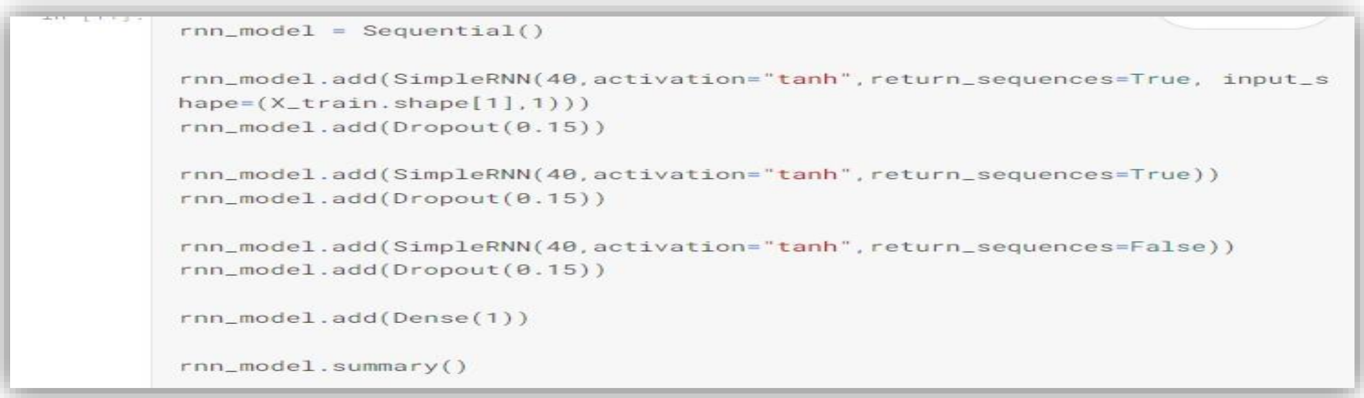
Mean Absolute Error (MAE): The average absolute difference betwee n the predicted and actual values.

Mean Squared Error (MSE): The average squared difference between t he predicted and actual values.
Root Mean Squared Error (RMSE): The square root of the MSE, which is in the same unit as the target variable.

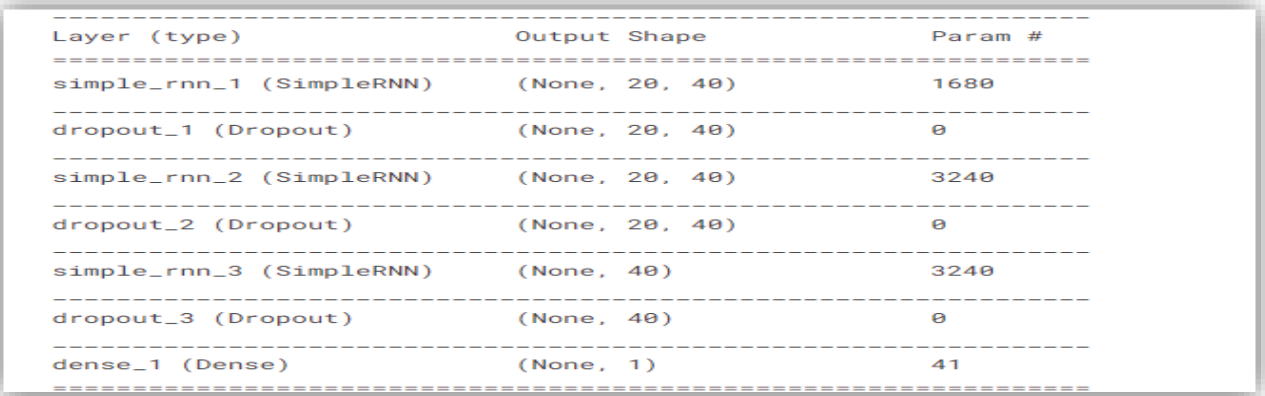
Mean Absolute Percentage Error (MAPE): Measures the percentage difference between predicted and actual values.

R-squared (R2) or coefficient of determination: Measures the propo rtion of the variance in the dependent variable that is predictable from the independent variables. Forecasting accuracy metrics like MDA (Mea n Directional Accuracy) for directional accuracy.

Simple RNN Model:



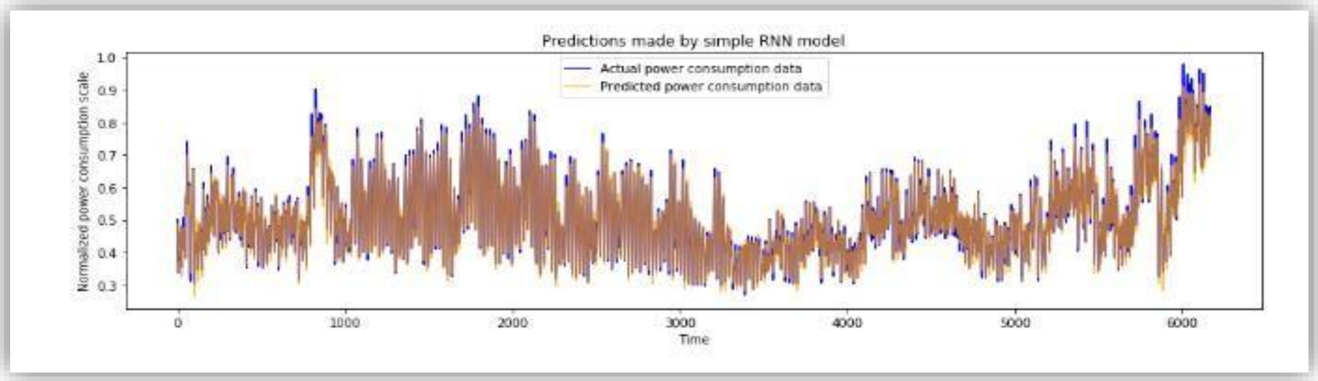
```
rnn_model = Sequential()

rnn_model.add(SimpleRNN(40,activation="tanh",return_sequences=True, input_s
hape=(X_train.shape[1],1)))
rnn_model.add(Dropout(0.15))

rnn_model.add(SimpleRNN(40,activation="tanh",return_sequences=True))
rnn_model.add(Dropout(0.15))

rnn_model.add(SimpleRNN(40,activation="tanh",return_sequences=False))
rnn_model.add(Dropout(0.15))

rnn_model.add(Dense(1))

rnn_model.summary()
```



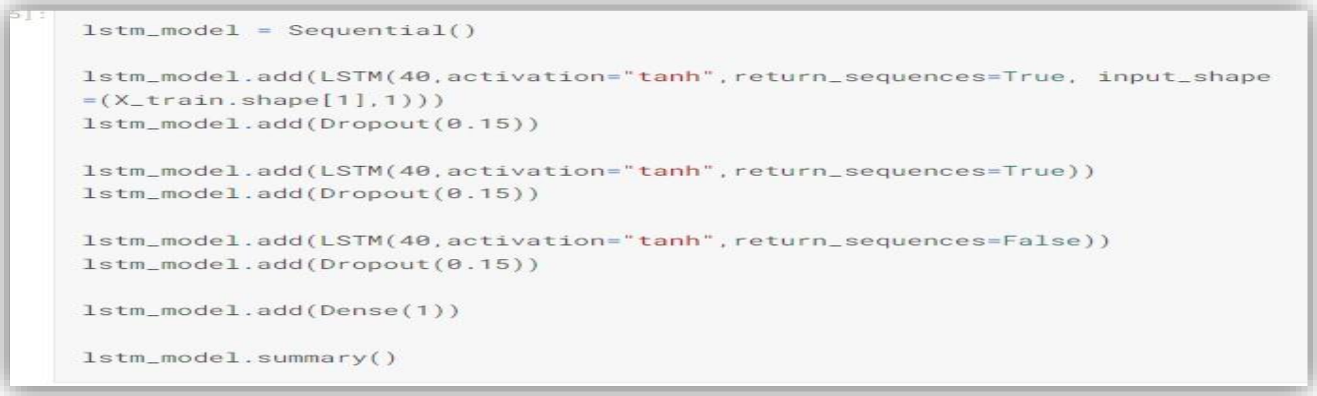
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
simple_rnn_1 (SimpleRNN)     (None, 20, 40)            1680
_________________________________________________________________
dropout_1 (Dropout)          (None, 20, 40)            0
_________________________________________________________________
simple_rnn_2 (SimpleRNN)     (None, 20, 40)            3240
_________________________________________________________________
dropout_2 (Dropout)          (None, 20, 40)            0
_________________________________________________________________
simple_rnn_3 (SimpleRNN)     (None, 40)                3240
_________________________________________________________________
dropout_3 (Dropout)          (None, 40)                0
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 41
=================================================================
```

```
Total params: 8,201
Trainable params: 8,201
Non-trainable params: 0
```

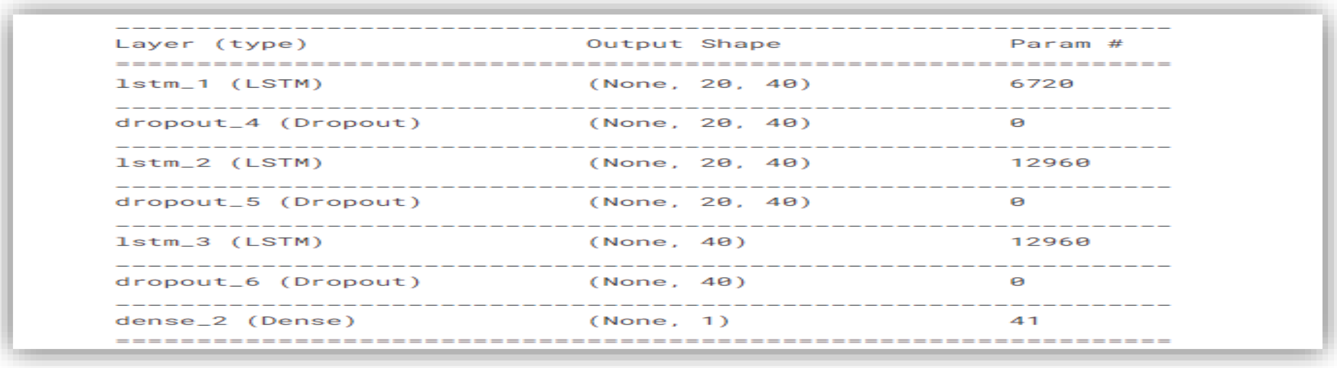
Predictions made by RNN model



Simple LSTM Model:

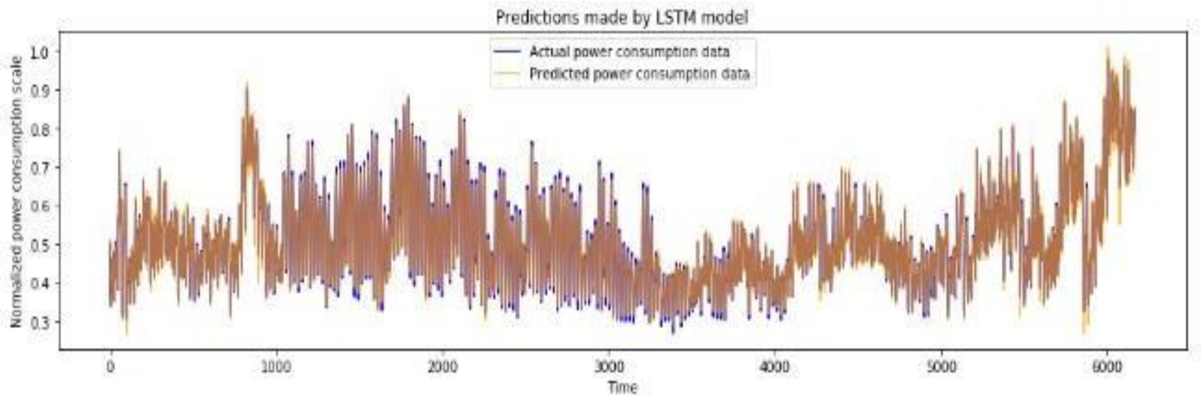


```
lstm_model = Sequential()

lstm_model.add(LSTM(40,activation="tanh",return_sequences=True, input_shape
=(X_train.shape[1],1)))
lstm_model.add(Dropout(0.15))

lstm_model.add(LSTM(40,activation="tanh",return_sequences=True))
lstm_model.add(Dropout(0.15))

lstm_model.add(LSTM(40,activation="tanh",return_sequences=False))
lstm_model.add(Dropout(0.15))

lstm_model.add(Dense(1))

lstm_model.summary()
```



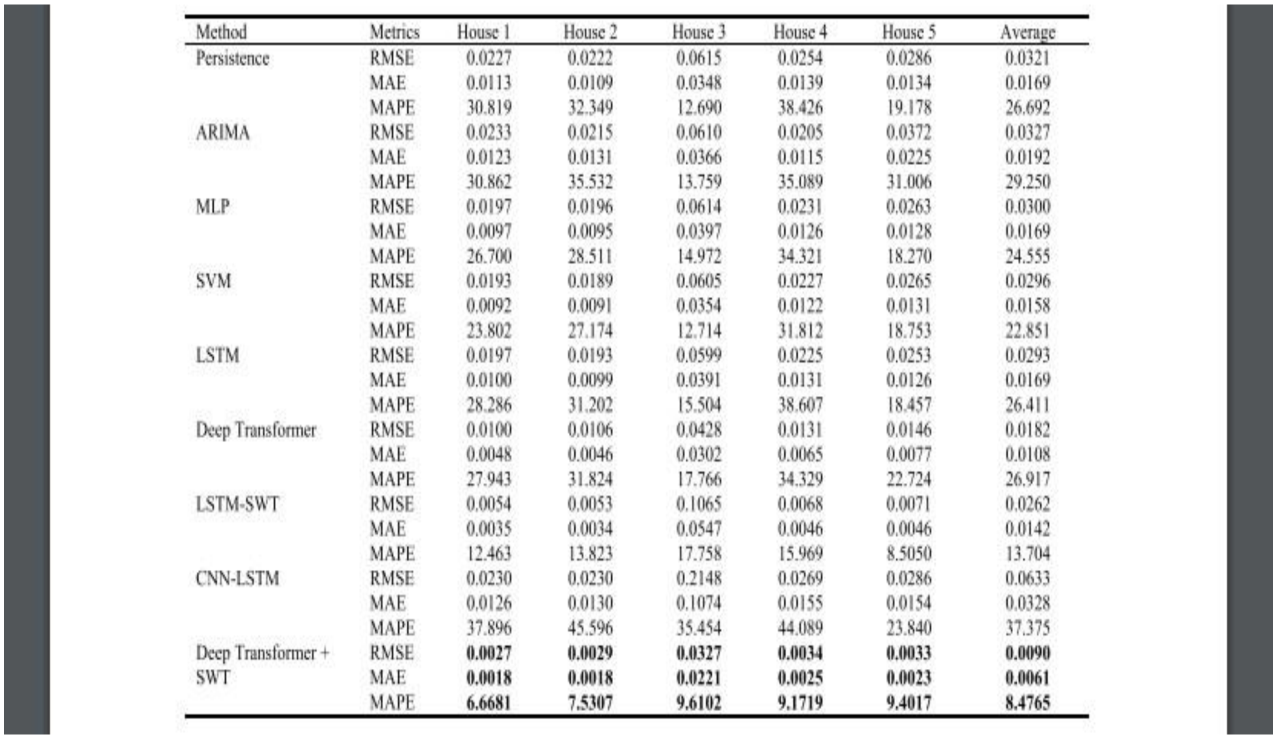
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
lstm_1 (LSTM)                (None, 20, 40)            6720      
_________________________________________________________________
dropout_4 (Dropout)          (None, 20, 40)            0         
_________________________________________________________________
lstm_2 (LSTM)                (None, 20, 40)            12960     
_________________________________________________________________
dropout_5 (Dropout)          (None, 20, 40)            0         
_________________________________________________________________
lstm_3 (LSTM)                (None, 40)                12960     
_________________________________________________________________
dropout_6 (Dropout)          (None, 40)                0         
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 41        
=================================================================
```

```
Total params: 32,681
 Trainable params: 32,681
 Non-trainable params: 0
```



<u>**Model Comparison**</u>**:**

Comparing different models for hourly energy consumption prediction is an essential step in selecting the most suitable approach for your specific needs



Method	Metrics	House 1	House 2	House 3	House 4	House 5	Average
Persistence	RMSE	0.0227	0.0222	0.0615	0.0254	0.0286	0.0321
	MAE	0.0113	0.0109	0.0348	0.0139	0.0134	0.0169
	MAPE	30.819	32.349	12.690	38.426	19.178	26.692
ARIMA	RMSE	0.0233	0.0215	0.0610	0.0205	0.0372	0.0327
	MAE	0.0123	0.0131	0.0366	0.0115	0.0225	0.0192
	MAPE	30.862	35.532	13.759	35.089	31.006	29.250
MLP	RMSE	0.0197	0.0196	0.0614	0.0231	0.0263	0.0300
	MAE	0.0097	0.0095	0.0397	0.0126	0.0128	0.0169
	MAPE	26.700	28.511	14.972	34.321	18.270	24.555
SVM	RMSE	0.0193	0.0189	0.0605	0.0227	0.0265	0.0296
	MAE	0.0092	0.0091	0.0354	0.0122	0.0131	0.0158
	MAPE	23.802	27.174	12.714	31.812	18.753	22.851
LSTM	RMSE	0.0197	0.0193	0.0599	0.0225	0.0253	0.0293
	MAE	0.0100	0.0099	0.0391	0.0131	0.0126	0.0169
	MAPE	28.286	31.202	15.504	38.607	18.457	26.411
Deep Transformer	RMSE	0.0100	0.0106	0.0428	0.0131	0.0146	0.0182
	MAE	0.0048	0.0046	0.0302	0.0065	0.0077	0.0108
	MAPE	27.943	31.824	17.766	34.329	22.724	26.917
LSTM-SWT	RMSE	0.0054	0.0053	0.1065	0.0068	0.0071	0.0262
	MAE	0.0035	0.0034	0.0547	0.0046	0.0046	0.0142
	MAPE	12.463	13.823	17.758	15.969	8.5050	13.704
CNN-LSTM	RMSE	0.0230	0.0230	0.2148	0.0269	0.0286	0.0633
	MAE	0.0126	0.0130	0.1074	0.0155	0.0154	0.0328
	MAPE	37.896	45.596	35.454	44.089	23.840	37.375
Deep Transformer +	RMSE	**0.0027**	**0.0029**	**0.0327**	**0.0034**	**0.0033**	**0.0090**
SWT	MAE	**0.0018**	**0.0018**	**0.0221**	**0.0025**	**0.0023**	**0.0061**
	MAPE	**6.6681**	**7.5307**	**9.6102**	**9.1719**	**9.4017**	**8.4765**

<u>Feature Engineering:-</u>

<u>Time-based Features:</u>

Hour of the day: Encode the hour as a categorical variable or as a numeric feature (0-23).

Day of the week: Encode the day of the week as a categorical variable (e.g., Monday, Tuesday, etc.).

Month of the year: Encode the month as a categorical variable (e.g., January, February, etc.).

Year: If you have data spanning multiple years, encode the year as a feature.

Lagged Values:

Include lagged values of the target variable (hourly energy consumption) as features. For example, the energy consumption in the previous hour or on the same hour of the previous day.

Seasonal and Holiday Indicators:

Create binary indicators for holidays, special events, or known seasonality in energy consumption. These can be useful for capturing irregular patterns.

Weather Data:

Include weather-related features such as temperature, humidity, wind speed, and precipitation. Weather conditions often influence energy consumption.

Daylight Information:

Consider incorporating daylight information, such as sunrise and sunset times, to account for variations in energy consumption due to daylight hours.

Special Events:

If there are known special events, such as sports events, holidays, or local festivities, create binary indicators for these events.

Time since Last Event:

Create features that capture the time elapsed since the last significant event or maintenance activity.

Economic Indicators:

Incorporate economic indicators such as GDP, unemployment rates, and energy prices, as they can impact energy consumption.

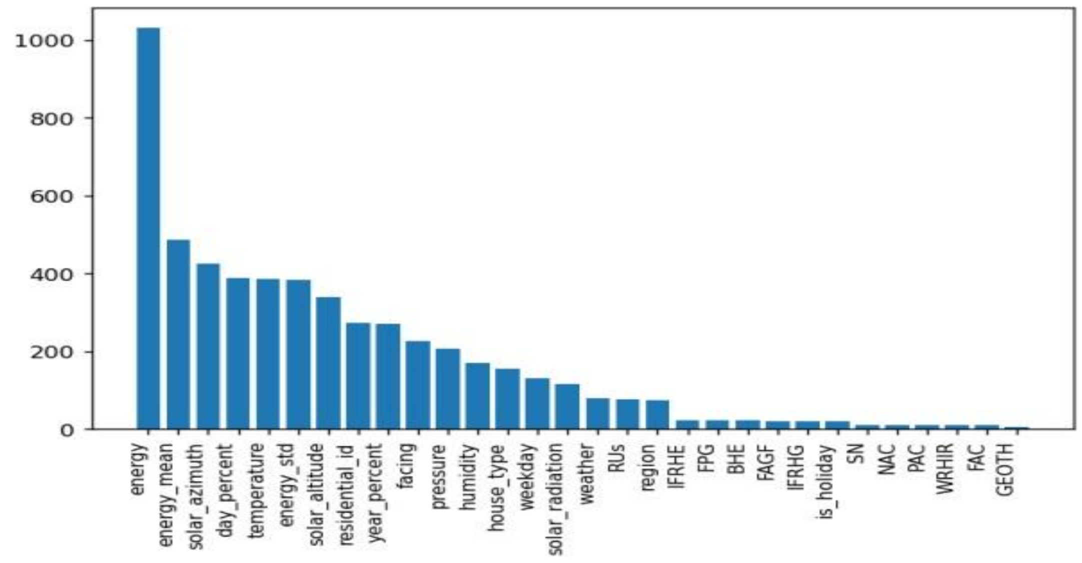
Time Series Decomposition Components:

Use decomposition techniques like seasonal decomposition to extract trend, seasonal, and residual components and use them as features.

External Factors:

Consider including external factors like population growth, urbanization rates, and changes in energy policies or regulations that can impact energy consumption.

Use a variety of feature engineering techniques:



Feature engineering is the process of transforming raw data into features that are more informative and predictive for machine learning models. By using a variety of feature engineering techniques, you can create a set of features that will help your model to predict house energy consumption more accurately.

Use cross-validation

Cross-validation is a technique for evaluating the performance of a machine learning model on unseen data. It is important to use cross validation to evaluate the performance of your model during the training process. This will help you to avoid over fitting and to ensure that your model will generalize well to new data.

Use ensemble methods

Ensemble methods are machine learning methods that combine the predictions of multiple models to produce a more accurate prediction.

Use a holdout test set

A holdout test set is a set of data that is not used to train or evaluate the model during the training process. This data is used to evaluate the performance of the model on unseen data after the training process is complete.

Analyze the model's predictions

Once you have evaluated the performance of the model, you can analyze the model's predictions to identify any patterns or biases. This will help you to understand the strengths and weaknesses.

ADVANTAGES:

Improved Accuracy and Predictability: AI models can provide more accurate and precise predictions of hourly energy consumption patterns. This helps utilities and consumers better understand their energy needs and make informed decisions.

Energy Efficiency: AI can identify patterns and anomalies in energy consumption data, allowing for the optimization of energy usage. This can lead to reduced energy waste and cost savings.

Real-time Insights: AI can process data in real-time, providing immediate feedback on energy consumption trends. This enables quick responses to changes in demand or supply.

Demand Response: AI can be used to implement demand response programs that automatically adjust energy consumption in response to grid conditions or pricing. This helps in grid stabilization and reduces the need for additional power generation during peak periods.

Anomaly Detection: AI models can detect abnormal energy consumption patterns, which can be indicative of equipment malfunction or security breaches in energy systems. This early detection can prevent costly breakdowns or security incidents.

Data-Driven Decision-Making: AI enables data-driven decision-making, allowing energy consumers and utilities to make informed choices about energy usage, investments in energy-efficient technologies, and sustainability efforts.

Optimization of Renewable Energy Integration: AI can optimize the integration of renewable energy sources like solar and wind into the grid by forecasting generation and adjusting consumption accordingly.

DISADVANTAGES:

Data Quality and Quantity: AI models require high-quality and abundant data to perform effectively. Inaccurate or insufficient data can lead to unreliable predictions.

Complexity and Expertise: Implementing AI solutions can be complex and requires expertise in machine learning and data science. Organizations may need to invest in training or hire specialists.

Computational Resources: Training and running AI models can be computationally intensive, requiring powerful hardware and cloud resources. This can be costly.

Privacy Concerns: Analyzing hourly energy consumption data could raise privacy concerns, especially when individual or sensitive data is involved. Care must be taken to protect privacy and comply with regulations.

Initial Costs: There may be significant upfront costs associated with implementing AI solutions, such as the purchase of sensors, meters, and software.

Model Interpretability: Some AI models are highly complex and may lack interpretability. Understanding how and why the model makes predictions can be challenging.

Maintenance and Updates: AI models require regular maintenance and updates to remain accurate and relevant. This can be an ongoing operational cost.

Regulatory and Ethical Considerations: The use of AI in energy consumption may be subject to regulatory and ethical considerations, such as fair use of algorithms, data protection, and transparency.

BENEFITS:

Enhanced Predictive Accuracy: AI models can analyze historical data to make highly accurate predictions of hourly energy consumption. This helps utilities and consumers plan for future energy needs more effectively.

Optimized Energy Usage: AI can identify consumption patterns and offer recommendations for optimizing energy usage. This can lead to energy cost savings and a reduced environmental footprint.

Real-time Insights: AI can process data in real-time, providing immediate feedback on energy consumption trends. This allows for quick responses to fluctuations in demand and supply, enabling grid stability and efficient energy management.

Demand Response: AI can enable automated demand response systems that adjust energy consumption based on real-time grid conditions, pricing, and renewable energy availability. This results in load balancing, reduced peak demand, and potential financial incentives for consumers.

Anomaly Detection: AI models can quickly detect anomalies or irregular consumption patterns, indicating equipment malfunctions, security breaches, or energy theft. Early detection can prevent costly breakdowns and enhance security.

Data-driven Decision Making: AI empowers utilities, businesses, and consumers to make data-driven decisions regarding energy usage, infrastructure investments, and sustainability initiatives. It provides insights into when, where, and how energy is consumed.

CONCLUSION:-

- ✓ Hourly energy consumption data analysis is crucial for understanding patterns and making informed decisions about energy management.
- ✓ The conclusion is patterns can help identify areas where energy efficiency measures can be implemented, reducing energy waste during low demand periods.
- ✓ The future work of this project is this project is a dynamic field with significant potential for improving energy efficiency, reducing costs.
- ✓ In conclusion, the development phase for energy consumption prediction is a crucial part of the process.
- ✓ It involves careful data preprocessing, feature engineering, model selection, and thorough evaluation.