PROJECT TITLE: MEASURE ENERGY CONSUMPTION

PHASE 5: PROJECT DOCUMENTATION & SUBMISSION

INTRODUCTION:

The project aims to develop a comprehensive Energy Consumption Monitoring System that allows individuals and organizations to effectively measure, analyze, and manage their energy usage. The primary objectives of this project are as follows:

1. DATA COLLECTION: Gather data from various energy sources, including electricity, gas, and water meters, to provide a holistic view of energy consumption.

2. REAL-TIME MONITORING: Enable real-time tracking of energy usage to help users identify patterns and anomalies.

3. HISTORICAL DATA STORAGE: Store historical energy consumption data for trend analysis and comparison.

4. USER-FRIENDLY INTERFACE: Design an intuitive and user-friendly interface accessible via web or mobile applications for easy interaction.

5. ENERGY ANALYTICS: Implement advanced analytics to provide insights into energy consumption patterns, peak usage times, and potential cost-saving opportunities.

6. ALERTING SYSTEM: Set up automated alerts for unusual energy spikes or deviations from expected consumption patterns.

7. ENERGY EFFICIENCY RECOMMENDATIONS: Offer personalized recommendations to users on how to reduce energy consumption and lower costs.

8. REPORTING: Generate detailed reports and visualizations to help users make informed decisions about energy management.

9. INTEGRATION: Ensure compatibility with various energy monitoring devices, such as smart meters, sensors, and IoT devices.

10. SECURITY: Implement robust security measures to protect sensitive energy data and user privacy.

11. SCALABILITY: Design the system to accommodate growth in the number of monitored devices and users.

12. COMPLIANCE: Ensure that the system complies with relevant energy efficiency and data privacy regulations.

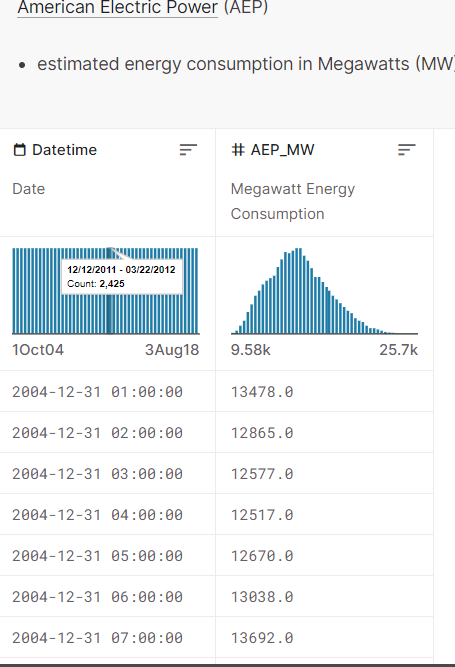
By addressing these objectives, the Energy Consumption Monitoring System will provide individuals and organizations with the tools they need to optimize their energy usage, reduce costs, and contribute to a more sustainable future.

DATA SOURCE:

A good data source for measure energy consumption for machine learning should be accurate, complete and accessible.

DATASET LINK: <https://www.kaggle.com/datasets/robikscube/hourly-energy-consumption>

GIVEN DATASET:



Here's a list of tools and software commonly used in the process:

1. 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.

2. Integrated Development Environment (IDE):

- Choose an IDE for coding and running machine learning experiments. Some popular options include Jupyter Notebook, Google Colab, or traditional IDEs like PyCharm .

3. Machine Learning Libraries:

- You'll need various machine learning libraries, including:

- scikit-learn for building and evaluating machine learning models.

- TensorFlow or PyTorch for deep learning, if needed.

- XGBoost, LightGBM, or CatBoost for gradient boosting models.

4. Data Visualization Tools:

- Tools like Matplotlib, Seaborn, or Plotly are essential for data exploration and visualization.

5. Data Preprocessing Tools:

- Libraries like pandas help with data cleaning, manipulation, and preprocessing.

6. Data Collection and Storage:

- Depending on your data source, you might need web scraping tools (e.g., BeautifulSoup or Scrapy) or databases (e.g., SQLite,PostgreSQL) for data storage.

7. Version Control:

- Version control systems like Git are valuable for tracking changes in your code and collaborating with others.

8. Notebooks and Documentation:

- Tools for documenting your work, such as Jupyter Notebooks or Markdown for creating README files and documentation.

9. Hyperparameter Tuning:

- Tools like GridSearchCV or RandomizedSearchCV from scikit-learn can help with hyperparameter tuning.

10. Web Development Tools (for Deployment):

- If you plan to create a web application for model deployment, knowledge of web development tools like Flask or Django for backend development, and HTML, CSS, and JavaScript for the front-end can be useful.

11. Cloud Services (for Scalability):

- For large-scale applications, cloud platforms like AWS, Google Cloud, or Azure can provide scalable computing and storage resources

12. External Data Sources (if applicable):

- Depending on your project's scope, you might require tools to access external data sources, such as APIs or data scraping tools.

13. Data Annotation and Labeling Tools (if applicable):

- For specialized projects, tools for data annotation and labeling may be necessary, such as Labelbox or Supervisely.

14. Geospatial Tools (for location-based features):

- If your dataset includes geospatial data, geospatial libraries like GeoPandas can be helpful.

1. DESIGN THINKING:

Design thinking is a problem-solving approach that emphasizes empathy, creativity, and iterative development. Here's a step-by-step guide on how to proceed with designing a system for measuring energy consumption using design thinking:

1. EMPATHIZE (Understand the User and Problem):

* Conduct interviews, surveys, and observations to understand the needs and pain points of users who want to measure energy consumption.
* Identify key user personas, such as homeowners, businesses, or utility companies, and their specific requirements.

2. DEFINE (Frame the Problem):

* Clearly define the problem you are solving, taking into account user insights. For example, "How might we help homeowners reduce their electricity bills by monitoring and managing energy consumption effectively?"

3. IDEATE (Generate Creative Solutions):

* Organize brainstorming sessions with a diverse team to generate ideas for measuring and managing energy consumption.
* Encourage wild and unconventional ideas, then narrow down to the most promising ones.

4. PROTOTYPE (Create a Prototype):

* Develop a low-fidelity prototype of the energy consumption monitoring system. This could be a paper sketch or a basic digital mockup.

Focus on the user interface, data visualization, and user interaction elements.

5. TEST (Gather Feedback):

* Test the prototype with a small group of users to gather feedback.
* Observe how users interact with the prototype and listen to their suggestions and concerns.

6. REFINE (Iterate and Improve):

* Based on user feedback, make iterative improvements to the prototype.
* Continue to refine the design, functionality, and user experience.

7. DEVELOP (Build the Solution):

* Begin the development of the energy consumption monitoring system based on the refined prototype.
* Implement the data collection, analytics, and user interface components.

8. TEST (Quality Assurance):

* Conduct extensive testing to ensure the system functions correctly and meets user expectations.
* Test for reliability, security, and performance.

9. LAUNCH (Release to Users):

* Deploy the system to a limited group of users as a pilot release to gather real-world feedback.
* Monitor system performance and user feedback during this phase.

10. EVALUATE (Assess Impact):

- Evaluate the impact of the system on energy consumption, cost savings, and user satisfaction.

- Collect data on how effectively the system achieves its goals.

11. ITERATE (Continuous Improvement):

- Use ongoing feedback and data to make continuous improvements to the system.

- Consider adding new features, optimizing algorithms, or expanding the user base.

Throughout the design thinking process, it's essential to keep the end-users at the center of your efforts, continuously gathering insights, and adapting the solution to their needs. This iterative approach ensures that the energy consumption monitoring system evolves and remains relevant to users over time.

2: INNOVATION

STEPS TO BE TAKEN TO EVALUATE THE ENERGY CONSUMPTION:

1. START

2. DEFINE OBJECTIVES

- Determine the purpose of the energy consumption evaluation.

- Decide on the specific metrics you want to measure.

3. IDENTIFY ENERGY SOURCES

- List all sources of energy consumption (e.g., lighting, HVAC, machinery, transportation).

4. DATA COLLECTION

a. Install Energy Meters

- Place energy meters at key points of consumption.

- Set up sub-meters for individual systems or equipment.

b. Collect Historical Utility Bills

- Gather past utility bills for reference.

5. DATA AGGREGATION

- Integrate all data into a central database or software system.

6. NORMALIZE DATA

- Adjust for external factors affecting consumption (e.g., weather, production).

7. ANALYSIS

a. Benchmarking

- Compare consumption data to industry benchmarks or similar facilities.

b. Create Energy Profiles

- Develop profiles for different systems to identify high consumption areas.

c. Calculate Energy Efficiency Ratios

- Compute ratios like energy per unit of production or occupancy.

8. IDENTIFY ANOMALIES

- Use data analysis to detect unusual consumption patterns.

9. ENERGY AUDITS

a. Walkthrough Audit

- Physically inspect the facility for energy-saving opportunities.

b. Detailed Energy Audit

- Conduct comprehensive analysis, including equipment testing and simulations.

10. IMPLEMENTATION

- Implement energy-saving measures identified in the audit.

- Continuously monitor the results of these changes.

11. CONTINUOUS MONITORING AND REPORTING

a. Regularly monitor energy consumption.

b. Generate reports tracking progress toward efficiency goals.

c. Share results and findings with stakeholders.

12. FEEDBACK AND IMPROVEMENT

a. Continuously collect and analyze data.

b. Adjust strategies and practices to achieve efficiency goals.

13. DOCUMENTATION

- Keep records of all data, audits, reports, and improvements.

14. REGULATORY COMPLIANCE

- Ensure compliance with energy regulations and reporting requirements.

15. TRAINING AND AWARENESS

- Educate employees and stakeholders on energy conservation.

16. BUDGETING

- Allocate resources for energy-saving projects.

17. REVIEW AND REFINEMENT

- Periodically review and refine the energy evaluation process.

18. END

3. BUILD LOADING AND PREPROCESSING THE DATASET:

DATA SOURCE:

A good data source for measure energy consumption for machine learning should be accurate, complete and accessible.

1. Loading the dataset:

 Loading the dataset using machine learning is the process of bringing the data into the machine learning environment so that it can be used to train and evaluate a model.

 The specific steps involved in loading the dataset will vary depending on the machine learning library or framework that is being used.However, there are some general steps that are common to most machine learning frameworks:

a.Identify the dataset:

The first step is to identify the dataset that you want to load. This may be stored in a local file, in a database, or in a cloud storage service.

b.Load the dataset:

Once you have identified the dataset, you need to load it into the machine learning environment. This may involve using a built-in function in the machine learning library, or it may involve writing your own code.

c.Preprocess the dataset:

Once the dataset is loaded into the machine learning environment, you may need to preprocess it before you can start training and evaluating your model. This may involve cleaning the data, transforming the data into a suitable format, and splitting the data into training and test sets.

PYTHON PROGRAM

LOAD THE DATASET:

In [1]:

df=pd.read\_csv("../input/hourly-energy-consumption/AEP\_hourly.csv",index\_col='Datetime',parse\_dates=True)

df.head()

Out [1]:

AEP\_MW

Datetime

2004-12-31 01:00:00 13478.0

2004-12-31 02:00:00 12865.0

2004-12-31 03:00:00 12577.0

2004-12-31 04:00:00 12517.0

2004-12-31 05:00:00 12670.0M:

PROCEDURE:

FEATURE SELECTION:

Feature selection is important to reduce dimensionality and improve the efficiency and accuracy of models for measuring energy consumption. Here are some methods and considerations for feature selection:

1. \*Correlation Analysis:\*

- Calculate the correlation between each feature and the target variable (energy consumption). Select features with the highest correlations.

2. \*Mutual Information:\*

- Use mutual information scores to assess the information shared between features and the target variable.

3. \*Recursive Feature Elimination (RFE):\*

- Train models (e.g., regression or tree-based models) and iteratively remove the least important features based on model performance.

4. \*L1 Regularization (Lasso):\*

- Apply L1 regularization to linear models, encouraging sparse feature selection by driving some feature coefficients to zero.

5. \*Tree-Based Methods:\*

- Decision trees and ensemble methods like Random Forest and Gradient Boosting can provide feature importances, which can guide feature selection.

6. \*Principal Component Analysis (PCA):\*

- Use PCA to reduce dimensionality while preserving variance. Be cautious as PCA may make it harder to interpret feature importance.

7. \*Univariate Feature Selection:\*

- Select features based on statistical tests like ANOVA or chi-squared tests, which evaluate the relationship between each feature and the target.

8. \*Feature Importance from Machine Learning Models:\*

- Some machine learning models provide feature importance scores. Use these scores to rank and select features.

9. \*Domain Knowledge:\*

- Consult domain experts to identify the most relevant features for energy consumption in a specific context.

10. \*Forward and Backward Selection:\*

- Start with no features and add them one by one (forward selection) or start with all features and remove them one by one (backward selection) based on performance.

11. \*Regularization Techniques:\*

- Techniques like Ridge regression can help reduce the impact of less important features by penalizing their coefficients.

12. \*Feature Importance Stability:\*

- Evaluate feature importance stability across different models and datasets to ensure robust selections.

13. \*Cross-Validation:\*

- Perform feature selection within a cross-validation framework to avoid overfitting and select features that generalize well.

It's important to strike a balance between feature reduction and maintaining model interpretability and performance. The choice of feature selection method depends on the dataset, the modeling approach, and the specific goals of your energy consumption measurement task

PROGRAM:

import numpy as np

import pandas as pd

from sklearn.linear\_model import LassoCV

from sklearn.preprocessing import StandardScaler

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

from sklearn.metrics import mean\_squared\_error

# Load your dataset

data = pd.read\_csv('energy\_data.csv') # Replace with your dataset file

# Define features and target variable

X = data.drop('energy\_consumption', axis=1)

y = data['energy\_consumption']

# Split the 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)

# Standardize the features

scaler = StandardScaler()

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

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

# Create a Lasso model with cross-validation to select features

lasso = LassoCV(alphas=np.logspace(-6, 6, 13), cv=5)

lasso.fit(X\_train, y\_train)

# Get the selected features

selected\_features = X.columns[lasso.coef\_ != 0]

# Apply feature selection to your dataset

X\_train\_selected = X\_train[:, lasso.coef\_ != 0]

X\_test\_selected = X\_test[:, lasso.coef\_ != 0]

# Train a model using the selected features (e.g., Linear Regression)

from sklearn.linear\_model import LinearRegression

model = LinearRegression()

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

# Make predictions and evaluate the model

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

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

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

# Now 'selected\_features' contains the names of the selected features for your model.

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

MODEL TRAINING:

1. \*Data Collection:\*

- Gather historical energy consumption data, along with relevant features, as discussed in previous responses.

2. \*Data Preprocessing:\*

- Clean the data by handling missing values, outliers, and data quality issues.

- Normalize or scale the features to ensure they have similar scales.

- Split the data into training, validation, and test sets.

3. \*Feature Engineering:\*

- Create or select relevant features as discussed earlier to improve model accuracy.

4. \*Model Selection:\*

- Choose an appropriate machine learning model for regression tasks. Common choices include:

- Linear Regression

- Decision Trees

- Random Forest

- Gradient Boosting (e.g., XGBoost, LightGBM)

- Neural Networks

5. \*Model Training:\*

- Train the selected model on the training dataset using the features and energy consumption as the target variable.

6. \*Model Evaluation:\*

- Assess the model's performance using appropriate evaluation metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), or R-squared (R²).

- Use the validation dataset to fine-tune hyperparameters and prevent overfitting.

7. \*Feature Importance Analysis:\*

- Analyze feature importances to understand which features have the most significant impact on energy consumption.

8. \*Model Interpretability:\*

- Consider using techniques to interpret the model's predictions, especially if the model is complex (e.g., SHAP values for tree-based models).

9. \*Hyperparameter Tuning:\*

- Optimize the model's hyperparameters using techniques like grid search or random search.

10. \*Cross-Validation:\*

- Perform cross-validation to ensure the model's generalizability and robustness.

11. \*Model Deployment:\*

- Deploy the trained model in a production environment for real-time or batch predictions.

12. \*Monitoring and Maintenance:\*

- Continuously monitor the model's performance in the production environment and retrain as needed to account for changes in energy consumption patterns.

13. \*Explainability and Reporting:\*

- Create reports and dashboards to communicate model predictions and insights to stakeholders.

MACHINE LEARNING MODELS:

Machine Learning Models:

models = pd.DataFrame(columns=["Model","MAE","MSE","RMSE","R2 S

core","RMSE (Cross-Validation)"])

MEASURE ENERGY CONSUMPTION

In [2]:

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import matplotlib.dates as mdates

%matplotlib inline

import seaborn as sns

import warnings

warnings.filterwarnings("ignore")

from pandas.plotting import lag\_plot

from pylab import rcParams

from statsmodels.tsa.seasonal import seasonal\_decompose

from pandas import DataFrame

from pandas import concat

In [3]:

df.sort\_values(by='Datetime', inplace=True)

print(df)

Out [3]:

AEP\_MW

Datetime

2004-10-01 01:00:00 12379.0

2004-10-01 02:00:00 11935.0

2004-10-01 03:00:00 11692.0

2004-10-01 04:00:00 11597.0

2004-10-01 05:00:00 11681.0

... ...

2018-08-02 20:00:00 17673.0

2018-08-02 21:00:00 17303.0

2018-08-02 22:00:00 17001.0

2018-08-02 23:00:00 15964.0

2018-08-03 00:00:00 14809.0

[121273 rows x 1 columns]

In [4]:

df.shape()

Out [4]:

(121273,1)

In [5]:

df.info()

Out [5]:

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

DatetimeIndex: 121273 entries, 2004-10-01 01:00:00 to 2018-08-03 00:00:00

Data columns (total 1 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 AEP\_MW 121273 non-null float64

dtypes: float64(1)

memory usage: 1.9 MB

In [6]:

df.describe()

Out [6]:

AEP\_MW

count 121273.000000

mean 15499.513717

std 2591.399065

min 9581.000000

25% 13630.000000

50% 15310.000000

75% 17200.000000

max 25695.000000

In [7]:

df.index = pd.to\_datetime(df.index)

In [8]:

# Extract all Data Like Year MOnth Day Time etc

df["Month"] = df.index.month

df["Year"] = df.index.year

df["Date"] = df.index.date

df["Hour"] = df.index.hour

df["Week"] = df.index.week

df["Day"] = df.index.day\_name()

df.head()

Out [8]:

AEP\_MW Month Year Date Hour Week Day

Datetime

2004-10-01 01:00:00 12379.0 10 2004 2004-10-01 1 40 Friday

2004-10-01 02:00:00 11935.0 10 2004 2004-10-01 2 40 Friday

2004-10-01 03:00:00 11692.0 10 2004 2004-10-01 3 40 Friday

2004-10-01 04:00:00 11597.0 10 2004 2004-10-01 4 40 Friday

2004-10-01 05:00:00 11681.0 10 2004 2004-10-01 5 40 Friday

MODEL EVALUATION:

1. \*Mean Absolute Error (MAE):\*

- MAE measures the average absolute difference between the model's predictions and the actual energy consumption values. It provides an easily interpretable measure of prediction accuracy.

2. \*Mean Squared Error (MSE):\*

- MSE measures the average of the squared differences between predictions and actual values. It penalizes larger errors more heavily than MAE.

3. \*Root Mean Squared Error (RMSE):\*

- RMSE is the square root of the MSE and provides an interpretable metric in the same units as the target variable (energy consumption). It is sensitive to outliers.

4. \*R-squared (R²) or Coefficient of Determination:\*

- R² measures the proportion of the variance in the energy consumption that is explained by the model. A higher R² indicates a better fit, but be cautious about overfitting.

5. \*Coefficient of Variation (CV):\*

- CV calculates the ratio of the standard deviation to the mean of the residuals. It can help assess the relative error.

6. \*Percentage Error or Relative Error:\*

- Calculate the percentage difference between predicted and actual energy consumption to gauge the relative error.

7. \*Residual Analysis:\*

- Examine the distribution of residuals (the differences between predictions and actual values) to check for patterns or biases.

8. \*Cross-Validation:\*

- Perform cross-validation, such as k-fold cross-validation, to assess the model's generalization performance and reduce overfitting.

9. \*Feature Importance Analysis:\*

- Analyze feature importances to understand which features contribute the most to energy consumption predictions.

10. \*Domain Expert Review:\*

- Consult domain experts to ensure that the model's predictions align with physical, economic, and behavioral realities.

11. \*Business Impact Assessment:\*

- Evaluate the practical impact of model predictions on energy management and decision-making within the specific application.

12. \*Benchmarking:\*

- Compare the model's performance against existing benchmarks or other models to assess its relative effectiveness.

13. \*Interpretability and Explainability:\*

- Use methods like SHAP (SHapley Additive exPlanations) to provide interpretability and insights into how the model is making predictions

In [9]:

df.plot(title="PJME Energy use in MegaWatts",

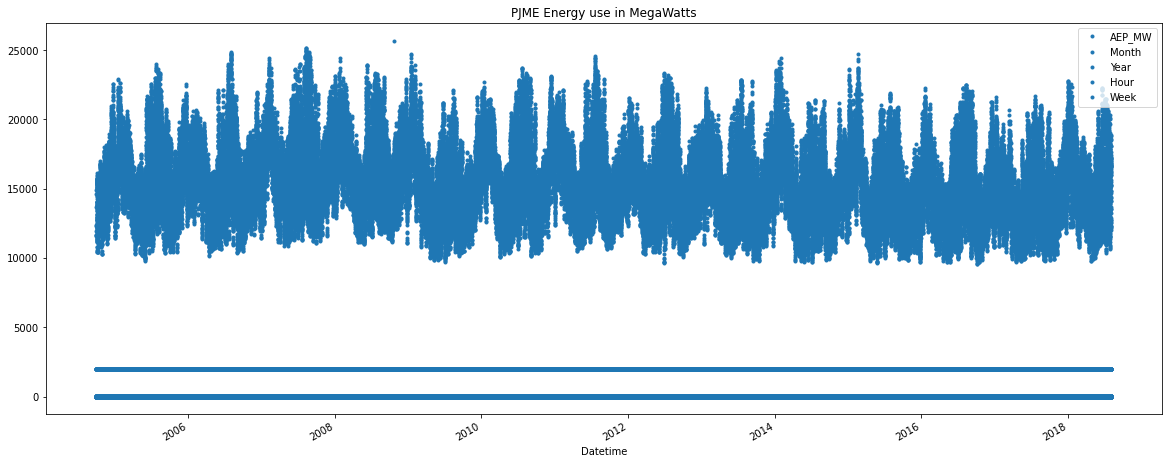
        figsize=(20, 8),

        style=".",

        color=sns.color\_palette()[0])

plt.show()

Out [9]:



In [10]:

df.tail()

Out [10]:

AEP\_MW Month Year Date Hour Week Day

Datetime

2018-08-02 20:00:00 17673.0 8 2018 2018-08-02 20 31 Thursday

2018-08-02 21:00:00 17303.0 8 2018 2018-08-02 21 31 Thursday

2018-08-02 22:00:00 17001.0 8 2018 2018-08-02 22 31 Thursday

2018-08-02 23:00:00 15964.0 8 2018 2018-08-02 23 31 Thursday

2018-08-03 00:00:00 14809.0 8 2018 2018-08-03 0 31 Friday

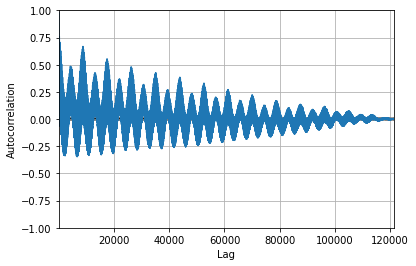
In [11]:

from pandas.plotting import autocorrelation\_plot

autocorrelation\_plot(df['AEP\_MW'])

plt.show()

Out [11]:



In [12]:

from sklearn.metrics import mean\_squared\_error, mean\_absolute\_error

from math import sqrt

from sklearn.preprocessing import MinMaxScaler

# Analysis imports

from pandas.plotting import lag\_plot

from pylab import rcParams

from statsmodels.tsa.seasonal import seasonal\_decompose

from pandas import DataFrame

from pandas import concat

# Modelling imports

from statsmodels.tsa.ar\_model import AR

from statsmodels.tsa.arima\_model import ARMA

from statsmodels.tsa.arima\_model import ARIMA

from keras.models import Sequential

from keras.layers import Dense

from keras.layers import LSTM, GRU, RNN

from keras.layers import Dropout

In [13]:

values = DataFrame(df['AEP\_MW'].values)

dataframe = concat([values.shift(1),values.shift(5),values.shift(10),values.shift(30), values], axis=1)

dataframe.columns = ['t', 't+1', 't+5', 't+10', 't+30']

result = dataframe.corr()

print(result)

Out [13]:

t t+1 t+5 t+10 t+30

t 1.000000 0.731161 0.345667 0.501972 0.976223

t+1 0.731161 1.000000 0.630009 0.847210 0.630007

t+5 0.345667 0.630009 1.000000 0.644479 0.317277

t+10 0.501972 0.847210 0.644479 1.000000 0.408315

t+30 0.976223 0.630007 0.317277 0.408315 1.000000

In [14]:

train\_data, test\_data = df[0:-60], df[-60:]

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

plt.grid(True)

plt.xlabel('Dates')

plt.ylabel('Energy in megawatts')

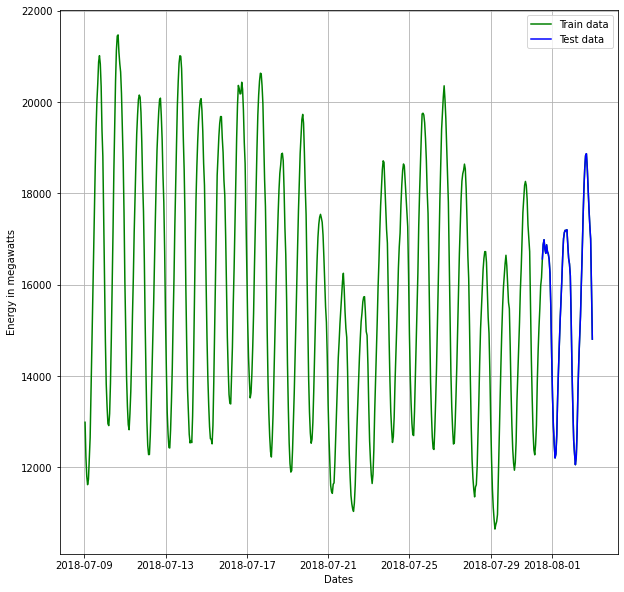
plt.plot(df['AEP\_MW'].tail(600), 'green', label='Train data')

plt.plot(test\_data['AEP\_MW'], 'blue', label='Test data')

plt.legend()

Out [14]:

<matplotlib.legend.Legend at 0x78af47a94e50>



In [15]:

mean\_value = df['AEP\_MW'].mean() # calculation of mean price

plt.figure(figsize=(16,8))

plt.grid(True)

plt.xlabel('Dates')

plt.ylabel('Energy in megawatts')

plt.plot(df['AEP\_MW'], 'green', label='Train data')

plt.plot(test\_data['AEP\_MW'], 'blue', label='Test data')

plt.axhline(y=mean\_value, xmin=0.864, xmax=1, color='red')

plt.legend()

plt.figure(figsize=(16,8))

plt.grid(True)

plt.xlabel('Dates')

plt.ylabel('Energy in megawatts')

plt.plot(df['AEP\_MW'].tail(600), 'green', label='Train data')

plt.plot(test\_data['AEP\_MW'], 'blue', label='Test data')

plt.axhline(y=mean\_value, xmin=0.864, xmax=1, color='red')

plt.legend()

print('MSE: '+str(mean\_squared\_error(test\_data['AEP\_MW'], np.full(len(test\_data), mean\_value))))

print('MAE: '+str(mean\_absolute\_error(test\_data['AEP\_MW'], np.full(len(test\_data), mean\_value))))

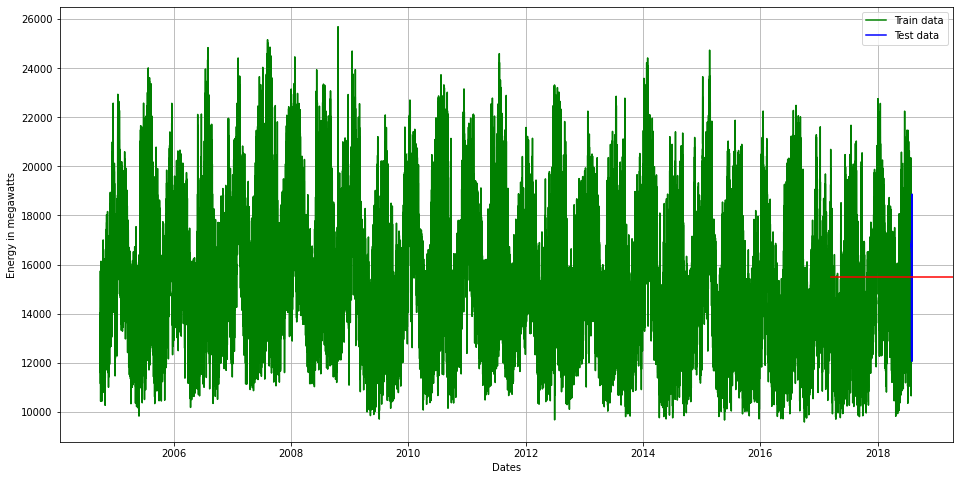
print('RMSE: '+str(sqrt(mean\_squared\_error(test\_data['AEP\_MW'], np.full(len(test\_data), mean\_value)))))

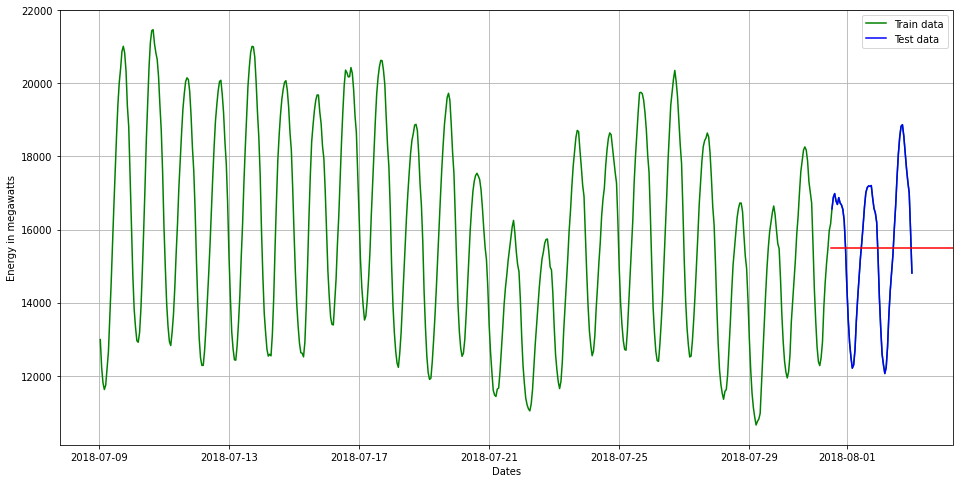
Out [15]:

MSE: 3700885.0406027567

MAE: 1667.1805899362046

RMSE: 1923.768447761517





In [16]:

from statsmodels.tsa.stattools import adfuller

def adf\_test(dataset):

     dftest = adfuller(dataset, autolag = 'AIC')

     print("1. ADF : ",dftest[0])

     print("2. P-Value : ", dftest[1])

     print("3. Num Of Lags : ", dftest[2])

     print("4. Num Of Observations Used For ADF Regression:",      dftest[3])

     print("5. Critical Values :")

     for key, val in dftest[4].items():

         print("\t",key, ": ", val)

In [17]:

adf\_test(df['AEP\_MW'])

Out [17]:

1. ADF : -18.285883882257217

2. P-Value : 2.3029539101747796e-30

3. Num Of Lags : 71

4. Num Of Observations Used For ADF Regression: 121201

5. Critical Values :

1% : -3.430403955318047

5% : -2.8615638474512295

10% : -2.566782693155802

In [18]:

import statsmodels.api as sm

In [19]:

*#Train Arima Model*

train\_arima = train\_data['AEP\_MW']

test\_arima = test\_data['AEP\_MW']

history = [x for x **in** train\_arima]

y = test\_arima

*# make first prediction*

predictions = list()

model = sm.tsa.arima.ARIMA(history, order=(5,1,0))

model\_fit = model.fit()

yhat = model\_fit.forecast()[0]

predictions.append(yhat)

history.append(y[0])

*# rolling forecasts*

for i **in** range(1, len(y)):

*# predict*

model = sm.tsa.arima.ARIMA(history, order=(5,1,0))

model\_fit = model.fit()

yhat = model\_fit.forecast()[0]

*# invert transformed prediction*

predictions.append(yhat)

*# observation*

obs = y[i]

history.append(obs)

plt.figure(figsize=(14,8))

plt.plot(df.index, df['AEP\_MW'], color='green', label = 'Train Energy AEP\_MW')

plt.plot(test\_data.index, y, color = 'red', label = 'Real Energy AEP\_MW')

plt.plot(test\_data.index, predictions, color = 'blue', label = 'Predicted Energy AEP\_MW')

plt.legend()

plt.grid(True)

plt.show()

plt.figure(figsize=(14,8))

plt.plot(df.index[-600:], df['AEP\_MW'].tail(600), color='green', label = 'Train Energy AEP\_MW')

plt.plot(test\_data.index, y, color = 'red', label = 'Real Energy AEP\_MW')

plt.plot(test\_data.index, predictions, color = 'blue', label = 'Predicted Energy AEP\_MW')

plt.legend()

plt.grid(True)

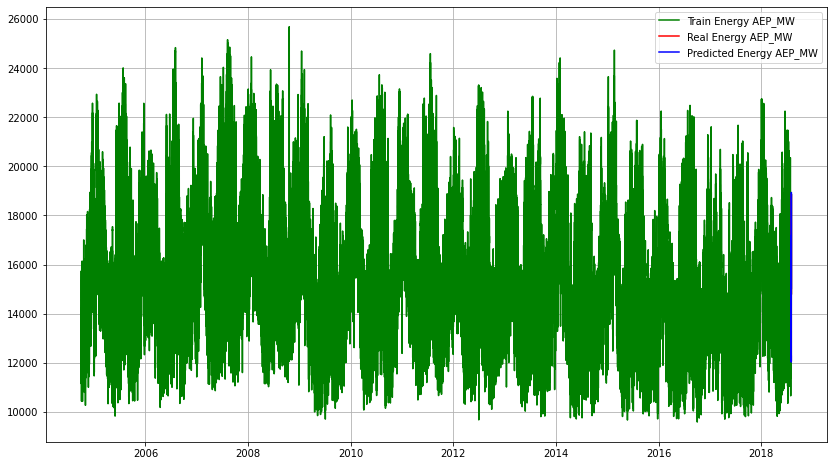
plt.show()

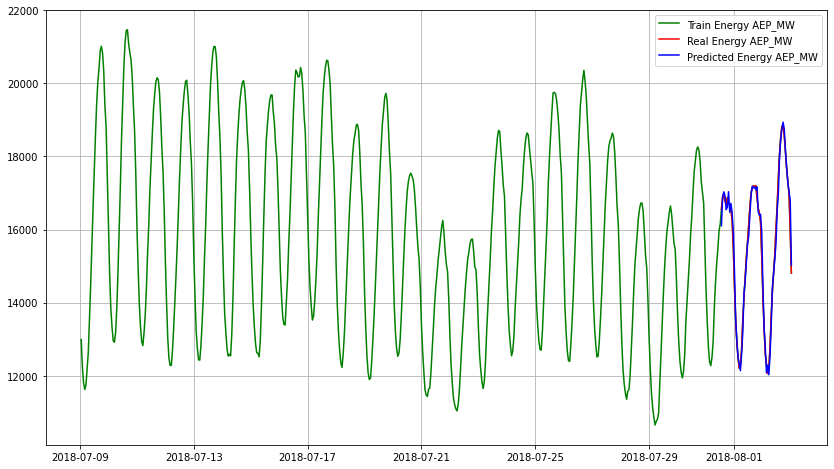
print('MSE: '+str(mean\_squared\_error(y, predictions)))

print('MAE: '+str(mean\_absolute\_error(y, predictions)))

print('RMSE: '+str(sqrt(mean\_squared\_error(y, predictions))))

Out [19]:





MSE: 57710.45153428949

MAE: 177.320844006739

RMSE: 240.2299971574938

FEATURE ENGINEERING:

Feature engineering is crucial when measuring energy consumption. Here are some key features to consider:

1. Time-Based Features:

- Timestamps: Extract features like hour of the day, day of the week, and season.

- Time intervals: Calculate the time between readings.

2. Weather Data:

- Temperature, humidity, and weather conditions can impact energy consumption.

3. Building Characteristics:

- Square footage, number of occupants, insulation, and building age can influence consumption.

4. Appliance Information:

- Identify major appliances and their energy ratings to attribute consumption.

5. Previous Usage:

- Lag features to account for historical energy consumption patterns.

6. Demographics:

- Location-specific data, such as population density, can be relevant.

7. Occupancy Data:

- Use occupancy sensors to detect when a building or room is in use.

8. Energy Price:

- If applicable, include the cost of energy, which can influence consumption.

9. Holidays and Events:

- Special occasions or holidays can affect energy usage.

10. Behavioral Data:

- User behaviors, routines, and preferences can provide insights.

ADVANTAGES:

Measuring energy consumption using machine learning offers several advantages:

1. Predictive Insights:

Machine learning models can provide accurate predictions of future energy consumption, helping businesses and individuals make informed decisions about energy management.

2. Anomaly Detection:

ML can identify abnormal energy usage patterns, enabling the early detection of equipment malfunctions or energy wastage.

3. Energy Efficiency:

ML can optimize energy usage by suggesting adjustments to HVAC systems, lighting, and other appliances, leading to cost savings and reduced carbon footprint.

4. Personalized Recommendations:

ML can offer personalized energy-saving tips based on user behavior and preferences.

5. Demand Forecasting:

ML can help utilities predict peak demand periods, allowing for better grid management and resource allocation.

6. Data Granularity:

ML can analyze large datasets with high granularity, capturing subtle variations in energy consumption.

7. Real-time Monitoring:

ML can provide real-time insights into energy usage, enabling immediate responses to abnormal patterns or demand fluctuations.

8. Continuous Improvement:

ML models can adapt and improve over time as they learn from more data, leading to increasingly accurate energy consumption predictions.

9. Cost Reduction:

By optimizing energy usage and reducing waste, businesses and individuals can lower their energy bills.

10. Environmental Impact:

ML-driven energy management can contribute to a reduction in greenhouse gas emissions and overall environmental sustainability.

Overall, machine learning enables more effective and data-driven energy management, which can lead to cost savings, reduced environmental impact, and improved efficiency.

DISADVANTAGES:

While using machine learning for evaluating energy consumption offers several advantages, it also comes with some disadvantages and challenges:

1. Data Quality: ML models rely on high-quality data. Inaccurate or incomplete energy consumption data can lead to unreliable predictions.

2. Data Privacy: Collecting and analyzing detailed energy usage data may raise privacy concerns, especially when dealing with personal or sensitive information.

3. Model Complexity: Developing and maintaining ML models can be complex, requiring expertise in machine learning and data science, which may not be readily available to everyone.

4. Initial Cost: Implementing ML solutions for energy consumption analysis can involve upfront costs for hardware, software, and personnel training.

5. Energy Consumption for Model Training: Building and training ML models itself consumes computational resources and energy, potentially offsetting some of the energy savings achieved.

6. Interpretability: ML models can be difficult to interpret, making it challenging to understand the reasoning behind their predictions, which is important for building trust and making informed decisions.

7. Scalability: Scaling ML solutions to accommodate a large number of devices or sensors can be challenging and may require substantial computing resources.

8. Model Maintenance: ML models require ongoing maintenance and retraining to remain accurate, which can be resource-intensive.

9. Overfitting: ML models may overfit to historical data, making them less effective when faced with unexpected events or changes in energy consumption patterns.

10. User Adoption: Encouraging users to adopt energy-saving recommendations generated by ML systems can be challenging if they do not trust or understand the technology.

11. Ethical Concerns: ML models may inadvertently reinforce biases in data, leading to unfair or inequitable energy management decisions.

It's essential to carefully consider these disadvantages and address them during the implementation of machine learning solutions for energy consumption evaluation to maximize the benefits and mitigate potential drawbacks.

BENEFITS:

Measuring energy consumption using machine learning (ML) offers various benefits, including:

1. Precision: ML models can provide highly accurate predictions and insights into energy consumption, helping users make informed decisions.

2. Cost Savings: Optimizing energy usage based on ML recommendations can lead to significant cost savings on energy bills for businesses and individuals.

3. Environmental Impact: ML-driven energy management can reduce greenhouse gas emissions and contribute to environmental sustainability by promoting energy efficiency.

4. Real-time Monitoring: ML enables real-time monitoring of energy usage, allowing for immediate responses to anomalies and fluctuations in demand.

5. Predictive Maintenance: ML can detect abnormal energy usage patterns, helping with the early identification of equipment malfunctions and reducing downtime.

6. Customization: ML can offer personalized energy-saving recommendations tailored to individual preferences and behaviors.

7. Demand Forecasting: ML can assist utilities in predicting peak demand periods, facilitating better grid management and resource allocation.

8. Data Analysis: ML can handle vast amounts of energy consumption data with high granularity, uncovering insights and patterns that might be missed using traditional methods.

9. Continuous Improvement: ML models can adapt and improve over time as they learn from more data, leading to increasingly accurate energy consumption predictions.

10. Enhanced User Experience: ML-driven energy management systems can provide users with user-friendly interfaces and smart devices that simplify energy-saving measures.

11. Scalability: ML solutions can be scaled to accommodate a large number of devices or sensors, making them suitable for various applications.

12. Reduced Wastage: ML can identify and reduce energy wastage, making it an effective tool for improving energy efficiency.

Overall, measuring energy consumption using ML can lead to improved energy management, cost-efficiency, sustainability, and user experiences, making it a valuable tool in today's energy-conscious world.

CONCLUSION:

In conclusion, measuring energy consumption, especially when leveraging machine learning, is a valuable and powerful approach with numerous benefits. It enables precision, cost savings, and a positive environmental impact. Real-time monitoring, predictive maintenance, customization, and demand forecasting are among the advantages that enhance energy management.

However, it's essential to be mindful of potential challenges such as data quality, privacy concerns, model complexity, and the ongoing maintenance of ML solutions. To maximize the benefits and address these challenges, careful planning and implementation are necessary.

Overall, measuring energy consumption using machine learning represents a significant step toward efficient and sustainable energy management, offering a data-driven and technologically advanced approach to reducing costs and minimizing environmental impact.