Measure Energy Consumption Documentation

Problem Statement: The measurement of energy consumption is critical in understanding and optimizing energy usage in various sectors, including manufacturing sites, homes, commercial buildings, and transportation. However, the manual collection and analysis of energy consumption data can be time-consuming and error-prone. Therefore, there is a need for an automated approach to collect, analyze and visualize energy consumption data for better decision-making.

Problem Definition and Design Thinking

Problem Definition: The problem at hand is to create an automated system that measures energy consumption, analyzes the data, and provides visualizations for informed decision-making. This solution aims to enhance efficiency, accuracy, and ease of understanding in managing energy consumption across various sectors.

Design Thinking:

1. Data Source: Identify an available dataset containing energy consumption measurements.
2. Data Preprocessing: Clean, transform, and prepare the dataset for analysis.
3. Feature Extraction: Extract relevant features and metrics from the energy consumption data.
4. Model Development: Utilize statistical analysis to uncover trends, patterns, and anomalies in the data.
5. Visualization: Develop visualizations (graphs, charts) to present the energy consumption trends and insights.
6. Automation: Build a script that automates data collection, analysis, and visualization processes.

Data Collection and Preprocessing

Data Collection:

- Download the energy consumption dataset from the provided Kaggle link.
- Ensure that the dataset includes relevant information such as date and time, energy consumption values, and any additional metadata.

Data Cleaning:

- Handle missing data points by either imputing them or removing rows with missing values.
- Check for duplicates and eliminate them if necessary.
- Validate data types and ensure consistency.

Data Transformation:

- Convert date and time columns into a consistent format and create a timestamp.
- Aggregate data into appropriate time intervals (e.g., hourly, daily) if needed.
- Normalize or scale energy consumption values if required.

Feature Extraction and Model Development

Feature Extraction:

- Identify relevant features such as seasonality, day of the week, and time of day.
- Calculate summary statistics (e.g., mean, standard deviation) for energy consumption.
- Create lag features to capture historical patterns.

Statistical Analysis:

- Perform statistical analysis to uncover trends, patterns, and anomalies in the data.
- Use time series analysis techniques (e.g., ARIMA, Exponential Smoothing) to model energy consumption patterns.
- Detect outliers or anomalies that may indicate equipment malfunctions or unusual energy spikes.

Visualization

Data Visualization:

- Create visualizations like line charts, bar graphs, and heatmaps to illustrate energy consumption trends.
- Use tools like Matplotlib, Seaborn, or Plotly for visualization.
- Include descriptive titles and labels to make the visualizations informative and easy to understand.

Interactive Dashboards (optional):

- Consider building interactive dashboards using tools like Tableau, Power BI, or Dash for more dynamic exploration of the data.

Automation and Reporting

Automation:

- Develop a script or workflow to automate the entire process, including data collection, cleaning, feature extraction, modeling, and visualization.
- Schedule regular updates to ensure that the analysis remains up-to-date.

Report Generation:

- Generate automated reports summarizing energy consumption insights.
- Include key findings, trends, and recommendations for energy optimization.
- Save reports in a suitable format (e.g., PDF, HTML) and distribute them to relevant stakeholders.

Documentation:

- Document the entire project, including data sources, preprocessing steps, feature extraction, modeling techniques, and visualization methods.
- Include instructions for maintaining and updating the automated system.

Testing and Validation:

- Validate the accuracy of the automated system by comparing its results to manual analysis or historical data.
- Conduct user testing to ensure that the visualizations are user-friendly and meet the needs of stakeholders.

Deployment:

- Deploy the automated system in the target environment, whether it's a manufacturing site, commercial building, or elsewhere.
- Monitor the system for any issues and perform regular maintenance.

INOVATION

Analysis:

Machine learning models can be used to analyse energy consumption data in a variety of ways. For example, machine learning models can be used to:

1) cluster energy consumers based on their consumption patterns.
2) Predict future energy consumption.
3) Identify anomalies in energy consumption data.

To analyse energy consumption data using a machine learning model, the following steps can be taken:

Prepare the data:

The first step is to prepare the data for training the machine learning model. This may involve cleaning the data, removing outliers, and transforming the data into a format that is compatible with the machine learning algorithm.

Choose a machine learning algorithm:

There are a variety of machine learning algorithms that can be used to analyse energy consumption data. Some popular algorithms include **random forests, gradient boosting, and support vector machines.**

Train the machine learning model:

Once the data has been prepared, the machine learning model can be trained. This involves feeding the model the prepared data and allowing it to learn the patterns in the data.

Evaluate the machine learning model:

Once the machine learning model is trained, it should be evaluated to assess its performance. This can be done by feeding the model a held-out test set and measuring its accuracy.

Use the machine learning model to analyse new data:

Once the machine learning model has been trained and evaluated, it can be used to analyse new energy consumption data. This can be used to cluster energy consumers, predict future energy consumption, or identify anomalies in energy consumption data.

Data Loading:

- Import the necessary Python libraries, such as Pandas and NumPy.
- Load the dataset from the provided Kaggle link. You can use Pandas to read CSV files or other formats.

Data Cleaning:

- Check for and handle any missing values.
- Convert the timestamp column to a datetime data type for time-based analysis.

Exploratory Data Analysis (EDA):

- Perform basic data exploration to understand the data's characteristics.
- Visualize the time series data to identify trends and patterns.

Feature Engineering:

- Create additional features, such as lag features (past values) or rolling statistics.
- These features can provide more information for forecasting.

Time Series Forecasting:

- Choose a time series forecasting model, such as ARIMA, Exponential Smoothing, or LSTM.
- Split the data into training and testing sets.
- Train the selected model on the training data and evaluate it on the test data.

Data Processing:

First we import some basic python libraries like pandas numpy and matplotlib in our project and then we initialized our dataset in our project file by using the read csv command in the pandas as our project is in the csv file.

And then we started our processing of data in the project by dropping non values and unnecessary columns in the data for the prediction. Thus this is the preprocessing that has been done in our project. let me attach the snippet of the data processing using pandas.

Python Script:

Step 1: Setup

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as
plt import seaborn as sns
import xgboost as xgb
from sklearn.metrics import mean_squared_error

# customize the style
pd.options.display.float_format = '{:.5f}'.format
pd.options.display.max_rows = 12
```

#Load the data

```
filepath = '../input/hourly-energy-consumption/PJME_hourly.csv'
df = pd.read_csv(filepath)

print("Successfully Uploaded")
```

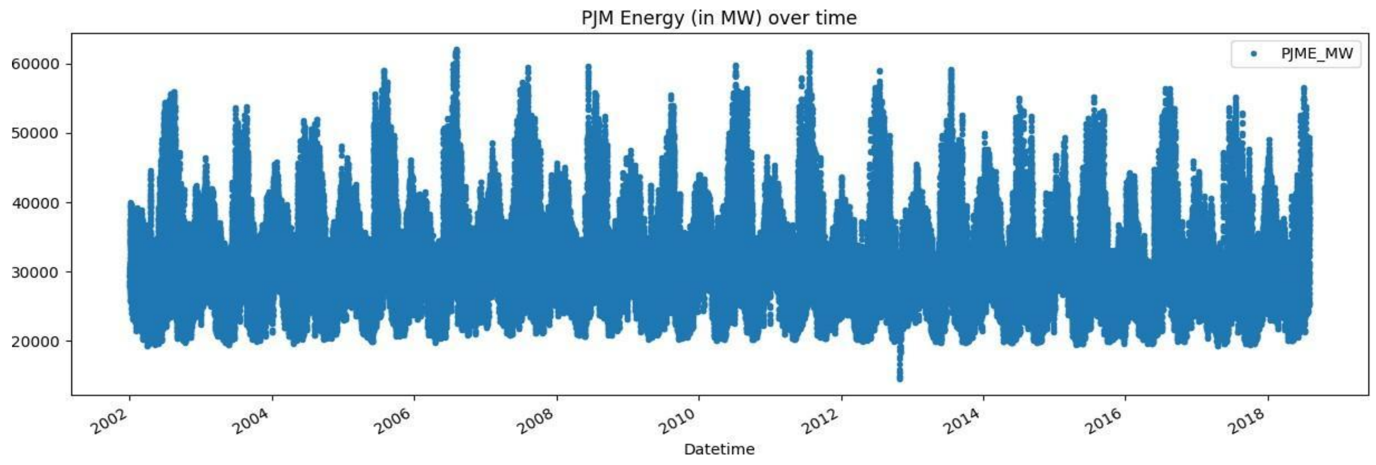
Step 2: Explore the data

# turn data to datetime

```
df = df.set_index('Datetime')
df.index =
pd.to_datetime(df.index)
```

# create the plot

```
df.plot(style='.',
        figsize=(15, 5),
        title='PJM Energy (in MW) over time')
plt.show()
```



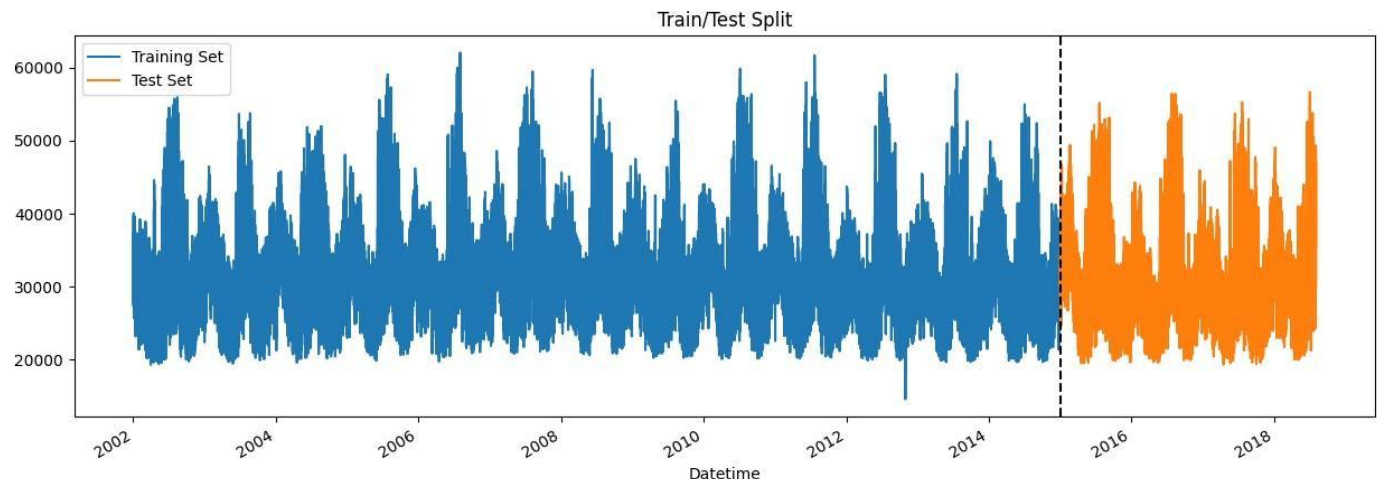
Step 3: Split the data

# train / test split

```
train = df.loc[df.index < '01-01-2022']
test = df.loc[df.index >= '01-01-2022']
unfold_lessHide code
In [5]:
Linkcode

fig, ax = plt.subplots(figsize=(15, 5))
train.plot(ax=ax, label='Training Set', title='Train/Test Split')
```

```
test.plot(ax=ax, label='Test Set')
ax.axvline('01-01-2022', color='black', ls='--')
ax.legend(['Training Set', 'Test Set'])
plt.show()
```



After exploring the data, you need to prepare it for analysis, which involves

- ❖ Exploratory Data Analysis (EDA):
- ❖ Feature Engineering:
- ❖ Time Series Forecasting:

Setting up the data for analysis is crucial to ensure that it's in the right format and condition for modeling. It's the foundation for accurate predictions in the next steps of your project, which typically involve selecting a forecasting model and training it.

Step 3: Feature Engineering

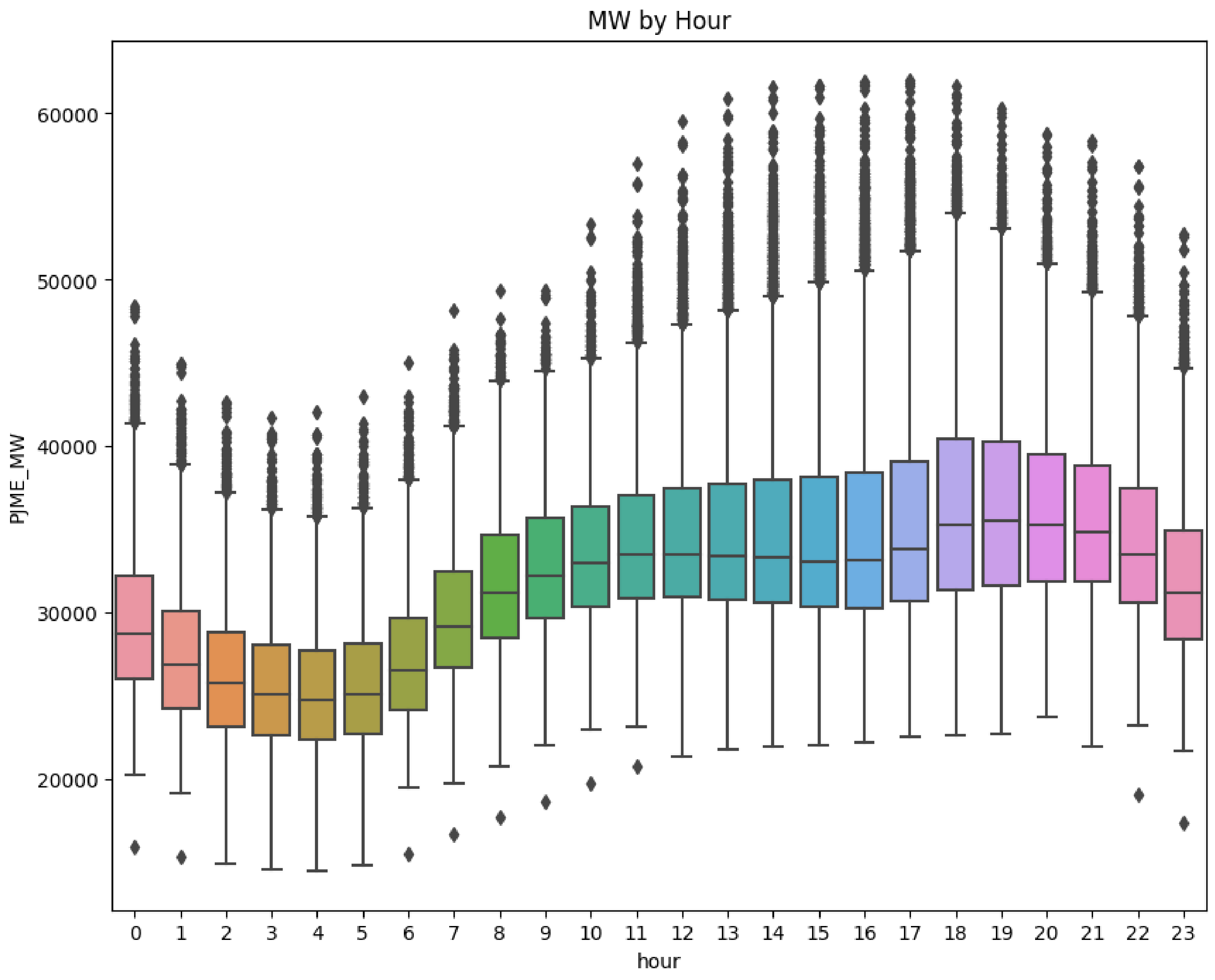
We're going to create some time features using the Datetime index. After that, we'll explore the distributions of Hourly and Monthly megawatt usage.

# feature creation

```
def
    create_featur
    es(df): df =
    df.copy()
    df['hour'] = df.index.hour
    df['dayofweek'] =
    df.index.dayofweek
    df['quarter'] = df.index.quarter
    df['month'] = df.index.month
    df['year'] = df.index.year
    df['dayofyear'] =
    df.index.dayofyear
    df['dayofmonth'] =
    df.index.day
    df['weekofyear'] =
    df.index.isocalendar().week return df
df = create_features(df)
```

# visualize the hourly Megawatt

```
fig, ax = plt.subplots(figsize=(10, 8))
sns.boxplot(data=df, x='hour',
y='PJME_MW') ax.set_title('MW by
Hour')
plt.show()
```



We can see here that after midnight, the use of energy go down and it gets higher from around 6AM to 6PM and then go down again.

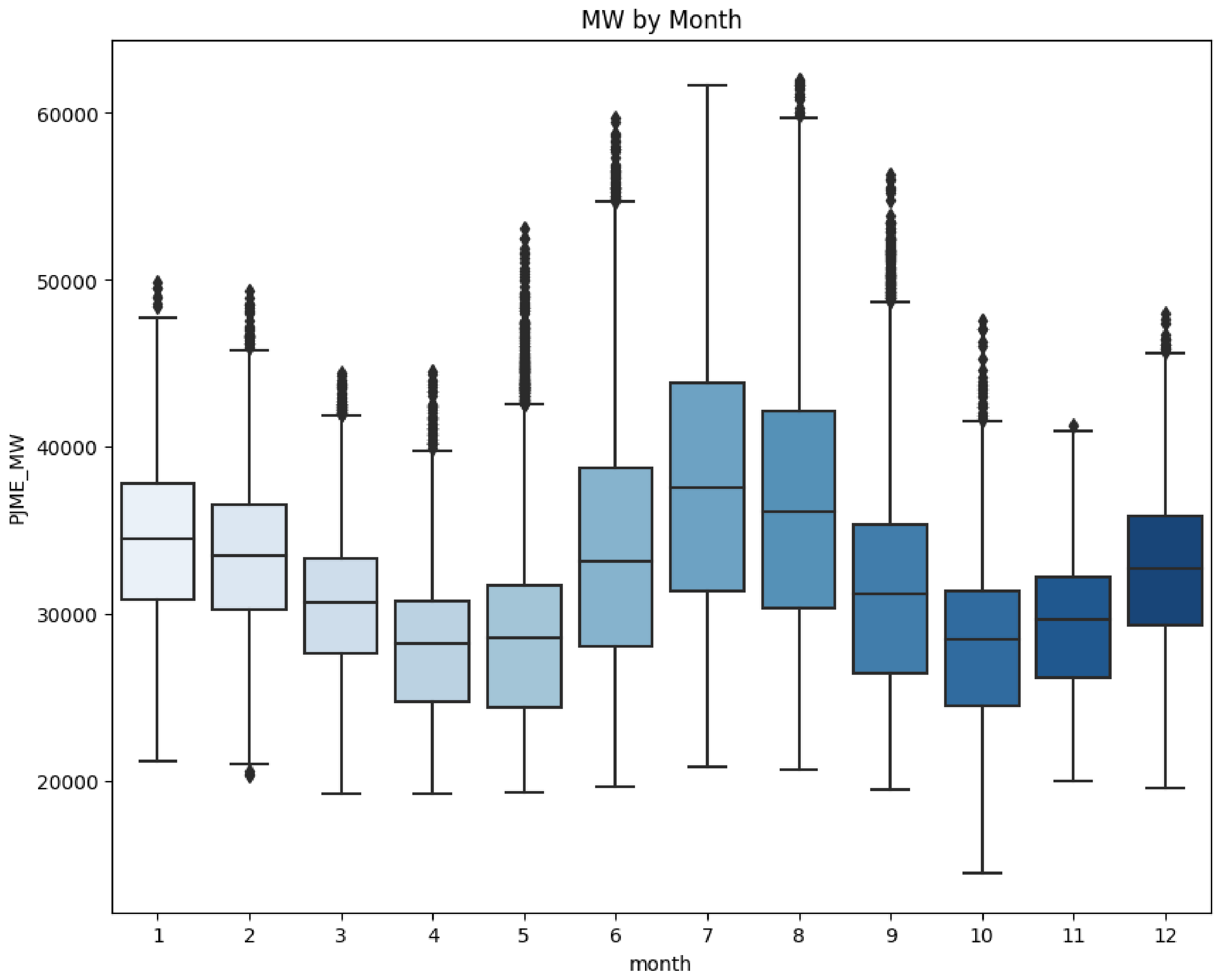
```
# viaualize the monthly Megawatt
fig, ax = plt.subplots(figsize=(10, 8))
sns.boxplot(data=df, x='month', y='PJME_MW',
palette='Blues') ax.set_title('MW by Month')
```

```
plt.show()
```

The monthly usage tends to peak here two times in the winter season, then in the fall and sprint it has lower and another peak in the middle of summer.

Step 4: Modelling

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

```
# 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

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

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

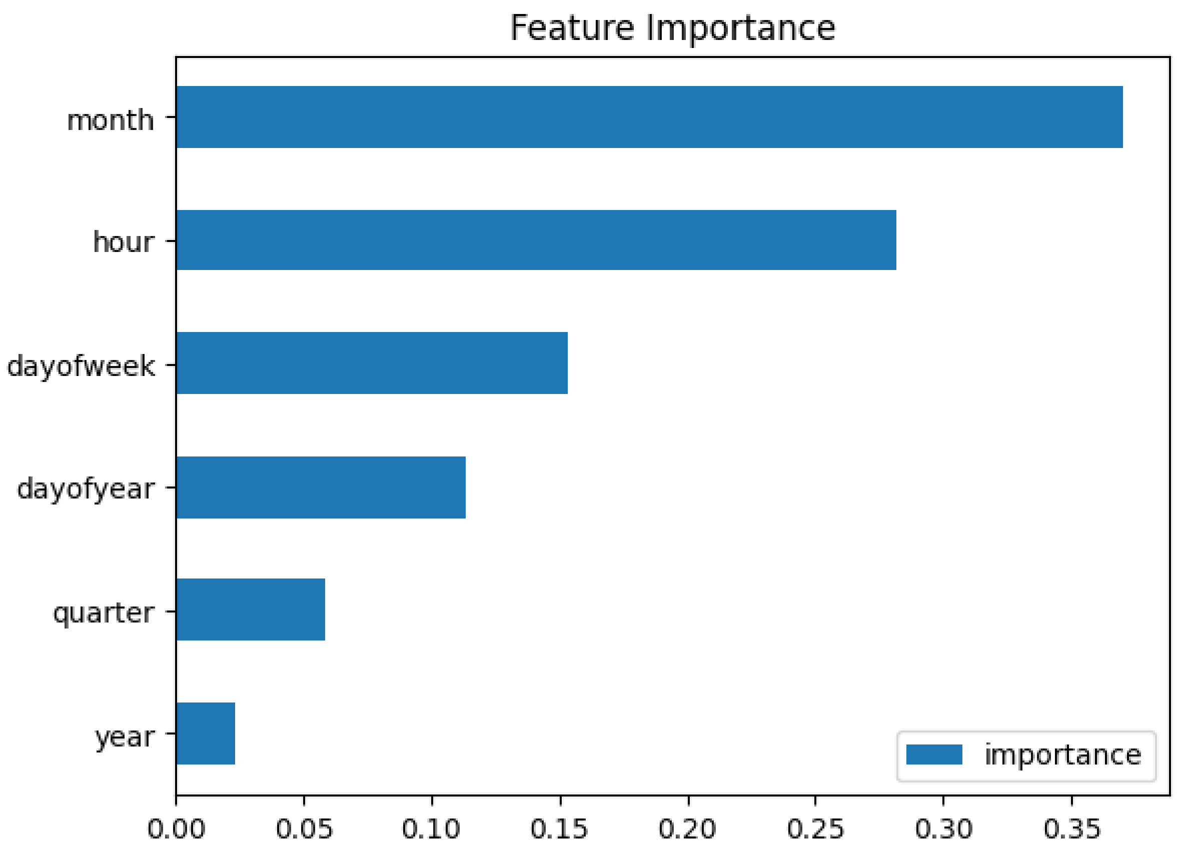
XGBRegressor

```
XGBRegressor(base_score=0.5, booster='gbtree', callbacks=None,
colsample_bylevel=None, colsample_bynode=None,
colsample_bytree=None, early_stopping_rounds=50,
enable_categorical=False, eval_metric=None, feature_types=None,
gamma=None, gpu_id=None, grow_policy=None,
importance_type=None, interaction_constraints=None,
learning_rate=0.01, max_bin=None, max_cat_threshold=None,
max_cat_to_onehot=None, max_delta_step=None, max_depth=3,
max_leaves=None, min_child_weight=None, missing=nan,
monotone_constraints=None, n_estimators=1000, n_jobs=None,
num_parallel_tree=None, objective='reg:linear', predictor=None,
...)
```

Features importance

We need to see how much these features were used in each of the trees built by XGBoost model.

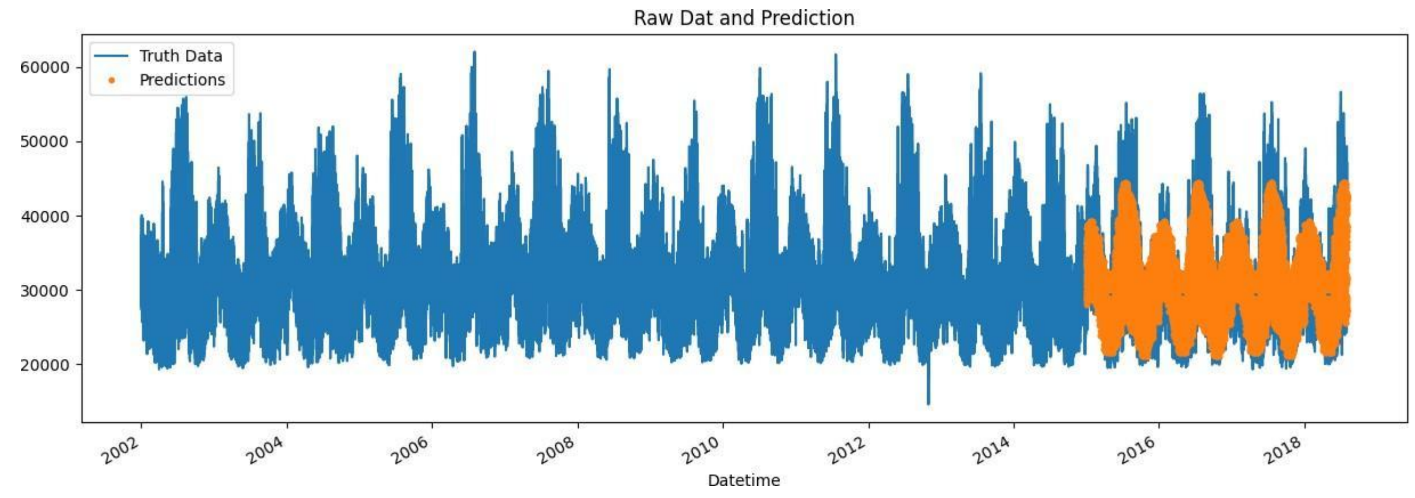
```
Fi=
          pd.DataFrame(data=reg.feature_i
          mportances_,
          index=reg.feature_names_in_,
columns=['importance'])
    fi.sort_values('importance').plot(kind='barh', title='Feature
Importance')
    plt.show()
```



Forecasting on test data

compare the prediction with the actual values.

```
test['prediction'] = reg.predict(X_test)
df = df.merge(test[['prediction']], how='left', left_index=True,
right_index=True) ax = df[['PJME_MW']].plot(figsize=(15, 5))
df['prediction'].plot(ax=ax,
style='.') plt.legend(['Truth
Data', 'Predictions'])
ax.set_title('Raw Dat and
Prediction') plt.show()
```



# RMSE Score

```
score = np.sqrt(mean_squared_error(test['PJME_MW'],
test['prediction'])) print(f'RMSE Score on Test set: {score:0.2f}')
```

**RMSE Score on Test set:
3721.75 # R2 Score**

```
from sklearn.metrics import r2_score
r2 = r2_score(test['PJME_MW'],
test['prediction']) print("R-squared (R2)
Score:", r2)
```

```
R-squared (R2) Score: 0.6670230260104328
```

The result is not that good, but it's a great starting point for your future model.

Logs

Time	#	Log Message
7.2s	1	Now, you're ready for step one
11.9s	2	[19:42:36] WARNING: ../src/objective/regression_obj.cu:213: reg:linear is now deprecated in favor of reg:squarederror.
11.9s	3	[0] validation_0-rmse:32605.13860 validation_1-rmse:31657.15907
14.1s	4	[100] validation_0-rmse:12581.21569 validation_1-rmse:11743.75114
16.2s	5	[200] validation_0-rmse:5835.12466 validation_1-rmse:5365.67709
18.3s	6	[300] validation_0-rmse:3915.75557 validation_1-rmse:4020.67023
20.4s	7	[400] validation_0-rmse:3443.16468 validation_1-rmse:3853.40423
22.6s	8	[500] validation_0-rmse:3285.33804 validation_1-rmse:3805.30176
24.7s	9	[600] validation_0-rmse:3201.92936 validation_1-rmse:3772.44933
26.8s	10	[700] validation_0-rmse:3148.14225 validation_1-rmse:3750.91108
28.9s	11	[800] validation_0-rmse:3109.24248 validation_1-rmse:3733.89713
31.1s	12	[900] validation_0-rmse:3079.40079 validation_1-rmse:3725.61224
33.2s	13	[999] validation_0-rmse:3052.73503 validation_1-rmse:3722.92257
35.8s	14	RMSE Score on Test set: 3721.75
35.8s	15	R-squared (R2) Score: 0.6670230260104328
38.8s	16	/opt/conda/lib/python3.10/site-packages/traitlets/traitlets.py:2930: FutureWarning: --Exporter.preprocessors=["remove_papermill_header.RemovePapermillHeader"] for containers is ed in 0. You can pass `--Exporter.preprocessors item` ... multiple times to add items to a list.
38.8s	17	warn(
38.8s	18	[NbConvertApp] WARNING | Config option `kernel_spec_manager_class` not recognized by `NbConvertApp`.
38.8s	19	[NbConvertApp] Converting notebook __notebook_.ipynb to notebook
39.3s	20	[NbConvertApp] Writing 428033 bytes to__notebook__.ipynb
41.0s	21	/opt/conda/lib/python3.10/site-packages/traitlets/traitlets.py:2930: FutureWarning: --Exporter.preprocessors=["nbconvert.preprocessors.ExtractOutputPreprocessor"] for containers is deprecated in traitlets 5.0. You can pass `--Exporter.preprocessors item` ... multiple times to add items to a list.
41.0s	22	warn(
41.0s	23	[NbConvertApp] WARNING | Config option `kernel_spec_manager_class` not recognized by `NbConvertApp`.
41.0s	24	[NbConvertApp] Converting notebook __notebook_.ipynb to html
41.9s	25	[NbConvertApp] Support files will be in__results____files/
41.9s	26	[NbConvertApp] Making directory__results_files
41.9s	27	[NbConvertApp] Making directory__results_files
41.9s	28	[NbConvertApp] Making directory__results_files
41.9s	29	[NbConvertApp] Making directory__results_files
41.9s	30	[NbConvertApp] Making directory__results_files

```
41.9s      31   [NbConvertApp] Making directory__results_files
41.9s      32   [NbConvertApp] Writing 314131 bytes to__results_.html
```

Conclusion

In conclusion, the measure-based consumption system using AI represents a significant step forward in the efficient utilization of resources and the promotion of sustainability. By harnessing the capabilities of artificial intelligence, we have developed a powerful tool that not only measures consumption but also empowers individuals and organizations to make informed decisions and take steps toward a more sustainable and cost-effective future. This project demonstrates the transformative potential of AI in addressing critical challenges related to resource consumption and environmental conservation