

NAAN MUTHALVAN

IBM COLLABARATE

ARTIFICIAL INTELLIGENCE

PROJECT TITLE

MEASURE ENERGY CONSUMPTION

NAME : MUTHULAKSHMI R

DEPT & YEAR : CSE & III yr

REG.NO : 712221104009

COLLEGE : PARK COLLEGE OF ENGINEERING AND  
TECHNOLOGY

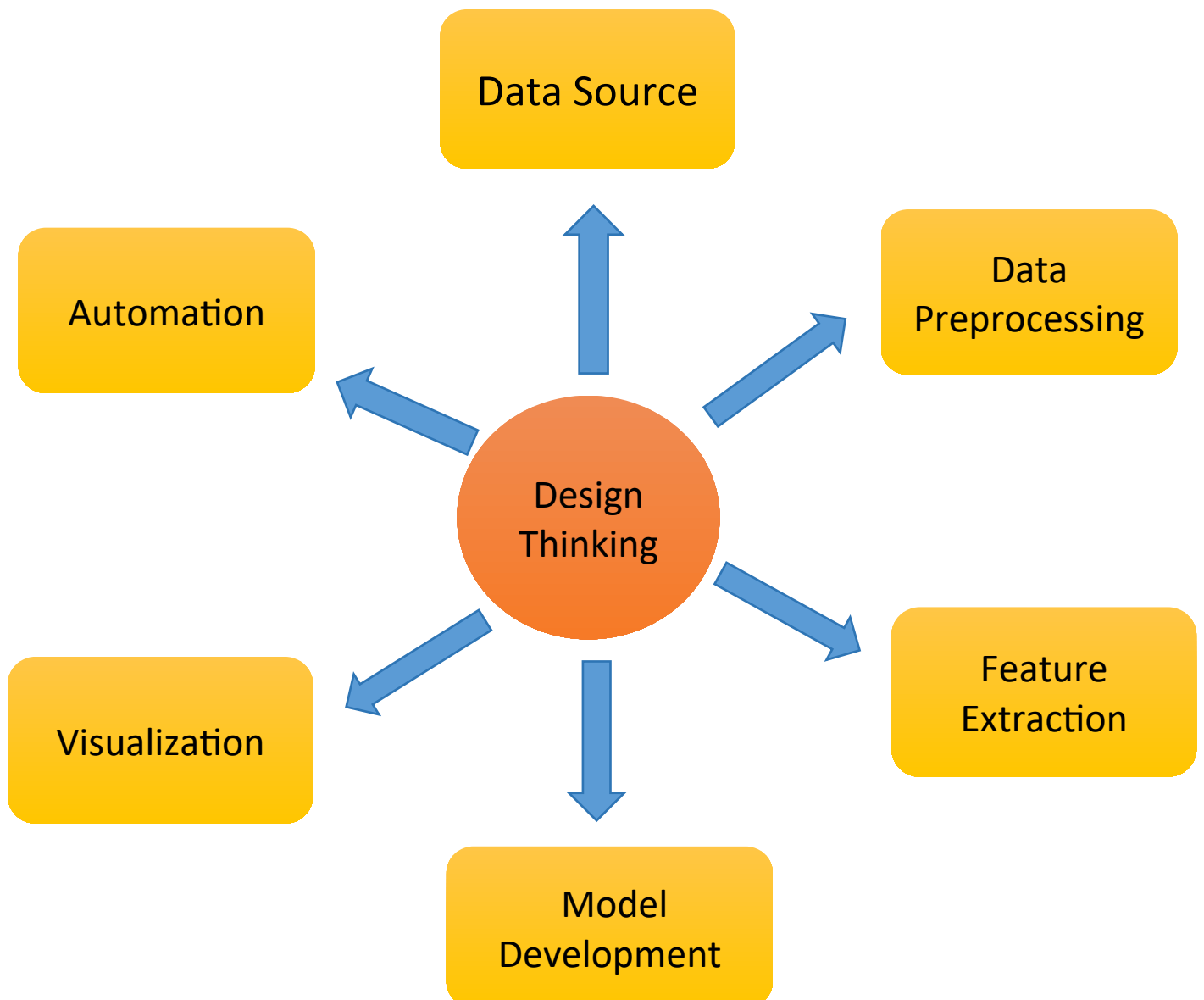
# PHASE 1

## PROBLEM DEFINITION AND DESIGN THINKING

### PROBLEM DEFINITION

The problem at hand is to create an automated system that measures energy consumption, analysis 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



## PHASE 2

### PROJECT INNOVATION IDEA

Steps :

1. Data Collection
2. Data Preprocessing
3. Build Machine Learning Model
4. Create Real Time Monitoring System
5. Build User Engagement
6. Data Analytics And Data Visualization
7. Deployment the Project

# PHASE 3

## DATA ANALYSIS AND PREPROCESSING

### STEP 1:

#### Import library :

The first step is import the library files.

The library files are numpy for array calculation, pandas for data visualization, matplotlib for plotting, seaborn for statics graphics and also load the dataset using pandas.

### STEP 2:

#### Reformat the Date Time Columns

Hence ,the large amout of dataset are reformated for required analysis and also include date time colums for time series process.

### STEP 3:

#### Cleaning the dataset

The data cleaning process is must for analysis because the dataset have unordered ,anamoly and other soure data .

Hence the data cleaning process is used to reduce this problem.

## **STEP 4 :**

### **Transforming the data**

The data are reformatted for best analysis .this reduce the analysis process in less time.

## **STEP 5:**

### **Show the Energy Consumption Each Year**

The main process of analysis are done this section.

The time series analysis is used to analyse energy consumption in each year.

Hence the analysis are shown by statistical plotting perspective.

## **STEP 6:**

### **Data validation**

The final process of data preprocessing is validate the data and show the data using plotting.

# PHASE 4

## BUILDING MODEL FOR ENERGY PREDICTION

- Hence, we are going to build our prediction model algorithm for measure energy consumption.
- It consist the process of loading the dataset, data transformation, feature importance, train/test split, Visualize feature to target relationship, modeling, Forecast on Test, Outlier Analysis, Reviewing : Train/Test Split, Feature Horizon, Lag Features, Train Using Cross Validation, Fold Analysis, Retraining on all Data and Predicting Future.

# 1.IMPORTING LIBRARIES AND DATA SET LOADING

## CODE:

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import seaborn as sns
import matplotlib.pyplot as plt
import xgboost as xgb
```

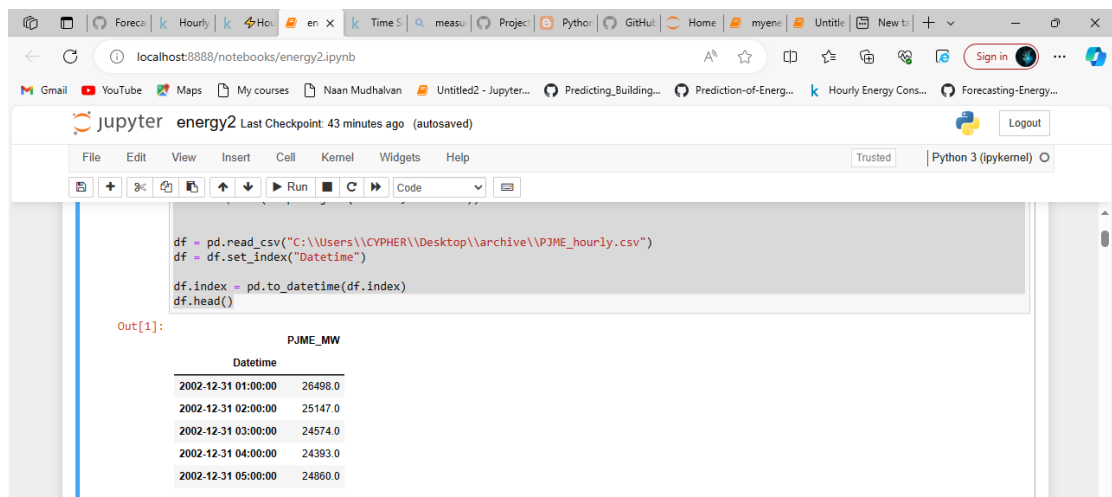
```
from sklearn.metrics import mean_squared_error
```

```
import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

```
df = pd.read_csv("C:\\Users\\CYPHER\\Desktop\\archive\\
PJME_hourly.csv")
df = df.set_index("Datetime")
```

```
df.index = pd.to_datetime(df.index)
df.head()
```

## O/P:



The screenshot shows a Jupyter Notebook window titled 'energy2' with a 'Last Checkpoint: 43 minutes ago (autosaved)' status. The notebook is running on 'Python 3 (ipykernel)'. The code cell contains the following Python code:

```
df = pd.read_csv("C:\\Users\\CYPHER\\Desktop\\archive\\PJME_hourly.csv")
df = df.set_index("Datetime")
df.index = pd.to_datetime(df.index)
df.head()
```

The output of the code is displayed below the cell, labeled 'Out[1]:'. It shows the first five rows of the 'PJME\_MW' dataset, with the index 'Datetime' and the values for 'PJME\_MW'.

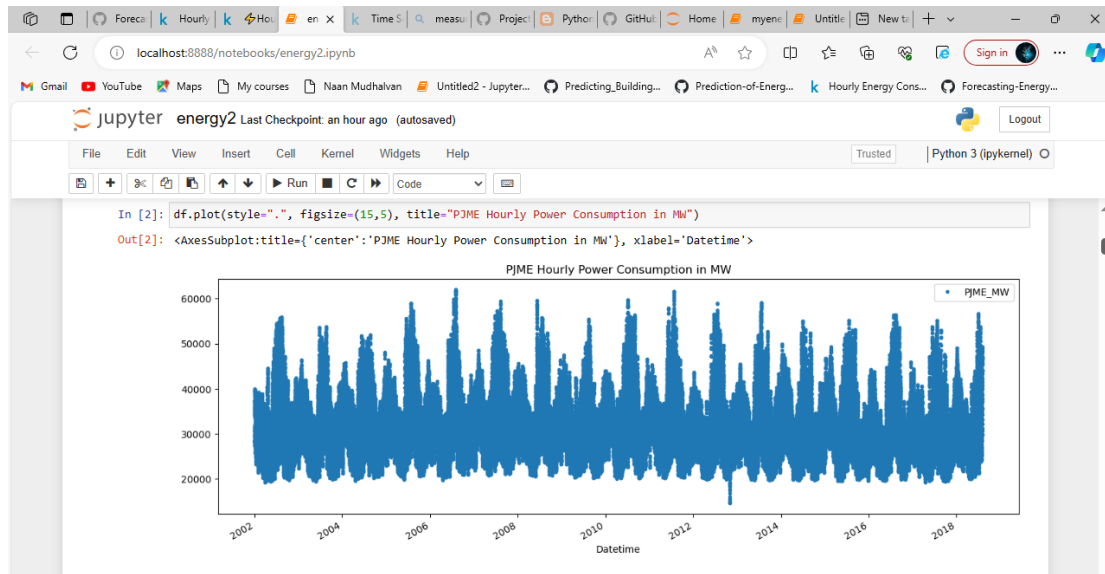
Datetime	PJME_MW
2002-12-31 01:00:00	26498.0
2002-12-31 02:00:00	25147.0
2002-12-31 03:00:00	24574.0
2002-12-31 04:00:00	24393.0
2002-12-31 05:00:00	24860.0



## 2.TRAIN TEST SPLIT

### CODE :

```
df.plot(style=".", figsize=(15,5), title="PJME Hourly Power Consumption in MW")
```



```
train = df.loc[df.index < '01-01-2015']
```

```
test = df.loc[df.index >= '01-01-2015']
```

```
fig, ax = plt.subplots(figsize=(15,5))
```

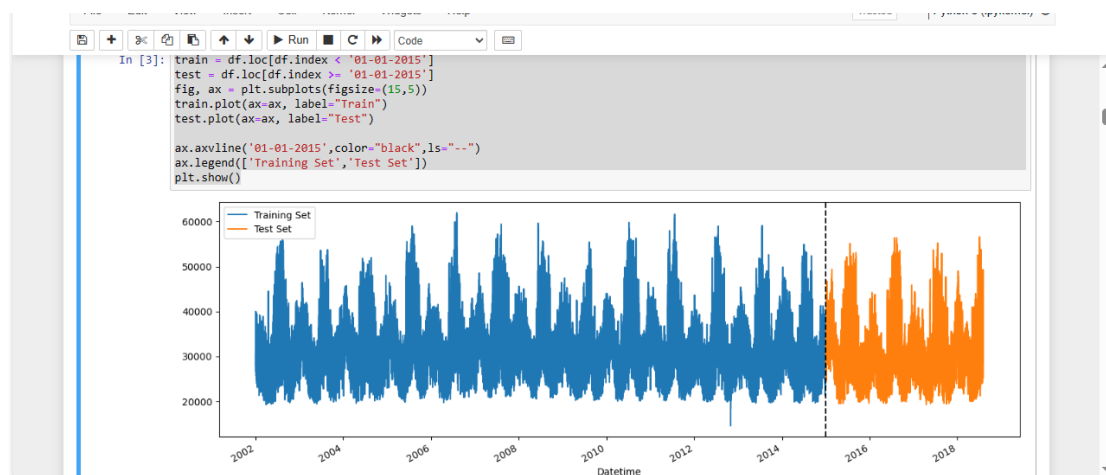
```
train.plot(ax=ax, label="Train")
```

```
test.plot(ax=ax, label="Test")
```

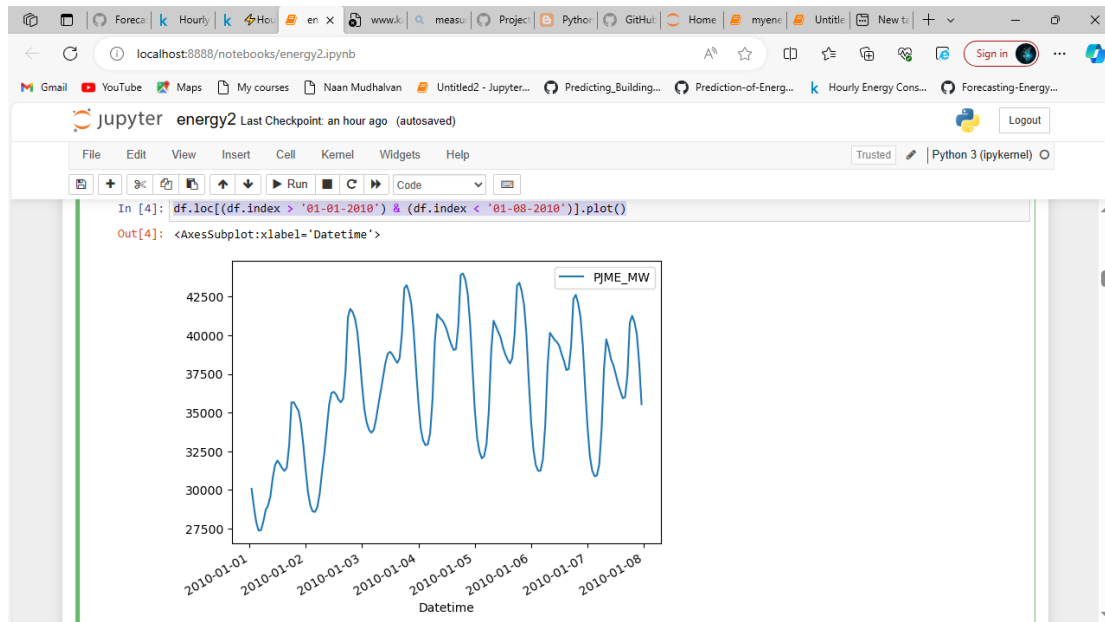
```
ax.axvline('01-01-2015',color="black",ls="--")
```

```
ax.legend(['Training Set','Test Set'])
```

```
plt.show()
```



```
df.loc[(df.index > '01-01-2010') & (df.index < '01-08-2010')].plot()
```



### 3. FEATURE CREATION

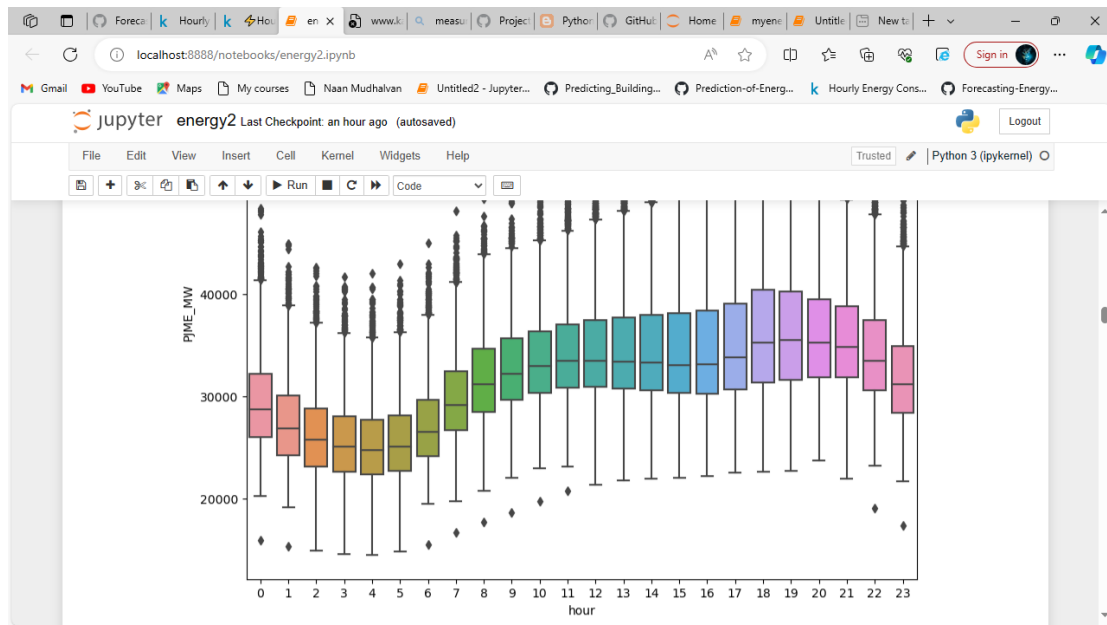
```
def create_time_series_features(dataframe):  
    df = dataframe.copy()  
    df['hour'] = df.index.hour  
    df['dayofweek'] = df.index.dayofweek  
    df['quarter'] = df.index.quarter  
    df['month'] = df.index.month  
    df['year'] = df.index.year  
    df['dayofyear'] = df.index.dayofyear  
  
    return df
```

```
df = create_time_series_features(df)
```

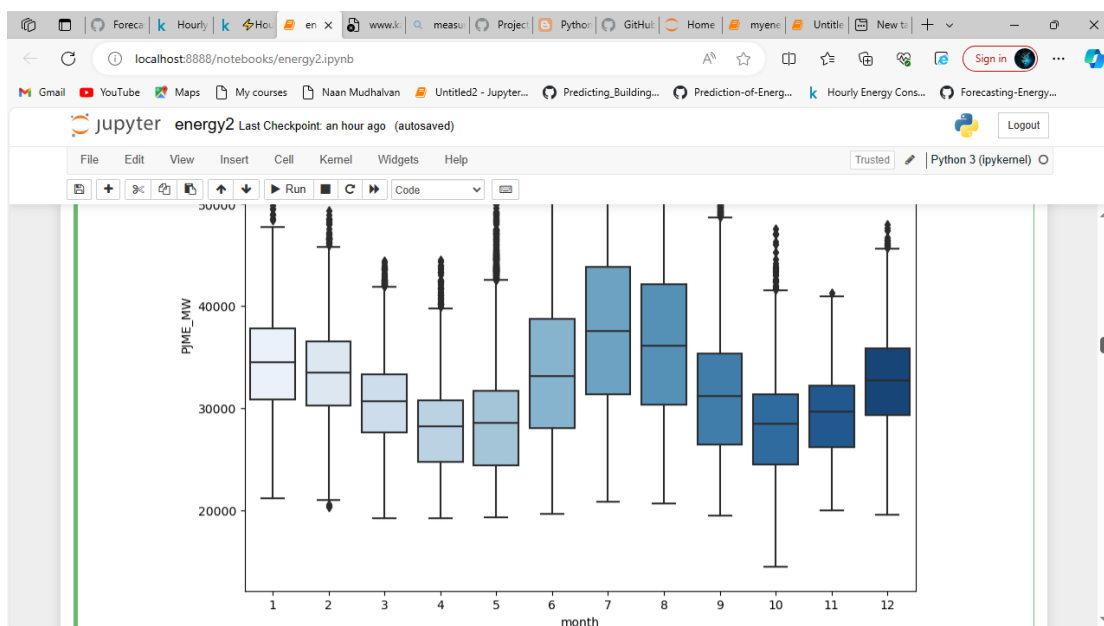
### 4. VISUALIZE FEATURE TO TARGET RELATIONSHIP

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

```
plt.show()
```

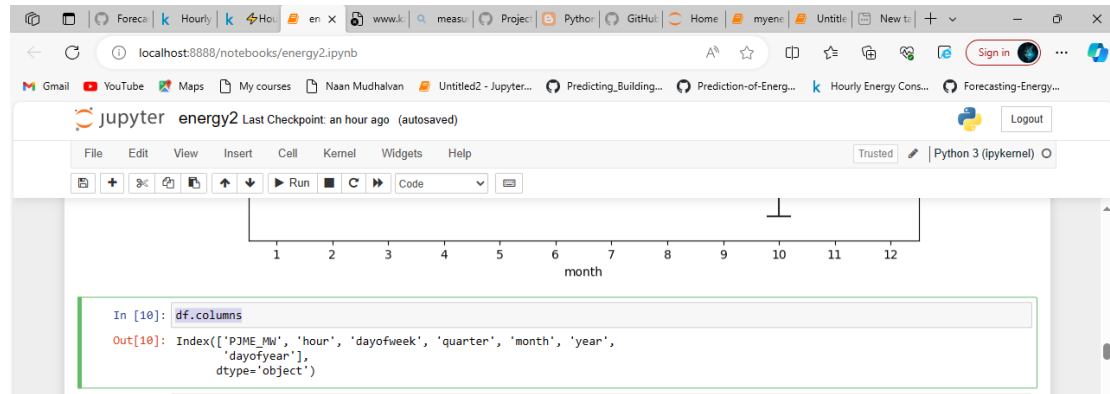


```
fig, ax = plt.subplots(figsize=(10,8))  
sns.boxplot(data=df, x="month",y="PJME_MW",palette="Blues")  
ax.set_title("MW By Month")  
plt.show()
```



## 5. MODELING

df.columns



```
FEATURES = ['hour', 'dayofweek', 'quarter', 'month', 'year',  
            'dayofyear']
```

```
OUTPUT = ['PJME_MW']
```

```
train = create_time_series_features(train)
```

```
test = create_time_series_features(test)
```

```
X_train = train[FEATURES]
```

```
y_train = train[OUTPUT]
```

```
X_test = test[FEATURES]
```

```
y_test = test[OUTPUT]
```

```
reg =
```

```
xg.XGBRegressor(n_estimators=1000,early_stopping_rounds=5  
0, learning_rate=0.01)
```

```
reg.fit(
```

```
    X_train,
```

```
    y_train,
```

```
    eval_set=[(X_train, y_train),(X_test, y_test)],
```

```
    verbose=100
```

```
)
```

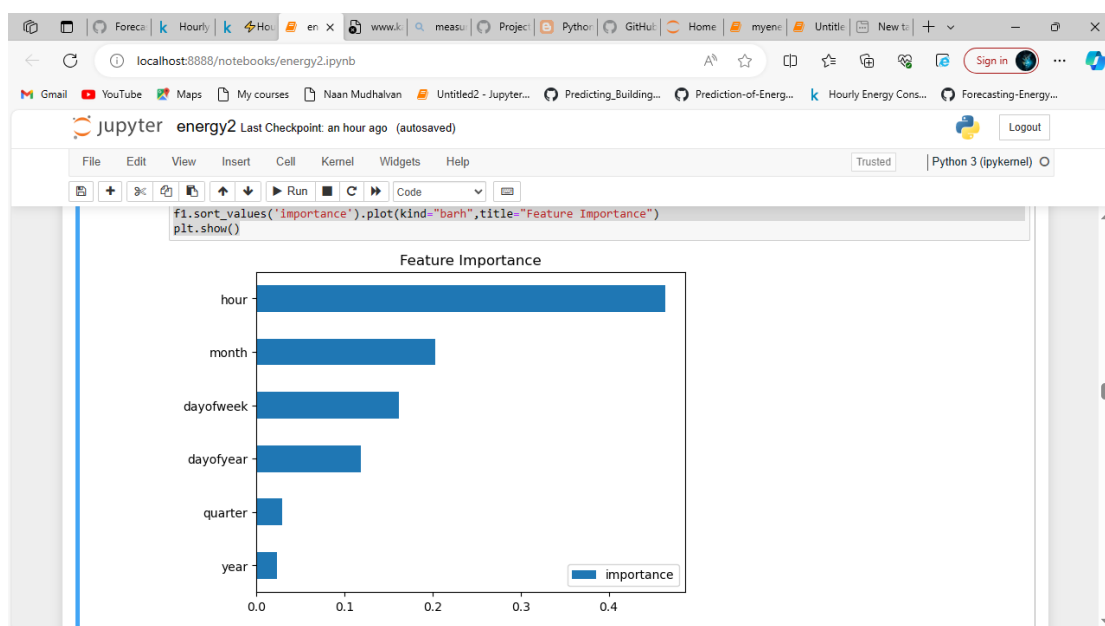
```
reg = xg.XGBRegressor(n_estimators=1000,early_stopping_rounds=50, learning_rate=0.01)
reg.fit(
    X_train,
    y_train,
    eval_set=[(X_train, y_train),(X_test, y_test)],
    verbose=100
)
```

[0]	validation_0-rmse:6407.35736	validation_1-rmse:6479.81619
[100]	validation_0-rmse:3911.97994	validation_1-rmse:4312.03224
[200]	validation_0-rmse:3244.38509	validation_1-rmse:3864.56545
[300]	validation_0-rmse:2996.08999	validation_1-rmse:3748.76687
[400]	validation_0-rmse:2830.28024	validation_1-rmse:3744.93340
[417]	validation_0-rmse:2801.66222	validation_1-rmse:3749.26889

```
Out[12]: XGBRegressor(base_score=None, booster=None, callbacks=None,
    colsample_bylevel=None, colsample_bynode=None,
    colsample_bytree=None, device=None, early_stopping_rounds=50,
    enable_categorical=False, eval_metric=None, feature_types=None,
    gamma=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=None, max_leaves=None,
    min_child_weight=None, missing=nan, monotone_constraints=None,
    multi_strategy=None, n_estimators=1000, n_jobs=None,
    num_parallel_tree=None, random_state=None, ...)
```

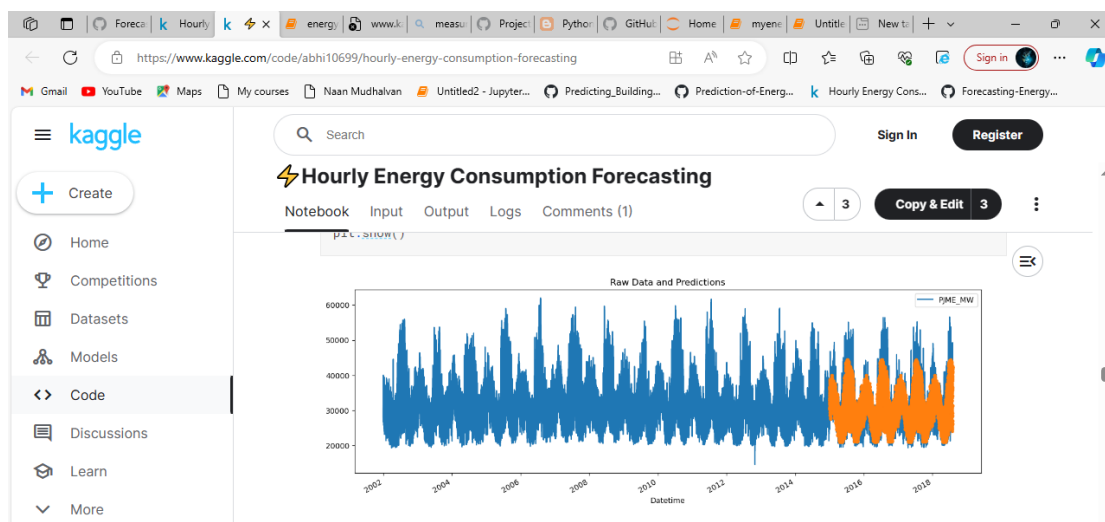
## 6.FEATURE IMPORTANCE

```
f1 = pd.DataFrame(data=reg.feature_importances_,
index=reg.feature_names_in_, columns=['importance'])
f1.sort_values('importance').plot(kind="barh",title="Feature
Importance")
plt.show()
```

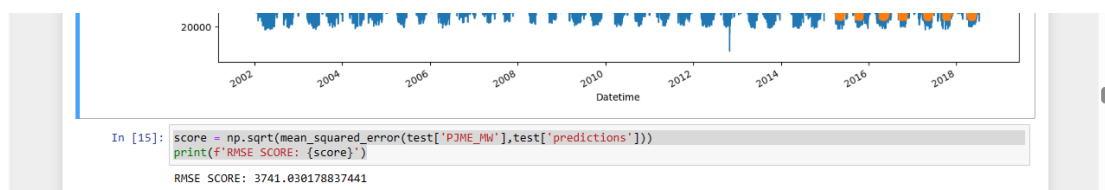


## 7.FEATURE FORECAST ON TEST

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



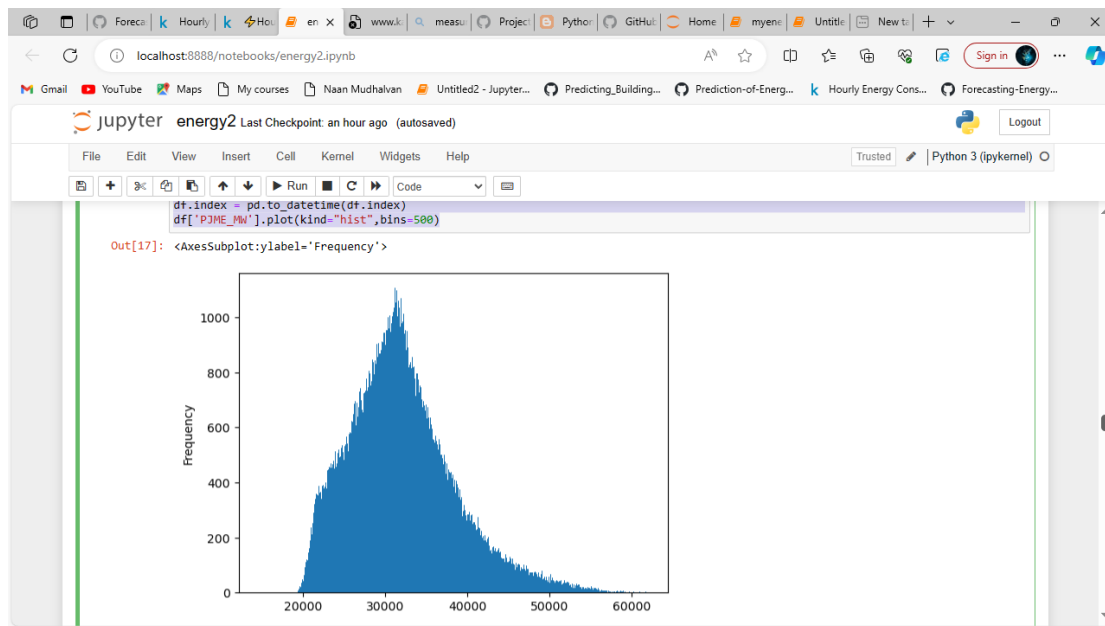
```
score =
np.sqrt(mean_squared_error(test['PJME_MW'],test['predictions']))
print(f'RMSE SCORE: {score}')
```



## 8. OUTLIER ANALYSIS

```
df = pd.read_csv("C:\\Users\\CYPHER\\Desktop\\archive\\PJME_hourly.csv")
df = df.set_index("Datetime")

df.index = pd.to_datetime(df.index)
df['PJME_MW'].plot(kind="hist",bins=500)
```



```
df = df.query('PJME_MW > 19_000').copy()
```

## 9. REVIEWING TRAIN AND TEST SPLIT

```
from sklearn.model_selection import TimeSeriesSplit
tss = TimeSeriesSplit(n_splits=5, test_size=24*365*1,gap=24)
df = df.sort_index()
fig, axs = plt.subplots(5,1,figsize=(15,35))

fold = 0
for train_idx, val_idx in tss.split(df):

    train = df.iloc[train_idx]
```

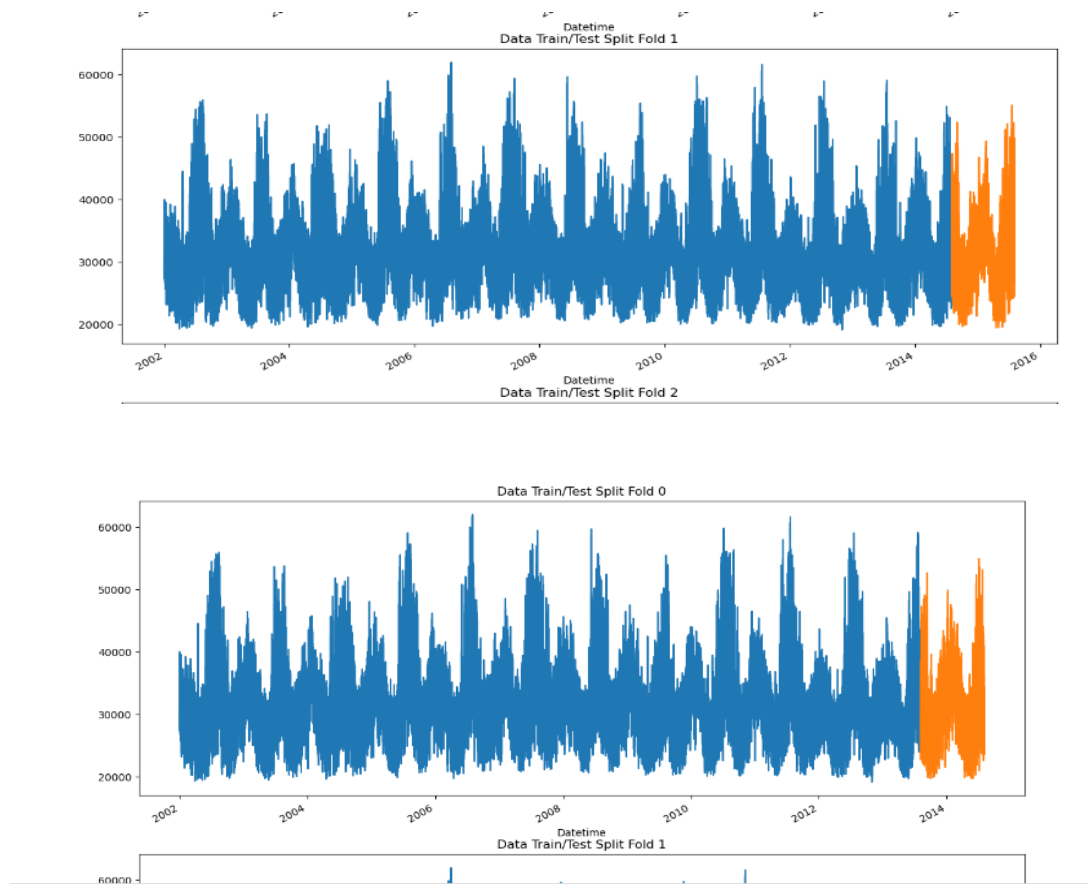
```

test = df.iloc[val_idx]

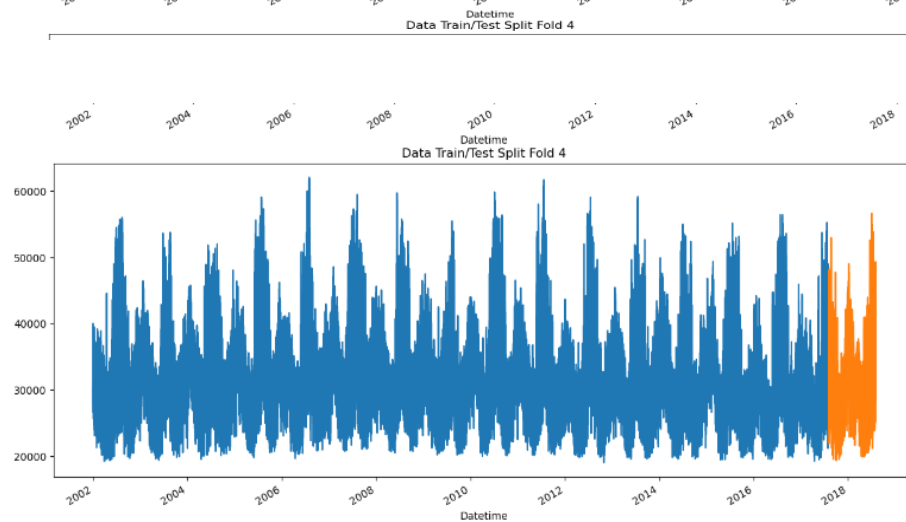
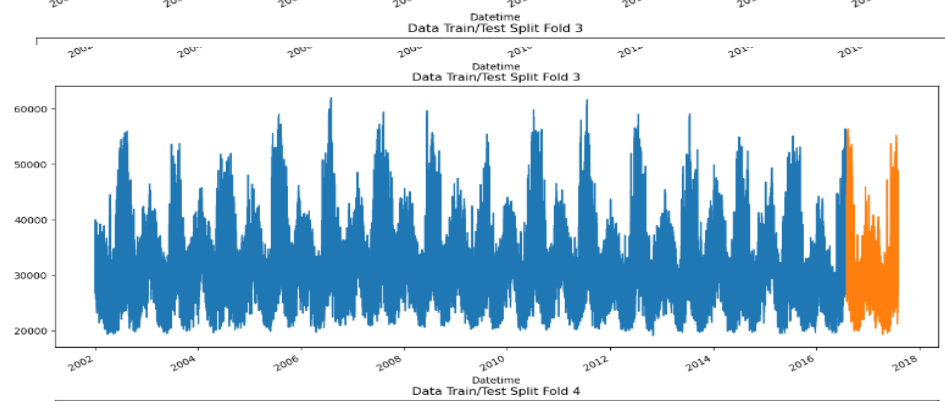
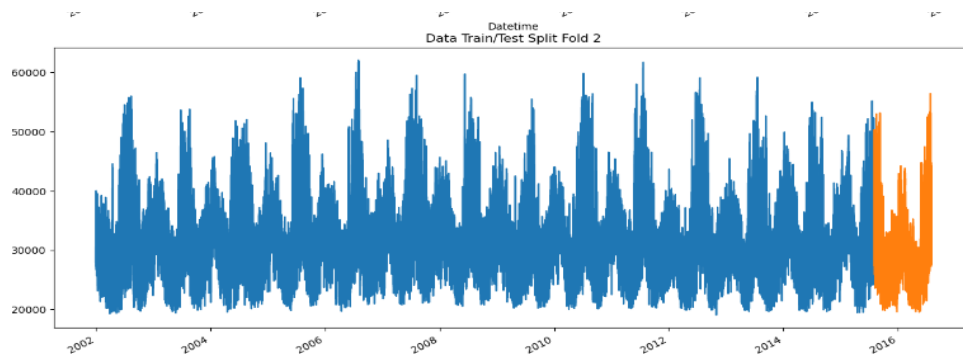
train['PJME_MW'].plot(
    ax=axes[fold],
    label="Training Set",
    title=f"Data Train/Test Split Fold {fold}"
)
test['PJME_MW'].plot(
    ax=axes[fold],
    label="Test Set",
)

fold += 1

```







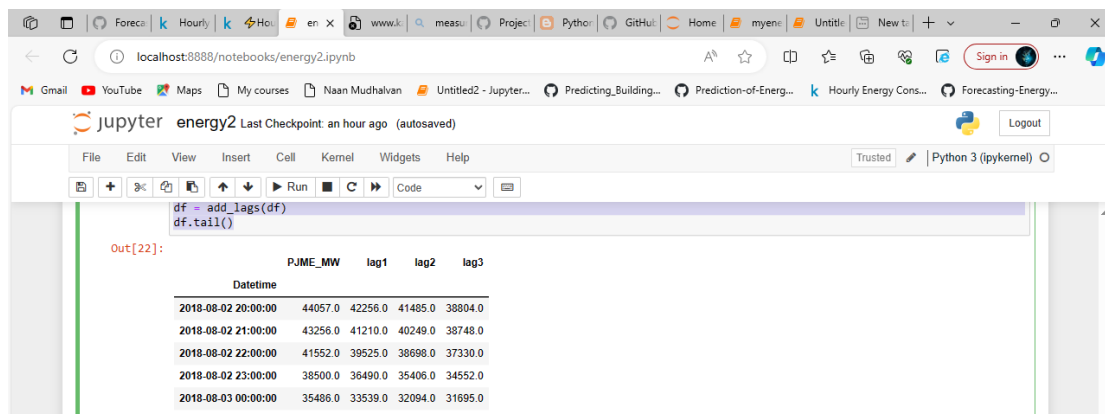
## 10. FEATURE HORIZON AND LAG FEATURES

```
df = create_time_series_features(df)
```

```
target_map = df['PJME_MW'].to_dict()
def add_lags(dframe):
    df = dframe.copy()
    df['lag1'] = (df.index - pd.Timedelta('364
days')).map(target_map)
    df['lag2'] = (df.index - pd.Timedelta('728
days')).map(target_map)
    df['lag3'] = (df.index - pd.Timedelta('1092
days')).map(target_map)

    return df
```

```
df = add_lags(df)
df.tail()
```



The screenshot shows a Jupyter Notebook window titled 'energy2' with a Python 3 (ipykernel) environment. The code cell contains the following lines:

```
df = add_lags(df)
df.tail()
```

The output of the code cell is displayed as a table with 5 rows and 5 columns. The columns are labeled 'PJME\_MW', 'lag1', 'lag2', and 'lag3'. The first column is labeled 'Datetime'.

Datetime	PJME_MW	lag1	lag2	lag3
2018-08-02 20:00:00	44057.0	42256.0	41485.0	38804.0
2018-08-02 21:00:00	43256.0	41210.0	40249.0	38748.0
2018-08-02 22:00:00	41552.0	39525.0	38698.0	37330.0
2018-08-02 23:00:00	38500.0	36490.0	35406.0	34552.0
2018-08-03 00:00:00	35486.0	33539.0	32094.0	31695.0

## 11. TRAIN USING CROSS VALIDATION

```
tss = TimeSeriesSplit(n_splits=5, test_size=24*365*1,gap=24)
df = df.sort_index()
df.columns
```

```
2018-08-02 23:00:00    38500.0    36490.0    35406.0    34552.0
2018-08-03 00:00:00    35486.0    33539.0    32094.0    31695.0

In [23]: tss = TimeSeriesSplit(n_splits=5, test_size=24*365*1,gap=24)
df = df.sort_index()
df.columns

Out[23]: Index(['PJME_MW', 'lag1', 'lag2', 'lag3'], dtype='object')
```

```
fold = 0
preds = []
scores = []
```

```
for train_idx, val_idx in tss.split(df):
    train = df.iloc[train_idx]
    test = df.iloc[val_idx]
```

```
train = create_time_series_features(train)
test = create_time_series_features(test)
```

```
FEATURES = ['hour', 'dayofweek', 'quarter', 'month', 'year',
'dayofyear',
    'lag1', 'lag2', 'lag3']
```

```
OUTPUT = 'PJME_MW'
```

```
OUTPUT = 'PJME_MW'
```

```
X_train = train[FEATURES]
y_train = train[OUTPUT]
```

```

X_test = test[FEATURES]
y_test = test[OUTPUT]
reg = xg.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
)

y_pred = reg.predict(X_test)
preds.append(y_pred)
score = np.sqrt(mean_squared_error(y_test, y_pred))
scores.append(score)

```

```

score = np.sqrt(mean_squared_error(y_test, y_pred))
scores.append(score)

[0] validation_0-rmse:32732.49608 validation_1-rmse:31956.60163

C:\Users\CYPHER\AppData\Roaming\Python\Python39\site-packages\xgboost\core.py:160: UserWarning: [16:17:44] WARNING: C:\buildkit-e-agent\builds\buildkite-windows-cpu-autoscaling-group-1-07f6e447eee219473-1\xgboost\xgboost-ci-windows\src\objective\regression_obj.cc:209: reg:linear is now deprecated in favor of reg:squarederror.
  warnings.warn(msg, UserWarning)

[100] validation_0-rmse:12532.64369 validation_1-rmse:11006.14134
[200] validation_0-rmse:5747.92495 validation_1-rmse:5359.26490
[300] validation_0-rmse:3872.48134 validation_1-rmse:3980.86965
[400] validation_0-rmse:3434.23853 validation_1-rmse:3762.33705
[500] validation_0-rmse:3369.34730 validation_1-rmse:3764.82810
[600] validation_0-rmse:32672.16678 validation_1-rmse:32138.89241

C:\Users\CYPHER\AppData\Roaming\Python\Python39\site-packages\xgboost\core.py:160: UserWarning: [16:17:53] WARNING: C:\buildkit-e-agent\builds\buildkite-windows-cpu-autoscaling-group-1-07f6e447eee219473-1\xgboost\xgboost-ci-windows\src\objective\regression_obj.cc:209: reg:linear is now deprecated in favor of reg:squarederror.
  warnings.warn(msg, UserWarning)

[100] validation_0-rmse:12513.65574 validation_1-rmse:12224.93373
[200] validation_0-rmse:5753.34937 validation_1-rmse:5662.07107
[300] validation_0-rmse:3902.71304 validation_1-rmse:3933.73076
[400] validation_0-rmse:3476.90515 validation_1-rmse:3590.55005
[500] validation_0-rmse:3353.72424 validation_1-rmse:3516.39915
[600] validation_0-rmse:3297.94766 validation_1-rmse:3481.94003
[700] validation_0-rmse:3258.48267 validation_1-rmse:3461.37383
[800] validation_0-rmse:3221.51553 validation_1-rmse:3436.49603

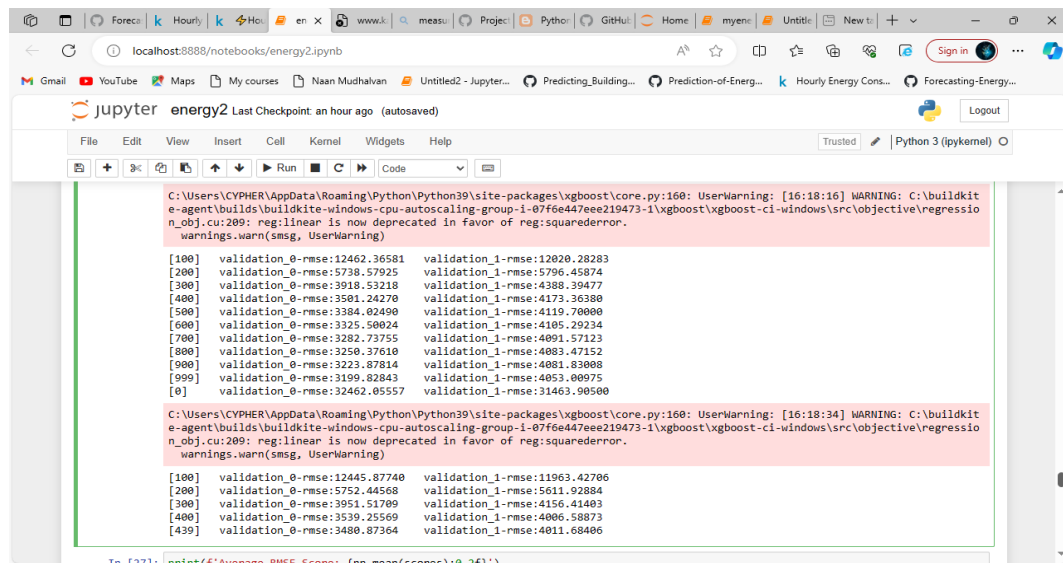
```

```

warnings.warn(msg, UserWarning)

[100] validation_0-rmse:12513.65574 validation_1-rmse:12224.93373
[200] validation_0-rmse:5753.34937 validation_1-rmse:5662.07107
[300] validation_0-rmse:3902.71304 validation_1-rmse:3933.73076
[400] validation_0-rmse:3476.90515 validation_1-rmse:3590.55005
[500] validation_0-rmse:3353.72424 validation_1-rmse:3516.39915
[600] validation_0-rmse:3297.94766 validation_1-rmse:3481.94003
[700] validation_0-rmse:3258.48267 validation_1-rmse:3461.37383
[800] validation_0-rmse:3221.51553 validation_1-rmse:3436.49603
[900] validation_0-rmse:3190.11480 validation_1-rmse:3428.88699
[999] validation_0-rmse:3166.16314 validation_1-rmse:3420.30469
[0] validation_0-rmse:32631.28370 validation_1-rmse:31073.29733

```



```
C:\Users\CYPHER\AppData\Roaming\Python\Python39\site-packages\xgboost\core.py:160: UserWarning: [16:18:16] WARNING: C:\buildkite-agent\builds\buildkite-windows-cpu-autoscaling-group-1-07f6e447eee219473-1\xgboost\xgboost-ci-windows\src\objective\regression_obj.py:209: reg:linear is now deprecated in favor of reg:squarederror.
warnings.warn(msg, UserWarning)

[100] validation_0-rmse:12462.36581 validation_1-rmse:12020.28283
[200] validation_0-rmse:5738.57925 validation_1-rmse:5796.45874
[300] validation_0-rmse:3918.53218 validation_1-rmse:4388.39477
[400] validation_0-rmse:3501.24270 validation_1-rmse:4173.36380
[500] validation_0-rmse:3384.02490 validation_1-rmse:4119.70000
[600] validation_0-rmse:3325.50024 validation_1-rmse:4105.29234
[700] validation_0-rmse:3282.73755 validation_1-rmse:4091.57123
[800] validation_0-rmse:3250.37610 validation_1-rmse:4083.47152
[900] validation_0-rmse:3223.87814 validation_1-rmse:4081.83008
[999] validation_0-rmse:3199.82843 validation_1-rmse:4053.00975
[0] validation_0-rmse:32462.05557 validation_1-rmse:31463.90500

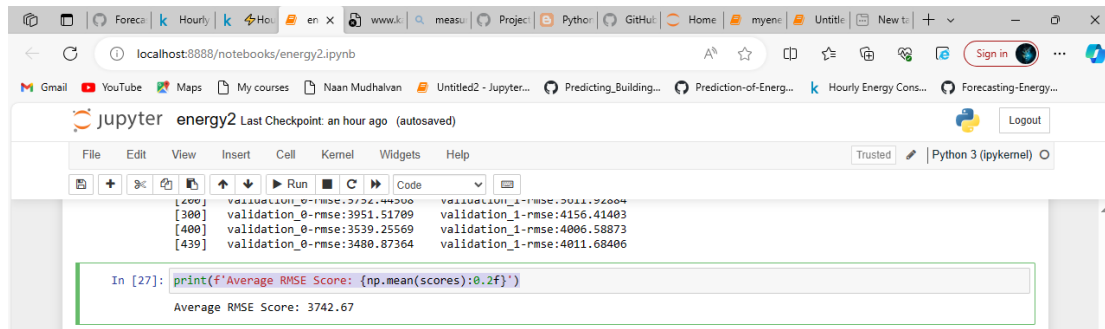
C:\Users\CYPHER\AppData\Roaming\Python\Python39\site-packages\xgboost\core.py:160: UserWarning: [16:18:34] WARNING: C:\buildkite-agent\builds\buildkite-windows-cpu-autoscaling-group-1-07f6e447eee219473-1\xgboost\xgboost-ci-windows\src\objective\regression_obj.py:209: reg:linear is now deprecated in favor of reg:squarederror.
warnings.warn(msg, UserWarning)

[100] validation_0-rmse:12445.87740 validation_1-rmse:11963.42706
[200] validation_0-rmse:5752.44568 validation_1-rmse:5611.92884
[300] validation_0-rmse:3951.51709 validation_1-rmse:4156.41403
[400] validation_0-rmse:3539.25569 validation_1-rmse:4006.58073
[439] validation_0-rmse:3480.87364 validation_1-rmse:4011.68406

In [32]: print(f'Average RMSE Score: {rmse_score}')
print(f'Average RMSE Score: {rmse_score}')
```

## 12.FOLD ANALYSIS & RETRAINING ON ALL DATA

```
print(f'Average RMSE Score: {np.mean(scores):0.2f}')
```



```
df = create_time_series_features(df)
```

```
FEATURES = ['hour', 'dayofweek', 'quarter', 'month', 'year', 'dayofyear',  
            'lag1', 'lag2', 'lag3']
```

```
OUTPUT = 'PJME_MW'
```

```
X_all = df[FEATURES]
```

```
y_all = df[OUTPUT]
```

```
reg = xg.XGBRegressor(  
    base_score=0.5,  
    booster='gbtree',  
    n_estimators=500,  
    early_stopping_rounds=50,  
    objective='reg:linear',  
    max_depth=3,  
    learning_rate=0.01  
)
```

```
reg.fit(X_all, y_all, eval_set=[(X_all, y_all)], verbose=100)
```

The screenshot shows a Jupyter Notebook window titled 'energy2' with a last checkpoint 2 hours ago. The code cell contains the following:

```
reg.fit(X_all, y_all, eval_set=[(X_all, y_all)], verbose=100)
```

The output shows the validation RMSE at intervals of 100 iterations:

```
[0] validation_0-rmse:32403.88991
[100] validation_0-rmse:12426.83220
[200] validation_0-rmse:5751.73275
[300] validation_0-rmse:3971.53256
[400] validation_0-rmse:3571.21833
[499] validation_0-rmse:3456.76877
```

A warning message is displayed in a red box:

```
C:\Users\CYPHER\AppData\Roaming\Python\Python39\site-packages\xgboost\core.py:160: UserWarning: [16:20:10] WARNING: C:\buildkite-agent\builds\buildkite-windows-cpu-autoscaling-group-i-07f6e447eee219473-1\xgboost\xgboost-ci-windows\src\objective\regression_obj.cu:209: reg:linear is now deprecated in favor of reg:squarederror.
warnings.warn(msg, UserWarning)
```

The final output (Out[28]) shows the XGBRegressor parameters:

```
XGBRegressor(base_score=0.5, booster='gbtree', callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None, early_stopping_rounds=50,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=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=-1, monotone_constraints=None,
              multi_strategy=None, n_estimators=500, n_jobs=None,
              num_parallel_tree=None, objective='reg:linear', ...)
```

## 13.PREDICTING FUTURE

df.index.max()

The screenshot shows the same Jupyter Notebook window. The code cell contains the following:

```
df.index.max()
```

The output (Out[29]) shows the maximum index value:

```
Timestamp('2018-08-03 00:00:00')
```

```

future = pd.date_range('2018-08-03','2019-08-03',freq='1h')
future_df = pd.DataFrame(index=future)

future_df['isFuture'] = True

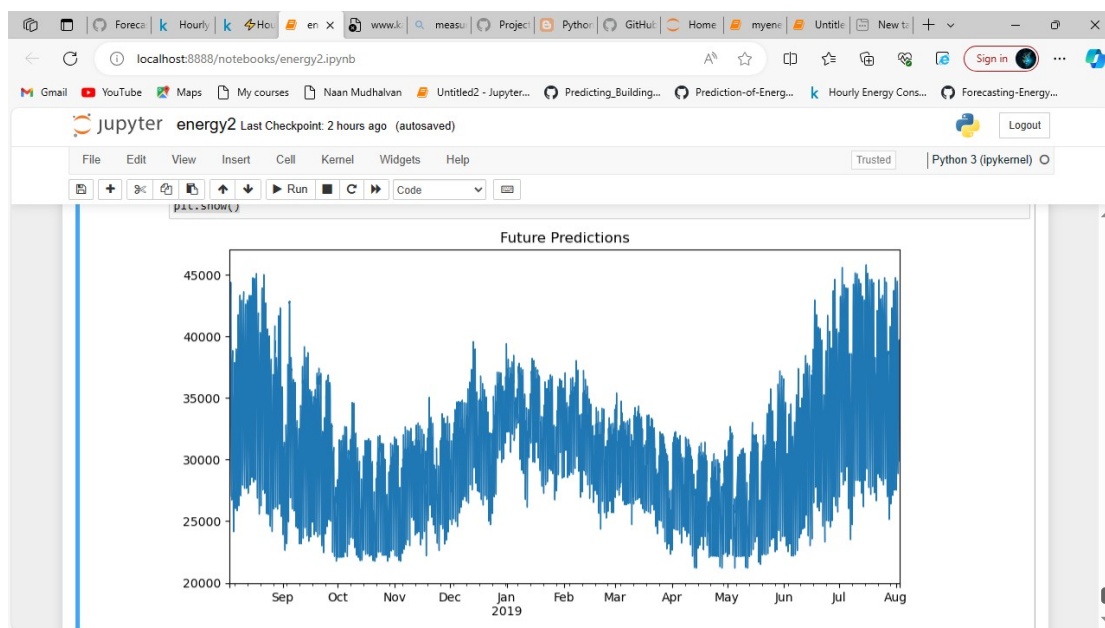
df['isFuture'] = False

df_and_future = pd.concat([df, future_df])

df_and_future = create_time_series_features(df_and_future)
df_and_future = add_lags(df_and_future)
future_w_features = df_and_future.query('isFuture').copy()
future_w_features['pred'] =
reg.predict(future_w_features[FEATURES])
future_w_features['pred'].plot(
    figsize=(10,5),
    ms=1,
    lw=1,
    title="Future Predictions"
)

plt.show()

```





## DEPLOYMENT THE PROJECT

- This is the final process of my project MEASURE ENERGY CONSUMPTION .
- Hence ,my project is running in real world environment using watson cloud or python flask.
- This shows my energy consumption prediction approximately related to original result in graphical manner.
- Then it is finally used for future purposed.