MEASURE ENERGY CONSUMPTION

Phase 3 submission document

TOPIC :

Measure Energy Consumption by Loading and Preprocessing the Dataset

DATA PREPROCESSING

Data preprocessing is the process of transforming raw data into an understandable format. It is also an important step in data mining as we cannot work with raw data. The quality of the data should be checked before applying machine learning or data mining algorithm

DATA PREPROCESSING STEPS

The steps involved in data preprocessing are

- Data quality assessment
- Data cleaning
- Data transformation
- Data reduction

Data quality assessment

The data quality assessment is the application of business approved data quality requirements to a selected data set. Data quality requirements should be expressed in terms of data quality dimensions and should be aligned with organizational objectives. Targets and thresholds should be established for each dimension.

Data cleaning

Data cleaning is the process of fixing or removing incorrect, corrupted, incorrectly formatted, duplicate, or incomplete data within a dataset.

Data transformation

Data transformation is the process of converting data from one format to another, typically from the format of a source system into the required format of a destination system.

Data reduction

Data reduction is a capacity optimization technique in which data is reduced to its simplest possible form to free up capacity on a storage device.

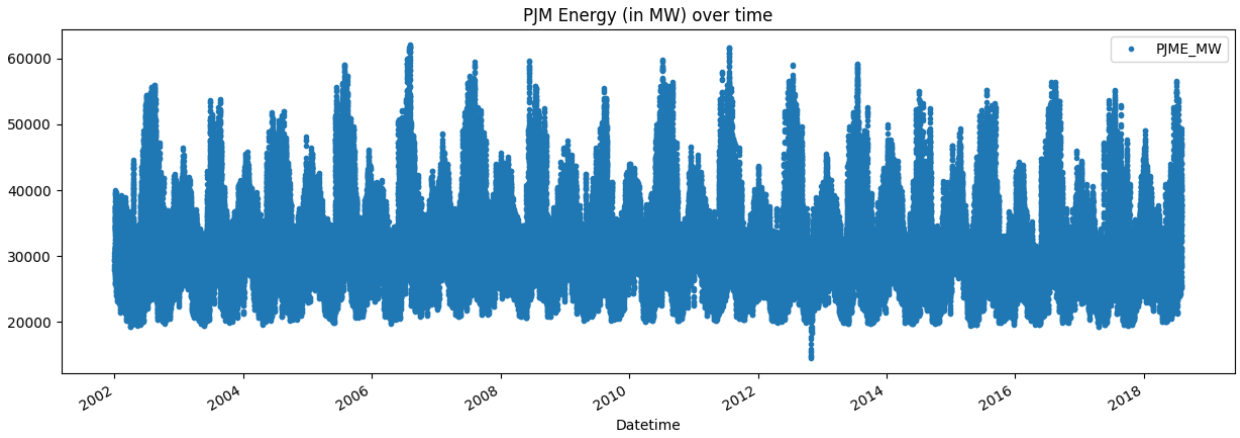
IMPORT LIBRARIES

```
 # import the libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# 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)
```

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()
```

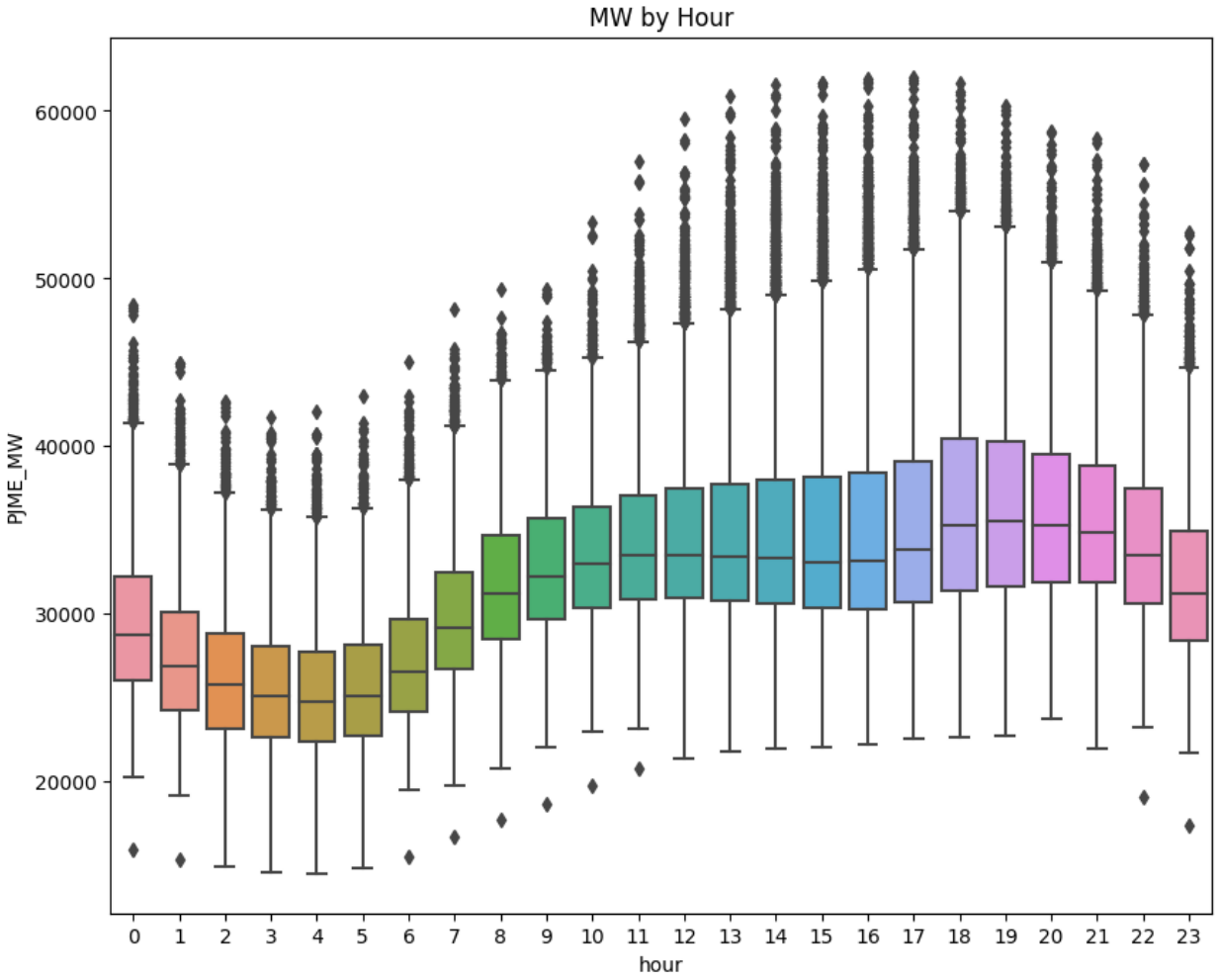


Split the data

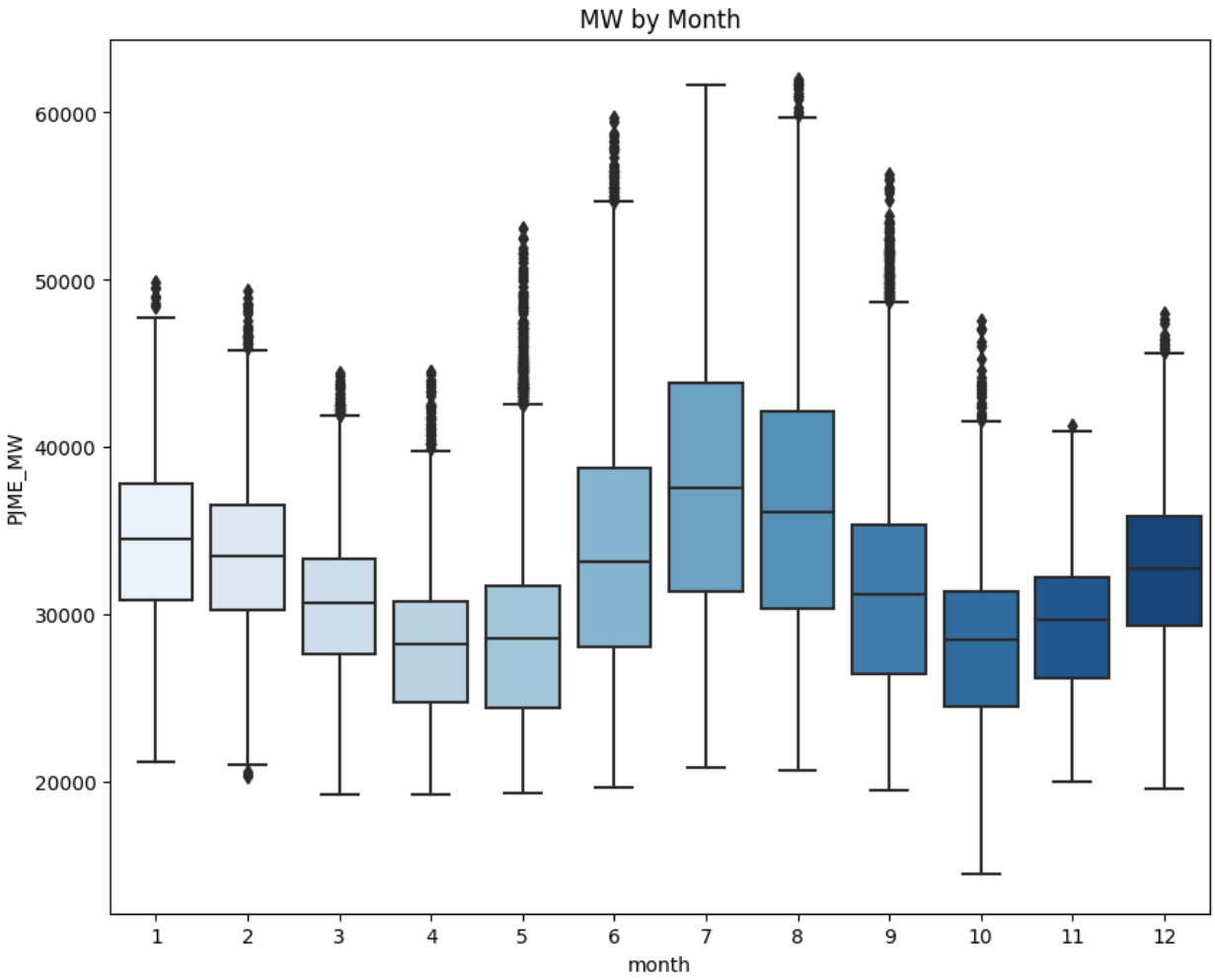
```
# train / test split
train = df.loc[df.index < '01-01-2015']
test = df.loc[df.index >= '01-01-2015']
```

Feature Engineering

```
 # feature creation
def create_features(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()
```




```
# 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()
```



Modelling

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]
```

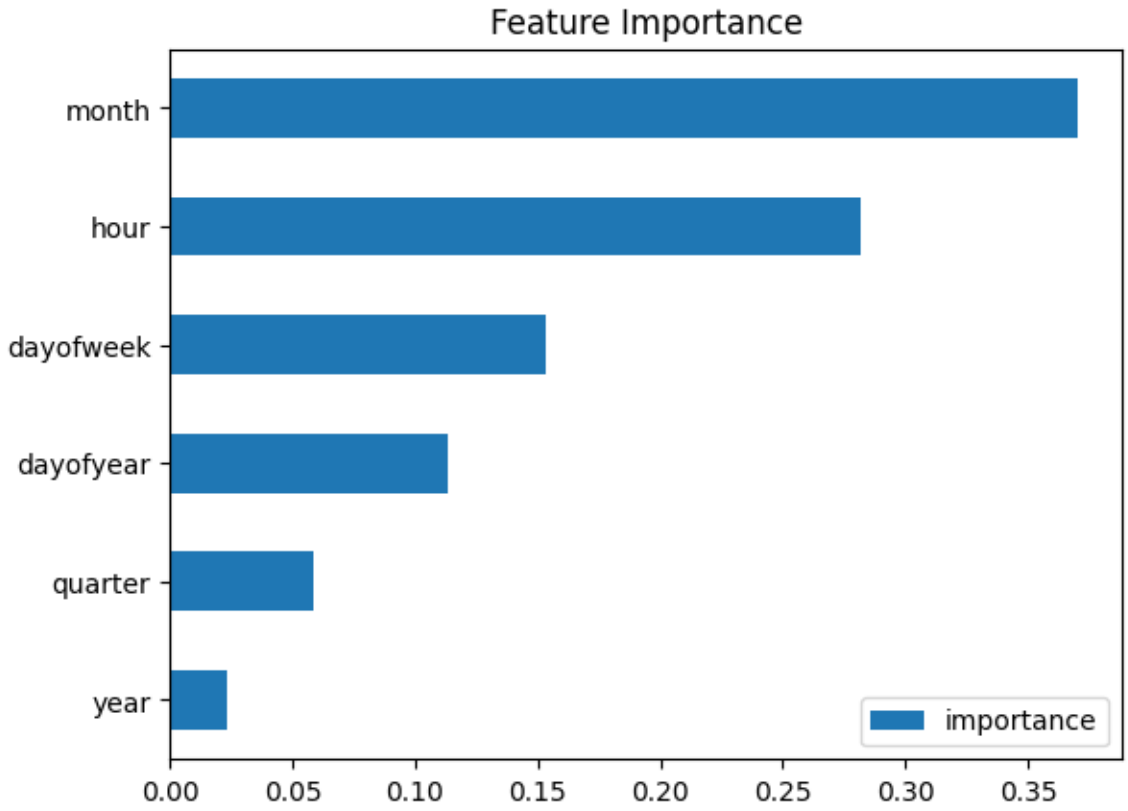
BUILD THE MODEL

```
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_rate=0.01)
reg.fit(X_train, y_train,
```

```
        eval_set=[(X_train, y_train), (X_test, y_test)]
verbose=100)
```

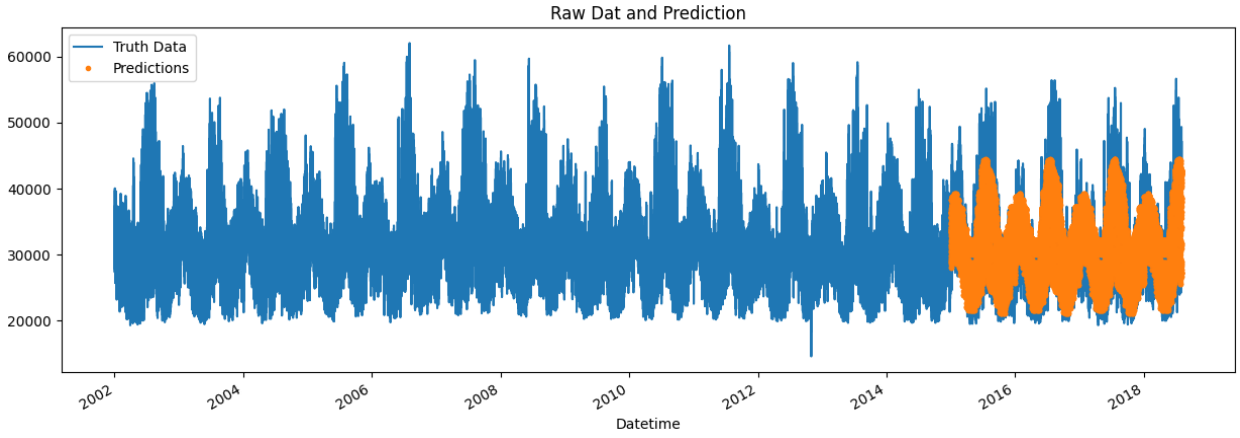
FEATURES IMPORTANCE

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



FORECASTING ON TEST DATA

```
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}')
```

OUTPUT

```
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)
```

OUTPUT

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