MEASURE ENERGY CONSUMPTION

NAME: LINGESH.G

NM ID: aut2221l302

Phase 4 document

Development Part-2

Topic: Building the project by performing different activities like feature engineering, model training, evaluation

INTRODUCTION:-

- ❖ The development of measuring energy consumption is a critical and rapidly evolving field that plays a pivotal role in our efforts to manage resources efficiently, reduce carbon emissions, and promote sustainability.

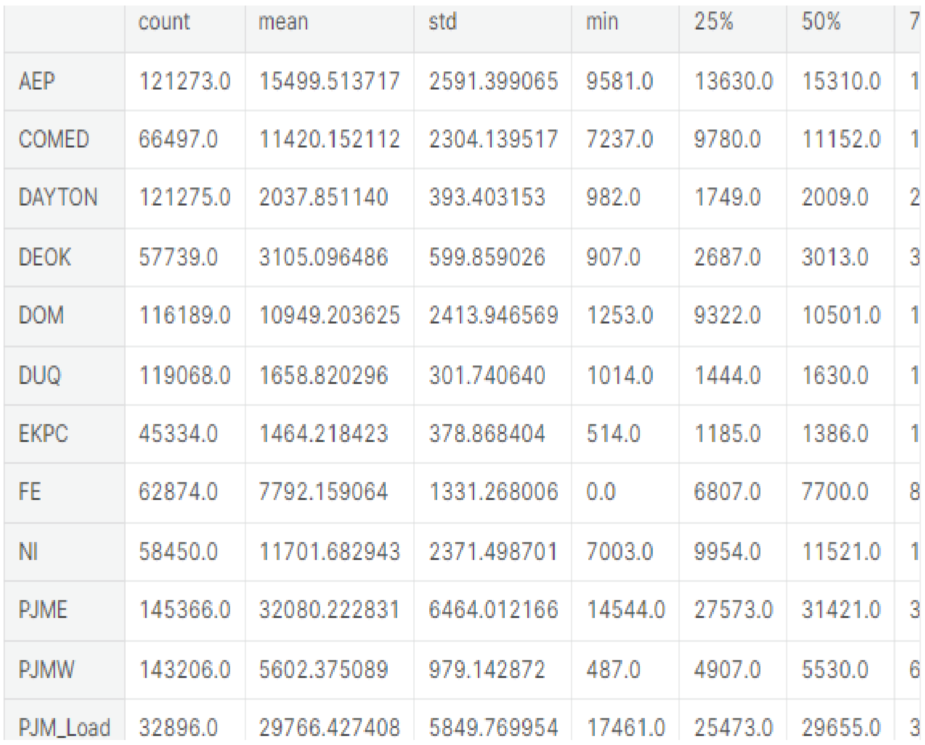
- ❖ Measuring energy consumption has historically been a fundamental practice for both individuals and industries. It provides essential insights into resource allocation, billing, and environmental impact.

- ❖ This journey into the development of measuring energy consumption with steps of **feature engineering, model training, evaluation.**

- ❖ 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.

- ❖ Data Model training helps create a simplified, logical database that eliminates redundancy, reduces storage requirements, and enables efficient retrieval.

Given Dataset:



	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

Overview of the Process:

The following is an overview of the

1. Prepare the data: This includes cleaning the data, removing outliers, and handling missing values.

2. Perform feature selection: This can be done using a variety of methods, such as correlation analysis, information gain, and recursive feature elimination.

3. Train the model: There are many different machine learning algorithms that can be used for measuring energy consumption. Some popular choices include linear regression, random forests, and gradient boosting machines.

4. Evaluate the model: This can be done by calculating the multiply the power in kW by the hours you use the devices per day, per week or per month.

5. Deploy the model: Once the model has been evaluated and found to be performing well, it can be deployed to production so that it can be used to measure energy consumption of households.

PROCEDURE:-

Feature Selection:

1. Identify the target variable: This is the variable that you want to predict, such as energy consumption.

2. Explore the data: This will help you to understand the relationships between the different features and the target variable. You can use data visualization and correlation analysis to identify features that are highly correlated with the target variable.

3. Remove redundant features: If two features are highly correlated with each other, then you can remove one of the features, as they are likely to contain redundant information.

4. Remove irrelevant features: If a feature is not correlated with the target variable, then you can remove it, as it is unlikely to be useful for energy consumption.

In [1]:

```
# 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)
```

#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)
```

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:

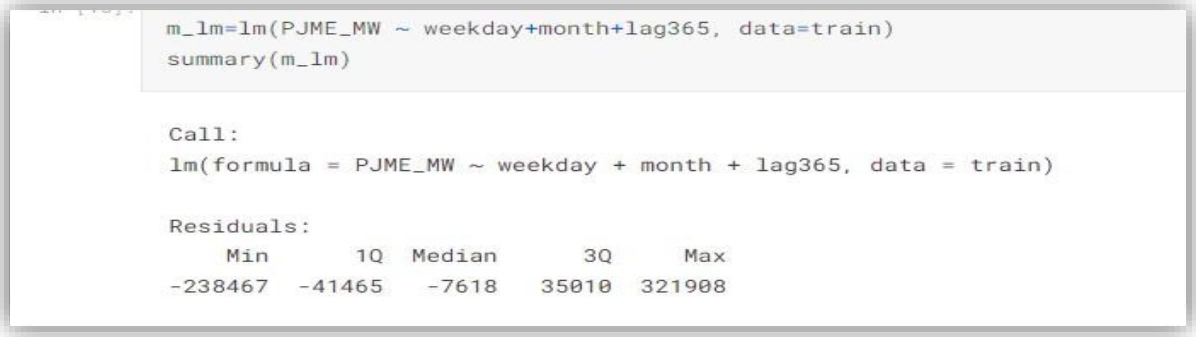
In [3]:

```
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:

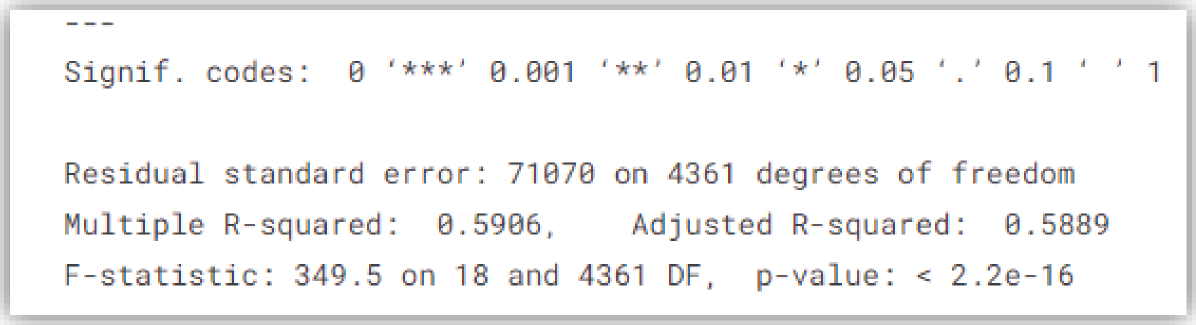
Linear regression is a simple and widely used technique. You would use it to model the relationship between independent variables (such as temperature, time of day, day of the week) and the dependent variable (energy consumption) as a linear equation.

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 training:

- Model training is the process of teaching machine learning model to predict house energy consumption.

- It involves feeding the model historical data on energy consumption and features, such as electricity bill rate, usage.

- Once the model is trained, it can be used to measure energy Usage for new data. For example, you could use the model to predict the price of a house energy consumption that you are interested in buying.

1. Prepare the data:-

- This involves cleaning the data, removing any errors or inconsistencies, and transforming the data into a format that is compatible with the machine learning algorithm that you will be using.

2. Split the data into training and test sets:-

- The training set will be used to train the model, and the test set will be used to evaluate the performance of the model on unseen data.

3. Choose a machine learning algorithm:-

- There are a number of different machine learning algorithms that can be used for measuring energy consumption, such **as linear regression, ridge regression, lasso regression, decision trees, and random forests**.

4. Tune the hyper parameters of the algorithm:-

- The hyper parameters of a machine learning algorithm are parameters that control the learning process.
- It is important to tune the hyper parameters of the algorithm to optimize its performance.
- Train the model on the training set. This involves feeding the training data to the model and allowing it to learn the relationships between the features and house prices.

5. Evaluate the model on the test set:-

- This involves feeding the test data to the model and measuring how well it predicts the house prices of energy consumption.

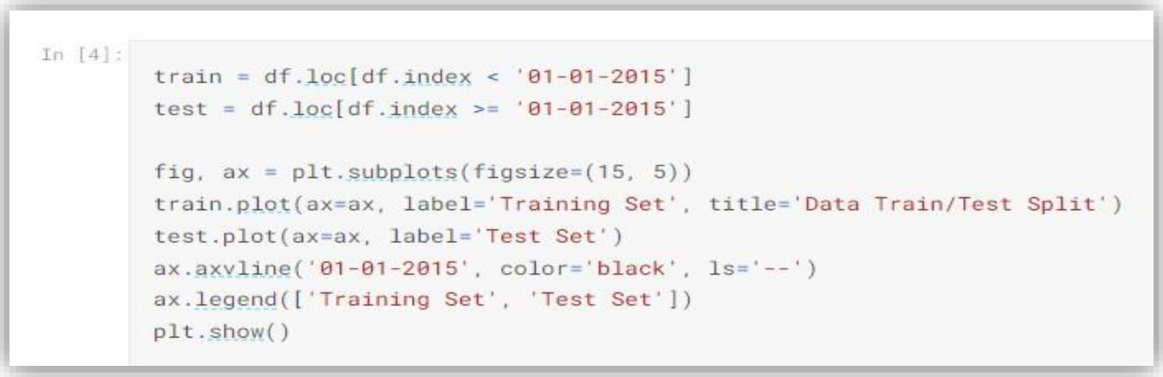
Dividing Dataset Features and Target Variables: -

In [5]:

```
#MW by Month
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()

#MW by Year
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()
```

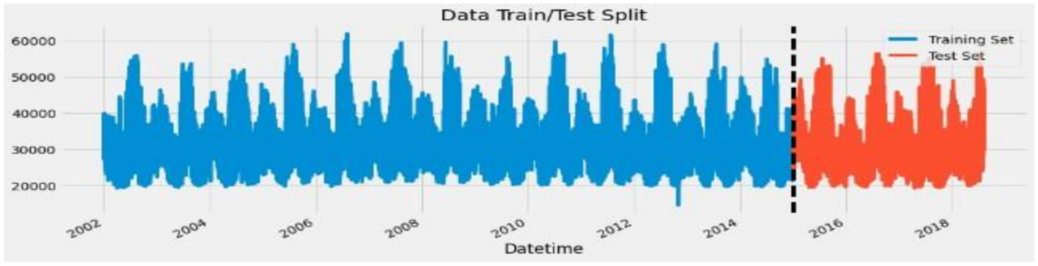
Split the data into training and test sets. The training set will be us ed to train the model, and test set will be used to evaluate the perfor mance of the model.

In [6]:



```
In [4]:
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='Training Set', title='Data Train/Test Split')
test.plot(ax=ax, label='Test Set')
ax.axvline('01-01-2015', color='black', ls='--')
ax.legend(['Training Set', 'Test Set'])
plt.show()
```

Out [6]:



```
df.loc[(df.index > '01-01-2010') & (df.index < '01-08-2010')] \
    .plot(figsize=(15, 5), title='Week Of Data')
plt.show()
```

Train the model on the training set:-

This involves feeding the training data to the model and allowing it to learn the relationships between the features and the target variable.

Evaluate the model on the test set:-

This involves feeding the test data to the model and measuring how w ell it predicts the target variable.

Model Evaluation:-

Choose an appropriate AI model for time series forecasting, such as rec urrent neural networks (RNNs), Long Short-Term Memory (LSTM) netw orks, or traditional statistical models like ARIMA.

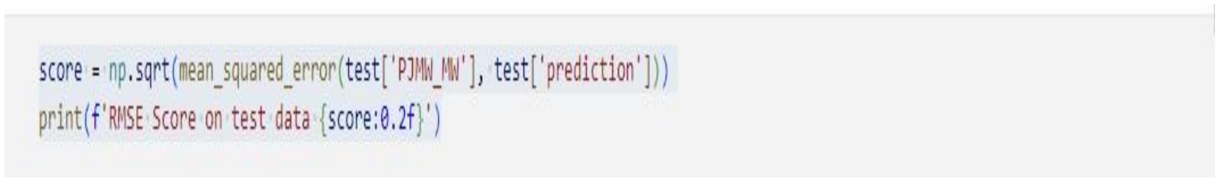
1. **Calculate the evaluation metrics:-**

   There are a number of different evaluation metrics that can be used to assess the performance of a machine learning model, such as R-squared, mean squared error(MSE), and root mean squa red error (RMSE).

2. **Interpret the evaluation metrics:-**

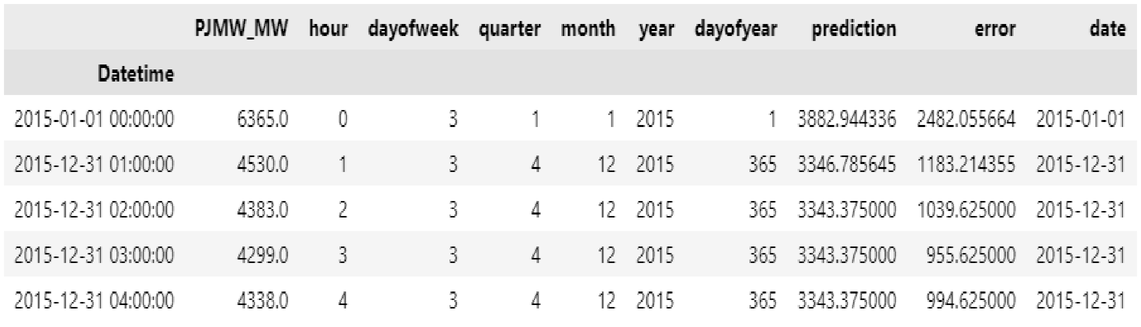
   The evaluation metrics will give you an idea of how well the mode l is performing on unseen data. If the model is performing well, t hen you can be confident that it will generalize well to new data. However, if the model is performing poorly, then you may need to try a different model or retune the hyper parameters of the curren t model.

Training a model for hourly energy consumption prediction typically in volves using time-series data and machine learning techniques.



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

```
test['error'] = np.abs(test[TARGET] - test['prediction'])

test['date'] = test.index.date
```



	PJMW_MW	hour	dayofweek	quarter	month	year	dayofyear	prediction	error	date
Datetime										
2015-01-01 00:00:00	6365.0	0	3	1	1	2015	1	3882.944336	2482.055664	2015-01-01
2015-12-31 01:00:00	4530.0	1	3	4	12	2015	365	3346.785645	1183.214355	2015-12-31
2015-12-31 02:00:00	4383.0	2	3	4	12	2015	365	3343.375000	1039.625000	2015-12-31
2015-12-31 03:00:00	4299.0	3	3	4	12	2015	365	3343.375000	955.625000	2015-12-31
2015-12-31 04:00:00	4338.0	4	3	4	12	2015	365	3343.375000	994.625000	2015-12-31

<u>Model evaluation:</u>

- ❖ 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.

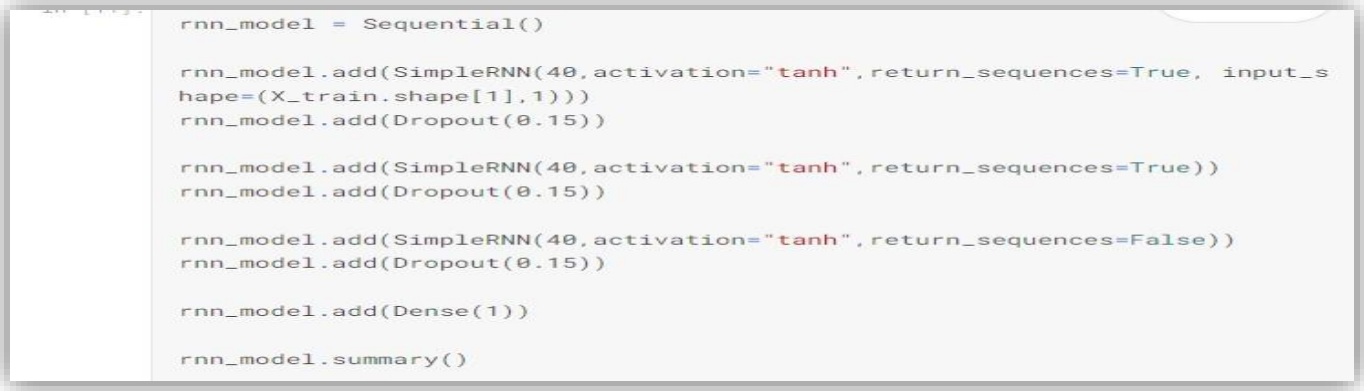
Mean Squared Error (MSE): The average squared difference between t he predicted and actual values.

Root Mean Squared Error (RMSE): The square root of the MSE, which is in the same unit as the target variable.

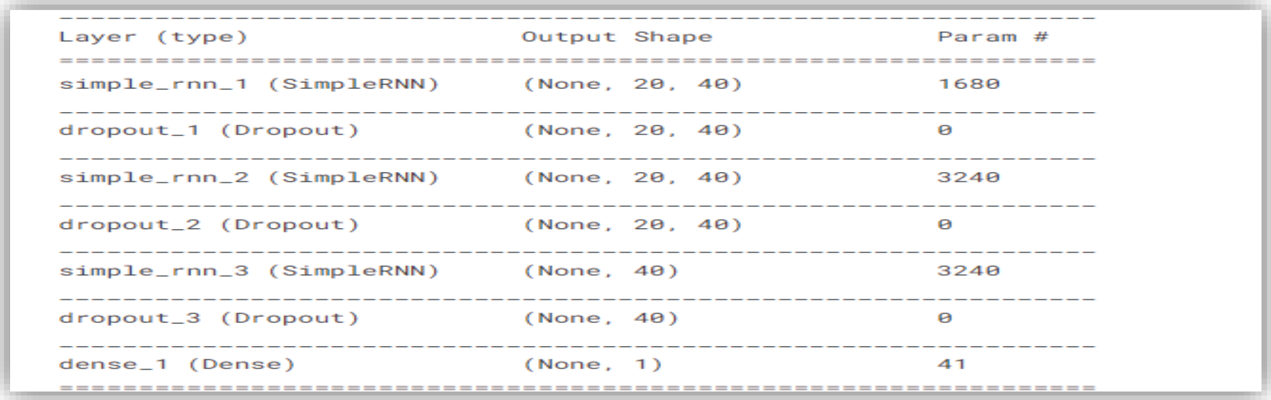
Mean Absolute Percentage Error (MAPE): Measures the percentage difference between predicted and actual values.

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:



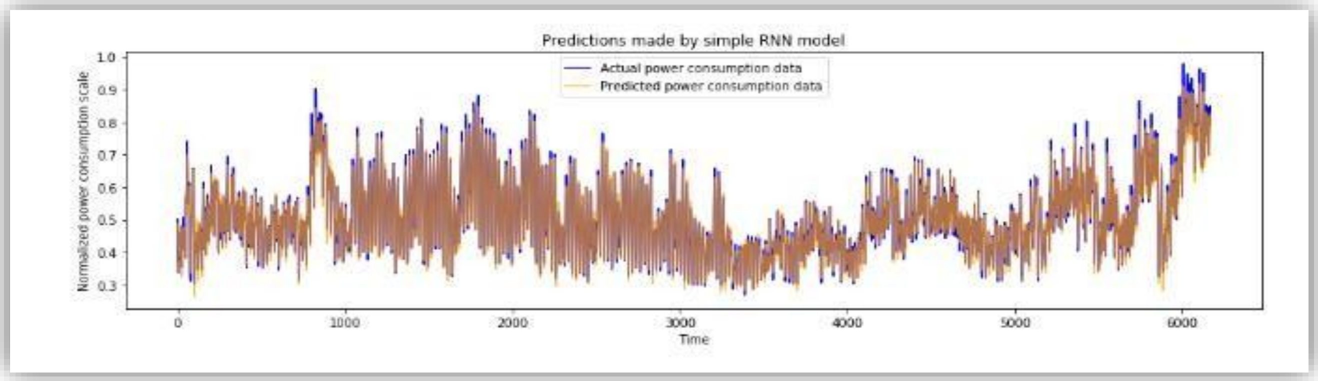
```
rnn_model = Sequential()

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()
```



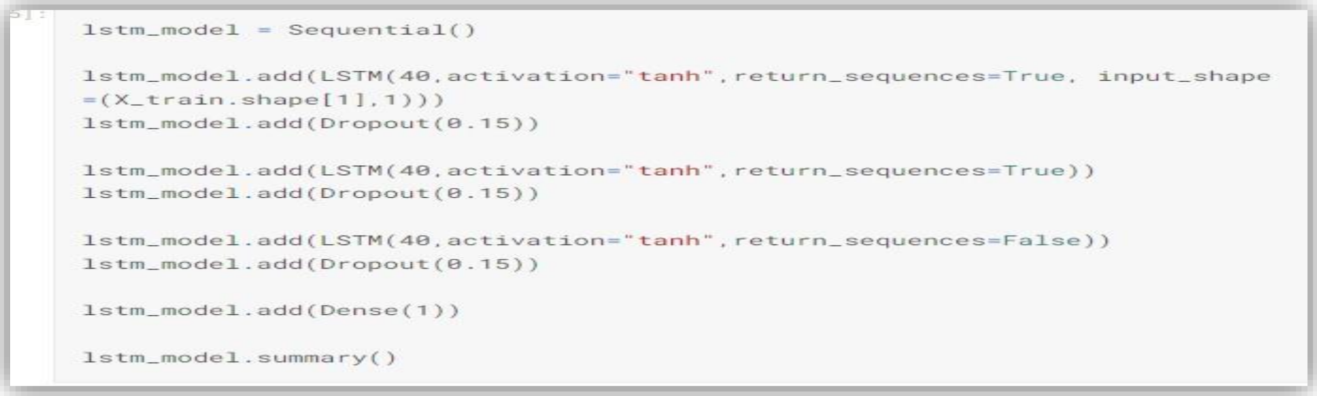
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
simple_rnn_1 (SimpleRNN)     (None, 20, 40)            1680
_________________________________________________________________
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
```

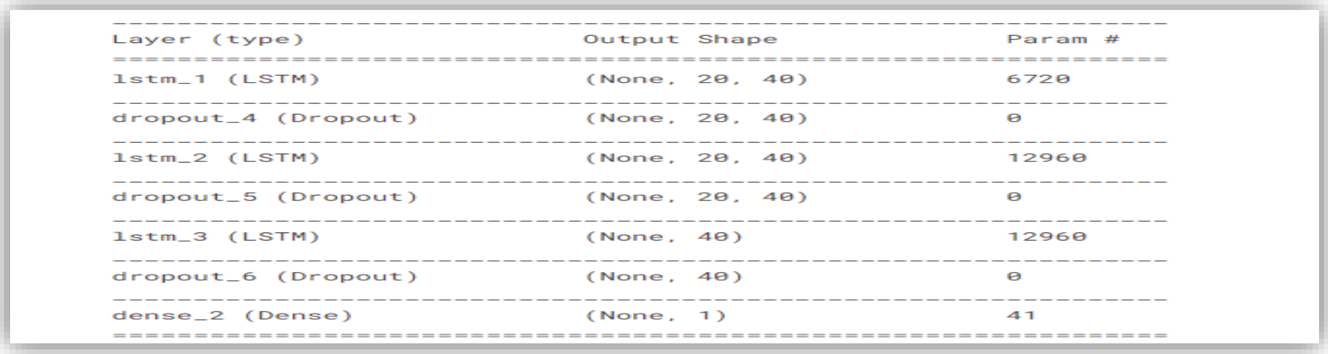
Predictions made by RNN model



Simple LSTM Model:

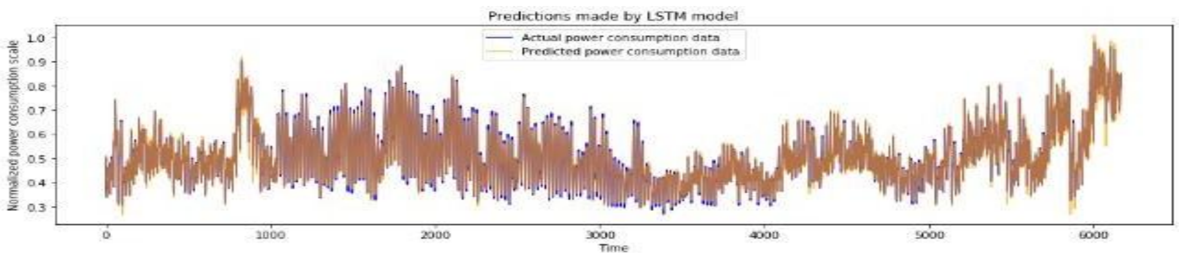


```
lstm_model = Sequential()

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()
```

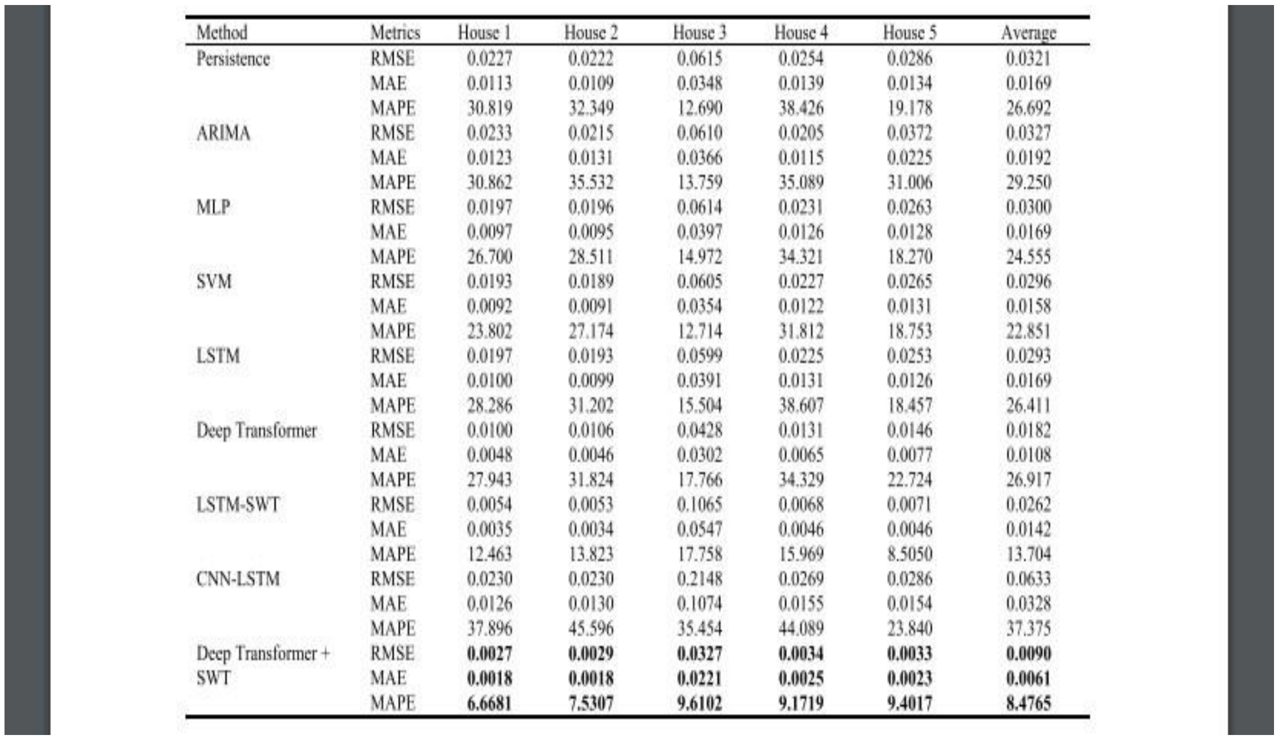


```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_1 (LSTM)                (None, 20, 40)            6720
_________________________________________________________________
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
```



Model Comparison:



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**

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

Feature Engineering:-

Time-based Features:

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.

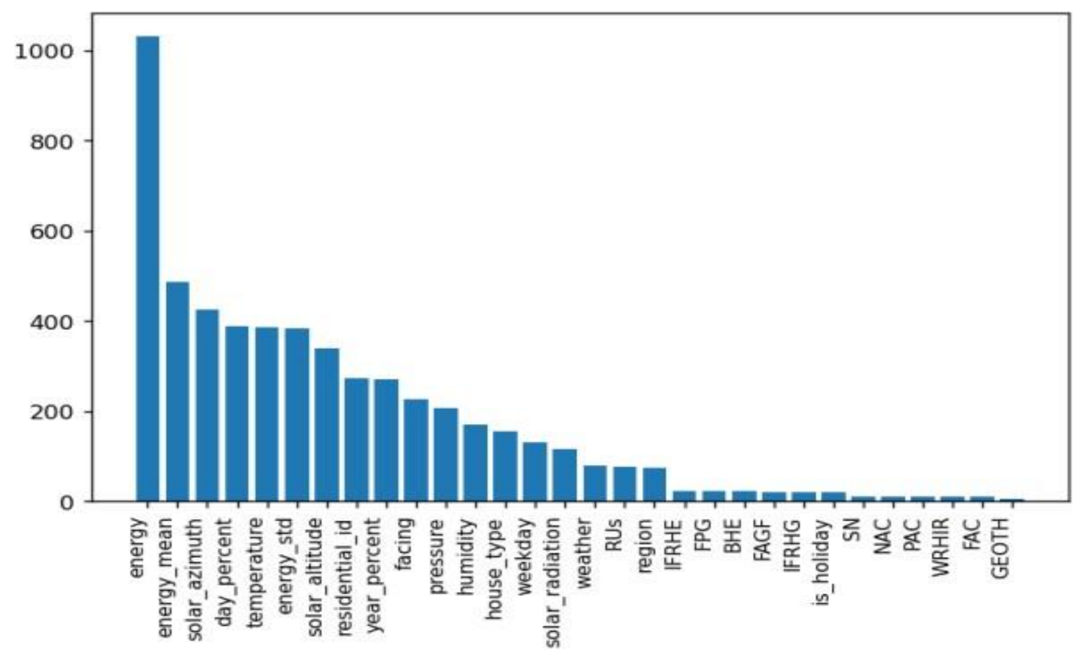
Time Series Decomposition Components:

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.

<u>Use a variety of feature engineering techniques:</u>



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.

Conclusion:

- 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.
- The choice of the best model and features depends on the specific characteristics of your dataset and the goals of your forecasting task. Continuous improvement and monitoring of the model's performance are essential for making accurate and reliable energy consumption predictions.