MEASURE ENERGY CONSUMPTION

TEAM MEMBER

510521104024: LOKESH L

PHASE 4: DOCUMENT SUBMISSION

PROJECT TITLE: Measure Energy Consumption

PHASE 4:Development part-2

TOPIC: Continue building the project by performing different activities like feature engineering, model training, evaluation etc..

INTRODUCTION:

- Measuring energy consumption is important for various purposes, whether it's to monitor and reduce energy usage at home or to optimize energy efficiency in a commercial or industrial setting. Here are several methods and tools for measuring energy consumption

- When measuring energy consumption, it's essential to consider factors like the granularity of data needed, the frequency of measurements, and your specific goals, whether it's reducing energy costs, improving efficiency, or meeting sustainability targets.

Feature Engineering:

- Feature engineering in the context of artificial intelligence (AI) and machine learning is the process of selecting, modifying, or creating relevant features from raw data to improve the performance and predictive power of AI models. Feature engineering is a critical step, as the quality of the features has a significant impact on the model's ability to generalize from data and make accurate predictions.

- Feature engineering is often an iterative process, involving experimentation and evaluation of different feature sets. The goal is to create features that best represent the underlying patterns in the data, improve model generalization, and enhance the model's predictive power.

Model Training:

- Model training is a crucial step in the development of machine learning and artificial intelligence models. During this phase, the model learns from the provided data and tries to generalize patterns and relationships within the data, enabling it to make predictions on new, unseen data

- Model training is an iterative and resource-intensive process that often requires experimentation and fine-tuning to achieve the best results. The quality of your training data, choice of model, and careful monitoring of performance are key factors in the success of your AI project.

Evaluation:

- Evaluating the performance of artificial intelligence (AI) models is a critical step in the development process. Effective evaluation helps you understand how well your model is performing, identify areas for improvement, and make informed decisions. The choice of evaluation metrics depends on the type of problem (classification, regression, etc.) and the specific goals of your AI project.

- The choice of evaluation metric should align with the specific objectives of your AI project. Additionally, you should consider using cross-validation techniques to assess your model's generalization performance and to avoid overfitting.

FEATURE ENGINEERING:

The Dataset:

Name	Date	Type	Size	Tags
AEP_hourly.csv	04-10-2019 19:25	OpenOffice.org 1....	3,316 KB	
COMED_hourly.csv	04-10-2019 19:25	OpenOffice.org 1....	1,800 KB	
DAYTON_hourly.csv	04-10-2019 19:25	OpenOffice.org 1....	3,198 KB	
DEOK_hourly.csv	04-10-2019 19:25	OpenOffice.org 1....	1,523 KB	
DOM_hourly.csv	04-10-2019 19:25	OpenOffice.org 1....	3,132 KB	
DUQ_hourly.csv	04-10-2019 19:25	OpenOffice.org 1....	3,140 KB	
EKPC_hourly.csv	04-10-2019 19:25	OpenOffice.org 1....	1,193 KB	
est_hourly.paruqet	04-10-2019 19:25	PARUQET File	3,595 KB	
FE_hourly.csv	04-10-2019 19:25	OpenOffice.org 1....	1,662 KB	
NI_hourly.csv	04-10-2019 19:25	OpenOffice.org 1....	1,584 KB	
pjm_hourly_est.csv	04-10-2019 19:25	OpenOffice.org 1....	12,409 KB	
PJM_Load_hourly.csv	04-10-2019 19:25	OpenOffice.org 1....	900 KB	
PJME_hourly.csv	04-10-2019 19:25	OpenOffice.org 1....	3,975 KB	
PJMW_hourly.csv	04-10-2019 19:25	OpenOffice.org 1....	3,776 KB	

Libraries:

```python
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt

from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import r2_score

import tensorflow as tf
from keras.layers import Dense,Dropout,SimpleRNN,LSTM
from keras.models import Sequential
```

FEATURE ENGINEERING:

Reading the CSV file:

CODE:

```
import pandas as pd
df =  pd.read_csv('../input/energy-data/AEP_hourly.csv')
df.head()
```

	Datetime	AEP_MW
0	2004-12-31 01:00:00	13478.0
1	2004-12-31 02:00:00	12865.0
2	2004-12-31 03:00:00	12577.0
3	2004-12-31 04:00:00	12517.0
4	2004-12-31 05:00:00	12670.0

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 116189 entries, 0 to 116188
Data columns (total 2 columns):
 #   Column    Non-Null Count   Dtype  
---  ------    --------------   -----  
 0   Datetime  116189 non-null  object 
 1   DOM_MW    116189 non-null  float64
dtypes: float64(1), object(1)
memory usage: 1.8+ MB
```

+ Code   + Markdown

Conversion of Date time column to Date time format:

```
# We must convert the Datetime column to Datetime format
df['Datetime'] = pd.to_datetime(df['Datetime'])

# We index the Datetime column after transformation
df.set_index('Datetime', inplace=True)
df.head()
```

	DOM_MW
Datetime	
2005-12-31 01:00:00	9389.0
2005-12-31 02:00:00	9070.0
2005-12-31 03:00:00	9001.0
2005-12-31 04:00:00	9042.0
2005-12-31 05:00:00	9132.0

The years in the dataset:

```
years = df.index.year.unique()
years
```

```
[11]: Index([2005, 2006, 2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015, 2016,
             2017, 2018],
            dtype='int32', name='Datetime')
```

\+ Code   + Markdown

Average energy consumed per year:

```
df_yearly_avg = df['DOM_MW'].resample('Y').mean()
df_yearly_avg.to_frame()
```

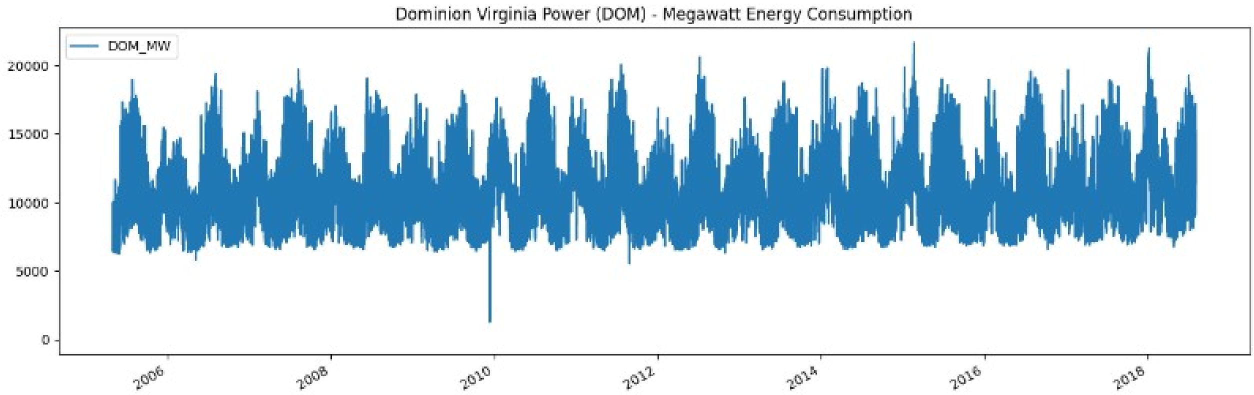
	DOM_MW
Datetime	
2005-12-31	10833.524668
2006-12-31	10457.146951
2007-12-31	10991.015871
2008-12-31	10786.751765
2009-12-31	10696.930235
2010-12-31	11280.065548
2011-12-31	10865.571021
2012-12-31	10614.735368
2013-12-31	10904.946677
2014-12-31	11074.416324
2015-12-31	11150.607420
2016-12-31	11142.317737
2017-12-31	11057.906279
2018-12-31	11710.409463

let's observe our data set on the graph:

```
df.plot(figsize=(16,5),legend=True)
plt.axhspan(0, 1, facecolor='gray', alpha=0.3)

plt.title('Dominion Virginia Power (DOM) - Megawatt Energy Consumption')

plt.show()
```

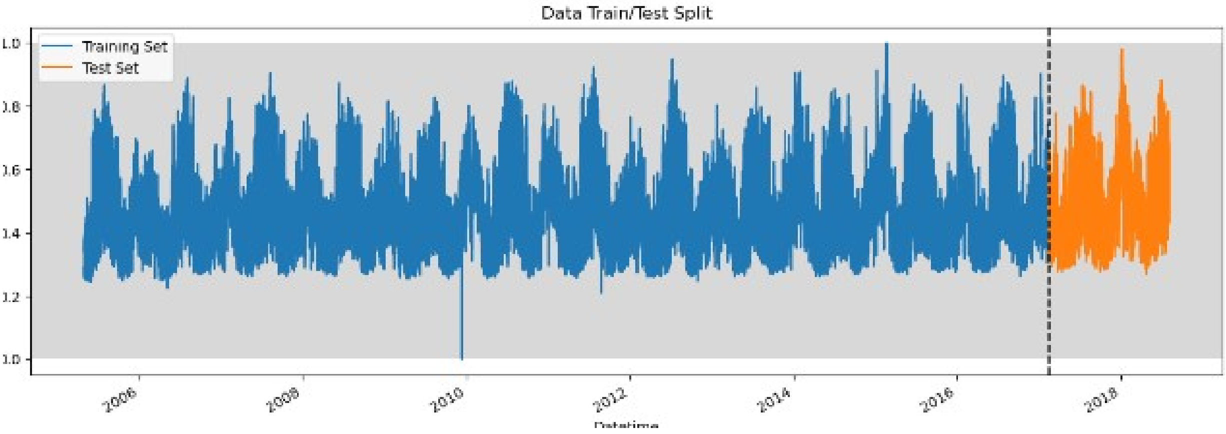


Let's observe train and test data on the graph:

```
# 2017-02-13 after this date we will choose the test set
split_date = '2017-02-13'

DOM_train = df_norm.loc[df_norm.index <= split_date].copy()
DOM_test = df_norm.loc[df_norm.index > split_date].copy()
```

```
fig, ax = plt.subplots(figsize=(15, 5))
DOM_train.plot(ax=ax, label='Training Set', title='Data Train/Test Split')
DOM_test.plot(ax=ax, label='Test Set')
ax.axvline('2017-02-13', color='black', ls='--')
ax.legend(['Training Set', 'Test Set'])
plt.axhspan(0, 1, facecolor='gray', alpha=0.3)
plt.show()
```



(MODEL TRAINING)Prepare Data for Training the RNN:

With the following function block, let's set our data set as training and test data set in a model appropriate way

```
def load_data(data, seq_len):
    X_train = []
    y_train = []

    for i in range(seq_len, len(data)):
        X_train.append(data.iloc[i-seq_len : i, 0])
        y_train.append(data.iloc[i, 0])

    # last 6189 days are going to be used in test
    X_test = X_train[110000:]
    y_test = y_train[110000:]

    # first 110000 days are going to be used in training
    X_train = X_train[:110000]
    y_train = y_train[:110000]

    # convert to numpy array
    X_train = np.array(X_train)
    y_train = np.array(y_train)
```

```
    X_test = np.array(X_test)
    y_test = np.array(y_test)

    # reshape data to input into RNN&LSTM models
    X_train = np.reshape(X_train, (110000, seq_len, 1))

    X_test = np.reshape(X_test, (X_test.shape[0], seq_len, 1))

    return [X_train, y_train, X_test, y_test]
```

- The seq_len parameter determines how far back the model will look at historical data, helping the model to capture time dependencies in a memory-aware way.
- We should note that if "seq_len" is too large, the model can become complex and prone to overlearning.
- We can specify separate seq_len values for RNN and LSTM

```
seq_len = 20

# Let's create train, test data
X_train, y_train, X_test, y_test = load_data(df, seq_len)

print('X_train.shape = ',X_train.shape)
print('y_train.shape = ', y_train.shape)
print('X_test.shape = ', X_test.shape)
print('y_test.shape = ',y_test.shape)
```

```
X_train.shape =  (110000, 20, 1)
y_train.shape =  (110000,)
X_test.shape =  (6169, 20, 1)
y_test.shape =  (6169,)
```

Build a RNN model:

```
rnn_model = Sequential()

rnn_model.add(SimpleRNN(40,activation="tanh",return_sequences=True, input_shape=(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()
```

```
Model: "sequential_10"
_________________________________________________________________
=================================================================
simple_rnn_15 (SimpleRNN)    (None, 20, 40)            1680

dropout_30 (Dropout)         (None, 20, 40)            0

simple_rnn_16 (SimpleRNN)    (None, 20, 40)            3240

dropout_31 (Dropout)         (None, 20, 40)            0

simple_rnn_17 (SimpleRNN)    (None, 40)                3240

dropout_32 (Dropout)         (None, 40)                0

dense_10 (Dense)             (None, 1)                 41

=================================================================
Total params: 8,201
Trainable params: 8,201
Non-trainable params: 0
_________________________________________________________________
```

```
rnn_model.compile(optimizer="adam",loss="MSE")
rnn_model.fit(X_train, y_train, epochs=10, batch_size=1000)
```

```
Epoch 1/10
110/110 [==============================] - 14s 98ms/step - loss: 0.0969
Epoch 2/10
110/110 [==============================] - 12s 113ms/step - loss: 0.0180
Epoch 3/10
110/110 [==============================] - 12s 111ms/step - loss: 0.0100
Epoch 4/10
110/110 [==============================] - 10s 94ms/step - loss: 0.0070
Epoch 5/10
110/110 [==============================] - 10s 92ms/step - loss: 0.0054
Epoch 6/10
110/110 [==============================] - 10s 94ms/step - loss: 0.0044
Epoch 7/10
110/110 [==============================] - 10s 92ms/step - loss: 0.0038
Epoch 8/10
110/110 [==============================] - 10s 93ms/step - loss: 0.0032
Epoch 9/10
110/110 [==============================] - 10s 90ms/step - loss: 0.0029
```

```
rnn_predictions = rnn_model.predict(X_test)

rnn_score = r2_score(y_test,rnn_predictions)
print("R2 Score of RNN model = ",rnn_score)
```

```
193/193 [==============================] - 2s 8ms/step
R2 Score of RNN model =  0.9504153338386626
```

```
# Reverse transform scaler to convert to real values
y_test_inverse = scaler.inverse_transform(y_test.reshape(-1, 1))
rnn_predictions_inverse = scaler.inverse_transform(rnn_predictions)

# Get values after inverse transformation
y_test_inverse = y_test_inverse.flatten()
rnn_predictions_inverse = rnn_predictions_inverse.flatten()
```

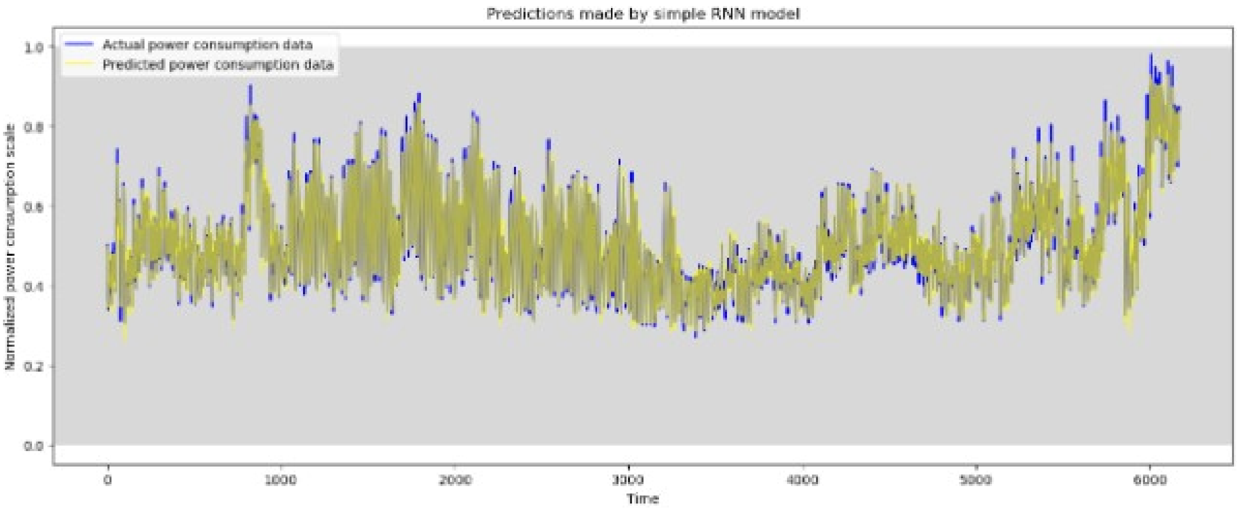
```
last_6169_index_dates = df.index[-6169:]

# Now let's see our actual y and predicted y values as dataframes
results_RNN = pd.DataFrame({"Date":last_6169_index_dates, 'Actual': y_test_
inverse, 'Predicted': rnn_predictions_inverse})
results_RNN
```

	Date	Actual	Predicted
0	2017-02-13 23:00:00	11494.0	11527.064453
1	2017-02-14 00:00:00	10975.0	10914.053711
2	2017-02-12 01:00:00	8728.0	10440.536133
3	2017-02-12 02:00:00	8390.0	8713.056641
4	2017-02-12 03:00:00	8283.0	7953.261719
...	...	...	...
6164	2018-01-01 20:00:00	18418.0	18187.222656
6165	2018-01-01 21:00:00	18567.0	17826.732422
6166	2018-01-01 22:00:00	18307.0	17805.630859
6167	2018-01-01 23:00:00	17814.0	17588.916016
6168	2018-01-02 00:00:00	17428.0	17013.826172

6169 rows × 3 columns

```python
plt.figure(figsize=(16,6))
plt.plot(y_test, color='blue',label='Actual power consumption data')
plt.plot(rnn_predictions, alpha=0.7, color='yellow', label='Predicted power consumption data')
plt.axhspan(0, 1, facecolor='gray', alpha=0.3)
plt.title("Predictions made by simple RNN model")
plt.xlabel('Time')
plt.ylabel('Normalized power consumption scale')
plt.legend()
plt.show()
```



Evaluation:

‣ Evaluation of energy consumption measures is the process of assessing the effectiveness of energy conservation measures (ECMs) in reducing energy use and costs. This process can be used to identify the most effective ECMs for a particular facility or organization, and to track the progress of ECM implementation over time.

‣ There are a number of different methods that can be used to evaluate energy consumption measures. Some of the most common methods include:

•**Energy benchmarking:** This involves comparing the energy performance of a facility or organization to similar facilities or organizations. This can help to identify areas where energy use is excessive and where ECMs could be implemented to reduce energy consumption.

•**Life cycle assessment (LCA):**This involves evaluating the environmental and economic impacts of an ECM over its entire lifespan. This can help to identify ECMs that are both effective at reducing energy consumption and cost-effective.

•**Energy simulation:**This involves using computer models to predict the energy savings that would be achieved by implementing an ECM. This can be a useful tool for evaluating ECMs in new construction projects or for projects where it is not possible to collect historical energy data.

- Once the effectiveness of an ECM has been evaluated, the results can be used to make decisions about whether or not to implement the ECM, and to prioritize ECMs for implementation.

- **Here are some specific examples of how evaluation can be used to measure energy consumption:**

- A building owner could compare the energy consumption of a building before and after implementing a new energy efficiency measure, such as installing LED lighting or upgrading the HVAC system.

- A utility company could use energy benchmarking to compare the energy performance of different customer groups, such as commercial businesses or residential households.

- A government agency could use LCA to evaluate the environmental and economic impacts of different energy efficiency measures.

- A manufacturing company could use energy simulation to predict the energy savings that would be achieved by implementing a new energy efficiency measure in a new production line.

Evaluation of energy consumption measures is an important tool for reducing energy use and costs. By carefully evaluating the effectiveness of ECMs, organizations can make informed decisions about how to best invest their resources to achieve their energy conservation goals.