jupyter Multi-Step LSTM Time Series Forecasting Models for Power Usage Last Checkpoint: 18 hours ago

File Edit View Run Kernel Settings Help

Trusted

Code

JupyterLab Python 3 (ipykernel)

[2]:
```
import pandas as pd
import numpy as np
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout
import matplotlib.pyplot as plt
from sklearn.metrics import mean_squared_error, mean_absolute_error
```

[3]:
```
# Load and preprocess the data
data = pd.read_csv('Dataset.csv')
data
```

[3]:

	Unnamed: 0	Freq	Ia	Ib	Ic	PF	UpdateTime	Va	Vb	Vc	...	kWh+	kWh-	kWh_abs	kWh_net	θIA	θIB	θIC	θVA	θVB	θVC
0	0	59.9	2.2	2.0	2.8	0.66	2022-09-02 00:00:28	6832.0	6717.0	6837.0	...	311268.0	0.0	311270.0	311266.0	326.88	64.26	191.73	0.0	120.57	240.55
1	1	59.9	2.4	2.0	2.8	0.66	2022-09-02 00:00:48	6831.0	6712.0	6837.0	...	311268.0	0.0	311270.0	311266.0	327.52	66.49	191.57	0.0	120.65	240.38
2	2	60.0	2.4	2.0	2.8	0.64	2022-09-02 00:01:33	6826.0	6693.0	6827.0	...	311268.0	0.0	311270.0	311268.0	325.71	62.76	187.86	0.0	120.48	240.30
3	3	60.0	2.4	2.0	2.8	0.64	2022-09-02 00:01:54	6825.0	6708.0	6832.0	...	311268.0	0.0	311270.0	311268.0	325.71	64.26	188.95	0.0	120.40	240.30
4	4	59.9	2.4	2.0	2.8	0.65	2022-09-02 00:02:39	6814.0	6679.0	6819.0	...	311270.0	0.0	311270.0	311268.0	325.71	65.03	188.03	0.0	120.65	240.46
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2564	2564	60.0	2.4	2.0	2.6	0.67	2022-09-02	6803.0	6689.0	6807.0	...	312282.0	0.0	312284.0	312282.0	325.71	67.97	191.39	0.0	120.57	240.38

2568	2568	59.9	2.4	2.0	2.6	0.68	23:59:37	6808.0	6689.0	6813.0	...	312284.0	0.0	312286.0	312282.0	327.52	68.36	192.86	0.0	120.48	240.46

2569 rows × 29 columns

[6]:
```
missing_values = data.isnull().sum()
print(missing_values)
```

```
Unnamed: 0    0
Freq          0
Ia            0
Ib            0
Ic            0
PF            0
Va            0
Vb            0
Vc            0
kPt           0
kQt           0
kSt           0
kVARh_Q1      0
kVARh_Q2      0
kVARh_Q3      0
kVARh_Q4      0
kVAh+         0
kVAh-         0
kWh+          0
kWh-          0
kWh_abs       0
kWh_net       0
θIA           0
θIB           0
θIC           0
θVA           0
```

jupyter Multi-Step LSTM Time Series Forecasting Models for Power Usage Last Checkpoint: 18 hours ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)

[ ]:
```
# Convert 'UpdateTime' to datetime and set as index
data['UpdateTime'] = pd.to_datetime(data['UpdateTime'])
data.set_index('UpdateTime', inplace=True)
df = data[['kWh+']]
```

[23]:
```
data
```

[23]:

UpdateTime	Unnamed: 0	Freq	Ia	Ib	Ic	PF	Va	Vb	Vc	kPt	...	kWh+	kWh-	kWh_abs	kWh_net	θIA	θIB	θIC	θVA	θVB	θVC
2022-09-02 00:00:28	0	59.9	2.2	2.0	2.8	0.66	6832.0	6717.0	6837.0	32.0	...	311268.0	0.0	311270.0	311266.0	326.88	64.26	191.73	0.0	120.57	240.55
2022-09-02 00:00:48	1	59.9	2.4	2.0	2.8	0.66	6831.0	6712.0	6837.0	32.0	...	311268.0	0.0	311270.0	311266.0	327.52	66.49	191.57	0.0	120.65	240.38
2022-09-02 00:01:33	2	60.0	2.4	2.0	2.8	0.64	6826.0	6693.0	6827.0	32.0	...	311268.0	0.0	311270.0	311268.0	325.71	62.76	187.86	0.0	120.48	240.30
2022-09-02 00:01:54	3	60.0	2.4	2.0	2.8	0.64	6825.0	6708.0	6832.0	32.0	...	311268.0	0.0	311270.0	311268.0	325.71	64.26	188.95	0.0	120.40	240.30
2022-09-02 00:02:39	4	59.9	2.4	2.0	2.8	0.65	6814.0	6679.0	6819.0	32.0	...	311270.0	0.0	311270.0	311268.0	325.71	65.03	188.03	0.0	120.65	240.46
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2022-09-02 23:57:26	2564	60.0	2.4	2.0	2.6	0.67	6803.0	6689.0	6807.0	32.0	...	312282.0	0.0	312284.0	312282.0	325.71	67.97	191.39	0.0	120.57	240.38
2022-09-02 23:58:11	2565	59.9	2.4	2.0	2.8	0.67	6803.0	6688.0	6808.0	32.0	...	312284.0	0.0	312284.0	312282.0	326.88	67.89	190.35	0.0	120.48	240.55

jupyter Multi-Step LSTM Time Series Forecasting Models for Power Usage Last Checkpoint: 18 hours ago

File Edit View Run Kernel Settings Help

Trusted

Code

JupyterLab Python 3 (ipykernel)

[8]:
```
# Normalize the data
scaler = MinMaxScaler()
scaled_data = scaler.fit_transform(df)
```

[9]:
```
# Create sequences for LSTM
def create_sequences(data, sequence_length, forecast_horizon):
    X, y = [], []
    for i in range(len(data) - sequence_length - forecast_horizon):
        X.append(data[i:i + sequence_length])
        y.append(data[i + sequence_length:i + sequence_length + forecast_horizon])
    return np.array(X), np.array(y)

sequence_length = 60
forecast_horizon = 10
X, y = create_sequences(scaled_data, sequence_length, forecast_horizon)
```

[10]:
```
# Split into training and validation sets
split = int(X.shape[0] * 0.8)
X_train, X_val = X[:split], X[split:]
y_train, y_val = y[:split], y[split:]
```

[11]:
```
# Build the LSTM model
model = Sequential([
    LSTM(128, activation='relu', input_shape=(sequence_length, 1), return_sequences=True),
    Dropout(0.2),
    LSTM(64, activation='relu'),
    Dropout(0.2),
    Dense(forecast_horizon)
])

model.compile(optimizer='adam', loss='mse')
```

jupyter Multi-Step LSTM Time Series Forecasting Models for Power Usage Last Checkpoint: 18 hours ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)

[11]:
```
# Build the LSTM model
model = Sequential([
    LSTM(128, activation='relu', input_shape=(sequence_length, 1), return_sequences=True),
    Dropout(0.2),
    LSTM(64, activation='relu'),
    Dropout(0.2),
    Dense(forecast_horizon)
])

model.compile(optimizer='adam', loss='mse')
```

```
C:\Users\Usama Zafar\Desktop\AI_Course\ml_project\LSTM\env\Lib\site-packages\keras\src\layers\rnn\rnn.py:204: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(**kwargs)
```

[12]:
```
# Train the model
history = model.fit(X_train, y_train, epochs=20, batch_size=32, validation_data=(X_val, y_val))
```

```
Epoch 1/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 15s 106ms/step - loss: 0.1014 - val_loss: 0.0121
Epoch 2/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 4s 59ms/step - loss: 0.0077 - val_loss: 7.9988e-04
Epoch 3/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 4s 64ms/step - loss: 0.0059 - val_loss: 3.8172e-04
Epoch 4/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 4s 65ms/step - loss: 0.0048 - val_loss: 3.9095e-04
Epoch 5/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 7s 91ms/step - loss: 0.0038 - val_loss: 0.0011
Epoch 6/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 4s 57ms/step - loss: 0.0032 - val_loss: 3.1055e-04
Epoch 7/20
63/63 ━━━━━━━━━━━━━━━━━━━━ 4s 60ms/step - loss: 0.0029 - val_loss: 3.0378e-04
Epoch 8/20
```

jupyter Multi-Step LSTM Time Series Forecasting Models for Power Usage Last Checkpoint: 18 hours ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)

[16]:
```
# Make predictions on the validation set
predictions = model.predict(X_val)
predictions
```

```
16/16 ━━━━━━━━━━━━━━━━━━━━ 1s 56ms/step
```

[16]:
```
array([[0.84137416, 0.8436718 , 0.8438607 , ..., 0.8496277 , 0.85105515,
        0.8453314 ],
       [0.8418927 , 0.8441903 , 0.8443835 , ..., 0.8501444 , 0.8515784 ,
        0.8458504 ],
       [0.8424321 , 0.84473044, 0.8449311 , ..., 0.8506854 , 0.85212654,
        0.84639233],
       ...,
       [1.0018115 , 1.0030583 , 1.0051404 , ..., 1.0086414 , 1.013875  ,
        1.0079546 ],
       [1.0020906 , 1.0033355 , 1.0054209 , ..., 1.0089166 , 1.0141585 ,
        1.0082381 ],
       [1.0023564 , 1.0035985 , 1.005685  , ..., 1.0091759 , 1.0144262 ,
        1.0085077 ]], dtype=float32)
```

[26]:
```
# Assuming predictions and y_val are 3D arrays
predictions_reshaped = predictions.reshape(-1, predictions.shape[-1])  # Reshape to (n_samples * sequence_length, n_features)
y_val_reshaped = y_val.reshape(-1, y_val.shape[-1])  # Similar reshaping

# Now apply inverse_transform
predictions_rescaled = scaler.inverse_transform(predictions_reshaped)
y_val_rescaled = scaler.inverse_transform(y_val_reshaped)
```

[27]:
```
plt.plot(y_val_rescaled.flatten(), label='Actual')
plt.plot(predictions_rescaled.flatten(), label='Predicted')
plt.title('Actual vs Predicted Power Usage')
plt.xlabel('Time Step')
plt.ylabel('kWh+')
plt.legend()
```

jupyter Multi-Step LSTM Time Series Forecasting Models for Power Usage Last Checkpoint: 18 hours ago

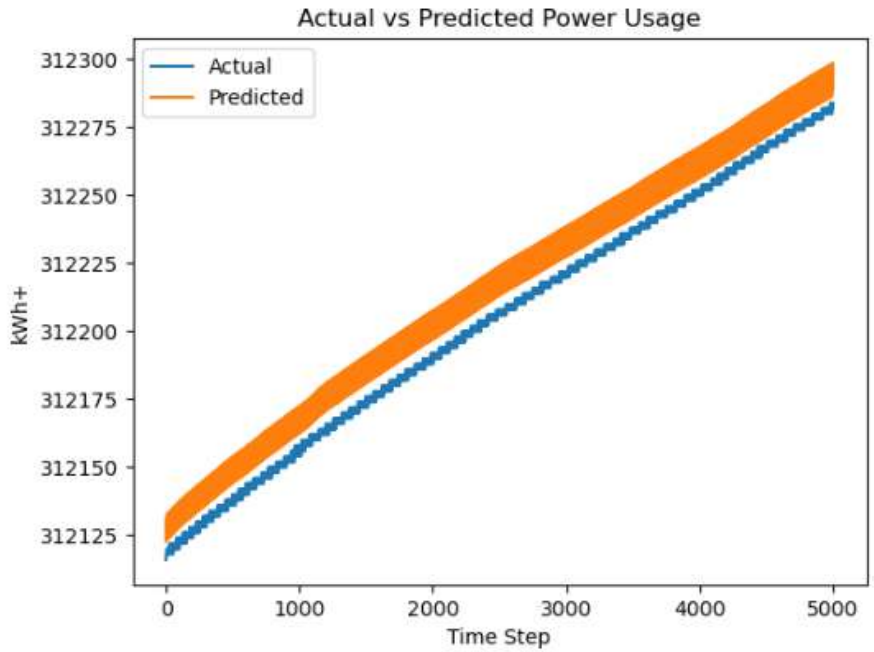
File Edit View Run Kernel Settings Help

Trusted

Code

JupyterLab Python 3 (ipykernel)

[27]:
```
plt.plot(y_val_rescaled.flatten(), label='Actual')
plt.plot(predictions_rescaled.flatten(), label='Predicted')
plt.title('Actual vs Predicted Power Usage')
plt.xlabel('Time Step')
plt.ylabel('kWh+')
plt.legend()
plt.show()
```



jupyter Multi-Step LSTM Time Series Forecasting Models for Power Usage Last Checkpoint: 18 hours ago

File Edit View Run Kernel Settings Help

Trusted

Code

JupyterLab Python 3 (ipykernel)

[29]:
```
# Calculate Mean Absolute Percentage Error (MAPE) for rescaled values
mape = np.mean(np.abs((y_val_rescaled - predictions_rescaled.flatten()) / y_val_rescaled)) * 100

# Calculate accuracy as a percentage
accuracy = 100 - mape

# Print the evaluation metrics
print(f"Mean Absolute Percentage Error (MAPE): {mape:.2f}%")
print(f"Accuracy: {accuracy:.2f}%")
```

```
Mean Absolute Percentage Error (MAPE): 0.02%
Accuracy: 99.98%
```

[31]:
```
# Calculate Mean Absolute Percentage Error (MAPE) for Scaled Values
mape = np.mean(np.abs((y_val - predictions.flatten()) / y_val)) * 100

# Calculate accuracy as a percentage
accuracy = 100 - mape

# Print the evaluation metrics
print(f"Mean Absolute Percentage Error (MAPE): {mape:.2f}%")
print(f"Accuracy: {accuracy:.2f}%")
```

```
Mean Absolute Percentage Error (MAPE): 5.84%
Accuracy: 94.16%
```

[ ]: