Import libraries

```
# Install necessary packages
!pip install pandas scikit-learn tensorflow
```

```
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (1.5.3)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-packages (1.2.2)
Requirement already satisfied: tensorflow in /usr/local/lib/python3.10/dist-packages (2.14.0)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10/dist-packages (from pandas) (2
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas) (2023.3.post
Requirement already satisfied: numpy>=1.21.0 in /usr/local/lib/python3.10/dist-packages (from pandas) (1.23.5)
Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.11.
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.3.
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn
Requirement already satisfied: absl-py>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.4.0
Requirement already satisfied: astunparse>=1.6.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.
Requirement already satisfied: flatbuffers>=23.5.26 in /usr/local/lib/python3.10/dist-packages (from tensorflow)
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in /usr/local/lib/python3.10/dist-packages (fr
Requirement already satisfied: google-pasta>=0.1.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (
Requirement already satisfied: h5py>=2.9.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (3.9.0)
Requirement already satisfied: libclang>=13.0.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (16.
Requirement already satisfied: ml-dtypes==0.2.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.2
Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (3.
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from tensorflow) (23.2)
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<5.0.0dev,>=3.20.3 i
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-packages (from tensorflow) (67.7.2)
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.16.0)
Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.3
Requirement already satisfied: typing-extensions>=3.6.6 in /usr/local/lib/python3.10/dist-packages (from tensorfl
Requirement already satisfied: wrapt<1.15,>=1.11.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in /usr/local/lib/python3.10/dist-packages (f
Requirement already satisfied: grpcio<2.0,>=1.24.3 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (
Requirement already satisfied: tensorboard<2.15,>=2.14 in /usr/local/lib/python3.10/dist-packages (from tensorflo
Requirement already satisfied: tensorflow-estimator<2.15,>=2.14.0 in /usr/local/lib/python3.10/dist-packages (fro
Requirement already satisfied: keras<2.15,>=2.14.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (
Requirement already satisfied: wheel<1.0,>=0.23.0 in /usr/local/lib/python3.10/dist-packages (from astunparse>=1.
Requirement already satisfied: google-auth<3,>=1.6.3 in /usr/local/lib/python3.10/dist-packages (from tensorboard
Requirement already satisfied: google-auth-oauthlib<1.1,>=0.5 in /usr/local/lib/python3.10/dist-packages (from te
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.15,
Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in /usr/local/lib/python3.10/dist-packages (
Requirement already satisfied: werkzeug>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.15,
Requirement already satisfied: cachetools<6.0,>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from google-aut
Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/python3.10/dist-packages (from google-auth
Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6
Requirement already satisfied: requests-oauthlib>=0.7.0 in /usr/local/lib/python3.10/dist-packages (from google-a
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0-
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2
Requirement already satisfied: MarkupSafe>=2.1.1 in /usr/local/lib/python3.10/dist-packages (from werkzeug>=1.0.1
Requirement already satisfied: pyasn1<0.6.0,>=0.4.6 in /usr/local/lib/python3.10/dist-packages (from pyasn1-modul
Requirement already satisfied: oauthlib>=3.0.0 in /usr/local/lib/python3.10/dist-packages (from requests-oauthlib
```

For training validation and testing

```
import csv
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.regularizers import l1
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error
import math

# Load the dataset
dataset = pd.read_csv('/content/dataset.csv')

# Extract features and target variable
X = dataset[['Wind Speed (m/s)', 'Theoretical_Power_Curve (KWh)', 'Wind Direction (°)']]
y = dataset['LV ActivePower (kW)']

# Normalize features
scaler = MinMaxScaler()
X_normalized = scaler.fit_transform(X)

# Split the data into train, validate, and test sets
X_train, X_temp, y_train, y_temp = train_test_split(X_normalized, y, test_size=0.4, random_state=42)
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.5, random_state=42)

# Build the LSTM model
model = Sequential()
model.add(LSTM(50, activation='relu', input_shape=(X_train.shape[1], 1), kernel_regularizer=l1(0.01)))
model.add(Dense(1))

# Compile the model
model.compile(optimizer=Adam(learning_rate=0.01), loss='mse', metrics=['mae'])

# Reshape data for LSTM (add time dimension)
X_train = X_train.reshape((X_train.shape[0], X_train.shape[1], 1))
X_val = X_val.reshape((X_val.shape[0], X_val.shape[1], 1))
X_test = X_test.reshape((X_test.shape[0], X_test.shape[1], 1))

# Train the model
model.fit(X_train, y_train, epochs=50, batch_size=32, validation_data=(X_val, y_val))

# Evaluate the model on the test set
loss, mae = model.evaluate(X_test, y_test)
print(f'Test Mean Absolute Error: {mae}')

# Assuming 'y_test_last_20_pred' is a 2D array
# Inverse transform the normalized predictions to get actual ActivePower values
scaler = MinMaxScaler()
y = y.values.reshape(-1, 1)
scaler.fit(y)

# Get last 1440 rows from test set
last_1440 = X_test.shape[0] - 1440

X_test_last_1440 = X_test[last_1440:]
y_test_last_1440 = y_test[last_1440:]

# Make predictions
y_pred_last_1440 = model.predict(X_test_last_1440)

# Inverse transform predictions
y_pred_last_1440 = scaler.inverse_transform(y_pred_last_1440).flatten()

# Calculate RMSE
rmse = math.sqrt(mean_squared_error(y_test_last_1440, y_pred_last_1440))

# Create DataFrame
result_df = pd.DataFrame({
    'Actual': y_test_last_1440,
    'Predicted': y_pred_last_1440,
    'RMSE': rmse
})

print(result_df)
```

```
# Create rows to write
rows = [
    [actual, predicted, rmse]
    for actual, predicted in zip(y_test_last_1440, y_pred_last_1440)
]


# Open csv file for writing
with open('predictions.csv', 'w') as f:

    # Create csv writer
    writer = csv.writer(f)

    # Write column headers
    writer.writerow(['Actual', 'Predicted', 'RMSE'])

    # Write each row
    writer.writerows(rows)

print('CSV saved successfully!')
```

```
948/948 [==============================] - 2s 2ms/step - loss: 156542.7812 - mae: 165.4916 - val_loss: 147191.6
Epoch 32/50
948/948 [==============================] - 3s 3ms/step - loss: 155994.7188 - mae: 165.2886 - val_loss: 146839.2
Epoch 33/50
948/948 [==============================] - 3s 3ms/step - loss: 156113.6562 - mae: 165.5784 - val_loss: 154812.1
Epoch 34/50
948/948 [==============================] - 2s 2ms/step - loss: 156032.9531 - mae: 165.2811 - val_loss: 148608.1
Epoch 35/50
948/948 [==============================] - 3s 3ms/step - loss: 155692.0312 - mae: 164.5092 - val_loss: 146388.2
Epoch 36/50
948/948 [==============================] - 3s 3ms/step - loss: 156025.9062 - mae: 164.7264 - val_loss: 148373.1
Epoch 37/50
948/948 [==============================] - 3s 3ms/step - loss: 156319.1094 - mae: 166.1944 - val_loss: 147515.0
Epoch 38/50
948/948 [==============================] - 2s 3ms/step - loss: 155490.4062 - mae: 164.3020 - val_loss: 152632.7
Epoch 39/50
948/948 [==============================] - 3s 3ms/step - loss: 155999.9531 - mae: 164.4905 - val_loss: 149924.9
Epoch 40/50
948/948 [==============================] - 2s 2ms/step - loss: 155475.1562 - mae: 164.0137 - val_loss: 146002.2
Epoch 41/50
948/948 [==============================] - 3s 3ms/step - loss: 156006.9844 - mae: 165.5979 - val_loss: 147206.3
Epoch 42/50
948/948 [==============================] - 3s 3ms/step - loss: 155207.4531 - mae: 164.0490 - val_loss: 146233.5
Epoch 43/50
948/948 [==============================] - 3s 3ms/step - loss: 155571.5625 - mae: 165.0288 - val_loss: 151378.4
Epoch 44/50
948/948 [==============================] - 2s 2ms/step - loss: 155613.5938 - mae: 164.3455 - val_loss: 149317.3
Epoch 45/50
948/948 [==============================] - 2s 2ms/step - loss: 155217.0469 - mae: 163.5653 - val_loss: 146856.5
Epoch 46/50
948/948 [==============================] - 2s 2ms/step - loss: 155785.1406 - mae: 165.1904 - val_loss: 146938.0
Epoch 47/50
948/948 [==============================] - 3s 3ms/step - loss: 155809.5312 - mae: 164.3073 - val_loss: 145931.0
Epoch 48/50
948/948 [==============================] - 3s 3ms/step - loss: 156110.3281 - mae: 164.8430 - val_loss: 151877.2
Epoch 49/50
948/948 [==============================] - 2s 2ms/step - loss: 155121.9219 - mae: 164.4887 - val_loss: 150465.2
Epoch 50/50
948/948 [==============================] - 3s 3ms/step - loss: 156267.7500 - mae: 166.2858 - val_loss: 147108.2
316/316 [==============================] - 0s 1ms/step - loss: 157534.2969 - mae: 152.2066
Test Mean Absolute Error: 152.20657348632812
45/45 [==============================] - 0s 1ms/step
            Actual     Predicted          RMSE
35410   720.594482  2.178398e+06  6.668939e+06
23250   632.309814  2.379462e+06  6.668939e+06
35570   470.704193  1.712162e+06  6.668939e+06
11715  1920.000000  6.645934e+06  6.668939e+06
40256   556.119385  2.066000e+06  6.668939e+06
...            ...           ...           ...
2254   3414.837891  1.243454e+07  6.668939e+06
6224   1328.156982  3.682776e+06  6.668939e+06
33427   215.810501  8.227719e+05  6.668939e+06
36504  3389.184082  1.181621e+07  6.668939e+06
28445     0.000000  2.881901e+03  6.668939e+06

[1440 rows x 3 columns]
CSV saved successfully!
```

Future Prediction - 24 hrs

```
import numpy as np

# Train model only on training data
model.fit(X_train, y_train, epochs=50)

# No of features
no_features = X_train.shape[1]

# Create dummy 2D input
next_1440 = pd.DataFrame(np.zeros((1440, no_features)))

# Make predictions
y_pred_next_1440 = model.predict(next_1440)

# Inverse transform predictions
y_pred_next_1440 = scaler.inverse_transform(y_pred_next_1440)

# RMSE will be Nan as we don't have actual values
rmse = [np.nan]*len(y_pred_next_1440)

# Prepare rows for writing to CSV
rows = [
    [np.nan, pred, rmse_val]
    for pred, rmse_val in zip(y_pred_next_1440, rmse)
]

# Write predictions to CSV
with open('future_predictions.csv', 'w') as f:
    writer = csv.writer(f)
    writer.writerow(['Actual','Predicted','RMSE'])
    writer.writerows(rows)

print(rmse)

print('Future predictions saved to CSV')
```

```
Epoch 45/50
948/948 [==============================] - 2s 2ms/step - loss: 152646.2031 - mae: 161.8010
Epoch 46/50
948/948 [==============================] - 2s 2ms/step - loss: 153158.4062 - mae: 162.5629
Epoch 47/50
948/948 [==============================] - 2s 2ms/step - loss: 153060.6250 - mae: 161.9228
Epoch 48/50
948/948 [==============================] - 2s 2ms/step - loss: 152677.3750 - mae: 161.1163
Epoch 49/50
948/948 [==============================] - 2s 2ms/step - loss: 152651.3594 - mae: 161.9837
Epoch 50/50
948/948 [==============================] - 2s 2ms/step - loss: 152710.7656 - mae: 162.2879
45/45 [==============================] - 0s 1ms/step
[nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan, nan,
Future predictions saved to CSV
```

Please use following link to see output files and resources -
https://drive.google.com/drive/folders/1P3PMPAP9C5IkDpmq3iyoC6WPdJL2SYMV?usp=sharing