Untitled

```
import os
import random
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout
from tensorflow.keras.callbacks import EarlyStopping




df = pd.read_csv("../Data/cleaned_data.csv")
```

```
df['Date'] = pd.to_datetime(df['Year'].astype(str) + '-' + df['Month'].astype(str) + '-01')
df = df.sort_values('Date')

features = [
    'Transportarion Petroleum Consumption',
    'Total Fossil Fuels Production',
    'Total Renewable Energy Production',
    'Commercial_Consumption',
    'Industrial_Consumption',
    'Residential_Consumption',
    'Month'
]
target = 'Total_CO2_Emissions'
data_all = df[features + [target, 'Date']].dropna()
```

```
# normalize the data
feature_scaler = MinMaxScaler()
target_scaler = MinMaxScaler()

data_scaled = data_all.copy()
data_scaled[features] = feature_scaler.fit_transform(data_all[features])
data_scaled[[target]] = target_scaler.fit_transform(data_all[[target]])

# Construct sliding window samples
def make_sequences(data, seq_len=12):
    X, y, dates = [], [], []
    for i in range(len(data) - seq_len):
        X.append(data.iloc[i:i+seq_len][features].values)
        y.append(data.iloc[i+seq_len][target])
        dates.append(data.iloc[i+seq_len]['Date'])
    return np.array(X), np.array(y), np.array(dates)

seq_len = 24
X_all, y_all, dates_all = make_sequences(data_scaled, seq_len)

# train validation test set split
train_end = pd.Timestamp("2020-12-31")
valid_end = pd.Timestamp("2021-12-31")

train_idx = dates_all <= train_end
valid_idx = (dates_all > train_end) & (dates_all <= valid_end)
test_idx = dates_all > valid_end

X_train, y_train = X_all[train_idx], y_all[train_idx]
X_valid, y_valid = X_all[valid_idx], y_all[valid_idx]
X_test, y_test = X_all[test_idx], y_all[test_idx]
```

```
def set_seed(seed=42):
    os.environ['PYTHONHASHSEED'] = str(seed)
    random.seed(seed)
    np.random.seed(seed)
    tf.random.set_seed(seed)
    tf.config.experimental.enable_op_determinism()

set_seed(42)

# Construct the model
```

```
model = Sequential([
    LSTM(128, return_sequences=True, input_shape=(X_train.shape[1], X_train.shape[2])),
    Dropout(0.3),
    LSTM(64),
    Dense(1)
])
model.compile(optimizer='adam', loss='mse')

early_stop = EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True)

history = model.fit(
    X_train, y_train,
    validation_data=(X_valid, y_valid),
    epochs=100,
    batch_size=16,
    callbacks=[early_stop],
    verbose=0
)
```

```
# loss function
plt.figure(figsize=(8, 5))
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title("Loss Curve During Training")
plt.xlabel("Epoch")
plt.ylabel("MSE Loss")
plt.legend()
plt.grid(True)
plt.tight_layout()

os.makedirs("Charts", exist_ok=True)
plt.savefig("Charts/loss_curve.png", dpi=300)
plt.close()
```

```
def predict_and_inverse(X, y, scaler):
    y_pred = model.predict(X, verbose=0)
    y_pred_inv = scaler.inverse_transform(y_pred.reshape(-1, 1))
    y_true_inv = scaler.inverse_transform(y.reshape(-1, 1))
    return y_true_inv, y_pred_inv

y_train_inv, y_pred_train_inv = predict_and_inverse(X_train, y_train, target_scaler)
y_valid_inv, y_pred_valid_inv = predict_and_inverse(X_valid, y_valid, target_scaler)
```

```
y_test_inv,  y_pred_test_inv  = predict_and_inverse(X_test,  y_test,  target_scaler)

# MSE evaluation for both training validation and prediction set
def print_rmse(label, y_true, y_pred):
    mse = mean_squared_error(y_true, y_pred)
    rmse = np.sqrt(mse)
    print(f"{label} MSE: {mse:.2f}, RMSE: {rmse:.2f}")
    return rmse

train_rmse = print_rmse("Train", y_train_inv, y_pred_train_inv)
valid_rmse = print_rmse("Valid", y_valid_inv, y_pred_valid_inv)
test_rmse  = print_rmse("Test",  y_test_inv,  y_pred_test_inv)
```

```
2025-06-20 21:14:05.324530: E tensorflow/core/framework/node_def_util.cc:680] NodeDef mentio
2025-06-20 21:14:05.324826: E tensorflow/core/framework/node_def_util.cc:680] NodeDef mentio
```

```
Train MSE: 2329.53, RMSE: 48.27
Valid MSE: 469.48, RMSE: 21.67
Test MSE: 1828.74, RMSE: 42.76
```

```
# Prediction on Test dataset
plt.figure(figsize=(10, 6))
plt.plot(dates_all[test_idx], y_test_inv, label='True')
plt.plot(dates_all[test_idx], y_pred_test_inv, label='Predicted')
plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%Y'))
plt.gca().xaxis.set_major_locator(mdates.YearLocator())
plt.title('LSTM Forecast (2022 and beyond)')
plt.xlabel("Year")
plt.ylabel("Total CO  Emissions")
plt.legend()
plt.tight_layout()

os.makedirs("Charts", exist_ok=True)
plt.savefig("Charts/test_forecast.png", dpi=300)
plt.close()
```