In [1]:

```python
import pandas as pd
import numpy as np
from sklearn.model_selection import KFold
from sklearn.metrics import mean_squared_error, mean_absolute_error
from sklearn.preprocessing import StandardScaler
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv1D, MaxPooling1D, Flatten, Dense
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping
```

In [2]:

```python
# Load the dataset
data = pd.read_csv('1M_ahead_dataset.csv')
```

In [3]:

```python
# Separate predictors (X) and target (y)
X = data.drop(['Yt.1M'], axis=1).values
y = data['Yt.1M'].values
```

In [4]:

```python
# Scale the features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

In [5]:

```python
# Reshape X for 1D CNN (samples, timesteps, channels)
# Here we treat each feature as a "time step" with one channel.
X_cnn = X_scaled.reshape(X_scaled.shape[0], X_scaled.shape[1], 1)
```

In [6]:

```python
# Set up 5-fold cross validation
kf = KFold(n_splits=5, shuffle=True, random_state=42)
fold_metrics = []
fold_counter = 1
```

In [7]:

```python
# Iterate over each fold
for train_index, test_index in kf.split(X_cnn):
    print(f"\n--- Fold {fold_counter} ---")
    X_train, X_test = X_cnn[train_index], X_cnn[test_index]
    y_train, y_test = y[train_index], y[test_index]

    # Define the 1D CNN model
    model = Sequential()
    # First convolutional block
    model.add(Conv1D(filters=32, kernel_size=3, activation='relu', input_shape=(X_train.shape[1], 1)))
    model.add(MaxPooling1D(pool_size=2))
    # Second convolutional block
    model.add(Conv1D(filters=64, kernel_size=3, activation='relu'))
    model.add(MaxPooling1D(pool_size=2))
    model.add(Flatten())
    # A Dense layer before the output
    model.add(Dense(50, activation='relu'))
    # Output layer for regression
    model.add(Dense(1))

    # Compile the model
    model.compile(optimizer=Adam(), loss='mse')

    # Early stopping callback to prevent overfitting
    es = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)

    # Train the model – note that validation_split applies only to the training data
    history = model.fit(X_train, y_train, epochs=50, batch_size=32, validation_split=0.1, verbose=0, callbacks=

    # Make predictions on the test set
    y_pred = model.predict(X_test).flatten()

    # Compute evaluation metrics
    mse = mean_squared_error(y_test, y_pred)
    mae = mean_absolute_error(y_test, y_pred)
    rmse = np.sqrt(mse)

    fold_metrics.append({'Fold': fold_counter, 'MSE': mse, 'RMSE': rmse, 'MAE': mae})
    print(f"Fold {fold_counter} -- MSE: {mse:.4f}, RMSE: {rmse:.4f}, MAE: {mae:.4f}")

    fold_counter += 1
```

```
--- Fold 1 ---
6/6 [==============================] - 0s 10ms/step
Fold 1 -- MSE: 0.0162, RMSE: 0.1273, MAE: 0.0736

--- Fold 2 ---
6/6 [==============================] - 0s 5ms/step
Fold 2 -- MSE: 0.0191, RMSE: 0.1382, MAE: 0.0760

--- Fold 3 ---
6/6 [==============================] - 0s 14ms/step
Fold 3 -- MSE: 0.0140, RMSE: 0.1181, MAE: 0.0725

--- Fold 4 ---
6/6 [==============================] - 0s 3ms/step
Fold 4 -- MSE: 0.0132, RMSE: 0.1151, MAE: 0.0798

--- Fold 5 ---
6/6 [==============================] - 0s 3ms/step
Fold 5 -- MSE: 0.0220, RMSE: 0.1482, MAE: 0.0800
```

In [8]:

```
# Summarize the results into a DataFrame
results_df = pd.DataFrame(fold_metrics)
print("\nOverall Cross-Validation Results:")
print(results_df)
```

```
Overall Cross-Validation Results:
   Fold       MSE      RMSE       MAE
0     1  0.016205  0.127300  0.073573
1     2  0.019089  0.138162  0.075952
2     3  0.013952  0.118118  0.072540
3     4  0.013242  0.115076  0.079827
4     5  0.021969  0.148220  0.079978
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js