In [1]:
```
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 Dense, Dropout
from tensorflow.keras.callbacks import EarlyStopping
```

In [2]:
```
# Load the dataset
data = pd.read_csv('1M_ahead_dataset.csv')
```

In [3]:
```
# Separate predictors (X) and target (y)
X = data.drop(['Yt.1M'], axis=1).values
y = data['Yt.1M'].values
```

In [4]:
```
# Scale the predictor features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

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

In [6]:
```
# Iterate through each fold
for train_index, test_index in kf.split(X_scaled):
    print(f"\n--- Fold {fold_counter} ---")
    # Split the data for this fold
    X_train, X_test = X_scaled[train_index], X_scaled[test_index]
    y_train, y_test = y[train_index], y[test_index]

    # Build a deep neural network model
    model = Sequential()
    # Input layer and first hidden layer
    model.add(Dense(64, input_dim=X_train.shape[1], activation='relu'))
    model.add(Dropout(0.2))
    # Second hidden layer
    model.add(Dense(32, activation='relu'))
    model.add(Dropout(0.2))
    # Third hidden layer
    model.add(Dense(16, activation='relu'))
    # Output layer for regression
    model.add(Dense(1))

    # Compile the model using Mean Squared Error loss
    model.compile(optimizer='adam', loss='mse')

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

    # Train the model using a validation split from training data
    history = model.fit(X_train, y_train, epochs=100, batch_size=32,
                        validation_split=0.1, verbose=0, callbacks=[es])

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

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

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

    fold_counter += 1
```

```
--- Fold 1 ---
6/6 [==============================] - 0s 4ms/step
Fold 1 -- MSE: 0.0167, RMSE: 0.1293, MAE: 0.0729

--- Fold 2 ---
6/6 [==============================] - 0s 3ms/step
Fold 2 -- MSE: 0.0188, RMSE: 0.1372, MAE: 0.0744

--- Fold 3 ---
6/6 [==============================] - 0s 3ms/step
Fold 3 -- MSE: 0.0149, RMSE: 0.1220, MAE: 0.0750

--- Fold 4 ---
6/6 [==============================] - 0s 4ms/step
Fold 4 -- MSE: 0.0131, RMSE: 0.1144, MAE: 0.0778

--- Fold 5 ---
6/6 [==============================] - 0s 2ms/step
Fold 5 -- MSE: 0.0182, RMSE: 0.1348, MAE: 0.0744
```

In [7]:

```
# Summarize the cross-validation results in 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  0.016710  0.129269  0.072892
1     2  0.018835  0.137242  0.074356
2     3  0.014872  0.121952  0.074955
3     4  0.013078  0.114359  0.077816
4     5  0.018179  0.134831  0.074386
```

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