1: Import Libraries

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from statsmodels.tsa.arima.model import ARIMA
from statsmodels.tsa.stattools import adfuller
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf

from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_sco
re
from statsmodels.tools.sm_exceptions import ValueWarning

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout, GroupNormalizatio
n
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.models import load_model

from tqdm import tqdm

from tcn import TCN
import shap

import joblib

import warnings
warnings.filterwarnings("ignore", category=UserWarning)
warnings.filterwarnings("ignore", category=FutureWarning)
warnings.filterwarnings("ignore", category=ValueWarning)

import os
import sys
print(sys.executable)
```

```
b:\DCU\Practicum\Proj\App\venv_3_11\Scripts\python.exe

b:\DCU\Practicum\Proj\App\venv_3_11\Lib\site-packages\tqdm\auto.py:21: Tqd
mWarning: IProgress not found. Please update jupyter and ipywidgets. See h
ttps://ipywidgets.readthedocs.io/en/stable/user_install.html
  from .autonotebook import tqdm as notebook_tqdm
```

2: Load and Explore Dataset

In [2]:
```
multimodal = pd.read_csv("train_dataset.csv", parse_dates=["Date"])
multimodal.drop(columns=['Next_Close', 'Next_3_Close', 'Next_7_Close'], inp
lace=True)
multimodal.head()
```

Out[2]:

	Date	Open	High	Low	Close	Volume	Adj Close	l
0	2008-08-08	11432.089844	11759.959961	11388.040039	11734.320312	212830000	11734.320312	
1	2008-08-11	11729.669922	11867.110352	11675.530273	11782.349609	183190000	11782.349609	
2	2008-08-12	11781.700195	11782.349609	11601.519531	11642.469727	173590000	11642.469727	
3	2008-08-13	11632.809570	11633.780273	11453.339844	11532.959961	182550000	11532.959961	
4	2008-08-14	11532.070312	11718.280273	11450.889648	11615.929688	159790000	11615.929688	

5 rows × 53 columns

In [3]:
```
multimodal.shape
```

Out[3]: (1591, 53)

In [4]:
```
missing_values = multimodal.isnull().sum()
print("\nMissing values per column:")
missing_values[missing_values > 0]
```

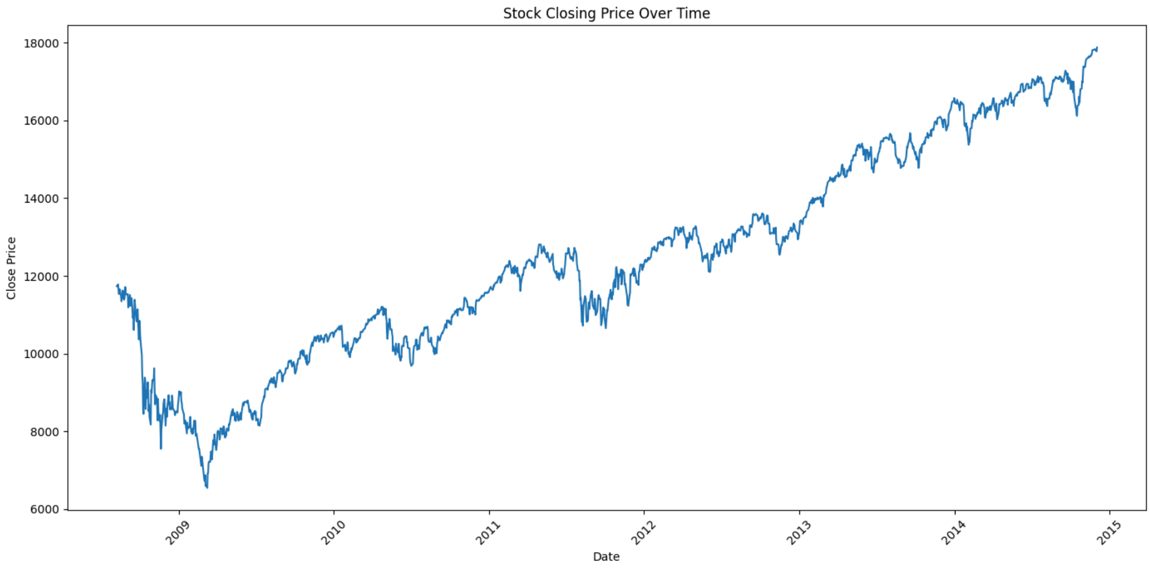
```
Missing values per column:
```

Out[4]: Series([], dtype: int64)

In [5]:
```
multimodal.describe()
multimodal.dtypes
```

Out[5]:
```
Date                              datetime64[ns]
Open                                     float64
High                                     float64
Low                                      float64
Close                                    float64
Volume                                     int64
Adj Close                                float64
Log_Returns                              float64
Volatility_Log_10                        float64
cl-op                                    float64
hi-lo                                    float64
Label                                      int64
vader_news_sentiment                     float64
FinBERT_news_sentiment                   float64
Smart_news_sentiment                     float64
news_buying_intent                       float64
news_selling_intent                      float64
news_uncertainty_intent                  float64
news_urgency_intent                      float64
news_prediction_intent                   float64
news_fear_intent                         float64
news_greed_intent                        float64
news_question_intent                     float64
news_action_intent                       float64
vader_reddit_sentiment                   float64
FinBERT_reddit_sentiment                 float64
Smart_reddit_sentiment                   float64
reddit_buying_intent                     float64
reddit_selling_intent                    float64
reddit_uncertainty_intent                float64
reddit_urgency_intent                    float64
reddit_prediction_intent                 float64
reddit_fear_intent                       float64
reddit_greed_intent                      float64
reddit_question_intent                   float64
reddit_action_intent                     float64
Target                                     int64
pct_change                               float64
finbert_final_sentiment                  float64
total_buying_intent                      float64
total_selling_intent                     float64
total_uncertainty_intent                 float64
total_urgency_intent                     float64
total_prediction_intent                  float64
total_fear_intent                        float64
total_greed_intent                       float64
total_question_intent                    float64
total_action_intent                      float64
sentiment_minus_uncertainty              float64
sentiment_minus_fear                     float64
sentiment_minus_action                   float64
sentiment_minus_urgency                  float64
sentiment_minus_prediction               float64
dtype: object
```

In [6]:
```
plt.figure(figsize=(14, 7))
plt.plot(pd.to_datetime(multimodal['Date']), multimodal['Close'])
plt.title('Stock Closing Price Over Time')
plt.xlabel('Date')
plt.ylabel('Close Price')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



3: Data Preprocessing

In [7]:
```
print(f"Shape: {multimodal.shape}")
print(multimodal.isnull().sum())
```

```
Shape: (1591, 53)
Date                           0
Open                           0
High                           0
Low                            0
Close                          0
Volume                         0
Adj Close                      0
Log_Returns                    0
Volatility_Log_10              0
cl-op                          0
hi-lo                          0
Label                          0
vader_news_sentiment           0
FinBERT_news_sentiment         0
Smart_news_sentiment           0
news_buying_intent             0
news_selling_intent            0
news_uncertainty_intent        0
news_urgency_intent            0
news_prediction_intent         0
news_fear_intent               0
news_greed_intent              0
news_question_intent           0
news_action_intent             0
vader_reddit_sentiment         0
FinBERT_reddit_sentiment       0
Smart_reddit_sentiment         0
reddit_buying_intent           0
reddit_selling_intent          0
reddit_uncertainty_intent      0
reddit_urgency_intent          0
reddit_prediction_intent       0
reddit_fear_intent             0
reddit_greed_intent            0
reddit_question_intent         0
reddit_action_intent           0
Target                         0
pct_change                     0
finbert_final_sentiment        0
total_buying_intent            0
total_selling_intent           0
total_uncertainty_intent       0
total_urgency_intent           0
total_prediction_intent        0
total_fear_intent              0
total_greed_intent             0
total_question_intent          0
total_action_intent            0
sentiment_minus_uncertainty    0
sentiment_minus_fear           0
sentiment_minus_action         0
sentiment_minus_urgency        0
sentiment_minus_prediction     0
dtype: int64
```

In [8]:
```
df_targets = multimodal[["Date", "Target", "Label", "Close"]].copy()

df_arima = multimodal[["Date", "Close"]].copy()
df_arima.set_index("Date", inplace=True)

drop_cols_lstm = ["Date", "Label", "Target"]
df_lstm = multimodal.drop(columns=drop_cols_lstm).copy()

drop_cols_tcn = ["Date", "Label", "Target"]
df_tcn = multimodal.drop(columns=drop_cols_tcn).copy()
```

In [9]:
```
df_tcn.columns
```

Out[9]:
```
Index(['Open', 'High', 'Low', 'Close', 'Volume', 'Adj Close', 'Log_Return
s',
       'Volatility_Log_10', 'cl-op', 'hi-lo', 'vader_news_sentiment',
       'FinBERT_news_sentiment', 'Smart_news_sentiment', 'news_buying_inte
nt',
       'news_selling_intent', 'news_uncertainty_intent', 'news_urgency_int
ent',
       'news_prediction_intent', 'news_fear_intent', 'news_greed_intent',
       'news_question_intent', 'news_action_intent', 'vader_reddit_sentime
nt',
       'FinBERT_reddit_sentiment', 'Smart_reddit_sentiment',
       'reddit_buying_intent', 'reddit_selling_intent',
       'reddit_uncertainty_intent', 'reddit_urgency_intent',
       'reddit_prediction_intent', 'reddit_fear_intent', 'reddit_greed_int
ent',
       'reddit_question_intent', 'reddit_action_intent', 'pct_change',
       'finbert_final_sentiment', 'total_buying_intent',
       'total_selling_intent', 'total_uncertainty_intent',
       'total_urgency_intent', 'total_prediction_intent', 'total_fear_inte
nt',
       'total_greed_intent', 'total_question_intent', 'total_action_inten
t',
       'sentiment_minus_uncertainty', 'sentiment_minus_fear',
       'sentiment_minus_action', 'sentiment_minus_urgency',
       'sentiment_minus_prediction'],
      dtype='object')
```

4: Time Series Stationarity Analysis for ARIMA

In [10]:
```
df_arima.head()
```

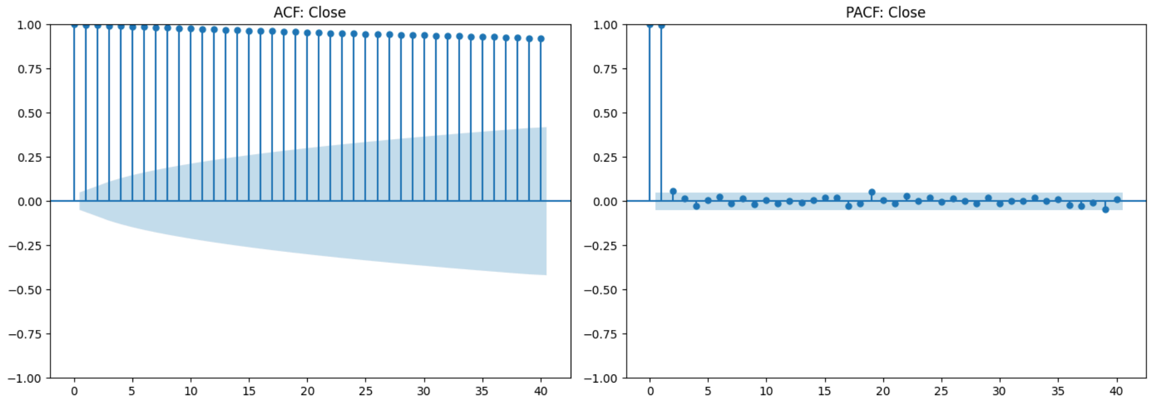
Out[10]:

	Close
Date	
2008-08-08	11734.320312
2008-08-11	11782.349609
2008-08-12	11642.469727
2008-08-13	11532.959961
2008-08-14	11615.929688

In [11]:
```
result = adfuller(df_arima["Close"])
print(f"ADF Statistic: {result[0]}")
print(f"p-value: {result[1]}")
```

```
ADF Statistic: 0.49003049114940356
p-value: 0.9845669104126585
```

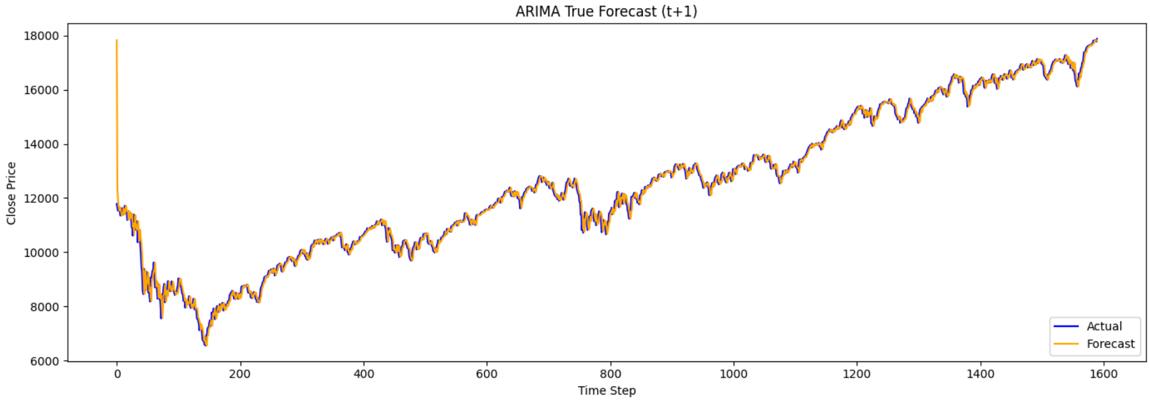
In [12]:
```
plt.figure(figsize=(14, 5))
plt.subplot(1, 2, 1)
plot_acf(df_arima["Close"], lags=40, ax=plt.gca())
plt.title("ACF: Close")

plt.subplot(1, 2, 2)
plot_pacf(df_arima["Close"], lags=40, ax=plt.gca(), method='ywm')
plt.title("PACF: Close")

plt.tight_layout()
plt.show()
```



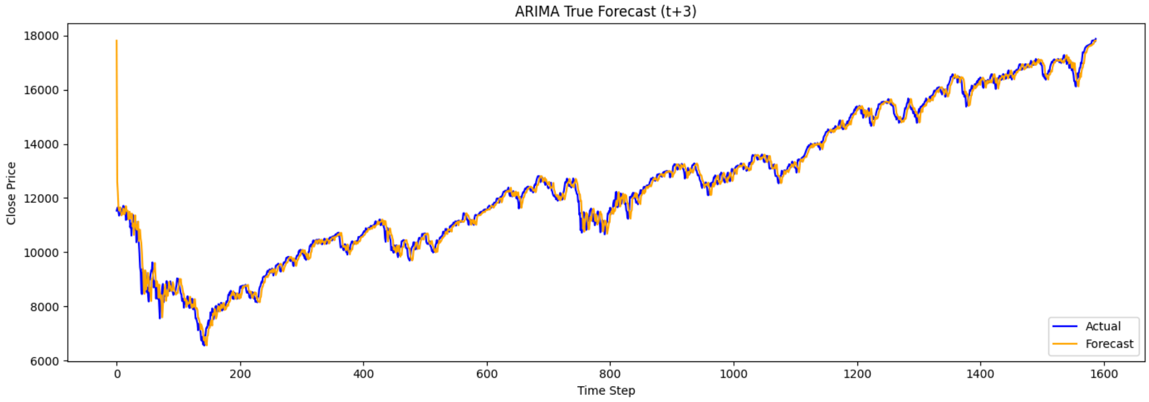
In [13]:
```
def arima_true_forecast(df_arima, forecast_horizon=1, order=(1, 1, 1), save
_dir=None, plot=True):
    close_series = df_arima['Close'].values
    history = close_series[:-(forecast_horizon + 1)].tolist()
    y_true = []
    y_pred = []

    for t in range(len(close_series) - forecast_horizon):
        try:
            model = ARIMA(history, order=order).fit()
            forecast = model.forecast(steps=forecast_horizon)
            y_pred.append(forecast[-1])
            y_true.append(close_series[t + forecast_horizon])
        except:
            y_pred.append(np.nan)
            y_true.append(np.nan)
        history.append(close_series[t])

    y_true = np.array(y_true)
    y_pred = np.array(y_pred)
    mask = ~np.isnan(y_pred)
    y_true = y_true[mask]
    y_pred = y_pred[mask]

    # Metrics
    r2 = r2_score(y_true, y_pred)
    rmse = np.sqrt(mean_squared_error(y_true, y_pred))
    mae = mean_absolute_error(y_true, y_pred)

    print(f"✅ ARIMA Forecast Horizon = {forecast_horizon}")
    print(f"R² = {r2:.4f}, RMSE = {rmse:.2f}, MAE = {mae:.2f}")

    if plot:
        plt.figure(figsize=(14, 5))
        plt.plot(y_true, label='Actual', color='blue')
        plt.plot(y_pred, label='Forecast', color='orange')
        plt.title(f"ARIMA True Forecast (t+{forecast_horizon})")
        plt.xlabel("Time Step")
        plt.ylabel("Close Price")
        plt.legend()
        plt.tight_layout()
        plt.show()

    # Optionally save model trained on full data
    if save_dir:
        os.makedirs(save_dir, exist_ok=True)
        model = ARIMA(close_series.tolist(), order=order).fit()
        save_path = os.path.join(save_dir, f'arima_t_plus_{forecast_horizo
n}.pkl')
        joblib.dump(model, save_path)
        print(f"💾 Model saved to {save_path}")

    return {
        "horizon": forecast_horizon,
        "r2": r2,
        "rmse": rmse,
        "mae": mae
    }

results = []
```

```
for h in [1, 3, 7]:
    res = arima_true_forecast(df_arima, forecast_horizon=h, order=(1,1,1),
save_dir="B:/DCU/Practicum/Proj/Models")
    results.append(res)
```

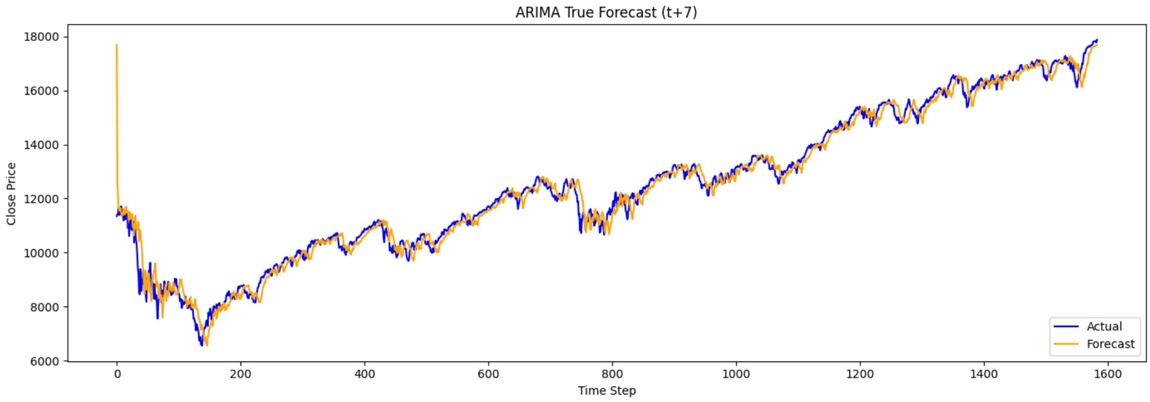
✅ ARIMA Forecast Horizon = 1
R² = 0.9921, RMSE = 238.87, MAE = 136.61



💾 Model saved to B:/DCU/Practicum/Proj/Models\arima_t_plus_1.pkl
✅ ARIMA Forecast Horizon = 3
R² = 0.9877, RMSE = 297.54, MAE = 190.21



💾 Model saved to B:/DCU/Practicum/Proj/Models\arima_t_plus_3.pkl
✅ ARIMA Forecast Horizon = 7
R² = 0.9800, RMSE = 380.06, MAE = 260.24



💾 Model saved to B:/DCU/Practicum/Proj/Models\arima_t_plus_7.pkl

LSTM Model

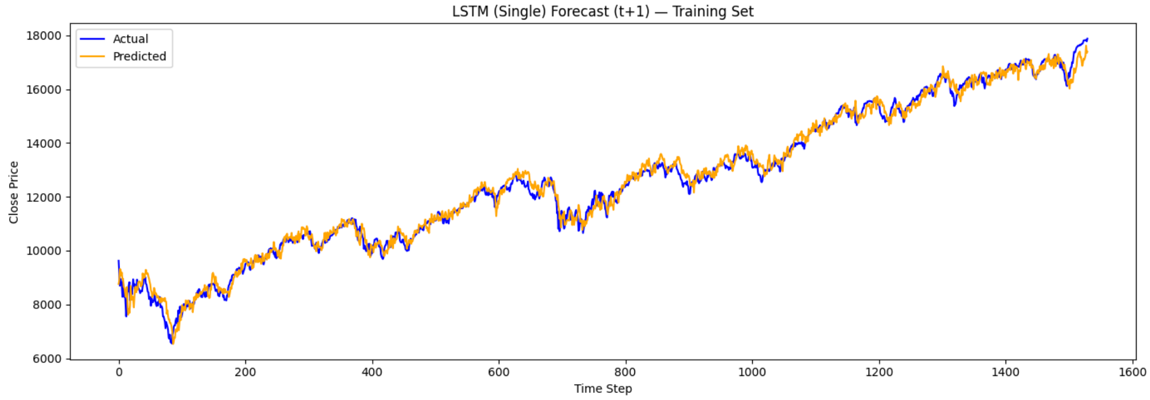
In [37]:
```
df_lstm.columns
```

Out[37]:
```
Index(['Open', 'High', 'Low', 'Close', 'Volume', 'Adj Close', 'Log_Return
s',
       'Volatility_Log_10', 'cl-op', 'hi-lo', 'vader_news_sentiment',
       'FinBERT_news_sentiment', 'Smart_news_sentiment', 'news_buying_inte
nt',
       'news_selling_intent', 'news_uncertainty_intent', 'news_urgency_int
ent',
       'news_prediction_intent', 'news_fear_intent', 'news_greed_intent',
       'news_question_intent', 'news_action_intent', 'vader_reddit_sentime
nt',
       'FinBERT_reddit_sentiment', 'Smart_reddit_sentiment',
       'reddit_buying_intent', 'reddit_selling_intent',
       'reddit_uncertainty_intent', 'reddit_urgency_intent',
       'reddit_prediction_intent', 'reddit_fear_intent', 'reddit_greed_int
ent',
       'reddit_question_intent', 'reddit_action_intent', 'pct_change',
       'finbert_final_sentiment', 'total_buying_intent',
       'total_selling_intent', 'total_uncertainty_intent',
       'total_urgency_intent', 'total_prediction_intent', 'total_fear_inte
nt',
       'total_greed_intent', 'total_question_intent', 'total_action_inten
t',
       'sentiment_minus_uncertainty', 'sentiment_minus_fear',
       'sentiment_minus_action', 'sentiment_minus_urgency',
       'sentiment_minus_prediction'],
      dtype='object')
```

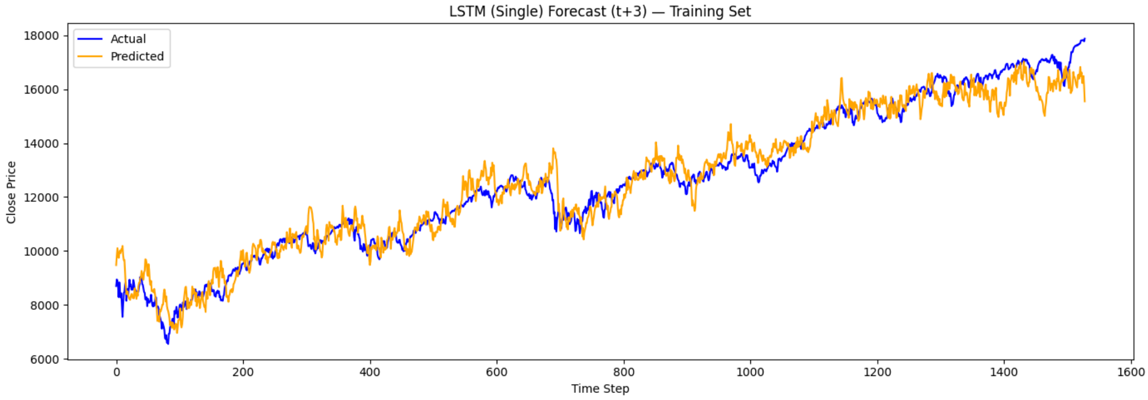
In [28]:
```
def train_lstm_multistep(df_lstm, forecast_horizon=1, window_size=60, epoch
s=50, batch_size=32, stacked=False):
    print(f"\n🧠 Training LSTM ({'Stacked' if stacked else 'Single'}) mode
l for horizon t+{forecast_horizon}")

    close_values = df_lstm['Close'].values

    # Scale input and output
    X_scaler = MinMaxScaler()
    y_scaler = MinMaxScaler()
    X_scaled = X_scaler.fit_transform(df_lstm.values)
    y_scaled = y_scaler.fit_transform(close_values.reshape(-1, 1))

    # Create sequences
    X_seq, y_seq = [], []
    for i in range(window_size, len(X_scaled) - forecast_horizon):
        X_seq.append(X_scaled[i - window_size:i])
        y_seq.append(y_scaled[i + forecast_horizon])
    X_seq, y_seq = np.array(X_seq), np.array(y_seq)

    X_train, y_train = X_seq, y_seq

    # Build model
    model = Sequential()
    if stacked:
        model.add(LSTM(64, return_sequences=True, input_shape=(X_train.shap
e[1], X_train.shape[2])))
        model.add(Dropout(0.2))
        model.add(LSTM(32))
    else:
        model.add(LSTM(64, input_shape=(X_train.shape[1], X_train.shape
[2])))
    model.add(Dropout(0.2))
    model.add(Dense(1))

    optimizer = Adam(learning_rate=1e-4)
    model.compile(optimizer=optimizer, loss='mse')
    early_stop = EarlyStopping(monitor='val_loss', patience=5, restore_best
_weights=True)

    model.fit(
        X_train, y_train,
        epochs=epochs,
        batch_size=batch_size,
        validation_split=0.1,
        callbacks=[early_stop],
        verbose=1
    )

    # Save model and scalers
    model_name = f"lstm_tplus{forecast_horizon}{'_stacked' if stacked else
'_simple'}"
    base_path = "B:/DCU/Practicum/Proj/Models"
    os.makedirs(base_path, exist_ok=True)

    model.save(f"{base_path}/{model_name}.keras")
    joblib.dump(X_scaler, f"{base_path}/{model_name}_scalerX.pkl")
    joblib.dump(y_scaler, f"{base_path}/{model_name}_scalerY.pkl")
    print(f"✅ Saved model and scalers: {model_name}")
```

```
    # Predict on train set to evaluate
    y_pred_scaled = model.predict(X_train)
    y_pred = y_scaler.inverse_transform(y_pred_scaled)
    y_true = y_scaler.inverse_transform(y_train.reshape(-1, 1))

    # Metrics
    r2 = r2_score(y_true, y_pred)
    rmse = np.sqrt(mean_squared_error(y_true, y_pred))
    mae = mean_absolute_error(y_true, y_pred)
    print(f"📊 Train R²: {r2:.4f}, RMSE: {rmse:.2f}, MAE: {mae:.2f}")

    # Plot
    plt.figure(figsize=(14, 5))
    plt.plot(y_true, label='Actual', color='blue')
    plt.plot(y_pred, label='Predicted', color='orange')
    plt.title(f"LSTM ({'Stacked' if stacked else 'Single'}) Forecast (t+{fo
recast_horizon}) — Training Set")
    plt.xlabel("Time Step")
    plt.ylabel("Close Price")
    plt.legend()
    plt.tight_layout()
    plt.show()


    return model, {"r2": r2, "rmse": rmse, "mae": mae}

for horizon in [1, 3, 7]:
    train_lstm_multistep(df_lstm, forecast_horizon=horizon, stacked=False)

for horizon in [1, 3, 7]:
    train_lstm_multistep(df_lstm, forecast_horizon=horizon, stacked=True)
```

```
🧠 Training LSTM (Single) model for horizon t+1
Epoch 1/50
44/44 ━━━━━━━━━━━━━━━━━━━━ 2s 22ms/step - loss: 0.0477 - val_loss: 0.0278
Epoch 2/50
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 17ms/step - loss: 0.0202 - val_loss: 0.0090
Epoch 3/50
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 16ms/step - loss: 0.0154 - val_loss: 0.0108
Epoch 4/50
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 15ms/step - loss: 0.0127 - val_loss: 0.0045
Epoch 5/50
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 15ms/step - loss: 0.0100 - val_loss: 0.0029
Epoch 6/50
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 15ms/step - loss: 0.0070 - val_loss: 0.0051
Epoch 7/50
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step - loss: 0.0088 - val_loss: 0.0025
Epoch 8/50
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 15ms/step - loss: 0.0068 - val_loss: 0.0011
Epoch 9/50
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 18ms/step - loss: 0.0063 - val_loss: 0.0015
Epoch 10/50
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 17ms/step - loss: 0.0056 - val_loss: 0.0015
Epoch 11/50
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 18ms/step - loss: 0.0052 - val_loss: 8.3695e
-04
Epoch 12/50
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 19ms/step - loss: 0.0044 - val_loss: 7.6831e
-04
Epoch 13/50
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 18ms/step - loss: 0.0046 - val_loss: 0.0011
Epoch 14/50
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 18ms/step - loss: 0.0038 - val_loss: 0.0016
Epoch 15/50
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 19ms/step - loss: 0.0037 - val_loss: 0.0013
Epoch 16/50
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 18ms/step - loss: 0.0039 - val_loss: 0.0012
Epoch 17/50
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 17ms/step - loss: 0.0035 - val_loss: 7.2697e
-04
Epoch 18/50
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 19ms/step - loss: 0.0034 - val_loss: 0.0013
Epoch 19/50
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 19ms/step - loss: 0.0032 - val_loss: 0.0010
Epoch 20/50
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 19ms/step - loss: 0.0034 - val_loss: 8.6535e
-04
Epoch 21/50
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 19ms/step - loss: 0.0029 - val_loss: 0.0013
Epoch 22/50
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 20ms/step - loss: 0.0029 - val_loss: 0.0010
✅ Saved model and scalers: lstm_tplus1_simple
48/48 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step
📊 Train R²: 0.9894, RMSE: 278.03, MAE: 218.80
```

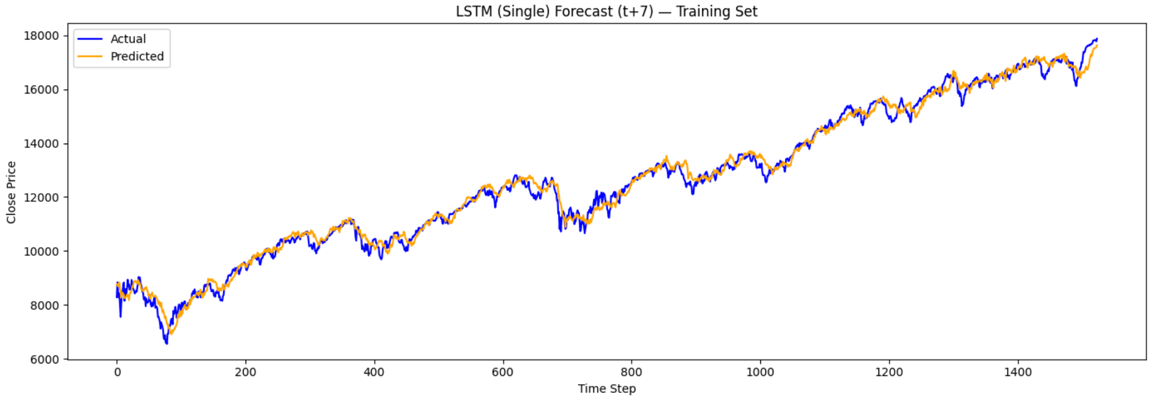



```
🧠 Training LSTM (Single) model for horizon t+3
Epoch 1/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 2s 24ms/step - loss: 0.2549 - val_loss: 0.1135
Epoch 2/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 19ms/step - loss: 0.0365 - val_loss: 0.0613
Epoch 3/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 18ms/step - loss: 0.0240 - val_loss: 0.0245
Epoch 4/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 18ms/step - loss: 0.0209 - val_loss: 0.0164
Epoch 5/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 18ms/step - loss: 0.0206 - val_loss: 0.0192
Epoch 6/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 18ms/step - loss: 0.0171 - val_loss: 0.0156
Epoch 7/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 18ms/step - loss: 0.0161 - val_loss: 0.0146
Epoch 8/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 19ms/step - loss: 0.0145 - val_loss: 0.0137
Epoch 9/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 18ms/step - loss: 0.0133 - val_loss: 0.0098
Epoch 10/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 16ms/step - loss: 0.0123 - val_loss: 0.0121
Epoch 11/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 17ms/step - loss: 0.0108 - val_loss: 0.0077
Epoch 12/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 17ms/step - loss: 0.0112 - val_loss: 0.0077
Epoch 13/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 17ms/step - loss: 0.0101 - val_loss: 0.0086
Epoch 14/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 16ms/step - loss: 0.0091 - val_loss: 0.0078
Epoch 15/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 17ms/step - loss: 0.0086 - val_loss: 0.0080
Epoch 16/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 17ms/step - loss: 0.0083 - val_loss: 0.0082
✅ Saved model and scalers: lstm_tplus3_simple
48/48 ━━━━━━━━━━━━━━━━━━━━ 1s 8ms/step
📊 Train R²: 0.9489, RMSE: 609.61, MAE: 467.26
```

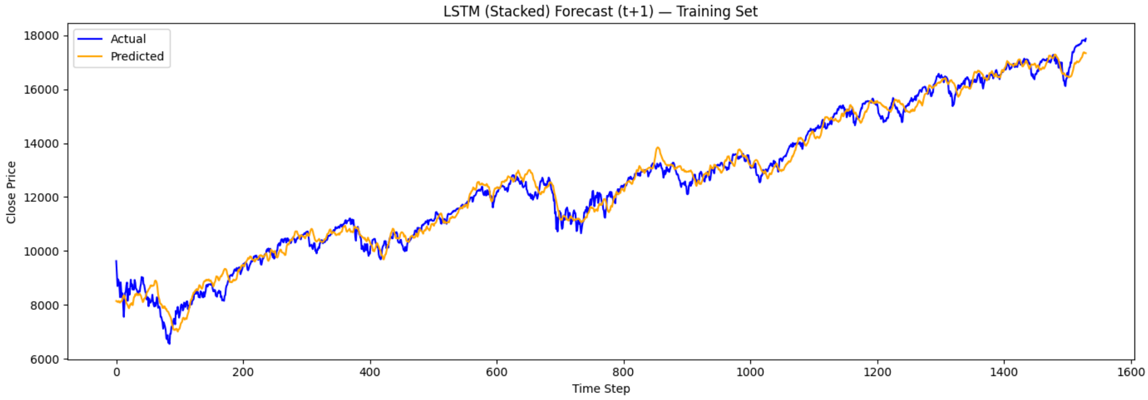

LSTM (Single) Forecast (t+3) — Training Set
Actual
Predicted
Close Price
Time Step
18000
16000
14000
12000
10000
8000
6000
0
200
400
600
800
1000
1200
1400
1600

```
🧠 Training LSTM (Single) model for horizon t+7
Epoch 1/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 2s 21ms/step - loss: 0.0691 - val_loss: 0.0853
Epoch 2/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 17ms/step - loss: 0.0244 - val_loss: 0.0221
Epoch 3/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 17ms/step - loss: 0.0172 - val_loss: 0.0137
Epoch 4/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 16ms/step - loss: 0.0152 - val_loss: 0.0122
Epoch 5/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 16ms/step - loss: 0.0125 - val_loss: 0.0086
Epoch 6/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 16ms/step - loss: 0.0110 - val_loss: 0.0086
Epoch 7/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 17ms/step - loss: 0.0097 - val_loss: 0.0073
Epoch 8/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 16ms/step - loss: 0.0079 - val_loss: 0.0060
Epoch 9/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 16ms/step - loss: 0.0075 - val_loss: 0.0050
Epoch 10/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 16ms/step - loss: 0.0065 - val_loss: 0.0047
Epoch 11/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 16ms/step - loss: 0.0063 - val_loss: 0.0052
Epoch 12/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 17ms/step - loss: 0.0062 - val_loss: 0.0038
Epoch 13/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 17ms/step - loss: 0.0055 - val_loss: 0.0042
Epoch 14/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 17ms/step - loss: 0.0047 - val_loss: 0.0031
Epoch 15/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 16ms/step - loss: 0.0049 - val_loss: 0.0030
Epoch 16/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 16ms/step - loss: 0.0044 - val_loss: 0.0029
Epoch 17/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 16ms/step - loss: 0.0042 - val_loss: 0.0026
Epoch 18/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 17ms/step - loss: 0.0038 - val_loss: 0.0025
Epoch 19/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 16ms/step - loss: 0.0036 - val_loss: 0.0024
Epoch 20/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 16ms/step - loss: 0.0038 - val_loss: 0.0021
Epoch 21/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 16ms/step - loss: 0.0038 - val_loss: 0.0021
Epoch 22/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 16ms/step - loss: 0.0036 - val_loss: 0.0021
Epoch 23/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 17ms/step - loss: 0.0038 - val_loss: 0.0018
Epoch 24/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 16ms/step - loss: 0.0033 - val_loss: 0.0017
Epoch 25/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 17ms/step - loss: 0.0035 - val_loss: 0.0021
Epoch 26/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 16ms/step - loss: 0.0033 - val_loss: 0.0017
Epoch 27/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 16ms/step - loss: 0.0031 - val_loss: 0.0015
Epoch 28/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 17ms/step - loss: 0.0032 - val_loss: 0.0020
Epoch 29/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 16ms/step - loss: 0.0029 - val_loss: 0.0014
Epoch 30/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 16ms/step - loss: 0.0029 - val_loss: 0.0017
```

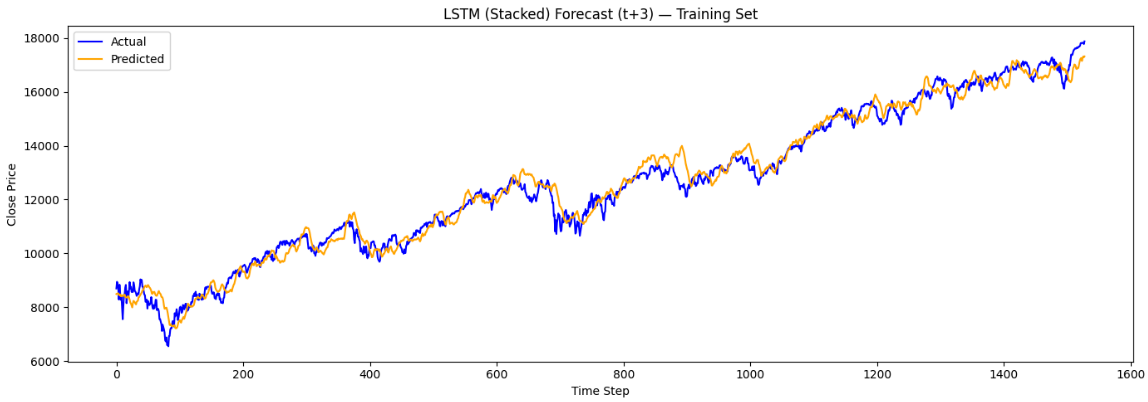
```
Epoch 31/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 16ms/step - loss: 0.0030 - val_loss: 0.0013
Epoch 32/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 17ms/step - loss: 0.0030 - val_loss: 0.0013
Epoch 33/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 16ms/step - loss: 0.0027 - val_loss: 0.0019
Epoch 34/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 17ms/step - loss: 0.0028 - val_loss: 0.0011
Epoch 35/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 17ms/step - loss: 0.0028 - val_loss: 0.0012
Epoch 36/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 17ms/step - loss: 0.0030 - val_loss: 0.0015
Epoch 37/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 16ms/step - loss: 0.0025 - val_loss: 0.0013
Epoch 38/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 16ms/step - loss: 0.0030 - val_loss: 0.0013
Epoch 39/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 17ms/step - loss: 0.0026 - val_loss: 9.6312e
-04
Epoch 40/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 16ms/step - loss: 0.0027 - val_loss: 9.4938e
-04
Epoch 41/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 16ms/step - loss: 0.0025 - val_loss: 0.0011
Epoch 42/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 17ms/step - loss: 0.0025 - val_loss: 0.0011
Epoch 43/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 17ms/step - loss: 0.0026 - val_loss: 0.0011
Epoch 44/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 18ms/step - loss: 0.0026 - val_loss: 9.1878e
-04
Epoch 45/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 19ms/step - loss: 0.0024 - val_loss: 0.0012
Epoch 46/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 17ms/step - loss: 0.0023 - val_loss: 0.0011
Epoch 47/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 16ms/step - loss: 0.0024 - val_loss: 8.5051e
-04
Epoch 48/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 18ms/step - loss: 0.0024 - val_loss: 8.2293e
-04
Epoch 49/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 16ms/step - loss: 0.0025 - val_loss: 9.2927e
-04
Epoch 50/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 15ms/step - loss: 0.0023 - val_loss: 7.6456e
-04
✅ Saved model and scalers: lstm_tplus7_simple
48/48 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step
📊 Train R²: 0.9894, RMSE: 277.03, MAE: 211.98
```


LSTM (Single) Forecast (t+7) — Training Set
Actual
Predicted
Close Price
Time Step
18000
16000
14000
12000
10000
8000
6000
0
200
400
600
800
1000
1200
1400

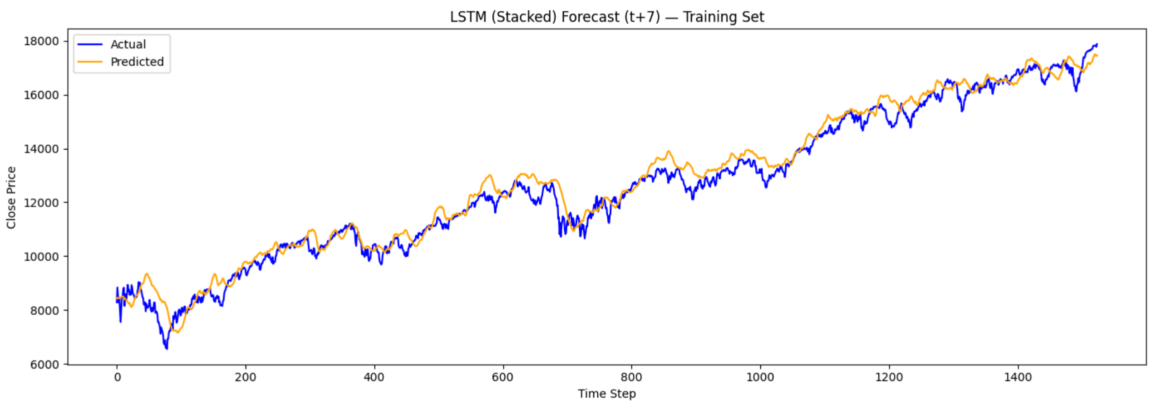
```
🧠 Training LSTM (Stacked) model for horizon t+1
Epoch 1/50
44/44 ━━━━━━━━━━━━━━━━━━━━ 4s 35ms/step - loss: 0.4146 - val_loss: 0.0736
Epoch 2/50
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 28ms/step - loss: 0.0362 - val_loss: 0.0546
Epoch 3/50
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - loss: 0.0197 - val_loss: 0.0175
Epoch 4/50
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0157 - val_loss: 0.0050
Epoch 5/50
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0112 - val_loss: 0.0046
Epoch 6/50
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 27ms/step - loss: 0.0110 - val_loss: 0.0041
Epoch 7/50
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0088 - val_loss: 0.0016
Epoch 8/50
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0082 - val_loss: 0.0033
Epoch 9/50
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0084 - val_loss: 0.0011
Epoch 10/50
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0075 - val_loss: 0.0022
Epoch 11/50
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - loss: 0.0072 - val_loss: 0.0019
Epoch 12/50
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0076 - val_loss: 7.8025e
-04
Epoch 13/50
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0073 - val_loss: 0.0012
Epoch 14/50
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0072 - val_loss: 9.6069e
-04
Epoch 15/50
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0063 - val_loss: 8.0513e
-04
Epoch 16/50
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 25ms/step - loss: 0.0066 - val_loss: 6.6437e
-04
Epoch 17/50
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 24ms/step - loss: 0.0062 - val_loss: 7.6041e
-04
Epoch 18/50
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 28ms/step - loss: 0.0060 - val_loss: 0.0017
Epoch 19/50
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 29ms/step - loss: 0.0062 - val_loss: 0.0011
Epoch 20/50
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 29ms/step - loss: 0.0058 - val_loss: 7.3722e
-04
Epoch 21/50
44/44 ━━━━━━━━━━━━━━━━━━━━ 1s 28ms/step - loss: 0.0059 - val_loss: 6.7478e
-04
✅ Saved model and scalers: lstm_tplus1_stacked
48/48 ━━━━━━━━━━━━━━━━━━━━ 1s 14ms/step
📊 Train R²: 0.9841, RMSE: 340.30, MAE: 267.62
```




```
🧠 Training LSTM (Stacked) model for horizon t+3
Epoch 1/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 3s 35ms/step - loss: 0.1797 - val_loss: 0.1047
Epoch 2/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 29ms/step - loss: 0.0346 - val_loss: 0.0419
Epoch 3/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 29ms/step - loss: 0.0149 - val_loss: 0.0086
Epoch 4/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 29ms/step - loss: 0.0112 - val_loss: 0.0038
Epoch 5/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 29ms/step - loss: 0.0103 - val_loss: 0.0055
Epoch 6/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 30ms/step - loss: 0.0090 - val_loss: 0.0051
Epoch 7/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 29ms/step - loss: 0.0093 - val_loss: 0.0034
Epoch 8/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 29ms/step - loss: 0.0081 - val_loss: 0.0014
Epoch 9/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 29ms/step - loss: 0.0072 - val_loss: 0.0023
Epoch 10/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 29ms/step - loss: 0.0073 - val_loss: 0.0015
Epoch 11/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 29ms/step - loss: 0.0070 - val_loss: 0.0029
Epoch 12/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 28ms/step - loss: 0.0072 - val_loss: 0.0015
Epoch 13/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 29ms/step - loss: 0.0056 - val_loss: 0.0015
✅ Saved model and scalers: lstm_tplus3_stacked
48/48 ━━━━━━━━━━━━━━━━━━━━ 1s 13ms/step
📊 Train R²: 0.9779, RMSE: 401.24, MAE: 314.26
```



```
🧠 Training LSTM (Stacked) model for horizon t+7
Epoch 1/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 4s 36ms/step - loss: 0.2339 - val_loss: 0.0743
Epoch 2/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 30ms/step - loss: 0.0342 - val_loss: 0.0427
Epoch 3/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 30ms/step - loss: 0.0185 - val_loss: 0.0049
Epoch 4/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 30ms/step - loss: 0.0101 - val_loss: 0.0028
Epoch 5/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 31ms/step - loss: 0.0099 - val_loss: 0.0018
Epoch 6/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 30ms/step - loss: 0.0082 - val_loss: 0.0023
Epoch 7/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 31ms/step - loss: 0.0075 - val_loss: 0.0025
Epoch 8/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 30ms/step - loss: 0.0078 - val_loss: 0.0033
Epoch 9/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 32ms/step - loss: 0.0073 - val_loss: 0.0021
Epoch 10/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - loss: 0.0060 - val_loss: 0.0017
Epoch 11/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 31ms/step - loss: 0.0060 - val_loss: 0.0016
Epoch 12/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 30ms/step - loss: 0.0059 - val_loss: 0.0026
Epoch 13/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - loss: 0.0056 - val_loss: 0.0010
Epoch 14/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 2s 35ms/step - loss: 0.0055 - val_loss: 0.0020
Epoch 15/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 32ms/step - loss: 0.0057 - val_loss: 0.0018
Epoch 16/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 1s 31ms/step - loss: 0.0055 - val_loss: 0.0014
Epoch 17/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 2s 35ms/step - loss: 0.0053 - val_loss: 0.0013
Epoch 18/50
43/43 ━━━━━━━━━━━━━━━━━━━━ 2s 34ms/step - loss: 0.0050 - val_loss: 0.0014
✅ Saved model and scalers: lstm_tplus7_stacked
48/48 ━━━━━━━━━━━━━━━━━━━━ 1s 13ms/step
📊 Train R²: 0.9725, RMSE: 446.37, MAE: 342.64
```



Temporal Convolutional Networks

In [15]:
```
df_tcn.columns
```

Out[15]:
```
Index(['Open', 'High', 'Low', 'Close', 'Volume', 'Adj Close', 'Log_Return
s',
       'Volatility_Log_10', 'cl-op', 'hi-lo', 'vader_news_sentiment',
       'FinBERT_news_sentiment', 'Smart_news_sentiment', 'news_buying_inte
nt',
       'news_selling_intent', 'news_uncertainty_intent', 'news_urgency_int
ent',
       'news_prediction_intent', 'news_fear_intent', 'news_greed_intent',
       'news_question_intent', 'news_action_intent', 'vader_reddit_sentime
nt',
       'FinBERT_reddit_sentiment', 'Smart_reddit_sentiment',
       'reddit_buying_intent', 'reddit_selling_intent',
       'reddit_uncertainty_intent', 'reddit_urgency_intent',
       'reddit_prediction_intent', 'reddit_fear_intent', 'reddit_greed_int
ent',
       'reddit_question_intent', 'reddit_action_intent', 'pct_change',
       'finbert_final_sentiment', 'total_buying_intent',
       'total_selling_intent', 'total_uncertainty_intent',
       'total_urgency_intent', 'total_prediction_intent', 'total_fear_inte
nt',
       'total_greed_intent', 'total_question_intent', 'total_action_inten
t',
       'sentiment_minus_uncertainty', 'sentiment_minus_fear',
       'sentiment_minus_action', 'sentiment_minus_urgency',
       'sentiment_minus_prediction'],
      dtype='object')
```

In [31]:
```
Dropped_tcn_cols = [
    'Open', 'High', 'Low', 'Volume', 'Adj Close',
    'Volatility_Log_10', 'cl-op', 'hi-lo', 'pct_change',
    'total_buying_intent', 'total_prediction_intent'
]
df_tcn_filtered = df_tcn.drop(columns=Dropped_tcn_cols).copy()
```

In [38]:
```
df_tcn_filtered.columns
```

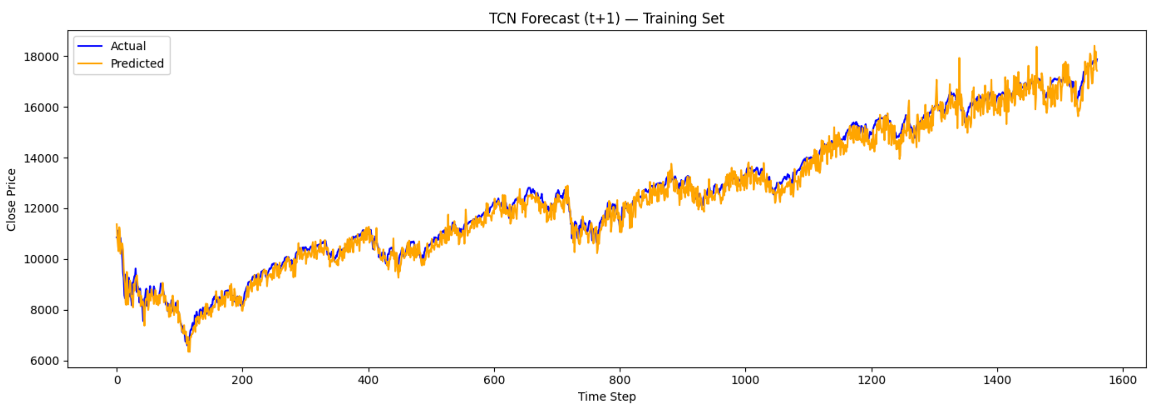
Out[38]:
```
Index(['Close', 'Log_Returns', 'vader_news_sentiment',
       'FinBERT_news_sentiment', 'Smart_news_sentiment', 'news_buying_inte
nt',
       'news_selling_intent', 'news_uncertainty_intent', 'news_urgency_int
ent',
       'news_prediction_intent', 'news_fear_intent', 'news_greed_intent',
       'news_question_intent', 'news_action_intent', 'vader_reddit_sentime
nt',
       'FinBERT_reddit_sentiment', 'Smart_reddit_sentiment',
       'reddit_buying_intent', 'reddit_selling_intent',
       'reddit_uncertainty_intent', 'reddit_urgency_intent',
       'reddit_prediction_intent', 'reddit_fear_intent', 'reddit_greed_int
ent',
       'reddit_question_intent', 'reddit_action_intent',
       'finbert_final_sentiment', 'total_selling_intent',
       'total_uncertainty_intent', 'total_urgency_intent', 'total_fear_int
ent',
       'total_greed_intent', 'total_question_intent', 'total_action_inten
t',
       'sentiment_minus_uncertainty', 'sentiment_minus_fear',
       'sentiment_minus_action', 'sentiment_minus_urgency',
       'sentiment_minus_prediction'],
      dtype='object')
```

In [34]:
```
def train_tcn_logreturn_model(df_tcn_filtered, df_targets, forecast_horizon
=1, window_size=30, epochs=50, batch_size=32):
    print(f"\n🧠 Training TCN model to predict Log Returns at t+{forecast_
horizon}")

    # Step 1: Create shifted log return target
    target_series = df_tcn_filtered['Log_Returns'].shift(-forecast_horizo
n).dropna()
    df_inputs = df_tcn_filtered.iloc[:len(target_series)]

    # Step 2: Scale features and target
    X_scaler = MinMaxScaler()
    y_scaler = MinMaxScaler()
    X_scaled = X_scaler.fit_transform(df_inputs)
    y_scaled = y_scaler.fit_transform(target_series.values.reshape(-1, 1))

    # Step 3: Create sequences
    X_seq, y_seq = [], []
    for i in range(window_size, len(X_scaled)):
        X_seq.append(X_scaled[i - window_size:i])
        y_seq.append(y_scaled[i])
    X_seq, y_seq = np.array(X_seq), np.array(y_seq)
    X_train, y_train = X_seq, y_seq

    # Step 4: Build and compile model
    model = Sequential()
    model.add(TCN(input_shape=(X_train.shape[1], X_train.shape[2]), nb_filt
ers=32, kernel_size=2, dropout_rate=0.1))
    model.add(Dropout(0.2))
    model.add(Dense(1))
    model.compile(optimizer=Adam(learning_rate=1e-4), loss='mse')
    early_stop = EarlyStopping(monitor='val_loss', patience=5, restore_best
_weights=True)

    # Step 5: Fit model
    print("🚀 Starting model.fit()")
    model.fit(
        X_train, y_train,
        epochs=epochs,
        batch_size=batch_size,
        validation_split=0.1,
        callbacks=[early_stop],
        verbose=2
    )
    print("✅ model.fit() complete")

    # Step 6: Save model and scalers
    model_name = f"tcn_logret_tplus{forecast_horizon}"
    base_path = "B:/DCU/Practicum/Proj/Models"
    os.makedirs(base_path, exist_ok=True)
    model.save(f"{base_path}/{model_name}.keras")
    joblib.dump(X_scaler, f"{base_path}/{model_name}_scalerX.pkl")
    joblib.dump(y_scaler, f"{base_path}/{model_name}_scalerY.pkl")
    print(f"✅ Saved model and scalers: {model_name}")

    # Step 7: Predict and reconstruct Close price
    y_pred_scaled = model.predict(X_train)
    y_pred_log = y_scaler.inverse_transform(y_pred_scaled).flatten()

    # Reconstruct Close from predicted log returns using Close_t
```

```
    close_t = df_targets['Close'].iloc[window_size - 1 : len(y_pred_log) +
window_size - 1].values
    y_pred_close = close_t * np.exp(y_pred_log)

    # Get ACTUAL Close at t+h (NOT reconstructed)
    y_true_close = df_targets['Close'].shift(-forecast_horizon).dropna().il
oc[window_size:].values

    # Step 8: Evaluate performance
    r2 = r2_score(y_true_close, y_pred_close)
    rmse = np.sqrt(mean_squared_error(y_true_close, y_pred_close))
    mae = mean_absolute_error(y_true_close, y_pred_close)
    print(f"📊 TCN Train R²: {r2:.4f}, RMSE: {rmse:.2f}, MAE: {mae:.2f}")

    # Step 9: Save metrics
    metrics_path = os.path.join(base_path, f"{model_name}_metrics.txt")
    with open(metrics_path, "w") as f:
        f.write(f"Forecast Horizon = t+{forecast_horizon}\n")
        f.write(f"Train R²   = {r2:.4f}\n")
        f.write(f"Train RMSE = {rmse:.2f}\n")
        f.write(f"Train MAE  = {mae:.2f}\n")
    print(f"📄 Saved training metrics to {metrics_path}")

    # Step 10: Plot and save
    plot_path = os.path.join(base_path, f"{model_name}_trainplot.png")
    plt.figure(figsize=(14, 5))
    plt.plot(y_true_close, label='Actual', color='blue')
    plt.plot(y_pred_close, label='Predicted', color='orange')
    plt.title(f"TCN Forecast (t+{forecast_horizon}) — Training Set")
    plt.xlabel("Time Step")
    plt.ylabel("Close Price")
    plt.legend()
    plt.tight_layout()
    plt.savefig(plot_path)
    plt.show()
    print(f"🖼️ Saved training plot to {plot_path}")

    return model, {"r2": r2, "rmse": rmse, "mae": mae}


# 🔁 Train for multiple horizons
for h in [1, 3, 7]:
    train_tcn_logreturn_model(df_tcn_filtered, df_targets, forecast_horizon
=h)
```

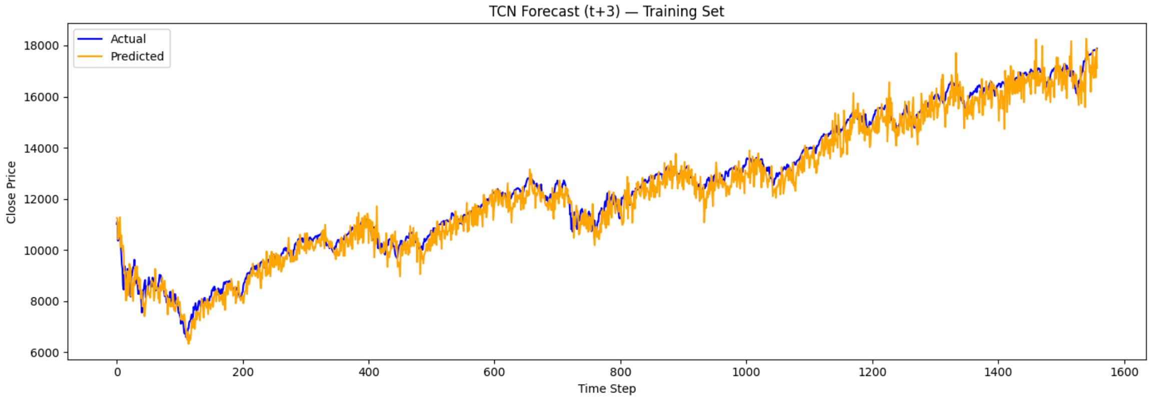
```
🧠 Training TCN model to predict Log Returns at t+1
🚀 Starting model.fit()
Epoch 1/50
44/44 - 6s - 140ms/step - loss: 8.8064 - val_loss: 0.2651
Epoch 2/50
44/44 - 1s - 14ms/step - loss: 4.0031 - val_loss: 0.2055
Epoch 3/50
44/44 - 1s - 13ms/step - loss: 2.5692 - val_loss: 0.1727
Epoch 4/50
44/44 - 1s - 15ms/step - loss: 1.9341 - val_loss: 0.1333
Epoch 5/50
44/44 - 1s - 15ms/step - loss: 1.3157 - val_loss: 0.1148
Epoch 6/50
44/44 - 1s - 15ms/step - loss: 1.2735 - val_loss: 0.0997
Epoch 7/50
44/44 - 1s - 13ms/step - loss: 0.9225 - val_loss: 0.0890
Epoch 8/50
44/44 - 1s - 12ms/step - loss: 0.7899 - val_loss: 0.0815
Epoch 9/50
44/44 - 0s - 11ms/step - loss: 0.6918 - val_loss: 0.0749
Epoch 10/50
44/44 - 0s - 11ms/step - loss: 0.5817 - val_loss: 0.0713
Epoch 11/50
44/44 - 0s - 11ms/step - loss: 0.5434 - val_loss: 0.0670
Epoch 12/50
44/44 - 0s - 11ms/step - loss: 0.5188 - val_loss: 0.0633
Epoch 13/50
44/44 - 0s - 11ms/step - loss: 0.4487 - val_loss: 0.0597
Epoch 14/50
44/44 - 0s - 11ms/step - loss: 0.4320 - val_loss: 0.0564
Epoch 15/50
44/44 - 1s - 11ms/step - loss: 0.3470 - val_loss: 0.0536
Epoch 16/50
44/44 - 1s - 11ms/step - loss: 0.3270 - val_loss: 0.0510
Epoch 17/50
44/44 - 0s - 11ms/step - loss: 0.3078 - val_loss: 0.0488
Epoch 18/50
44/44 - 1s - 12ms/step - loss: 0.2851 - val_loss: 0.0458
Epoch 19/50
44/44 - 0s - 11ms/step - loss: 0.2756 - val_loss: 0.0441
Epoch 20/50
44/44 - 0s - 11ms/step - loss: 0.2455 - val_loss: 0.0424
Epoch 21/50
44/44 - 0s - 11ms/step - loss: 0.2376 - val_loss: 0.0414
Epoch 22/50
44/44 - 0s - 11ms/step - loss: 0.2069 - val_loss: 0.0399
Epoch 23/50
44/44 - 0s - 11ms/step - loss: 0.2066 - val_loss: 0.0387
Epoch 24/50
44/44 - 0s - 11ms/step - loss: 0.1841 - val_loss: 0.0377
Epoch 25/50
44/44 - 0s - 11ms/step - loss: 0.1791 - val_loss: 0.0363
Epoch 26/50
44/44 - 0s - 11ms/step - loss: 0.1746 - val_loss: 0.0351
Epoch 27/50
44/44 - 0s - 11ms/step - loss: 0.1562 - val_loss: 0.0337
Epoch 28/50
44/44 - 0s - 11ms/step - loss: 0.1478 - val_loss: 0.0330
Epoch 29/50
44/44 - 0s - 11ms/step - loss: 0.1444 - val_loss: 0.0323
Epoch 30/50
```

```
44/44 - 0s - 11ms/step - loss: 0.1396 - val_loss: 0.0313
Epoch 31/50
44/44 - 0s - 11ms/step - loss: 0.1182 - val_loss: 0.0303
Epoch 32/50
44/44 - 0s - 11ms/step - loss: 0.1169 - val_loss: 0.0295
Epoch 33/50
44/44 - 1s - 12ms/step - loss: 0.1167 - val_loss: 0.0289
Epoch 34/50
44/44 - 0s - 11ms/step - loss: 0.1160 - val_loss: 0.0282
Epoch 35/50
44/44 - 0s - 11ms/step - loss: 0.1125 - val_loss: 0.0273
Epoch 36/50
44/44 - 0s - 11ms/step - loss: 0.1083 - val_loss: 0.0264
Epoch 37/50
44/44 - 0s - 11ms/step - loss: 0.0973 - val_loss: 0.0257
Epoch 38/50
44/44 - 0s - 11ms/step - loss: 0.0967 - val_loss: 0.0251
Epoch 39/50
44/44 - 0s - 11ms/step - loss: 0.0892 - val_loss: 0.0245
Epoch 40/50
44/44 - 0s - 11ms/step - loss: 0.0921 - val_loss: 0.0239
Epoch 41/50
44/44 - 0s - 11ms/step - loss: 0.0896 - val_loss: 0.0235
Epoch 42/50
44/44 - 0s - 11ms/step - loss: 0.0797 - val_loss: 0.0228
Epoch 43/50
44/44 - 0s - 11ms/step - loss: 0.0848 - val_loss: 0.0224
Epoch 44/50
44/44 - 0s - 11ms/step - loss: 0.0813 - val_loss: 0.0217
Epoch 45/50
44/44 - 0s - 11ms/step - loss: 0.0774 - val_loss: 0.0213
Epoch 46/50
44/44 - 0s - 11ms/step - loss: 0.0768 - val_loss: 0.0209
Epoch 47/50
44/44 - 0s - 11ms/step - loss: 0.0699 - val_loss: 0.0201
Epoch 48/50
44/44 - 0s - 11ms/step - loss: 0.0734 - val_loss: 0.0199
Epoch 49/50
44/44 - 0s - 11ms/step - loss: 0.0693 - val_loss: 0.0198
Epoch 50/50
44/44 - 0s - 11ms/step - loss: 0.0668 - val_loss: 0.0192
✅ model.fit() complete
✅ Saved model and scalers: tcn_logret_tplus1
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step
📊 TCN Train R²: 0.9808, RMSE: 374.82, MAE: 304.05
📄 Saved training metrics to B:/DCU/Practicum/Proj/Models\tcn_logret_tplus1_metrics.txt
```



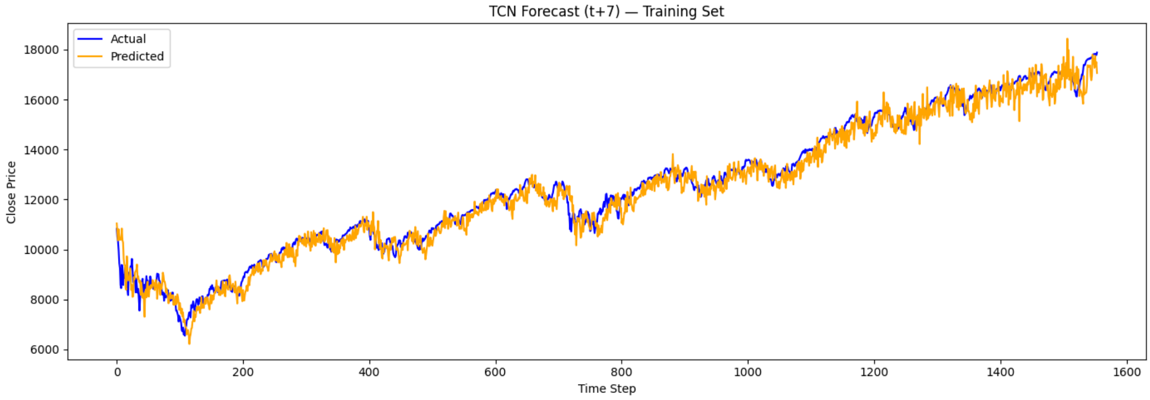
```
🖼️ Saved training plot to B:/DCU/Practicum/Proj/Models\tcn_logret_tplus1_t
rainplot.png

🧠 Training TCN model to predict Log Returns at t+3
🚀 Starting model.fit()
Epoch 1/50
44/44 - 4s - 94ms/step - loss: 52.7855 - val_loss: 0.6365
Epoch 2/50
44/44 - 0s - 11ms/step - loss: 21.0135 - val_loss: 0.6757
Epoch 3/50
44/44 - 0s - 11ms/step - loss: 10.5472 - val_loss: 0.5265
Epoch 4/50
44/44 - 0s - 11ms/step - loss: 7.9254 - val_loss: 0.4337
Epoch 5/50
44/44 - 1s - 12ms/step - loss: 5.7447 - val_loss: 0.4067
Epoch 6/50
44/44 - 1s - 14ms/step - loss: 4.4373 - val_loss: 0.3269
Epoch 7/50
44/44 - 1s - 13ms/step - loss: 3.5911 - val_loss: 0.2810
Epoch 8/50
44/44 - 1s - 13ms/step - loss: 2.7978 - val_loss: 0.2756
Epoch 9/50
44/44 - 1s - 12ms/step - loss: 2.3894 - val_loss: 0.2274
Epoch 10/50
44/44 - 1s - 11ms/step - loss: 2.2915 - val_loss: 0.1910
Epoch 11/50
44/44 - 1s - 12ms/step - loss: 1.8633 - val_loss: 0.1712
Epoch 12/50
44/44 - 0s - 11ms/step - loss: 1.6861 - val_loss: 0.1485
Epoch 13/50
44/44 - 0s - 11ms/step - loss: 1.4608 - val_loss: 0.1380
Epoch 14/50
44/44 - 0s - 11ms/step - loss: 1.2667 - val_loss: 0.1301
Epoch 15/50
44/44 - 1s - 12ms/step - loss: 1.1116 - val_loss: 0.1226
Epoch 16/50
44/44 - 1s - 14ms/step - loss: 1.1271 - val_loss: 0.1126
Epoch 17/50
44/44 - 1s - 12ms/step - loss: 0.9920 - val_loss: 0.1053
Epoch 18/50
44/44 - 0s - 11ms/step - loss: 0.9505 - val_loss: 0.0994
Epoch 19/50
44/44 - 0s - 11ms/step - loss: 0.8012 - val_loss: 0.0924
Epoch 20/50
44/44 - 0s - 11ms/step - loss: 0.7182 - val_loss: 0.0866
Epoch 21/50
44/44 - 0s - 11ms/step - loss: 0.7184 - val_loss: 0.0834
Epoch 22/50
44/44 - 0s - 11ms/step - loss: 0.6213 - val_loss: 0.0777
Epoch 23/50
44/44 - 0s - 11ms/step - loss: 0.5542 - val_loss: 0.0734
Epoch 24/50
44/44 - 0s - 11ms/step - loss: 0.5886 - val_loss: 0.0679
Epoch 25/50
44/44 - 0s - 11ms/step - loss: 0.4999 - val_loss: 0.0644
Epoch 26/50
44/44 - 0s - 11ms/step - loss: 0.5220 - val_loss: 0.0607
Epoch 27/50
44/44 - 0s - 11ms/step - loss: 0.5130 - val_loss: 0.0594
Epoch 28/50
44/44 - 0s - 11ms/step - loss: 0.4381 - val_loss: 0.0562
```

```
Epoch 29/50
44/44 - 0s - 11ms/step - loss: 0.4442 - val_loss: 0.0520
Epoch 30/50
44/44 - 0s - 11ms/step - loss: 0.4139 - val_loss: 0.0483
Epoch 31/50
44/44 - 0s - 11ms/step - loss: 0.3662 - val_loss: 0.0470
Epoch 32/50
44/44 - 0s - 11ms/step - loss: 0.3645 - val_loss: 0.0483
Epoch 33/50
44/44 - 0s - 11ms/step - loss: 0.3449 - val_loss: 0.0460
Epoch 34/50
44/44 - 0s - 11ms/step - loss: 0.3271 - val_loss: 0.0417
Epoch 35/50
44/44 - 0s - 11ms/step - loss: 0.3240 - val_loss: 0.0413
Epoch 36/50
44/44 - 0s - 11ms/step - loss: 0.3033 - val_loss: 0.0408
Epoch 37/50
44/44 - 0s - 11ms/step - loss: 0.2733 - val_loss: 0.0378
Epoch 38/50
44/44 - 0s - 11ms/step - loss: 0.2894 - val_loss: 0.0366
Epoch 39/50
44/44 - 0s - 11ms/step - loss: 0.2610 - val_loss: 0.0366
Epoch 40/50
44/44 - 0s - 11ms/step - loss: 0.2378 - val_loss: 0.0368
Epoch 41/50
44/44 - 0s - 11ms/step - loss: 0.2349 - val_loss: 0.0347
Epoch 42/50
44/44 - 0s - 11ms/step - loss: 0.2533 - val_loss: 0.0336
Epoch 43/50
44/44 - 0s - 11ms/step - loss: 0.2232 - val_loss: 0.0336
Epoch 44/50
44/44 - 0s - 11ms/step - loss: 0.2270 - val_loss: 0.0325
Epoch 45/50
44/44 - 0s - 11ms/step - loss: 0.2102 - val_loss: 0.0313
Epoch 46/50
44/44 - 0s - 11ms/step - loss: 0.1960 - val_loss: 0.0313
Epoch 47/50
44/44 - 0s - 11ms/step - loss: 0.1870 - val_loss: 0.0323
Epoch 48/50
44/44 - 0s - 11ms/step - loss: 0.1968 - val_loss: 0.0306
Epoch 49/50
44/44 - 0s - 11ms/step - loss: 0.1806 - val_loss: 0.0307
Epoch 50/50
44/44 - 0s - 11ms/step - loss: 0.1683 - val_loss: 0.0317
✅ model.fit() complete
✅ Saved model and scalers: tcn_logret_tplus3
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 8ms/step
📊 TCN Train R²: 0.9690, RMSE: 476.77, MAE: 380.55
📄 Saved training metrics to B:/DCU/Practicum/Proj/Models\tcn_logret_tplus3_metrics.txt
```


TCN Forecast (t+3) — Training Set
Actual
Predicted
Close Price
Time Step
18000
16000
14000
12000
10000
8000
6000
0
200
400
600
800
1000
1200
1400
1600

```
🖼️ Saved training plot to B:/DCU/Practicum/Proj/Models\tcn_logret_tplus3_t
rainplot.png

🧠 Training TCN model to predict Log Returns at t+7
🚀 Starting model.fit()
Epoch 1/50
44/44 - 5s - 115ms/step - loss: 8.0907 - val_loss: 0.3650
Epoch 2/50
44/44 - 1s - 12ms/step - loss: 3.9813 - val_loss: 0.2259
Epoch 3/50
44/44 - 0s - 11ms/step - loss: 2.6725 - val_loss: 0.1870
Epoch 4/50
44/44 - 1s - 11ms/step - loss: 1.8606 - val_loss: 0.1652
Epoch 5/50
44/44 - 0s - 11ms/step - loss: 1.5288 - val_loss: 0.1451
Epoch 6/50
44/44 - 0s - 11ms/step - loss: 1.2023 - val_loss: 0.1346
Epoch 7/50
44/44 - 1s - 12ms/step - loss: 0.9987 - val_loss: 0.1225
Epoch 8/50
44/44 - 1s - 12ms/step - loss: 0.8570 - val_loss: 0.1143
Epoch 9/50
44/44 - 1s - 11ms/step - loss: 0.8057 - val_loss: 0.1050
Epoch 10/50
44/44 - 1s - 12ms/step - loss: 0.6900 - val_loss: 0.0956
Epoch 11/50
44/44 - 1s - 12ms/step - loss: 0.5807 - val_loss: 0.0919
Epoch 12/50
44/44 - 1s - 11ms/step - loss: 0.5687 - val_loss: 0.0866
Epoch 13/50
44/44 - 1s - 11ms/step - loss: 0.4707 - val_loss: 0.0807
Epoch 14/50
44/44 - 0s - 11ms/step - loss: 0.4592 - val_loss: 0.0760
Epoch 15/50
44/44 - 0s - 11ms/step - loss: 0.4062 - val_loss: 0.0717
Epoch 16/50
44/44 - 1s - 12ms/step - loss: 0.3795 - val_loss: 0.0681
Epoch 17/50
44/44 - 1s - 11ms/step - loss: 0.3457 - val_loss: 0.0645
Epoch 18/50
44/44 - 1s - 12ms/step - loss: 0.3345 - val_loss: 0.0600
Epoch 19/50
44/44 - 1s - 11ms/step - loss: 0.2806 - val_loss: 0.0561
Epoch 20/50
44/44 - 1s - 12ms/step - loss: 0.2914 - val_loss: 0.0535
Epoch 21/50
44/44 - 0s - 11ms/step - loss: 0.2597 - val_loss: 0.0514
Epoch 22/50
44/44 - 0s - 11ms/step - loss: 0.2384 - val_loss: 0.0493
Epoch 23/50
44/44 - 0s - 11ms/step - loss: 0.2398 - val_loss: 0.0462
Epoch 24/50
44/44 - 0s - 11ms/step - loss: 0.2032 - val_loss: 0.0446
Epoch 25/50
44/44 - 0s - 11ms/step - loss: 0.1985 - val_loss: 0.0429
Epoch 26/50
44/44 - 0s - 11ms/step - loss: 0.1834 - val_loss: 0.0420
Epoch 27/50
44/44 - 0s - 11ms/step - loss: 0.1855 - val_loss: 0.0410
Epoch 28/50
44/44 - 0s - 11ms/step - loss: 0.1744 - val_loss: 0.0397
```

```
Epoch 29/50
44/44 - 0s - 11ms/step - loss: 0.1763 - val_loss: 0.0382
Epoch 30/50
44/44 - 1s - 11ms/step - loss: 0.1562 - val_loss: 0.0361
Epoch 31/50
44/44 - 0s - 11ms/step - loss: 0.1594 - val_loss: 0.0342
Epoch 32/50
44/44 - 1s - 11ms/step - loss: 0.1408 - val_loss: 0.0338
Epoch 33/50
44/44 - 0s - 11ms/step - loss: 0.1354 - val_loss: 0.0328
Epoch 34/50
44/44 - 0s - 11ms/step - loss: 0.1378 - val_loss: 0.0321
Epoch 35/50
44/44 - 1s - 12ms/step - loss: 0.1210 - val_loss: 0.0305
Epoch 36/50
44/44 - 1s - 12ms/step - loss: 0.1115 - val_loss: 0.0297
Epoch 37/50
44/44 - 0s - 11ms/step - loss: 0.1092 - val_loss: 0.0301
Epoch 38/50
44/44 - 1s - 12ms/step - loss: 0.0995 - val_loss: 0.0295
Epoch 39/50
44/44 - 1s - 12ms/step - loss: 0.0963 - val_loss: 0.0283
Epoch 40/50
44/44 - 1s - 11ms/step - loss: 0.0999 - val_loss: 0.0272
Epoch 41/50
44/44 - 1s - 12ms/step - loss: 0.0995 - val_loss: 0.0268
Epoch 42/50
44/44 - 0s - 11ms/step - loss: 0.0965 - val_loss: 0.0268
Epoch 43/50
44/44 - 1s - 12ms/step - loss: 0.0892 - val_loss: 0.0252
Epoch 44/50
44/44 - 1s - 11ms/step - loss: 0.0865 - val_loss: 0.0247
Epoch 45/50
44/44 - 1s - 11ms/step - loss: 0.0792 - val_loss: 0.0235
Epoch 46/50
44/44 - 0s - 11ms/step - loss: 0.0780 - val_loss: 0.0235
Epoch 47/50
44/44 - 0s - 11ms/step - loss: 0.0780 - val_loss: 0.0221
Epoch 48/50
44/44 - 1s - 12ms/step - loss: 0.0783 - val_loss: 0.0222
Epoch 49/50
44/44 - 1s - 12ms/step - loss: 0.0758 - val_loss: 0.0216
Epoch 50/50
44/44 - 1s - 12ms/step - loss: 0.0666 - val_loss: 0.0207
✅ model.fit() complete
✅ Saved model and scalers: tcn_logret_tplus7
49/49 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step
📊 TCN Train R²: 0.9729, RMSE: 446.05, MAE: 350.91
📄 Saved training metrics to B:/DCU/Practicum/Proj/Models\tcn_logret_tplus
7_metrics.txt
```



🖼️ Saved training plot to B:/DCU/Practicum/Proj/Models\tcn_logret_tplus7_trainplot.png

In [36]:
```
def run_shap_kernel_on_tcn(model_dir, df_input, forecast_horizon, window_size=30, num_samples=25):
    model_name = f"tcn_tplus{forecast_horizon}"
    model_path = os.path.join(model_dir, f"{model_name}.keras")
    scalerX_path = os.path.join(model_dir, f"{model_name}_scalerX.pkl")

    # Load model and scaler
    model = load_model(model_path, compile=False)
    X_scaler = joblib.load(scalerX_path)

    # Scale and sequence input
    df_scaled = X_scaler.transform(df_input)
    X_seq = []
    for i in range(window_size, len(df_scaled)):
        X_seq.append(df_scaled[i - window_size:i])
    X_seq = np.array(X_seq)

    # Sample last N sequences
    X_sampled = X_seq[-num_samples:]

    # Track shape dimensions
    num_samples_actual = X_sampled.shape[0]
    num_timesteps = X_sampled.shape[1]
    num_features = X_sampled.shape[2]

    # Flatten input for KernelExplainer
    X_flat = X_sampled.reshape((num_samples_actual, num_timesteps * num_features))

    # Define prediction wrapper
    def predict_fn(x_flat):
        x_reshaped = x_flat.reshape((-1, num_timesteps, num_features))
        preds = model.predict(x_reshaped)
        return np.array(preds).astype(np.float64).reshape(-1, 1)

    # Run SHAP KernelExplainer
    explainer = shap.KernelExplainer(predict_fn, X_flat)
    try:
        shap_values = explainer(X_flat)
        with open(f"{model_name}_shap_dump.pkl", "wb") as f:
            joblib.dump(shap_values, f)
    except Exception as e:
        print(f"❌ SHAP failed: {e}")
        return

    shap_array = np.abs(shap_values.values)  # shape: [samples, window_size * features]
    shap_array_3d = shap_array.reshape(shap_array.shape[0], window_size, num_features)
    shap_feature_mean = np.mean(shap_array_3d, axis=(0, 1))  # average across time and samples

    feature_names = df_input.columns.tolist()

    # Rank + plot
    shap_df = list(zip(feature_names, shap_feature_mean))
    shap_df.sort(key=lambda x: x[1], reverse=True)
    sorted_features, sorted_importance = zip(*shap_df)

    plt.figure(figsize=(10, 6))
```

```
    plt.barh(sorted_features[::-1], sorted_importance[::-1], color='steelbl
ue')
    plt.title(f"SHAP Feature Importance — TCN KernelExplainer (t+{forecast_
horizon})")
    plt.tight_layout()

    plot_path = os.path.join(model_dir, f"{model_name}_shap_kernel.png")
    plt.savefig(plot_path)
    print(f"✅ SHAP saved: {plot_path}")

model_dir = "B:/DCU/Practicum/Proj/Models"

# for h in [1, 3, 7]:
#     print(f"\n🔎 Running SHAP KernelExplainer for TCN t+{h}")
#     run_shap_kernel_on_tcn(model_dir, df_tcn, forecast_horizon=h, num_sam
ples=100)

run_shap_kernel_on_tcn(model_dir, df_tcn_filtered, forecast_horizon=1, num_
samples=25)
```

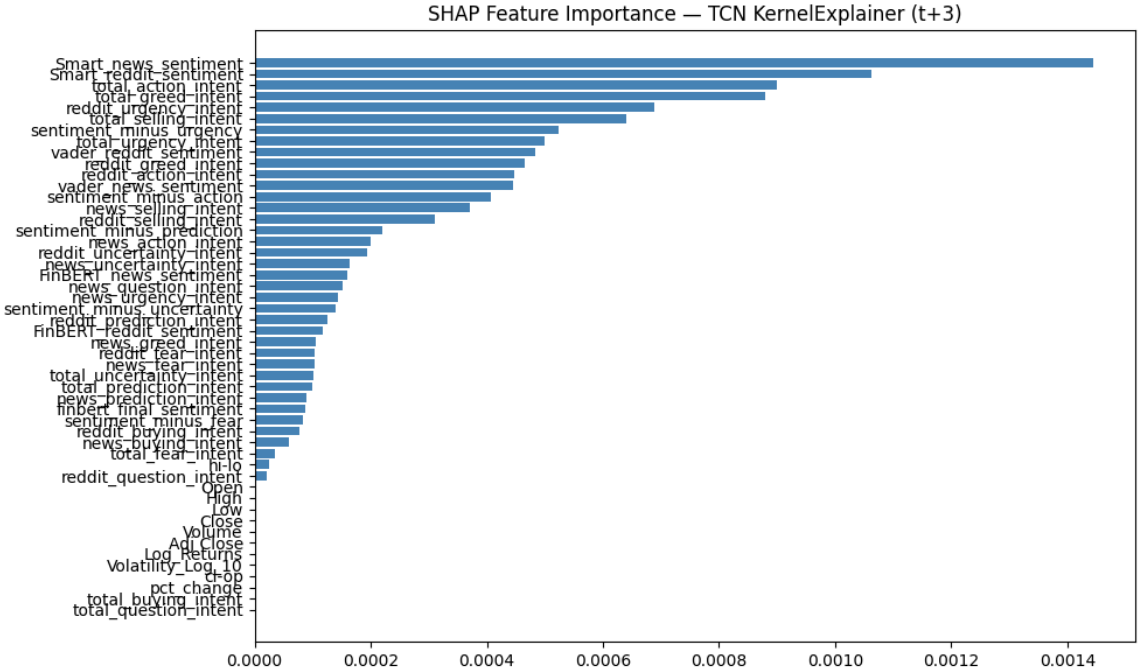
```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[36], line 70
     64 model_dir = "B:/DCU/Practicum/Proj/Models"
     66 # for h in [1, 3, 7]:
     67 #     print(f"\n🔎 Running SHAP KernelExplainer for TCN t+{h}")
     68 #     run_shap_kernel_on_tcn(model_dir, df_tcn, forecast_horizon=h, num_samples=100)
---> 70 run_shap_kernel_on_tcn(model_dir, df_tcn_filtered, forecast_horizon=1, num_samples=25)

Cell In[36], line 11, in run_shap_kernel_on_tcn(model_dir, df_input, forecast_horizon, window_size, num_samples)
      8 X_scaler = joblib.load(scalerX_path)
     10 # Scale and sequence input
---> 11 df_scaled = X_scaler.transform(df_input)
     12 X_seq = []
     13 for i in range(window_size, len(df_scaled)):

File b:\DCU\Practicum\Proj\App\venv_3_11\Lib\site-packages\sklearn\utils\_set_output.py:316, in _wrap_method_output.<locals>.wrapped(self, X, *args, **kwargs)
    314 @wraps(f)
    315 def wrapped(self, X, *args, **kwargs):
--> 316     data_to_wrap = f(self, X, *args, **kwargs)
    317     if isinstance(data_to_wrap, tuple):
    318         # only wrap the first output for cross decomposition
    319         return_tuple = (
    320             _wrap_data_with_container(method, data_to_wrap[0], X, self),
    321             *data_to_wrap[1:],
    322         )

File b:\DCU\Practicum\Proj\App\venv_3_11\Lib\site-packages\sklearn\preprocessing\_data.py:545, in MinMaxScaler.transform(self, X)
    541 check_is_fitted(self)
    543 xp, _ = get_namespace(X)
--> 545 X = validate_data(
    546     self,
    547     X,
    548     copy=self.copy,
    549     dtype=_array_api.supported_float_dtypes(xp),
    550     force_writeable=True,
    551     ensure_all_finite="allow-nan",
    552     reset=False,
    553 )
    555 X *= self.scale_
    556 X += self.min_

File b:\DCU\Practicum\Proj\App\venv_3_11\Lib\site-packages\sklearn\utils\validation.py:2929, in validate_data(_estimator, X, y, reset, validate_separately, skip_check_array, **check_params)
   2845 def validate_data(
   2846     _estimator,
   2847     /,
   (...)   2853     **check_params,
   2854 ):
   2855     """Validate input data and set or check feature names and counts of the input.
```

```
   2856
   2857     This helper function should be used in an estimator that requi
res input
   (...)   2927         validated.
   2928     """
-> 2929     _check_feature_names(_estimator, X, reset=reset)
   2930     tags = get_tags(_estimator)
   2931     if y is None and tags.target_tags.required:

File b:\DCU\Practicum\Proj\App\venv_3_11\Lib\site-packages\sklearn\utils\v
alidation.py:2787, in _check_feature_names(estimator, X, reset)
   2784 if not missing_names and not unexpected_names:
   2785     message += "Feature names must be in the same order as they we
re in fit.\n"
-> 2787 raise ValueError(message)

ValueError: The feature names should match those that were passed during f
it.
Feature names unseen at fit time:
- Log_Returns
```

In [24]:
```
run_shap_kernel_on_tcn(model_dir, df_tcn, forecast_horizon=3, num_samples=2
5)
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 319ms/step

  0%|          | 0/25 [00:00<?, ?it/s]

1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 303ms/step
3944/3944 ━━━━━━━━━━━━━━━━━━━━ 15s 4ms/step

  4%|▍         | 1/25 [00:17<07:01, 17.55s/it]

1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
3944/3944 ━━━━━━━━━━━━━━━━━━━━ 13s 3ms/step

  8%|▊         | 2/25 [00:33<06:23, 16.67s/it]

1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
3944/3944 ━━━━━━━━━━━━━━━━━━━━ 14s 3ms/step

 12%|█▏        | 3/25 [00:49<06:03, 16.52s/it]

1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step
3944/3944 ━━━━━━━━━━━━━━━━━━━━ 14s 3ms/step

 16%|█▌        | 4/25 [01:06<05:44, 16.39s/it]

1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step
3944/3944 ━━━━━━━━━━━━━━━━━━━━ 14s 3ms/step

 20%|██        | 5/25 [01:22<05:26, 16.33s/it]

1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
3944/3944 ━━━━━━━━━━━━━━━━━━━━ 14s 3ms/step

 24%|██▍       | 6/25 [01:38<05:10, 16.34s/it]

1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step
3944/3944 ━━━━━━━━━━━━━━━━━━━━ 14s 3ms/step

 28%|██▊       | 7/25 [01:55<04:54, 16.37s/it]

1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step
3944/3944 ━━━━━━━━━━━━━━━━━━━━ 14s 4ms/step

 32%|███▏      | 8/25 [02:11<04:39, 16.44s/it]

1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
3944/3944 ━━━━━━━━━━━━━━━━━━━━ 14s 4ms/step

 36%|███▌      | 9/25 [02:28<04:24, 16.54s/it]

1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step
3944/3944 ━━━━━━━━━━━━━━━━━━━━ 14s 4ms/step

 40%|████      | 10/25 [02:45<04:10, 16.70s/it]

1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step
3944/3944 ━━━━━━━━━━━━━━━━━━━━ 15s 4ms/step

 44%|████▍     | 11/25 [03:03<03:57, 16.94s/it]

1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
3944/3944 ━━━━━━━━━━━━━━━━━━━━ 15s 4ms/step

 48%|████▊     | 12/25 [03:20<03:42, 17.14s/it]

1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step
3944/3944 ━━━━━━━━━━━━━━━━━━━━ 15s 4ms/step
```

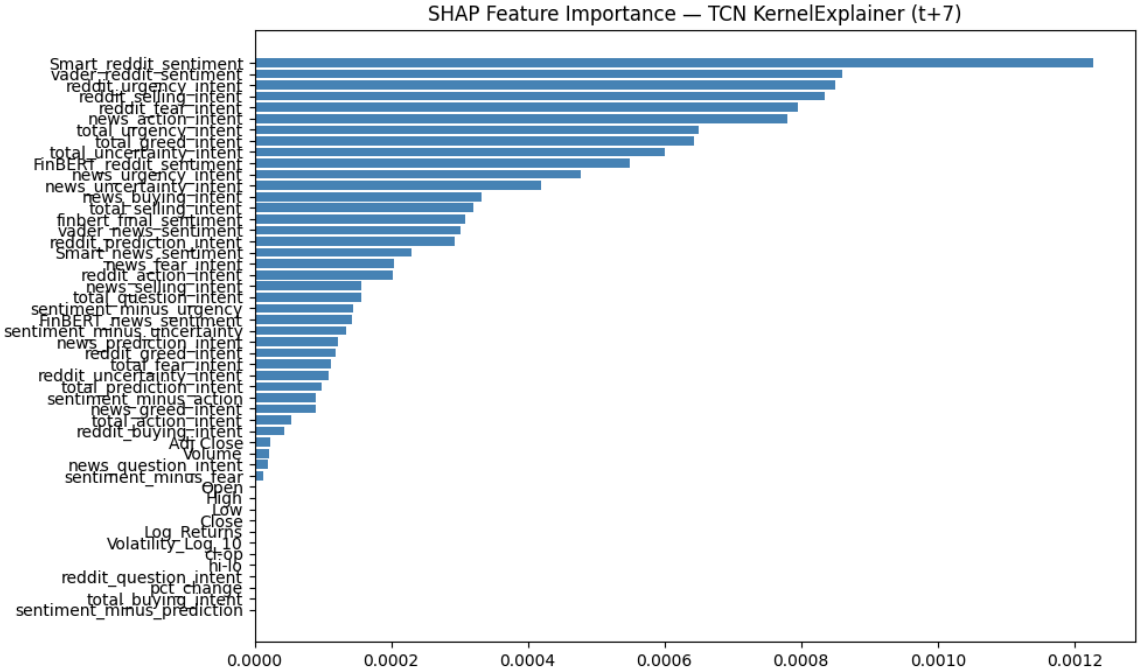
```
 52%|█████     | 13/25 [03:38<03:27, 17.27s/it]
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step
3944/3944 ━━━━━━━━━━━━━━━━━━━━ 15s 4ms/step
 56%|█████▌    | 14/25 [03:55<03:10, 17.35s/it]
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 42ms/step
3944/3944 ━━━━━━━━━━━━━━━━━━━━ 16s 4ms/step
 60%|██████    | 15/25 [04:14<02:57, 17.74s/it]
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step
3944/3944 ━━━━━━━━━━━━━━━━━━━━ 16s 4ms/step
 64%|██████▍   | 16/25 [04:33<02:43, 18.19s/it]
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step
3944/3944 ━━━━━━━━━━━━━━━━━━━━ 17s 4ms/step
 68%|██████▊   | 17/25 [04:53<02:29, 18.69s/it]
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step
3944/3944 ━━━━━━━━━━━━━━━━━━━━ 17s 4ms/step
 72%|███████▏  | 18/25 [05:13<02:14, 19.17s/it]
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step
3944/3944 ━━━━━━━━━━━━━━━━━━━━ 17s 4ms/step
 76%|███████▌  | 19/25 [05:33<01:55, 19.26s/it]
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step
3944/3944 ━━━━━━━━━━━━━━━━━━━━ 17s 4ms/step
 80%|████████  | 20/25 [05:52<01:36, 19.40s/it]
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step
3944/3944 ━━━━━━━━━━━━━━━━━━━━ 17s 4ms/step
 84%|████████▍ | 21/25 [06:12<01:17, 19.50s/it]
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step
3944/3944 ━━━━━━━━━━━━━━━━━━━━ 17s 4ms/step
 88%|████████▊ | 22/25 [06:32<00:58, 19.57s/it]
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step
3944/3944 ━━━━━━━━━━━━━━━━━━━━ 17s 4ms/step
 92%|█████████▏| 23/25 [06:52<00:39, 19.76s/it]
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step
3944/3944 ━━━━━━━━━━━━━━━━━━━━ 17s 4ms/step
 96%|█████████▌| 24/25 [07:12<00:19, 19.79s/it]
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 38ms/step
3944/3944 ━━━━━━━━━━━━━━━━━━━━ 17s 4ms/step
100%|██████████| 25/25 [07:32<00:00, 18.09s/it]
✅ SHAP saved: B:/DCU/Practicum/Proj/Models\tcn_tplus3_shap_kernel.png
```


SHAP Feature Importance — TCN KernelExplainer (t+3)
Smart_news_sentiment
Smart_reddit_sentiment
total_action_intent
total_greed_intent
reddit_urgency_intent
total_selling_intent
sentiment_minus_urgency
total_urgency_intent
vader_reddit_sentiment
reddit_greed_intent
reddit_action_intent
vader_news_sentiment
sentiment_minus_action
news_selling_intent
reddit_selling_intent
sentiment_minus_prediction
news_action_intent
reddit_uncertainty_intent
news_uncertainty_intent
FinBERT_news_sentiment
news_question_intent
news_urgency_intent
sentiment_minus_uncertainty
reddit_prediction_intent
FinBERT_reddit_sentiment
news_greed_intent
reddit_fear_intent
news_fear_intent
total_uncertainty_intent
total_prediction_intent
news_prediction_intent
finbert_final_sentiment
sentiment_minus_fear
reddit_buying_intent
news_buying_intent
total_fear_intent
hi-lo
reddit_question_intent
Open
High
Low
Close
Volume
Adj Close
Log_Returns
Volatility_Log_10
cl-op
pct_change
total_buying_intent
total_question_intent
0.0000
0.0002
0.0004
0.0006
0.0008
0.0010
0.0012
0.0014

In [25]:
```
run_shap_kernel_on_tcn(model_dir, df_tcn, forecast_horizon=7, num_samples=2
5)
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 326ms/step
  0%|          | 0/25 [00:00<?, ?it/s]
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 343ms/step
3944/3944 ━━━━━━━━━━━━━━━━━━━━ 16s 4ms/step
  4%|▍         | 1/25 [00:19<07:44, 19.35s/it]
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step
3944/3944 ━━━━━━━━━━━━━━━━━━━━ 17s 4ms/step
  8%|▊         | 2/25 [00:39<07:29, 19.53s/it]
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 37ms/step
3944/3944 ━━━━━━━━━━━━━━━━━━━━ 16s 4ms/step
 12%|█▏        | 3/25 [00:58<07:07, 19.44s/it]
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step
3944/3944 ━━━━━━━━━━━━━━━━━━━━ 16s 4ms/step
 16%|█▌        | 4/25 [01:17<06:47, 19.41s/it]
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step
3944/3944 ━━━━━━━━━━━━━━━━━━━━ 16s 4ms/step
 20%|██        | 5/25 [01:36<06:26, 19.34s/it]
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step
3944/3944 ━━━━━━━━━━━━━━━━━━━━ 16s 4ms/step
 24%|██▍       | 6/25 [01:56<06:06, 19.30s/it]
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 37ms/step
3944/3944 ━━━━━━━━━━━━━━━━━━━━ 16s 4ms/step
 28%|██▊       | 7/25 [02:15<05:46, 19.25s/it]
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step
3944/3944 ━━━━━━━━━━━━━━━━━━━━ 16s 4ms/step
 32%|███▏      | 8/25 [02:34<05:26, 19.23s/it]
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step
3944/3944 ━━━━━━━━━━━━━━━━━━━━ 19s 5ms/step
 36%|███▌      | 9/25 [02:57<05:26, 20.39s/it]
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 43ms/step
3944/3944 ━━━━━━━━━━━━━━━━━━━━ 17s 4ms/step
 40%|████      | 10/25 [03:17<05:03, 20.24s/it]
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step
3944/3944 ━━━━━━━━━━━━━━━━━━━━ 15s 4ms/step
 44%|████▍     | 11/25 [03:35<04:35, 19.69s/it]
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
3944/3944 ━━━━━━━━━━━━━━━━━━━━ 15s 4ms/step
 48%|████▊     | 12/25 [03:53<04:10, 19.23s/it]
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step
3944/3944 ━━━━━━━━━━━━━━━━━━━━ 17s 4ms/step
```

```
 52%|█████▏    | 13/25 [04:14<03:53, 19.50s/it]
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 37ms/step
3944/3944 ━━━━━━━━━━━━━━━━━━━━ 16s 4ms/step
 56%|█████▌    | 14/25 [04:33<03:33, 19.42s/it]
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step
3944/3944 ━━━━━━━━━━━━━━━━━━━━ 16s 4ms/step
 60%|██████    | 15/25 [04:51<03:11, 19.16s/it]
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step
3944/3944 ━━━━━━━━━━━━━━━━━━━━ 16s 4ms/step
 64%|██████▍   | 16/25 [05:10<02:51, 19.02s/it]
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 39ms/step
3944/3944 ━━━━━━━━━━━━━━━━━━━━ 15s 4ms/step
 68%|██████▊   | 17/25 [05:28<02:30, 18.75s/it]
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step
3944/3944 ━━━━━━━━━━━━━━━━━━━━ 15s 4ms/step
 72%|███████▏  | 18/25 [05:46<02:09, 18.47s/it]
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step
3944/3944 ━━━━━━━━━━━━━━━━━━━━ 17s 4ms/step
 76%|███████▌  | 19/25 [06:06<01:53, 18.85s/it]
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step
3944/3944 ━━━━━━━━━━━━━━━━━━━━ 19s 5ms/step
 80%|████████  | 20/25 [06:28<01:38, 19.73s/it]
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step
3944/3944 ━━━━━━━━━━━━━━━━━━━━ 15s 4ms/step
 84%|████████▍ | 21/25 [06:46<01:17, 19.33s/it]
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step
3944/3944 ━━━━━━━━━━━━━━━━━━━━ 14s 4ms/step
 88%|████████▊ | 22/25 [07:03<00:55, 18.65s/it]
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
3944/3944 ━━━━━━━━━━━━━━━━━━━━ 16s 4ms/step
 92%|█████████▏| 23/25 [07:22<00:37, 18.72s/it]
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step
3944/3944 ━━━━━━━━━━━━━━━━━━━━ 15s 4ms/step
 96%|█████████▌| 24/25 [07:39<00:18, 18.37s/it]
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
3944/3944 ━━━━━━━━━━━━━━━━━━━━ 16s 4ms/step
100%|██████████| 25/25 [07:58<00:00, 19.13s/it]
✅ SHAP saved: B:/DCU/Practicum/Proj/Models\tcn_tplus7_shap_kernel.png
```


SHAP Feature Importance — TCN KernelExplainer (t+7)
Smart_reddit_sentiment
vader_reddit_sentiment
reddit_urgency_intent
reddit_selling_intent
reddit_fear_intent
news_action_intent
total_urgency_intent
total_greed_intent
total_uncertainty_intent
FinBERT_reddit_sentiment
news_urgency_intent
news_uncertainty_intent
news_buying_intent
total_selling_intent
finbert_final_sentiment
vader_news_sentiment
reddit_prediction_intent
Smart_news_sentiment
news_fear_intent
reddit_action_intent
news_selling_intent
total_question_intent
sentiment_minus_urgency
FinBERT_news_sentiment
sentiment_minus_uncertainty
news_prediction_intent
reddit_greed_intent
total_fear_intent
reddit_uncertainty_intent
total_prediction_intent
sentiment_minus_action
news_greed_intent
total_action_intent
reddit_buying_intent
Adj Close
Volume
news_question_intent
sentiment_minus_fear
Open
High
Low
Close
Log_Returns
Volatility_Log_10
cl-op
hi-lo
reddit_question_intent
pct_change
total_buying_intent
sentiment_minus_prediction
0.0000
0.0002
0.0004
0.0006
0.0008
0.0010
0.0012