In [12]:
```
import numpy as np
import pandas as pd
import joblib
from tensorflow.keras.models import load_model
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_sco
re
import matplotlib.pyplot as plt
from tcn import TCN
from statsmodels.tsa.arima.model import ARIMA
import os
```

In [2]:
```
df_test = pd.read_csv("test_dataset.csv", parse_dates=["Date"])
df_test = df_test.sort_values("Date").reset_index(drop=True)
```

In [3]:
```
df_train = pd.read_csv("train_dataset.csv", parse_dates=["Date"])
```

In [4]:
```
df_train.drop(columns=['Next_Close', 'Next_3_Close', 'Next_7_Close'], inpla
ce=True)
df_test.drop(columns=['Next_Close', 'Next_3_Close', 'Next_7_Close'], inplac
e=True)
```

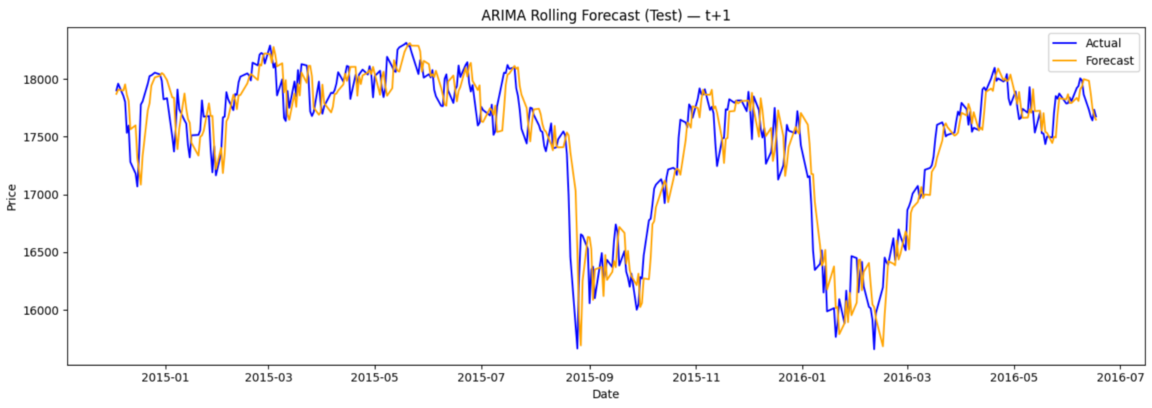
In [5]:
```
df_test = df_test.iloc[:-10]
```

In [6]:
```
df_test.shape
```

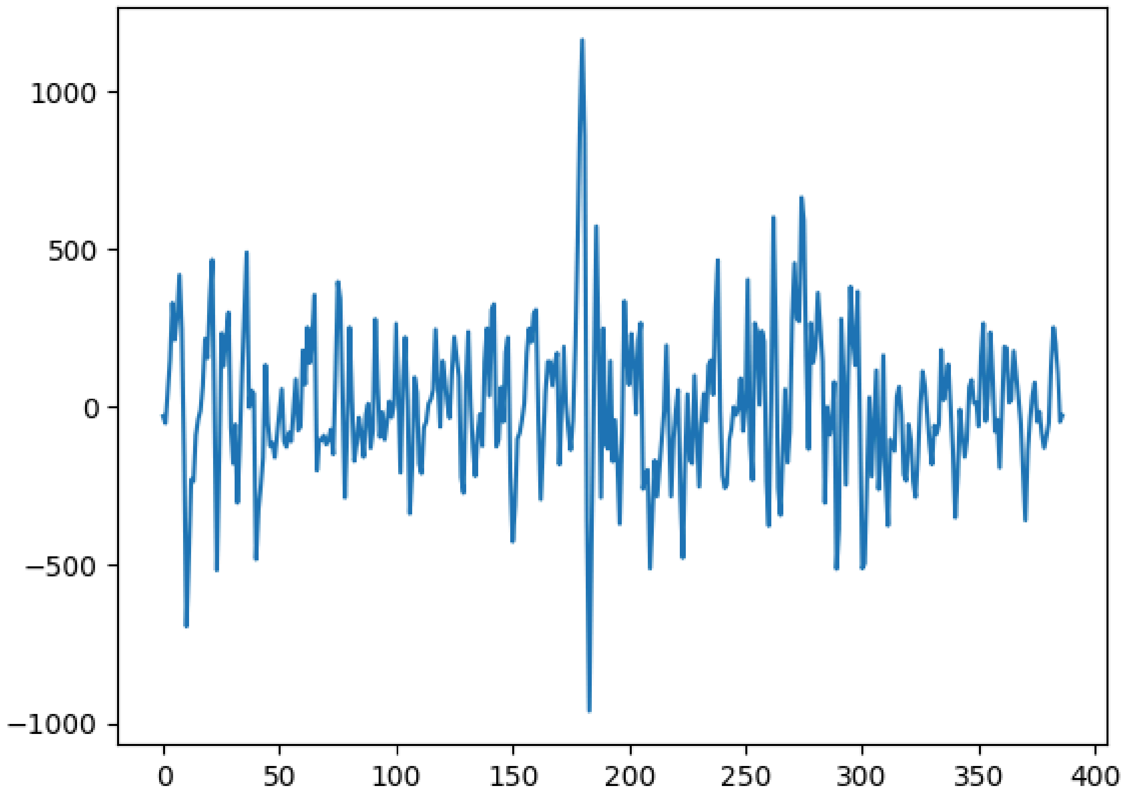
Out[6]:
```
(388, 53)
```

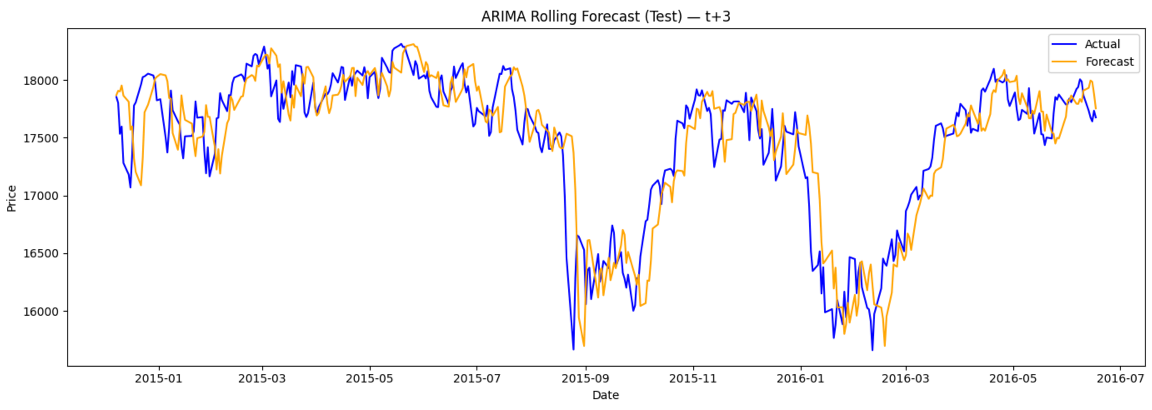
In [35]:
```
def arima_true_rolling_test(df_train, df_test, forecast_horizon=1, order=
(1, 1, 1)):
    print(f"\n🧪 Rolling Forecast ARIMA TEST (t+{forecast_horizon}) with o
rder {order}")

    close_series = df_train['Close'].tolist() + df_test['Close'].tolist()
    start_idx = len(df_train)
    history = close_series[:start_idx]  # Only train data initially

    y_true = []
    y_pred = []

    for t in range(len(df_test) - 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[start_idx + t + forecast_horizon])
        except:
            y_pred.append(np.nan)
            y_true.append(np.nan)

        history.append(close_series[start_idx + t])  # simulate real-time u
pdate

    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]

    date_series = df_test['Date'].iloc[forecast_horizon: len(y_true) + fore
cast_horizon].values

    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 Test (t+{forecast_horizon}): R² = {r2:.4f}, RMSE = {r
mse:.2f}, MAE = {mae:.2f}")

    plt.figure(figsize=(14, 5))
    plt.plot(date_series, y_true, label='Actual', color='blue')
    plt.plot(date_series, y_pred, label='Forecast', color='orange')
    plt.title(f"ARIMA Rolling Forecast (Test) — t+{forecast_horizon}")
    plt.xlabel("Date")
    plt.ylabel("Price")
    plt.legend()
    plt.tight_layout()
    plt.show()

    plt.plot(np.array(y_pred) - np.array(y_true), label='Prediction Error
(t+h - forecast)')

    return {"horizon": forecast_horizon, "R2": r2, "RMSE": rmse, "MAE": ma
e}

test_results = []
for horizon in [1, 3, 7]:
    res = arima_true_rolling_test(df_train, df_test, forecast_horizon=horiz
on, order=(1, 1, 1))
    test_results.append(res)
```

```
🧪 Rolling Forecast ARIMA TEST (t+1) with order (1, 1, 1)
📊 ARIMA Test (t+1): R² = 0.8655, RMSE = 228.79, MAE = 170.45
```

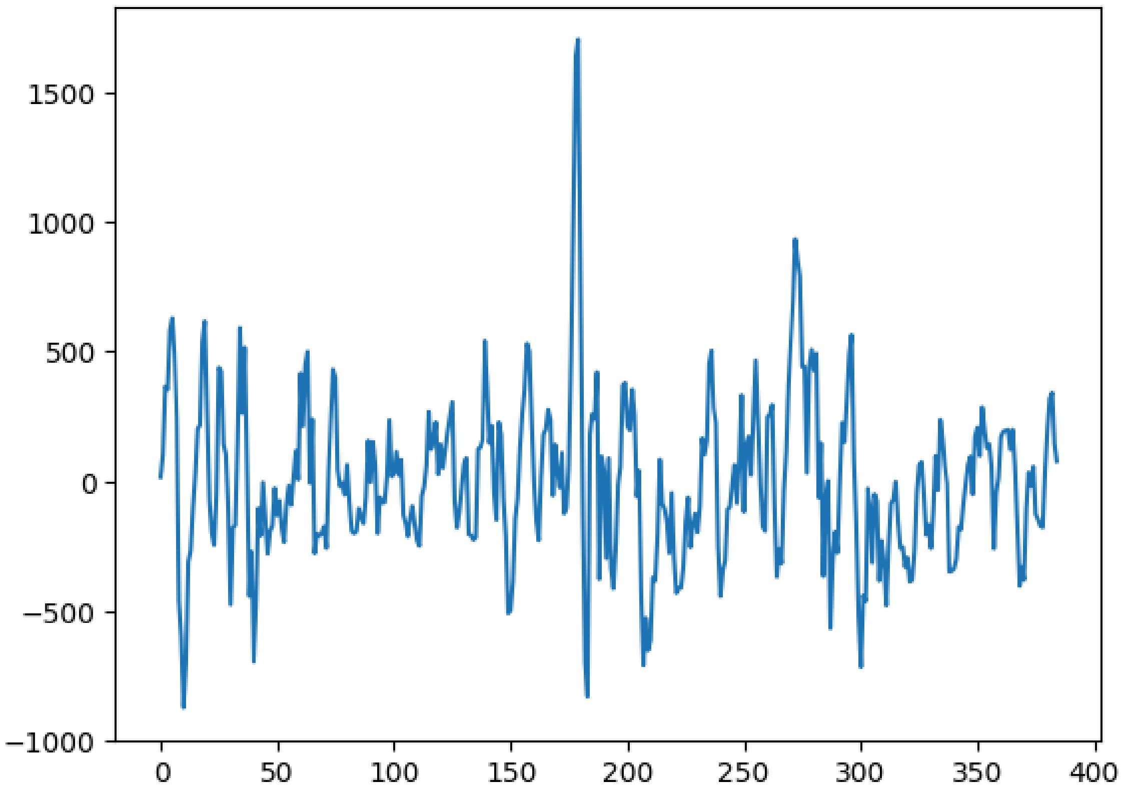


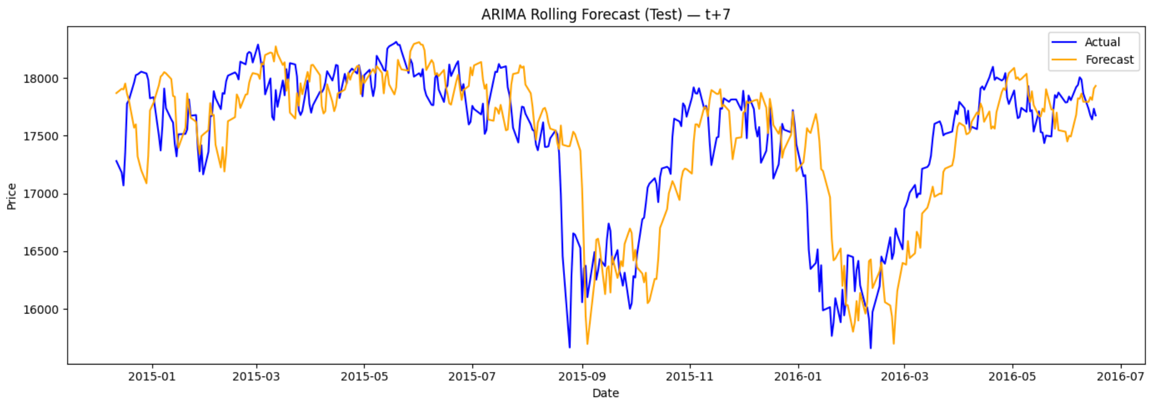
```
🧪 Rolling Forecast ARIMA TEST (t+3) with order (1, 1, 1)
📊 ARIMA Test (t+3): R² = 0.7433, RMSE = 316.38, MAE = 236.20
```

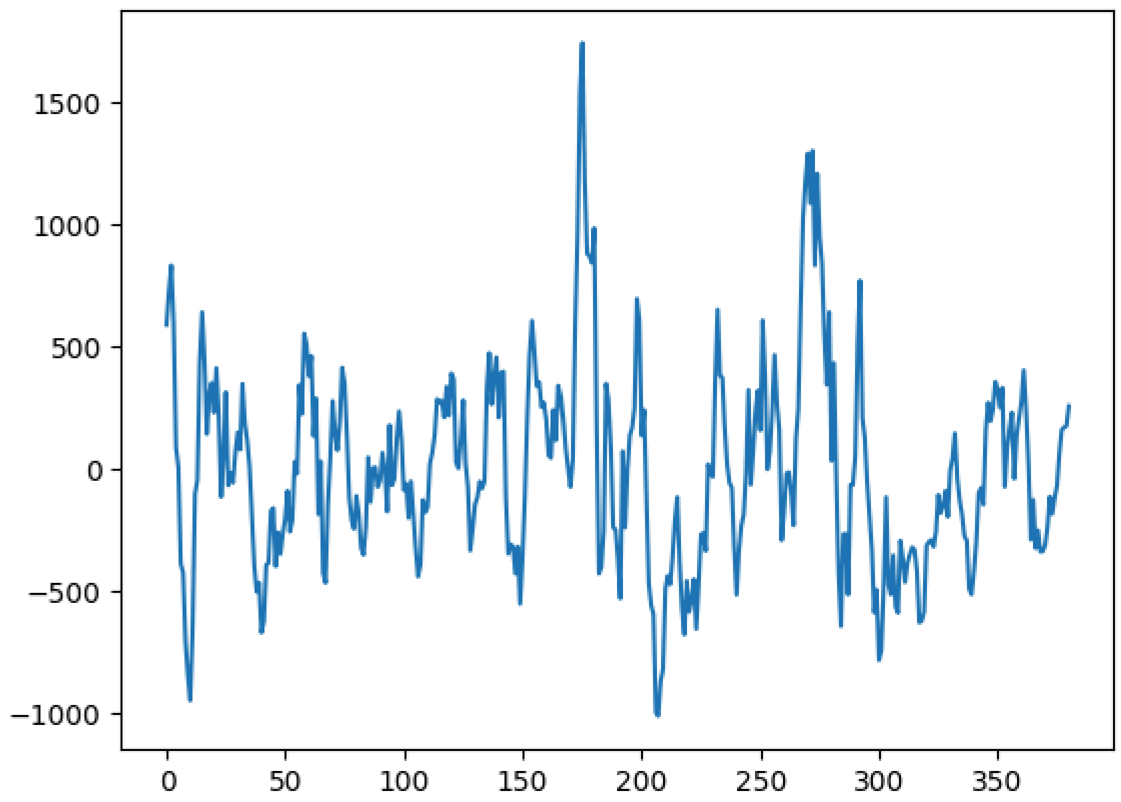




```
🧪 Rolling Forecast ARIMA TEST (t+7) with order (1, 1, 1)
📊 ARIMA Test (t+7): R² = 0.5662, RMSE = 413.11, MAE = 317.17
```


1500
1000
500
0
−500
−1000
0
50
100
150
200
250
300
350
400


ARIMA Rolling Forecast (Test) — t+7
Actual
Forecast
Price
18000
17500
17000
16500
16000
2015-01
2015-03
2015-05
2015-07
2015-09
2015-11
2016-01
2016-03
2016-05
2016-07
Date


1500
1000
500
0
−500
−1000
0
50
100
150
200
250
300
350

In [8]:
```
df_test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 388 entries, 0 to 387
Data columns (total 53 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   Date                         388 non-null    datetime64[ns]
 1   Open                         388 non-null    float64
 2   High                         388 non-null    float64
 3   Low                          388 non-null    float64
 4   Close                        388 non-null    float64
 5   Volume                       388 non-null    int64
 6   Adj Close                    388 non-null    float64
 7   Log_Returns                  388 non-null    float64
 8   Volatility_Log_10            388 non-null    float64
 9   cl-op                        388 non-null    float64
 10  hi-lo                        388 non-null    float64
 11  Label                        388 non-null    int64
 12  vader_news_sentiment         388 non-null    float64
 13  FinBERT_news_sentiment       388 non-null    float64
 14  Smart_news_sentiment         388 non-null    float64
 15  news_buying_intent           388 non-null    float64
 16  news_selling_intent          388 non-null    float64
 17  news_uncertainty_intent      388 non-null    float64
 18  news_urgency_intent          388 non-null    float64
 19  news_prediction_intent       388 non-null    float64
 20  news_fear_intent             388 non-null    float64
 21  news_greed_intent            388 non-null    float64
 22  news_question_intent         388 non-null    float64
 23  news_action_intent           388 non-null    float64
 24  vader_reddit_sentiment       388 non-null    float64
 25  FinBERT_reddit_sentiment     388 non-null    float64
 26  Smart_reddit_sentiment       388 non-null    float64
 27  reddit_buying_intent         388 non-null    float64
 28  reddit_selling_intent        388 non-null    float64
 29  reddit_uncertainty_intent    388 non-null    float64
 30  reddit_urgency_intent        388 non-null    float64
 31  reddit_prediction_intent     388 non-null    float64
 32  reddit_fear_intent           388 non-null    float64
 33  reddit_greed_intent          388 non-null    float64
 34  reddit_question_intent       388 non-null    float64
 35  reddit_action_intent         388 non-null    float64
 36  Target                       388 non-null    int64
 37  pct_change                   388 non-null    float64
 38  finbert_final_sentiment      388 non-null    float64
 39  total_buying_intent          388 non-null    float64
 40  total_selling_intent         388 non-null    float64
 41  total_uncertainty_intent     388 non-null    float64
 42  total_urgency_intent         388 non-null    float64
 43  total_prediction_intent      388 non-null    float64
 44  total_fear_intent            388 non-null    float64
 45  total_greed_intent           388 non-null    float64
 46  total_question_intent        388 non-null    float64
 47  total_action_intent          388 non-null    float64
 48  sentiment_minus_uncertainty  388 non-null    float64
 49  sentiment_minus_fear         388 non-null    float64
 50  sentiment_minus_action       388 non-null    float64
 51  sentiment_minus_urgency      388 non-null    float64
 52  sentiment_minus_prediction   388 non-null    float64
dtypes: datetime64[ns](1), float64(49), int64(3)
memory usage: 160.8 KB
```

In [9]:
```
df_test.isnull().sum()
```

Out[9]:
```
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 [10]:
```
df_test.tail()
```

Out[10]:

	Date	Open	High	Low	Close	Volume	Adj Close
383	2016-06-13	17830.500000	17893.279297	17731.349609	17732.480469	101690000	17732.48046
384	2016-06-14	17710.769531	17733.919922	17595.789062	17674.820312	93740000	17674.82031
385	2016-06-15	17703.650391	17762.960938	17629.009766	17640.169922	94130000	17640.16992
386	2016-06-16	17602.230469	17754.910156	17471.289062	17733.099609	91950000	17733.09960
387	2016-06-17	17733.439453	17733.439453	17602.779297	17675.160156	248680000	17675.16015

5 rows × 53 columns

In [20]:
```
df_test.columns
```

Out[20]:
```
Index(['Date', 'Open', 'High', 'Low', 'Close', 'Volume', 'Adj Close',
       'Log_Returns', 'Volatility_Log_10', 'cl-op', 'hi-lo', 'Label',
       'vader_news_sentiment', 'FinBERT_news_sentiment',
       'Smart_news_sentiment', 'news_buying_intent', 'news_selling_inten
t',
       'news_uncertainty_intent', 'news_urgency_intent',
       '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', 'Target',
       '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')
```

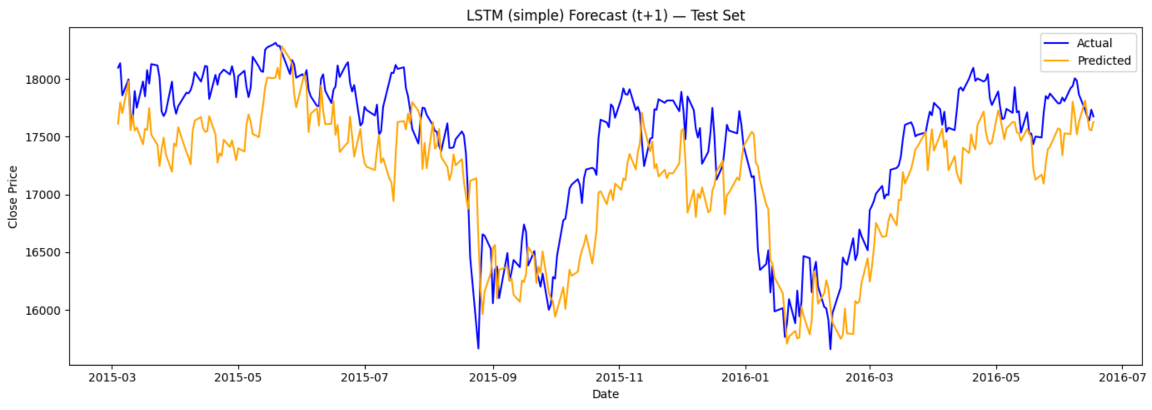
In [25]:
```
df_lstm = df_test.drop(columns=['Date', 'Label', 'Target']).copy()
```

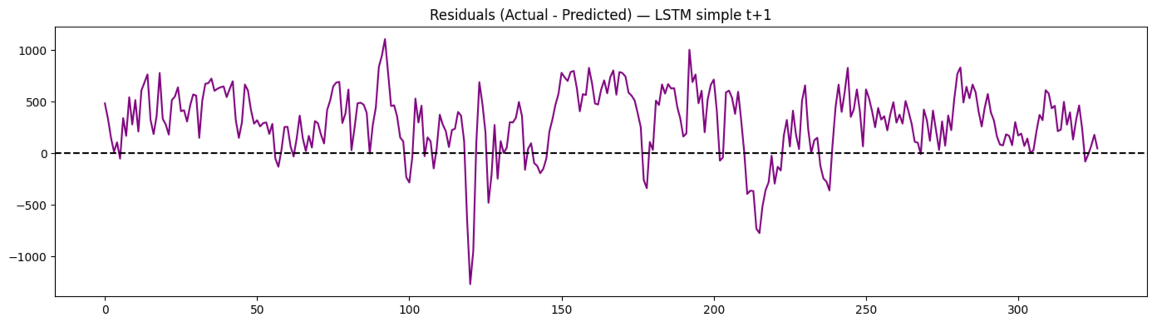
In [26]:
```
df_lstm.columns
```

Out[26]:
```
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')
```

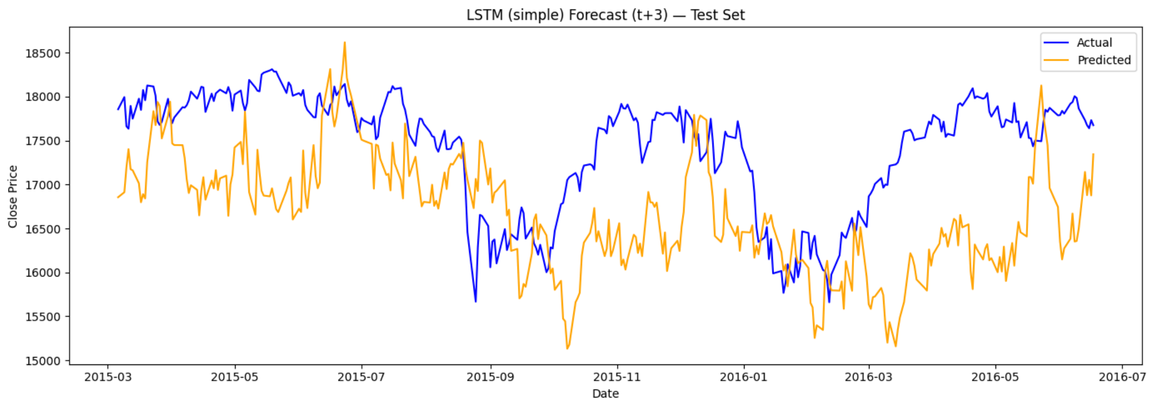
In [ ]:
```
def load_and_test_lstm(model_dir, model_type='simple', forecast_horizon=1,
df_lstm=None, window_size=60):
    assert model_type in ['simple', 'stacked'], "model_type must be 'simpl
e' or 'stacked'"

    model_name = f"lstm_tplus{forecast_horizon}_{model_type}"
    model_path = os.path.join(model_dir, f"{model_name}.keras")
    scalerX_path = os.path.join(model_dir, f"{model_name}_scalerX.pkl")
    scalerY_path = os.path.join(model_dir, f"{model_name}_scalerY.pkl")

    if not all(os.path.exists(p) for p in [model_path, scalerX_path, scaler
Y_path]):
        print(f"⚠️ Missing files for {model_name}")
        return None

    print(f"\n🔍 Testing {model_name}")

    # Load model and scalers
    model = load_model(model_path)
    X_scaler = joblib.load(scalerX_path)
    y_scaler = joblib.load(scalerY_path)

    # Scale test data
    X_scaled = X_scaler.transform(df_lstm)
    y_scaled = y_scaler.transform(df_lstm['Close'].values.reshape(-1, 1))

    # Create test 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)

    # Predict
    y_pred_scaled = model.predict(X_seq)
    y_pred = y_scaler.inverse_transform(y_pred_scaled)
    y_true = y_scaler.inverse_transform(y_seq.reshape(-1, 1))

    date_series = df_test['Date'].iloc[window_size + forecast_horizon: wind
ow_size + forecast_horizon + len(y_true)].values
    # Evaluation
    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"📊 Test R²: {r2:.4f}, RMSE: {rmse:.2f}, MAE: {mae:.2f}")

    # Plot predictions
    plt.figure(figsize=(14, 5))
    plt.plot(date_series, y_true, label='Actual', color='blue')
    plt.plot(date_series, y_pred, label='Predicted', color='orange')
    plt.title(f"LSTM ({model_type}) Forecast (t+{forecast_horizon}) — Test
Set")
    plt.xlabel("Date")
    plt.ylabel("Close Price")
    plt.legend()
    plt.tight_layout()
    plt.show()

    # Plot residuals
    residuals = y_true.flatten() - y_pred.flatten()
    plt.figure(figsize=(14, 4))
    plt.plot(residuals, color='purple')
    plt.axhline(0, linestyle='--', color='black')
    plt.title(f"Residuals (Actual - Predicted) — LSTM {model_type} t+{forec
ast_horizon}")
    plt.tight_layout()
    plt.show()

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

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

# Simple LSTM evaluation
for h in [1, 3, 7]:
    result = load_and_test_lstm(model_dir, model_type='simple', forecast_ho
rizon=h, df_lstm=df_lstm)
    if result is not None:
        print(result)

# Stacked LSTM evaluation
for h in [1, 3, 7]:
    result = load_and_test_lstm(model_dir, model_type='stacked', forecast_h
orizon=h, df_lstm=df_lstm)
    if result is not None:
        print(result)
```

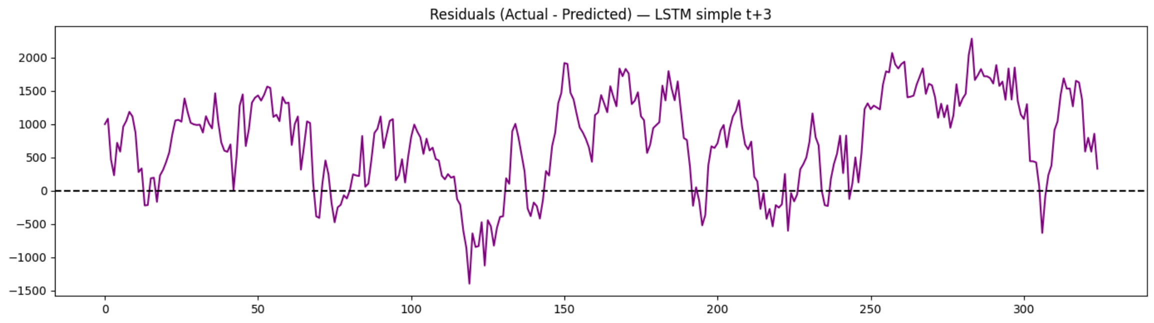
```
🔍 Testing lstm_tplus1_simple
 1/11 ━━━━━━━━━━━━━━━━━━━━ 1s 148ms/step

b:\DCU\Practicum\Proj\App\venv_3_11\Lib\site-packages\sklearn\utils\valida
tion.py:2742: UserWarning: X has feature names, but MinMaxScaler was fitte
d without feature names
  warnings.warn(

11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step
📊 Test R²: 0.5364, RMSE: 444.72, MAE: 376.64
```



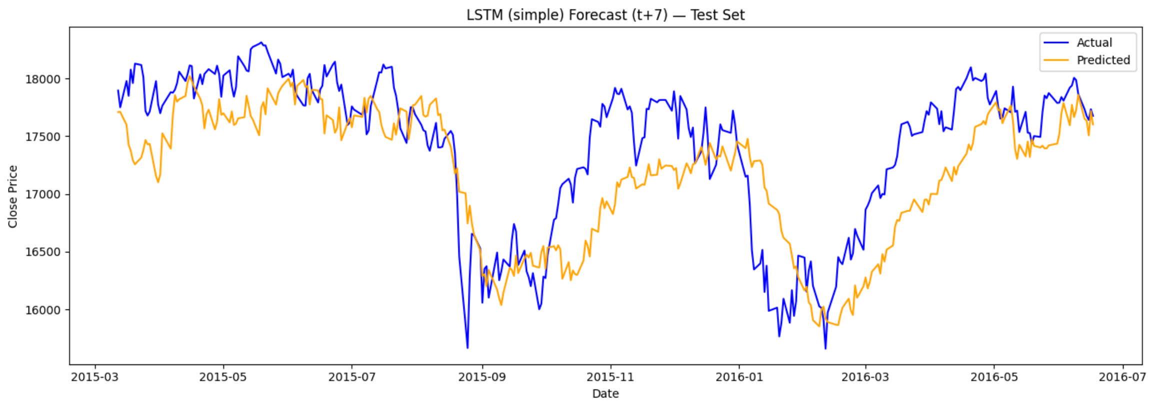


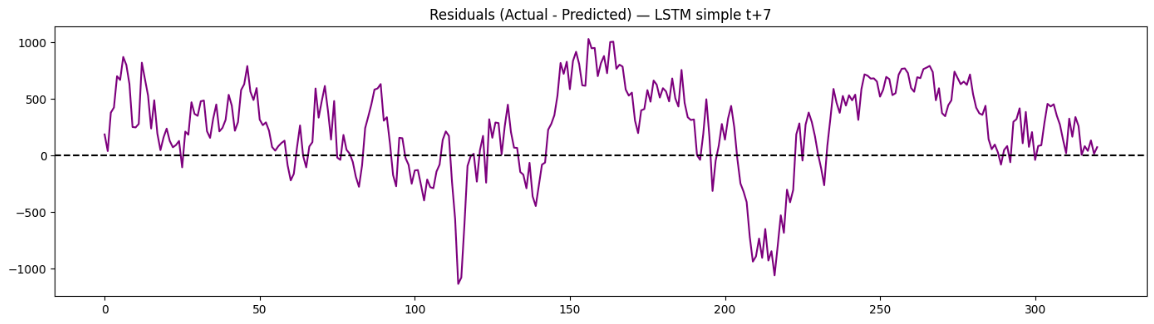
```
🔍 Testing lstm_tplus3_simple
 1/11 ━━━━━━━━━━━━━━━━━━━━ 1s 145ms/step

b:\DCU\Practicum\Proj\App\venv_3_11\Lib\site-packages\sklearn\utils\valida
tion.py:2742: UserWarning: X has feature names, but MinMaxScaler was fitte
d without feature names
  warnings.warn(

11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
📊 Test R²: -1.4745, RMSE: 1027.04, MAE: 874.86
```



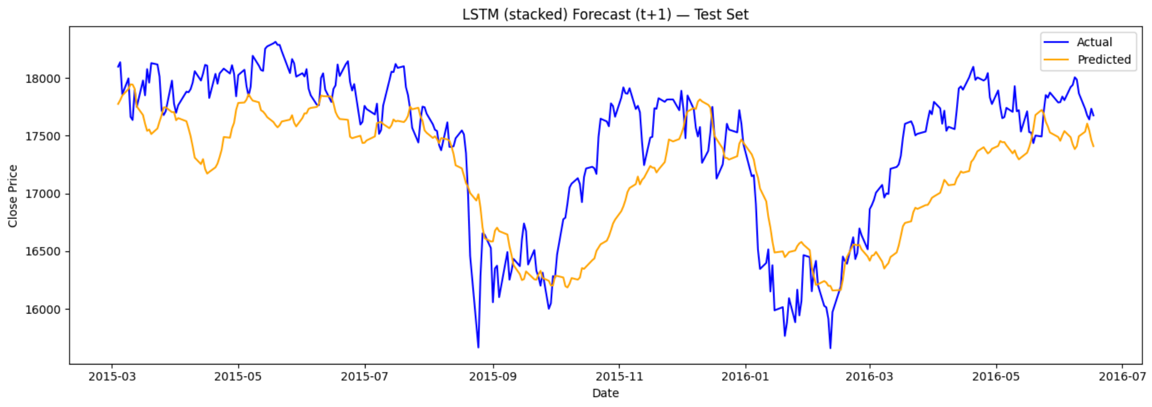


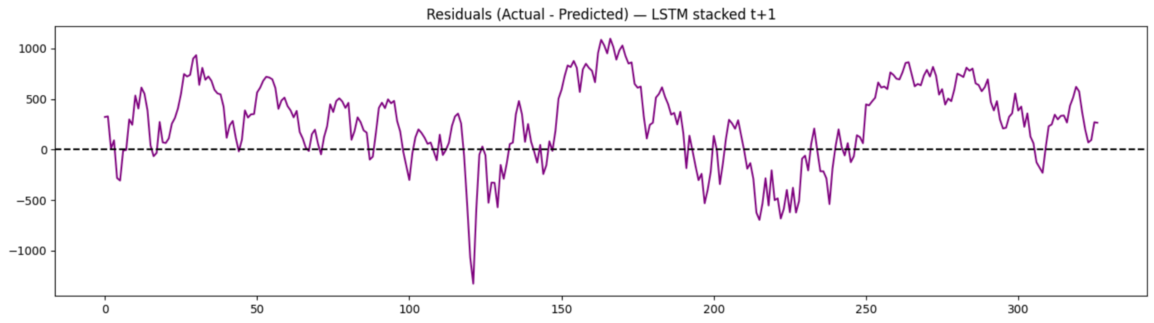
```
🔍 Testing lstm_tplus7_simple
 1/11 ━━━━━━━━━━━━━━━━━━━━ 1s 148ms/step

b:\DCU\Practicum\Proj\App\venv_3_11\Lib\site-packages\sklearn\utils\valida
tion.py:2742: UserWarning: X has feature names, but MinMaxScaler was fitte
d without feature names
  warnings.warn(

11/11 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
📊 Test R²: 0.4921, RMSE: 467.12, MAE: 385.83
```





```
🔍 Testing lstm_tplus1_stacked

b:\DCU\Practicum\Proj\App\venv_3_11\Lib\site-packages\sklearn\utils\valida
tion.py:2742: UserWarning: X has feature names, but MinMaxScaler was fitte
d without feature names
  warnings.warn(

11/11 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step
📊 Test R²: 0.4652, RMSE: 477.63, MAE: 392.49
```

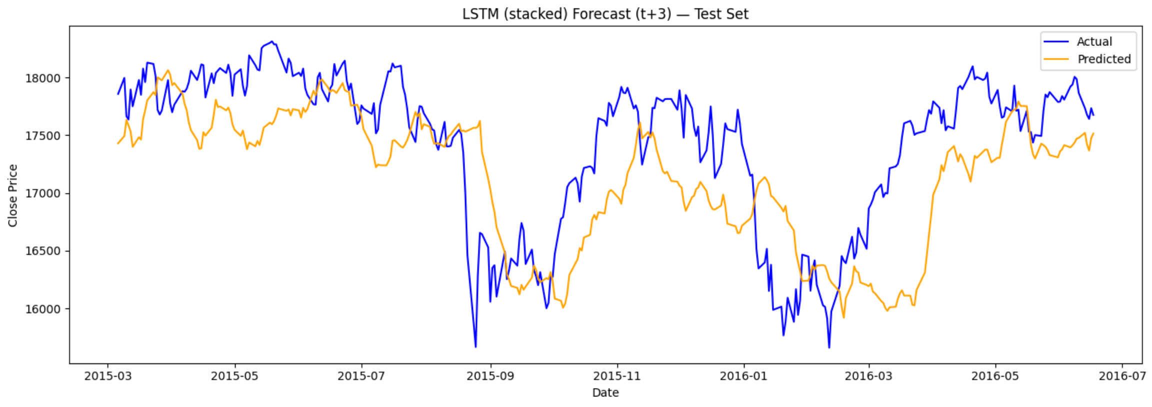


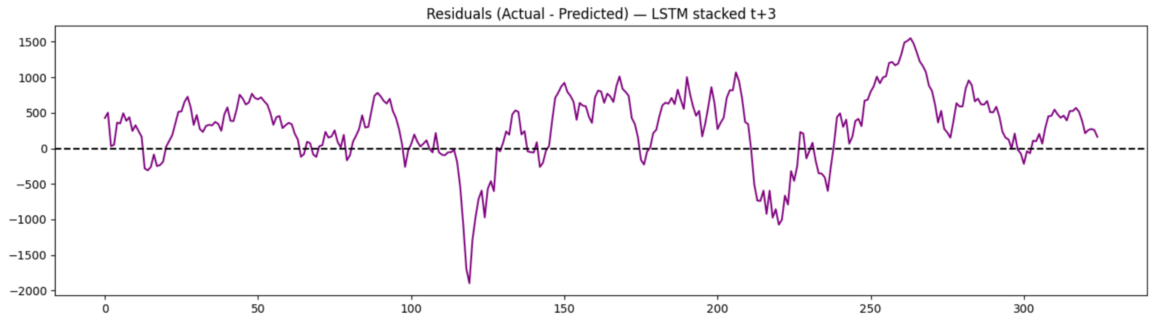


🔍 Testing lstm_tplus3_stacked

```
b:\DCU\Practicum\Proj\App\venv_3_11\Lib\site-packages\sklearn\utils\valida
tion.py:2742: UserWarning: X has feature names, but MinMaxScaler was fitte
d without feature names
  warnings.warn(
```

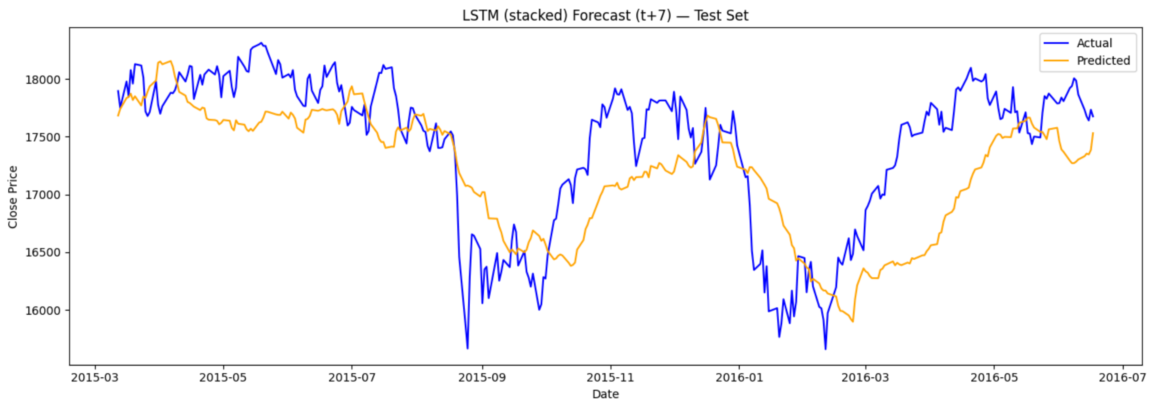
```
11/11 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step
📊 Test R²: 0.2065, RMSE: 581.59, MAE: 470.92
```

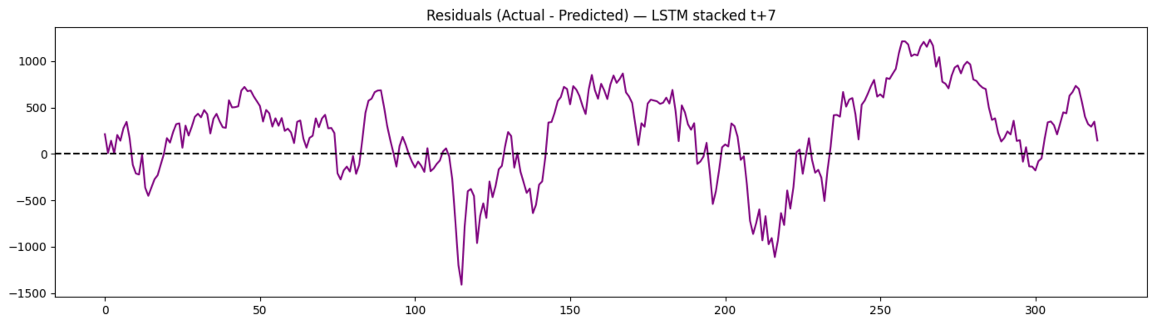




🔍 Testing lstm_tplus7_stacked

```
b:\DCU\Practicum\Proj\App\venv_3_11\Lib\site-packages\sklearn\utils\valida
tion.py:2742: UserWarning: X has feature names, but MinMaxScaler was fitte
d without feature names
  warnings.warn(
11/11 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step
📊 Test R²: 0.3454, RMSE: 530.32, MAE: 437.21
```


LSTM (stacked) Forecast (t+7) — Test Set


Residuals (Actual - Predicted) — LSTM stacked t+7

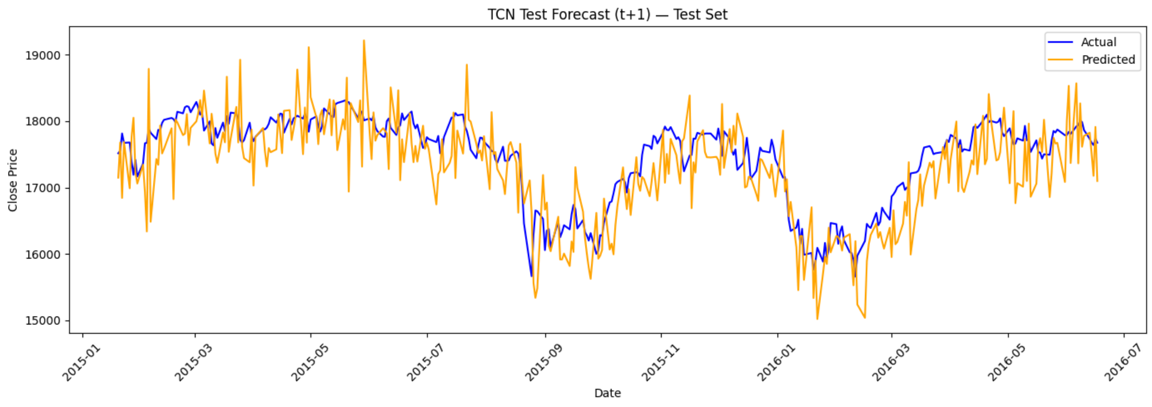
In [17]:
```
Dropped_tcn_cols = [
    'Open', 'High', 'Low', 'Volume', 'Adj Close', 'Date', 'Label', 'Targe
t',
    'Volatility_Log_10', 'cl-op', 'hi-lo', 'pct_change',
    'total_buying_intent', 'total_prediction_intent'
]
df_tcn = df_test.drop(columns=Dropped_tcn_cols).copy()
```

In [30]:
```
df_tcn.columns
```

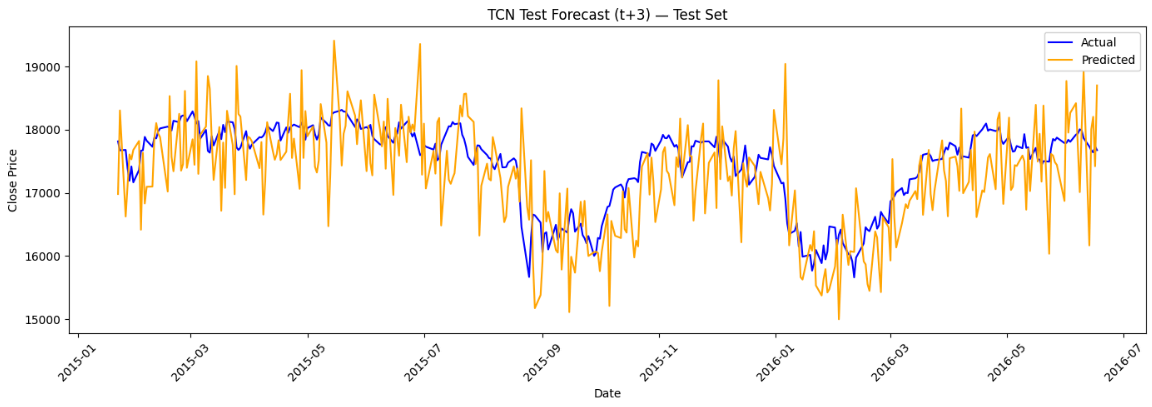
Out[30]:
```
Index(['Close', 'Log_Returns', 'vader_news_sentiment',
       'FinBERT_news_sentiment', 'Smart_news_sentiment', 'news_buying_inte
nt',
       'news_selling_intent', 'news_uncertainty_intent', 'news_urgency_int
ent',
       'news_prediction_intent', 'news_fear_intent', 'news_greed_intent',
       '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 [ ]:
```
def test_tcn_logreturn_model(df_tcn, forecast_horizon=1, window_size=30):
    print(f"\n🧪 Testing TCN model for Log Returns → Close at t+{forecast_
horizon}")

    model_name = f"tcn_logret_tplus{forecast_horizon}"
    base_path = "B:/DCU/Practicum/Proj/Models"

    # === Load model and scalers ===
    model = load_model(f"{base_path}/{model_name}.keras", custom_objects=
{"TCN": TCN})
    X_scaler = joblib.load(f"{base_path}/{model_name}_scalerX.pkl")
    y_scaler = joblib.load(f"{base_path}/{model_name}_scalerY.pkl")

    # === Use df_tcn directly ===
    X_scaled = X_scaler.transform(df_tcn.values)
    y_scaled = y_scaler.transform(df_tcn['Log_Returns'].shift(-forecast_hor
izon).dropna().values.reshape(-1, 1))

    # === Create sequences ===
    X_seq, y_seq = [], []
    valid_len = min(len(X_scaled), len(y_scaled))
    for i in range(window_size, valid_len):
        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)

    # === Predict
    y_pred_scaled = model.predict(X_seq)
    y_pred_log = y_scaler.inverse_transform(y_pred_scaled).flatten()
    y_true_log = y_scaler.inverse_transform(y_seq.reshape(-1, 1)).flatten()

    # === Get Close_t from df_tcn
    close_t = df_tcn['Close'].iloc[window_size - 1 : window_size - 1 + len
(y_pred_log)].values

    # === Predicted & Actual Close at t+h
    y_pred_close = close_t * np.exp(y_pred_log)
    y_true_close = df_tcn['Close'].shift(-forecast_horizon).dropna().iloc[w
indow_size:].values

    # === Dates for plotting
    date_series = df_test['Date'].iloc[window_size + forecast_horizon: wind
ow_size + forecast_horizon + len(y_pred_close)]

    # === Metrics
    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 Test R²: {r2:.4f}, RMSE: {rmse:.2f}, MAE: {mae:.2f}")

    # === Save metrics
    metrics_path = os.path.join(base_path, f"{model_name}_test_metrics.tx
t")
    with open(metrics_path, "w") as f:
        f.write(f"Test Forecast Horizon = t+{forecast_horizon}\n")
        f.write(f"Test R²   = {r2:.4f}\n")
        f.write(f"Test RMSE = {rmse:.2f}\n")
        f.write(f"Test MAE  = {mae:.2f}\n")

    # === Plot
    plot_path = os.path.join(base_path, f"{model_name}_testplot.png")
    plt.figure(figsize=(14, 5))
    plt.plot(date_series, y_true_close, label='Actual', color='blue')
    plt.plot(date_series, y_pred_close, label='Predicted', color='orange')
    plt.title(f"TCN Test Forecast (t+{forecast_horizon}) — Test Set")
    plt.xlabel("Date")
    plt.ylabel("Close Price")
    plt.xticks(rotation=45)
    plt.legend()
    plt.tight_layout()
    plt.savefig(plot_path)
    plt.show()
    print(f"🖼️ Saved test plot to {plot_path}")

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

# Final TCN test loop
for h in [1, 3, 7]:
    test_tcn_logreturn_model(df_tcn=df_tcn, forecast_horizon=h)
```

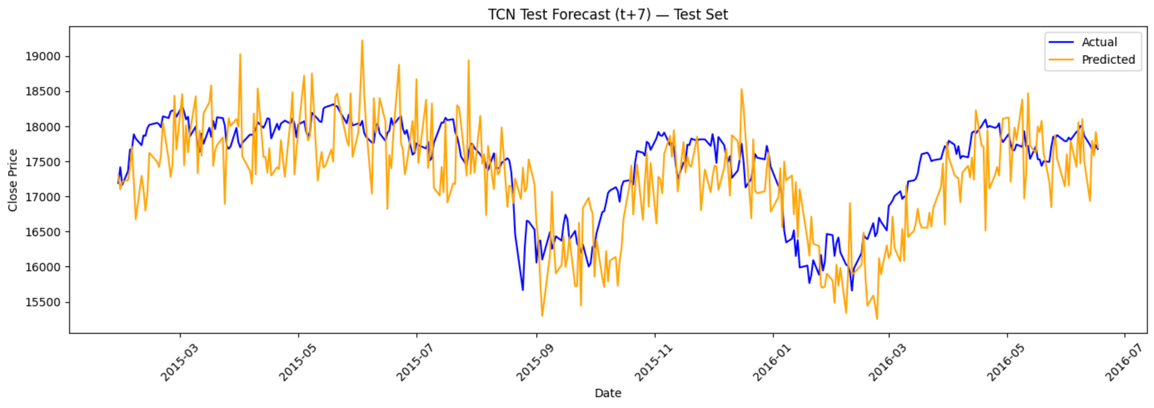
```
🧪 Testing TCN model for Log Returns → Close at t+1

b:\DCU\Practicum\Proj\App\venv_3_11\Lib\site-packages\sklearn\utils\valida
tion.py:2749: UserWarning: X does not have valid feature names, but MinMax
Scaler was fitted with feature names
  warnings.warn(

12/12 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step
📊 TCN Test R²: 0.3939, RMSE: 499.45, MAE: 404.61
```



```
🖼️ Saved test plot to B:/DCU/Practicum/Proj/Models\tcn_logret_tplus1_testp
lot.png

🧪 Testing TCN model for Log Returns → Close at t+3

b:\DCU\Practicum\Proj\App\venv_3_11\Lib\site-packages\sklearn\utils\valida
tion.py:2749: UserWarning: X does not have valid feature names, but MinMax
Scaler was fitted with feature names
  warnings.warn(

12/12 ━━━━━━━━━━━━━━━━━━━━ 1s 29ms/step
📊 TCN Test R²: -0.0241, RMSE: 651.03, MAE: 525.44
```



```
🖼️ Saved test plot to B:/DCU/Practicum/Proj/Models\tcn_logret_tplus3_testp
lot.png

🧪 Testing TCN model for Log Returns → Close at t+7

b:\DCU\Practicum\Proj\App\venv_3_11\Lib\site-packages\sklearn\utils\valida
tion.py:2749: UserWarning: X does not have valid feature names, but MinMax
Scaler was fitted with feature names
  warnings.warn(

11/11 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step
📊 TCN Test R²: 0.1690, RMSE: 589.31, MAE: 489.92
```



```
🖼️ Saved test plot to B:/DCU/Practicum/Proj/Models\tcn_logret_tplus7_testp
lot.png
```