1: Import Libraries

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

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

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_sco
re

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 tcn import TCN

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

```
WARNING:tensorflow:From b:\Dublin City University\Practicum\Proj\venv_311
\Lib\site-packages\keras\src\losses.py:2976: The name tf.losses.sparse_sof
tmax_cross_entropy is deprecated. Please use tf.compat.v1.losses.sparse_so
ftmax_cross_entropy instead.

b:\Dublin City University\Practicum\Proj\venv_311\Scripts\python.exe
```

2: Load and Explore Dataset

In [2]:
```
multimodal = pd.read_csv("B:/Dublin City University/Practicum/Proj/Dataset/main/processed/multimodal_dataset_final_v4_1.csv", parse_dates=["Date"])
multimodal.head()
```

Out[2]:

	Date	Open	High	Low	Close	Volume	Adj Close
0	2016-07-01	17924.240234	18002.380859	17916.910156	17949.369141	82160000	17949.369141
1	2016-06-30	17712.759766	17930.609375	17711.800781	17929.990234	133030000	17929.990234
2	2016-06-29	17456.019531	17704.509766	17456.019531	17694.679688	106380000	17694.679688
3	2016-06-28	17190.509766	17409.720703	17190.509766	17409.720703	112190000	17409.720703
4	2016-06-27	17355.210938	17355.210938	17063.080078	17140.240234	138740000	17140.240234

5 rows × 56 columns

In [3]:
```
missing_values = multimodal.isnull().sum()
print("\nMissing values per column:")
missing_values[missing_values > 0]
```

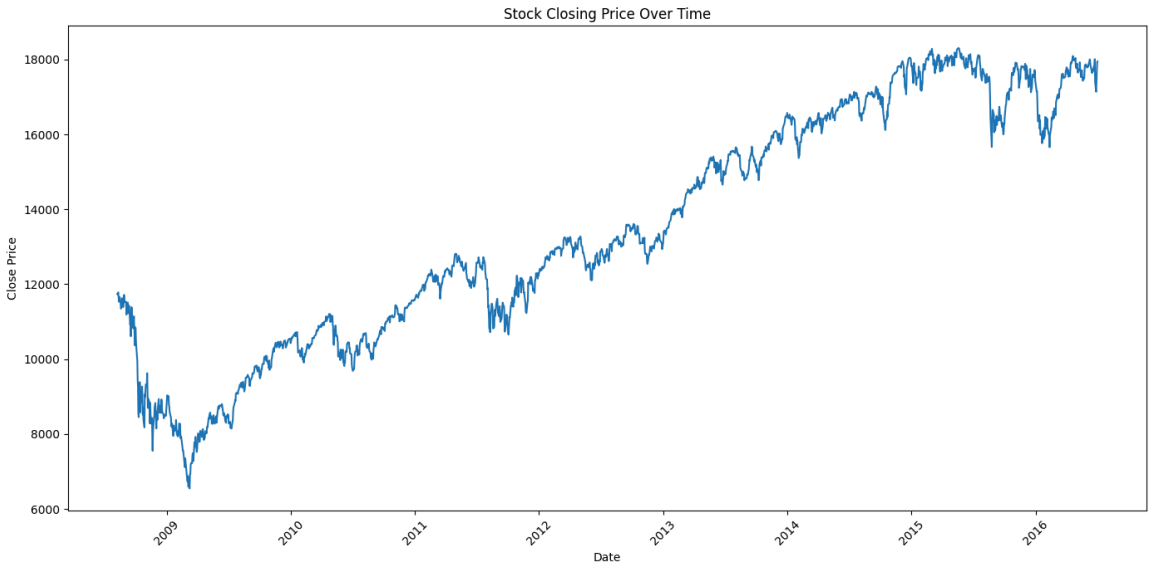
```
Missing values per column:
```

Out[3]:
```
Log_Returns           1
Volatility_Log_10    10
pct_change            1
Next_3_Close          3
Next_7_Close          7
Next_Close            1
dtype: int64
```

In [4]:
```
multimodal.describe()
multimodal.dtypes
```

Out[4]:
```
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
Next_3_Close                         float64
Next_7_Close                         float64
Next_Close                           float64
dtype: object
```

In [5]:
```
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 [6]:
```
# Drop top 10 (rolling NaNs) and bottom 7 (from shift(-7))
multimodal_modelling = multimodal.iloc[10:-7].copy()

# Optional: reset index
# multimodal_modelling.reset_index(drop=True, inplace=True)

# Sanity check
print(f"Shape: {multimodal_modelling.shape}")
print(multimodal_modelling.isnull().sum())
```

```
Shape: (1972, 56)
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
Next_3_Close                  0
Next_7_Close                  0
Next_Close                    0
dtype: int64
```

In [7]:
```
df_targets = multimodal_modelling[["Date", "Target", "Label", "Close", "Next_Close", "Next_3_Close", "Next_7_Close"]].copy()

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

drop_cols_lstm = ["Date", "Label", "Target", "Next_Close", "Next_3_Close", "Next_7_Close"]
df_lstm = multimodal_modelling.drop(columns=drop_cols_lstm).copy()

drop_cols_tcn = ["Date", "Label", "Target", "Next_Close", "Next_3_Close", "Next_7_Close"]
df_tcn = multimodal_modelling.drop(columns=drop_cols_tcn).copy()
```

In [8]:
```
df_tcn.columns
```

Out[8]:
```
Index(['Open', 'High', 'Low', 'Close', 'Volume', 'Adj Close', 'Log_Returns',
       'Volatility_Log_10', 'cl-op', 'hi-lo', 'vader_news_sentiment',
       'FinBERT_news_sentiment', 'Smart_news_sentiment', 'news_buying_intent',
       'news_selling_intent', 'news_uncertainty_intent', 'news_urgency_intent',
       'news_prediction_intent', 'news_fear_intent', 'news_greed_intent',
       'news_question_intent', 'news_action_intent', 'vader_reddit_sentiment',
       '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_intent',
       '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_intent',
       'total_greed_intent', 'total_question_intent', 'total_action_intent',
       '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 [9]:
```
df_arima.head()
```

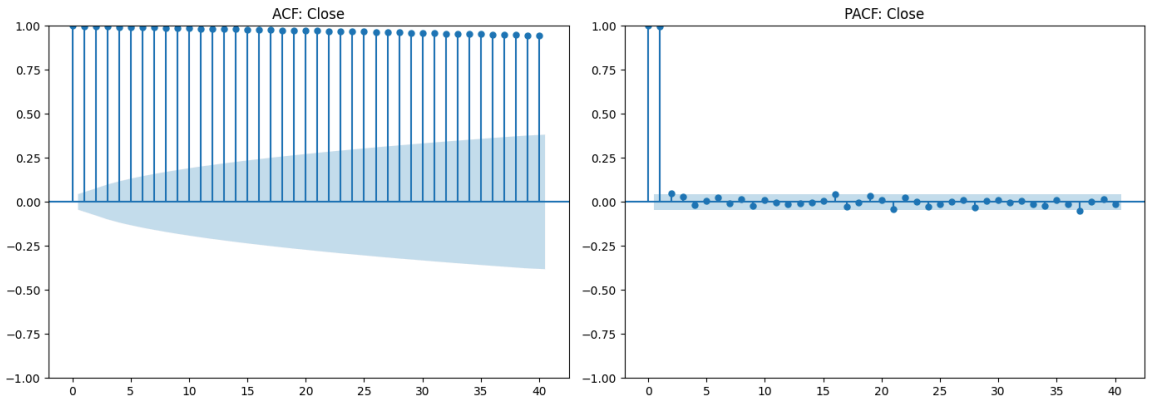
Out[9]:

	Close
Date	
2016-06-17	17675.160156
2016-06-16	17733.099609
2016-06-15	17640.169922
2016-06-14	17674.820312
2016-06-13	17732.480469

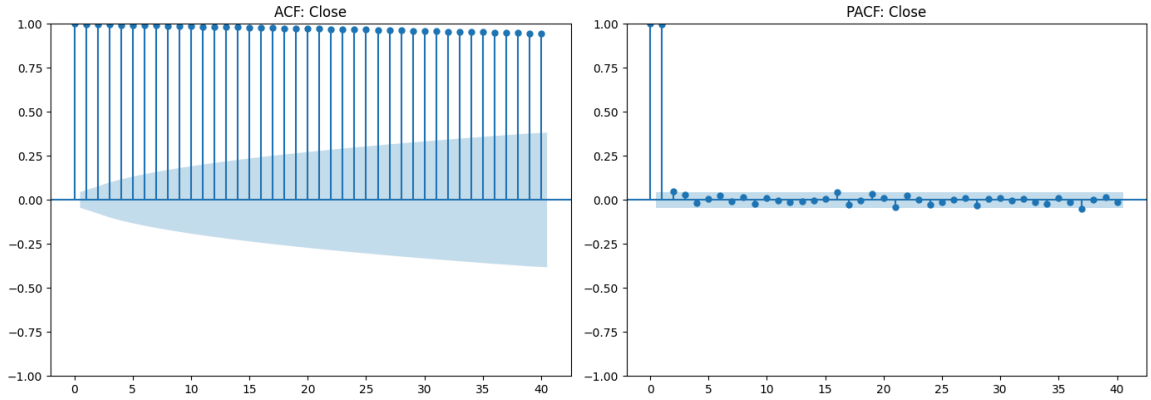
In [10]:
```
result = adfuller(df_arima["Close"])
print(f"ADF Statistic: {result[0]}")
print(f"p-value: {result[1]}")
```

```
ADF Statistic: -1.210171226984291
p-value: 0.6692050737972017
```

In [11]:
```
plt.figure(figsize=(14, 5))
plt.subplot(1, 2, 1)
plot_acf(multimodal_modelling["Close"], lags=40, ax=plt.gca())
plt.title("ACF: Close")

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

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



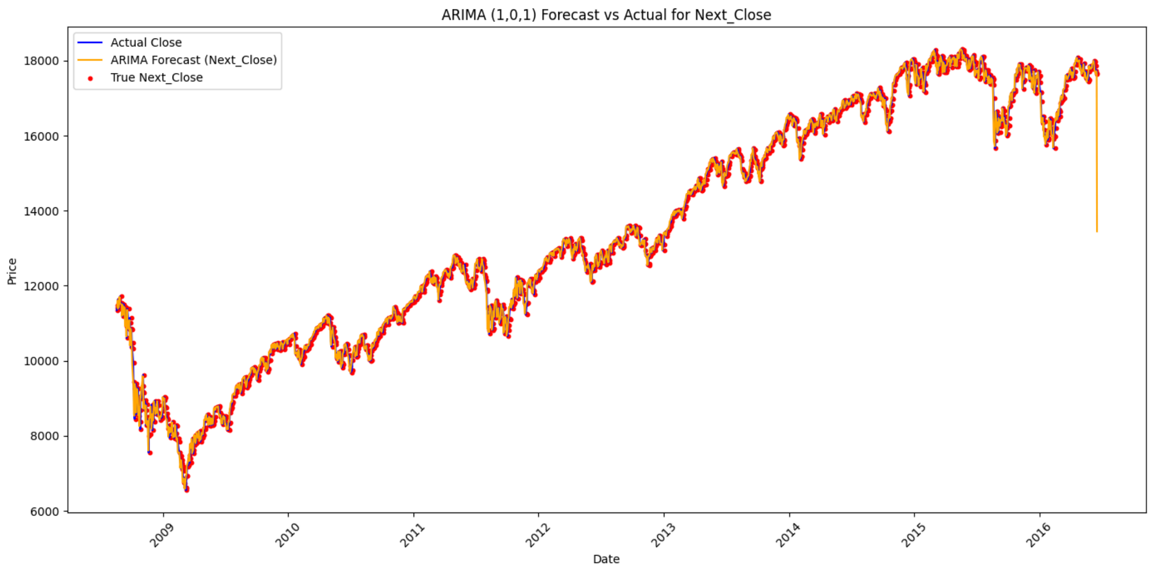
In [12]:
```
plt.figure(figsize=(14, 5))
plt.subplot(1, 2, 1)
plot_acf(df_arima["Close"], lags=40, ax=plt.gca())
plt.title("ACF: Close")

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()
```

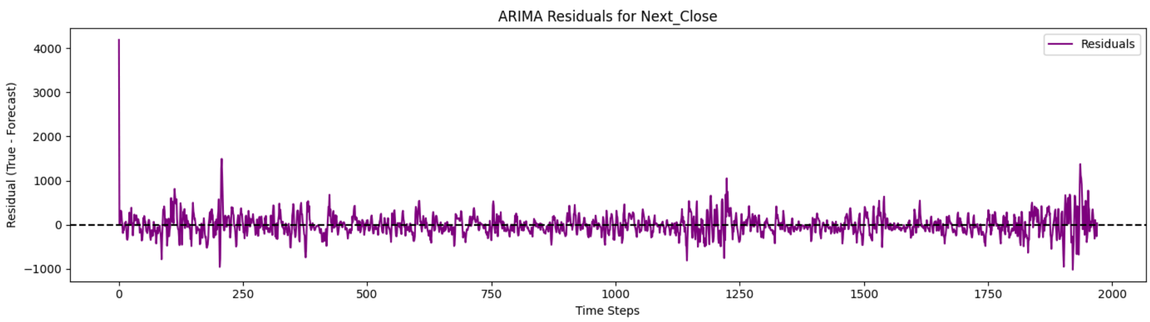


In [13]:
```
def train_arima(df_arima, df_targets, target_col='Next_Close', order=(1, 0,
1)):
    print(f"\nTraining ARIMA model for target: {target_col}")

    if target_col == 'Next_Close':
        shift_n = 1
    elif 'Next_' in target_col and '_Close' in target_col:
        shift_n = int(target_col.replace('Next_', '').replace('_Close',
''))
    else:
        shift_n = 1


    model = ARIMA(df_arima["Close"], order=order)
    model_fit = model.fit()

    forecast = model_fit.predict(start=0, end=len(df_arima) - 1)

    # Shift forecast to align with future target (Next_N_Close)
    forecast = forecast.shift(shift_n)
    forecast = forecast[:len(df_targets)]

    true = df_targets[target_col]

    min_len = min(len(forecast), len(true))
    forecast = forecast[:min_len]
    true = true[:min_len]

    mask = forecast.notna()
    forecast_clean = forecast[mask].values
    true_clean = true.values[mask.values]

    r2 = r2_score(true_clean, forecast_clean)
    rmse = np.sqrt(mean_squared_error(true_clean, forecast_clean))
    mae = mean_absolute_error(true_clean, forecast_clean)
    print(f"{target_col} — R²: {r2:.4f}, RMSE: {rmse:.2f}, MAE: {mae:.2f}")

    return model_fit, forecast_clean, true_clean


def plot_arima_results(df_arima, forecast, true, target_col='Next_Close'):
    plt.figure(figsize=(14, 7))
    plt.plot(df_arima.index, df_arima["Close"], label='Actual Close', color
='blue')
    plt.plot(df_arima.index[:len(forecast)], forecast, label=f'ARIMA Foreca
st ({target_col})', color='orange')
    plt.scatter(df_arima.index[:len(true)], true, label=f'True {target_co
l}', color='red', s=10)
    plt.title(f'ARIMA (1,0,1) Forecast vs Actual for {target_col}')
    plt.xlabel('Date')
    plt.ylabel('Price')
    plt.legend()
    plt.xticks(rotation=45)
    plt.tight_layout()
    plt.show()

for target in ['Next_Close', 'Next_3_Close', 'Next_7_Close']:
    model, forecast, true = train_arima(df_arima, df_targets, target_col=ta
rget)
    plot_arima_results(df_arima, forecast, true, target_col=target)
```

```
    # Plot residuals
    residuals = true - forecast
    plt.figure(figsize=(14, 4))
    plt.plot(residuals, label="Residuals", color='purple')
    plt.axhline(0, linestyle='--', color='black')
    plt.title(f'ARIMA Residuals for {target}')
    plt.xlabel('Time Steps')
    plt.ylabel('Residual (True - Forecast)')
    plt.legend()
    plt.tight_layout()
    plt.show()
```

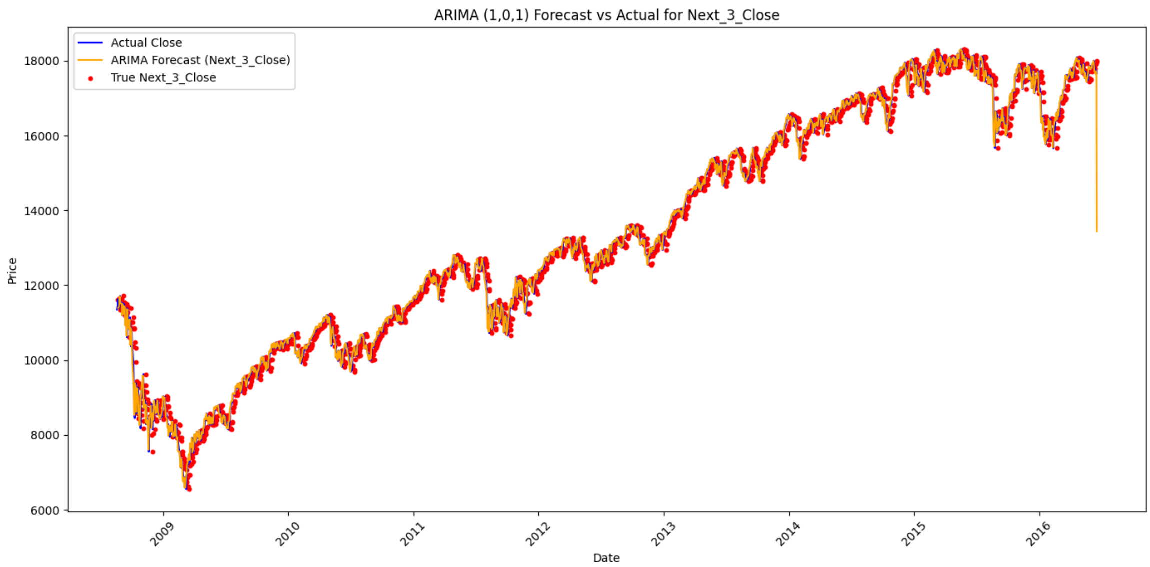
```
Training ARIMA model for target: Next_Close
Next_Close — R²: 0.9937, RMSE: 248.72, MAE: 171.23

b:\Dublin City University\Practicum\Proj\venv_311\Lib\site-packages\statsm
odels\tsa\base\tsa_model.py:473: ValueWarning: A date index has been provi
ded, but it has no associated frequency information and so will be ignored
when e.g. forecasting.
  self._init_dates(dates, freq)
b:\Dublin City University\Practicum\Proj\venv_311\Lib\site-packages\statsm
odels\tsa\base\tsa_model.py:473: ValueWarning: A date index has been provi
ded, but it is not monotonic and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
b:\Dublin City University\Practicum\Proj\venv_311\Lib\site-packages\statsm
odels\tsa\base\tsa_model.py:473: ValueWarning: A date index has been provi
ded, but it has no associated frequency information and so will be ignored
when e.g. forecasting.
  self._init_dates(dates, freq)
b:\Dublin City University\Practicum\Proj\venv_311\Lib\site-packages\statsm
odels\tsa\base\tsa_model.py:473: ValueWarning: A date index has been provi
ded, but it is not monotonic and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
b:\Dublin City University\Practicum\Proj\venv_311\Lib\site-packages\statsm
odels\tsa\base\tsa_model.py:473: ValueWarning: A date index has been provi
ded, but it has no associated frequency information and so will be ignored
when e.g. forecasting.
  self._init_dates(dates, freq)
b:\Dublin City University\Practicum\Proj\venv_311\Lib\site-packages\statsm
odels\tsa\base\tsa_model.py:473: ValueWarning: A date index has been provi
ded, but it is not monotonic and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
```

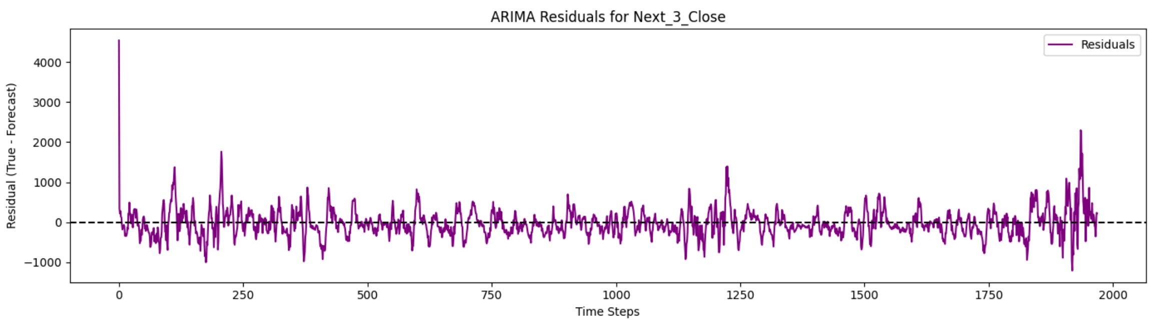





```
Training ARIMA model for target: Next_3_Close
Next_3_Close — R²: 0.9873, RMSE: 352.92, MAE: 254.10
```

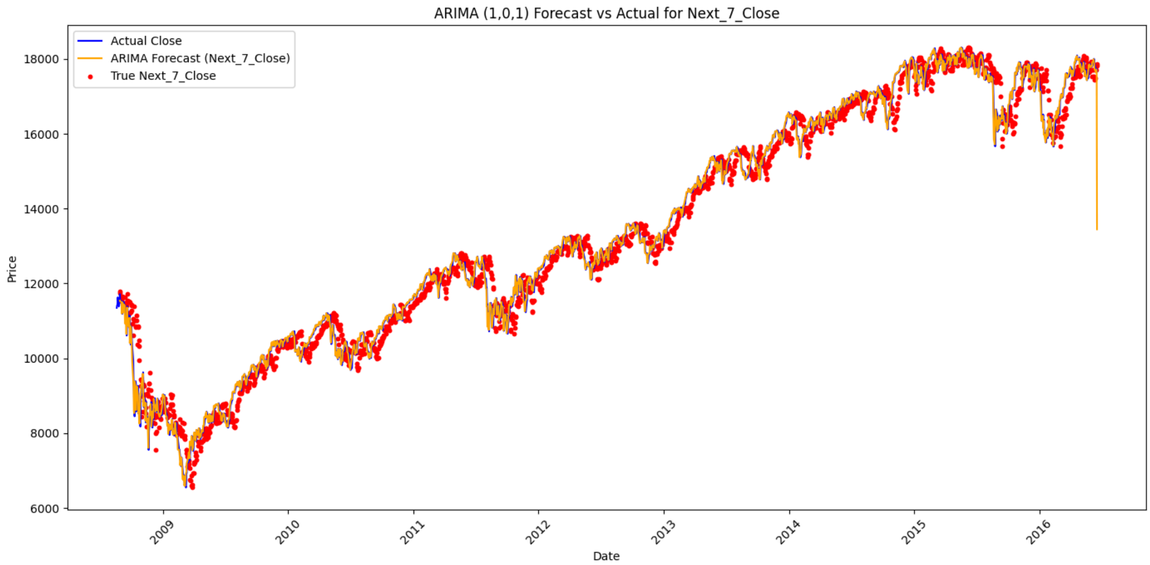
```
b:\Dublin City University\Practicum\Proj\venv_311\Lib\site-packages\statsm
odels\tsa\base\tsa_model.py:473: ValueWarning: A date index has been provi
ded, but it has no associated frequency information and so will be ignored
when e.g. forecasting.
  self._init_dates(dates, freq)
b:\Dublin City University\Practicum\Proj\venv_311\Lib\site-packages\statsm
odels\tsa\base\tsa_model.py:473: ValueWarning: A date index has been provi
ded, but it is not monotonic and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
b:\Dublin City University\Practicum\Proj\venv_311\Lib\site-packages\statsm
odels\tsa\base\tsa_model.py:473: ValueWarning: A date index has been provi
ded, but it has no associated frequency information and so will be ignored
when e.g. forecasting.
  self._init_dates(dates, freq)
b:\Dublin City University\Practicum\Proj\venv_311\Lib\site-packages\statsm
odels\tsa\base\tsa_model.py:473: ValueWarning: A date index has been provi
ded, but it is not monotonic and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
b:\Dublin City University\Practicum\Proj\venv_311\Lib\site-packages\statsm
odels\tsa\base\tsa_model.py:473: ValueWarning: A date index has been provi
ded, but it has no associated frequency information and so will be ignored
when e.g. forecasting.
  self._init_dates(dates, freq)
b:\Dublin City University\Practicum\Proj\venv_311\Lib\site-packages\statsm
odels\tsa\base\tsa_model.py:473: ValueWarning: A date index has been provi
ded, but it is not monotonic and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
```

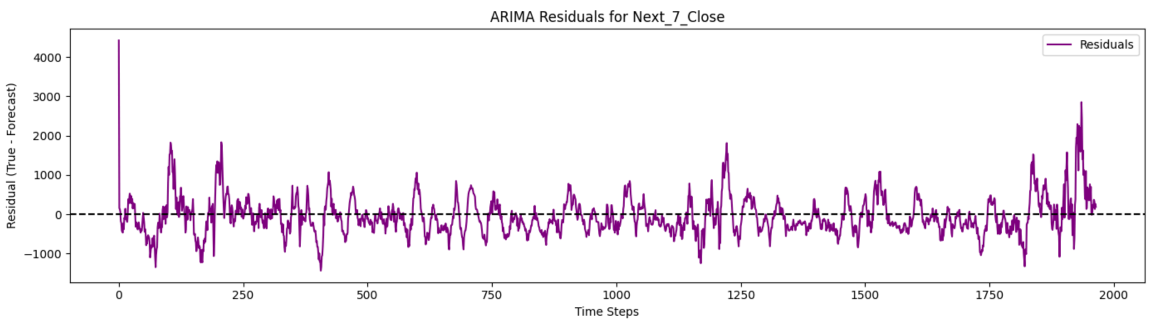





```
Training ARIMA model for target: Next_7_Close
Next_7_Close – R²: 0.9750, RMSE: 494.68, MAE: 367.74
```

```
b:\Dublin City University\Practicum\Proj\venv_311\Lib\site-packages\statsm
odels\tsa\base\tsa_model.py:473: ValueWarning: A date index has been provi
ded, but it has no associated frequency information and so will be ignored
when e.g. forecasting.
  self._init_dates(dates, freq)
b:\Dublin City University\Practicum\Proj\venv_311\Lib\site-packages\statsm
odels\tsa\base\tsa_model.py:473: ValueWarning: A date index has been provi
ded, but it is not monotonic and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
b:\Dublin City University\Practicum\Proj\venv_311\Lib\site-packages\statsm
odels\tsa\base\tsa_model.py:473: ValueWarning: A date index has been provi
ded, but it has no associated frequency information and so will be ignored
when e.g. forecasting.
  self._init_dates(dates, freq)
b:\Dublin City University\Practicum\Proj\venv_311\Lib\site-packages\statsm
odels\tsa\base\tsa_model.py:473: ValueWarning: A date index has been provi
ded, but it is not monotonic and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
b:\Dublin City University\Practicum\Proj\venv_311\Lib\site-packages\statsm
odels\tsa\base\tsa_model.py:473: ValueWarning: A date index has been provi
ded, but it has no associated frequency information and so will be ignored
when e.g. forecasting.
  self._init_dates(dates, freq)
b:\Dublin City University\Practicum\Proj\venv_311\Lib\site-packages\statsm
odels\tsa\base\tsa_model.py:473: ValueWarning: A date index has been provi
ded, but it is not monotonic and so will be ignored when e.g. forecasting.
  self._init_dates(dates, freq)
```





LSTM Model

In [14]:
```
df_lstm.columns
```

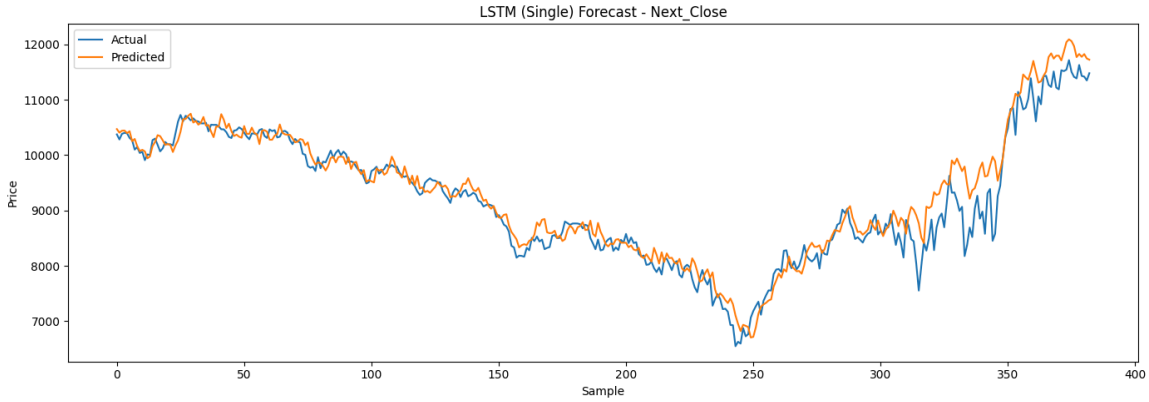
Out[14]:
```
Index(['Open', 'High', 'Low', 'Close', 'Volume', 'Adj Close', 'Log_Returns',
       'Volatility_Log_10', 'cl-op', 'hi-lo', 'vader_news_sentiment',
       'FinBERT_news_sentiment', 'Smart_news_sentiment', 'news_buying_intent',
       'news_selling_intent', 'news_uncertainty_intent', 'news_urgency_intent',
       'news_prediction_intent', 'news_fear_intent', 'news_greed_intent',
       'news_question_intent', 'news_action_intent', 'vader_reddit_sentiment',
       '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_intent',
       '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_intent',
       'total_greed_intent', 'total_question_intent', 'total_action_intent',
       'sentiment_minus_uncertainty', 'sentiment_minus_fear',
       'sentiment_minus_action', 'sentiment_minus_urgency',
       'sentiment_minus_prediction'],
      dtype='object')
```

In [15]:
```
def train_lstm(df_lstm, df_targets, target_col='Next_Close', window_size=60, epochs=50, batch_size=32, stacked=False):
    print(f"\n🧠 Training LSTM ({'Stacked' if stacked else 'Single'}) model for target: {target_col}")

    # Scale features and target
    X_scaler = MinMaxScaler()
    y_scaler = MinMaxScaler()
    X_scaled = X_scaler.fit_transform(df_lstm)
    y_scaled = y_scaler.fit_transform(df_targets[[target_col]])

    # 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)

    # Split
    split = int(0.8 * len(X_seq))
    X_train, X_test = X_seq[:split], X_seq[split:]
    y_train, y_test = y_seq[:split], y_seq[split:]

    # Model
    model = Sequential()
    if stacked:
        model.add(LSTM(64, return_sequences=True, input_shape=(X_train.shape[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='adam', loss='mse')
    early_stop = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)

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

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

    # Evaluation
    rmse = np.sqrt(mean_squared_error(y_true, y_pred))
    r2 = r2_score(y_true, y_pred)
    mae = mean_absolute_error(y_true, y_pred)
    print(f"{target_col} — R²: {r2:.4f}, RMSE: {rmse:.2f}, MAE: {mae:.2f}")
```

```
    # Plot
    plt.figure(figsize=(14, 5))
    plt.plot(y_true, label='Actual')
    plt.plot(y_pred, label='Predicted')
    plt.title(f"LSTM ({'Stacked' if stacked else 'Single'}) Forecast - {tar
get_col}")
    plt.xlabel('Sample')
    plt.ylabel('Price')
    plt.legend()
    plt.tight_layout()
    plt.show()

    return model, y_true, y_pred
for target in ['Next_Close', 'Next_3_Close', 'Next_7_Close']:
    model, y_true, y_pred = train_lstm(df_lstm, df_targets, target_col=targ
et, stacked=False)

for target in ['Next_Close', 'Next_3_Close', 'Next_7_Close']:
    model, y_true, y_pred = train_lstm(df_lstm, df_targets, target_col=targ
et, stacked=True)
```

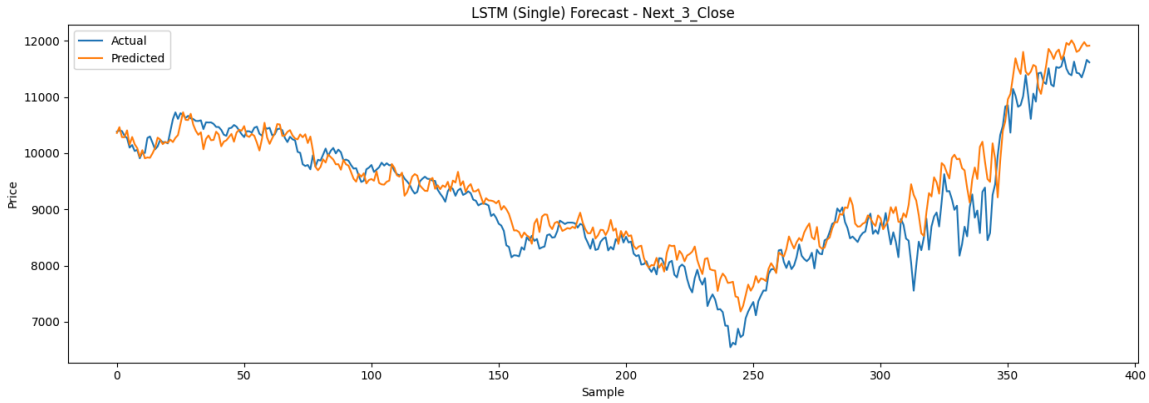
```
🧠 Training LSTM (Single) model for target: Next_Close
WARNING:tensorflow:From b:\Dublin City University\Practicum\Proj\venv_311
\Lib\site-packages\keras\src\backend.py:873: The name tf.get_default_graph
is deprecated. Please use tf.compat.v1.get_default_graph instead.

WARNING:tensorflow:From b:\Dublin City University\Practicum\Proj\venv_311
\Lib\site-packages\keras\src\optimizers\__init__.py:309: The name tf.trai
n.Optimizer is deprecated. Please use tf.compat.v1.train.Optimizer instea
d.

Epoch 1/50
WARNING:tensorflow:From b:\Dublin City University\Practicum\Proj\venv_311
\Lib\site-packages\keras\src\utils\tf_utils.py:492: The name tf.ragged.Rag
gedTensorValue is deprecated. Please use tf.compat.v1.ragged.RaggedTensorV
alue instead.

43/43 [==============================] - 2s 21ms/step - loss: 0.0371 - val
_loss: 0.0030
Epoch 2/50
43/43 [==============================] - 1s 12ms/step - loss: 0.0078 - val
_loss: 7.8116e-04
Epoch 3/50
43/43 [==============================] - 0s 12ms/step - loss: 0.0061 - val
_loss: 7.0894e-04
Epoch 4/50
43/43 [==============================] - 0s 12ms/step - loss: 0.0051 - val
_loss: 4.1611e-04
Epoch 5/50
43/43 [==============================] - 0s 11ms/step - loss: 0.0046 - val
_loss: 7.6185e-04
Epoch 6/50
43/43 [==============================] - 1s 12ms/step - loss: 0.0043 - val
_loss: 6.0504e-04
Epoch 7/50
43/43 [==============================] - 1s 12ms/step - loss: 0.0039 - val
_loss: 3.6248e-04
Epoch 8/50
43/43 [==============================] - 0s 12ms/step - loss: 0.0035 - val
_loss: 5.5363e-04
Epoch 9/50
43/43 [==============================] - 1s 12ms/step - loss: 0.0037 - val
_loss: 2.8764e-04
Epoch 10/50
43/43 [==============================] - 1s 12ms/step - loss: 0.0031 - val
_loss: 5.0133e-04
Epoch 11/50
43/43 [==============================] - 1s 12ms/step - loss: 0.0032 - val
_loss: 2.6693e-04
Epoch 12/50
43/43 [==============================] - 0s 11ms/step - loss: 0.0032 - val
_loss: 2.1703e-04
Epoch 13/50
43/43 [==============================] - 0s 11ms/step - loss: 0.0031 - val
_loss: 3.1758e-04
Epoch 14/50
43/43 [==============================] - 0s 11ms/step - loss: 0.0031 - val
_loss: 8.8360e-04
Epoch 15/50
43/43 [==============================] - 0s 12ms/step - loss: 0.0029 - val
_loss: 2.7700e-04
Epoch 16/50
```

```
43/43 [==============================] - 0s 11ms/step - loss: 0.0028 - val
_loss: 0.0014
Epoch 17/50
43/43 [==============================] - 0s 11ms/step - loss: 0.0026 - val
_loss: 1.9686e-04
Epoch 18/50
43/43 [==============================] - 0s 11ms/step - loss: 0.0027 - val
_loss: 2.1254e-04
Epoch 19/50
43/43 [==============================] - 0s 11ms/step - loss: 0.0028 - val
_loss: 2.1653e-04
Epoch 20/50
43/43 [==============================] - 0s 11ms/step - loss: 0.0026 - val
_loss: 2.1897e-04
Epoch 21/50
43/43 [==============================] - 0s 11ms/step - loss: 0.0026 - val
_loss: 4.5007e-04
Epoch 22/50
43/43 [==============================] - 0s 11ms/step - loss: 0.0028 - val
_loss: 3.7175e-04
12/12 [==============================] - 0s 5ms/step
Next_Close — R²: 0.9221, RMSE: 316.64, MAE: 213.90
```

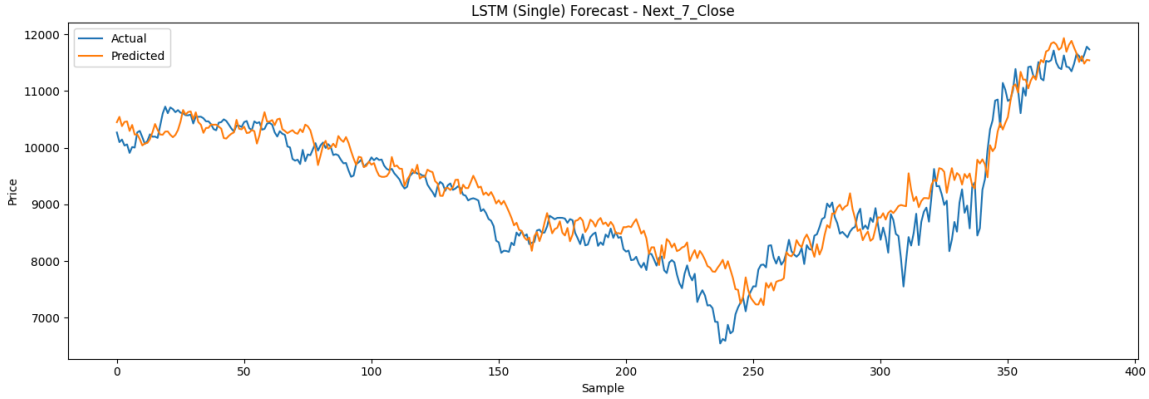


```
🧠 Training LSTM (Single) model for target: Next_3_Close
Epoch 1/50
43/43 [==============================] - 2s 18ms/step - loss: 0.1076 - val
_loss: 0.0496
Epoch 2/50
43/43 [==============================] - 0s 11ms/step - loss: 0.0179 - val
_loss: 0.0018
Epoch 3/50
43/43 [==============================] - 0s 11ms/step - loss: 0.0108 - val
_loss: 0.0025
Epoch 4/50
43/43 [==============================] - 0s 11ms/step - loss: 0.0084 - val
_loss: 0.0019
Epoch 5/50
43/43 [==============================] - 0s 11ms/step - loss: 0.0065 - val
_loss: 6.5805e-04
Epoch 6/50
43/43 [==============================] - 0s 11ms/step - loss: 0.0055 - val
_loss: 0.0013
Epoch 7/50
43/43 [==============================] - 0s 11ms/step - loss: 0.0048 - val
_loss: 8.1104e-04
Epoch 8/50
43/43 [==============================] - 0s 11ms/step - loss: 0.0044 - val
_loss: 0.0034
Epoch 9/50
43/43 [==============================] - 0s 12ms/step - loss: 0.0045 - val
_loss: 4.3273e-04
Epoch 10/50
43/43 [==============================] - 0s 11ms/step - loss: 0.0039 - val
_loss: 4.7779e-04
Epoch 11/50
43/43 [==============================] - 0s 11ms/step - loss: 0.0036 - val
_loss: 4.1743e-04
Epoch 12/50
43/43 [==============================] - 0s 11ms/step - loss: 0.0034 - val
_loss: 3.5359e-04
Epoch 13/50
43/43 [==============================] - 0s 11ms/step - loss: 0.0035 - val
_loss: 3.5066e-04
Epoch 14/50
43/43 [==============================] - 0s 11ms/step - loss: 0.0033 - val
_loss: 4.2102e-04
Epoch 15/50
43/43 [==============================] - 0s 12ms/step - loss: 0.0034 - val
_loss: 3.3037e-04
Epoch 16/50
43/43 [==============================] - 0s 11ms/step - loss: 0.0035 - val
_loss: 4.1485e-04
Epoch 17/50
43/43 [==============================] - 0s 12ms/step - loss: 0.0034 - val
_loss: 3.4768e-04
Epoch 18/50
43/43 [==============================] - 0s 11ms/step - loss: 0.0031 - val
_loss: 6.3116e-04
Epoch 19/50
43/43 [==============================] - 0s 11ms/step - loss: 0.0032 - val
_loss: 9.2941e-04
Epoch 20/50
43/43 [==============================] - 0s 12ms/step - loss: 0.0033 - val
_loss: 4.0185e-04
```

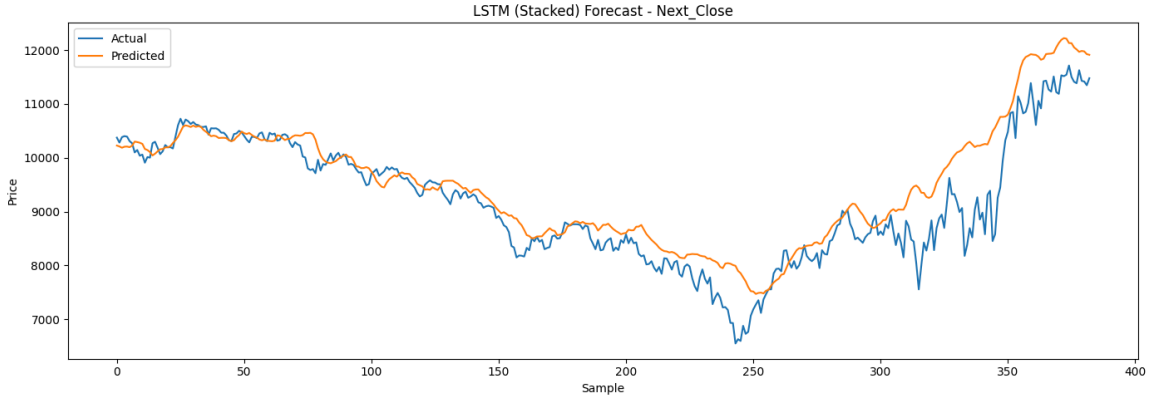
```
12/12 [==============================] - 0s 5ms/step
Next_3_Close — R²: 0.8766, RMSE: 402.36, MAE: 293.02
```



```
🧠 Training LSTM (Single) model for target: Next_7_Close
Epoch 1/50
43/43 [==============================] - 2s 19ms/step - loss: 0.0814 - val
_loss: 0.0166
Epoch 2/50
43/43 [==============================] - 1s 12ms/step - loss: 0.0172 - val
_loss: 0.0017
Epoch 3/50
43/43 [==============================] - 1s 12ms/step - loss: 0.0108 - val
_loss: 0.0015
Epoch 4/50
43/43 [==============================] - 1s 12ms/step - loss: 0.0075 - val
_loss: 0.0013
Epoch 5/50
43/43 [==============================] - 1s 12ms/step - loss: 0.0058 - val
_loss: 0.0019
Epoch 6/50
43/43 [==============================] - 1s 13ms/step - loss: 0.0047 - val
_loss: 6.6199e-04
Epoch 7/50
43/43 [==============================] - 1s 13ms/step - loss: 0.0049 - val
_loss: 6.0754e-04
Epoch 8/50
43/43 [==============================] - 1s 13ms/step - loss: 0.0041 - val
_loss: 6.5198e-04
Epoch 9/50
43/43 [==============================] - 1s 13ms/step - loss: 0.0043 - val
_loss: 5.4967e-04
Epoch 10/50
43/43 [==============================] - 1s 13ms/step - loss: 0.0037 - val
_loss: 5.8011e-04
Epoch 11/50
43/43 [==============================] - 1s 12ms/step - loss: 0.0039 - val
_loss: 6.1603e-04
Epoch 12/50
43/43 [==============================] - 1s 12ms/step - loss: 0.0039 - val
_loss: 5.9047e-04
Epoch 13/50
43/43 [==============================] - 1s 12ms/step - loss: 0.0033 - val
_loss: 5.3133e-04
Epoch 14/50
43/43 [==============================] - 1s 12ms/step - loss: 0.0035 - val
_loss: 4.9485e-04
Epoch 15/50
43/43 [==============================] - 1s 12ms/step - loss: 0.0030 - val
_loss: 5.8692e-04
Epoch 16/50
43/43 [==============================] - 1s 12ms/step - loss: 0.0032 - val
_loss: 6.5944e-04
Epoch 17/50
43/43 [==============================] - 1s 12ms/step - loss: 0.0034 - val
_loss: 4.9606e-04
Epoch 18/50
43/43 [==============================] - 1s 12ms/step - loss: 0.0034 - val
_loss: 0.0019
Epoch 19/50
43/43 [==============================] - 1s 13ms/step - loss: 0.0033 - val
_loss: 4.9761e-04
12/12 [==============================] - 0s 5ms/step
Next_7_Close — R²: 0.8696, RMSE: 421.34, MAE: 319.45
```

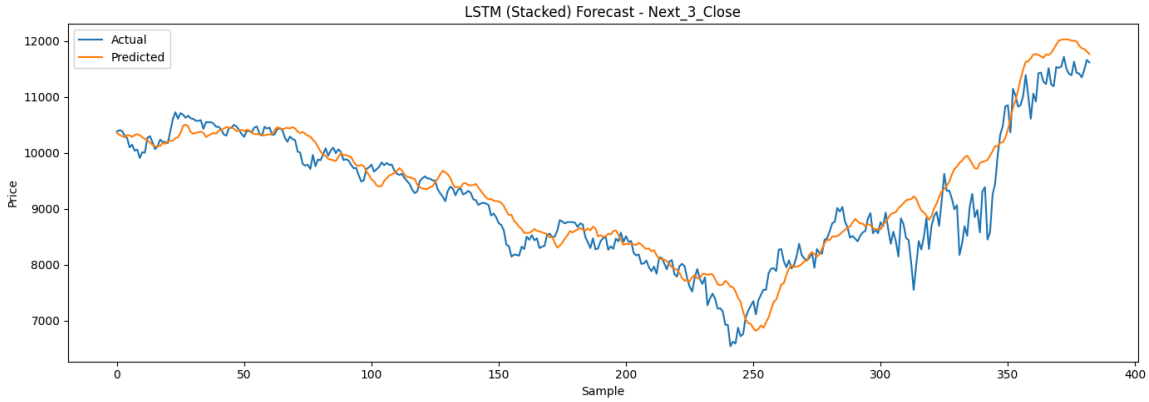

LSTM (Single) Forecast - Next_7_Close
Actual
Predicted
Price
Sample
12000
11000
10000
9000
8000
7000
0
50
100
150
200
250
300
350
400

```
🧠 Training LSTM (Stacked) model for target: Next_Close
Epoch 1/50
43/43 [==============================] - 4s 34ms/step - loss: 0.0344 - val
_loss: 0.0036
Epoch 2/50
43/43 [==============================] - 1s 21ms/step - loss: 0.0104 - val
_loss: 0.0038
Epoch 3/50
43/43 [==============================] - 1s 21ms/step - loss: 0.0088 - val
_loss: 6.4173e-04
Epoch 4/50
43/43 [==============================] - 1s 21ms/step - loss: 0.0076 - val
_loss: 8.3882e-04
Epoch 5/50
43/43 [==============================] - 1s 22ms/step - loss: 0.0069 - val
_loss: 0.0011
Epoch 6/50
43/43 [==============================] - 1s 21ms/step - loss: 0.0071 - val
_loss: 4.6374e-04
Epoch 7/50
43/43 [==============================] - 1s 21ms/step - loss: 0.0062 - val
_loss: 0.0012
Epoch 8/50
43/43 [==============================] - 1s 20ms/step - loss: 0.0063 - val
_loss: 4.9534e-04
Epoch 9/50
43/43 [==============================] - 1s 20ms/step - loss: 0.0052 - val
_loss: 5.0373e-04
Epoch 10/50
43/43 [==============================] - 1s 20ms/step - loss: 0.0052 - val
_loss: 3.4855e-04
Epoch 11/50
43/43 [==============================] - 1s 20ms/step - loss: 0.0053 - val
_loss: 8.8532e-04
Epoch 12/50
43/43 [==============================] - 1s 20ms/step - loss: 0.0053 - val
_loss: 3.1302e-04
Epoch 13/50
43/43 [==============================] - 1s 21ms/step - loss: 0.0048 - val
_loss: 2.9975e-04
Epoch 14/50
43/43 [==============================] - 1s 20ms/step - loss: 0.0048 - val
_loss: 0.0016
Epoch 15/50
43/43 [==============================] - 1s 20ms/step - loss: 0.0049 - val
_loss: 7.9835e-04
Epoch 16/50
43/43 [==============================] - 1s 20ms/step - loss: 0.0046 - val
_loss: 8.4127e-04
Epoch 17/50
43/43 [==============================] - 1s 21ms/step - loss: 0.0043 - val
_loss: 3.6753e-04
Epoch 18/50
43/43 [==============================] - 1s 20ms/step - loss: 0.0042 - val
_loss: 5.9595e-04
12/12 [==============================] - 1s 8ms/step
Next_Close — R²: 0.7810, RMSE: 531.10, MAE: 365.02
```

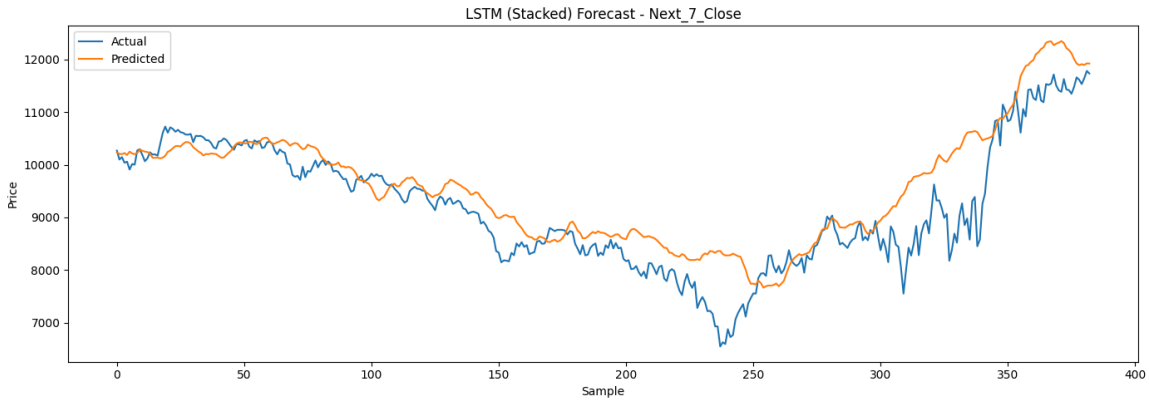

LSTM (Stacked) Forecast - Next_Close
Actual
Predicted
Price
Sample
12000
11000
10000
9000
8000
7000
0
50
100
150
200
250
300
350
400

```
🧠 Training LSTM (Stacked) model for target: Next_3_Close
Epoch 1/50
43/43 [==============================] - 4s 33ms/step - loss: 0.0385 - val
_loss: 7.8072e-04
Epoch 2/50
43/43 [==============================] - 1s 20ms/step - loss: 0.0090 - val
_loss: 0.0050
Epoch 3/50
43/43 [==============================] - 1s 20ms/step - loss: 0.0080 - val
_loss: 9.2754e-04
Epoch 4/50
43/43 [==============================] - 1s 21ms/step - loss: 0.0065 - val
_loss: 7.4764e-04
Epoch 5/50
43/43 [==============================] - 1s 20ms/step - loss: 0.0065 - val
_loss: 0.0031
Epoch 6/50
43/43 [==============================] - 1s 20ms/step - loss: 0.0065 - val
_loss: 0.0020
Epoch 7/50
43/43 [==============================] - 1s 21ms/step - loss: 0.0063 - val
_loss: 6.9790e-04
Epoch 8/50
43/43 [==============================] - 1s 21ms/step - loss: 0.0054 - val
_loss: 9.4436e-04
Epoch 9/50
43/43 [==============================] - 1s 20ms/step - loss: 0.0053 - val
_loss: 6.9543e-04
Epoch 10/50
43/43 [==============================] - 1s 21ms/step - loss: 0.0054 - val
_loss: 0.0014
Epoch 11/50
43/43 [==============================] - 1s 21ms/step - loss: 0.0059 - val
_loss: 0.0021
Epoch 12/50
43/43 [==============================] - 1s 21ms/step - loss: 0.0049 - val
_loss: 5.9410e-04
Epoch 13/50
43/43 [==============================] - 1s 21ms/step - loss: 0.0050 - val
_loss: 8.9455e-04
Epoch 14/50
43/43 [==============================] - 1s 21ms/step - loss: 0.0047 - val
_loss: 5.8762e-04
Epoch 15/50
43/43 [==============================] - 1s 21ms/step - loss: 0.0051 - val
_loss: 0.0016
Epoch 16/50
43/43 [==============================] - 1s 21ms/step - loss: 0.0045 - val
_loss: 5.8555e-04
Epoch 17/50
43/43 [==============================] - 1s 20ms/step - loss: 0.0043 - val
_loss: 5.4983e-04
Epoch 18/50
43/43 [==============================] - 1s 21ms/step - loss: 0.0044 - val
_loss: 6.2556e-04
Epoch 19/50
43/43 [==============================] - 1s 20ms/step - loss: 0.0046 - val
_loss: 7.0855e-04
Epoch 20/50
43/43 [==============================] - 1s 19ms/step - loss: 0.0040 - val
_loss: 0.0011
```

```
Epoch 21/50
43/43 [==============================] - 1s 20ms/step - loss: 0.0040 - val
_loss: 6.1686e-04
Epoch 22/50
43/43 [==============================] - 1s 20ms/step - loss: 0.0038 - val
_loss: 6.7309e-04
12/12 [==============================] - 1s 8ms/step
Next_3_Close — R²: 0.8754, RMSE: 404.30, MAE: 292.04
```



```
🧠 Training LSTM (Stacked) model for target: Next_7_Close
Epoch 1/50
43/43 [==============================] - 4s 33ms/step - loss: 0.0404 - val
_loss: 8.7341e-04
Epoch 2/50
43/43 [==============================] - 1s 21ms/step - loss: 0.0097 - val
_loss: 0.0022
Epoch 3/50
43/43 [==============================] - 1s 22ms/step - loss: 0.0094 - val
_loss: 7.1567e-04
Epoch 4/50
43/43 [==============================] - 1s 21ms/step - loss: 0.0072 - val
_loss: 7.7570e-04
Epoch 5/50
43/43 [==============================] - 1s 21ms/step - loss: 0.0066 - val
_loss: 6.8179e-04
Epoch 6/50
43/43 [==============================] - 1s 22ms/step - loss: 0.0067 - val
_loss: 5.1229e-04
Epoch 7/50
43/43 [==============================] - 1s 21ms/step - loss: 0.0064 - val
_loss: 0.0040
Epoch 8/50
43/43 [==============================] - 1s 21ms/step - loss: 0.0064 - val
_loss: 4.8793e-04
Epoch 9/50
43/43 [==============================] - 1s 21ms/step - loss: 0.0061 - val
_loss: 8.4030e-04
Epoch 10/50
43/43 [==============================] - 1s 20ms/step - loss: 0.0061 - val
_loss: 0.0012
Epoch 11/50
43/43 [==============================] - 1s 20ms/step - loss: 0.0055 - val
_loss: 5.2281e-04
Epoch 12/50
43/43 [==============================] - 1s 21ms/step - loss: 0.0058 - val
_loss: 8.1122e-04
Epoch 13/50
43/43 [==============================] - 1s 20ms/step - loss: 0.0055 - val
_loss: 6.7484e-04
12/12 [==============================] - 1s 8ms/step
Next_7_Close — R²: 0.7175, RMSE: 620.12, MAE: 449.93
```



Temporal Convolutional Networks

In [16]:
```
# def train_tcn_model_for_target(df_tcn, df_targets, target_col, window_siz
e=60, epochs=50, batch_size=32):
#     print(f"\n🧠 Training TCN model for target: {target_col}")

#     # Scale features and target
#     X_scaler = MinMaxScaler()
#     X_scaled = X_scaler.fit_transform(df_tcn.values)

#     y_scaler = MinMaxScaler()
#     y_scaled = y_scaler.fit_transform(df_targets[[target_col]])

#     # Create sequences
#     def create_sequences(X, y, window_size):
#         Xs, ys = [], []
#         for i in range(window_size, len(X)):
#             Xs.append(X[i - window_size:i])
#             ys.append(y[i])
#         return np.array(Xs), np.array(ys)

#     X_seq, y_seq = create_sequences(X_scaled, y_scaled, window_size)

#     # Split
#     split = int(0.8 * len(X_seq))
#     X_train, X_test = X_seq[:split], X_seq[split:]
#     y_train, y_test = y_seq[:split], y_seq[split:]

#     # Model
#     model = Sequential([
#         TCN(
#             input_shape=(X_train.shape[1], X_train.shape[2]),
#             nb_filters=64,
#             kernel_size=3,
#             dilations=[1, 2, 4, 8],
#             return_sequences=False,
#             activation='relu',
#             dropout_rate=0.2
#         ),
#         Dense(64, activation='relu'),
#         Dropout(0.2),
#         Dense(1)
#     ])

#     model.compile(optimizer='adam', loss='mse')
#     early_stop = EarlyStopping(monitor='val_loss', patience=5, restore_be
st_weights=True)

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

#     # Predict
```

```
#     y_pred_scaled = model.predict(X_test)
#     y_pred = y_scaler.inverse_transform(y_pred_scaled)
#     y_true = y_scaler.inverse_transform(y_test)

#     # Eval
#     rmse = np.sqrt(mean_squared_error(y_true, y_pred))
#     r2 = r2_score(y_true, y_pred)
#     print(f"{target_col} — RMSE: {rmse:.2f}, R²: {r2:.4f}")

#     # Plot
#     plt.figure(figsize=(14, 5))
#     plt.plot(y_true, label='Actual')
#     plt.plot(y_pred, label='Predicted')
#     plt.title(f"TCN Forecast - {target_col}")
#     plt.xlabel('Sample')
#     plt.ylabel('Price')
#     plt.legend()
#     plt.tight_layout()
#     plt.show()

#     return model, history, y_true, y_pred

# train_tcn_model_for_target(df_tcn, df_targets, 'Next_Close')
# train_tcn_model_for_target(df_tcn, df_targets, 'Next_3_Close')
# train_tcn_model_for_target(df_tcn, df_targets, 'Next_7_Close')
```

In [17]:
```
selected1_cols = [
    'Close', 'Log_Returns', 'finbert_final_sentiment',
    'total_buying_intent', 'total_selling_intent',
    'total_uncertainty_intent', 'total_urgency_intent'
]
df_tcn_filtered = df_tcn[selected1_cols].copy()
```

In [18]:
```
def train_tcn_logreturn_model(df_tcn_filtered, df_targets, target_col, wind
ow_size=30, epochs=50, batch_size=32):
    print(f"\n🧠 Training TCN model (Log Return Target) for: {target_co
l}")

    # Infer forecast horizon
    if target_col == 'Next_Close':
        shift_n = 1
    elif 'Next_' in target_col and '_Close' in target_col:
        shift_n = int(target_col.replace('Next_', '').replace('_Close',
''))
    else:
        shift_n = 1

    # Compute log returns for target
    close_now = df_targets["Close"].values[:-shift_n]
    close_future = df_targets[target_col].shift(-shift_n).dropna().values
    log_returns = np.log(close_future / close_now)

    # Align features
    df_tcn_filtered = df_tcn_filtered.iloc[:len(log_returns)]

    # Scale features and log returns
    X_scaler = MinMaxScaler()
    X_scaled = X_scaler.fit_transform(df_tcn_filtered)

    y_scaler = MinMaxScaler()
    y_scaled = y_scaler.fit_transform(log_returns.reshape(-1, 1))

    # Create sequences
    def create_sequences(X, y, window_size):
        Xs, ys = [], []
        for i in range(window_size, len(X)):
            Xs.append(X[i - window_size:i])
            ys.append(y[i])
        return np.array(Xs), np.array(ys)

    X_seq, y_seq = create_sequences(X_scaled, y_scaled, window_size)

    # Train-test split
    split = int(0.8 * len(X_seq))
    X_train, X_test = X_seq[:split], X_seq[split:]
    y_train, y_test = y_seq[:split], y_seq[split:]

    # Slice close_now to match y_test (for price reconstruction)
    close_now = close_now[window_size:]  # align with y_seq
    close_now_test = close_now[split:]   # align with y_test

    # Model
    model = Sequential([
        TCN(
            input_shape=(X_train.shape[1], X_train.shape[2]),
            nb_filters=32,
            kernel_size=2,
            dilations=[1, 2, 4],
            padding='causal',
            return_sequences=False,
            activation='relu',
            dropout_rate=0.1
        ),
```

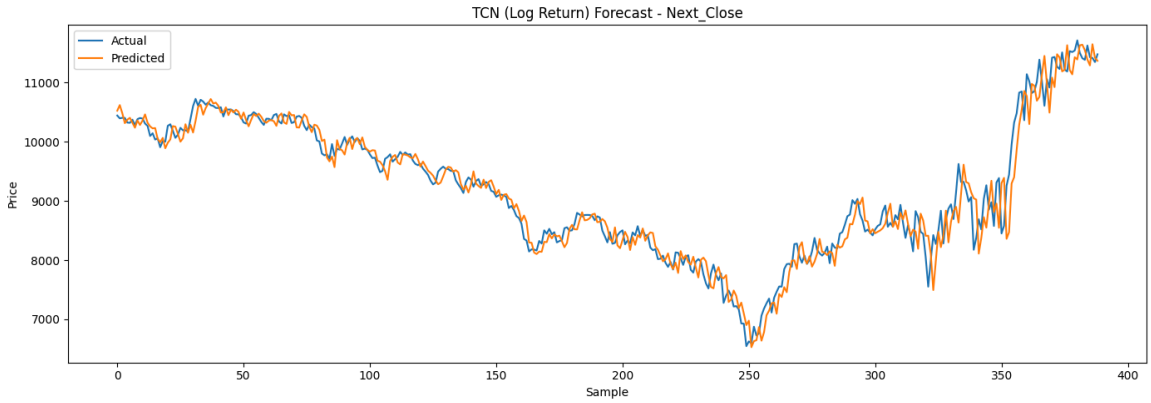
```
        GroupNormalization(groups=4),
        Dense(32, activation='relu'),
        Dropout(0.2),
        Dense(1)
    ])
    early_stop = EarlyStopping(monitor='val_loss', patience=5, restore_best
_weights=True)
    optimizer=Adam(learning_rate=1e-4)
    model.compile(optimizer= optimizer, loss='mse')
    model.fit(
        X_train, y_train,
        validation_split=0.1,
        epochs=epochs,
        batch_size=batch_size,
        verbose=1,
        callbacks=[early_stop]
    )

    # Predict
    y_pred_scaled = model.predict(X_test)
    y_pred_logr = y_scaler.inverse_transform(y_pred_scaled).flatten()
    y_true_logr = y_scaler.inverse_transform(y_test).flatten()

    # Convert log returns to price
    last_known_close = close_now_test
    # Reconstruct prices from log returns
    y_pred_price = last_known_close * np.exp(y_pred_logr)
    y_true_price = last_known_close * np.exp(y_true_logr)

    # Eval
    mae = mean_absolute_error(y_true_price, y_pred_price)
    rmse = np.sqrt(mean_squared_error(y_true_price, y_pred_price))
    r2 = r2_score(y_true_price, y_pred_price)
    print(f"{target_col} — R²: {r2:.4f}, RMSE: {rmse:.2f}, MAE: {mae:.2f}")

    # Plot
    plt.figure(figsize=(14, 5))
    plt.plot(y_true_price, label='Actual')
    plt.plot(y_pred_price, label='Predicted')
    plt.title(f"TCN (Log Return) Forecast - {target_col}")
    plt.xlabel('Sample')
    plt.ylabel('Price')
    plt.legend()
    plt.tight_layout()
    plt.show()

    # Actual vs Predicted
    plt.figure(figsize=(14,6))
    plt.plot(y_true, label='Actual', linewidth=2)
    plt.plot(y_pred, label='TCN Prediction', linewidth=2)
    plt.title(f"TCN Forecast vs Actual — {target_col}", fontsize=16)
    plt.xlabel("Sample", fontsize=12)
    plt.ylabel("Price", fontsize=12)
    plt.legend()
    plt.grid(True)
    plt.tight_layout()
    plt.show()

    # Residuals line plot
    residuals = y_true - y_pred
    plt.figure(figsize=(14,4))
```

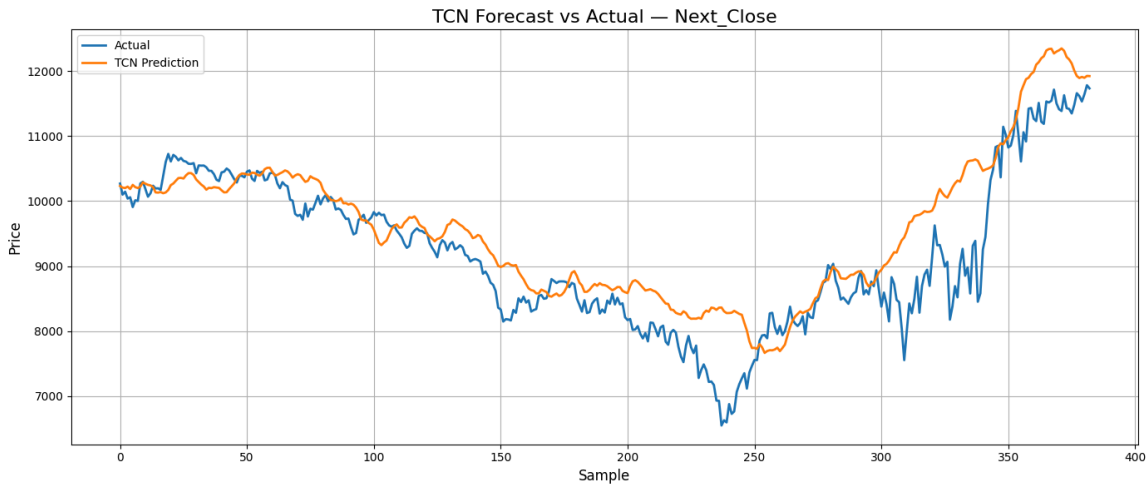
```
    plt.plot(residuals, color='red')
    plt.axhline(0, color='black', linestyle='--')
    plt.title(f"Residuals of TCN Prediction — {target_col}", fontsize=14)
    plt.xlabel("Sample", fontsize=12)
    plt.ylabel("Residual", fontsize=12)
    plt.grid(True)
    plt.tight_layout()
    plt.show()

    # Residuals histogram
    plt.figure(figsize=(8,4))
    sns.histplot(residuals, kde=True, color='purple')
    plt.title(f"Distribution of Residuals — {target_col}", fontsize=14)
    plt.xlabel("Residual")
    plt.tight_layout()
    plt.show()

    return model, y_true_price, y_pred_price


# Example usage:
train_tcn_logreturn_model(df_tcn_filtered, df_targets, 'Next_Close')
train_tcn_logreturn_model(df_tcn_filtered, df_targets, 'Next_3_Close')
train_tcn_logreturn_model(df_tcn_filtered, df_targets, 'Next_7_Close')
```

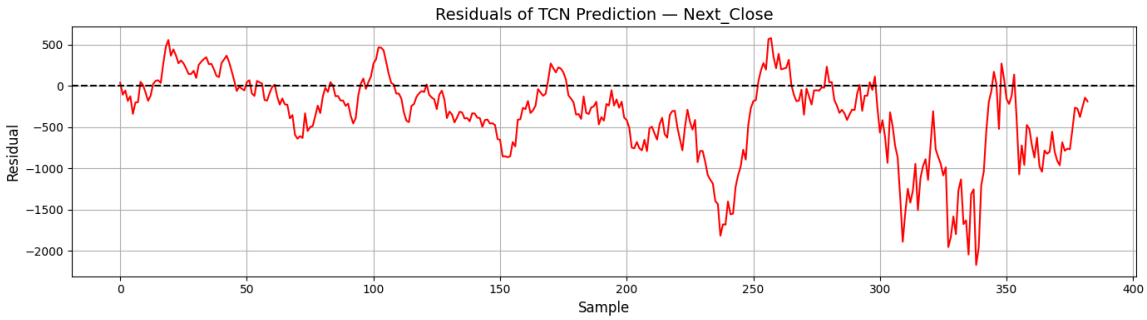
```
🧠 Training TCN model (Log Return Target) for: Next_Close
Epoch 1/50
44/44 [==============================] - 2s 12ms/step - loss: 0.7345 - val
_loss: 0.1670
Epoch 2/50
44/44 [==============================] - 0s 6ms/step - loss: 0.5523 - val_
loss: 0.0998
Epoch 3/50
44/44 [==============================] - 0s 6ms/step - loss: 0.4070 - val_
loss: 0.0605
Epoch 4/50
44/44 [==============================] - 0s 6ms/step - loss: 0.2974 - val_
loss: 0.0381
Epoch 5/50
44/44 [==============================] - 0s 6ms/step - loss: 0.2449 - val_
loss: 0.0325
Epoch 6/50
44/44 [==============================] - 0s 6ms/step - loss: 0.2010 - val_
loss: 0.0322
Epoch 7/50
44/44 [==============================] - 0s 6ms/step - loss: 0.1545 - val_
loss: 0.0277
Epoch 8/50
44/44 [==============================] - 0s 6ms/step - loss: 0.1213 - val_
loss: 0.0253
Epoch 9/50
44/44 [==============================] - 0s 6ms/step - loss: 0.1064 - val_
loss: 0.0200
Epoch 10/50
44/44 [==============================] - 0s 6ms/step - loss: 0.0912 - val_
loss: 0.0188
Epoch 11/50
44/44 [==============================] - 0s 6ms/step - loss: 0.0799 - val_
loss: 0.0160
Epoch 12/50
44/44 [==============================] - 0s 6ms/step - loss: 0.0715 - val_
loss: 0.0139
Epoch 13/50
44/44 [==============================] - 0s 6ms/step - loss: 0.0603 - val_
loss: 0.0119
Epoch 14/50
44/44 [==============================] - 0s 6ms/step - loss: 0.0540 - val_
loss: 0.0106
Epoch 15/50
44/44 [==============================] - 0s 6ms/step - loss: 0.0522 - val_
loss: 0.0097
Epoch 16/50
44/44 [==============================] - 0s 6ms/step - loss: 0.0478 - val_
loss: 0.0093
Epoch 17/50
44/44 [==============================] - 0s 6ms/step - loss: 0.0387 - val_
loss: 0.0085
Epoch 18/50
44/44 [==============================] - 0s 6ms/step - loss: 0.0384 - val_
loss: 0.0080
Epoch 19/50
44/44 [==============================] - 0s 6ms/step - loss: 0.0365 - val_
loss: 0.0081
Epoch 20/50
44/44 [==============================] - 0s 6ms/step - loss: 0.0361 - val_
loss: 0.0073
```

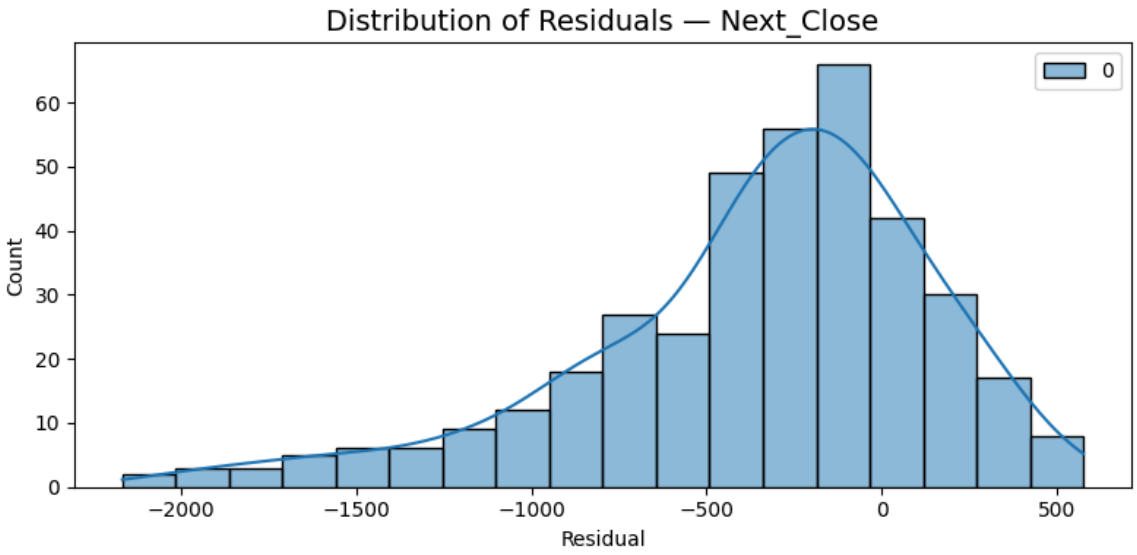
```
Epoch 21/50
44/44 [==============================] - 0s 6ms/step - loss: 0.0309 - val_
loss: 0.0074
Epoch 22/50
44/44 [==============================] - 0s 6ms/step - loss: 0.0312 - val_
loss: 0.0074
Epoch 23/50
44/44 [==============================] - 0s 6ms/step - loss: 0.0322 - val_
loss: 0.0073
Epoch 24/50
44/44 [==============================] - 0s 6ms/step - loss: 0.0309 - val_
loss: 0.0072
Epoch 25/50
44/44 [==============================] - 0s 6ms/step - loss: 0.0268 - val_
loss: 0.0069
Epoch 26/50
44/44 [==============================] - 0s 6ms/step - loss: 0.0288 - val_
loss: 0.0063
Epoch 27/50
44/44 [==============================] - 0s 6ms/step - loss: 0.0252 - val_
loss: 0.0063
Epoch 28/50
44/44 [==============================] - 0s 6ms/step - loss: 0.0257 - val_
loss: 0.0066
Epoch 29/50
44/44 [==============================] - 0s 6ms/step - loss: 0.0256 - val_
loss: 0.0064
Epoch 30/50
44/44 [==============================] - 0s 7ms/step - loss: 0.0256 - val_
loss: 0.0062
Epoch 31/50
44/44 [==============================] - 0s 8ms/step - loss: 0.0235 - val_
loss: 0.0061
Epoch 32/50
44/44 [==============================] - 0s 7ms/step - loss: 0.0229 - val_
loss: 0.0061
Epoch 33/50
44/44 [==============================] - 0s 8ms/step - loss: 0.0214 - val_
loss: 0.0059
Epoch 34/50
44/44 [==============================] - 0s 7ms/step - loss: 0.0221 - val_
loss: 0.0060
Epoch 35/50
44/44 [==============================] - 0s 7ms/step - loss: 0.0217 - val_
loss: 0.0057
Epoch 36/50
44/44 [==============================] - 0s 6ms/step - loss: 0.0205 - val_
loss: 0.0062
Epoch 37/50
44/44 [==============================] - 0s 6ms/step - loss: 0.0207 - val_
loss: 0.0055
Epoch 38/50
44/44 [==============================] - 0s 6ms/step - loss: 0.0192 - val_
loss: 0.0056
Epoch 39/50
44/44 [==============================] - 0s 7ms/step - loss: 0.0191 - val_
loss: 0.0055
Epoch 40/50
44/44 [==============================] - 0s 6ms/step - loss: 0.0165 - val_
loss: 0.0055
Epoch 41/50
```

```
44/44 [==============================] - 0s 6ms/step - loss: 0.0169 - val_
loss: 0.0052
Epoch 42/50
44/44 [==============================] - 0s 6ms/step - loss: 0.0175 - val_
loss: 0.0052
Epoch 43/50
44/44 [==============================] - 0s 6ms/step - loss: 0.0177 - val_
loss: 0.0052
Epoch 44/50
44/44 [==============================] - 0s 6ms/step - loss: 0.0163 - val_
loss: 0.0051
Epoch 45/50
44/44 [==============================] - 0s 6ms/step - loss: 0.0170 - val_
loss: 0.0050
Epoch 46/50
44/44 [==============================] - 0s 6ms/step - loss: 0.0174 - val_
loss: 0.0050
Epoch 47/50
44/44 [==============================] - 0s 6ms/step - loss: 0.0155 - val_
loss: 0.0049
Epoch 48/50
44/44 [==============================] - 0s 6ms/step - loss: 0.0158 - val_
loss: 0.0049
Epoch 49/50
44/44 [==============================] - 0s 6ms/step - loss: 0.0143 - val_
loss: 0.0049
Epoch 50/50
44/44 [==============================] - 0s 7ms/step - loss: 0.0143 - val_
loss: 0.0048
13/13 [==============================] - 0s 2ms/step
Next_Close — R²: 0.9477, RMSE: 259.74, MAE: 185.69
```





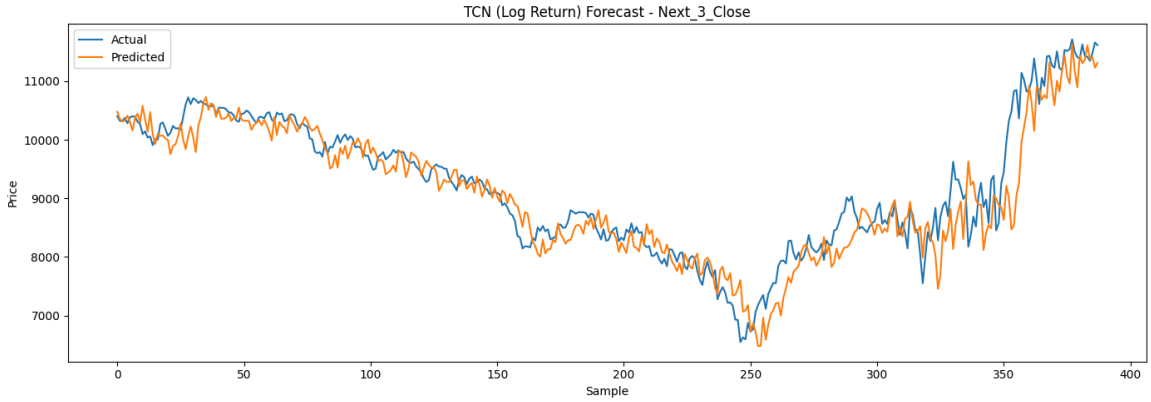

Residuals of TCN Prediction — Next_Close
Residual
Sample
500
0
−500
−1000
−1500
−2000
0
50
100
150
200
250
300
350
400

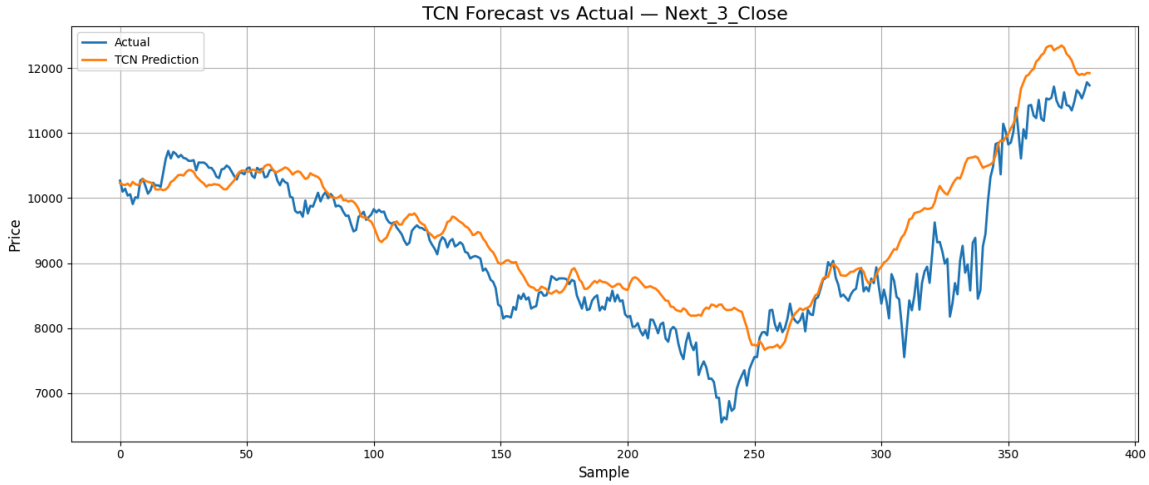

Distribution of Residuals — Next_Close
0
Count
Residual
60
50
40
30
20
10
0
−2000
−1500
−1000
−500
0
500

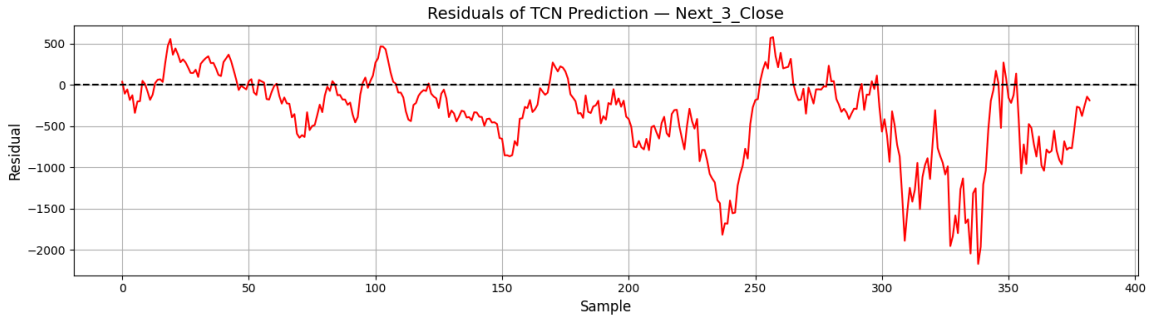
```
🧠 Training TCN model (Log Return Target) for: Next_3_Close
Epoch 1/50
44/44 [==============================] - 2s 12ms/step - loss: 0.4350 - val
_loss: 0.1438
Epoch 2/50
44/44 [==============================] - 0s 6ms/step - loss: 0.2539 - val_
loss: 0.0654
Epoch 3/50
44/44 [==============================] - 0s 6ms/step - loss: 0.1719 - val_
loss: 0.0446
Epoch 4/50
44/44 [==============================] - 0s 6ms/step - loss: 0.1356 - val_
loss: 0.0347
Epoch 5/50
44/44 [==============================] - 0s 7ms/step - loss: 0.1188 - val_
loss: 0.0251
Epoch 6/50
44/44 [==============================] - 0s 6ms/step - loss: 0.1077 - val_
loss: 0.0192
Epoch 7/50
44/44 [==============================] - 0s 6ms/step - loss: 0.0948 - val_
loss: 0.0158
Epoch 8/50
44/44 [==============================] - 0s 6ms/step - loss: 0.0722 - val_
loss: 0.0144
Epoch 9/50
44/44 [==============================] - 0s 6ms/step - loss: 0.0638 - val_
loss: 0.0128
Epoch 10/50
44/44 [==============================] - 0s 6ms/step - loss: 0.0632 - val_
loss: 0.0114
Epoch 11/50
44/44 [==============================] - 0s 6ms/step - loss: 0.0514 - val_
loss: 0.0104
Epoch 12/50
44/44 [==============================] - 0s 6ms/step - loss: 0.0461 - val_
loss: 0.0099
Epoch 13/50
44/44 [==============================] - 0s 7ms/step - loss: 0.0426 - val_
loss: 0.0094
Epoch 14/50
44/44 [==============================] - 0s 6ms/step - loss: 0.0407 - val_
loss: 0.0091
Epoch 15/50
44/44 [==============================] - 0s 6ms/step - loss: 0.0400 - val_
loss: 0.0085
Epoch 16/50
44/44 [==============================] - 0s 6ms/step - loss: 0.0367 - val_
loss: 0.0089
Epoch 17/50
44/44 [==============================] - 0s 6ms/step - loss: 0.0342 - val_
loss: 0.0086
Epoch 18/50
44/44 [==============================] - 0s 6ms/step - loss: 0.0292 - val_
loss: 0.0085
Epoch 19/50
44/44 [==============================] - 0s 7ms/step - loss: 0.0283 - val_
loss: 0.0082
Epoch 20/50
44/44 [==============================] - 0s 7ms/step - loss: 0.0241 - val_
loss: 0.0078
```

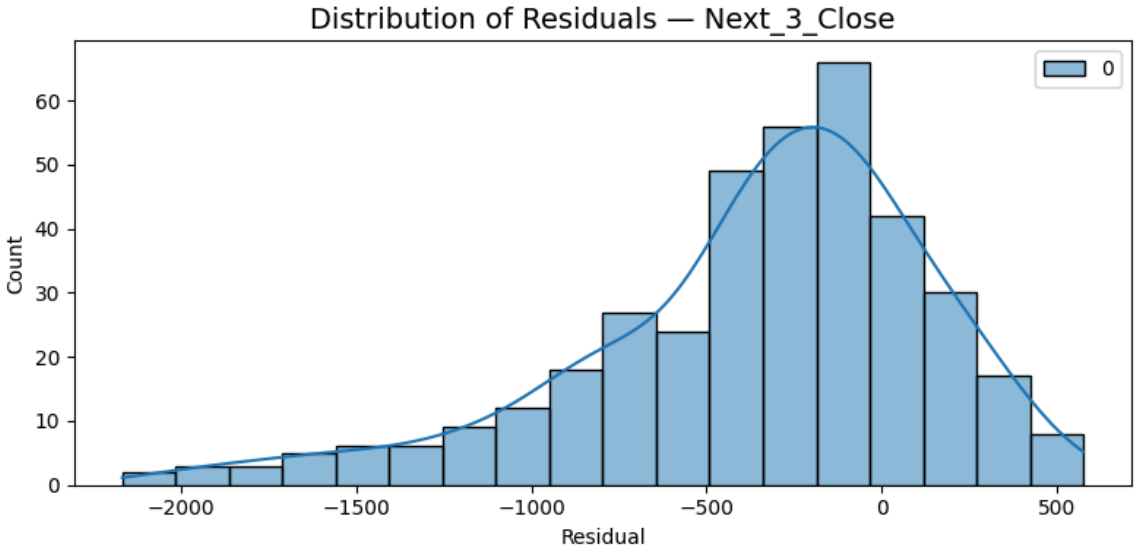
```
Epoch 21/50
44/44 [==============================] - 0s 7ms/step - loss: 0.0263 - val_
loss: 0.0082
Epoch 22/50
44/44 [==============================] - 0s 6ms/step - loss: 0.0235 - val_
loss: 0.0078
Epoch 23/50
44/44 [==============================] - 0s 6ms/step - loss: 0.0221 - val_
loss: 0.0073
Epoch 24/50
44/44 [==============================] - 0s 6ms/step - loss: 0.0219 - val_
loss: 0.0076
Epoch 25/50
44/44 [==============================] - 0s 6ms/step - loss: 0.0215 - val_
loss: 0.0078
Epoch 26/50
44/44 [==============================] - 0s 7ms/step - loss: 0.0202 - val_
loss: 0.0072
Epoch 27/50
44/44 [==============================] - 0s 7ms/step - loss: 0.0217 - val_
loss: 0.0074
Epoch 28/50
44/44 [==============================] - 0s 7ms/step - loss: 0.0189 - val_
loss: 0.0071
Epoch 29/50
44/44 [==============================] - 0s 7ms/step - loss: 0.0180 - val_
loss: 0.0071
Epoch 30/50
44/44 [==============================] - 0s 7ms/step - loss: 0.0165 - val_
loss: 0.0068
Epoch 31/50
44/44 [==============================] - 0s 7ms/step - loss: 0.0161 - val_
loss: 0.0068
Epoch 32/50
44/44 [==============================] - 0s 6ms/step - loss: 0.0163 - val_
loss: 0.0069
Epoch 33/50
44/44 [==============================] - 0s 6ms/step - loss: 0.0162 - val_
loss: 0.0070
Epoch 34/50
44/44 [==============================] - 0s 7ms/step - loss: 0.0155 - val_
loss: 0.0071
Epoch 35/50
44/44 [==============================] - 0s 6ms/step - loss: 0.0158 - val_
loss: 0.0066
Epoch 36/50
44/44 [==============================] - 0s 7ms/step - loss: 0.0146 - val_
loss: 0.0062
Epoch 37/50
44/44 [==============================] - 0s 6ms/step - loss: 0.0143 - val_
loss: 0.0061
Epoch 38/50
44/44 [==============================] - 0s 7ms/step - loss: 0.0139 - val_
loss: 0.0067
Epoch 39/50
44/44 [==============================] - 0s 7ms/step - loss: 0.0139 - val_
loss: 0.0066
Epoch 40/50
44/44 [==============================] - 0s 7ms/step - loss: 0.0129 - val_
loss: 0.0063
Epoch 41/50
```

```
44/44 [==============================] - 0s 7ms/step - loss: 0.0130 - val_
loss: 0.0062
Epoch 42/50
44/44 [==============================] - 0s 7ms/step - loss: 0.0132 - val_
loss: 0.0063
13/13 [==============================] - 0s 2ms/step
Next_3_Close — R²: 0.8721, RMSE: 409.53, MAE: 292.25
```





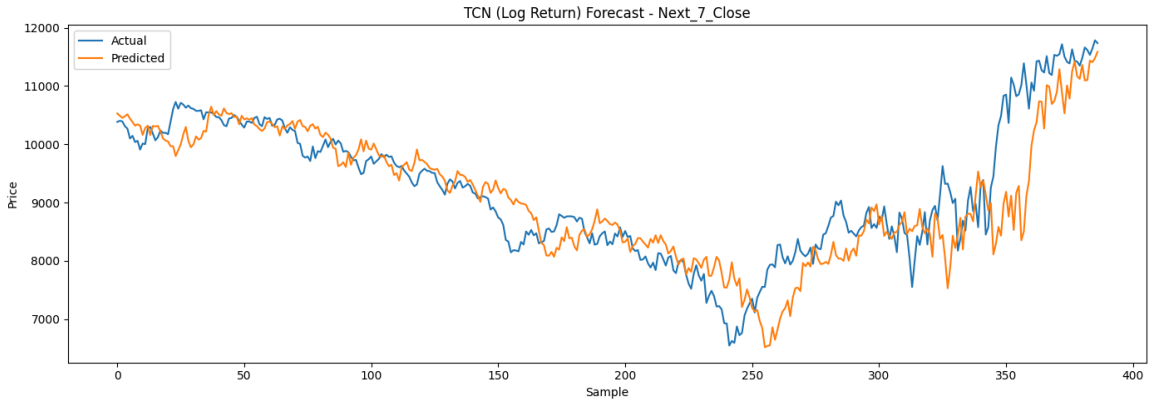


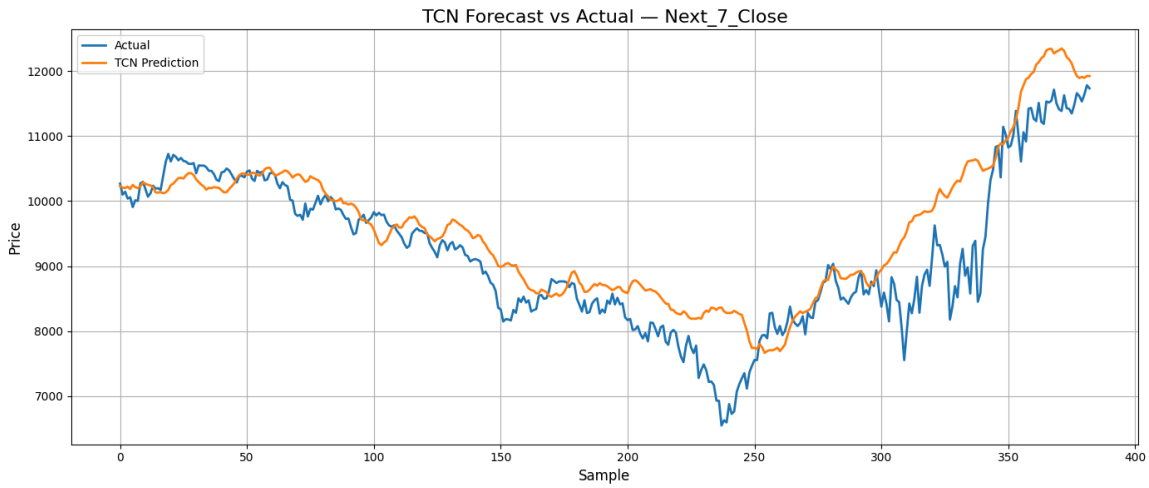

Distribution of Residuals — Next_3_Close
0
Count
Residual
0
10
20
30
40
50
60
−2000
−1500
−1000
−500
0
500

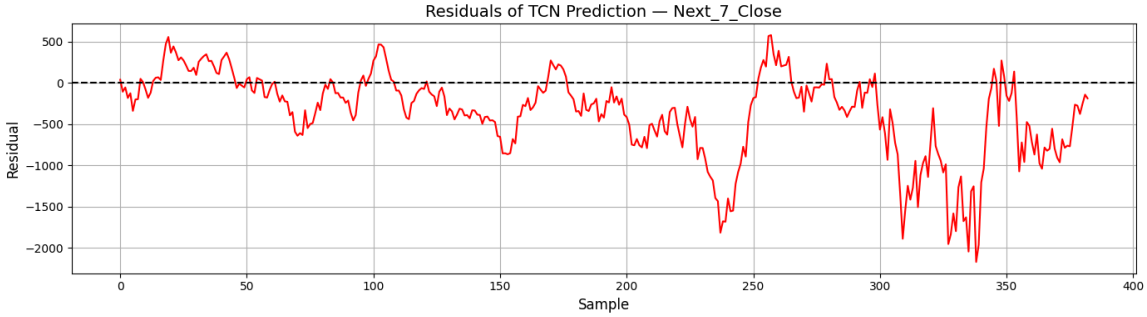
```
🧠 Training TCN model (Log Return Target) for: Next_7_Close
Epoch 1/50
44/44 [==============================] - 3s 13ms/step - loss: 0.4223 - val
_loss: 0.0570
Epoch 2/50
44/44 [==============================] - 0s 6ms/step - loss: 0.2719 - val_
loss: 0.0370
Epoch 3/50
44/44 [==============================] - 0s 7ms/step - loss: 0.2039 - val_
loss: 0.0274
Epoch 4/50
44/44 [==============================] - 0s 6ms/step - loss: 0.1611 - val_
loss: 0.0230
Epoch 5/50
44/44 [==============================] - 0s 6ms/step - loss: 0.1216 - val_
loss: 0.0175
Epoch 6/50
44/44 [==============================] - 0s 7ms/step - loss: 0.0990 - val_
loss: 0.0146
Epoch 7/50
44/44 [==============================] - 0s 7ms/step - loss: 0.0844 - val_
loss: 0.0124
Epoch 8/50
44/44 [==============================] - 0s 6ms/step - loss: 0.0699 - val_
loss: 0.0112
Epoch 9/50
44/44 [==============================] - 0s 7ms/step - loss: 0.0621 - val_
loss: 0.0100
Epoch 10/50
44/44 [==============================] - 0s 6ms/step - loss: 0.0634 - val_
loss: 0.0094
Epoch 11/50
44/44 [==============================] - 0s 7ms/step - loss: 0.0499 - val_
loss: 0.0089
Epoch 12/50
44/44 [==============================] - 0s 7ms/step - loss: 0.0443 - val_
loss: 0.0086
Epoch 13/50
44/44 [==============================] - 0s 6ms/step - loss: 0.0432 - val_
loss: 0.0085
Epoch 14/50
44/44 [==============================] - 0s 6ms/step - loss: 0.0378 - val_
loss: 0.0085
Epoch 15/50
44/44 [==============================] - 0s 7ms/step - loss: 0.0352 - val_
loss: 0.0083
Epoch 16/50
44/44 [==============================] - 0s 7ms/step - loss: 0.0334 - val_
loss: 0.0083
Epoch 17/50
44/44 [==============================] - 0s 7ms/step - loss: 0.0302 - val_
loss: 0.0082
Epoch 18/50
44/44 [==============================] - 0s 7ms/step - loss: 0.0281 - val_
loss: 0.0081
Epoch 19/50
44/44 [==============================] - 0s 6ms/step - loss: 0.0278 - val_
loss: 0.0082
Epoch 20/50
44/44 [==============================] - 0s 7ms/step - loss: 0.0284 - val_
loss: 0.0083
```

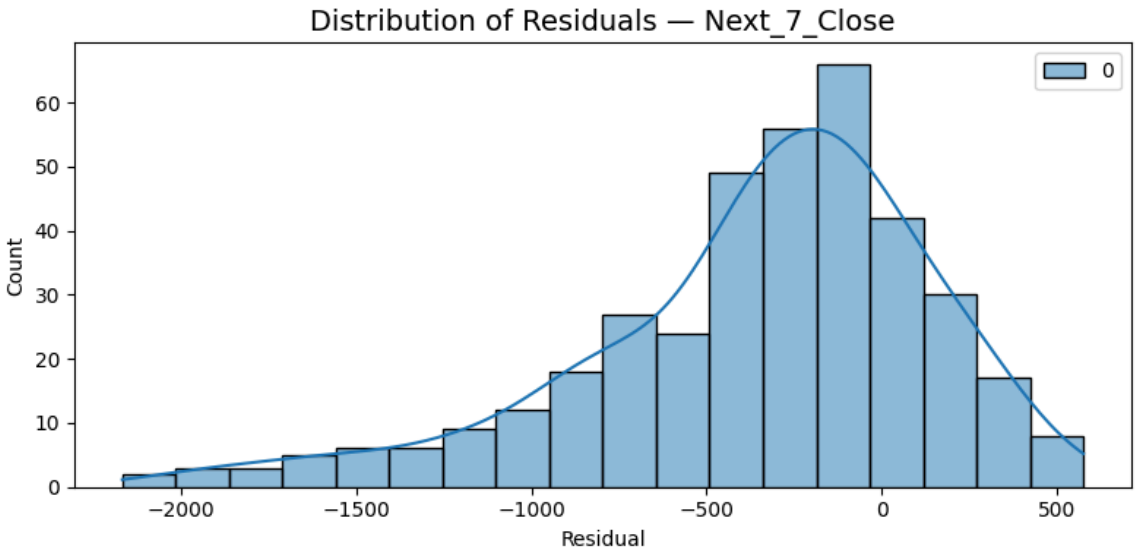
```
Epoch 21/50
44/44 [==============================] - 0s 6ms/step - loss: 0.0225 - val_
loss: 0.0081
Epoch 22/50
44/44 [==============================] - 0s 6ms/step - loss: 0.0225 - val_
loss: 0.0081
Epoch 23/50
44/44 [==============================] - 0s 7ms/step - loss: 0.0243 - val_
loss: 0.0080
Epoch 24/50
44/44 [==============================] - 0s 7ms/step - loss: 0.0215 - val_
loss: 0.0078
Epoch 25/50
44/44 [==============================] - 0s 7ms/step - loss: 0.0194 - val_
loss: 0.0077
Epoch 26/50
44/44 [==============================] - 0s 7ms/step - loss: 0.0180 - val_
loss: 0.0077
Epoch 27/50
44/44 [==============================] - 0s 7ms/step - loss: 0.0190 - val_
loss: 0.0077
Epoch 28/50
44/44 [==============================] - 0s 6ms/step - loss: 0.0179 - val_
loss: 0.0077
Epoch 29/50
44/44 [==============================] - 0s 6ms/step - loss: 0.0162 - val_
loss: 0.0077
Epoch 30/50
44/44 [==============================] - 0s 6ms/step - loss: 0.0147 - val_
loss: 0.0077
Epoch 31/50
44/44 [==============================] - 0s 7ms/step - loss: 0.0159 - val_
loss: 0.0076
Epoch 32/50
44/44 [==============================] - 0s 7ms/step - loss: 0.0140 - val_
loss: 0.0077
Epoch 33/50
44/44 [==============================] - 0s 7ms/step - loss: 0.0135 - val_
loss: 0.0076
Epoch 34/50
44/44 [==============================] - 0s 7ms/step - loss: 0.0142 - val_
loss: 0.0076
Epoch 35/50
44/44 [==============================] - 0s 7ms/step - loss: 0.0138 - val_
loss: 0.0076
Epoch 36/50
44/44 [==============================] - 0s 7ms/step - loss: 0.0137 - val_
loss: 0.0075
Epoch 37/50
44/44 [==============================] - 0s 6ms/step - loss: 0.0134 - val_
loss: 0.0075
Epoch 38/50
44/44 [==============================] - 0s 7ms/step - loss: 0.0127 - val_
loss: 0.0075
Epoch 39/50
44/44 [==============================] - 0s 7ms/step - loss: 0.0123 - val_
loss: 0.0074
Epoch 40/50
44/44 [==============================] - 0s 6ms/step - loss: 0.0121 - val_
loss: 0.0074
Epoch 41/50
```

```
44/44 [==============================] - 0s 7ms/step - loss: 0.0119 - val_
loss: 0.0073
Epoch 42/50
44/44 [==============================] - 0s 7ms/step - loss: 0.0113 - val_
loss: 0.0073
Epoch 43/50
44/44 [==============================] - 0s 7ms/step - loss: 0.0114 - val_
loss: 0.0073
Epoch 44/50
44/44 [==============================] - 0s 7ms/step - loss: 0.0116 - val_
loss: 0.0072
Epoch 45/50
44/44 [==============================] - 0s 7ms/step - loss: 0.0114 - val_
loss: 0.0072
Epoch 46/50
44/44 [==============================] - 0s 6ms/step - loss: 0.0113 - val_
loss: 0.0073
Epoch 47/50
44/44 [==============================] - 0s 6ms/step - loss: 0.0109 - val_
loss: 0.0072
Epoch 48/50
44/44 [==============================] - 0s 6ms/step - loss: 0.0102 - val_
loss: 0.0072
Epoch 49/50
44/44 [==============================] - 0s 7ms/step - loss: 0.0110 - val_
loss: 0.0072
Epoch 50/50
44/44 [==============================] - 0s 6ms/step - loss: 0.0096 - val_
loss: 0.0071
13/13 [==============================] - 0s 2ms/step
Next_7_Close — R²: 0.7373, RMSE: 597.92, MAE: 413.07
```






Residuals of TCN Prediction — Next_7_Close
Residual
Sample
500
0
−500
−1000
−1500
−2000
0
50
100
150
200
250
300
350
400


Distribution of Residuals — Next_7_Close
Count
Residual
0
60
50
40
30
20
10
0
−2000
−1500
−1000
−500
0
500

Out[18]: (<keras.src.engine.sequential.Sequential at 0x2c1c48fa5d0>,
array([10383.379883, 10402.349609, 10392.900391, 10309.240234,
10268.80957 , 10099.139648, 10144.19043 , 10038.379883,
10058.639648, 9908.389648, 10012.230469, 10002.179688,
10270.549805, 10296.849609, 10185.530273, 10067.330078,
10120.459961, 10236.160156, 10194.290039, 10196.860352,
10172.980469, 10389.879883, 10603.150391, 10725.429688,
10609.650391, 10710.549805, 10680.769531, 10627.259766,
10663.990234, 10618.19043 , 10606.860352, 10573.679688,
10572.019531, 10583.959961, 10428.049805, 10548.509766,
10545.410156, 10547.080078, 10520.089844, 10466.44043 ,
10464.929688, 10414.139648, 10328.889648, 10308.259766,
10441.120117, 10452. , 10501.049805, 10471.5 ,
10405.830078, 10337.049805, 10285.969727, 10390.110352,
10388.900391, 10366.150391, 10452.679688, 10471.580078,
10344.839844, 10309.919922, 10464.400391, 10433.709961,
10450.950195, 10318.160156, 10332.44043 , 10426.30957 ,
10437.419922, 10406.959961, 10270.469727, 10197.469727,
10291.259766, 10246.969727, 10226.94043 , 10023.419922,
10005.959961, 9802.139648, 9771.910156, 9789.44043 ,
9712.730469, 9962.580078, 9762.69043 , 9882.169922,
9867.959961, 9972.179688, 10081.30957 , 9949.360352,
10041.480469, 10092.19043 , 9995.910156, 10062.94043 ,
10015.860352, 9871.05957 , 9885.799805, 9864.94043 ,
9786.870117, 9725.580078, 9731.25 , 9599.75 ,
9487.669922, 9509.280273, 9712.280273, 9742.200195,
9789.360352, 9665.19043 , 9707.44043 , 9748.549805,
9829.870117, 9778.860352, 9820.200195, 9783.919922,
9791.709961, 9683.410156, 9626.799805, 9605.410156,
9627.480469, 9547.219727, 9497.339844, 9441.269531,
9344.610352, 9280.669922, 9310.599609, 9496.280273,
9544.200195, 9580.629883, 9543.519531, 9539.290039,
9509.280273, 9505.959961, 9350.049805, 9279.160156,
9217.94043 , 9135.339844, 9321.400391, 9398.19043 ,
9361.610352, 9241.450195, 9337.950195, 9370.070312,
9256.259766, 9280.969727, 9320.19043 , 9286.55957 ,
9171.610352, 9154.459961, 9070.719727, 9096.719727,
9108.509766, 9093.240234, 9069.290039, 8881.259766,
8915.94043 , 8848.150391, 8743.94043 , 8711.820312,
8616.209961, 8359.490234, 8331.679688, 8146.52002 ,
8183.169922, 8178.410156, 8163.600098, 8324.870117,
8280.740234, 8504.05957 , 8447. , 8529.379883,
8438.389648, 8472.400391, 8299.860352, 8322.910156,
8339.009766, 8539.730469, 8555.599609, 8497.179688,
8504.669922, 8612.129883, 8799.259766, 8770.919922,
8739.019531, 8763.05957 , 8764.490234, 8763.129883,
8750.240234, 8675.240234, 8740.870117, 8721.44043 ,
8500.330078, 8403.799805, 8300.019531, 8473.490234,
8277.320312, 8292.129883, 8422.040039, 8474.849609,
8504.080078, 8268.639648, 8331.320312, 8284.889648,
8469.110352, 8418.769531, 8574.650391, 8409.849609,
8512.280273, 8410.650391, 8426.740234, 8212.410156,
8168.120117, 8185.72998 , 8016.950195, 8025. ,
8076.290039, 7957.060059, 7886.569824, 7969.560059,
7841.72998 , 8131.330078, 8125.430176, 8029.620117,
7920.180176, 8057.810059, 8083.379883, 7837.109863,
7789.560059, 7975.850098, 8017.589844, 7978.080078,
7761.600098, 7608.919922, 7522.02002 , 7776.180176,
7924.560059, 7749.810059, 7659.970215, 7775.859863,
7278.379883, 7400.799805, 7486.580078, 7395.700195,
7216.970215, 7223.97998 , 7170.060059, 6930.399902,

```
        6926.490234,  6547.049805,  6626.939941,  6594.439941,
        6875.839844,  6726.02002 ,  6763.290039,  7062.930176,
        7182.080078,  7270.890137,  7350.939941,  7114.779785,
        7365.669922,  7465.950195,  7555.629883,  7552.600098,
        7850.410156,  7932.759766,  7939.529785,  7888.879883,
        8270.870117,  8280.589844,  8063.069824,  7956.660156,
        8078.359863,  7936.75    ,  8000.859863,  8149.009766,
        8375.450195,  8174.72998 ,  8116.029785,  8077.560059,
        8122.799805,  8228.099609,  7949.089844,  8281.219727,
        8212.490234,  8200.139648,  8448.55957 ,  8473.969727,
        8599.179688,  8742.459961,  8769.700195,  9015.099609,
        8952.889648,  9034.69043 ,  8776.389648,  8668.389648,
        8483.929688,  8515.549805,  8468.480469,  8419.490234,
        8519.69043 ,  8579.110352,  8604.990234,  8824.339844,
        8924.139648,  8564.530273,  8629.679688,  8565.089844,
        8761.419922,  8691.330078,  8934.179688,  8635.419922,
        8376.240234,  8591.69043 ,  8419.089844,  8149.089844,
        8829.040039,  8726.610352,  8479.469727,  8443.389648,
        8046.419922,  7552.290039,  7997.279785,  8424.75    ,
        8273.580078,  8497.30957 ,  8835.25    ,  8282.660156,
        8693.959961,  8870.540039,  8943.80957 ,  8695.790039,
        9139.269531,  9625.280273,  9319.830078,  9325.009766,
        9180.69043 ,  8990.959961,  9065.120117,  8175.77002 ,
        8378.950195,  8691.25    ,  8519.209961,  9033.660156,
        9265.429688,  8852.219727,  8979.259766,  8577.910156,
        9310.990234,  9387.610352,  8451.19043 ,  8579.19043 ,
        9258.099609,  9447.110352,  9955.5     , 10325.379883,
       10482.849609, 10831.070312, 10850.660156, 10365.450195,
       11143.129883, 11022.05957 , 10825.169922, 10854.169922,
       11015.69043 , 11388.44043 , 11019.69043 , 10609.660156,
       11059.019531, 10917.509766, 11421.990234, 11433.709961,
       11268.919922, 11230.730469, 11510.740234, 11220.959961,
       11188.230469, 11532.879883, 11516.919922, 11543.959961,
       11715.179688, 11502.509766, 11412.870117, 11386.25    ,
       11628.05957 , 11430.209961, 11417.429688, 11348.549805,
       11479.389648, 11659.900391, 11615.929688, 11532.959961,
       11642.469727, 11782.349609, 11734.320312]),
array([10528.30887687, 10489.18634109, 10451.97904905, 10480.84950174,
       10513.27752008, 10440.51838278, 10382.35461405, 10318.9663325 ,
       10342.48042975, 10320.10748676, 10160.9193435 , 10278.71133322,
       10319.20446586, 10156.65281805, 10313.32871248, 10305.82111485,
       10313.86717055, 10222.84331359, 10101.14741443, 10065.34465495,
       10045.94112622,  9966.68749509,  9966.56204245,  9797.76461352,
        9899.48287857,  9997.23738592, 10170.47246132, 10296.96008221,
       10058.78740857,  9947.93261244,  9997.2732839 , 10134.28736033,
       10079.93037734, 10105.26969331, 10230.43035863, 10217.64864607,
       10507.20313254, 10644.5792601 , 10501.30522919, 10568.35802663,
       10512.70157053, 10488.30777396, 10614.82985288, 10538.04839584,
       10515.51937129, 10533.82832565, 10466.92705742, 10478.2649277 ,
       10342.34908403, 10488.65621606, 10425.27121458, 10444.22206782,
       10421.82355001, 10448.03315722, 10338.18197774, 10311.1994655 ,
       10260.29210051, 10228.13937693, 10270.30603917, 10380.55265021,
       10387.88691185, 10350.60276183, 10292.79706435, 10307.60342461,
       10149.55378683, 10305.05894626, 10264.15345342, 10324.47624723,
       10346.55741543, 10397.68514164, 10264.70089307, 10389.74103884,
       10415.3180763 , 10311.00459409, 10297.24465418, 10222.08263648,
       10317.93689305, 10343.35693783, 10269.68925322, 10299.48313543,
       10160.70989494, 10122.93865251, 10198.5979089 , 10163.40275504,
       10098.54542753,  9940.06416316,  9922.1011123 ,  9624.55593292,
        9644.61543889,  9692.32450288,  9609.99290591,  9857.79945513,
        9648.47942049,  9766.813846  ,  9815.72696928,  9910.0548248 ,
```

```
10084.55078617,  9878.98392953, 10062.47411711,  9925.71203584,
 9907.23565242, 10013.30654655,  9912.92941409,  9812.40292018,
 9784.32862367,  9809.02845268,  9696.53888299,  9622.04922847,
 9648.07460792,  9471.66533819,  9503.25761746,  9378.27188497,
 9607.95550512,  9648.95221654,  9692.54746781,  9563.89993924,
 9541.4920295 ,  9680.33872575,  9910.32554564,  9726.2456664 ,
 9730.71442895,  9698.40371947,  9659.14100802,  9592.75539543,
 9571.96441852,  9564.5614953 ,  9577.51944093,  9485.11949569,
 9446.29647444,  9374.48200496,  9210.81042163,  9165.46541612,
 9288.09229978,  9363.65612758,  9539.57691811,  9475.38364731,
 9471.32942648,  9438.96531467,  9355.28938916,  9387.2318967 ,
 9308.06401295,  9212.45249809,  9116.68724306,  9011.38679763,
 9269.21688875,  9351.52666533,  9321.85317551,  9164.79787767,
 9224.31330701,  9379.21406929,  9249.47476171,  9156.77992449,
 9233.25939929,  9217.37207707,  9085.82844506,  9043.74627707,
 8967.81809699,  9064.36570814,  9010.08841928,  8983.73305852,
 8978.75777493,  8958.45061438,  8859.12668667,  8817.04329585,
 8699.98102699,  8746.32027561,  8522.78303703,  8309.44305826,
 8270.37556396,  8092.18711652,  8089.67949458,  8153.79383372,
 8070.33373693,  8224.22816416,  8192.2410547 ,  8403.21651932,
 8338.84736788,  8576.9652486 ,  8385.95430965,  8397.81170581,
 8249.00710883,  8180.33211211,  8445.83738346,  8518.76902698,
 8536.71246213,  8414.11534336,  8453.37693449,  8541.40989454,
 8687.02751018,  8883.38616556,  8646.86070628,  8675.92160474,
 8726.72144596,  8684.01358126,  8631.16558555,  8616.00819164,
 8657.91237193,  8620.89953271,  8518.51002179,  8316.04617008,
 8328.55001701,  8381.82852314,  8153.73145104,  8252.89290143,
 8291.88487648,  8390.49267754,  8387.68004804,  8330.27735648,
 8280.7892727 ,  8228.09565081,  8376.9175215 ,  8312.68145422,
 8438.89186678,  8308.10863151,  8438.99503456,  8340.5296205 ,
 8281.09265738,  8125.41856673,  8171.67807452,  8246.65694461,
 8071.05877525,  7938.72410953,  8018.31691584,  8040.6040826 ,
 7764.36928698,  7876.88928673,  7811.8834602 ,  8045.07327135,
 8024.65037353,  7956.51356149,  7880.78277916,  8020.66209733,
 8070.01455719,  7743.12601497,  7745.27105813,  7891.68473629,
 8069.67363918,  8006.1097554 ,  7791.12497974,  7544.98939986,
 7540.64257475,  7677.68717887,  7976.07163409,  7693.55594815,
 7572.83843367,  7700.97767905,  7209.62031658,  7330.16866677,
 7510.14217563,  7368.89845555,  7191.16892154,  7135.57660284,
 7156.73555197,  6967.92197626,  6846.4987033 ,  6519.72206774,
 6539.80689488,  6550.90784292,  6860.44614457,  6647.15016194,
 6821.24952338,  7009.99740166,  7131.13492692,  7190.68688497,
 7322.43783468,  7052.17460008,  7367.70876868,  7531.35173529,
 7540.83856674,  7481.67940862,  7963.48324358,  7908.78826473,
 7972.72139081,  7901.98474392,  8242.37278359,  8181.00490383,
 8025.76496926,  7944.7919521 ,  7953.59445868,  7981.85600799,
 7949.14942204,  8101.44429218,  8326.0637809 ,  8101.88436269,
 8040.71813919,  8038.2319909 ,  7999.73833365,  8212.08206871,
 8003.55061884,  8156.23943246,  8212.75456538,  8088.28301105,
 8437.65317748,  8430.93219249,  8509.92814845,  8710.57907904,
 8641.84936619,  8914.46301081,  8852.98699569,  8968.28412931,
 8624.86404317,  8770.13870895,  8427.16035282,  8499.43254404,
 8429.2085668 ,  8376.82975768,  8488.51324339,  8495.37395398,
 8616.65659795,  8669.97926449,  8834.65844063,  8459.54325606,
 8550.67775447,  8509.1338834 ,  8597.61423765,  8612.16731103,
 8891.32578565,  8584.25557614,  8472.38727006,  8557.53151312,
 8461.79947691,  8071.5421275 ,  8814.35814016,  8729.21316514,
 8378.23494607,  8443.66443082,  8040.18938642,  7530.4437334 ,
 7881.43536141,  8430.23955005,  8220.48821632,  8443.15226434,
 8737.57426947,  8197.2348755 ,  8756.68293683,  8811.04257109,
 8805.17670899,  8677.28017098,  9116.11959139,  9532.82782645,
```

```
 9258.34510025,  9380.99801123,  9135.42890106,  8865.25909517,
 8994.26968863,  8110.62249095,  8293.32938387,  8582.27586158,
 8432.15031131,  8977.20010309,  9184.73527933,  8760.32287085,
 9116.70805406,  8530.10874758,  9164.933829  ,  9285.12573541,
 8354.04303787,  8502.48291078,  9115.21150775,  9366.3247182 ,
 9976.11925614, 10238.70492153, 10378.79724743, 10735.52997166,
10729.35642073, 10269.48838049, 11012.99797365, 10988.69949626,
10690.0651269 , 10740.65450594, 10900.10563169, 11286.20712524,
10890.27888003, 10531.3765814 , 11009.07166444, 10783.24896552,
11247.10130229, 11420.66328249, 11166.12601955, 11121.35592903,
11361.37697494, 11095.03026759, 11101.25054932, 11433.62497721,
11407.50063766, 11476.92713962, 11584.54772436]))
```