CNN_BiLSTM

October 7, 2025

1 CNN-BiLSTM Modeling

1.1 Library Importing

[738]:
```
# Python Standard Libraries
import os
import csv
import math
import random
import unicodedata

# Data Libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# NLP - NLTK
import nltk
nltk.download('vader_lexicon')
from nltk.sentiment.vader import SentimentIntensityAnalyzer

# Scikit-learn
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler, StandardScaler
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

# PyTorch
import torch
import torch.nn as nn
from torch.utils.data import Dataset, DataLoader

# TensorFlow / Keras
import tensorflow as tf
from tensorflow.keras.models import Model, Sequential
from tensorflow.keras.layers import (
    Input, Dense, Dropout, LSTM, Bidirectional,
    Conv1D, Conv2D, MaxPooling1D, MaxPooling2D,
```

```python
    Flatten, GlobalAveragePooling1D, LayerNormalization,
    MultiHeadAttention, Add, Attention, Permute, Concatenate, Lambda
)
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau,␣
 ↪ModelCheckpoint
import tensorflow.keras.backend as K
from tensorflow.keras.losses import Huber

# XGBoost
import xgboost as xgb
from xgboost import XGBRegressor

# Shap
import shap
```

```
[nltk_data] Downloading package vader_lexicon to
[nltk_data]     /Users/yourth/nltk_data…
[nltk_data]   Package vader_lexicon is already up-to-date!
```

1.2 Data Importing

[739]:
```python
company_list = ['TSLA', 'AAPL', 'AMZN', 'GOOGL', 'MSFT']
stock_data_dict = {}

for symbol in company_list:
    path = f"./data/filtered/{symbol}_filtered.csv"
    stock_data_dict[symbol] = pd.read_csv(path)
```

[740]:
```python
TSLA = stock_data_dict['TSLA']
AAPL = stock_data_dict['AAPL']
AMZN = stock_data_dict['AMZN']
GOOGL = stock_data_dict['GOOGL']
MSFT = stock_data_dict['MSFT']
```

[741]:
```python
# df = GOOGL.copy()
# df.head()
```

1.3 Temp

[800]:
```python
def data_integration(stock, n_past=5, n_future=1):
    df = stock_data_dict[stock]

    # 1. Data Lagging
    # Lag technical indicators to avoid leakage
    lag_cols = [
        'Adj Close', 'High', 'Low', 'Volume', 'SMA_5', 'SMA_20',
```

```
    'BB_Mid', 'BB_Std', 'BB_Upper', 'BB_Lower',
    'RSI_14', 'Log_Return', 'OBV', 'Vader_Polarity'
]

for col in lag_cols:
    df[f"{col} (lag1)"] = df[col].shift(1)

feature_cols = [
    'Adj Close',  # should be the first one for Y
    # 'Open',
    'DayOfWeek',
    'Month',
    'Adj Close (lag1)',
    'SMA_5 (lag1)',              # short-term trend
    'Volume (lag1)',
    'BB_Mid (lag1)',            # risk signal
    'Log_Return (lag1)',
    'Vader_Polarity (lag1)'  # should be the last one for SENTIMENT
]

df = df[feature_cols]

train_size = 0.7
train_split_idx = int(train_size * len(df))

df_filtered = df[feature_cols]
df_filtered = df_filtered.dropna().reset_index(drop=True)

# df_filtered = df_filtered.iloc[1:] # delete nan       lag1

# 1   remove *every* row that has even one missing value
# df_filtered = df_filtered.dropna().reset_index(drop=True)

# Step 0: Define split boundaries BEFORE scaling
train_df = df_filtered.iloc[:train_split_idx] # 880
test_df  = df_filtered.iloc[train_split_idx - 5:] # 382

print(train_df.shape, test_df.shape)

# Step 1: Fit scaler only on training data (Avoid Data Leakage)
scaler = MinMaxScaler()
scaler.fit(train_df)

# Step 2: Scale training and test data separately
train_scaled = scaler.transform(train_df)
test_scaled  = scaler.transform(test_df)
```

```
    # Step 3: For inference later, only scale ['Adj Close']
    scaler_for_inference = MinMaxScaler()
    actual_scaled_close = scaler_for_inference.fit_transform(
        df_filtered[['Adj Close']]
    )

    # Step 3: Reconstruct sliding windows for train and test
    def create_sequences(data, n_past, n_future):
        X, y = [], []
        for i in range(n_past, len(data) - n_future + 1): # 5 to 880-1+1=880␣
↪(actual 879)
            X.append(data[i - n_past:i, 1:]) # 5-5=0 to 5 (actually 4th) ###␣
↪879-5=874 to 879 (actual 878)
            y.append(data[i + n_future - 2:i + n_future - 1, [0]])  # Predict␣
↪Adj Close  # 5+1-2=4 to 5 (actually 4th) ### 878 to 879 (actual 878)
        return np.array(X), np.array(y)

    trainX, trainY = create_sequences(train_scaled, n_past, n_future)
    testX, testY   = create_sequences(test_scaled, n_past, n_future)
    print(trainX.shape, testX.shape)


    # trainY = trainY.reshape(-1, 1)
    # testY = testY.reshape(-1, 1)

    # Without Sentiment (Baseline Model)
    trainX_wo_tweet = trainX[:, :, :-1]  # Exclude last feature
    testX_wo_tweet  = testX[:, :, :-1]
    trainY_wo_tweet = trainY
    testY_wo_tweet  = testY

    # With Sentiment (Tweet-based Model)
    trainX_with_tweet = trainX
    testX_with_tweet  = testX
    trainY_with_tweet = trainY
    testY_with_tweet  = testY

    return df_filtered, scaler, trainX, trainY, testX, testY, trainX_wo_tweet,␣
↪testX_wo_tweet, trainY_wo_tweet, testY_wo_tweet, trainX_with_tweet,␣
↪testX_with_tweet, trainY_with_tweet, testY_with_tweet
```

1.4 Data

[1032]:
```
company_list = ['TSLA', 'AAPL', 'AMZN', 'GOOGL', 'MSFT']
stock_data_dict = {}

for symbol in company_list:
```

```
    path = f"./data/filtered/{symbol}_filtered.csv"
    stock_data_dict[symbol] = pd.read_csv(path)

stock = 'GOOGL'

df_filtered, scaler, trainX, trainY, testX, testY, trainX_wo_tweet, testX_wo_tweet, trainY_wo_tweet, testY_wo_tweet, trainX_with_tweet, testX_with_tweet, trainY_with_tweet, testY_with_tweet = data_integration(stock=stock)
```

```
(880, 9) (382, 9)
(875, 5, 8) (377, 5, 8)
```

1.5 Plotting

1.5.1 1. CNN-BiLSTM

[996]:

```
def plot_metrics(model, history, X_train, y_train, X_val, y_val, sentiment_mode, stock):
    """
      RMSE   MAE         RMSE / MAE
    """
    history_data = history.history  #

    # 1.
    y_train_pred = model.predict(X_train, verbose=0).squeeze()
    y_val_pred   = model.predict(X_val, verbose=0).squeeze()

    y_train_true = y_train.squeeze()
    y_val_true   = y_val.squeeze()
    # y_train_true = y_train.reshape(-1, 1)
    # y_val_true   = y_val.reshape(-1, 1)

    # 2.
    train_rmse = np.sqrt(mean_squared_error(y_train_true, y_train_pred))
    val_rmse   = np.sqrt(mean_squared_error(y_val_true, y_val_pred))

    train_mae = mean_absolute_error(y_train_true, y_train_pred)
    val_mae   = mean_absolute_error(y_val_true, y_val_pred)

    title_suffix = " w/ Tweets" if sentiment_mode == "with" else "w/o Tweets"

    # 3.
    print(f"  {title_suffix} RMSE: Train = {train_rmse:.4f}, Test = {val_rmse:.4f}")
    print(f"  {title_suffix} MAE : Train = {train_mae:.4f}, Test = {val_mae:.4f}")
```

```
    # 4.    loss
    plt.figure(figsize=(21, 9))

    # ---- Huber(0.07) subplot
    plt.subplot(1, 2, 1)
    plt.plot(history_data['loss'], label='Train Huber')
    plt.plot(history_data['val_loss'], label='Val Huber')
    plt.title(f'{stock} - Huber(0.07) {title_suffix}', fontsize=18)
    plt.xlabel('Epoch', fontsize=18);
    plt.ylabel('Huber', fontsize=18)
    plt.xticks(fontsize=16)
    plt.yticks(fontsize=16)
    plt.legend()

    # ---- MAE subplot
    plt.subplot(1, 2, 2)
    if 'mse' in history_data:
        plt.plot(history_data['mse'], label='Train MSE')
        plt.plot(history_data['val_mse'], label='Val MSE')
    else:
        # fallback: flat line (not recommended long term)
        plt.plot([train_mae] * len(history_data['loss']), label='Train MAE␣
↪(flat)')
        plt.plot([val_mae]   * len(history_data['val_loss']), label='Val MAE␣
↪(flat)')
    plt.title(f'{stock} - MSE {title_suffix}', fontsize=18)
    plt.xlabel('Epoch', fontsize=18);
    plt.ylabel('MSE', fontsize=18)
    plt.xticks(fontsize=16)
    plt.yticks(fontsize=16)
    plt.legend()

    #  Add supertitle here
    plt.suptitle(f'Model Training Metrics - CNN+BiLSTM', fontsize=24)
    plt.tight_layout()

    plt.show()
```

[835]:
```
# Helper Function: get predicted and true values in original (unscaled) price␣
↪units
def inv_preds(model, X, y, scaler):
    # Predict using the model and reshape to [samples, output_dim]
    p = model.predict(X, verbose=0).reshape(-1, y.shape[-1])

    # Reshape ground truth to same shape for inverse transformation
    g = y.reshape(-1, y.shape[-1])
```

```
    # Apply inverse scaling to recover original price scale
    # inv_p = scaler_for_inference.inverse_transform(p)
    # inv_g = scaler_for_inference.inverse_transform(g)

    # Return only the first column ('Adj Close') from each
    # return inv_p[:, 0], inv_g[:, 0]

    # Use the full-feature scaler
    inv_p = scaler.inverse_transform(
        np.concatenate([p, np.zeros((p.shape[0], scaler.n_features_in_ - 1))],␣
 ↪axis=1)
    )[:, 0]
    inv_g = scaler.inverse_transform(
        np.concatenate([g, np.zeros((g.shape[0], scaler.n_features_in_ - 1))],␣
 ↪axis=1)
    )[:, 0]

    return inv_p, inv_g
```

[836]:
```
def plot_pred_vs_actual(
    model, X_train, y_train, X_test, y_test, scaler,
    model_name,
    df,
    stock_name,
    sentiment_mode="without"
):
    """
        +      vs



    - model
    - X_train / y_train
    - X_test / y_test
    - all_dates    index  df.index
    - all_adj        df['Adj Close']
    - scaler_for_inference      scaler    fit
    - title_prefix      'NFLX'
    - sentiment_mode 'with' / 'without'
    """

    # df=df[5:]

    all_dates=df.index

    all_adj=df['Adj Close']
    # Window Siz: number of lookback days
    lookback = X_train.shape[1] # 5
```

```python
    # Number of training samples
    n_train  = X_train.shape[0] # 875

    # Number of testing samples
    n_test   = X_test.shape[0] # 377

    # Predictions will start from the index equal to lookback (since earlier data was used for input windows)
    pred_start = lookback - 1

    # Time indices for training predictions aligned with full date range
    train_pts = np.arange(pred_start, pred_start + n_train) # 4 to 4+875=879 (Actual 878)

    # Time indices for test predictions aligned with full date range
    test_pts  = np.arange(pred_start + n_train,
                          pred_start + n_train + n_test) # 4+875=879 to 879+377=1256  (Actual 1255)

    # Get model predictions and true values (unscaled)
    pred_close_train, _ = inv_preds(model, X_train, y_train, scaler)
    pred_close_test,  _ = inv_preds(model, X_test, y_test, scaler)

    # Set title & label based on mode
    title_suffix = f"({model_name} w/ Tweets)" if sentiment_mode == "with" else f"({model_name} w/o Tweets)"

    x = all_dates[train_pts.tolist() + test_pts.tolist()]
    y = all_adj[np.arange(pred_start, pred_start + n_train + n_test)]

    plt.figure(figsize=(18, 7))
    # sns.lineplot(x=all_dates[pred_start:], y=all_adj[pred_start:], label='Actual Closing', color='black')
    sns.lineplot(x=x, y=y, label="Actual Closing", color='black')

    sns.lineplot(x=all_dates[train_pts], y=pred_close_train, label='Train Predicted', color='red')
    sns.lineplot(x=all_dates[test_pts],  y=pred_close_test,  label='Test Predicted')
    plt.axvline(x=all_dates[train_pts[-1]], color='gray', linestyle='--', label='Train/Test Split')

    plt.title(f'{stock_name} - Closing Price vs Actual Stock {title_suffix}', fontsize=20)
    plt.xlabel('Date', fontsize=18)
```

```python
    plt.ylabel('Price (Dollar)', fontsize=18)
    plt.xticks(fontsize=14)
    plt.yticks(fontsize=14)
    plt.legend(fontsize=16)
    plt.grid(True)
    plt.tight_layout()
    plt.show()
```

1.6 CNN-BiLSTM

[1033]:
```python
# def cnn_biLSTM(input_shape, output_dim):

#     model = Sequential()

#     model.add(Conv1D(128, kernel_size=2, strides=1, padding='valid',␣
↪input_shape=input_shape))
#     model.add(MaxPooling1D(pool_size=2, strides=2))

#     model.add(Conv1D(64, kernel_size=2, strides=1, padding='valid'))
#     model.add(MaxPooling1D(pool_size=1, strides=2))

#     model.add(Bidirectional(LSTM(256, return_sequences=True)))
#     model.add(Dropout(0.2))
#     model.add(Bidirectional(LSTM(256, return_sequences=True)))
#     model.add(Dropout(0.2))

#     model.add(Dense(32, activation='relu'))
#     model.add(Dense(output_dim, activation='relu'))
#     # model.summary()
#     return model
```

[1034]:
```python
def cnn_biLSTM(input_shape, output_dim):
    inputs = Input(shape=input_shape)

    x = Conv1D(128, kernel_size=2, strides=1, padding='valid')(inputs)
    x = MaxPooling1D(pool_size=2, strides=2)(x)

    x = Conv1D(64, kernel_size=2, strides=1, padding='valid')(x)
    x = MaxPooling1D(pool_size=1, strides=2)(x)

    x = Bidirectional(LSTM(256, return_sequences=True))(x)
    x = Dropout(0.20)(x)
    x = Bidirectional(LSTM(256, return_sequences=True))(x)
    x = Dropout(0.20)(x)

    # === Add attention here ===
```

```python
    attn_out = Attention(use_scale=True)([x, x])  # Self-attention: query =␣
↪value = key = x
    x = GlobalAveragePooling1D()(attn_out)

    x = Dense(64, activation='relu')(x)
    outputs = Dense(output_dim, activation='relu')(x)

    return Model(inputs, outputs)
```

[1035]:
```python
def cnn_biLSTM_SENT(input_shape, sentiment_index, output_dim):
    """
    Cross-attention version: BiLSTM attends to sentiment sequence separately.
    input_shape: (n_past, num_features)   e.g., (5, 9)
    sentiment_index: int, index of sentiment feature (e.g., 8)
    output_dim: usually 1
    """

    # === Input full sequence ===
    full_input = Input(shape=input_shape)

    # === Split sentiment and rest ===
    sentiment = Lambda(lambda x: tf.expand_dims(x[:, :, sentiment_index],␣
↪axis=-1))(full_input)
    rest = Lambda(lambda x: tf.concat([
        x[:, :, :sentiment_index], x[:, :, sentiment_index+1:]
    ], axis=-1))(full_input)

    # === CNN + BiLSTM on main inputs ===
    x = Conv1D(128, kernel_size=2, strides=1, padding='valid')(rest)
    x = MaxPooling1D(pool_size=2, strides=2)(x)

    x = Conv1D(128, kernel_size=2, strides=1, padding='valid')(x)
    x = MaxPooling1D(pool_size=1, strides=2)(x)

    x = Bidirectional(LSTM(256, return_sequences=True))(x)
    x = Dropout(0.2)(x)
    x = Bidirectional(LSTM(256, return_sequences=True))(x)
    x = Dropout(0.2)(x)

    # === Cross-attention: query = price features, key/value = sentiment ===
    sentiment_proj = Dense(x.shape[-1])(sentiment)  # (batch, time, feature_dim)
    attended_sentiment = Attention(use_scale=True)([x, sentiment_proj])

    # === Combine attended sentiment with price path ===
    x = Concatenate()([x, attended_sentiment])
    x = GlobalAveragePooling1D()(x)
```

```
    # === Dense output ===
    x = Dense(64, activation='relu')(x)
    x = Dropout(0.2)(x)
    output = Dense(output_dim, activation='relu')(x)

    return Model(inputs=full_input, outputs=output)
```

[1036]:
```
# Build models
cnnBiLSTM_woSent = cnn_biLSTM(
    (trainX_wo_tweet.shape[1], trainX_wo_tweet.shape[2]), trainY.shape[2]
)
cnnBiLSTM_woSent.compile(
    optimizer=Adam(learning_rate=0.001),
    # loss=integrated_loss(delta=0.1, lambda_dir=0.16),  #  adjust as needed
    # loss='mse',  #  adjust as needed
    loss=Huber(0.07),
    metrics=['mse']
)



cnnBiLSTM_withSent = cnn_biLSTM(
    (trainX_with_tweet.shape[1], trainX_with_tweet.shape[2]), trainY.shape[2]
)
cnnBiLSTM_withSent.compile(
    optimizer=Adam(learning_rate=0.001),
    # loss=integrated_loss(delta=0.1, lambda_dir=0.16),  #  adjust as needed
    # loss='mse',  #  adjust as needed
    loss=Huber(0.07),
    metrics=['mse']
)
```

[1037]:
```
early_stop = EarlyStopping(
    monitor='val_loss',
    patience=20,
    restore_best_weights=True
)

reduce_lr = ReduceLROnPlateau(
    monitor='val_loss',
    factor=0.5,         # ← good default
    patience=3,
    min_lr=1e-7,
    verbose=1
)
```

[1038]:
```
# Fit models
history_cnnBiLSTM_woSent = cnnBiLSTM_woSent.fit(
```

```
    trainX_wo_tweet,
    trainY_wo_tweet,
    epochs=50,
    batch_size=64,
    validation_data=(testX_wo_tweet, testY_wo_tweet),  # ← use your test split␣
  ↪here
    verbose=1,
    callbacks=[early_stop, reduce_lr]
)
```

```
Epoch 1/50

/Users/yourth/Desktop/aaa/venv/lib/python3.11/site-
packages/keras/src/ops/nn.py:908: UserWarning: You are using a softmax over axis
-1 of a tensor of shape (None, 1, 1). This axis has size 1. The softmax
operation will always return the value 1, which is likely not what you intended.
Did you mean to use a sigmoid instead?
  warnings.warn(

14/14            3s 38ms/step -
loss: 0.0194 - mse: 0.1546 - val_loss: 0.0161 - val_mse: 0.0749 - learning_rate:
0.0010
Epoch 2/50
14/14            0s 11ms/step -
loss: 0.0044 - mse: 0.0129 - val_loss: 0.0068 - val_mse: 0.0212 - learning_rate:
0.0010
Epoch 3/50
14/14            0s 9ms/step - loss:
0.0017 - mse: 0.0039 - val_loss: 0.0028 - val_mse: 0.0067 - learning_rate:
0.0010
Epoch 4/50
14/14            0s 9ms/step - loss:
0.0010 - mse: 0.0022 - val_loss: 0.0013 - val_mse: 0.0029 - learning_rate:
0.0010
Epoch 5/50
14/14            0s 8ms/step - loss:
8.3562e-04 - mse: 0.0018 - val_loss: 9.8202e-04 - val_mse: 0.0021 -
learning_rate: 0.0010
Epoch 6/50
14/14            0s 9ms/step - loss:
7.5788e-04 - mse: 0.0016 - val_loss: 0.0011 - val_mse: 0.0023 - learning_rate:
0.0010
Epoch 7/50
14/14            0s 8ms/step - loss:
5.8731e-04 - mse: 0.0012 - val_loss: 9.6089e-04 - val_mse: 0.0020 -
learning_rate: 0.0010
Epoch 8/50
10/14            0s 6ms/step - loss:
6.8245e-04 - mse: 0.0014
```

```
Epoch 8: ReduceLROnPlateau reducing learning rate to 0.0005000000237487257.
14/14            0s 8ms/step - loss:
6.7678e-04 - mse: 0.0014 - val_loss: 0.0013 - val_mse: 0.0028 - learning_rate:
0.0010
Epoch 9/50
14/14            0s 8ms/step - loss:
6.2913e-04 - mse: 0.0013 - val_loss: 0.0014 - val_mse: 0.0030 - learning_rate:
5.0000e-04
Epoch 10/50
14/14            0s 8ms/step - loss:
6.7475e-04 - mse: 0.0014 - val_loss: 0.0023 - val_mse: 0.0050 - learning_rate:
5.0000e-04
Epoch 11/50
14/14            0s 8ms/step - loss:
6.8026e-04 - mse: 0.0014 - val_loss: 8.7143e-04 - val_mse: 0.0018 -
learning_rate: 5.0000e-04
Epoch 12/50
14/14            0s 8ms/step - loss:
5.0685e-04 - mse: 0.0011 - val_loss: 7.7793e-04 - val_mse: 0.0016 -
learning_rate: 5.0000e-04
Epoch 13/50
14/14            0s 8ms/step - loss:
6.1574e-04 - mse: 0.0013 - val_loss: 7.9276e-04 - val_mse: 0.0016 -
learning_rate: 5.0000e-04
Epoch 14/50
10/14            0s 6ms/step - loss:
5.0655e-04 - mse: 0.0011
Epoch 14: ReduceLROnPlateau reducing learning rate to 0.0002500000118743628.
14/14            0s 8ms/step - loss:
5.2142e-04 - mse: 0.0011 - val_loss: 8.5341e-04 - val_mse: 0.0018 -
learning_rate: 5.0000e-04
Epoch 15/50
14/14            0s 9ms/step - loss:
5.1553e-04 - mse: 0.0011 - val_loss: 8.3655e-04 - val_mse: 0.0017 -
learning_rate: 2.5000e-04
Epoch 16/50
14/14            0s 8ms/step - loss:
5.3302e-04 - mse: 0.0011 - val_loss: 9.1990e-04 - val_mse: 0.0019 -
learning_rate: 2.5000e-04
Epoch 17/50
14/14            0s 8ms/step - loss:
5.1575e-04 - mse: 0.0011 - val_loss: 7.3682e-04 - val_mse: 0.0015 -
learning_rate: 2.5000e-04
Epoch 18/50
14/14            0s 8ms/step - loss:
4.7199e-04 - mse: 9.8654e-04 - val_loss: 8.1501e-04 - val_mse: 0.0017 -
learning_rate: 2.5000e-04
Epoch 19/50
```

```
14/14             0s 8ms/step - loss:
4.6148e-04 - mse: 9.5944e-04 - val_loss: 6.9363e-04 - val_mse: 0.0014 -
learning_rate: 2.5000e-04
Epoch 20/50
10/14             0s 6ms/step - loss:
4.5370e-04 - mse: 9.4359e-04
Epoch 20: ReduceLROnPlateau reducing learning rate to 0.0001250000059371814.
14/14             0s 8ms/step - loss:
4.5397e-04 - mse: 9.4731e-04 - val_loss: 0.0017 - val_mse: 0.0035 -
learning_rate: 2.5000e-04
Epoch 21/50
14/14             0s 8ms/step - loss:
5.1330e-04 - mse: 0.0011 - val_loss: 0.0013 - val_mse: 0.0026 - learning_rate:
1.2500e-04
Epoch 22/50
14/14             0s 8ms/step - loss:
4.5885e-04 - mse: 9.8144e-04 - val_loss: 6.9168e-04 - val_mse: 0.0014 -
learning_rate: 1.2500e-04
Epoch 23/50
10/14             0s 6ms/step - loss:
4.4481e-04 - mse: 9.1612e-04
Epoch 23: ReduceLROnPlateau reducing learning rate to 6.25000029685907e-05.
14/14             0s 8ms/step - loss:
4.5345e-04 - mse: 9.3355e-04 - val_loss: 6.8611e-04 - val_mse: 0.0014 -
learning_rate: 1.2500e-04
Epoch 24/50
14/14             0s 8ms/step - loss:
4.7691e-04 - mse: 9.9763e-04 - val_loss: 7.0589e-04 - val_mse: 0.0015 -
learning_rate: 6.2500e-05
Epoch 25/50
14/14             0s 9ms/step - loss:
4.0907e-04 - mse: 8.4936e-04 - val_loss: 8.9242e-04 - val_mse: 0.0018 -
learning_rate: 6.2500e-05
Epoch 26/50
 9/14             0s 7ms/step - loss:
4.2832e-04 - mse: 8.7785e-04
Epoch 26: ReduceLROnPlateau reducing learning rate to 3.125000148429535e-05.
14/14             0s 9ms/step - loss:
4.3617e-04 - mse: 8.9568e-04 - val_loss: 6.8058e-04 - val_mse: 0.0014 -
learning_rate: 6.2500e-05
Epoch 27/50
14/14             0s 8ms/step - loss:
4.7613e-04 - mse: 9.9989e-04 - val_loss: 9.2607e-04 - val_mse: 0.0019 -
learning_rate: 3.1250e-05
Epoch 28/50
14/14             0s 8ms/step - loss:
4.4343e-04 - mse: 9.2435e-04 - val_loss: 6.8633e-04 - val_mse: 0.0014 -
learning_rate: 3.1250e-05
```

```
Epoch 29/50
 9/14             0s 7ms/step - loss:
4.9447e-04 - mse: 0.0011
Epoch 29: ReduceLROnPlateau reducing learning rate to 1.5625000742147677e-05.
14/14             0s 9ms/step - loss:
4.7888e-04 - mse: 0.0010 - val_loss: 7.2927e-04 - val_mse: 0.0015 -
learning_rate: 3.1250e-05
Epoch 30/50
14/14             0s 9ms/step - loss:
4.0991e-04 - mse: 8.5679e-04 - val_loss: 7.7981e-04 - val_mse: 0.0016 -
learning_rate: 1.5625e-05
Epoch 31/50
14/14             0s 8ms/step - loss:
4.7384e-04 - mse: 0.0010 - val_loss: 8.3054e-04 - val_mse: 0.0017 -
learning_rate: 1.5625e-05
Epoch 32/50
10/14             0s 6ms/step - loss:
3.6241e-04 - mse: 7.6251e-04
Epoch 32: ReduceLROnPlateau reducing learning rate to 7.812500371073838e-06.
14/14             0s 9ms/step - loss:
3.7611e-04 - mse: 7.8806e-04 - val_loss: 7.5225e-04 - val_mse: 0.0016 -
learning_rate: 1.5625e-05
Epoch 33/50
14/14             0s 8ms/step - loss:
4.3614e-04 - mse: 9.2084e-04 - val_loss: 7.3243e-04 - val_mse: 0.0015 -
learning_rate: 7.8125e-06
Epoch 34/50
14/14             0s 8ms/step - loss:
4.3782e-04 - mse: 9.2357e-04 - val_loss: 7.6012e-04 - val_mse: 0.0016 -
learning_rate: 7.8125e-06
Epoch 35/50
11/14             0s 5ms/step - loss:
3.8955e-04 - mse: 7.9507e-04
Epoch 35: ReduceLROnPlateau reducing learning rate to 3.906250185536919e-06.
14/14             0s 8ms/step - loss:
3.9866e-04 - mse: 8.2076e-04 - val_loss: 7.6753e-04 - val_mse: 0.0016 -
learning_rate: 7.8125e-06
Epoch 36/50
14/14             0s 8ms/step - loss:
4.6768e-04 - mse: 9.7355e-04 - val_loss: 7.6236e-04 - val_mse: 0.0016 -
learning_rate: 3.9063e-06
Epoch 37/50
14/14             0s 10ms/step -
loss: 4.0530e-04 - mse: 8.3988e-04 - val_loss: 7.4101e-04 - val_mse: 0.0015 -
learning_rate: 3.9063e-06
Epoch 38/50
10/14             0s 6ms/step - loss:
4.3794e-04 - mse: 8.9773e-04
```

```
Epoch 38: ReduceLROnPlateau reducing learning rate to 1.9531250927684596e-06.
14/14               0s 8ms/step - loss:
4.4070e-04 - mse: 9.0830e-04 - val_loss: 7.9227e-04 - val_mse: 0.0016 -
learning_rate: 3.9063e-06
Epoch 39/50
14/14               0s 8ms/step - loss:
4.6224e-04 - mse: 9.7517e-04 - val_loss: 7.9789e-04 - val_mse: 0.0016 -
learning_rate: 1.9531e-06
Epoch 40/50
14/14               0s 8ms/step - loss:
4.3098e-04 - mse: 8.9051e-04 - val_loss: 7.8919e-04 - val_mse: 0.0016 -
learning_rate: 1.9531e-06
Epoch 41/50
10/14               0s 6ms/step - loss:
4.6161e-04 - mse: 9.7591e-04
Epoch 41: ReduceLROnPlateau reducing learning rate to 9.765625463842298e-07.
14/14               0s 8ms/step - loss:
4.5900e-04 - mse: 9.6631e-04 - val_loss: 7.8602e-04 - val_mse: 0.0016 -
learning_rate: 1.9531e-06
Epoch 42/50
14/14               0s 8ms/step - loss:
4.4697e-04 - mse: 9.3832e-04 - val_loss: 7.9486e-04 - val_mse: 0.0016 -
learning_rate: 9.7656e-07
Epoch 43/50
14/14               0s 8ms/step - loss:
4.2494e-04 - mse: 8.9314e-04 - val_loss: 7.9008e-04 - val_mse: 0.0016 -
learning_rate: 9.7656e-07
Epoch 44/50
10/14               0s 6ms/step - loss:
4.0554e-04 - mse: 8.4539e-04
Epoch 44: ReduceLROnPlateau reducing learning rate to 4.882812731921149e-07.
14/14               0s 8ms/step - loss:
4.1057e-04 - mse: 8.5912e-04 - val_loss: 7.9097e-04 - val_mse: 0.0016 -
learning_rate: 9.7656e-07
Epoch 45/50
14/14               0s 8ms/step - loss:
4.7039e-04 - mse: 9.9573e-04 - val_loss: 7.8497e-04 - val_mse: 0.0016 -
learning_rate: 4.8828e-07
Epoch 46/50
14/14               0s 8ms/step - loss:
4.6692e-04 - mse: 9.5930e-04 - val_loss: 7.7964e-04 - val_mse: 0.0016 -
learning_rate: 4.8828e-07
```

[1039]:
```
history_cnnBiLSTM_withSent = cnnBiLSTM_withSent.fit(
    trainX_with_tweet,
    trainY_with_tweet,
    epochs=50,
```

```
    batch_size=64,
    validation_data=(testX_with_tweet, testY_with_tweet),  # ← and here
    verbose=1,
    callbacks=[early_stop, reduce_lr]
)
```

```
Epoch 1/50
14/14              3s 33ms/step -
loss: 0.0197 - mse: 0.1532 - val_loss: 0.0203 - val_mse: 0.1128 - learning_rate:
0.0010
Epoch 2/50
14/14              0s 8ms/step - loss:
0.0063 - mse: 0.0221 - val_loss: 0.0037 - val_mse: 0.0098 - learning_rate:
0.0010
Epoch 3/50
14/14              0s 8ms/step - loss:
0.0019 - mse: 0.0045 - val_loss: 0.0029 - val_mse: 0.0071 - learning_rate:
0.0010
Epoch 4/50
14/14              0s 9ms/step - loss:
0.0011 - mse: 0.0023 - val_loss: 0.0019 - val_mse: 0.0044 - learning_rate:
0.0010
Epoch 5/50
14/14              0s 8ms/step - loss:
8.2237e-04 - mse: 0.0018 - val_loss: 0.0011 - val_mse: 0.0022 - learning_rate:
0.0010
Epoch 6/50
14/14              0s 8ms/step - loss:
7.6994e-04 - mse: 0.0017 - val_loss: 0.0012 - val_mse: 0.0025 - learning_rate:
0.0010
Epoch 7/50
14/14              0s 8ms/step - loss:
6.9746e-04 - mse: 0.0015 - val_loss: 0.0016 - val_mse: 0.0035 - learning_rate:
0.0010
Epoch 8/50
10/14              0s 6ms/step - loss:
8.7523e-04 - mse: 0.0019
Epoch 8: ReduceLROnPlateau reducing learning rate to 0.0005000000237487257.
14/14              0s 8ms/step - loss:
8.3672e-04 - mse: 0.0018 - val_loss: 9.9874e-04 - val_mse: 0.0021 -
learning_rate: 0.0010
Epoch 9/50
14/14              0s 8ms/step - loss:
6.1151e-04 - mse: 0.0013 - val_loss: 0.0013 - val_mse: 0.0027 - learning_rate:
5.0000e-04
Epoch 10/50
14/14              0s 8ms/step - loss:
6.2561e-04 - mse: 0.0013 - val_loss: 9.1403e-04 - val_mse: 0.0019 -
```

```
learning_rate: 5.0000e-04
Epoch 11/50
14/14            0s 8ms/step - loss:
5.5933e-04 - mse: 0.0012 - val_loss: 0.0010 - val_mse: 0.0021 - learning_rate:
5.0000e-04
Epoch 12/50
14/14            0s 8ms/step - loss:
5.4555e-04 - mse: 0.0012 - val_loss: 9.1755e-04 - val_mse: 0.0019 -
learning_rate: 5.0000e-04
Epoch 13/50
10/14            0s 6ms/step - loss:
5.6765e-04 - mse: 0.0012
Epoch 13: ReduceLROnPlateau reducing learning rate to 0.0002500000118743628.
14/14            0s 8ms/step - loss:
5.8551e-04 - mse: 0.0012 - val_loss: 9.1833e-04 - val_mse: 0.0019 -
learning_rate: 5.0000e-04
Epoch 14/50
14/14            0s 9ms/step - loss:
6.6820e-04 - mse: 0.0014 - val_loss: 0.0013 - val_mse: 0.0027 - learning_rate:
2.5000e-04
Epoch 15/50
14/14            0s 9ms/step - loss:
6.2479e-04 - mse: 0.0013 - val_loss: 8.5844e-04 - val_mse: 0.0018 -
learning_rate: 2.5000e-04
Epoch 16/50
11/14            0s 5ms/step - loss:
5.7472e-04 - mse: 0.0012
Epoch 16: ReduceLROnPlateau reducing learning rate to 0.0001250000059371814.
14/14            0s 8ms/step - loss:
5.6273e-04 - mse: 0.0012 - val_loss: 0.0011 - val_mse: 0.0023 - learning_rate:
2.5000e-04
Epoch 17/50
14/14            0s 8ms/step - loss:
5.0006e-04 - mse: 0.0010 - val_loss: 0.0010 - val_mse: 0.0021 - learning_rate:
1.2500e-04
Epoch 18/50
14/14            0s 8ms/step - loss:
4.8212e-04 - mse: 0.0010 - val_loss: 0.0012 - val_mse: 0.0026 - learning_rate:
1.2500e-04
Epoch 19/50
10/14            0s 6ms/step - loss:
4.9946e-04 - mse: 0.0010
Epoch 19: ReduceLROnPlateau reducing learning rate to 6.25000029685907e-05.
14/14            0s 8ms/step - loss:
4.9646e-04 - mse: 0.0010 - val_loss: 0.0013 - val_mse: 0.0028 - learning_rate:
1.2500e-04
Epoch 20/50
14/14            0s 8ms/step - loss:
```

```
4.8685e-04 - mse: 0.0010 - val_loss: 9.4135e-04 - val_mse: 0.0020 -
learning_rate: 6.2500e-05
Epoch 21/50
14/14            0s 8ms/step - loss:
4.9669e-04 - mse: 0.0010 - val_loss: 9.6453e-04 - val_mse: 0.0020 -
learning_rate: 6.2500e-05
Epoch 22/50
10/14            0s 6ms/step - loss:
5.1589e-04 - mse: 0.0011
Epoch 22: ReduceLROnPlateau reducing learning rate to 3.125000148429535e-05.
14/14            0s 8ms/step - loss:
5.1352e-04 - mse: 0.0011 - val_loss: 0.0012 - val_mse: 0.0026 - learning_rate:
6.2500e-05
Epoch 23/50
14/14            0s 8ms/step - loss:
4.9286e-04 - mse: 0.0010 - val_loss: 9.1979e-04 - val_mse: 0.0019 -
learning_rate: 3.1250e-05
Epoch 24/50
14/14            0s 8ms/step - loss:
4.9237e-04 - mse: 0.0010 - val_loss: 0.0011 - val_mse: 0.0023 - learning_rate:
3.1250e-05
Epoch 25/50
10/14            0s 6ms/step - loss:
5.6729e-04 - mse: 0.0012
Epoch 25: ReduceLROnPlateau reducing learning rate to 1.5625000742147677e-05.
14/14            0s 8ms/step - loss:
5.5058e-04 - mse: 0.0012 - val_loss: 0.0010 - val_mse: 0.0022 - learning_rate:
3.1250e-05
Epoch 26/50
14/14            0s 8ms/step - loss:
4.8395e-04 - mse: 0.0010 - val_loss: 0.0011 - val_mse: 0.0023 - learning_rate:
1.5625e-05
Epoch 27/50
14/14            0s 10ms/step -
loss: 5.4976e-04 - mse: 0.0012 - val_loss: 9.9938e-04 - val_mse: 0.0021 -
learning_rate: 1.5625e-05
Epoch 28/50
10/14            0s 6ms/step - loss:
5.1817e-04 - mse: 0.0011
Epoch 28: ReduceLROnPlateau reducing learning rate to 7.812500371073838e-06.
14/14            0s 8ms/step - loss:
5.0726e-04 - mse: 0.0011 - val_loss: 0.0010 - val_mse: 0.0022 - learning_rate:
1.5625e-05
Epoch 29/50
14/14            0s 8ms/step - loss:
4.9067e-04 - mse: 0.0010 - val_loss: 0.0010 - val_mse: 0.0021 - learning_rate:
7.8125e-06
Epoch 30/50
```

```
14/14            0s 8ms/step - loss:
5.0352e-04 - mse: 0.0010 - val_loss: 0.0010 - val_mse: 0.0022 - learning_rate:
7.8125e-06
Epoch 31/50
10/14            0s 6ms/step - loss:
4.7664e-04 - mse: 0.0010
Epoch 31: ReduceLROnPlateau reducing learning rate to 3.906250185536919e-06.
14/14            0s 8ms/step - loss:
4.7682e-04 - mse: 0.0010 - val_loss: 0.0010 - val_mse: 0.0021 - learning_rate:
7.8125e-06
Epoch 32/50
14/14            0s 8ms/step - loss:
4.6386e-04 - mse: 9.5640e-04 - val_loss: 9.7527e-04 - val_mse: 0.0020 -
learning_rate: 3.9063e-06
Epoch 33/50
14/14            0s 8ms/step - loss:
4.4053e-04 - mse: 8.9398e-04 - val_loss: 9.9796e-04 - val_mse: 0.0021 -
learning_rate: 3.9063e-06
Epoch 34/50
10/14            0s 6ms/step - loss:
4.9965e-04 - mse: 0.0010
Epoch 34: ReduceLROnPlateau reducing learning rate to 1.9531250927684596e-06.
14/14            0s 8ms/step - loss:
5.0146e-04 - mse: 0.0010 - val_loss: 0.0010 - val_mse: 0.0021 - learning_rate:
3.9063e-06
Epoch 35/50
14/14            0s 8ms/step - loss:
4.7981e-04 - mse: 0.0010 - val_loss: 9.9075e-04 - val_mse: 0.0021 -
learning_rate: 1.9531e-06
```

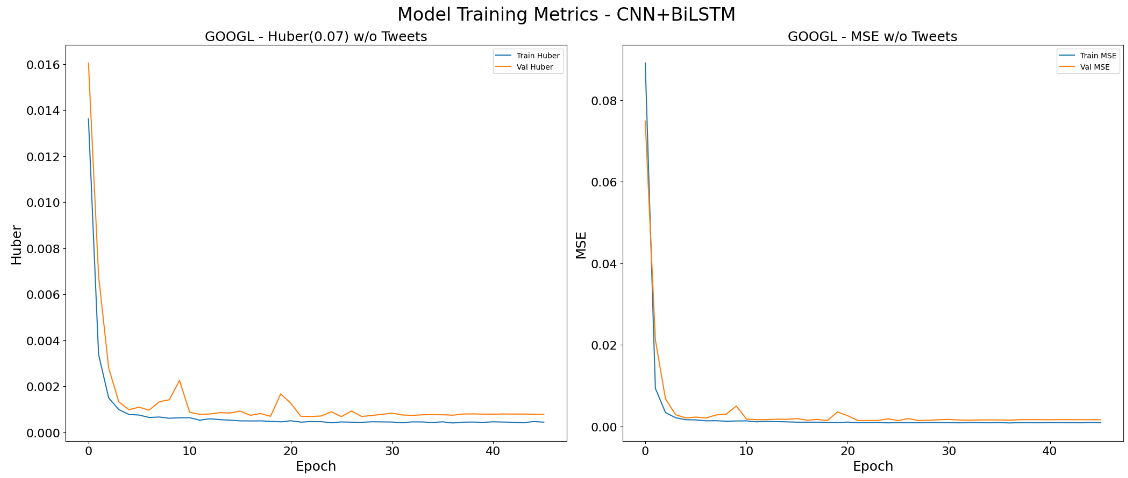
[1040]:

```python
plot_metrics(cnnBiLSTM_woSent, history_cnnBiLSTM_woSent,
             trainX_wo_tweet, trainY_wo_tweet,
             testX_wo_tweet, testY_wo_tweet,
             "without",
             stock=stock)

plot_metrics(cnnBiLSTM_withSent, history_cnnBiLSTM_withSent,
             trainX_with_tweet, trainY_with_tweet,
             testX_with_tweet, testY_with_tweet,
             "with",
             stock=stock)
```

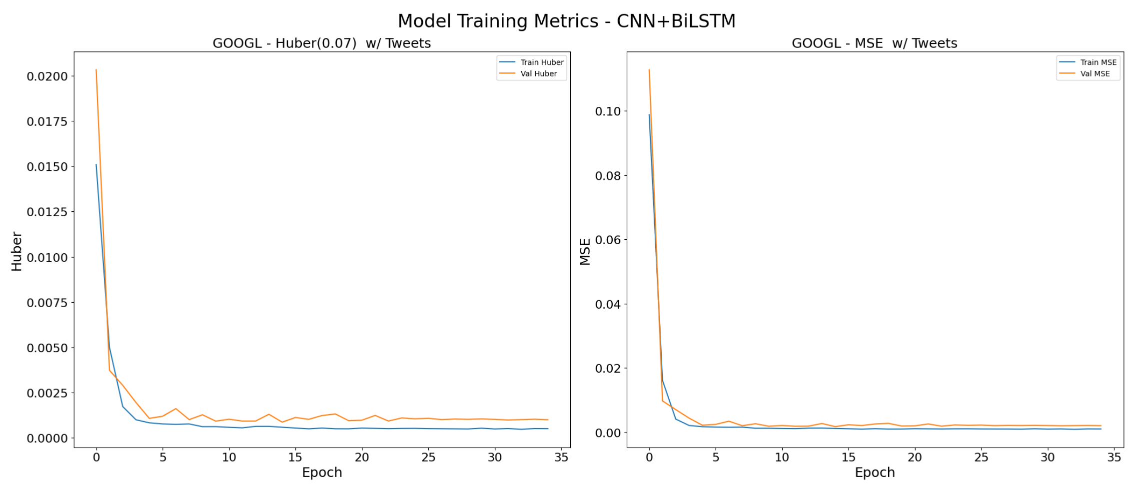
```
/Users/yourth/Desktop/aaa/venv/lib/python3.11/site-
packages/keras/src/ops/nn.py:908: UserWarning: You are using a softmax over axis
-1 of a tensor of shape (32, 1, 1). This axis has size 1. The softmax operation
will always return the value 1, which is likely not what you intended. Did you
mean to use a sigmoid instead?
  warnings.warn(
```

```
w/o Tweets RMSE: Train = 0.0248, Test = 0.0375
w/o Tweets MAE : Train = 0.0179, Test = 0.0297
```




```
/Users/yourth/Desktop/aaa/venv/lib/python3.11/site-
packages/keras/src/ops/nn.py:908: UserWarning: You are using a softmax over axis
-1 of a tensor of shape (32, 1, 1). This axis has size 1. The softmax operation
will always return the value 1, which is likely not what you intended. Did you
mean to use a sigmoid instead?
  warnings.warn(
/Users/yourth/Desktop/aaa/venv/lib/python3.11/site-
packages/keras/src/ops/nn.py:908: UserWarning: You are using a softmax over axis
-1 of a tensor of shape (None, 1, 1). This axis has size 1. The softmax
operation will always return the value 1, which is likely not what you intended.
Did you mean to use a sigmoid instead?
  warnings.warn(
```

```
w/ Tweets RMSE: Train = 0.0265, Test = 0.0422
w/ Tweets MAE : Train = 0.0193, Test = 0.0332
```



[1017]:
```
plot_pred_vs_actual(
    model=cnnBiLSTM_woSent,
    X_train=trainX_wo_tweet, y_train=trainY_wo_tweet,
    X_test=testX_wo_tweet,   y_test=testY_wo_tweet,
    scaler=scaler,
    df=df_filtered,
    model_name="CNN+BiLSTM",
    stock_name=stock,
    sentiment_mode="without"
)

plot_pred_vs_actual(
    model=cnnBiLSTM_withSent,
    X_train=trainX_with_tweet, y_train=trainY_with_tweet,
    X_test=testX_with_tweet,   y_test=testY_with_tweet,
    scaler=scaler,
    df=df_filtered,
    model_name="CNN+BiLSTM",
    stock_name=stock,
    sentiment_mode="with"
)
```

TSLA - Closing Price vs Actual Stock (CNN+BiLSTM w/ Tweets)
Actual Closing
Train Predicted
Test Predicted
Train/Test Split
Price (Dollar)
Date

[ ]:

[ ]: