Integrative Forecasting and Analysis of Stock Price Using Neural Network and ARIMA Model

Jin "Max" Li

Outline

Section 1

Executive Summary

Executive Summary

Summary of Methodologies

-Data Collection

-Data Cleaning

-Data Visulazation

-Prediction Analytics with Deep Learning models

Summary of experiments and results

-Predictive Analytics result

Section 2

Background

Background

Investors in finance and risk management are interested in forecasting direct investment returns from financial assets, including stocks, bonds, and commodities. Among all, the stock index is one of the essential indicators for investors. Investors rely heavily on the stock index to make optimal decisions to minimize loss and maximize their return.

However, precise forecasting of the stock index has been exacting since the financial market is inherent with its non-stationary nature in time series analysis. Besides, factors lying on the macro level, such as political and public events, company performance, consumer behavior, and national well-being, can significantly affect the stock price. Once the listed factors become public knowledge, the stock market will adjust in response, making the prediction seem impossible.

Since the market is not efficient, we can still predict future patterns based on historical market fluctuations.

Background

In the financial market analysis and predictions, studies have concerned short-term stock price forecasting on a scale of days to months and long-term forecasting on a scale of years. Both short-term and long-term predictions are valuable in decision-making. **Short-term forecasting** can capture the details of changing stock prices, while **long-term forecasting** reflects the market trend under specific changes and events in a macro-level concern. Therefore, various investors are concerned about the short-term and long-term performance of the stock market. Thus, combining short-term and long-term forecasting makes summarizing financial patterns more effective and comprehensive.

However, most studies focus on implementing forecasting models without regard to financial forecasting as an integrated subject. The short-term and long-term forecasting researches are usually done separately. This project proposes an integrated short-term and long-term forecasting method to solve the emerging problem.

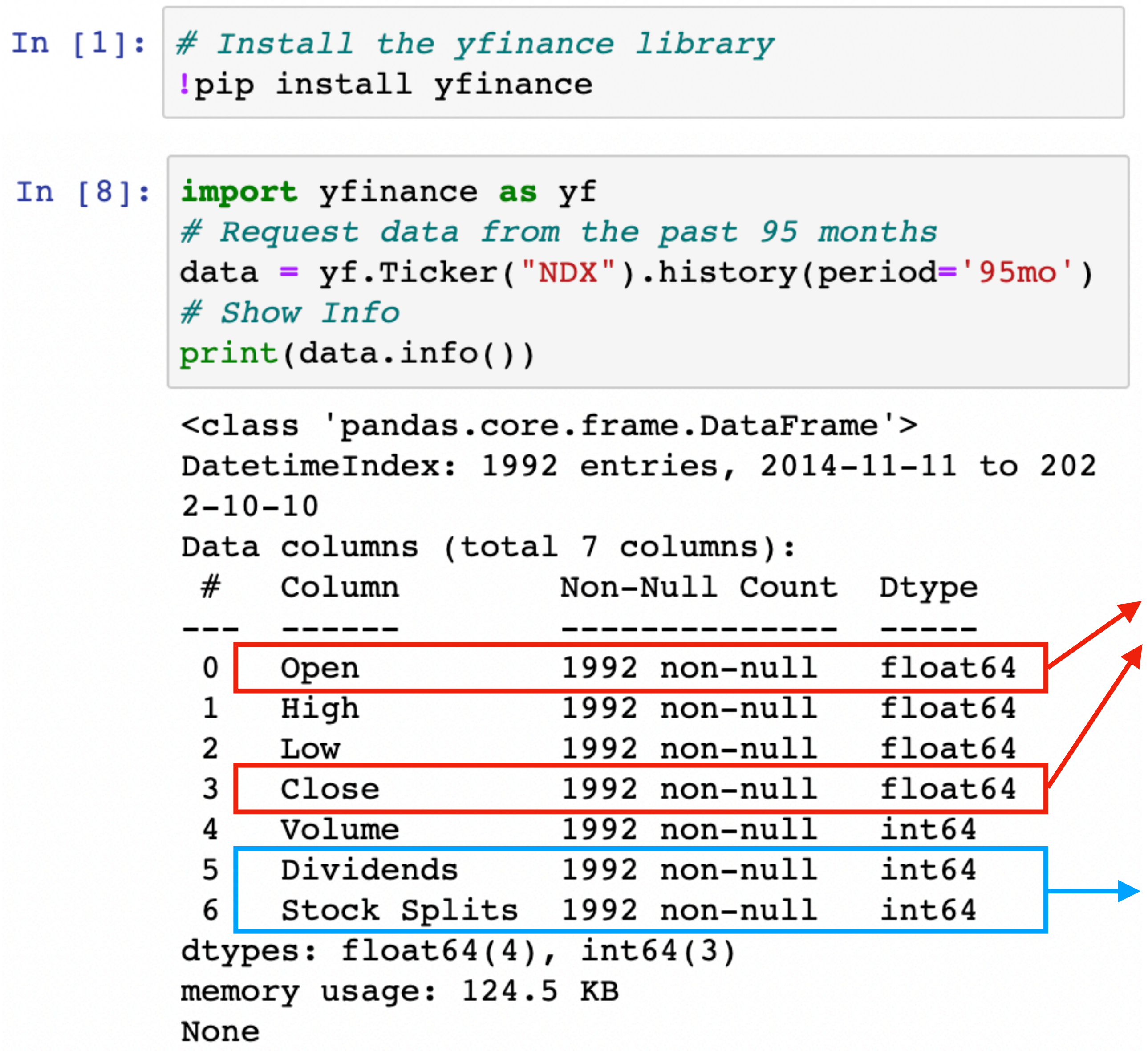
Section 3

Methodology

Methodology

Data collection and cleaning

We used the yfinance library to collect the data. The daily Nasdaq-100 Index data and monthly Nasdaq Composite Index, which contain a series of data, including **opening price**, **highest price**, **lowest price**, **closing price**, **volume**, **dividends** and **stock splits**.



```
In [1]: # Install the yfinance library
        !pip install yfinance
```

```
In [8]: import yfinance as yf
        # Request data from the past 95 months
        data = yf.Ticker("NDX").history(period='95mo')
        # Show Info
        print(data.info())
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 1992 entries, 2014-11-11 to 202
2-10-10
Data columns (total 7 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Open          1992 non-null   float64
 1   High          1992 non-null   float64
 2   Low           1992 non-null   float64
 3   Close         1992 non-null   float64
 4   Volume        1992 non-null   int64
 5   Dividends     1992 non-null   int64
 6   Stock Splits  1992 non-null   int64
dtypes: float64(4), int64(3)
memory usage: 124.5 KB
None
```

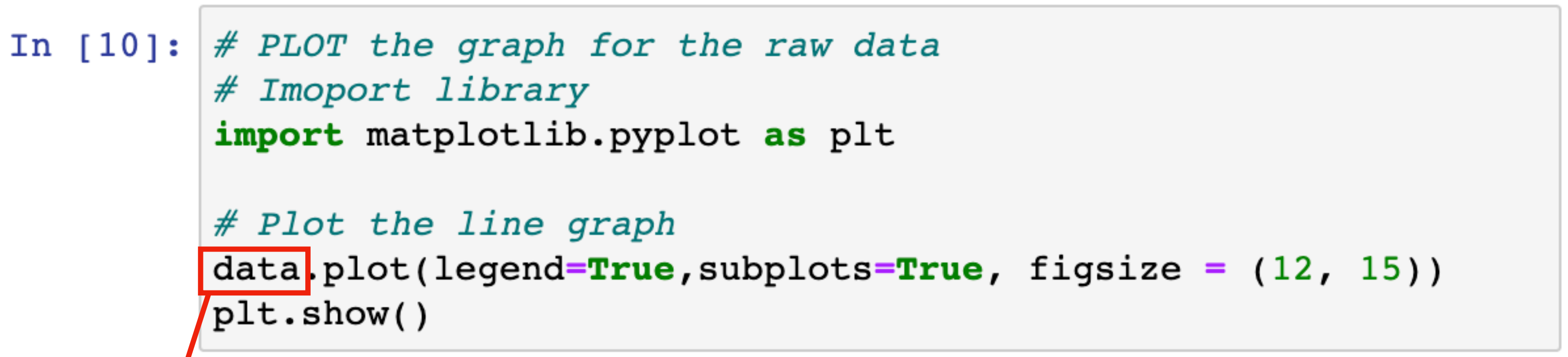
- **Further column calculations can be performed if we want to get the percentage change data**
- **Make sure to drop Nan for holidays etc.**

- **We are not using these two columns, so we have to create a new data frame to store values without them**

Methodology

Data visualization

We used the Matplotlib library to visualize the raw data.



```
In [10]: # PLOT the graph for the raw data
         # Imoport library
         import matplotlib.pyplot as plt

         # Plot the line graph
         data.plot(legend=True,subplots=True, figsize = (12, 15))
         plt.show()
```

Make sure that if we are plotting the graph with the whole dataset, the graph will be plotted separately by columns.

Methodology

The **autoregressive integrated moving average** (ARIMA) model is proposed for long-term data to fit the stock price pattern and market trend. For short-term patterns, we use a hybrid neural network model of **Convolutional Neural Network** (CNN) and **Long Short-Term Memory** (LSTM). The proposed approach is detailed in Figure 1.

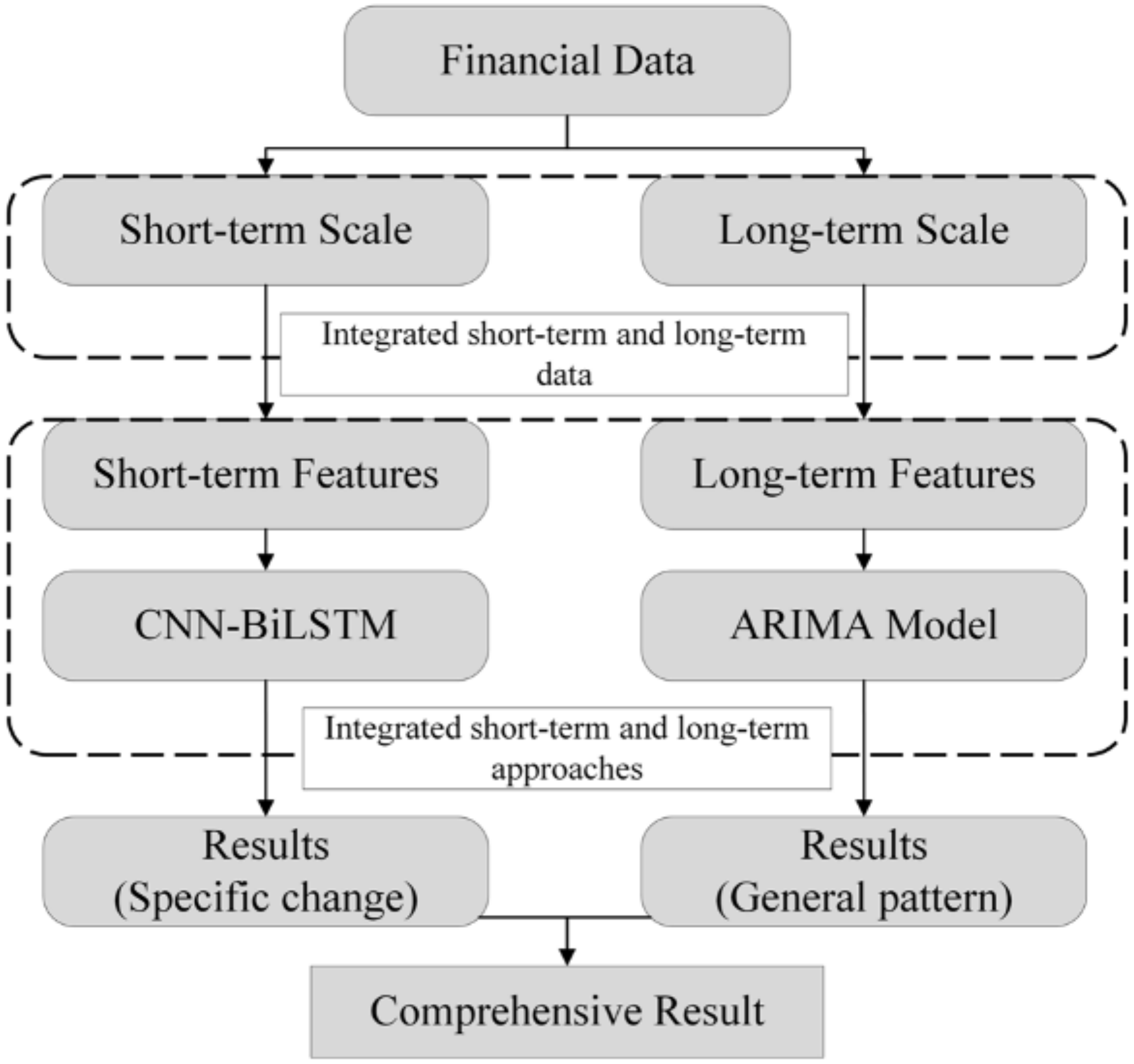

Financial Data
Short-term Scale
Long-term Scale
Integrated short-term and long-term data
Short-term Features
Long-term Features
CNN-BiLSTM
ARIMA Model
Integrated short-term and long-term approaches
Results (Specific change)
Results (General pattern)
Comprehensive Result

Figure 1. Process Summary.

Methodology

Forecasting of the daily stock price using neural network

In order to accurately predict the cyclical pattern and capture the short-term features, we propose a neural network approach built on **CNN-BiLSTM** to forecast the next-day stock closing price. The suggested method incorporates CNN and LSTM. The CNN layer can extract the local features of the stock time-series data, whereas the BiLSTM (forward LSTM layer and reverse LSTM layer) can find connections from the feedback of stock time-series data. Finally, the dense layer can fully receive outputs from the preceding BiLSTM layer for generating the final output. The model setup is given in Figure 2.

Methodology

CNN-BiLSTM Model

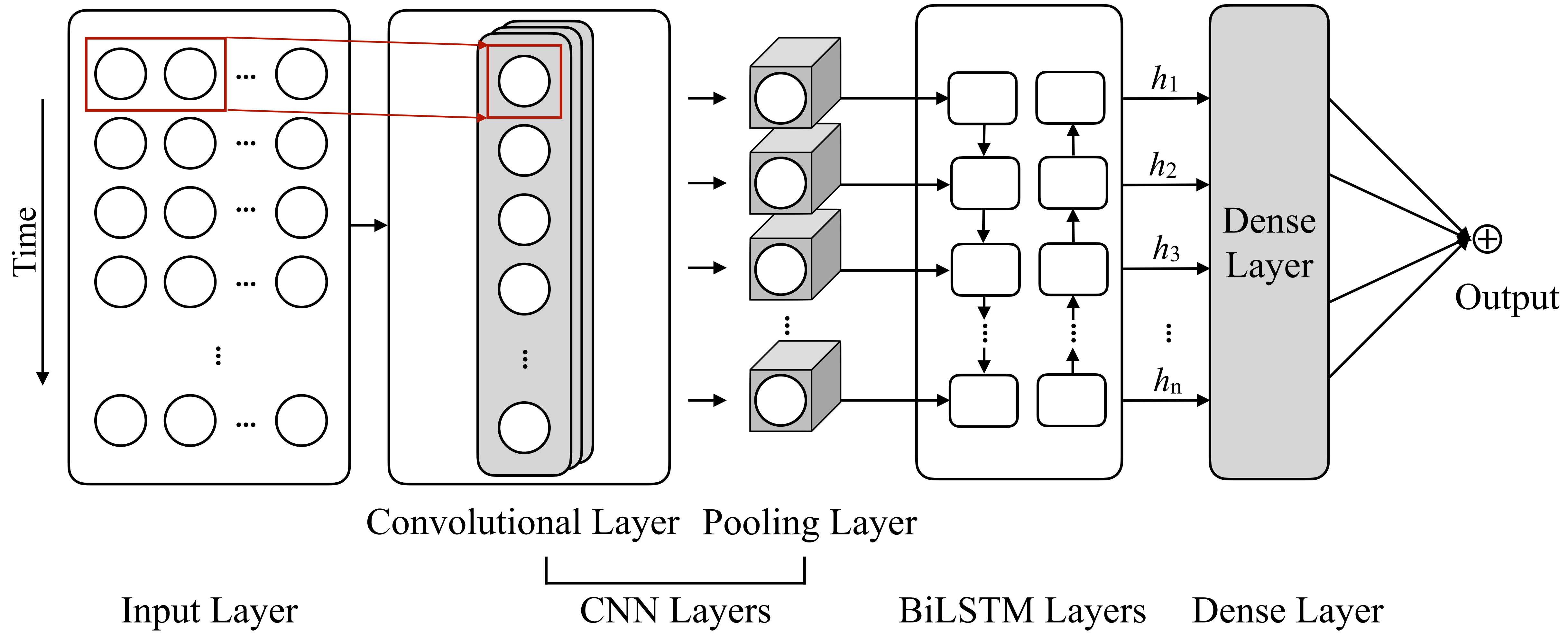
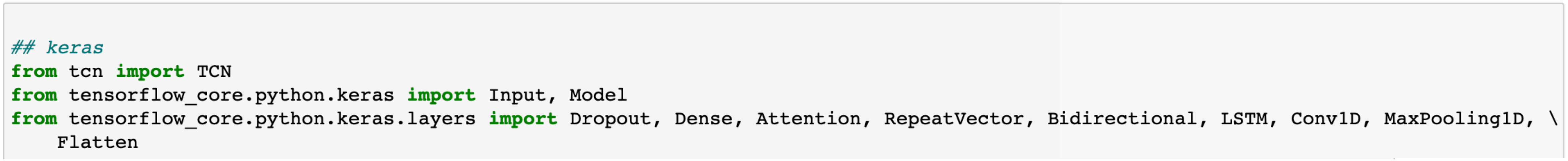


Figure 2. CNN-BiLSTM model setup diagram.

Methodology

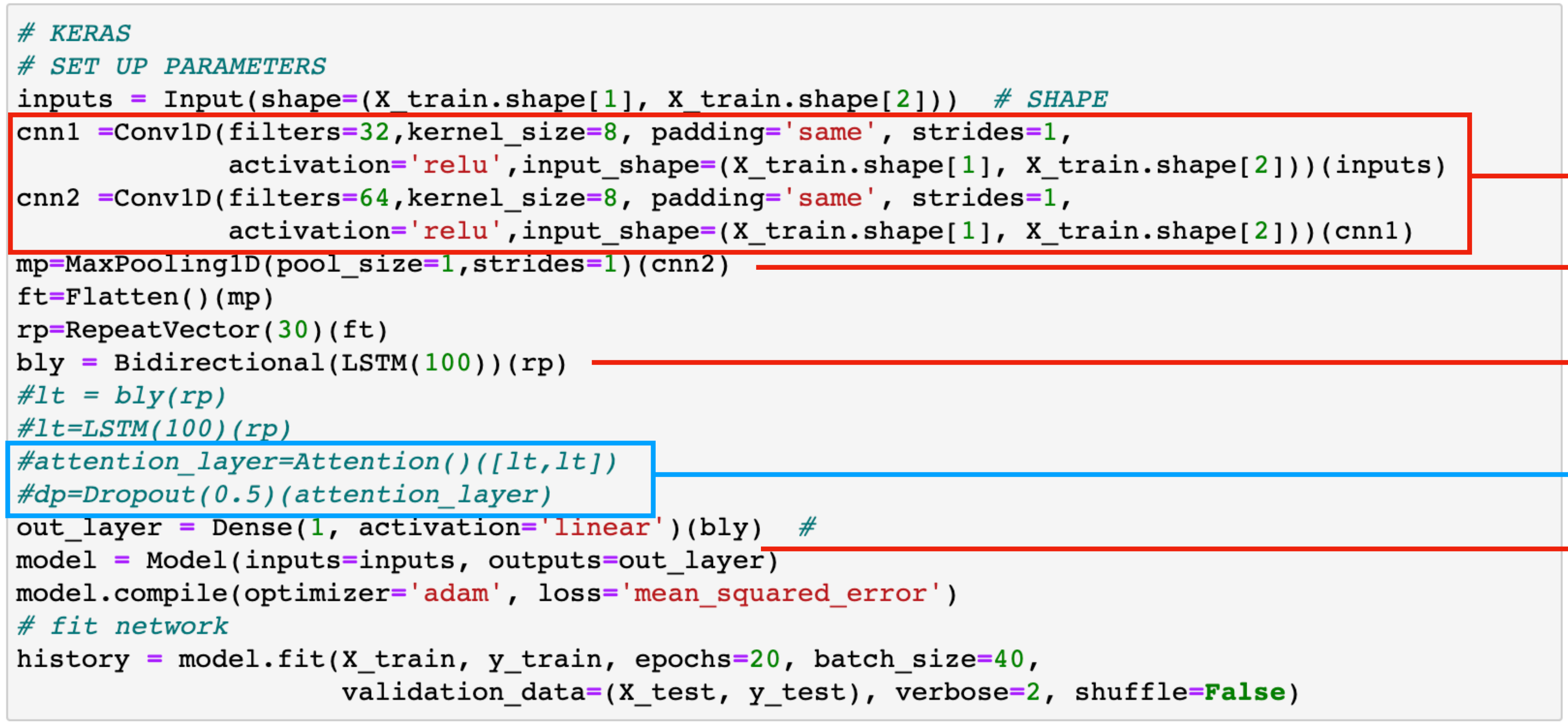
CNN-BiLSTM Model

- **Import Tensorflow, Keras library**



```
## keras
from tcn import TCN
from tensorflow_core.python.keras import Input, Model
from tensorflow_core.python.keras.layers import Dropout, Dense, Attention, RepeatVector, Bidirectional, LSTM, Conv1D, MaxPooling1D, \
    Flatten
```

- **Setup layers**



```
# KERAS
# SET UP PARAMETERS
inputs = Input(shape=(X_train.shape[1], X_train.shape[2]))  # SHAPE
cnn1 =Conv1D(filters=32,kernel_size=8, padding='same', strides=1,
             activation='relu',input_shape=(X_train.shape[1], X_train.shape[2]))(inputs)
cnn2 =Conv1D(filters=64,kernel_size=8, padding='same', strides=1,
             activation='relu',input_shape=(X_train.shape[1], X_train.shape[2]))(cnn1)
mp=MaxPooling1D(pool_size=1,strides=1)(cnn2)
ft=Flatten()(mp)
rp=RepeatVector(30)(ft)
bly = Bidirectional(LSTM(100))(rp)
#lt = bly(rp)
#lt=LSTM(100)(rp)
#attention_layer=Attention()([lt,lt])
#dp=Dropout(0.5)(attention_layer)
out_layer = Dense(1, activation='linear')(bly)  #
model = Model(inputs=inputs, outputs=out_layer)
model.compile(optimizer='adam', loss='mean_squared_error')
# fit network
history = model.fit(X_train, y_train, epochs=20, batch_size=40,
                    validation_data=(X_test, y_test), verbose=2, shuffle=False)
```

CNN layers

Pooling layer

BiLSTM layer

More layers can be added according to research interest

Dense layer

Methodology

Forecasting of the monthly stock price using hybrid ARIMA

As the volume of the financial data become extensive, we need a colossal amount of data to perform training, which does not fit the most satisfactory conditions for the neural network. Generally, the long-term data reveal the financial market trend, and it is considered to be more linear. Therefore, we propose using a **hybrid ARIMA** model to forecast the monthly stock price with the same dataset.

Methodology

Forecasting of the monthly stock price using hybrid ARIMA

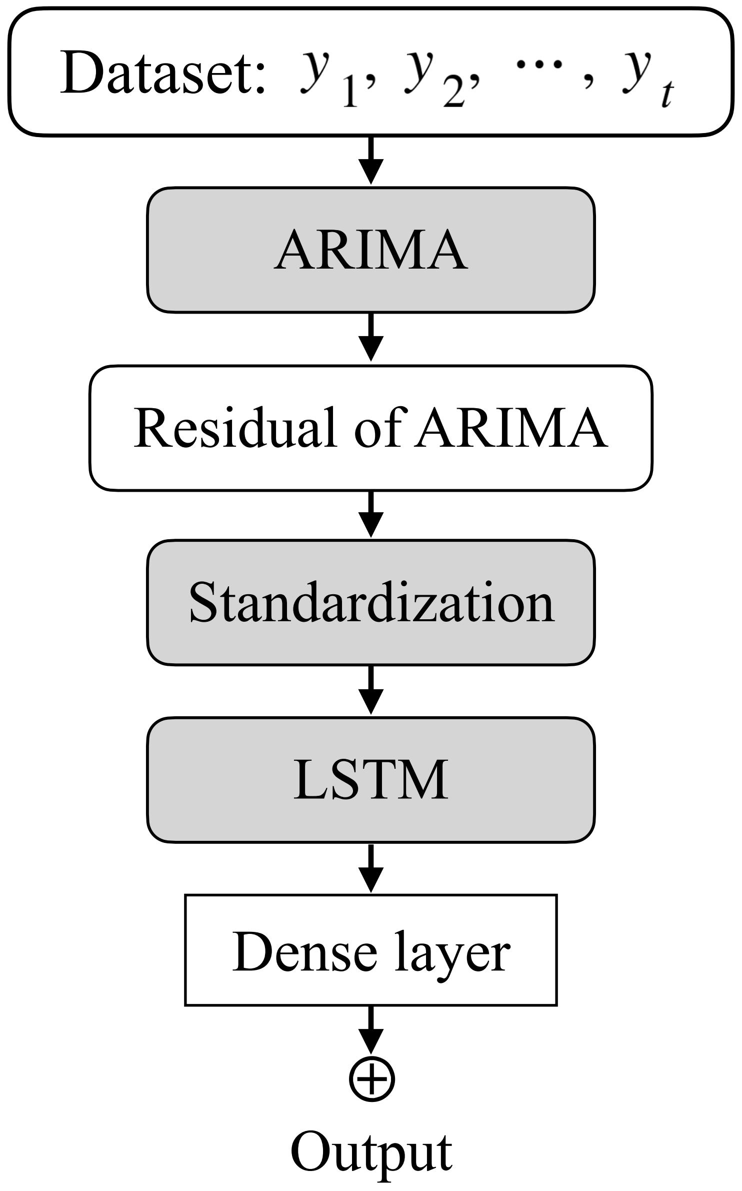
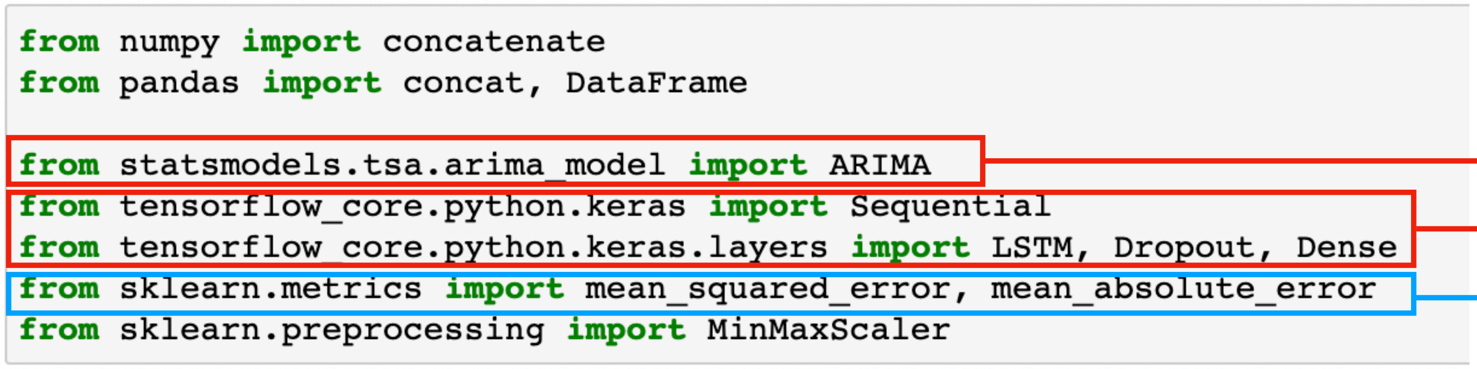


Figure 3. CNN-BiLSTM model setup diagram.

Methodology

ARIMA-LSTM Model

- **Import libraries**



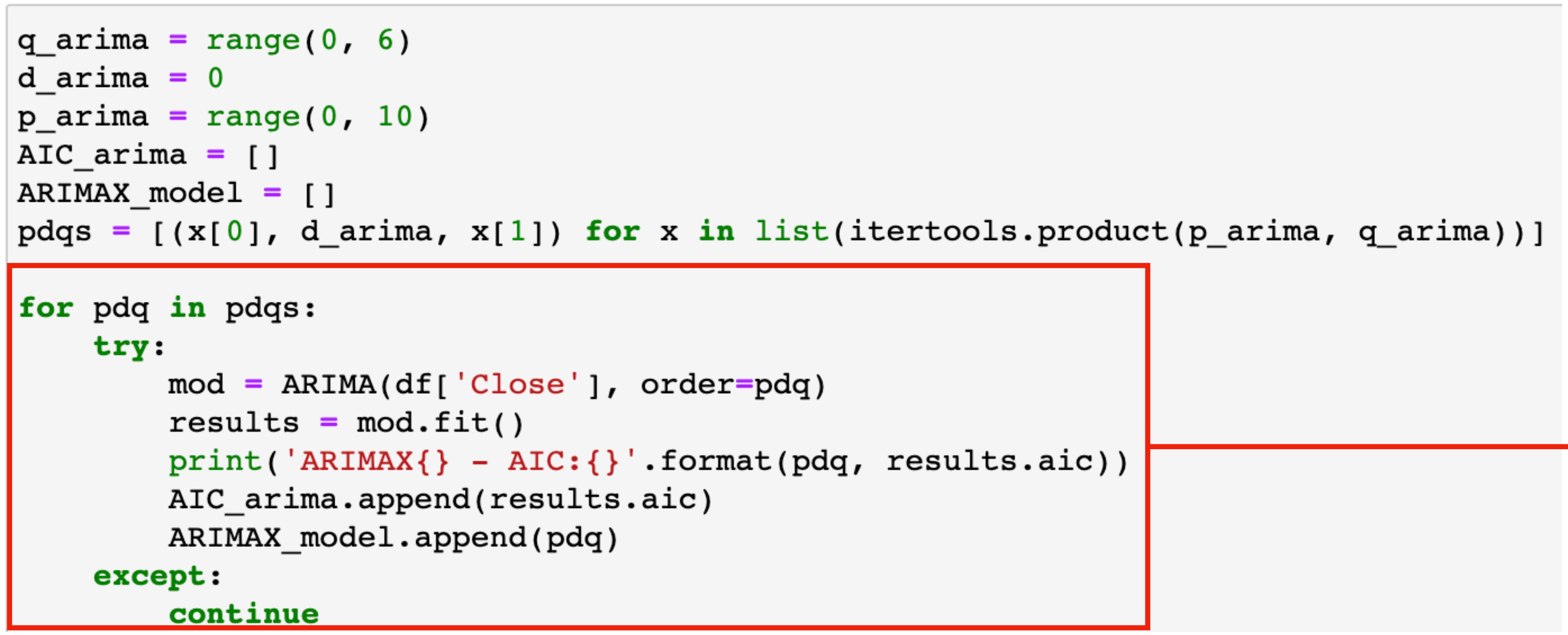
```
from numpy import concatenate
from pandas import concat, DataFrame

from statsmodels.tsa.arima_model import ARIMA
from tensorflow_core.python.keras import Sequential
from tensorflow_core.python.keras.layers import LSTM, Dropout, Dense
from sklearn.metrics import mean_squared_error, mean_absolute_error
from sklearn.preprocessing import MinMaxScaler
```

Import ARIMA model from Statsmodels

Import LSTM model from Tensorflow

Import Evaluation metrics

- **Optimized the ARIMA model**



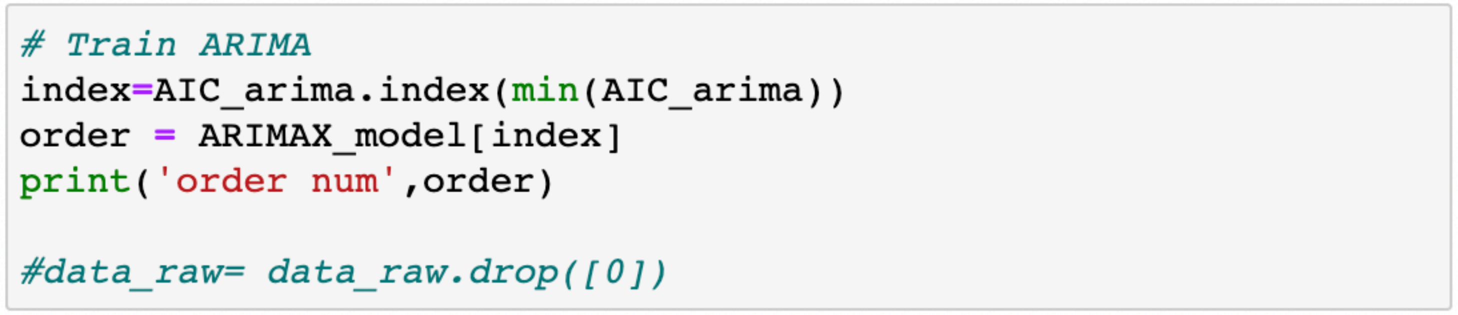
```
q_arima = range(0, 6)
d_arima = 0
p_arima = range(0, 10)
AIC_arima = []
ARIMAX_model = []
pdqs = [(x[0], d_arima, x[1]) for x in list(itertools.product(p_arima, q_arima))]

for pdq in pdqs:
    try:
        mod = ARIMA(df['Close'], order=pdq)
        results = mod.fit()
        print('ARIMAX{} - AIC:{}'.format(pdq, results.aic))
        AIC_arima.append(results.aic)
        ARIMAX_model.append(pdq)
    except:
        continue
```

Find the optimized p, d, q for the ARIMA model

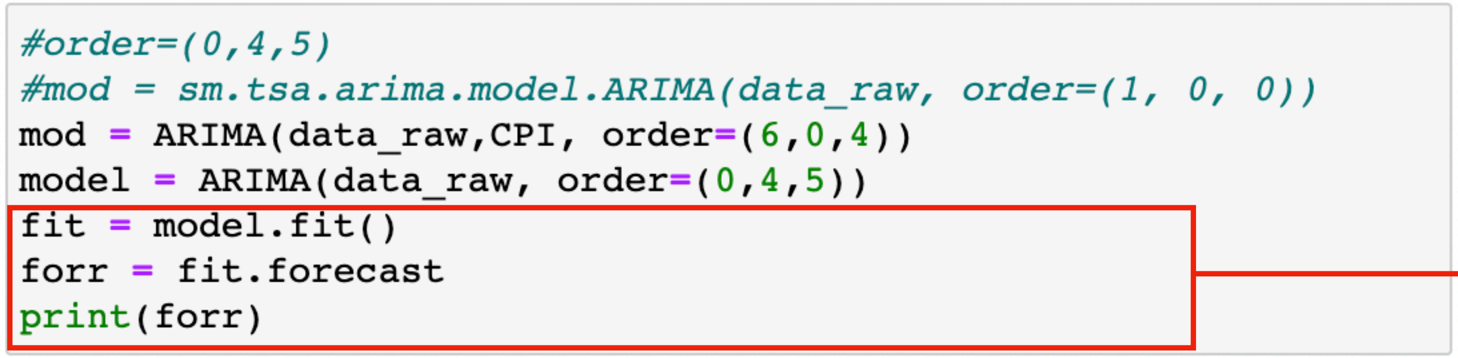
Methodology

ARIMA-LSTM Model

- **Setup ARIMA**



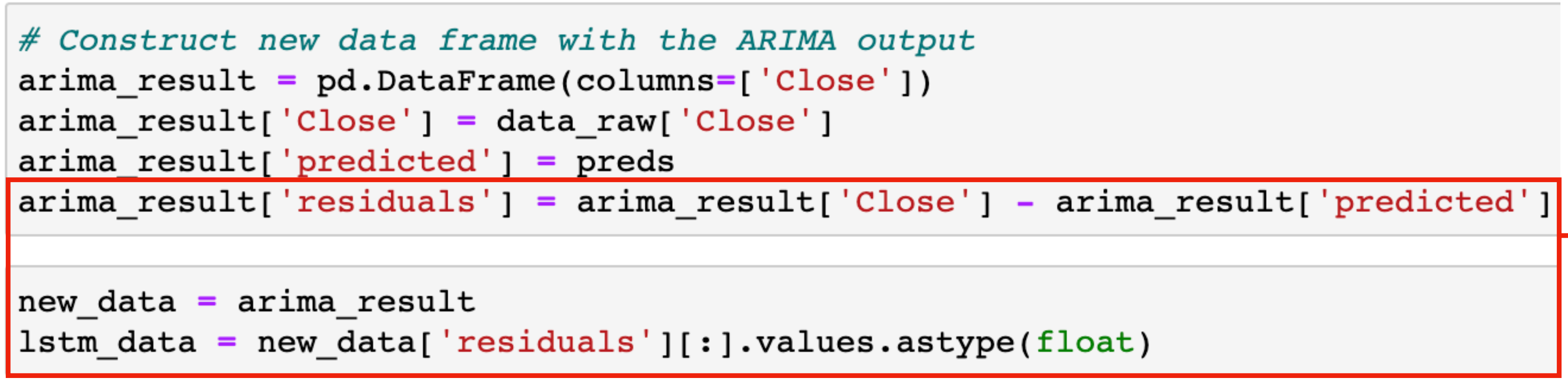
```
# Train ARIMA
index=AIC_arima.index(min(AIC_arima))
order = ARIMAX_model[index]
print('order num',order)

#data_raw= data_raw.drop([0])
```



```
#order=(0,4,5)
#mod = sm.tsa.arima.model.ARIMA(data_raw, order=(1, 0, 0))
mod = ARIMA(data_raw,CPI, order=(6,0,4))
model = ARIMA(data_raw, order=(0,4,5))
fit = model.fit()
forr = fit.forecast
print(forr)
```

Get the ARIMA forecast

- **Get the result**



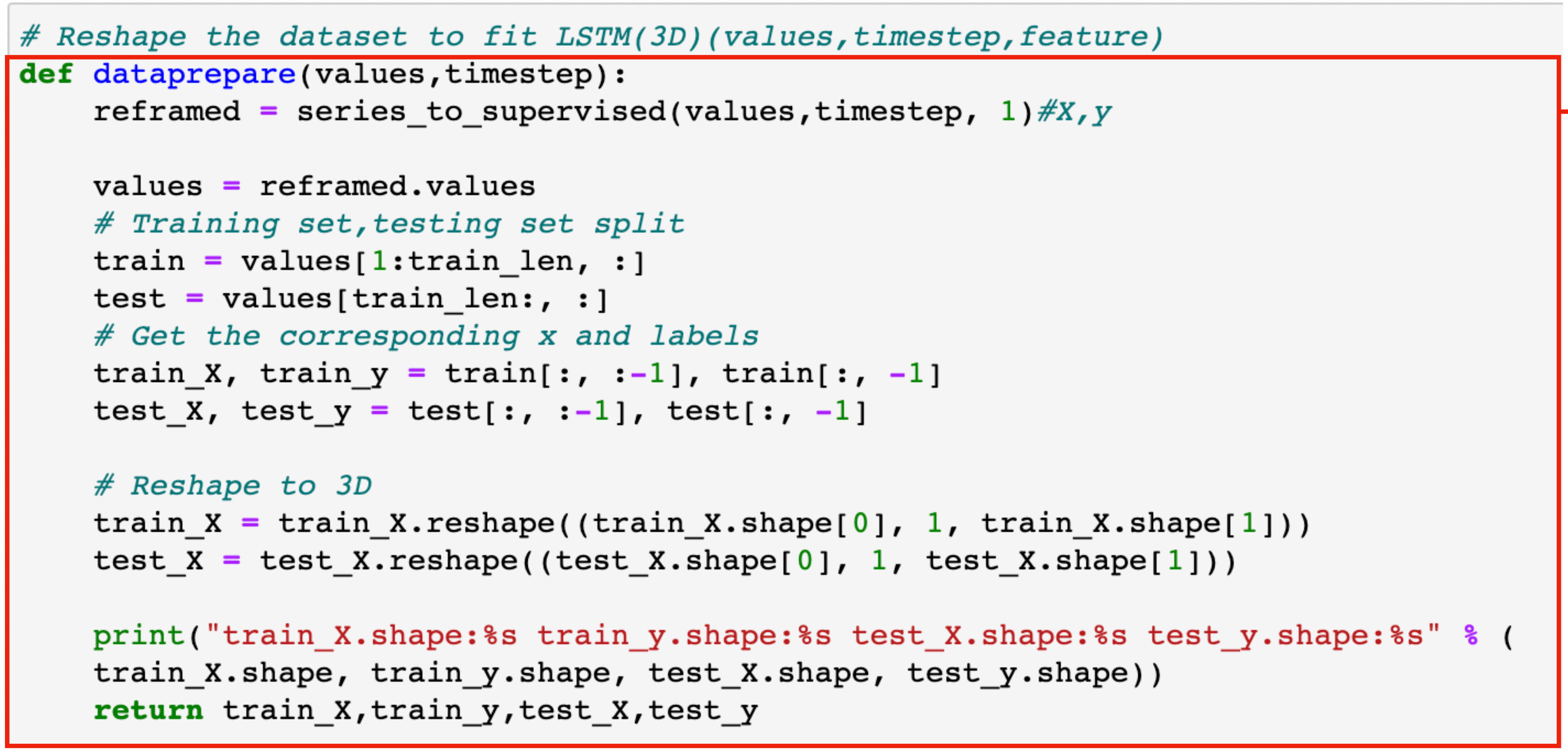
```
# Construct new data frame with the ARIMA output
arima_result = pd.DataFrame(columns=['Close'])
arima_result['Close'] = data_raw['Close']
arima_result['predicted'] = preds
arima_result['residuals'] = arima_result['Close'] - arima_result['predicted']

new_data = arima_result
lstm_data = new_data['residuals'][:].values.astype(float)
```

Calculate the residual of the prediction and use the column as the input of LSTM

Methodology

ARIMA-LSTM Model

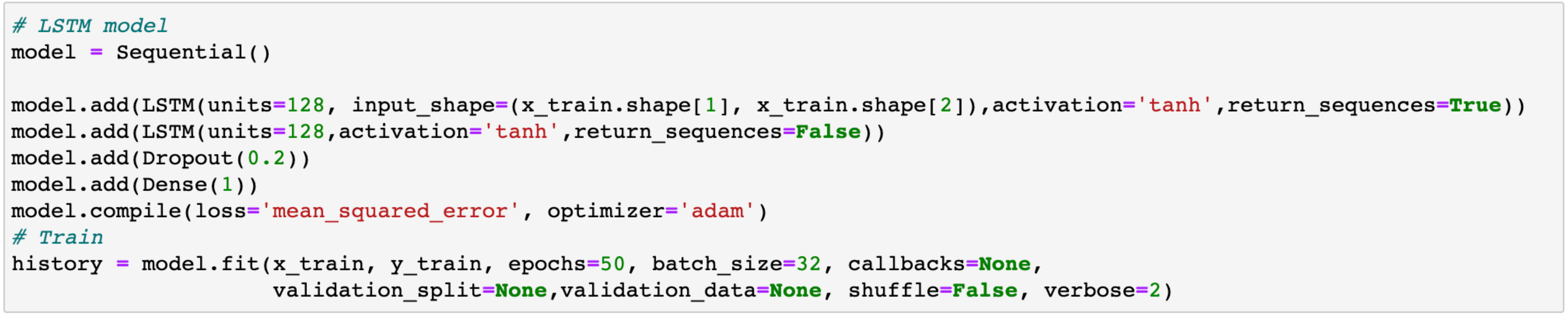
- **Data transform**



```
# Reshape the dataset to fit LSTM(3D)(values,timestep,feature)
def dataprepare(values,timestep):
    reframed = series_to_supervised(values,timestep, 1)#X,y

    values = reframed.values
    # Training set,testing set split
    train = values[1:train_len, :]
    test = values[train_len:, :]
    # Get the corresponding x and labels
    train_X, train_y = train[:, :-1], train[:, -1]
    test_X, test_y = test[:, :-1], test[:, -1]

    # Reshape to 3D
    train_X = train_X.reshape((train_X.shape[0], 1, train_X.shape[1]))
    test_X = test_X.reshape((test_X.shape[0], 1, test_X.shape[1]))

    print("train_X.shape:%s train_y.shape:%s test_X.shape:%s test_y.shape:%s" % (
    train_X.shape, train_y.shape, test_X.shape, test_y.shape))
    return train_X,train_y,test_X,test_y
```

We need to reshape the ARIMA output as to fit in the LSTM input data format

- **Setup LSTM layer**



```
# LSTM model
model = Sequential()

model.add(LSTM(units=128, input_shape=(x_train.shape[1], x_train.shape[2]),activation='tanh',return_sequences=True))
model.add(LSTM(units=128,activation='tanh',return_sequences=False))
model.add(Dropout(0.2))
model.add(Dense(1))
model.compile(loss='mean_squared_error', optimizer='adam')
# Train
history = model.fit(x_train, y_train, epochs=50, batch_size=32, callbacks=None,
                    validation_split=None,validation_data=None, shuffle=False, verbose=2)
```

Section 4

Experiments and Results

Experiments and Results

Parameters (CNN-BiLSTM)

Parameters	Value
Convolutional layer filters	64
Convolutional layer kernel size	1
Convolutional layer padding	Same
Convolutional layer activation function	ReLU
Max pooling layer pool size	1
BiLSTM layer units	40

Table 1. Parameter specification of the proposed CNN-BiLSTM model.

Models	Details
$LSTM_1$	LSTM layer with 100 units
$LSTM_2$	LSTM layer with 200 units
$BiLSTM_1$	BiLSTM layer with 100 units
$BiLSTM_2$	BiLSTM layer with 200 units
CNN-LSTM	Convolutional layer with 64 filters Max pooling layer with size 1 LSTM layer with 100 units
CNN-BiLSTM	Convolutional layer with 64 filters Max pooling layer with size 1 BiLSTM layer with 40 units

Table 2. Parameter specification of all neural network models.

Experiments and Results

Parameters (ARIMA-LSTM)

Parameters	Value
Order of the autoregressive: p	0
Degree of differencing: d	4
Order of the moving average:	5
LSTM layer units	128

Table 3. Parameters specification of the proposed ARIMA-LSTM model.

Experiments and Results

Evaluation Criteria

- Mean absolute error (**MAE**)

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i|$$

- Root mean square error (**RMSE**)

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}$$

- R-square (**R²**)

$$R^2 = 1 - \frac{SS_{RES}}{SS_{TOT}} = 1 - \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{n} (y_i - \bar{y}_i)^2}$$

Experiments and Results

Experiment results (CNN-BiLSTM)

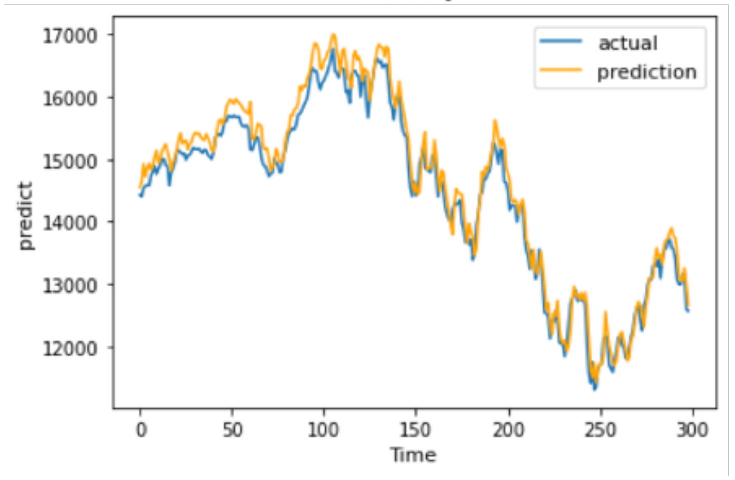
The orange curve in each model is the forecast result while the blue curve represents the actual stock price. The x-axis denotes the time, and the y-axis denotes the stock price in United States Dollars. The degree of fitting of the broken-line graphs between actual observation values and predicted values are ranked from low to high. More specifically, the $LSTM_1$ model has achieved the lowest degree of fitting of the broken-line graphs between actual observation values and predicted values, then $BiLSTM_2$, $LSTM_2$, $BiLSTM_1$, CNN-LSTM, and CNN-BiLSTM. The **CNN-BiLSTM** model showed the predicted values almost perfectly fitting with the actual observations.

Table 2. Parameter specification of all neural network models.

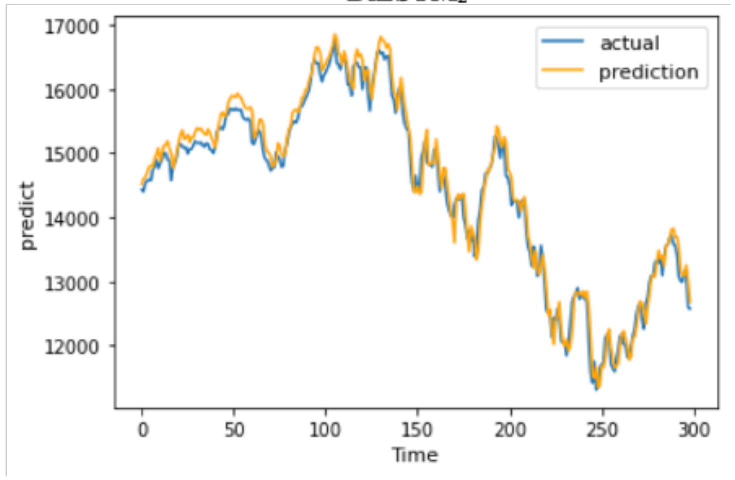
Experiments and Results

Experiment results (CNN-BiLSTM)

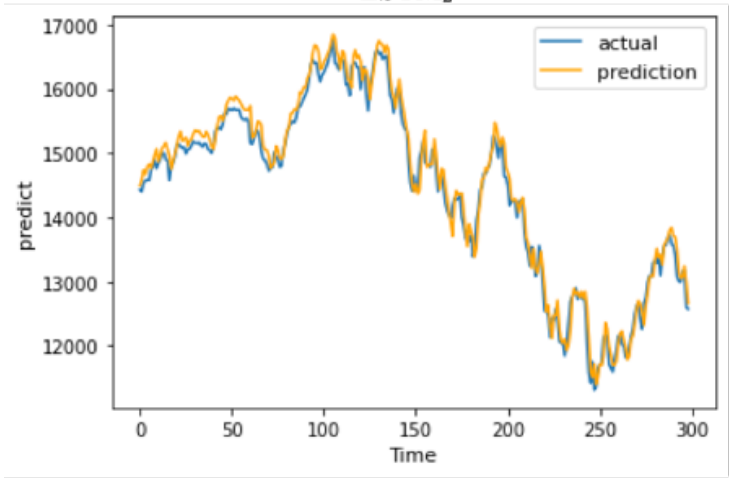
LSTM$_1$



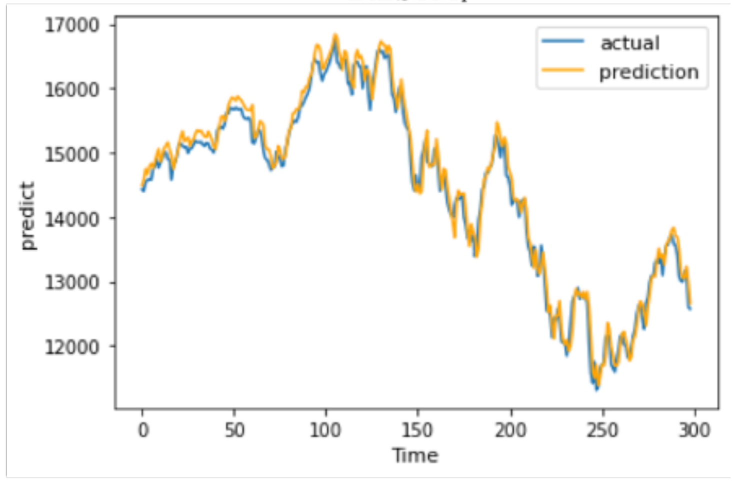
BiLSTM$_2$



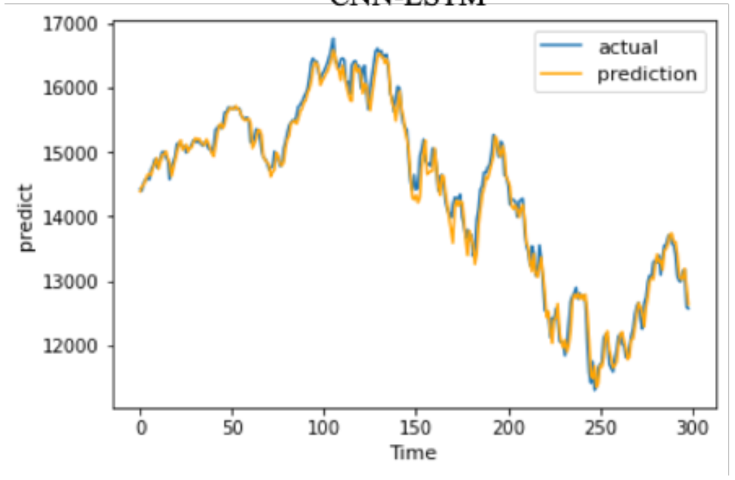
LSTM$_2$



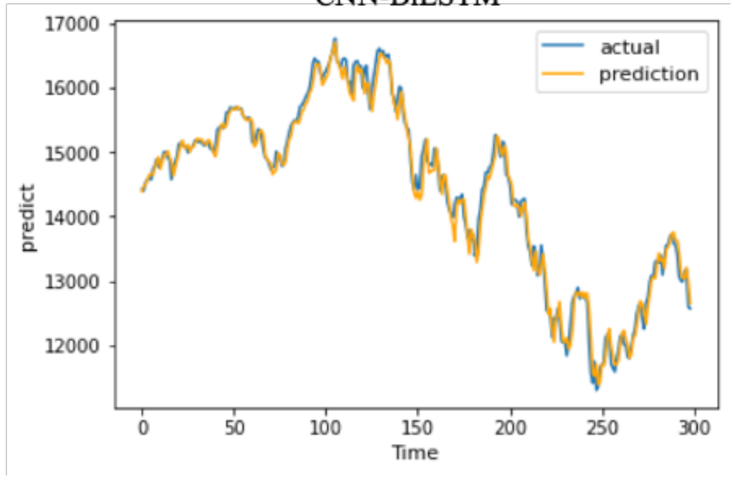
BiLSTM$_1$



CNN-LSTM



CNN-BiLSTM



Experiments and Results

Experiment results (CNN-BiLSTM)

Models	MAE	RMSE	R^2
$LSTM_1$	211.158	253.405	0.96749
$BiLSTM_2$	166.655	204.117	0.97890
$LSTM_2$	162.994	199.605	0.97983
$BiLSTM_1$	159.250	196.269	0.98049
CNN-LSTM	130.299	174.116	0.98465
CNN-BiLSTM	129,252	173.724	0.98472

Table 4. Comparison of the evaluation criteria of neural network models.

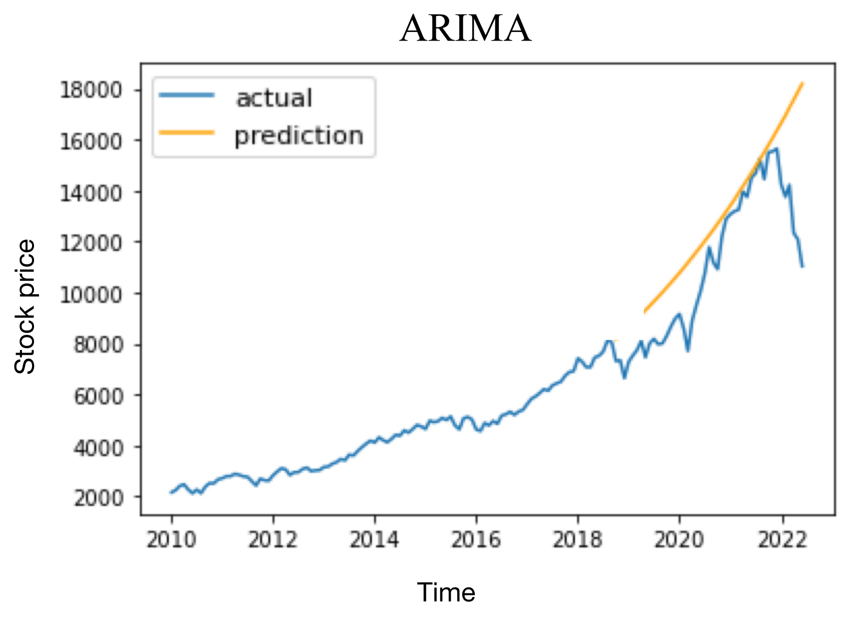
Experiments and Results

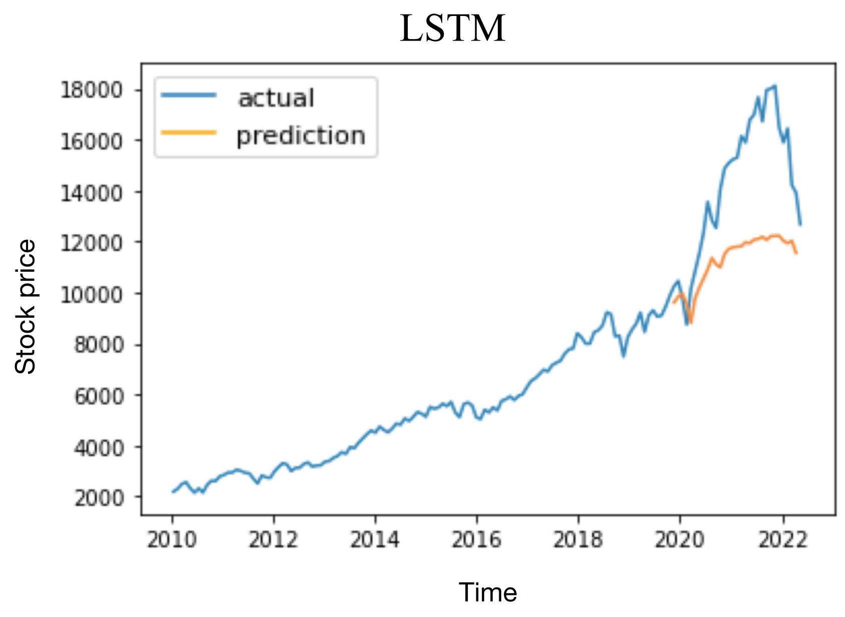
Experiment results (ARIMA-LSTM)

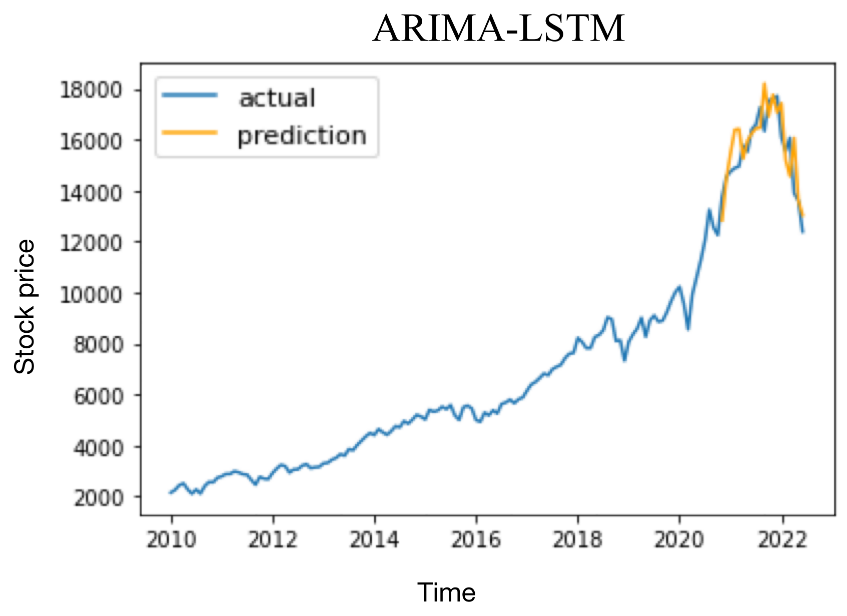
The orange curve in each model is the forecast result, while the blue curve represents the actual stock price. The x-axis denotes the time, and the y-axis denotes the stock price in United States Dollars. From the degree of fitting of the broken-line graphs between actual observation values and predicted values, the optimal forecast was achieved by using the **ARIMA-LSTM model**. The degree of fitting in the graph displayed that the ARIMA model had failed in capturing nonlinear variations of the time series. Moreover, the LSTM model was used only for monthly stock closing price forecasting, and the neural network model also failed to perform an accurate forecast with a small amount of data, despite the discoverable trend displayed in the graph.

Experiments and Results

Experiment results (ARIMA-LSTM)







Experiments and Results

Experiment results (ARIMA-LSTM)

Models	RMSE
LSTM	2688.831
ARIMA	1487.514
ARIMA-LSTM	878.532

Table 5. Comparison of the evaluation criteria of ARIMA models.

Section 5

Conclusion

Conclusion

In this project, two major models were used in Nasdaq index stock price forecasting. We proposed a **neural network model** to predict the daily price of Nasdaq-100 on a short-term scale, represented the daily stock price of the most representative Nasdaq-listed stocks, and **an improved ARIMA model** was proposed to study the stock price forecasting from the Nasdaq Composite in long-term scale, represented the monthly performance of all stocks listed on the Nasdaq stock exchange. The attainability of the integrated models on stock price forecasting is shown by the result, which showed a comparatively satisfying accuracy.

Moreover, the integrated forecasting model can provide results that reflect the pattern from different perspectives, which helps investors better understand the market trend and guide their decisions. The experimental analysis indicated that although the neural network models are broadly used in financial forecasting, the utilization of feature processing with neural networks in statistical models also showed noticeable potential. It is worth noting that due to the market's volatility, more economic indicators could be added as the training features for the model to improve the forecasting ability further.

Conclusion

Our future work will improve the proposed model's forecasting accuracy by adding different features with long-term and short-term data, such as major stock market indices, gold prices, and other economic indicators. Furthermore, we plan to closely inspect the short-term fluctuations and long-term trends of financial data to explore the significance of the correlation between them.

Finally, more optimized configurations of the proposed CNN-BiLSTM will be another primary concern, as parameters should be continually adjusted to achieve a forecast performance.

Thank you!