1. Problem description

- Finance stock market generates trading data every day.
- The project is considering leveraging a machine learning model to predict the stock price in the future.
- There could be multiple factors impacting the daily stock price, such as marco ecnomic, subject motions etc. in order to build up a prediction model successfully, I select SPX (index for top 500 stocks minimized the individual uncertain impacts) as the target to build up the prediction model.

2. Data set

- I downloaded the SPX historic data form yahoo finance. It has history data since year 1927. Considered the correlation of the economic environment, I picked up the daily price data between year 2010 and recent (3/20/2025).
- The sample of data are show in the following list
- I got 3827 sample data downloaded.

Date	Open	High	Low	Close	Adj Close	Volume
3/20/25	5646.92	5711.15	5632.33	5672.83	5672.83	1257423003
3/19/25	5632.37	5715.33	5622.20	5675.29	5675.29	4660090000
3/18/25	5654.53	5654.53	5597.76	5614.66	5614.66	4666270000
3/17/25	5635.60	5703.52	5631.12	5675.12	5675.12	5031770000
3/14/25	5563.85	5645.27	5563.85	5638.94	5638.94	4863180000
3/13/25	5594.45	5597.78	5504.65	5521.52	5521.52	5018980000
3/12/25	5624.84	5642.19	5546.09	5599.30	5599.30	5219830000
3/11/25	5603.65	5636.30	5528.41	5572.07	5572.07	6221240000
3/10/25	5705.37	5705.37	5564.02	5614.56	5614.56	6409370000
3/7/25	5726.01	5783.01	5666.29	5770.20	5770.20	5705140000
3/6/25	5785.87	5812.08	5711.64	5738.52	5738.52	5165080000

Shape of the DataFrame: data: (3827, 7)

- Visualized the data by following chart.

2.1. Visualization:

SPX price by day

3. Challenges

- From the data visualization chat, observably, it is a time serial data
- When do the price prediction, the prices in the past days could impact the price to the next day differently, the price of last day may have bigger correlation to the next day, the price of the day before yesterday may have smaller correlation etc.
- It is difficult to draw a line to close all sample data as mu as possible. A simple linear regression model cannot be a good choice.

4. Strategy

- Considered that stock trading happens every working day. The small cycle is by week. And investors references the prices in this and last week often to make trading price policy.
- I made decision to set the time window by 5 days, which is leveraging the price in the past 5 days to predict the prices in next 5 days in the future. Which clarified the project as a sequence to sequence prediction problem
- As I choose 5 days' price in the past days to predict future 5 days' price, the input and output has same length
- Considering models, the possible candindates are LSTM and rnn. I developed the code for the two models and compared the result following.

5. Result:

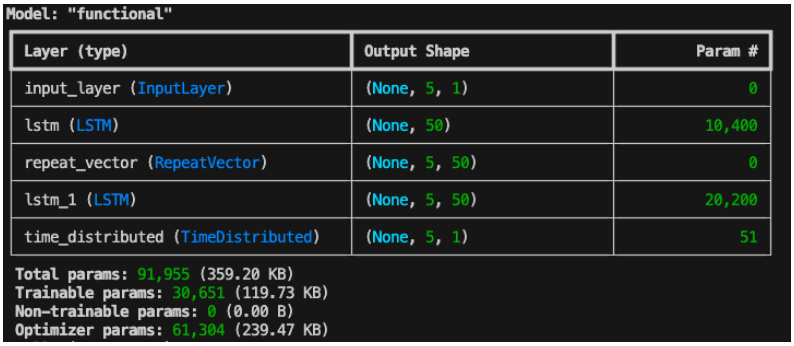
5.1. Model 1: LSTM + Encoder-Decoder model

I combined LSTM with encoder and decoder architecture, in fact it is not necessary. As I have the same length of input and output (5 data). The model is good for the use case that when the past days is difference from future days such as leveraging 10 past_days' prices to predict the price in coming 5 days (input is 10 but out put is 5).

5.1.1. code:

```
def LSTMEncoder_Decoder(epochs, batch_size, X_train, y_train, X_test, y_test) :
  # LSTM Encoder-Decoder Model
  latent_dim = 50

  # Encoder
  encoder_inputs = Input(shape=(past_days, 1))
  encoder = tf.keras.layers.LSTM(latent_dim, activation="relu", return_sequences=False)(encoder_inputs)
  encoded = RepeatVector(future_days)(encoder)

  # Decoder
  decoder = tf.keras.layers.LSTM(latent_dim, activation="relu", return_sequences=True)(encoded)
  decoder_outputs = TimeDistributed(Dense(1))(decoder)

  # Define model
  model = Model(encoder_inputs, decoder_outputs)
  model.compile(optimizer="adam", loss="mse")

  # Train model
  with tf.device('/gpu:0'):
   history = model.fit(X_train, y_train, epochs=epochs, batch_size=batch_size, validation_data=(X_test, y_test))

  return history, model
```

5.1.2. Model summary()



Model: "functional"

Layer (type)	Output Shape	Param #
input_layer (InputLayer)	(None, 5, 1)	0
lstm (LSTM)	(None, 50)	10,400
repeat_vector (RepeatVector)	(None, 5, 50)	0
lstm_1 (LSTM)	(None, 5, 50)	20,200
time_distributed (TimeDistributed)	(None, 5, 1)	51

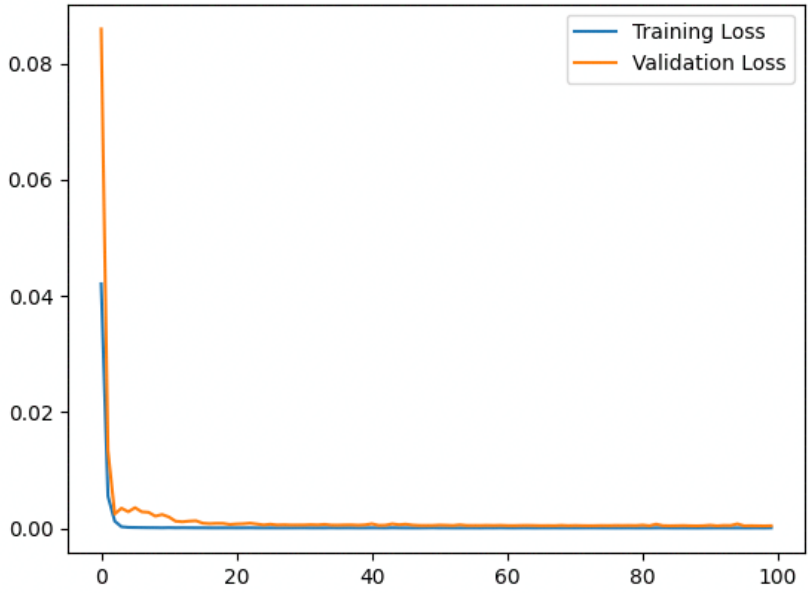
Total params: 91,955 (359.20 KB)
Trainable params: 30,651 (119.73 KB)
Non-trainable params: 0 (0.00 B)
Optimizer params: 61,304 (239.47 KB)

5.2. Hyperparameters

Hyper parameter	Value	Comments
Encoder		

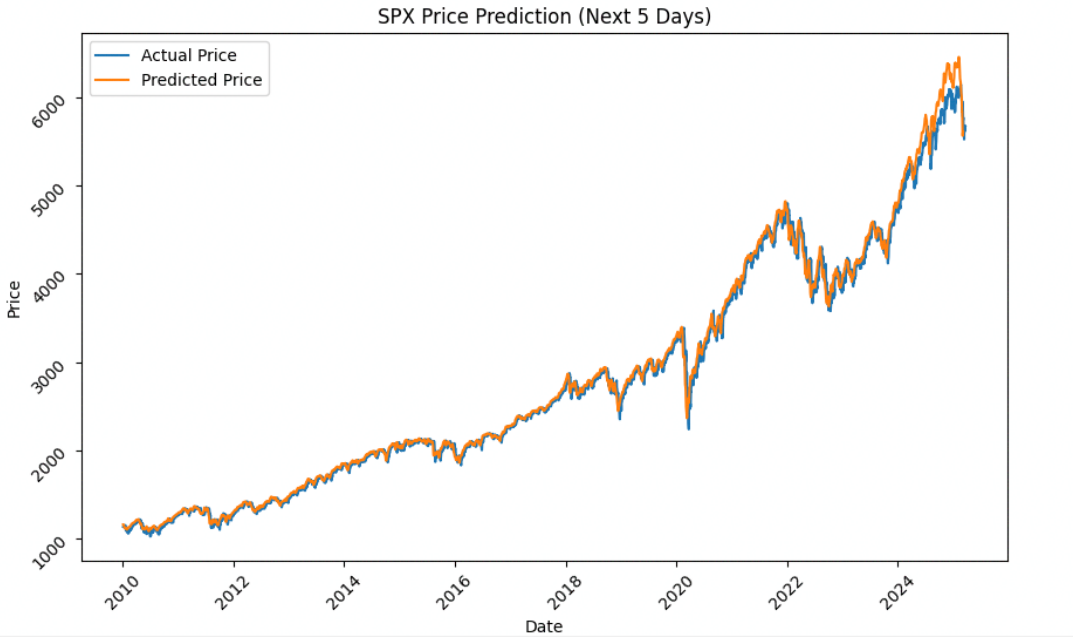
LSTM layer	1	
Units in LSTM layer	50	
Decoder		
LSTM layer	1	
Units in LSTM layer	50	
Optimizer	Adam	
Loos	MSE	
EPOCHS	100	
Batch_size	64	

5.3. Loss



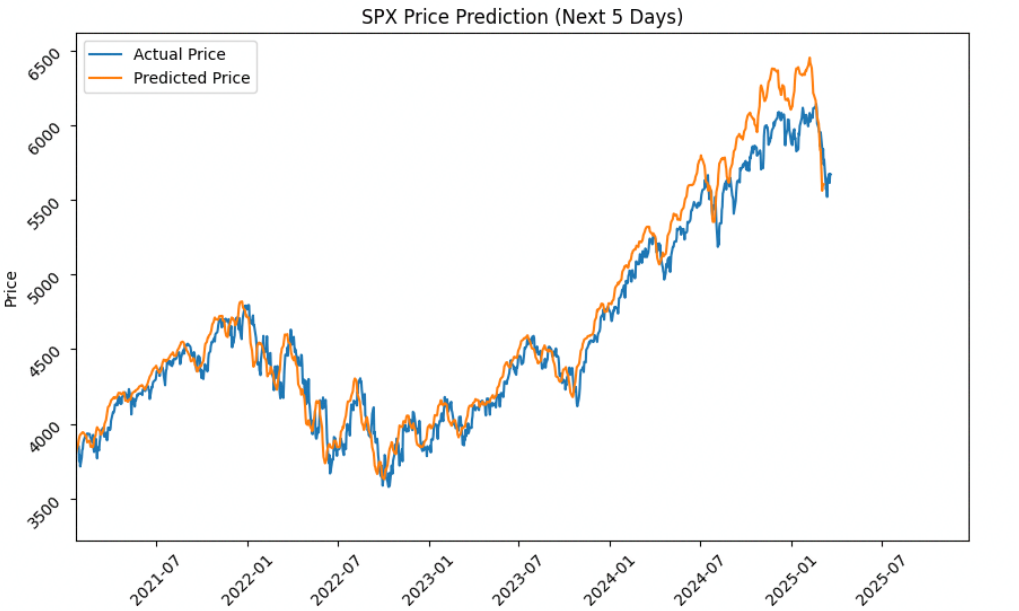
From the result of loss function by each epoch in above chart, the model fitted well, neither overfitting nor under fitting.

5.4. Prediction



Compare the predicted price and the actual price, the predicated price match with the acctual value well. The error is small by 0.00017429136216920926 (MSE)

5.5. Zoom-in



5.6. Model 2: LSTM stack model

As I take 5 past_days' price to predict 5 futur day's price. I can simplify the model by LSTM directly. Compared between one layer and stacked LSTM model, there is no big different in between.

5.6.1. Code

```
def LSTM(epochs, batch_size, X_train, y_train, X_test, y_test) :

latent_dim = 50

model = Sequential([
tf.keras.layers.LSTM(latent_dim, activation="relu", input_shape=(past_days, 1), return_sequences=True),
tf.keras.layers.LSTM(latent_dim, activation="relu", input_shape=(latent_dim, 1), return_sequences=False),
Dense(25, activation="relu"),
Dense(future_days) #
])

model.compile(optimizer="adam", loss="mse")
# Train model
with tf.device('/gpu:0'):
history = model.fit(X_train, y_train, epochs=epochs, batch_size=batch_size, validation_data=(X_test, y_test))
```

5.6.2. Model.summary()

Model: "sequential"

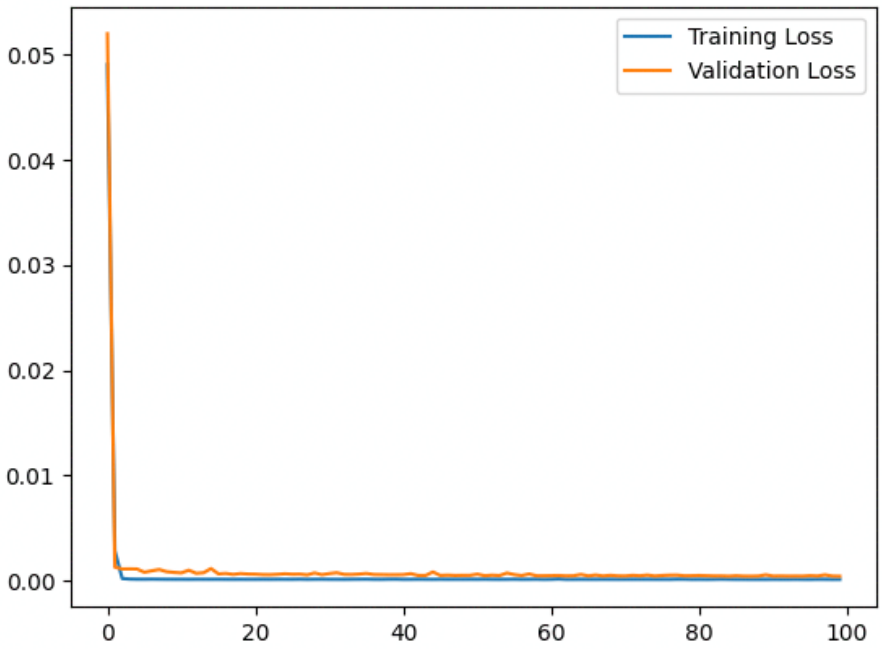
Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 5, 50)	10,400
lstm_1 (LSTM)	(None, 50)	20,200
dense (Dense)	(None, 25)	1,275
dense_1 (Dense)	(None, 5)	130

Total params: 96,017 (375.07 KB)
Trainable params: 32,005 (125.02 KB)
Non-trainable params: 0 (0.00 B)
Optimizer params: 64,012 (250.05 KB)

5.6.3. Hyperparameters

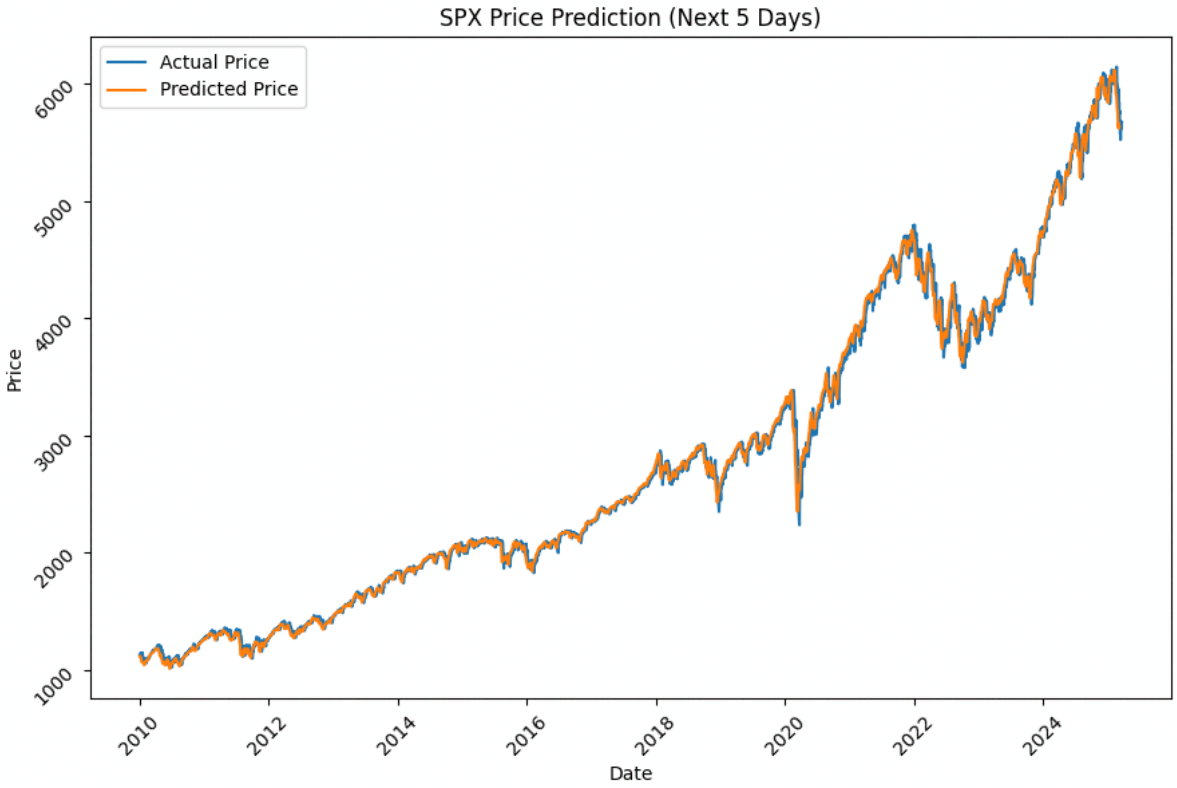
Hyper parameter	Value	Comments
LSTM layer	2	Not big impact when add the second layer of LSTM
Units in LSTM layer	50	
Optimizer	Adam	
Loos	MSE	
EPOCHS	100	
Batch_size	64	

5.6.4. Loss function result



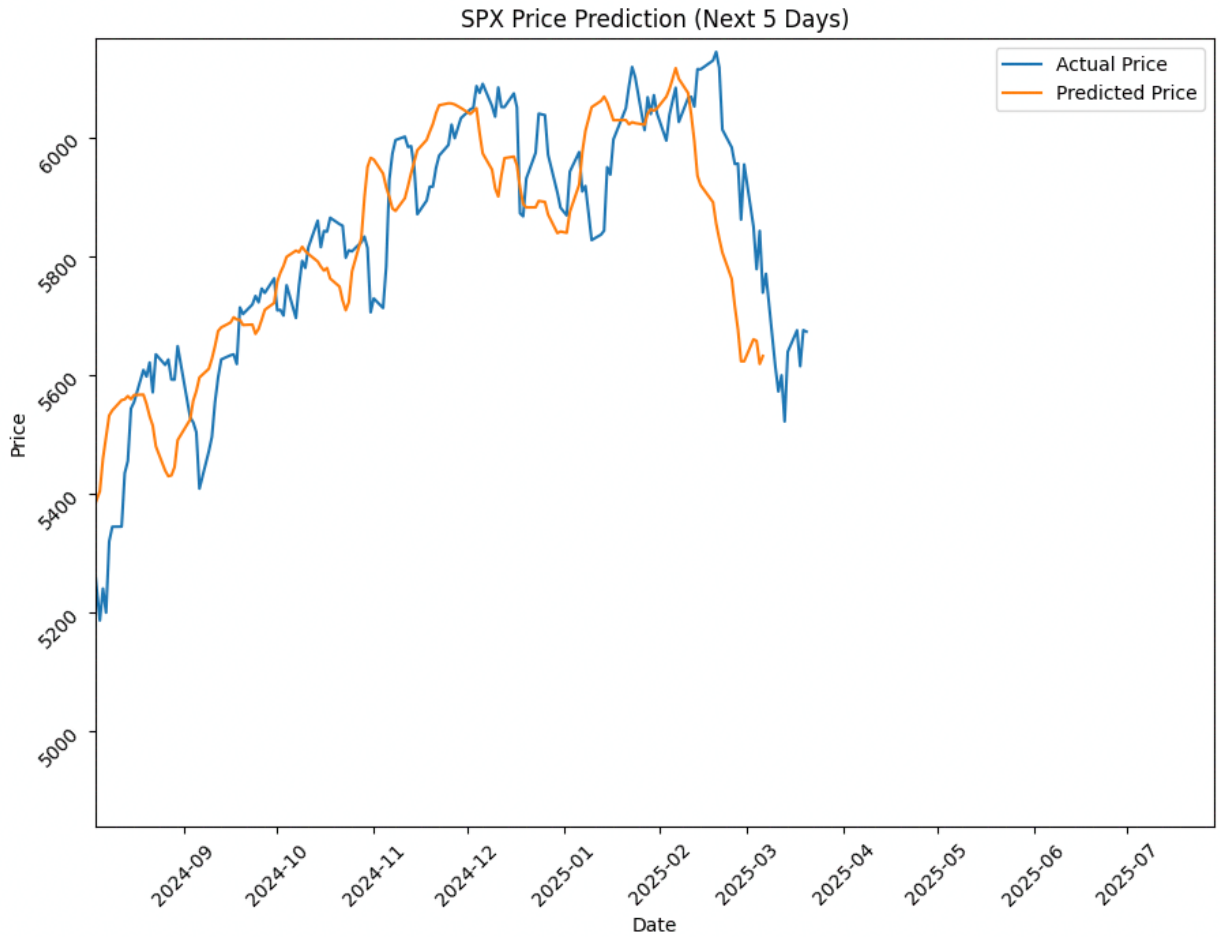
From the result of loss function by each epoch in above chart, the model fitted well, neither overfitting nor under fitting.

5.6.5. Prediction result



Compare the predicted price and the actual price, the predicated price match with the acctual value well. The error is small by 0.12997036388668592 (MSE)

5.6.6. Zoom-in



Zoom in the prediction chart shows the gap between prediction and actual data.