Predict SPX price by Deep Learning Models

Arthur Wang (arthurwhg@gmail.com)

(a project for class: AISV.X401 - Deep Learning and Artificial Intelligence)

Table of Contents

1. Problem description

- Finance stock market generates trading data every day.
- The project is considering leveraging a deep learning models to predict the stock price in the future days.
- There could be multiple factors impacting the daily stock price, such as marco economics, subject motions etc. 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.

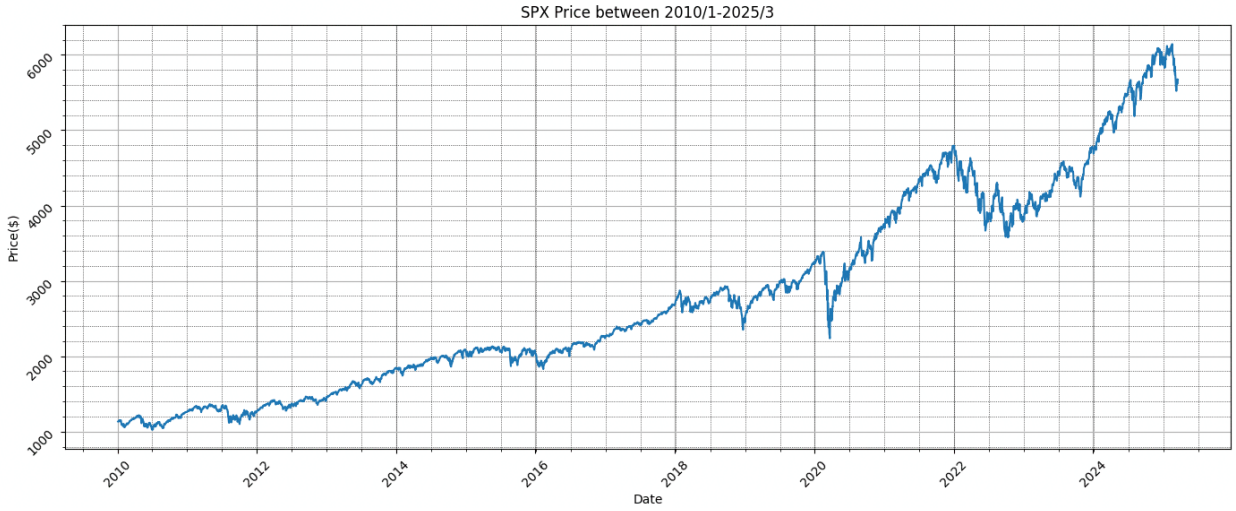
1.1. Data set

- 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.



1.2. Challenges

- It is a typical time series 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 not a good to develop a linear regression model to do the prediction.

1.3. Strategy

- Considered that stock trading happens every working day. The prices in the past days impacts the price next couple of days.
- 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, possible candidates are LSTM and RNN. I developed the code for the two models and compared the result following.
- Set the target to get precision (all sample data) less than (+/- $10)

1.4. Data preparation

```
# Define sequence length
past_days = 5  # Input sequence
future_days = 5  # Output sequence

# Function to create sequences for X y, dates grouped by past_days
def create_sequences(data, date, past_days, future_days):
  X, y, dateList = [], [], []
  quotient, mod = divmod(len(data), past_days)
  sections = quotient if mod > 0 else quotient -1
  for i in range(sections):
    #date_train.append(data.index[i:i + past_days].to_list())
    start = i * past_days
    end = start + past_days + future_days
    X.append(data[start:start+past_days])
    # ignore the last group target (not sufficent features to predict)
    if end < len(data) - mod :
      y.append(data[start+past_days:end])
    dateList.append(date[start:start+past_days])

xArray = np.array(X)
yArray = np.array(y)
dateArray = np.array(dateList)

return xArray, yArray, dateArray

## main
# Load SPX data (Replace with actual file path)
data = pd.read_csv("./data/spx2010.csv",encoding="utf-8")
data['Date'] = pd.to_datetime(data['Date'])
# sort data by date
data.sort_values('Date',inplace=True)

# Normalize price data
scaler = MinMaxScaler()
data_scaled = scaler.fit_transform(data[['Close']])

# create features and target array
X, y, DateList = create_sequences(data_scaled, data['Date'], past_days, future_days)

# Split data into train & test sets
split = int(min(len(X),len(y)) * 0.8)
len_test = len(y) if len(y) < len(X) else len(x)
X_train, X_test = X[:split], X[split:len_test]
y_train, y_test = y[:split], y[split:len_test]
```

2. Result:

2.1. Model 1: LSTM (stacked) model.

I started from LSTM architecture model and got following results. To get better result, I created two layers of LSTM in the model.

2.1.1. code:

```
# model of LSTM on LSTM
def LSTM(epochs, batch_size, X_train, y_train, X_test, y_test, past_days, future_days) :

  latent_dim = 100

  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=True),
    Dense(50, activation="relu"),
    Dense(25, activation="relu"),
    Dense(future_days) #
  ])

  model.compile(optimizer="adam", loss="mse",metrics=['mae'])
  # 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
```

2.1.2. Model summary ()

Model: "sequential"

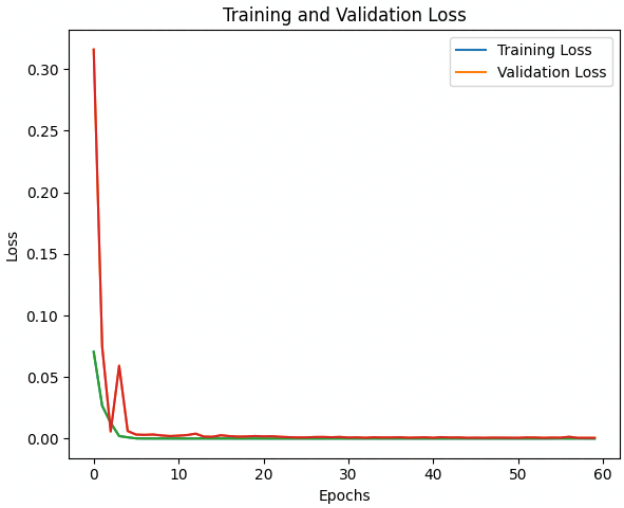
Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 5, 50)	10,400
lstm_1 (LSTM)	(None, 50)	20,200
dense (Dense)	(None, 50)	2,550
dense_1 (Dense)	(None, 25)	1,275
dense_2 (Dense)	(None, 5)	130

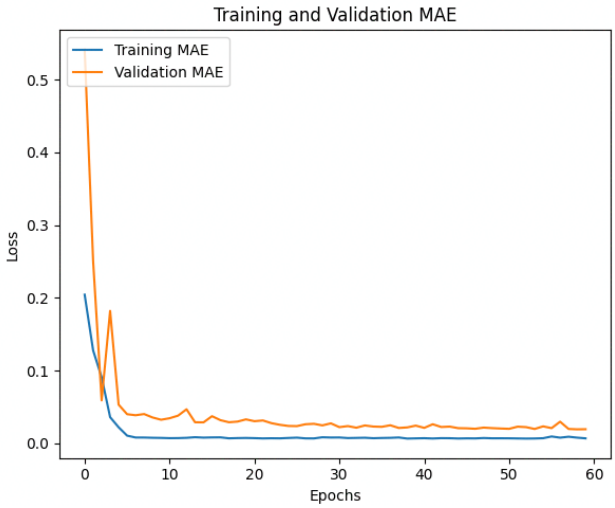
Total params: 103,667 (404.95 KB)
Trainable params: 34,555 (134.98 KB)
Non-trainable params: 0 (0.00 B)
Optimizer params: 69,112 (269.97 KB)

2.2. Hyperparameters

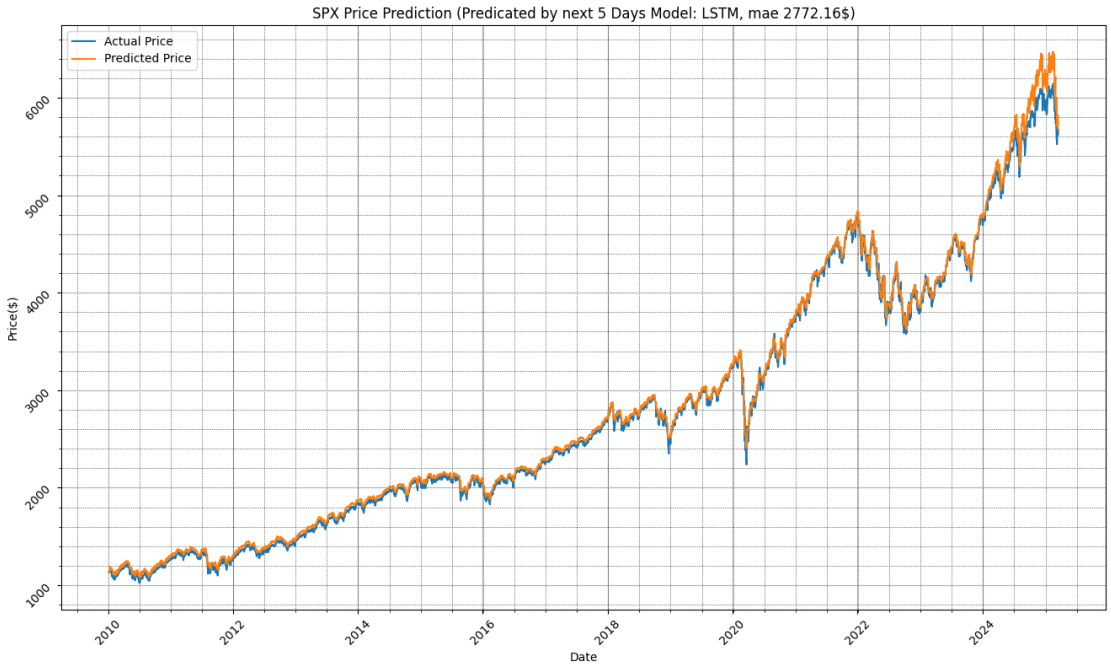
Hyper parameter	Value	Comments
Units in LSTM layer	50	
Units in LSTM layer 1	50	Better result
Optimizer	Adam	
Loss	MSE	
EPOCHS	60	
Batch_size	32	

2.3. Loss & Metrics (MAE)





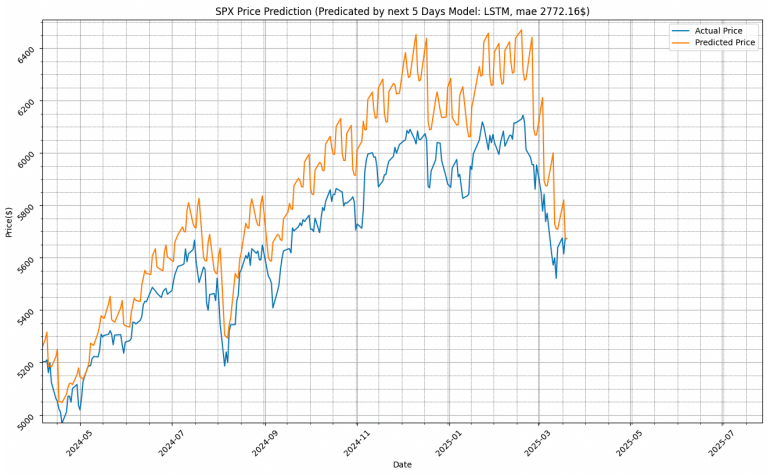
2.4. Validate the model (by all sample data)



Compare the predicted price and the actual price. The metrics are bellow.

Indicator	Value	Comment
MAE	2772.1591441795117	Not good
MSE	9401551.0633827040	Too big

2.4.1. Zoom-in



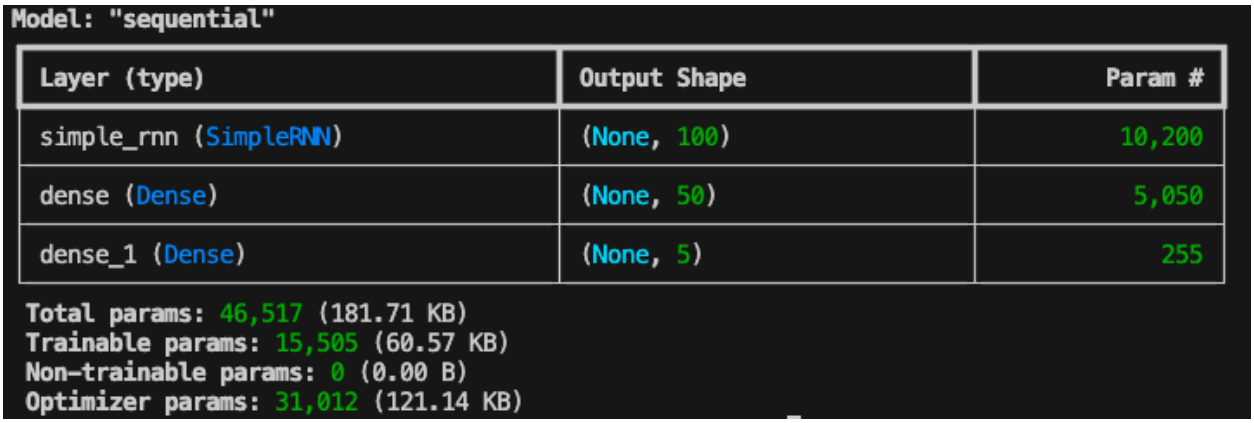
2.5. RNN model

I implemented a RNN model to do the prediction trying to reduce mae

2.5.1. Code

```
###
# model of RNN
def rnn(epochs, batch_size, X_train, y_train, X_test, y_test, past_days, future_days) :
  model = Sequential([
    SimpleRNN(100, activation='relu', input_shape=(past_days, 1), return_sequences=False),
    Dense(50, activation='relu'),
    Dense(future_days)  # This outputs a vector of length future_days
  ])
  model.compile(optimizer='adam', loss='mse',metrics=['mae'])
  # 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
```

2.5.2. Model.summary()



Model: "sequential"

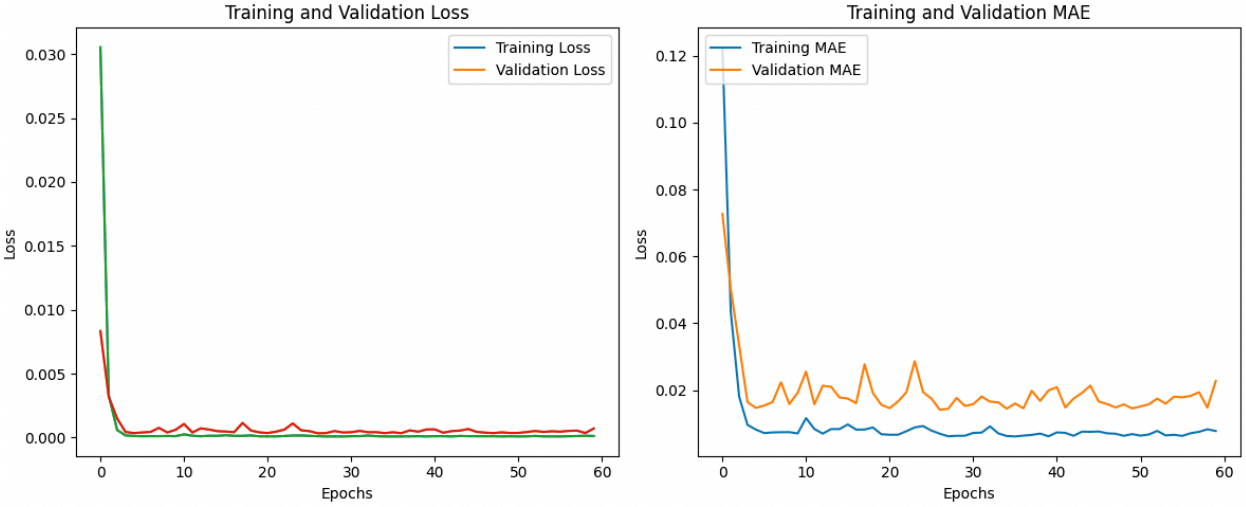
Layer (type)	Output Shape	Param #
simple_rnn (SimpleRNN)	(None, 100)	10,200
dense (Dense)	(None, 50)	5,050
dense_1 (Dense)	(None, 5)	255

Total params: 46,517 (181.71 KB)
Trainable params: 15,505 (60.57 KB)
Non-trainable params: 0 (0.00 B)
Optimizer params: 31,012 (121.14 KB)

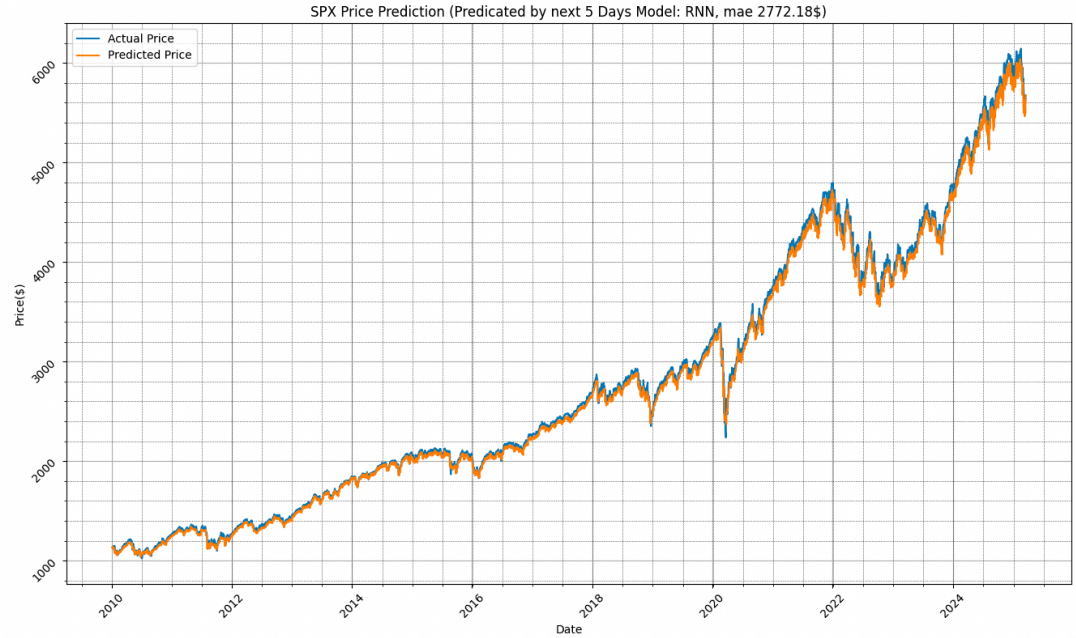
2.5.3. Hyperparameters

Hyper parameter	Value	Comments
RNN layer	1	When add the second layer of RNN, the result is worse
Units in RNN layer	50	
Optimizer	Adam	
Loss	MSE	
EPOCHS	60	
Batch_size	32	

2.5.4. Loss & Metrics (MAE)



2.5.5. Validate the model (with all sample data)



Compare the predicted price and the actual price. The metrics are bellow.

Indicator	Value	Comment
MAE	2772.1780303868577	No improved
MSE	9401672.35340898	Too big

2.6. Complex model (stacked by LSTM and RNN)

considered to make a complex model to improve MAE, I created a new model with stacked LSTM and rnn and more Dense layers.

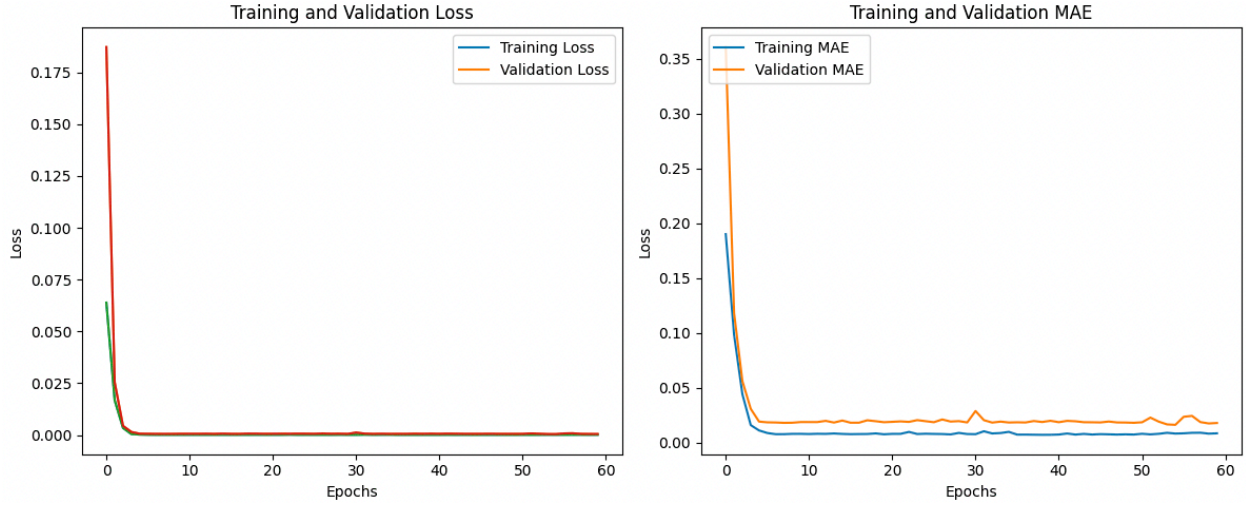
```
###
# model of lstm stack on RNN

def stack(epochs, batch_size, X_train, y_train, X_test, y_test, past_days, future_days):
  latent_dim = 60
  model = Sequential([
    tf.keras.layers.LSTM(latent_dim, activation="relu", input_shape=(past_days, 1), return_sequences=True),
    SimpleRNN(latent_dim, activation='relu', input_shape=(latent_dim, 1), return_sequences=False),
    Dense(25, activation="relu"),
    #Dense(10, activation="relu"),
    Dense(future_days) #
  ])
  model.compile(optimizer="adam", loss="mse", metrics=['mae'])
  # 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
```

```
Model: "sequential"
```

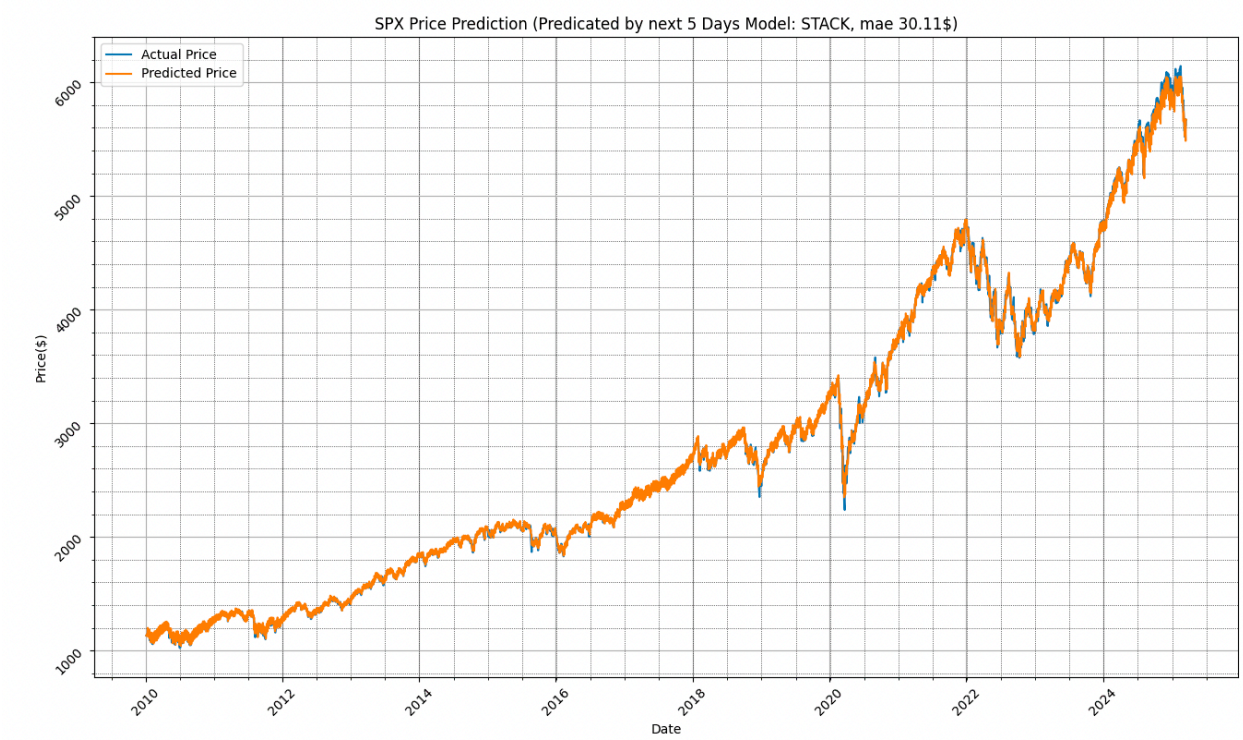
Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 5, 60)	14,880
simple_rnn (SimpleRNN)	(None, 60)	7,260
dense (Dense)	(None, 25)	1,525
dense_1 (Dense)	(None, 5)	130

```
Total params: 71,387 (278.86 KB)
Trainable params: 23,795 (92.95 KB)
Non-trainable params: 0 (0.00 B)
Optimizer params: 47,592 (185.91 KB)
```

2.6.1. Loss & Metrics (mae)



1.1.1. Validate the model (with all sample data)



Compare the predicted price and the actual price. The metrics are bellow.

Indicator	Value	Comment
MAE	30.111811640584918	Good enough
MSE	1806.8682086366314	Still too big

3. Conclusion

Compared the three models for seq2seq prediction, I have following summary.

Architecture of model	MAE (all sample data)	Conclusion
LSTM (stacked)	$2772	
RNN (single layer)	$2772	
LSTM + RNN (stacked)	$30.11	Best result

Leverage deep learning network for regression (Seq2Seq), a complex model is better than simple model.

Considering following enhancement to improve precision.

- Consider adding other features (such as “volume”, “CPI”, ”Unemployed Rate” etc.).
- Consider adding CNN layer (enhance features) to improve the model complex.
- Consider using transformer model.

The code is share via github by link https://github.com/arthurwhg/LSTM_SPX