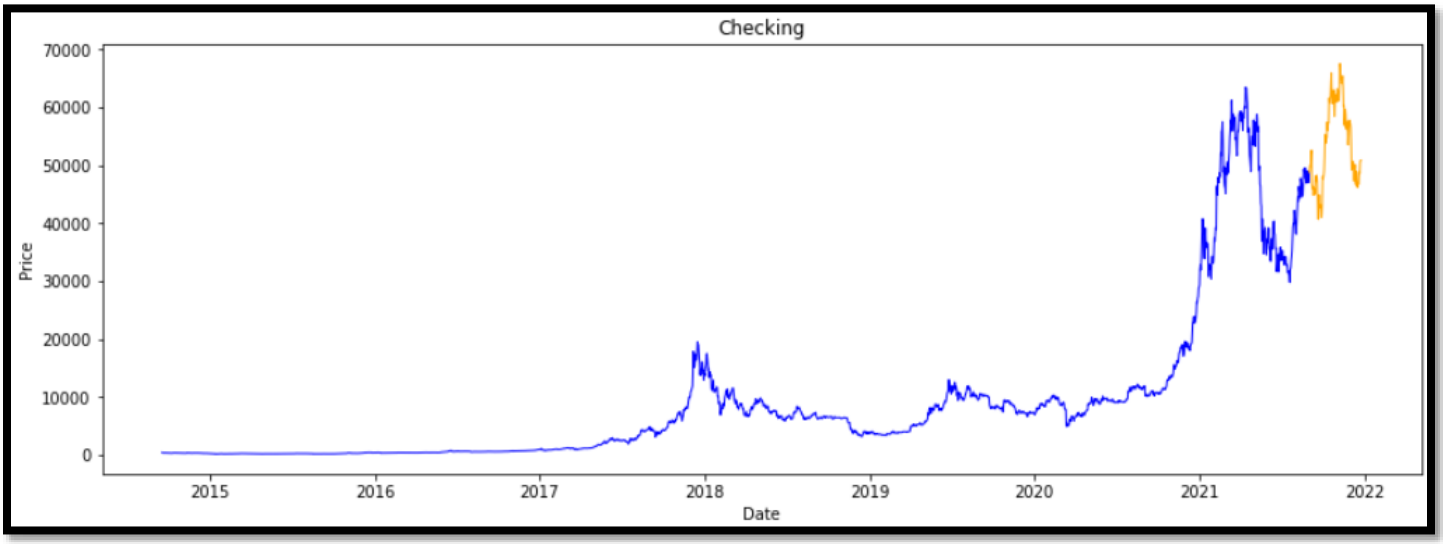
به نام خدا

ملیکا احمدی رنجبر 97521036

❖ در این سوال مرحله به مرحله طبق دستور Document جلو رفته و نتایج زیر را گرفتیم:




```
# Normalize
TrainData = np.array(TrainData).reshape(-1, 1)
TestData = np.array(TestData).reshape(-1, 1)

Scaler = MinMaxScaler()
Scaler.fit(TrainData)
TrainData = Scaler.transform(TrainData)
TestData = Scaler.transform(TestData)
```

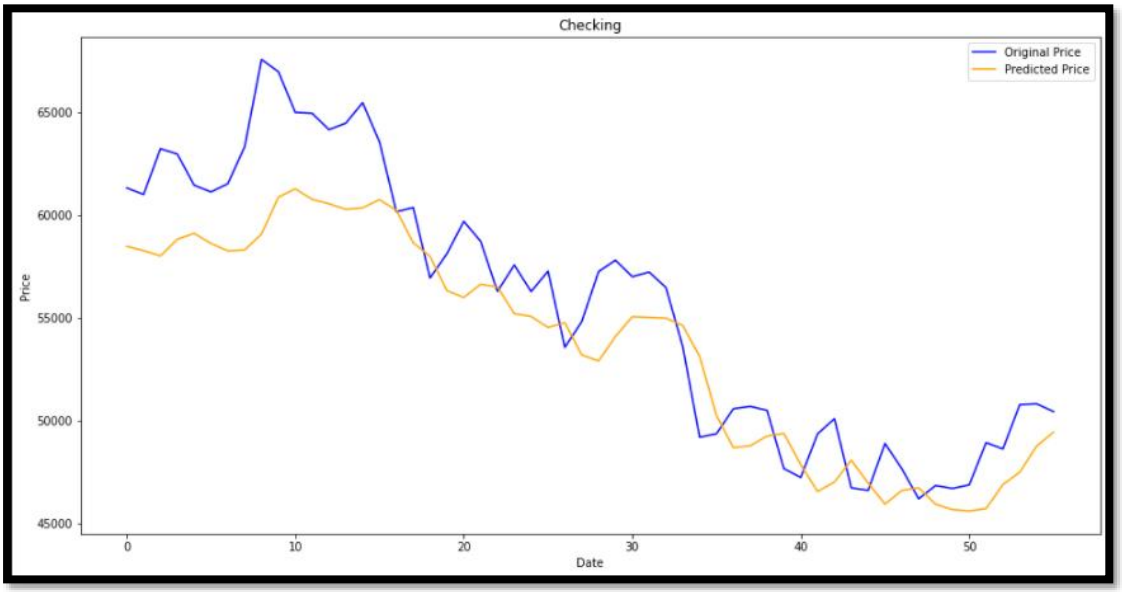
```
# Print
print(XTrain.shape)
print(YTrain.shape)
print(XTest.shape)
```

```
(2482, 60, 1)
(2482,)
(56, 60, 1)
```

پس از انجام شدن مراحل بالا، و Train کردن Model، نوبت به آزمایش Test Data رسید. در این مدل تنها کاهش Loss را در نظر گرفتیم. در زیر مقادیر ابتدا و انتهای Loss را برای Train نشان دادیم:

```
Epoch 1/100
78/78 [==============================] - 16s 63ms/step - loss: 0.0051
Epoch 2/100
78/78 [==============================] - 5s 63ms/step - loss: 0.0020
Epoch 3/100
78/78 [==============================] - 5s 63ms/step - loss: 0.0022
Epoch 4/100
78/78 [==============================] - 5s 63ms/step - loss: 0.0019
Epoch 5/100
78/78 [==============================] - 5s 63ms/step - loss: 0.0016
```

```
Epoch 95/100
78/78 [==============================] - 5s 63ms/step - loss: 5.8273e-04
Epoch 96/100
78/78 [==============================] - 5s 63ms/step - loss: 5.9362e-04
Epoch 97/100
78/78 [==============================] - 5s 64ms/step - loss: 5.2785e-04
Epoch 98/100
78/78 [==============================] - 5s 63ms/step - loss: 5.9383e-04
Epoch 99/100
78/78 [==============================] - 5s 69ms/step - loss: 5.5250e-04
Epoch 100/100
78/78 [==============================] - 5s 65ms/step - loss: 5.2630e-04
```




نمودار بالا نیز مربوط به قسمتی است که مقادیر پیش‌بینی شده توسط Model را با مقادیر واقعی در کنار یکدیگر قرار دادیم، تا مقدار دقت پیش‌بینی را بسنجیم.

افزایش و یا کاهش مقدار Variableای که تعداد داده‌های لازم را برای پیش‌بینی دخیر می‌کند نیز بسیار اهمیت دارد. در صورت کاهش این مقدار Model برای آموزش تنها به نزدیک‌ترین Data نگاه کرده، و بنابراین از Smooth بودن نمودار پیش‌بینی شده کاسته می‌شود. در حالی که هر چه این عدد بزرگ‌تر شود، داده‌های قدیمی‌تر را نیز برای بررسی و پیش‌بینی آینده در نظر می‌گیرد، پس نمودار Smoothتر است (همانند میانگین‌گیری‌های مختلفی که در گذشته داشتیم). از طرفی نیز میزان حافظه و پیچیدگی نیز بیشتر خواهد شد، به دلیل نگه داشتن تعداد Parameterهای زیاد.

❖ در این سوال نیز ابتدا File داده شده و Data داخل آن را آماده می‌کنیم تا بتوانیم از آن استفاده کنیم (Preprocessing). برای این کار هر یک از کلمات Encrypt شده را (در واقع برای هر حرف آن) یک Array با اندازه 27 در نظر می‌گیریم (تعداد حروف الفبای English به همراه Space Character) و سپس آن را Code می‌کنیم، به این شکل که هر کدام از این حروف در جایگاهی بود، آن Index را برابر با عدد 1 می‌گذاریم. به همین طریق این Stringها را تبدیل به بردارهایی با اعداد می‌کنیم و آن را آماده Train می‌کنیم.

```
def CreateSecondModel():
    Model = Sequential()
    Model.add(layers.GRU(units=256, input_shape=(XTrainSecond[0].shape)))
    Model.add(layers.RepeatVector(10))
    Model.add(layers.GRU(units=256, return_sequences=True))
    Model.add(layers.GRU(units=256, return_sequences=True))
    Model.add(layers.Dense(units=len(Alphabets)))

    Model.compile(optimizer=Optimizer, loss=Loss, metrics=['accuracy'])
    return Model
```

افزایش دقت در این سوال بسیار واضح است.

حال این مدل خاص را Test کرده تا پاسخ را با توجه به این Layers ببینیم.

```
Epoch 1/5
4759/4759 [==============================] - 97s 19ms/step - loss: 0.0234 - accuracy: 0.4791
Epoch 2/5
4759/4759 [==============================] - 89s 19ms/step - loss: 0.0082 - accuracy: 0.8626
Epoch 3/5
4759/4759 [==============================] - 89s 19ms/step - loss: 0.0043 - accuracy: 0.9287
Epoch 4/5
4759/4759 [==============================] - 89s 19ms/step - loss: 0.0033 - accuracy: 0.9431
Epoch 5/5
4759/4759 [==============================] - 90s 19ms/step - loss: 0.0028 - accuracy: 0.9499
```

```
# Test & See The Answer
Preds = ModelSecond.predict(XTestSecond)
Decrypted = ""
for i in range(Preds.shape[0]):
    for j in range(Preds.shape[1]):
        Decrypted += Alphabets[np.argmax(Preds[i, j])]

print(Decrypted)
```

```
i       love    deep    olearding
```

در حالت دوم مدل به شکل زیر بود:

```
def CreateSecondModelPrime():
    Model = Sequential()
    Model.add(layers.GRU(units=128, input_shape=(XTrainSecond[0].shape)))
    Model.add(layers.RepeatVector(10))
    Model.add(layers.GRU(units=256, return_sequences=True))
    Model.add(layers.GRU(units=512, return_sequences=True))
    Model.add(layers.Dense(units=len(Alphabets)))

    Model.compile(optimizer=Optimizer, loss=Loss, metrics=['accuracy'])
    return Model
```

```
Epoch 1/5
4759/4759 [==============================] - 104s 21ms/step - loss: 0.0248 - accuracy: 0.4338
Epoch 2/5
4759/4759 [==============================] - 98s 21ms/step - loss: 0.0120 - accuracy: 0.7756
Epoch 3/5
4759/4759 [==============================] - 99s 21ms/step - loss: 0.0061 - accuracy: 0.8990
Epoch 4/5
4759/4759 [==============================] - 99s 21ms/step - loss: 0.0041 - accuracy: 0.9325
Epoch 5/5
4759/4759 [==============================] - 99s 21ms/step - loss: 0.0033 - accuracy: 0.9443
```

```
# Test & See The Answer
Preds = ModelSecondPrime.predict(XTestSecond)
Decrypted = ""
for i in range(Preds.shape[0]):
    for j in range(Preds.shape[1]):
        Decrypted += Alphabets[np.argmax(Preds[i, j])]

print(Decrypted)
```

```
ic      love    s deep    plearning
```

مدل قبلی بهتر عمل کرد.

❖ منابع:


- https://thomas-20.medium.com/plot-finance-data-with-python-898b652c1716
- https://www.section.io/engineering-education/stock-price-prediction-using-python/