امیرحسین احمدی ۹۷۵۲۲۲۹۲ – تمرین دوازدهم

۱- برای این سوال ابتدا data frame های مربوط به داده های ترین و تست را همانطور که در سوال گفته شده از ۱-۱-۲۰۱۴ تا ۱-۹-۲۰۲۱ برای داده های ترین و از آن به بعد تا الآن را برای تست دانلود میکنیم.

```
train_data = yf.download("BTC-USD", start="2014-01-01", end="2021-09-01")
test_data = yf.download("BTC-USD", start="2021-09-02", end="2021-12-26")

train_data.shape, test_data.shape
```

```
[**********100%**********]  1 of 1 completed
[**********100%**********]  1 of 1 completed
((2542, 6), (116, 6))
```

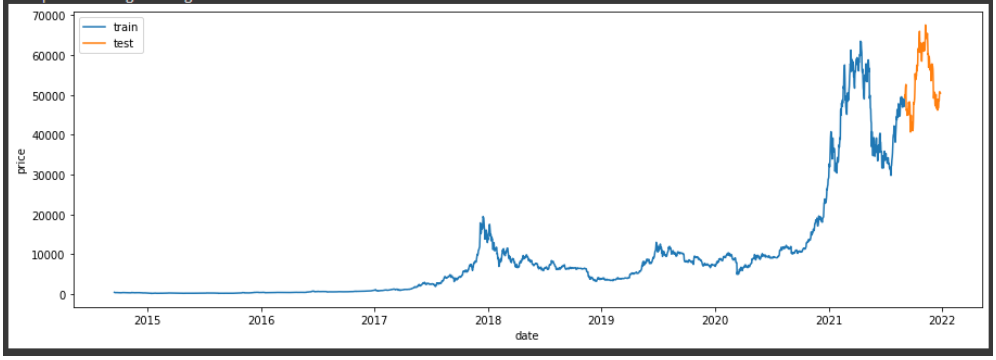
سپس نمودار قیمت زمان بسته شدن (close) را برای کل بازه ی داده های تست و ترین میکشیم.

```
plt.figure(figsize=(15,5))

plt.plot(train_data.Close)
plt.plot(test_data.Close)

plt.ylabel('price')
plt.xlabel('date')
plt.legend(['train', 'test'])
```

```
<matplotlib.legend.Legend at 0x7f30c91586d0>
```



سپس قیمت close را به عنوان داده هایی که میخواهیم روی آن مراحل آموزش را انجام دهیم انتخاب کرده و با MinMaxScaler مقادیر را برای داده های ترین و تست، اسکیل میکنیم.

```
train_data = train_data.Close.values.reshape(-1, 1)
test_data = test_data.Close.values.reshape(-1, 1)

scaler = MinMaxScaler()
scaler.fit(train_data)

train_normalized = scaler.fit_transform(train_data)
test_normalized = scaler.fit_transform(test_data)

train_normalized.shape, test_normalized.shape
```

```
((2542, 1), (116, 1))
```

سپس x_train را به این صورت ساخته که به ازای هر داده، ۶۰ داده قبلی اش را به عنوان فیچرهای آن در نظر میگیریم و داده هایی که ۶۰ داده قبل آن ها را نداریم را در نظر نمیگیریم. برای داده های تست نیز همینطور عمل میکنیم، با این تفاوت که بعضی داده ها ۶۰ روز قبلشان در داده های ترین قرار میگیرد، به همین دلیل ابتدا دو داده را چسبانده و سپس x_test را محاسبه میکنیم.

```
threshold = 60

x_train = np.array([train_normalized[i - threshold : i] for i in range(threshold, train_normalized.shape[0])])
y_train = np.array([train_normalized[i][0] for i in range(threshold, train_normalized.shape[0])])

train_test = np.concatenate((train_normalized, test_normalized))
x_test = np.array([train_test[i - threshold : i] for i in range(train_normalized.shape[0], train_test.shape[0])])
y_test = np.array([train_test[i][0] for i in range(train_normalized.shape[0], train_test.shape[0])])

x_train.shape, y_train.shape, x_test.shape, y_test.shape
```

```
((2482, 60, 1), (2482,), (116, 60, 1), (116,))
```

سپس مدل را طبق صورت سوال ساخته و با داده های ترین آموزش میدهیم.

```
model.summary()
```

```
Model: "q1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_4 (LSTM)               (None, 60, 50)            10400

 dropout_4 (Dropout)         (None, 60, 50)            0

 lstm_5 (LSTM)               (None, 60, 50)            20200

 dropout_5 (Dropout)         (None, 60, 50)            0

 lstm_6 (LSTM)               (None, 60, 50)            20200

 dropout_6 (Dropout)         (None, 60, 50)            0

 lstm_7 (LSTM)               (None, 50)                20200

 dense_6 (Dense)             (None, 1)                 51

=================================================================
Total params: 71,051
Trainable params: 71,051
Non-trainable params: 0
_________________________________________________________________
```

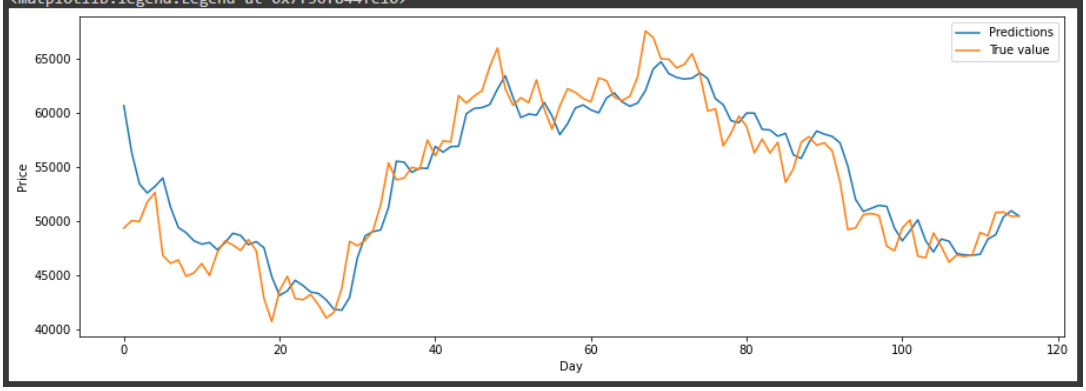
در نهایت داده های تست را به مدل داده و آن ها را به همراه لیبل اصلیشان روی نمودار میبریم. قبل این کار نیز با استفاده از scaler قیمت ها را به حالت عادیشان برمیگردانیم.

```
prediction = model.predict(x_test)

plt.figure(figsize=(15,5))

plt.plot(np.squeeze(scaler.inverse_transform(prediction)))
plt.plot(np.squeeze(scaler.inverse_transform([y_test])))
plt.xlabel('Day')
plt.ylabel('Price')
plt.legend(["Predictions", "True value"])
```

```
<matplotlib.legend.Legend at 0x7f30f844fe10>
```



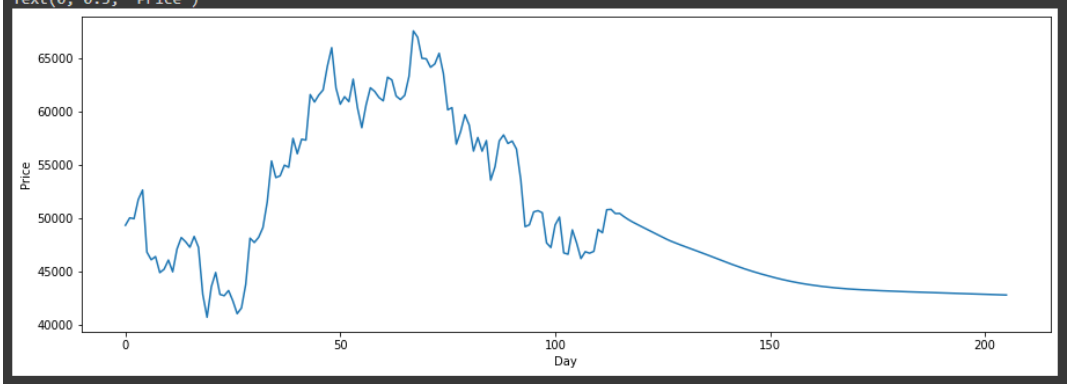
حال برای پیشبینی ۹۰ روز آینده، روزها را یکی یکی جلو رفته و از داده های قبلی و داده های پیشبینی شده قبلی برای آن استفاده میکنیم و در نهایت آن را روی نمودار میبریم.

```
future_pred = test_normalized.copy()
for i in range(90):
    x_future = future_pred[future_pred.shape[0] - 60:].T
    x_future = np.expand_dims(x_future, axis=2)
    pred = model.predict(x_future)
    future_pred = np.concatenate((future_pred, pred))
```

```
plt.figure(figsize=(15,5))

plt.plot(np.squeeze(scaler.inverse_transform(future_pred)))
plt.xlabel('Day')
plt.ylabel('Price')
```

```
Text(0, 0.5, 'Price')
```



در ارتباط با threshold در نظر گرفته شده برای فیچرها که در کد برابر با ۶۰ در نظر گرفته شده، هر چقدر این threshold بیشتر شود، فیچرهای بیشتری داریم و احتمالن به مدل کمک میکند تا پیشبینی بهتری داشته باشد، همچنین با توجه به ترتیبی بودن آموزش در شبکه های بازگشتی، با زیاد کردن تعداد فیچرها، سرعت به شدت پایین میاید. همچنین اگر threshold را خیلی خیلی زیاد کنیم

ممکنه باعث شود که دیتاهای قدیمی که دیگر زیاد ربطی به داده های جدید ندارند استفاده شوند و مدل به اشتباه بیوفتد.

۲- برای این سوال در ابتدا فایل enc-dec.txt را گرفته و x_train و y_train را از آن استخراج میکنیم. اگر هر کدام از کلمات نیز کمتر از ۱۰ حرف داشت به آن اسپیس اضافه میکنیم و آن ها را به صورت one hot در میاوریم.

```
input_file = open('enc-dec.txt').read().split('\n')
symbols = 'abcdefghijklmnopqrstuvwxyz '

x_train = np.zeros((len(input_file), 10, 27))
y_train = np.zeros((len(input_file), 10, 27))

for i in range(len(input_file)):
    [e, d] = input_file[i].split('\t')

    e += (10 - len(d)) * ' '
    d += (10 - len(d)) * ' '

    for j in range(10):
        x_train[i, j, symbols.find(e[j])] = 1
        y_train[i, j, symbols.find(d[j])] = 1

x_train.shape, y_train.shape
```

```
((152273, 10, 27), (152273, 10, 27))
```

سپس جمله ی داده شده در صورت سوال برای تست را گرفته و ۱۰ کلمه ۱۰ کلمه جدا میکنیم و آن را نیز به صورت one hot در میاوریم و در x_test نگه میداریم.

```
enc_sentence = "onmltsrqpoihgrezcba lknrvjihgfueiizltflk"

splited = []
for i in range(0, len(enc_sentence), 10):
    splited.append(enc_sentence[i:i + 10])

x_test = np.zeros((len(splited), 10, 27))
for i in range(len(splited)):
    for j in range(10):
        x_test[i, j, symbols.find(splited[i][j])] = 1

x_test.shape
```

```
(4, 10, 27)
```

در ادامه تابع train را تعریف میکنیم که ورودی آن یک اسم، لایه های مدل مورد نظر و تعداد ایپوک ها برای train است. هر بار که train صدا زده میشود چک میکند که مدلی با اسم مدنظر قبلن train شده است (در drive موجود است) یا خیر. در صورت وجود داشتن بالاترین ایپوک آن را لود کرده (برای هر ایپوک هر مدل آن را سیو میکنیم) و در صورت موجود نبود مدل را با اسم و لایه های داده شده میسازیم. سپس آن را از ایپوکی که ترین شده تا آخرین ایپوک ترین کرده و به ازای هر ایپوک مدل را سیو میکنیم. در آخر نیز تابع evaluate را صدا میزنیم تا نتیجه نهایی مدل را ببینیم.

```
def train(name, layers, epochs=10):
    current_epoch = 0
    while os.path.isfile(base_path + name + '_ep_' + str(current_epoch + 1) + '.h5'):
        current_epoch += 1

    if current_epoch == 0:
        model = Sequential(name=name, layers=layers)
    else:
        model = load_model(base_path + name + '_ep_' + str(current_epoch) + '.h5')

    model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
    for i in range(current_epoch, epochs):
        print('Epoch ' + str(i + 1) + '/' + str(epochs))
        model.fit(x_train, y_train, 32, 1)
        model.save(base_path + name + '_ep_' + str(i + 1) + '.h5')

    print('Model', name, 'successfully trained!')
    model.evaluate(x_train, y_train)

    return model
```

در کنار تابع train تابع test را نیز تعریف کرده که x_test را به مدل داده و جمله خروجی مدل را چاپ میکند.

```
def test(model):
    prediction = model.predict(x_test)

    result = ''
    for i in range(prediction.shape[0]):
        for j in range(prediction.shape[1]):
            result += symbols[np.argmax(prediction[i, j])]

    print(result)
```

حال مدل های مختلف با لایه های GRU را امتحان کرده تا ببینیم کدام نتیجه دلخواه را به ما میدهد.
ساختار کلی مدل ها به این صورت است که در ابتدا یک لایه GRU به عنوان many to one داریم، سپس تعدادی لایه GRU به عنوان many to one و در آخر نیز یک لایه dense با فعالساز softmax و ۲۷ یونیت (تعداد حروف الفبا + اسپیس) برای انتخاب حرف مورد نظر داریم.
مدل اولی که امتحان کرده ایم یک لایه GRU میانی با ۱۲۸ یونیت دارد.

```
model = train(
    name='1',
    layers=[GRU(units=128, input_shape=(10, 27)),
            RepeatVector(10),
            GRU(units=128, return_sequences=True),
            Dense(units=27, activation='softmax')])

test(model)
```

```
Model 1 successfully trained!
4759/4759 [==============================] - 33s 6ms/step - loss: 0.1285 - accuracy: 0.9614
    i        love     deep     plearning
```

همانطور که میبیند مدل دقت خوبی دارد و پیشبینی خوبی داشته و فقط اسپیس اول learning را p تشخیص داده است.

سپس تعداد یونیت های لایه های GRU را به ۲۵۶ افزایش میدهیم.

```
model = train(
    name='2',
    layers=[GRU(units=256, input_shape=(10, 27)),
            RepeatVector(10),
            GRU(units=256, return_sequences=True),
            Dense(units=27, activation='softmax')])

test(model)
```

```
Epoch 4/10
4759/4759 [==============================] - 70s 13ms/step - loss: 0.1597 - accuracy: 0.9517
Epoch 5/10
4759/4759 [==============================] - 62s 13ms/step - loss: 0.1412 - accuracy: 0.9570
Epoch 6/10
4759/4759 [==============================] - 62s 13ms/step - loss: 0.1311 - accuracy: 0.9597
Epoch 7/10
4759/4759 [==============================] - 62s 13ms/step - loss: 0.1232 - accuracy: 0.9620
Epoch 8/10
4759/4759 [==============================] - 62s 13ms/step - loss: 0.1181 - accuracy: 0.9634
Epoch 9/10
4759/4759 [==============================] - 63s 13ms/step - loss: 0.1131 - accuracy: 0.9650
Epoch 10/10
4759/4759 [==============================] - 62s 13ms/step - loss: 0.1096 - accuracy: 0.9661
Model 2 successfully trained!
4759/4759 [==============================] - 33s 7ms/step - loss: 0.1008 - accuracy: 0.9687
a p i        love     deep     flearning
```

مشاهده میکنیم که دقت بسیار بهتر شده که با توجه به یونیت های بیشتر طبیعی است که مدل زودتر همگرا شود. ولی میبینیم که خروجی تست خوبی ندارد که احتمالن نشان از overfit شدن مدل دارد.

برای مدل سوم یونیت ها را به ۱۲۸ برگردانده و در عوض یک لایه gru میانی دیگر اضافه میکنیم.

```
model = train(
    name='3',
    layers=[GRU(units=128, input_shape=(10, 27)),
            RepeatVector(10),
            GRU(units=128, return_sequences=True),
            GRU(units=128, return_sequences=True),
            Dense(units=27, activation='softmax')])

test(model)
```

```
Epoch 1/10
4759/4759 [==============================] - 76s 15ms/step - loss: 2.1586 - accuracy: 0.3594
Epoch 2/10
4759/4759 [==============================] - 75s 16ms/step - loss: 1.1232 - accuracy: 0.6448
Epoch 3/10
4759/4759 [==============================] - 72s 15ms/step - loss: 0.5089 - accuracy: 0.8365
Epoch 4/10
4759/4759 [==============================] - 70s 15ms/step - loss: 0.3472 - accuracy: 0.8917
Epoch 5/10
4759/4759 [==============================] - 70s 15ms/step - loss: 0.2815 - accuracy: 0.9138
Epoch 6/10
4759/4759 [==============================] - 70s 15ms/step - loss: 0.2393 - accuracy: 0.9276
Epoch 7/10
4759/4759 [==============================] - 70s 15ms/step - loss: 0.2148 - accuracy: 0.9349
Epoch 8/10
4759/4759 [==============================] - 70s 15ms/step - loss: 0.1961 - accuracy: 0.9406
Epoch 9/10
4759/4759 [==============================] - 70s 15ms/step - loss: 0.1808 - accuracy: 0.9451
Epoch 10/10
4759/4759 [==============================] - 70s 15ms/step - loss: 0.1705 - accuracy: 0.9481
Model 3 successfully trained!
4759/4759 [==============================] - 37s 7ms/step - loss: 0.1620 - accuracy: 0.9504
    i        love    cadeeg     plearning
```

مشاهده میشود که مدل دیرتر از دو مدل قبلی همگرا شده و دقت کمتری دارد و نتیجه تست خوبی هم ندارد و به نظر میاد که underfit است و نیاز به ایپوک های بیشتر دارد.

برای مدل چهارم یونیت های مدل سوم را افزایش دادیم تا سریع تر همگرا شود.

```
model = train(
    name='4',
    layers=[GRU(units=256, input_shape=(10, 27)),
            RepeatVector(10),
            GRU(units=256, return_sequences=True),
            GRU(units=256, return_sequences=True),
            Dense(units=27, activation='softmax')])

test(model)
```

```
Epoch 1/10
4759/4759 [==============================] - 93s 19ms/step - loss: 1.7679 - accuracy: 0.4636
Epoch 2/10
4759/4759 [==============================] - 91s 19ms/step - loss: 0.4668 - accuracy: 0.8506
Epoch 3/10
4759/4759 [==============================] - 90s 19ms/step - loss: 0.2627 - accuracy: 0.9178
Epoch 4/10
4759/4759 [==============================] - 88s 19ms/step - loss: 0.2080 - accuracy: 0.9353
Epoch 5/10
4759/4759 [==============================] - 88s 19ms/step - loss: 0.1781 - accuracy: 0.9447
Epoch 6/10
4759/4759 [==============================] - 88s 19ms/step - loss: 0.1611 - accuracy: 0.9500
Epoch 7/10
4759/4759 [==============================] - 89s 19ms/step - loss: 0.1490 - accuracy: 0.9538
Epoch 8/10
4759/4759 [==============================] - 89s 19ms/step - loss: 0.1405 - accuracy: 0.9563
Epoch 9/10
4759/4759 [==============================] - 88s 19ms/step - loss: 0.1336 - accuracy: 0.9584
Epoch 10/10
4759/4759 [==============================] - 88s 19ms/step - loss: 0.1269 - accuracy: 0.9605
Model 4 successfully trained!
4759/4759 [==============================] - 43s 9ms/step - loss: 0.1153 - accuracy: 0.9640
   ei        love    b deep     clearning
```

همانطور که میبینید مانند مدل ۲ دقت خوبی دارد ولی overfit شده و روی داده تست دقت خوبی نداشته است.