دانشکده مهندسی کامپیوتر

مباحث ویژه ۱ (یادگیری عمیق)

تمرین سری دوازدهم

علی صداقی

۹۷۵۲۱۳۷۸

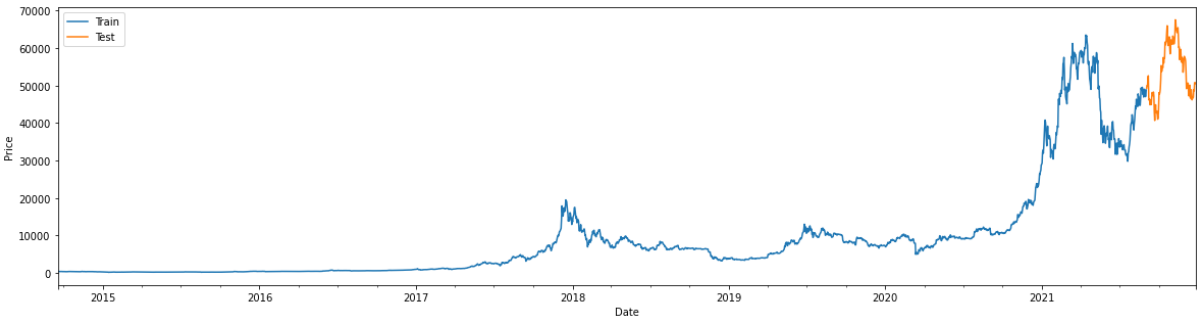
۱ سوال اول

ابتدا دیتاست مربوطه را لود می‌کنیم سپس ۱۰ داده اول آن را که به صورت یک فایل CSV است نمایش می‌دهیم.

```
1 # Showing first 10 days
2 train_data.head(10)
```

Date	Open	High	Low	Close	Adj Close	Volume
2014-09-17	465.864014	468.174011	452.421997	457.334015	457.334015	21056800
2014-09-18	456.859985	456.859985	413.104004	424.440002	424.440002	34483200
2014-09-19	424.102997	427.834991	384.532013	394.795990	394.795990	37919700
2014-09-20	394.673004	423.295990	389.882996	408.903992	408.903992	36863600
2014-09-21	408.084991	412.425995	393.181000	398.821014	398.821014	26580100
2014-09-22	399.100006	406.915985	397.130005	402.152008	402.152008	24127600
2014-09-23	402.092010	441.557007	396.196991	435.790985	435.790985	45099500
2014-09-24	435.751007	436.112000	421.131989	423.204987	423.204987	30627700
2014-09-25	423.156006	423.519989	409.467987	411.574005	411.574005	26814400
2014-09-26	411.428986	414.937988	400.009003	404.424988	404.424988	21460800

همچنین نمودار خواسته شده در صورت تمرین را رسم می‌کنیم.



مقادیر (values) مربوط به ستون Close را در یک تنسور نامپای لود می‌کنیم سپس آن را تغییر شکل می‌دهیم.

با استفاده از دستورات زیر یک نرمال‌سازی با مقیاس بیشینه-کمینه انجام می‌دهیم تا مقادیر بین عدد 0 و 1 قرار بگیرند.

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

# Normalize close values
scaler = MinMaxScaler()
scaler.fit(train_data)
```

```
train_data = scaler.fit_transform(train_data)
val_data = scaler.fit_transform(val_data)

print("Train shape:", train_data.shape)
print("Val shape:", val_data.shape)
```

شکل تنسورهای تولید شده به صورت زیر است:

```
Train shape: (2542, 1)
Val shape: (115, 1)
```

هایپرپارامترها را به صورت زیر تعریف می‌کنیم. توجه کنید که هایپرپارامتر PAST یک بار برابر 60 و بار دیگر برابر 90 در نظر گرفته شده است.

```
FUTURE = 3*30
BATCH_SIZE = 32
EPOCHS = 100
```

با استفاده از دو تابع به نام‌های create_train_set و create_val_set که خودم پیاده‌سازی کردم تنسورهای بر اساس PAST را ایجاد می‌کنیم. در حالت اول PAST برابر 60 است.

```
x_train shape: (2482, 60, 1)
y_train shape: (2482,)
x_val shape: (115, 60, 1)
y_val shape: (115,)
```

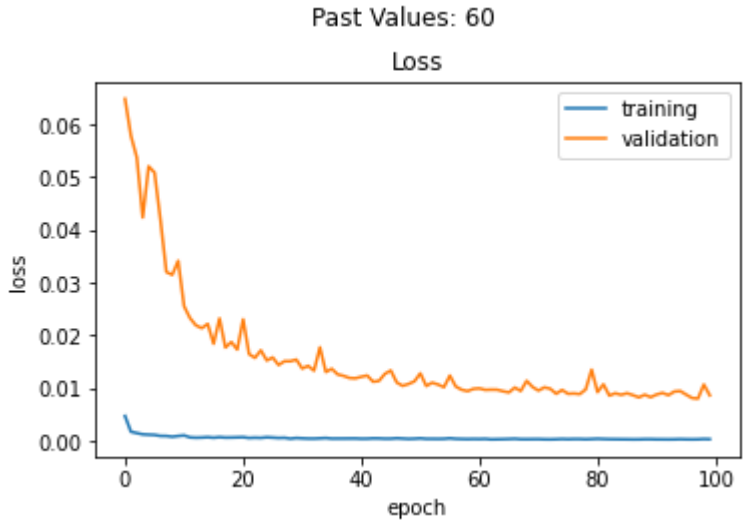
در واقع برای هر نقطه، ۶۰ نقطه قبل را نیز داریم.

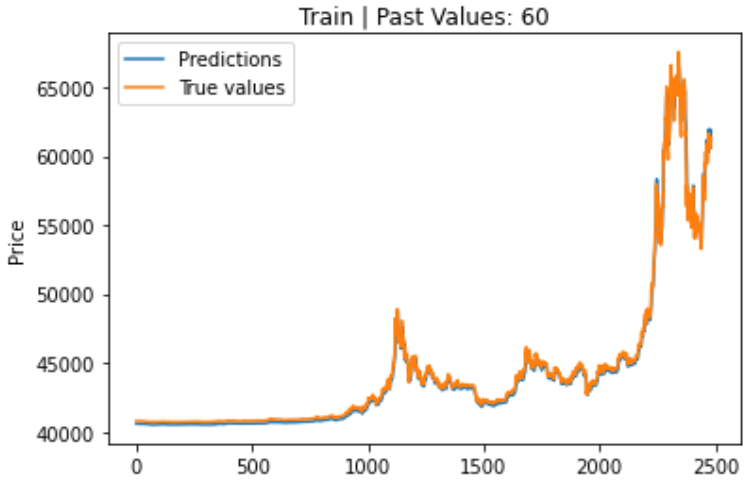
مدل را مطابق آنچه در صورت تمرین بود ایجاد می‌کنیم. خلاصه مدل به صورت زیر است:

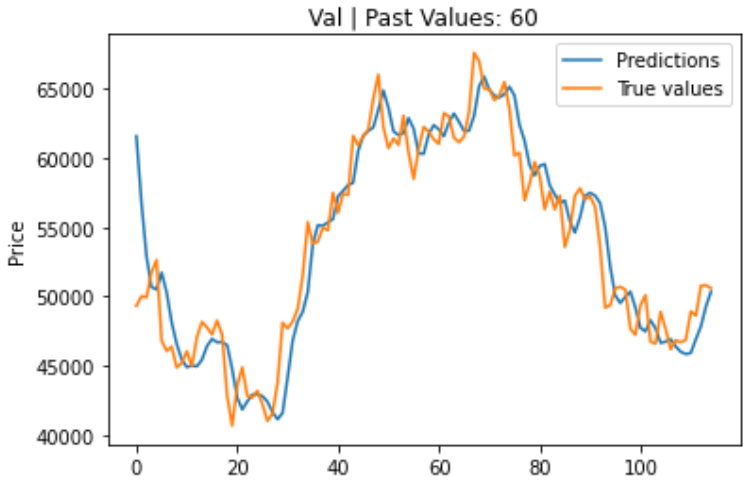
```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 60, 1)]           0

 lstm (LSTM)                 (None, 60, 50)            10400

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

 lstm_1 (LSTM)               (None, 60, 50)            20200

 dropout_1 (Dropout)         (None, 60, 50)            0

 lstm_2 (LSTM)               (None, 60, 50)            20200

 dropout_2 (Dropout)         (None, 60, 50)            0

 lstm_3 (LSTM)               (None, 50)                20200

 dense (Dense)               (None, 1)                 51

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

مدل را در ۱۰۰ ایپاک و تابع ضرر و بهینه‌ساز گفته شده آموزش می‌دهیم. نتیجه ایپاک آخر و نمودارها در زیر آورده شده است.

```
Epoch 100/100
78/78 [==============================] - 5s 64ms/step - loss: 3.0034e-04
- val_loss: 0.0086
```

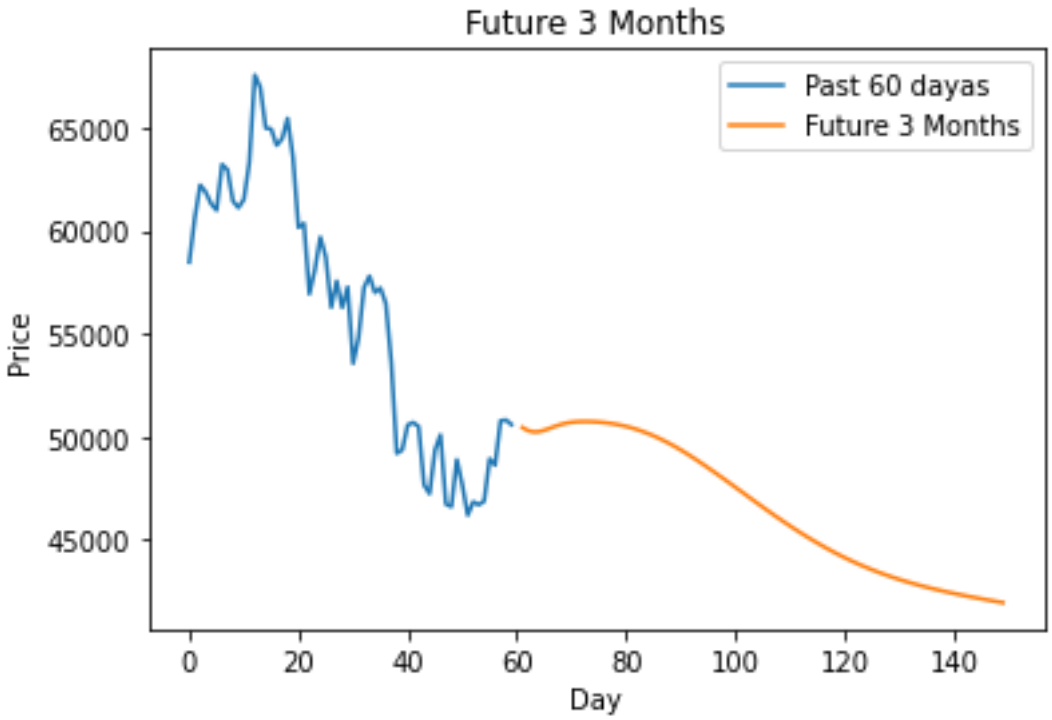






همانطور که مشاهده شد، مدل تقریبا به صورت کامل روی داده آموزش Fit شده است و Generalization خوبی نیز روی داده ارزیابی داشته است.

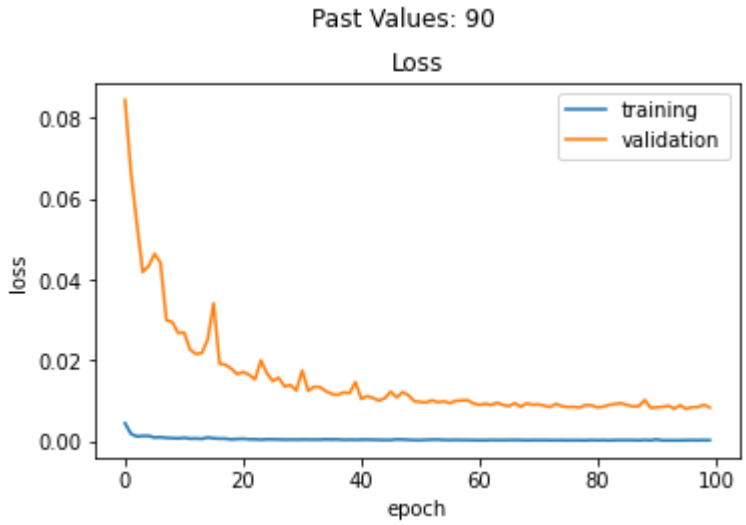
اکنون قیمت بیتکوین در ۳ ماه آینده را پیشبینی می‌کنیم. برای اینکار از ۶۰ روز گذشته که در داده Validation موجود بود استفاده کردیم. همچنین هرجا که گذشته ای در کار نبود از مقداری که مدل برای روزهای آینده پیشبینی کرده است استفاده کردیم. نتیجه به صورت زیر است.

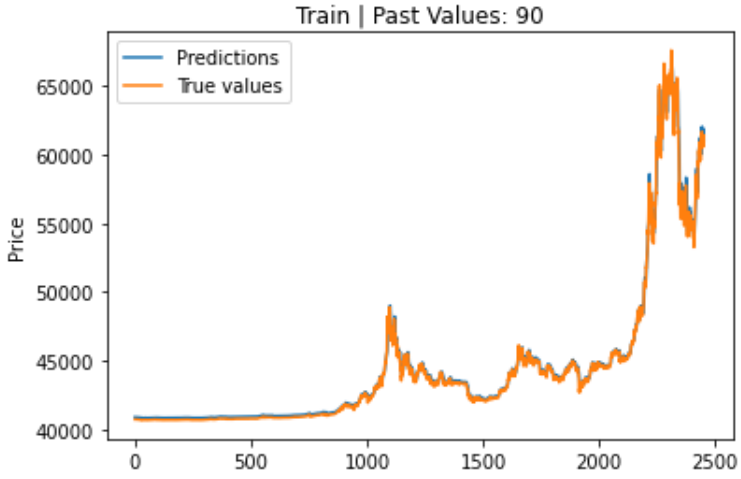


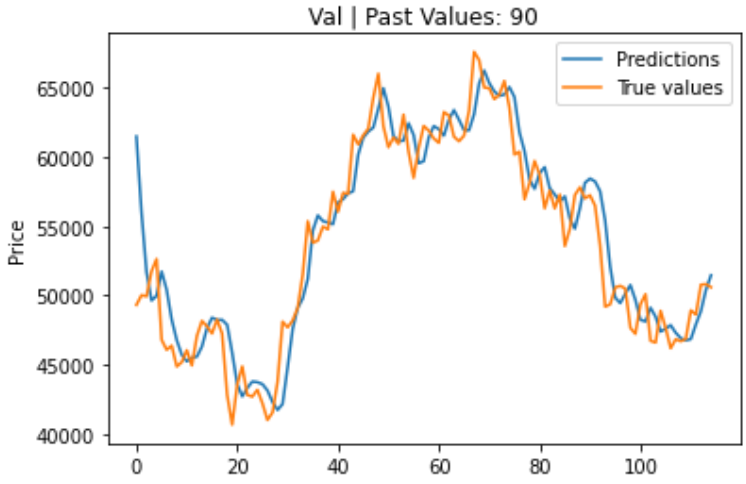
در واقع مدل ابتدا صعود و سپس سقوط را پیشبینی کرده است. به دلیل استفاده از داده‌های پیشبینی شده در مدل برای پیشبینی قسمت‌های بعدی، پیشبینی نهایی بسیار پیش ذهنیت (Bias) دارد و منحنی آن نوسان کمی دارد.

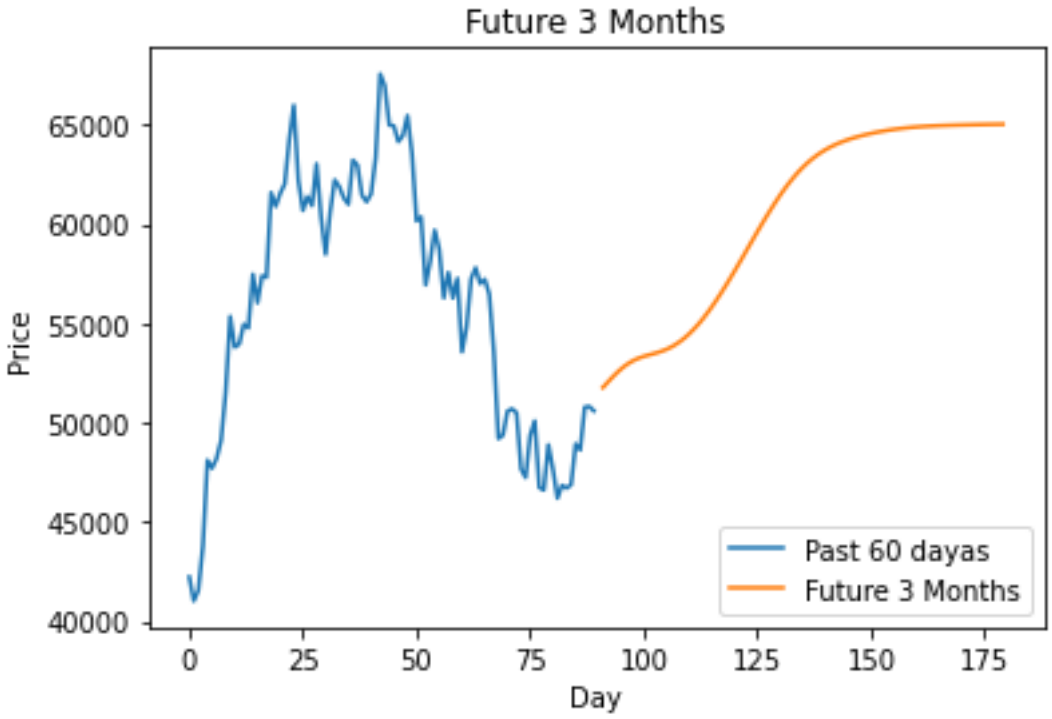
برای پاسخ به قسمت آخر سوال که تاثیر تعداد داده‌های گذشته هست، یکبار دیگر تمام کارها را با PAST برابر 90 انجام می‌دهیم. نتایج به صورت زیر است:

```
Epoch 100/100
77/77 [==============================] - 7s 87ms/step - loss: 2.9027e-04
- val_loss: 0.0083
```


Past Values: 90
Loss
training
validation
loss
epoch
0.08
0.06
0.04
0.02
0.00
0
20
40
60
80
100


Train | Past Values: 90
Predictions
True values
Price
65000
60000
55000
50000
45000
40000
0
500
1000
1500
2000
2500


Val | Past Values: 90
Predictions
True values
Price
65000
60000
55000
50000
45000
40000
0
20
40
60
80
100



نکته قابل توجه در پیشبینی ۳ ماه آینده رخ داده است. در حالت گذشته برابر ۶۰ پیشبینی شد که در ابتدا کمی افزایش قیمت و در ادامه کاهش قیمت اکید داریم. اما با حالت گذشته ۹۰ افرایش قیمت اکید پیشبینی شد. در واقع با افزایش این هایپرپارامتر توجه شبکه به الگوهای بلندمدت‌تر بیشتر می‌شود. مثلا اگر نزدیک تعطیلات کریسمس همواره کاهش قیمت داشته باشیم و شبکه بتواند یک سال گذشته را ببیند می‌تواند الگوی مربوط به آن زمان را در پیشبینی آینده لحاظ کند. مزیت دیگر افزایش این مقدار کم کردن اثر اتفاقات ناگهانی می‌باشد یعنی در یک نقطه ممکن است اتفاقی نادر بیفتد که باعث افزایش یا کاهش ناگهانی قیمت شود اما ما نمی‌خواهیم شبکه این الگوی نادر را حفظ کند. از معایب افزایش این مقدار می‌توان به افزایش حافظه مورد نیاز اشاره کرد، زیرا برای هر نقطه باید نقاط قبلی بیشتری را در نظر بگیریم (این مشکل را می‌توان حل کرد). همچنین محاسبات ریاضی در شبکه نیز افزایش خواهند یافت و زمان آموزش طولانی‌تر خواهد شد. کم بودن این مقدار در بعضی حالات نیز ممکن است باعث پیشبینی اشتباه شود. مثلا همانطور که در اسلاید ۲۲ درس در پیشبینی دما مشاهده کردیم اگر مبنا را ۱۲ ساعت قبل قرار دهیم شبکه دمای روز و شب را اشتباه می‌کند اما اگر آن را ۲۴ قرار دهیم شبکه می‌تواند برای پیشبینی صبح امروز صبح دیروز را مشاهده کند.

۲ سوال دوم

ابتدا فایل txt را بارگذاری می‌کنیم. نکته جالب این بود که کولب بدون GPU تنها 49000 خط از این فایل را می‌توانست لود کند. یک لیست به نام text می‌سازیم که هر المان آن یک خط از این فایل 152273 خطی است.

با split کردن هر خط با کاراکتر tab رشته کد شده و رشته واقعی را از هم جدا می‌کنیم.

سپس تابع add_spaces را تعریف می‌کنیم که با ورودی گرفتن یک رشته، با اضافه کردن Space آن را به رشته‌ای به طول ۱۰ تبدیل می‌کند.

دو تنسور x و y تعریف می‌کنیم که هر یک دارای 152273 المان است. x برای رشته‌های کد شده و y برای رشته اصلی است. به هر سمبل (الفبا) یک عدد بین ۰ تا ۲۶ اختصاص می‌دهیم سپس آن را به فرم One Hot تبدیل می‌کنیم. با توجه به این که هر رشته طولی برابر ۱۰ دارد شکل این دو تنسور به صورت زیر است.

```
x_train shape: (152273, 10, 27)
y_train shape: (152273, 10, 27)
```

با استفاده از لایه‌های GRU، RepeatVector و Dense یک مدل ایجاد می‌کنیم که خلاصه آن در زیر آمده است:

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 10, 27)]          0

 gru (GRU)                   (None, 256)               218880

 repeat_vector (RepeatVector  (None, 10, 256)          0
 )

 gru_1 (GRU)                 (None, 10, 256)           394752

 gru_2 (GRU)                 (None, 10, 256)           394752

 gru_3 (GRU)                 (None, 10, 256)           394752

 dense (Dense)               (None, 10, 27)            6939

=================================================================
Total params: 1,410,075
Trainable params: 1,410,075
Non-trainable params: 0
_________________________________________________________________
```

مدل را با موارد زیر Compile می‌کنیم:

```
loss=CategoricalCrossentropy(),
optimizer=Adam(0.001),
metrics=["accuracy"],
```

همچنین از یک Callback به نام ReduceLROnPlateau استفاده می‌کنیم که اگر تغییری در Loss مشاهده نکردیم، نرخ آموزش را کاهش دهیم.

رشته داده شده در متن سوال را ۱۰ حرفی جدا می‌کنیم و قسمت‌های ایجاد شده را درون یک لیست Append می‌کنیم.

```
['onmltsrqpo', 'ihgrezcba ', 'lknrvjihgf', 'ueiizltflk']
```

کارهایی که برای ایجاد x و y انجام دادیم را برای لیست بالا نیز انجام می‌دهیم تا تبدیل به یک Batch قابل Feed شدن به مدل شود.

```
x_test shape: (4, 10, 27)
```

مدل را در Epoch=30 و Batch=32 آموزش می‌دهیم و سپس روی سمپل کد شده تست می‌کنیم. نتایج به صورت زیر است:

```
Epoch 30/30
4759/4759 [==============================] - 117s 25ms/step - loss:
0.0641 - accuracy: 0.9803 - lr: 1.0000e-06
```

```
    i        love     deep    clearning
```

دقت روی داده آموزش به عدد 98.03 درصد رسید و نمونه کد شده با دقت خوبی رمزگشایی شد. تنها یک حرف c قبل از کلمه learning اضافه است.