فهرست گزارش سوالات

سوال 1 – Recurrent Networks in sequential Modeling

۱و ۲)

در این بخش ابتدا به جداسازی داده ی train , test می پردازیم که تا روز ۲۰۰۰ ام برای داده ی ترین (و ولیدیشن) و باقی که ۴۳۲ روز هست برای تست در نظر گرفته می شود. هم چنین در این قسمت داده ها توسط MinMaxscalar از کتابخانه Sklearn پیش پردازش شده اند.

در این بخش بردار ویژگی های خود را ['Open', 'Low'] در نظر می گیریم و در بخش دوم سوال عملکرد سه سلول SimpleRNN,GRU,LSTM را بررسی می نماییم.

آرایش کلی بکار رفته در این قسمت در جدول ۱ـ۱) آورده شده است.

جدول ۱ـ۱) معماری کلی بکار رفته در شبکه

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_1 (LSTM)                (None, 27)                3240
_________________________________________________________________
dense_4 (Dense)              (None, 10)                280
_________________________________________________________________
dense_5 (Dense)              (None, 1)                 11
=================================================================
Total params: 3,531
Trainable params: 3,531
Non-trainable params: 0
```

در این مرحله نتایج با در نظر گرفتن ۲۰ ایپاک و loss='MSE' و optimizer='Adam' در جدول ۱ـ۲ آورده شده است

جدول ۱ـ۲) بررسی و مقایسه ی سه روش LSTM_GRU_SimpleRNN

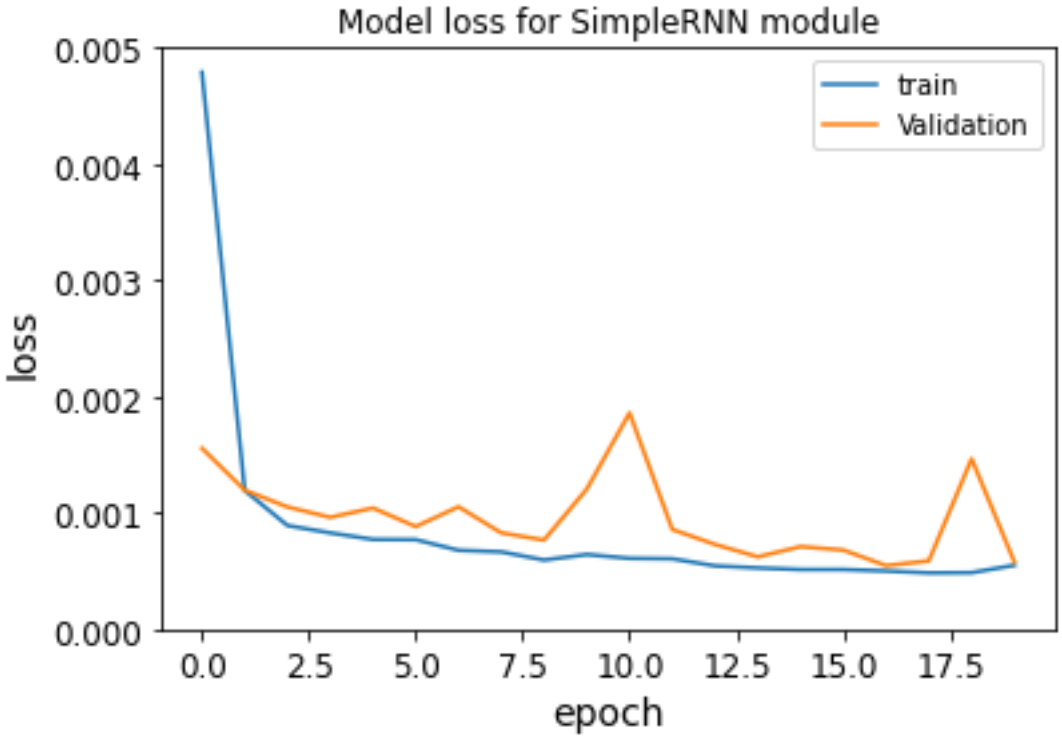
	Test_loss	Time
SimpleRNN	8 × 10^(-4)	43.13 s
GRU	7.8 × 10^(-4)	27.71 s
LSTM	0.013	12.43 s

با توجه به جدول ۱ـ۱ مشخص است که روش SimpleRNN به مراتب سرعت عمل کمتری دارد گرچه از نظر دقت تقریبا با GRU مشابه بوده و. خطای تست برای روش LSTM از اوردر بالاتری می باشد و خطای بیشتری دارد. هم چنین سادگی ساختار GRU و عدم استفاده از memory units ، برای فهم نیز می تواند بعنوان یکی از برتری ها ی این روش در نظر گرفته شود.

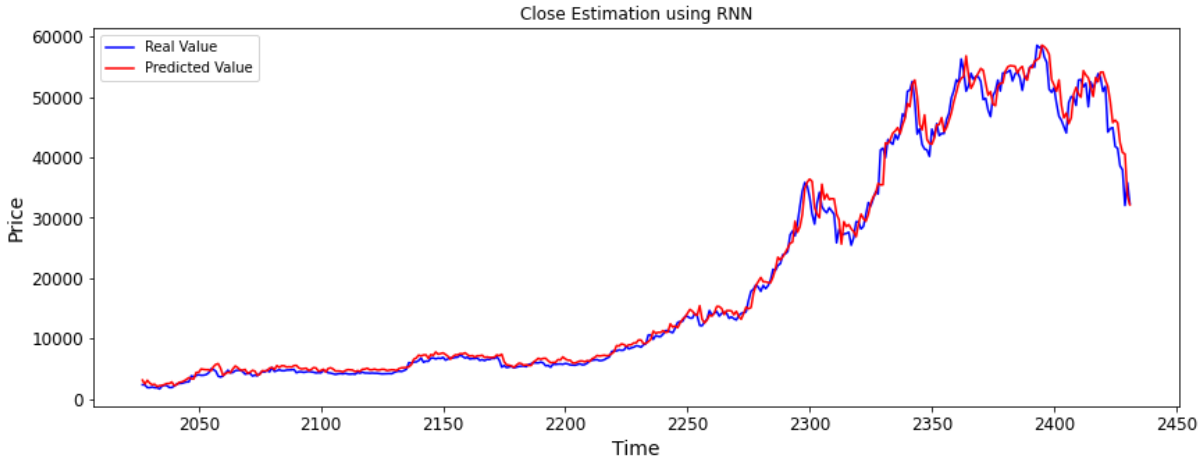
در مجموع می توان گفت سلول ساده ی GRU از نظر زمان و خطا در شرایط بهتری قرار دارد.

در ادامه نمودارهای مرتبط با loss ترین ـ ولیدیشن و مقادیر واقعی ـ تخمینی تست برای هر روش آورده شده است.

1) simple RNN :

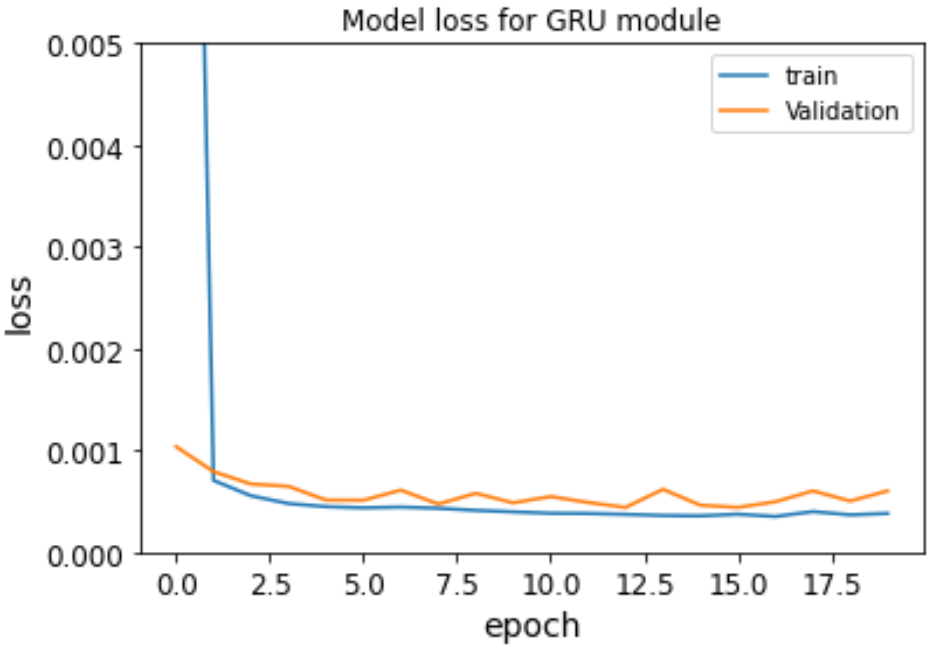


شکل ۱ـ۲) نمودار loss در ترین ـ ولیدیشن برای روش SimpleRNN

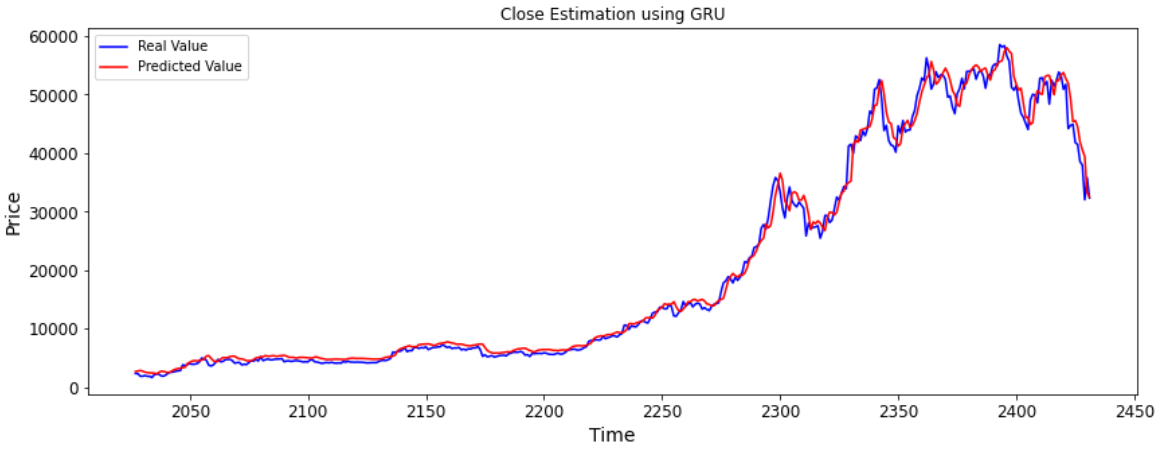


شکل ۱ـ۳) مقادیر واقعی ـ تخمینی در روش SimpleRNN

2) GRU :

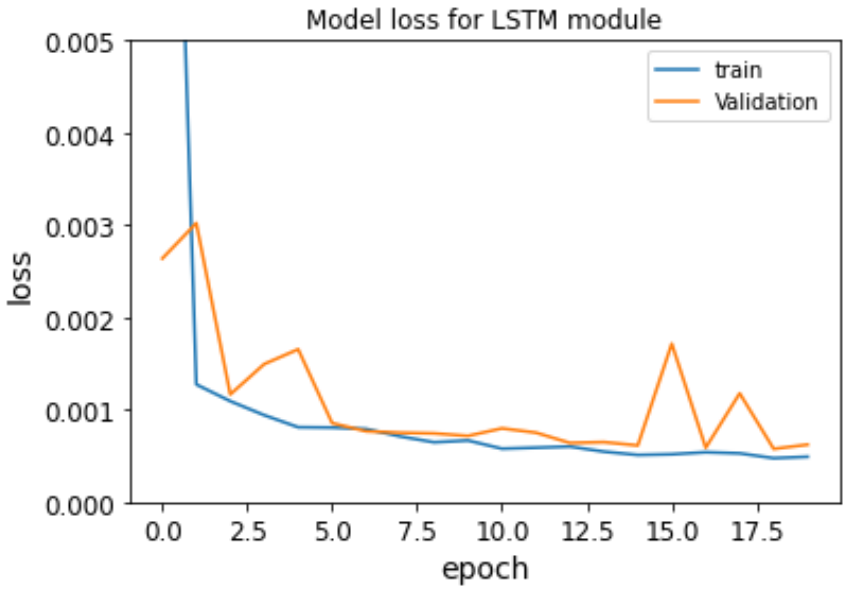


شکل ۱ـ۴) نمودار loss در ترین ـ ولیدیشن برای روش GRU

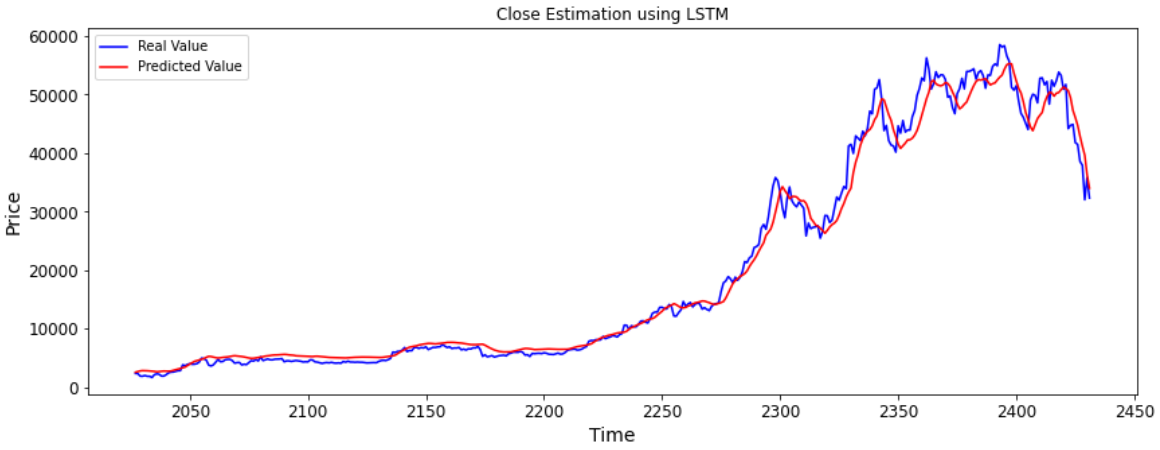


شکل ۱ـ۵) مقادیر واقعی ـ تخمینی در روش GRU

3)LSTM



شکل ۱ـ ۶) نمودار loss در ترین ـ ولیدیشن برای روش LSTM



شکل ۱ـ۷) مقادیر واقعی ـ تخمینی در روش LSTM

۳) **در ادامه ی این سوال شبکه ی GRU به کار رفته را بعنوان شبکه بهینه ی خود برمی گزینیم.

A) Loss=MSE , Optimizer=Adam :

همان نتایج قبلی مربوط به حالت GRU برقرار می باشد.

Procces_time=27.71 s

Test_Loss=0.00078

B) Loss=MSE , Optimizer=ADAgrad :

Procces_time=20.25 s

Test_Loss=0.021

C) Loss=MSE , Optimizer=RMSprop :

Procces_time=25.28s

Test_Loss=0.00082

D) Loss=MSE , Optimizer=SGD :

Procces_time=20.52s

Test_Loss=0.0013

E) Loss=MAE , Optimizer=Adam :

Procces_time=20.50 s

Test_Loss=0.0193

F) Loss=MAE , Optimizer= ADAgrad :

Procces_time=20.29 s

Test_Loss=0.062

G) Loss=MAE , Optimizer= RMSprop :

Procces_time=24.33s

Test_Loss=0.0167

G) Loss=MAE , Optimizer= SGD :

Procces_time=24.11s

Test_Loss=0.0247

با توجه به نتایج فوق در حالت 'Optimizer='Adam' 'RMSprop و به ازای 'Loss='MSE بهترین مدل را از نظر خطا داریم گرچه کمی زمان برتر از باقی روش ها می باشند.

پس در ادامه سلول ساده ی GRU در حالت 'Loss='MSE و 'Optimizer='Adam را به عنوان حالت بهینه در نظر می گیریم.

4) در این بخش می خواهیم تاثیر dropout درون سلول بازگشتی را بررسی کنیم.

`Recurrent_dropout=0.2:`

جدول۱ـ۳) بررسی و مقایسه ی سه روش LSTM_GRU_SimpleRNN با در نظر گرفتن drop_out درون سلولی

	Test_loss	Time
SimpleRNN	8.0512× 10^(-4)	44.06s
GRU	8.6181× 10^(-4)	99.18s
LSTM	0.0014	160.94s

در این شبکه با اضافه شدن dropout علی رغم صرف زمان بیشتر با کاهش خطا مواجه نبودیم که علت آن می تواند طراحی مناسب این مدل ابتدایی باشد.

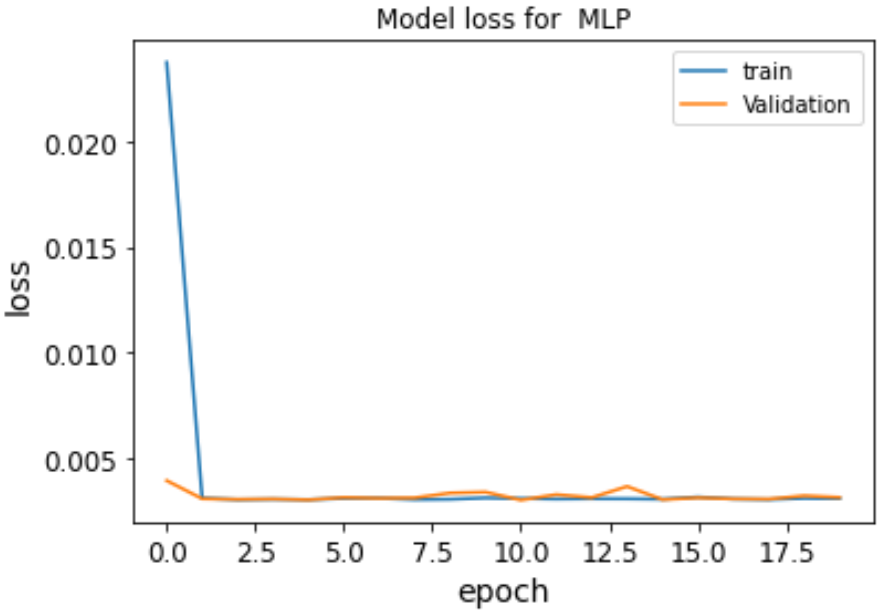
۵)

در این قسمت شبکه MLP با دو لایه ی مخفی با ساختار معرفی شده ی زیر طراحی می کنیم.

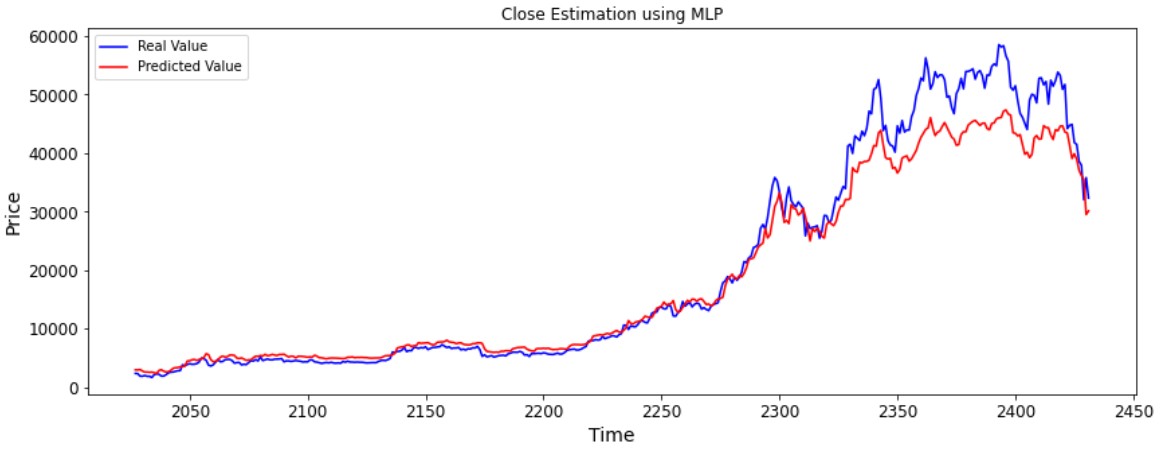
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_11 (Dense)             (None, 27, 27)            81
_________________________________________________________________
dense_12 (Dense)             (None, 27, 10)            280
_________________________________________________________________
dense_13 (Dense)             (None, 27, 1)             11
=================================================================
Total params: 372
Trainable params: 372
Non-trainable params: 0
_________________________________________________________________
```

Process_time=4.08s

Test_loss=0.010



شکل ۱ـ۸) نمودار loss برای ترین ـ ولیدیشن بر حسب ایپاک



شکل ۱ـ۹)مقادیر واقعی و تخمینی تست در روش MLP

با توجه به نتایج فوق در شبکه MLP علی رغم کم بودن پارامترها و زمان مناسب الگوریتم این روش با خطای به نسبت زیادتری نسبت به بهترین شبکه های قبلی بخصوص در مراحل پایانی می باشد گرچه می توان با tuning کردن کمی دقت آن را بهبود بخشید

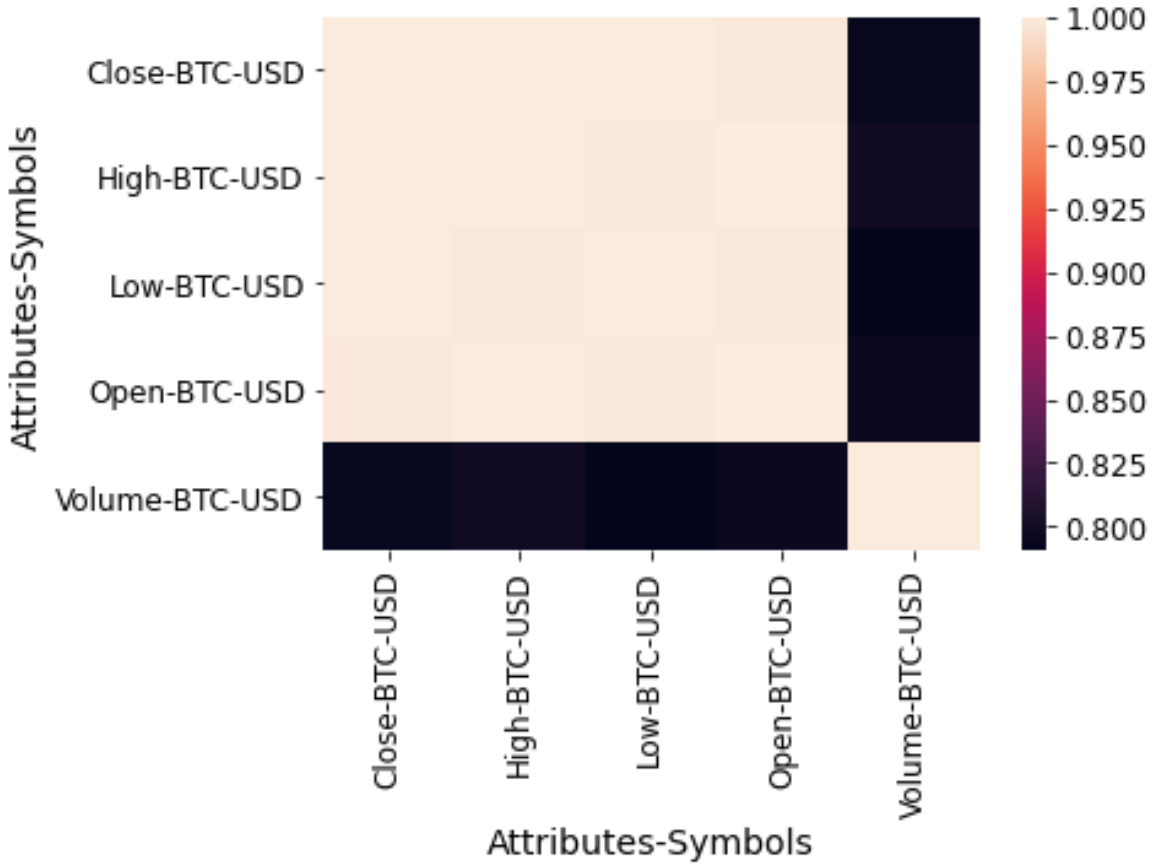
در نهایت شبکه بهینه ی انتخاب شده ی ما ترکیبی از لایه ی RNN و بعد از آن یک شبکه ی MLP ، Fully Connected می باشد

جدول ۱ـ۴) مقایسه ی بهترین شبکه بازگشتی با MLP

	Test_Loss	Time
بهترین شبکه GRU بازگشتی	7.8 × 10^(-4)	27.71s
شبکه MLP	0.010	4.08s

6) برای بررسی اهمیت Feature ها از روش های مختلفی مثل بررسی correlation و یا استفاده از DecisiontreeRegressor و یا بررسی ضرایب Coefficient مرتبط با Linear reggresion برای Feature selection استفاده نمود.

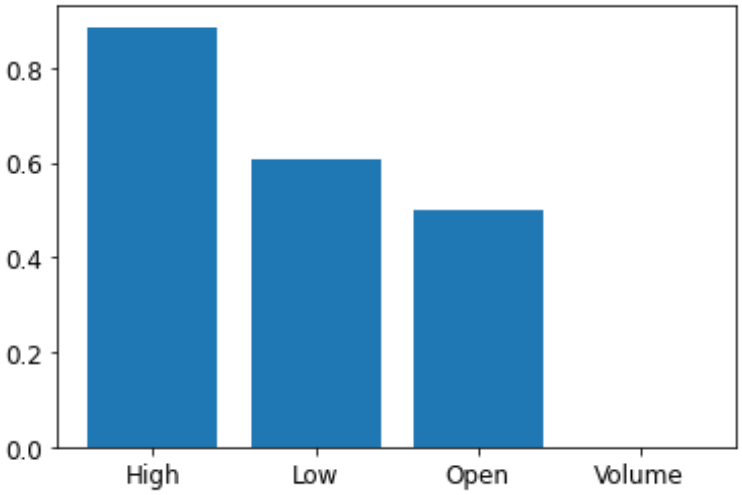
در ادامه ابتدا نتایج بدست آمده از ماتریس Correlation را بررسی می کنیم.



شکل ۱ـ۱۰) ماتریس Correlation فضای ویژگی

مطابق شکل ۱ـ۱۰) میان ابعاد ویژگی به شدت Correlation وجود دارد.نتیج بدست آمده از بخش های شبیه سازی شده ی قبلی نیز ان را به خوبی نشان میدادند جایی که تنها به کمک بردار ویژگی "Open' مشاهده می شد عملکرد Regressor ما با خطای بسیار مناسبی عمل می کرد.

هم چنین در شکل ۱ـ۱۱) اهمیت بردار ویژگی را بر اساس روش LinearRegression نشان می دهد.



شکل ۱ـ۱۱) بررسی اهمیت ویژگی ها بر اساس روش linearRegression

۷)

در این قسمت ابتدا به شبکه ی بهینه ی خود چند لایه ی دیگر اضافه کردیم.

مشاهده می شود که که علی رغم صرف زمان زیاد و پارامترهای زیاد سیستم خطای تست در همان اوردر قبلی بوده و کاهشی را مشاهده نمی کنیم

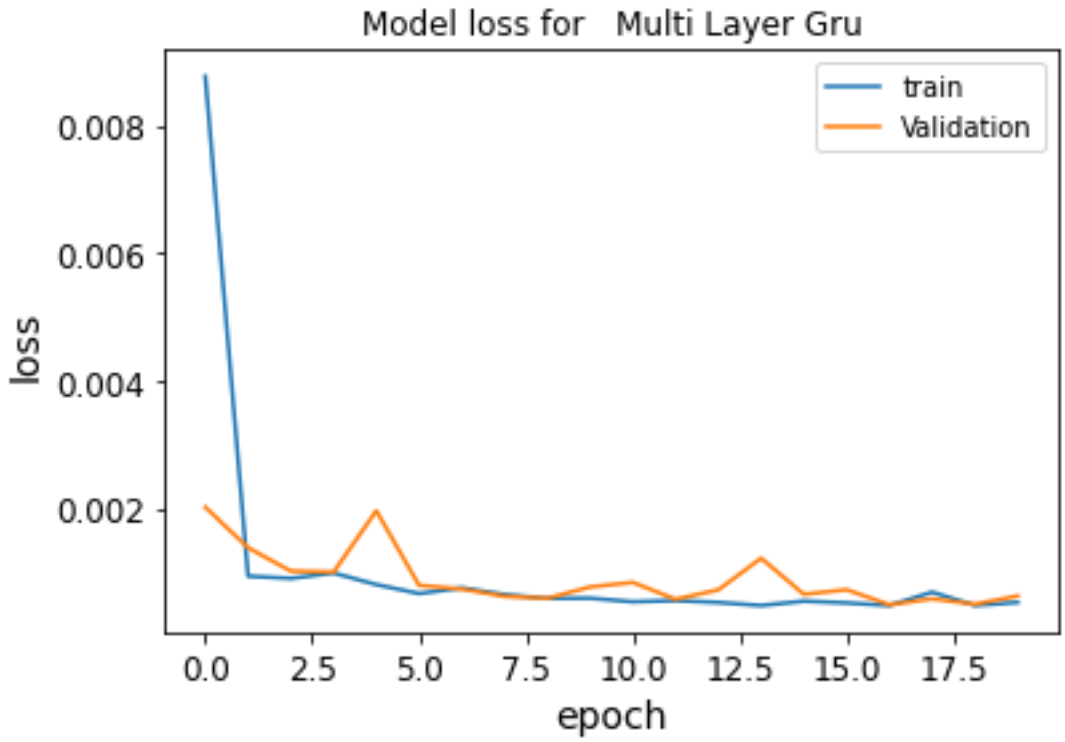
سپس به این لایه ها مقداری recurrent_Dropout در هر لایه اضافی نمودیم. با بررسی نتایج آن ها می بینیم که علی رغم صرف زمان زیاد و پارامترهای زیاد سیستم خطای تست کاهشی را مشاهده نمی کنیم.گرچه می توان گفت که در مراحل و روزهای پایانی تخمین خطا کمی کاهش پیدا کرده (اما در عوض در روز های ابتدایی تست کمی خطای بیشتری دارد.)معماری شبکه ی بکار رفته در ادامه آورده شده است.

Layer (type)	Output Shape	Param #
gru (GRU)	(None, 27, 27)	2511
gru_1 (GRU)	(None, 27, 60)	16020
gru_2 (GRU)	(None, 80)	34080
dense (Dense)	(None, 1)	81

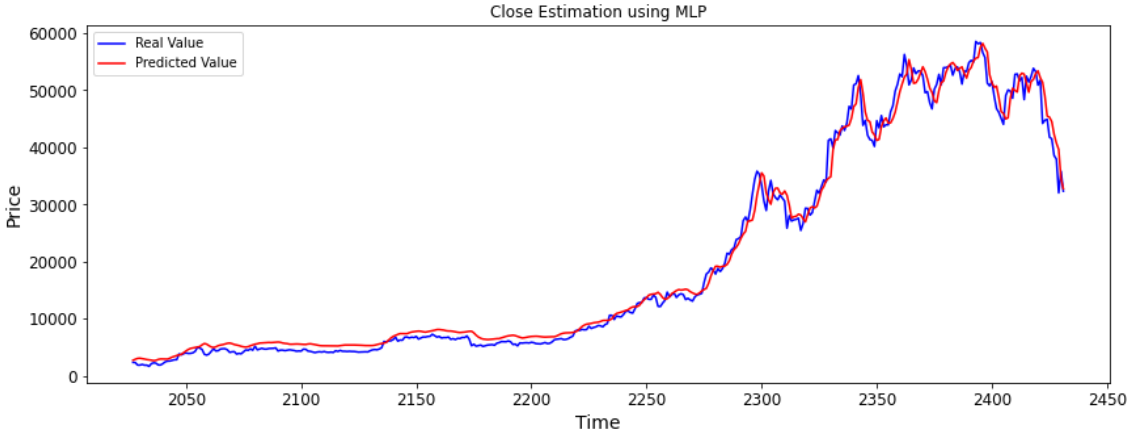
Total params: 52,692
Trainable params: 52,692
Non-trainable params: 0

Test_Loss=9.05 ×10^(-4)

Time=286.01s



شکل ۱ـ۱۲) loss برای ترین و ولیدشیشن در شبکه GrU چند لایه



شکل ۱ـ۱۳) مقادیر واقعی و تخمینی تست

۹)

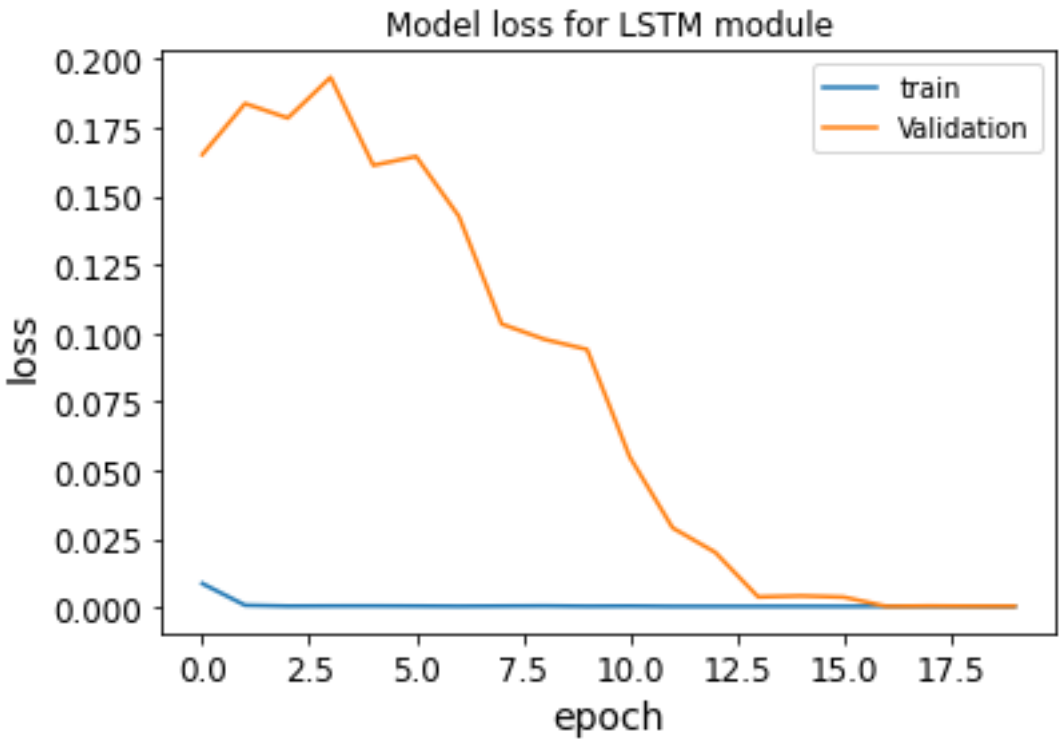
در این بخش با اضافه کردن یک لایه ی CONV1D در ابتدا با پارامتر filter=32 و Kernel=3 و پس از آن یک لایه ی Averagepooling و در ادامه ی آن قرار دادن lstm به علاوه ی ساختار Fully_ connected معماری شبکه خود را می سازیم.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d_6 (Conv1D)            (None, 27, 32)            320
_________________________________________________________________
average_pooling1d_6 (Average (None, 27, 32)            0
_________________________________________________________________
batch_normalization_6 (Batch (None, 27, 32)            128
_________________________________________________________________
lstm_5 (LSTM)                (None, 27)                6480
_________________________________________________________________
dense_10 (Dense)             (None, 10)                280
_________________________________________________________________
dense_11 (Dense)             (None, 1)                 11
=================================================================
Total params: 7,219
Trainable params: 7,155
Non-trainable params: 64
_________________________________________________________________
```

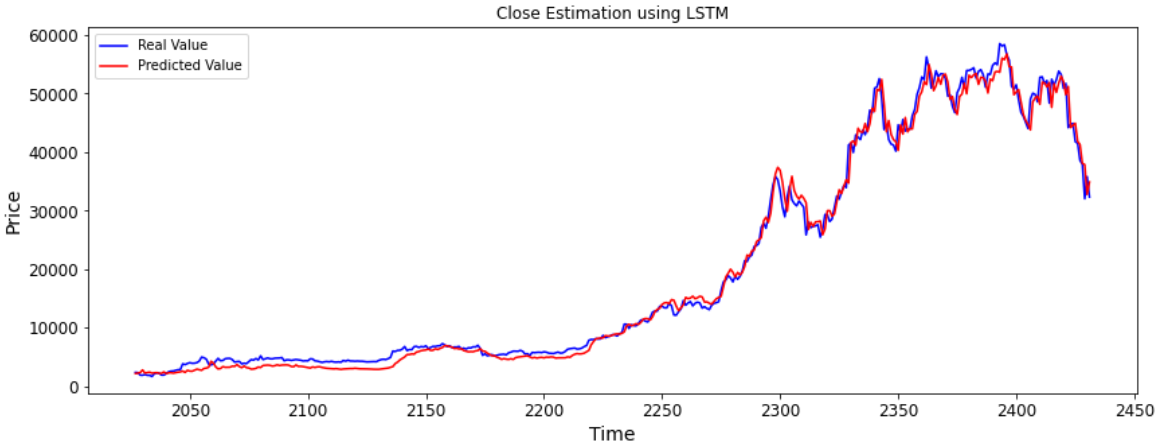
با فیت کردن مدل فوق مقادیر زیر داریم

Test_Loss=7.25×10^(-4)

Time=42.60s



شکل ۱ـ۱۴) Loss ترین و ولیدیشن برای CNN_LSTM



شکل ۱ـ۱۵)مقادیر واقعی و تخمینی برای شبکه CNN_LSTM

با توجه به نتایج فوق استفاده از CNN_LSTM توانست خطا را کاهش دهد و عملکرد بهتری از بهترین شبکه ی ما که توسط GRU داشته باشد(در زمان بیشتر).هم چنین برای بهبود بخشیدن عملکرد شبکه اضافه کردن لایه های CNN و تنظیم پارامترهای آن می تواند به ما کمک کند.نکته دیگری که باید در نظر گرفته شود این است که با هر چه پیچیده شدن مدل برای آنکه آلگوریتم ما به خوبی عمل کند نیاز به افزایش داده داریم.

(۱۰

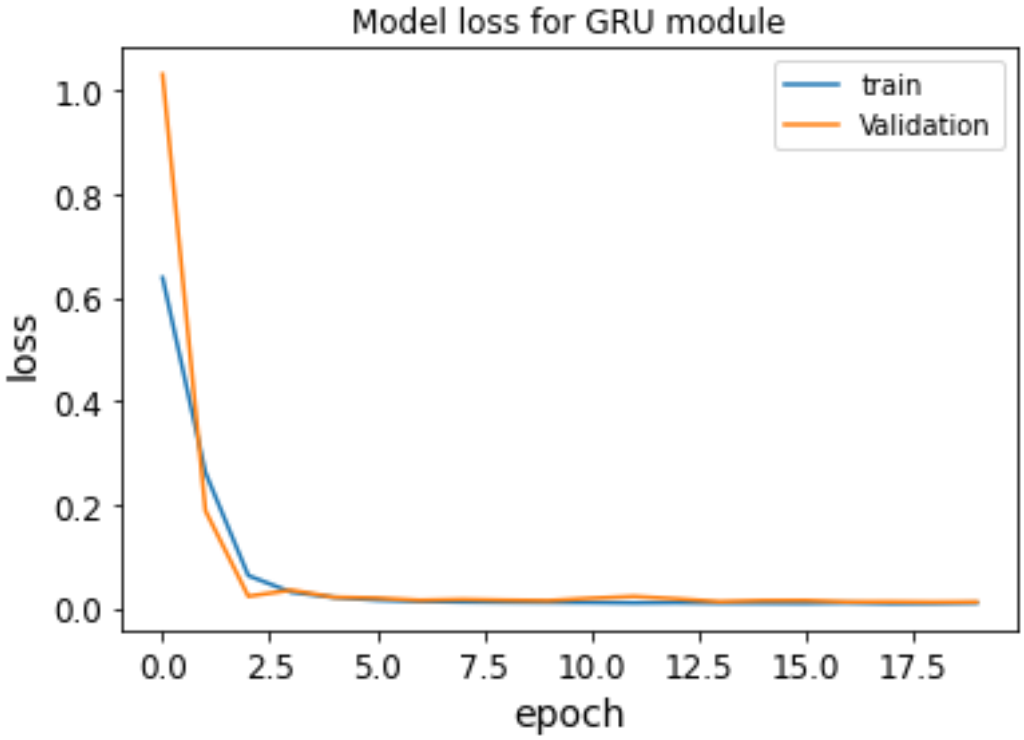
در ادامه از شبکه بهینه ی GRU از سه حالت پیش پردازش بیان شده استفاده گردیده و نتایج در ادامه بیان می گردند.

I)در بخش های قبلی در بخش پیش پردازش داده ها از MinMaxscalar استفاده شده که همان Normalization بوده و ستون های داده را بین (0,1) نرمالایز می کنندعلت استفاده از آن نیز آن است که داده های ستون ها به یک رنج در آمده و اثر یک ستون به علت بزرگی داده ها در جواب نهایی بیشتر نگردد.(نتایج مرتبط با این قسمت همان نتایج بخش دوم می باشد که به خاطر عدم تکرار در اینجا آورده نشده است.)

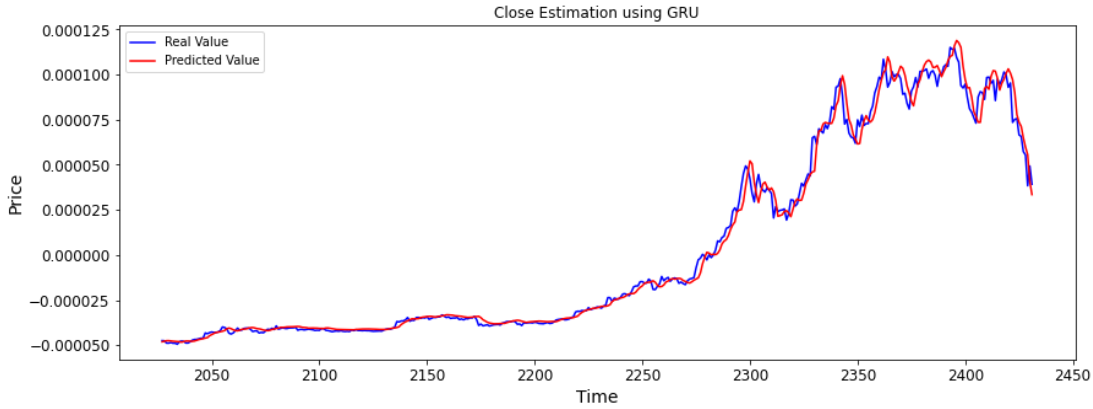
II)حال از Standardization استفاده کرده که معادل توزیع نرمال داده با میانگین صفر و واریانس یک میباشد نتایج در ادامه آورده شده است.

Test_Loss=0.0091

Time=34.91s



شکل ۱ـ۱۶) Loss ترین و ولیدیشن برای داده ی استاندار شده

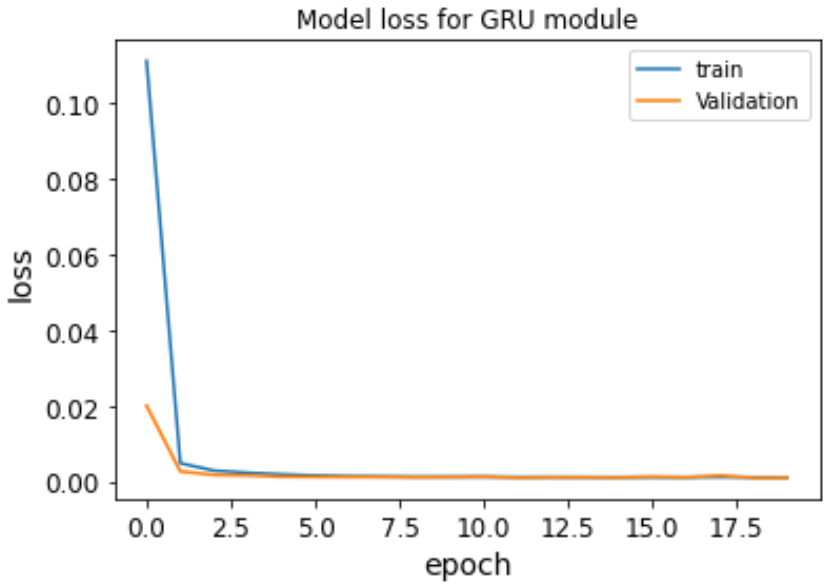


شکل ۱ـ۱۷)نمودار مقادیر تهمینی و واقعی برای داده ی استاندارد شده

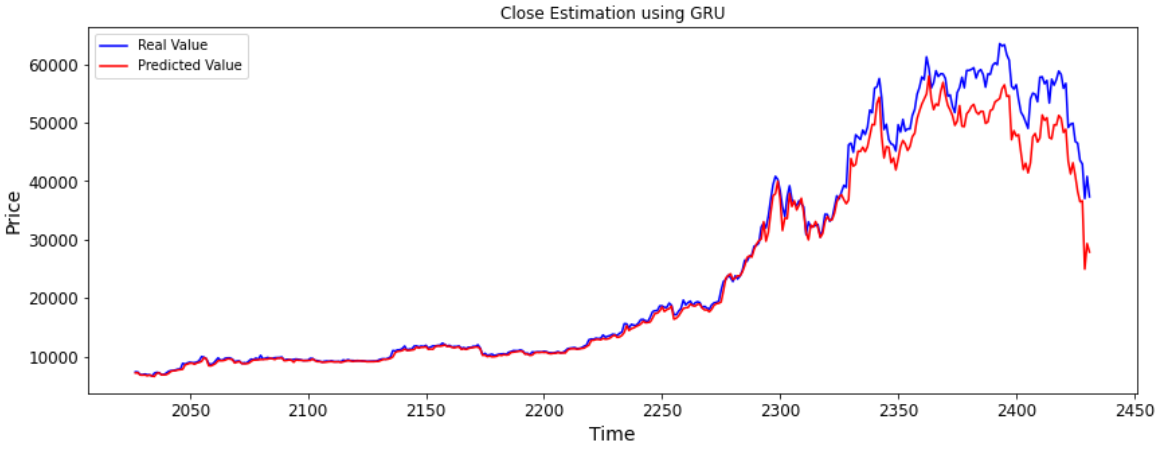
III) در این قسمت از power Transformer استفاده می کنیم که سعی بر آن دارد داده را تا حد امکان به فرم گوسی نزدیک کرده و فضای داده به شکل گاوسین در آورد.در ادامه نتایج حاصل از شبیه سازی سیستم پس از اعمال این پیش پردازش را بررسی می نماییم.

Test_Loss=0.0049

Time=26.05s



شکل ۱ـ۱۸) Loss ترین و ولیدیشن برای داده ی Power تبدیل شده



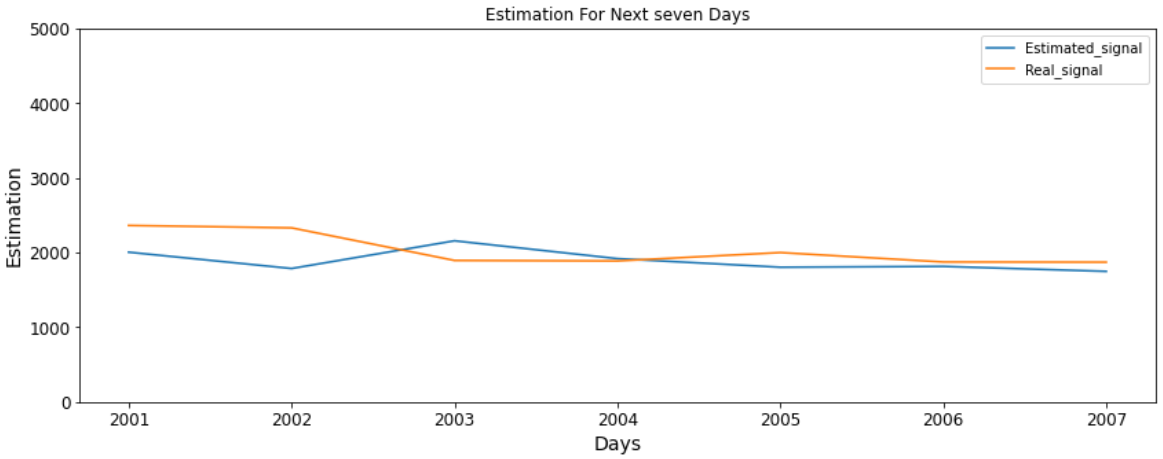
شکل ۱ـ۱۹)نمودار مقادیر تهمینی و واقعی برای داده ی power تبدیل شده

مطابق نتایج فوق مشاهده می شود که برای این داده و این مدل بکار رفته Normalization در تست بهتر عمل می کند، استفاده ازین پیش پردازش های موجود هر کدام در کاربرد های مختلف برای شبکه های مختلف را با توجه به داده ی مد نظر دارد.برای مثال برای شبکه ای به دنبال ورودی Non_Gassian باشیم استفاده از Normalization مناسب تر می باشد.بعد از آن Standardization در این شبکه بهتر عمل می کرد و در انتها power_Transform در تست با خطای نسبتا بیشتری مواجه شد.

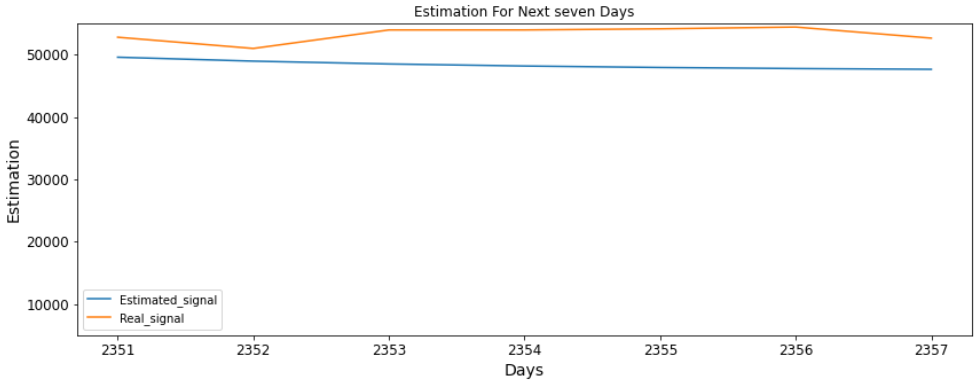
(۱۱

در این بخش با توجه به اینکه به دیتای آینده برای تست دسترسی نداریم فرض های زیر را لحاظ می کنیم

فرض می شود که فضای ویژگی ما برای تشکیل سکانس ها همان بردار 'Close' باشد که از روی آن می خواهیم برای یک هفته ی آینده قیمت را پیش بینی نماییم.حال در هر روز مقدار تخمینی برای Close را جایگزین آن در بردار مربوطه می کنیم، در دو شکل زیر نتایج این بررسی ها برای مثال در دو روز 2000 و 2350 برای هفته ی آتی آن آورده شده است.



شکل ۱ـ۲۰) تخمین یک هفته ای برای روز ۲۲۰۱ـ۲۲۰۷



شکل ۱ـ۲۱) تخمین یک هفته ای برای روز ۲۳۵۱ـ۲۳۵۷

نتایج فوق نشان می دهد که شبکه ی ما علی رغم سادگی در افق زمانی به نسبت کوتاه عملکرد خوبی دارد(گرچه با نزدیک شدن به روز ها ی انتهایی و به علت نوسانات زیادی که در حافظه سابقه ندارد مطابق شکل ۱ـ۲۱ عملکرد شبکه در تخمین به خوبی روز های قبل تر نمی باشد.)