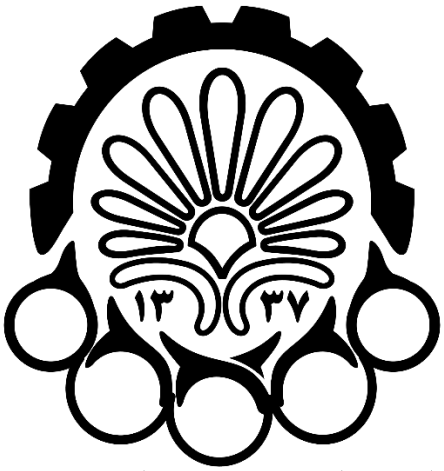


دانشگاه صنعتی امیرکبیر

(پلی تکنیک تهران)

دانشکده مهندسی کامپیوتر

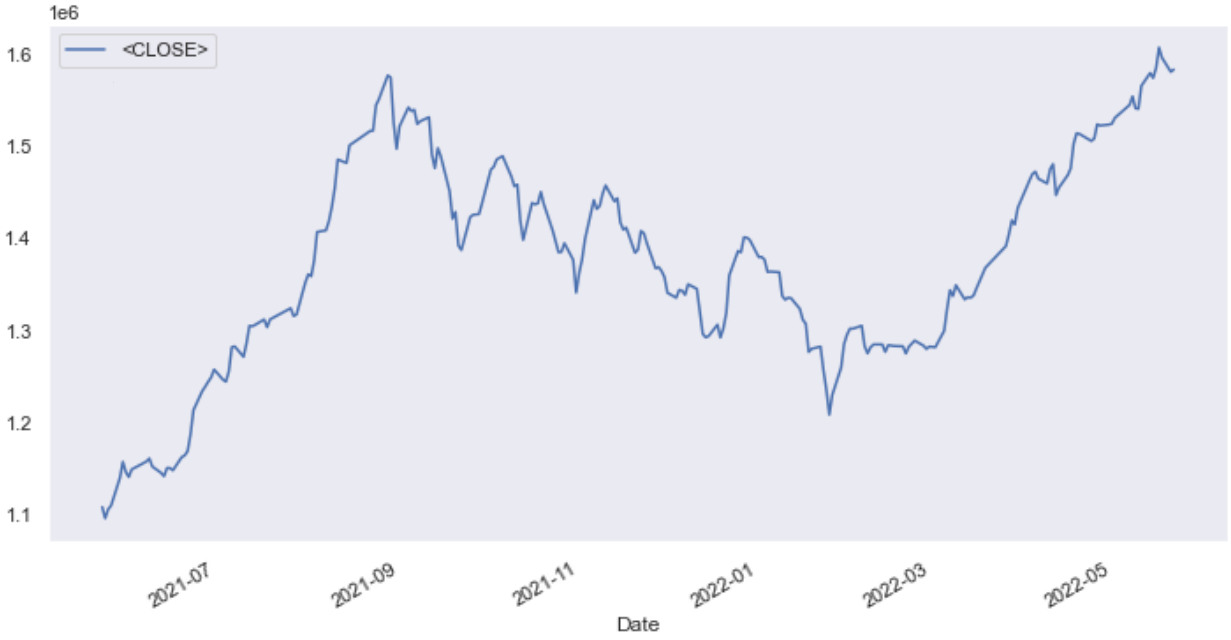
تمرین ششم درس شبکه عصبی

دکتر صفابخش

غلامرضا دار ۴۰۰۱۳۱۰۱۸

بهار ۱۴۰۱

نمودار ویژگی Close شاخص کل در یک سال اخیر



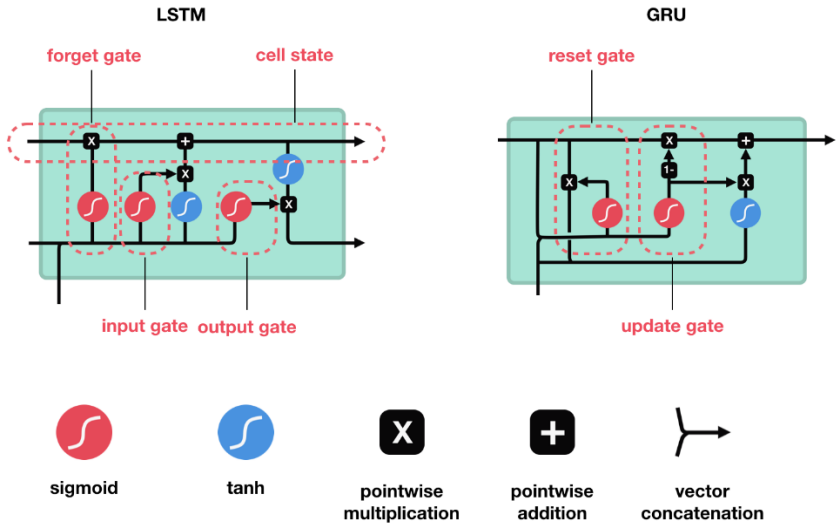
فهرست مطالب

سوال ۱) ۳
سوال ۲) ۴
سوال ۳) ۱۰

سوال ۱)

مقایسه معماری LSTM و GRU: معماری های LSTM و GRU هر دو نسخه بهبود یافته RNN ها هستند که با استفاده از مکانیزم Gate و تغییردادن حافظه، باعث جلوگیری از پدیده های **Vanishing Gradient** و **Exploding Gradient** می‌شوند. این دو پدیده در RNN های معمولی مخصوصا در مواجهه با Sequence هایی با طول بالا بسیار رایج است.

همانطور که در تصویر زیر می‌بینید، معماری LSTM از سه Gate با نام های **Input Gate, Output Gate, Forget Gate** استفاده میکند.



Forget Gate تصمیم میگیرد چه اطلاعاتی از حافظه بلند مدت باید حذف شود و دیگر مورد نیاز نیست. **Input Gate** تصمیم میگیرد چه اطلاعاتی در حافظه بلند مدت قرار بگیرد. **Output Gate** با استفاده از ورودی و حافظه های کوتاه مدت و بلند مدت، حافظه کوتاه مدت سلول بعدی را محاسبه میکند.

معماری GRU بسیار شبیه به LSTM است و در عمل در بسیاری از سناریو ها نتایج این دو معماری بسیار به یکدیگر شبیه هستند. **معماری GRU**، Input Gate و Output Gate در LSTM را با یک **Update Gate** تعویض میکند. هم چنین لازم به ذکر است که GRU بر خلاف LSTM دارای حافظه نهان داخلی نیست.

GRU به دلیل داشتن عملیات ماتریسی کمتر، معمولا هم در آموزش و هم در Inference **سریع تر** از LSTM است و به دلیل ساده تر بودن، در کاربرد هایی با سایز Sequence پایین بهتر عمل میکند. اگر **سایز Sequence بالایی** داشته باشیم و سرعت برای ما اولویت نباشد، LSTM گزینه مناسب تری خواهد بود. اما همان طور که ذکر شد، معمولا در بسیاری از کاربرد های رایج، این دو شبکه **نتایج یکسانی** به نمایش خواهند گذاشت.

سوال ۲)

در این سوال قصد داریم ابتدا مجموعه داده مربوط به سوال را فراخوانی کنیم، پیش‌پردازش های لازم را بر روی آن انجام دهیم، به زیرمجموعه های آموزش و آزمون تقسیم کنیم و درنهایت با کمک لایه های LSTM و GRU به پیش‌بینی صعودی یا نزولی بودن شاخص کل بپردازیم.

نکاتی در مورد مجموعه داده: مجموعه داده مربوط به این سوال در چندین فایل توزیع شده است. هر کدام از این فایل ها شامل داده های مربوط به وضعیت یک زیر شاخص، از سال ۲۰۰۹ تا ۲۰۲۲ هستند. یکی از این فایل‌ها مربوط به شاخص کل بازار بورس است.

```
<TICKER>,<DTYYYYMMDD>,<OPEN>,<HIGH>,<LOW>,<CLOSE>,<VOL>,<OPENINT>,<OPENINT>,<OPENINT>,<Col12>,<Col13>,<LAST>
شاخص_کل6,20081206,9248.80,9248.80,9178.30,9178.30,8539624,18557689733.00,1095,0.00,Overall Index,شاخص_کل,9178.30
شاخص_کل6,20081207,9178.30,9178.30,9130.50,9130.50,11752353,20519114974.00,1666,9178.30,Overall Index,شاخص_کل,9130.50
شاخص_کل6,20081208,9102.70,9103.40,9089.20,9089.20,15299115,60071242548.00,1873,9130.50,Overall Index,شاخص_کل,9089.20
شاخص_کل6,20081210,9071.60,9071.60,9023.70,9023.70,15689653,32587689271.00,1737,9089.20,Overall Index,شاخص_کل,9023.70
شاخص_کل6,20081213,8973.30,8973.00,8973.30,8973.30,31428174,65622897471.00,3108,9023.70,Overall Index,شاخص_کل,8973.30
شاخص_کل6,20081214,8963.90,8964.50,8907.00,8907.00,33424537,63777428978.00,2969,8973.30,Overall Index,شاخص_کل,8907.00
```

برای این سوال، میخواهیم تمام داده های مربوط به زیرشاخص‌ها را فراخوانی کنیم و ویژگی های زیرشاخص‌های مختلف را با هم ترکیب کنیم و یک مجموعه داده با تعداد ۱۲۸ ویژگی بسازیم. همچنین یک سری ویژگی ها مانند Date, Col12, Col13 را نیز حذف میکنیم.

هدف از ترکیب داده‌های زیرشاخص های مختلف این است که برای پیش‌بینی صعودی یا نزولی بودن شاخص کلی بازار بورس، به قیمت شروع و پایان و حجم تمام زیرشاخص ها نیازمندیم. در نهایت شبکه یاد خواهد گرفت که از کدام این ویژگی ها بیشتر استفاده کند و از کدام کمتر استفاده کند. به عنوان مثال فرض کنیم زیرشاخص **صنعت** تاثیر زیادی بر روی شاخص کل بازار داشته باشد و با کم و زیاد شدن ارزش سهام ها در این زیر شاخص، شاخص کلی بازار نیز تغییر زیادی کند، اما زیرشاخص **ارکان و نهادهای مالی** این گونه نباشد و تاثیر کمی بر روی شاخص کلی بازار بگذارد. به این ترتیب ما تمام اطلاعات مورد نیاز را در اختیار شبکه میگذاریم و شبکه در حین آموزش و با دیدن داده های مربوطه تصمیم میگیرد به کدام یک از این ویژگی ها وزن بیشتری بدهد.

پس از ترکیب داده‌های زیرشاخص های مختلف به مجموعه داده زیر میرسیم. همانطور که مشخص است، مجموعه داده ما دارای ۲۳۸ داده (۲۳۸ روز از ۱ سال اخیر در فایل ها ذخیره شده بود) به ابعاد ۱۲۹ است. اما در ادامه ستون Date که برای Merge کردن و فیلتر کردن مجموعه داده مورد نیاز بود را حذف میکنیم بنابراین در کل ۱۲۸ فیچر خواهیم داشت.

```
[209]  1  merged_df.info()

</>  <class 'pandas.core.frame.DataFrame'>
     Int64Index: 238 entries, 0 to 237
     Columns: 129 entries, Date to <OPENINT>.2_15
     dtypes: datetime64[ns](1), float64(106), int64(22)
     memory usage: 241.7 KB
```

نکته بسیار مهم برای کار با LSTM و GRU این است که حتما باید فیچر ها را Scale کنیم و در بازه معقولی قرار دهیم. این کار را با کمک **MinMaxScaler** کتابخانه Sklearn انجام می‌دهیم.

مرحله بعد، لغزاندن یک پنجره به طول دلخواه و تقسیم بندی مجموعه داده به تعدادی Sequence به طول مثلا ۱۵ روز است.

```
[215]
time_steps = 15
x_data, y_data = create_dataset(data, kol_df, time_steps)

print("x_data shape:", x_data.shape)
print("y_data shape:", y_data.shape)

# Note: LSTM requires input shape to be (sample, timesteps, feature_size) in this case: (222, 15, 128)
```

```
x_data shape: (222, 15, 128)
y_data shape: (222, 1)
```

دقت کنید که این Sequence ها دارای همپوشانی هستند و به این ترتیب مجموعه داده ما به ۲۲۲ Sequence ۱۵ روزه، که هر کدام ۱۲۸ ویژگی دارند تبدیل می‌شود.

در ادامه این مجموعه بزرگ را به زیرمجموعه های Train, Validation, Test تقسیم بندی میکنیم. طبق خواسته سوال مجموعه داده را به سه بخش ۷۰-۲۰-۱۰ تقسیم میکنیم.

```
x_train shape: (154, 15, 128)
y_train shape: (154, 1)
x_valid shape: (23, 15, 128)
y_valid shape: (23, 1)
x_test shape: (45, 15, 128)
y_test shape: (45, 1)
```

همچنین یک نکته مهم نحوه توزیع Label ها در این دسته ها است. در تصویر زیر میتوانیم مشاهده کنیم که در همه دسته ها حدودا ۵۷ درصد داده ها دارای Label یک هستند.

```
y_data counts: {0: 94, 1: 128} 57.658% label 1
y_train counts: {0: 65, 1: 89} 57.792% label 1
y_valid counts: {0: 10, 1: 13} 56.522% label 1
y_test counts: {0: 19, 1: 26} 57.778% label 1
```

لازم به ذکر است که لیبل ها طبق تعریف سوال از روی تفاوت ستون **CLOSE** شاخص کل بین دو روز متوالی محاسبه شده اند.

آزمایش های مربوط به یک لایه LSTM:

در این بخش می‌خواهیم شاخص کل بازار را به کمک یک لایه **LSTM** پیش بینی کنیم. نتایج آزمایش های مربوط به این بخش را در جدول زیر مشاهده میکنید.

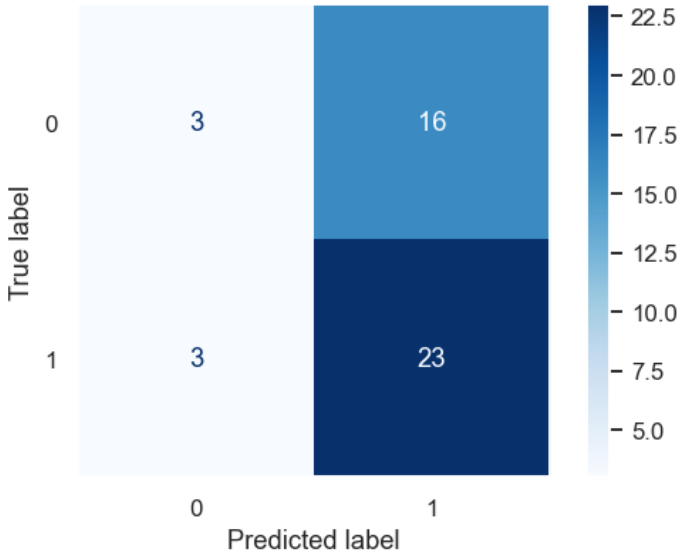
در این آزمایش‌ها از **binary_crossentropy** به عنوان loss و از **Adam** به عنوان Optimizer استفاده شد. تعداد epoch برابر ۵۰ در نظر گرفته شد.

نتایج دسته بندی به کمک یک لایه LSTM

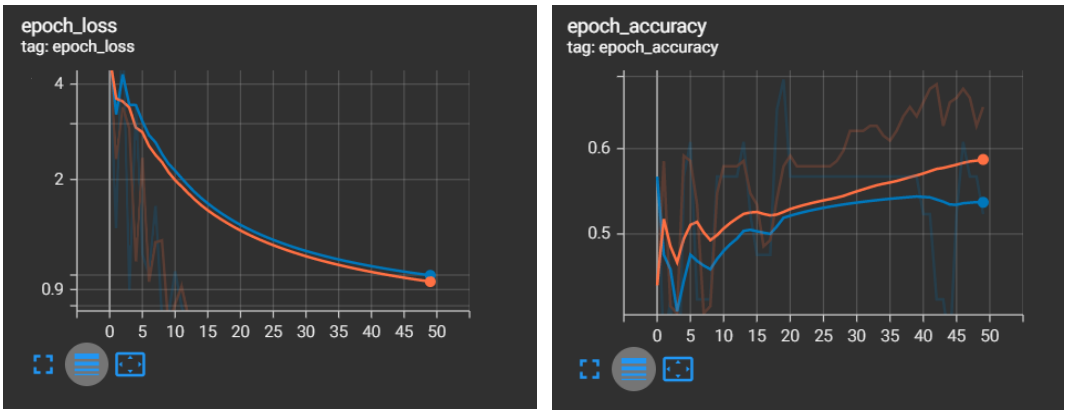
توضیحات مدل	دقت آموزش	دقت تست
LSTM(10)	۰,۴۲۲۱	**۰,۴۲۲۲**
LSTM(10)	۰,۴۲۲۱	**۰,۴۲۲۲**
LSTM(50)	۰,۶۸۱۸	**۰,۶۰۰۰**
LSTM(100)	۰,۶۵۵۸	**۰,۶۲۲۲**

نکته مهمی که در این سوال وجود دارد، تعداد فیچر خیلی زیاد (**۱۲۸**) و تعداد داده خیلی کم **۲۲۲** است. همچنین پس از تقسیم بندی داده ها به نسبت های داده شده، تعداد داده های Validation و Test بسیار کم می‌شود و این باعث می‌شود اکثر نمودارها به شدت نویزی شوند و اشتباه تشخیص دادن حتی یک نمونه نیز تاثیر زیادی بر دقت پیش بینی بگذارد.

توجه کنید زمانی که مدل دقت آموزش و تست ۴۲ درصد بدست می‌آورد به این معنی است که تقریبا به همه داده ها یک برچسب نسبت می‌دهد. ماتریس درهم‌ریختگی این مدل را در تصویر زیر مشاهده میکنید. همانطور که در ادامه آزمایش ها هم خواهیم دید، داده این مسئله با این تعداد ویژگی کافی نیست.



نمودارهای acc, loss بهترین مدل با یک لایه LSTM را در تصویر زیر مشاهده میکنید.



به طور کلی افزایش ابعاد لایه LSTM به معنی داشتن پارامترهای بیشتر و مدلی پیچیده تر است. اگر مدل Overfit شده است می‌توان با افزایش ابعاد لایه LSTM پیچیدگی مدل را افزایش داد و از Overfitting کاست. همانطور که قبلا هم ذکر شد این مسئله بهترین مثال برای بررسی نحوه بررسی LSTM وGRU نیست و توانایی این مدل‌ها را به خوبی نمایش نمی‌هد. با این وجود طبق مشاهداتی که داریم میتوان گفت که افزایش ابعاد لایه LSTM تا حدی به دقت کمک میکند و از حدی به بعد به صحت مدل ضربه میزند.

آزمایش های مربوط به یک لایه GRU:

در این بخش می‌خواهیم شاخص کل بازار را به کمک یک لایه **GRU** پیش بینی کنیم. نتایج آزمایش های مربوط به این بخش را در جدول زیر مشاهده میکنید.

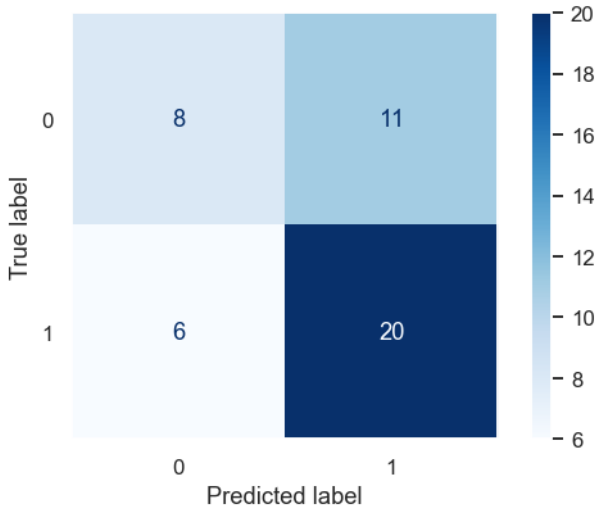
در این آزمایش‌ها از **binary_crossentropy** به عنوان loss و از **Adam** به عنوان Optimizer استفاده شد. تعداد epoch برابر ۵۰ در نظر گرفته شد.

نتایج دسته بندی به کمک یک لایه GRU

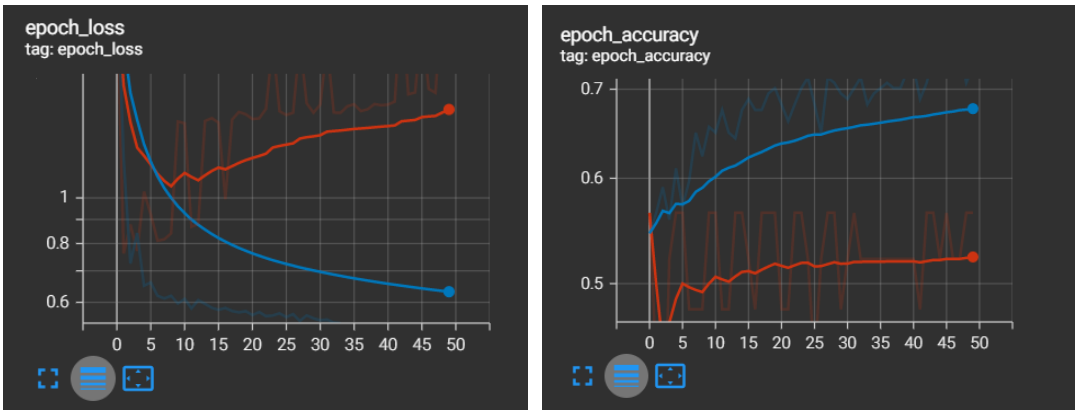
توضیحات مدل	دقت آموزش	دقت تست
GRU(5)	۰,۶۵۵۳	**۰,۵۳۳۳**
GRU(10)	۰,۷۰۱۳	**۰,۵۷۷۸**
GRU(50)	۰,۷۲۷۳	**۰,۶۲۲۲**
GRU(100)	۰,۵۷۷۹	**۰,۵۷۷۸**

مشکلاتی که در بخش قبل ذکر شد در این بخش نیز صادق است.

ماتریس درهم ریختگی بهترین مدل با یک لایه GRU را در تصویر زیر مشاهده میکنید.



نمودارهای acc, loss بهترین مدل با یک لایه GRU را در تصویر زیر مشاهده میکنید.



توضیحات این بخش نیز مانند بخش قبل است و مشکلات این سوال باعث می‌شود نتوان به درستی در مورد تغییرات مختلف و نتایج آنها صحبت کرد اما با این وجود طبق مشاهدات از این مسئله خاص مانند LSTM، افزایش ابعاد لایه GRU به افزایش صحت مدل کمک میکند اما از حدی به بعد از صحت مدل میکاهد.

نتیجه گیری و مقایسه بین بهترین مدل LSTM و GRU:

با توجه به آزمایش‌های انجام شده، برای این سوال معماری GRU نتایج بهتری مخصوصا با تعداد نورون کمتر داد. همچنین همانطور که در بخش معرفی این دو معماری ذکر شد، معماری GRU به دلیل داشتن عملیات ماتریسی کمتر سریعتر از LSTM است.

سوال ۳)

در این سوال می‌خواهیم با به هم چسباندن لایه های LSTM و GRU مختلف، یک مدل عمیق تر تولید کنیم و برای دسته بندی شاخص کل استفاده کنیم. برای این منظور، باید پارامتر **return_sequences** لایه های LSTM و GRU میانی را برابر با True قرار دهیم. با این کار، لایه های بعدی از این خروجی ها به عنوان ورودی خود استفاده می‌کنند و از یادگرفته های لایه های قبلی به عنوان نقطه ای برای شروع است بهره می‌برند.

آزمایش های مربوط به چندین لایه LSTM:

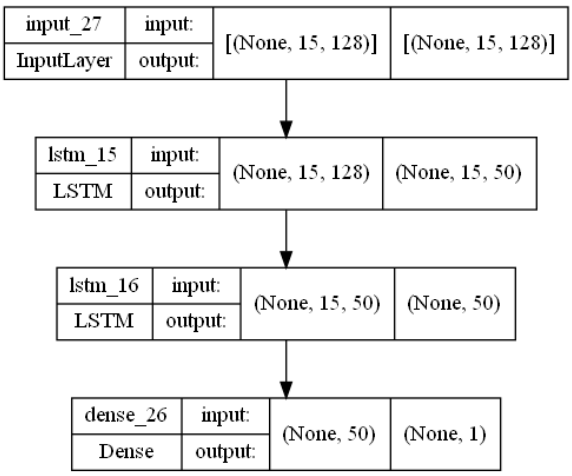
در این بخش می‌خواهیم شاخص کل بازار را به کمک چندین لایه **LSTM** که بر روی یکدیگر سوار شده اند پیش بینی کنیم. نتایج آزمایش های مربوط به این بخش را در جدول زیر مشاهده میکنید.

در این آزمایش‌ها از **binary_crossentropy** به عنوان loss و از **Adam** به عنوان Optimizer استفاده شد. تعداد epoch برابر ۵۰ در نظر گرفته شد.

نتایج دسته بندی به کمک یک لایه LSTM

توضیحات مدل	دقت آموزش	دقت تست
LSTM(50,50)	۰,۶۸۸۳	**۰,۵۷۷۸**
LSTM(50,50,50)	۰,۶۴۹۴	**۰,۴۶۶۷**
LSTM(50,50,50,50)	۰,۶۳۶۴	**۰,۵۱۱۱**

دیارگرام مربوط به شبکه برتر را در تصویر زیر مشاهده میکنید.



آزمایش های مربوط به چندین لایه GRU:

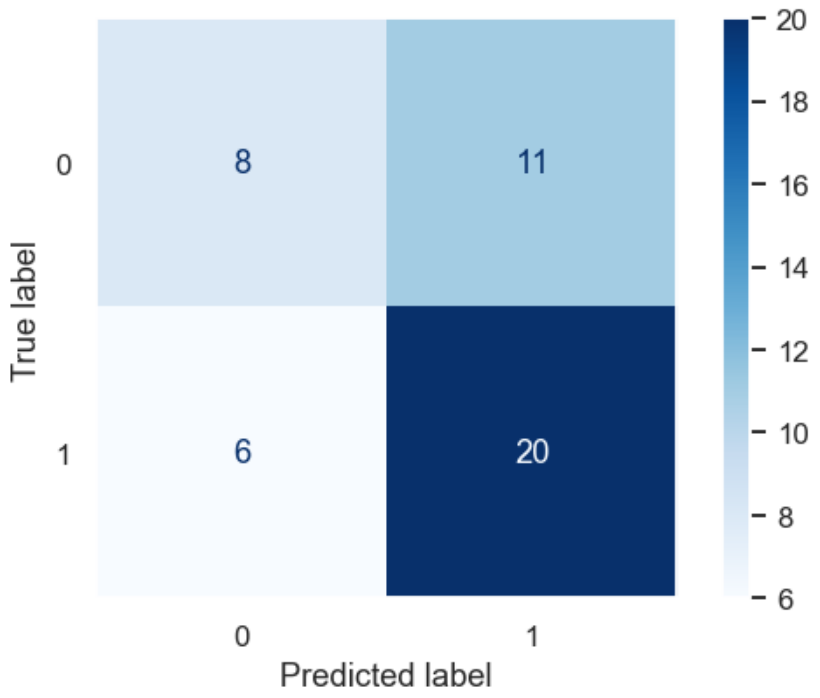
در این بخش می‌خواهیم شاخص کل بازار را به کمک چندین لایه **GRU** که بر روی یکدیگر سوار شده اند پیش بینی کنیم. نتایج آزمایش های مربوط به این بخش را در جدول زیر مشاهده میکنید.

در این آزمایش‌ها از **binary_crossentropy** به عنوان loss و از **Adam** به عنوان Optimizer استفاده شد. تعداد epoch برابر ۵۰ در نظر گرفته شد.

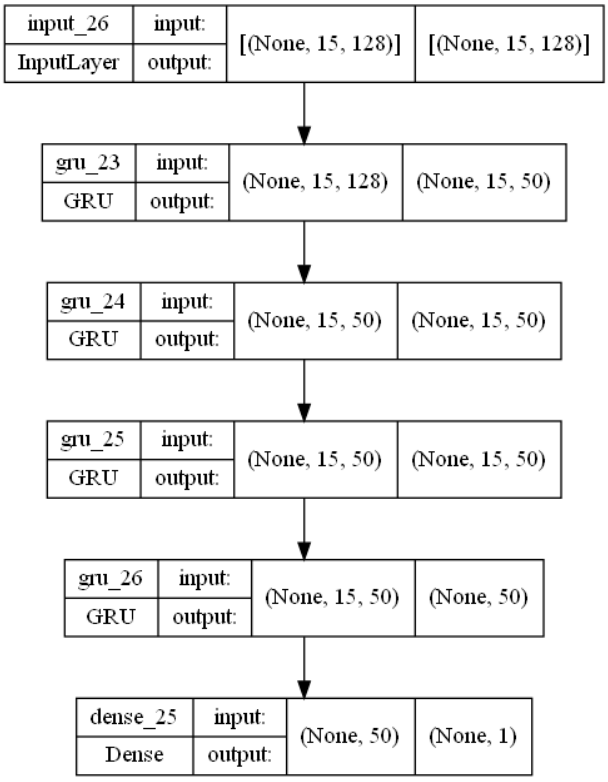
نتایج دسته بندی به کمک یک لایه LSTM

دقت تست	دقت آموزش	توضیحات مدل
۰,۵۷۷۷	۰,۶۶۲۳	GRU(50,50)
۰,۶۰۰۰	۰,۶۹۴۸	GRU(50,50,50)
۰,۶۲۲۲	۰,۷۲۷۳	GRU(50,50,50,50)

ماتریس درهم‌ریختگی مربوط به بهترین مدل این بخش:



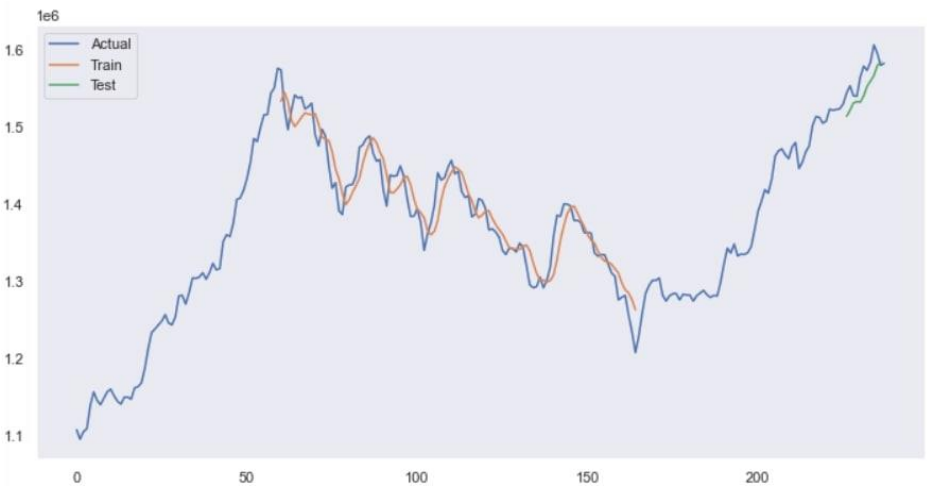
دیارگرام مربوط به شبکه برتر را در تصویر زیر مشاهده میکنید.



جمع بندی و نتیجه گیری مدل های Stacked:

با توجه به نتایج بدست آمده در دو آزمایش قبل میتوان گفت که معماری GRU نسبت به LSTM، رابطه بهتری با افزایش لایه ها دارد. طبق توضیحات داده شده در بخش اول این تمرین، معماری LSTM برای مسائلی با طول Sequence بالا بهتر است. در این مثال طول Sequence ۱۵ است که آنقدر بالا نیست.

از آنجایی که در این تمرین به نتیجه قابل قبولی نرسیدیم، برای آزمایش و بررسی داده ها و مدل، یک مسئله رگرسیون برای پیش بینی قیمت Close شاخص کل را حل کردیم. در این سوال بر خلاف سوال اصلی تنها از ویژگی Close دیتاست شاخص کل استفاده کردیم. به این ترتیب ۲۲۲ داده و فقط ۱ فیچر داشتیم. عمل رگرسیون را با کمک معماری LSTM انجام دادیم و نتایج نسبتا قابل قبول بودند. با این اوصاف حدس میزنیم داشتن ۱۲۸ ویژگی و فقط ۲۲۲ داده در مسئله اصلی یکی از دلایل اصلی به نتیجه نرسیدن ما بود.



در این تصویر، منحنی نارنجی رنگ تخمین مدل در حین آموزش، پس از دیدن قیمت Close ۶۰ روز قبل از هر نقطه است. و منحنی سبزرنگ، نتیجه مشابه برای مجموعه داده آزمون می‌باشد. همچنین تخمین زدن قیمت روز های متوالی به صورت Autoregressive انجام شده به این معنی که پیش بینی یک روز به مجموعه داده ورودی برای تخمین روز بعد اضافه میشود.

پایان