به نام خدا

دانشگاه تهران

دانشکده مهندسی برق و کامپیوتر

شبکه های عصبی و یادگیری عمیق

استاد

دکتر کلهر

مینی پروژه دوم

نام و نام خانوادگی	شقایق طربخواه- سحر رجبی
شماره دانشجویی	810199165-810698193
تاریخ ارسال گزارش	1400/04/04

فهرست گزارش سوالات

سوال 1 – آشنایی با کاربرد (شبکه های عصبی بازگشتی) در سری زمانی

یکی از مسائلی که از گذشته تا به امروز از اهمیت بالایی برخوردار بوده است، مسائل پیشبینی سری های زمانی است. این دسته از مسائل کاربرد شبکه عصبی را پررنگ می کند. در این سوال قصد بر این است که با کمک این نوع از شبکه ها ارزش بیت کوین را بر حسب دلار پیشبینی می کنیم. برای این منظور از مجموعه دادگان شرکت یاهو استفاده می کنیم. این سوال از اطلاعات "BTC-USD" در بازه 2014/9/20 تا 2021/5/20 برای پیشبینی بهره می گیریم.

انتظار است در طول این سوال شبکه عصبی طراحی کنید که داده ها را به صورت سری زمانی 24 الی 28 روزه دریافت کند و و پارامتر "CLOSE" را برای روز آینده پیشبینی کند.

قسمت 1:

در انجام پیش پردازش های لازم بر روی دادگان به این نکته توجه کنید که حداقل به میزان 7 روز داده برای تست وجود داشته باشد، سپس برای هر یک از شبکه هایی که طراحی می کنید نمودار تابع هزینه برای روند یادگیری، ارزیابی و همچنین نمودار مقدار حقیقی به همراه مقدار پیشبینی شده را برای بازه زمانی تست را رسم کنید.

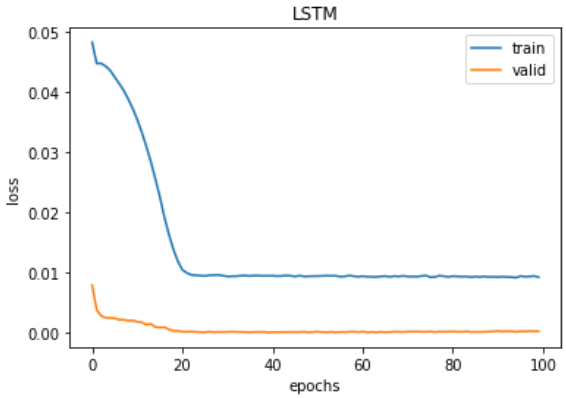
در این قسمت پس از فراخوانی داده های مربوطه و نرمال سازی های لازم برای رسیدن به نتیجه بهتر اقدام به تقسیم بندی مناسب داده ها به نحو مناسبی از یکدیگر جدا شوند. به منظور تست داده ها از بازه زمانی 100 روزه ای استفاده می شود که محل شروع آن به صورت تصادفی انتخاب می شود. در تولید داده‌های آموزش هم باید این نکته که تعدادی روز در میانه‌ی داده‌ها حذف شده‌اند را در نظر بگیریم تا با توجه به time-series بودن داده‌ها، به مشکلی برنخوریم. در ادامه نیز توابع مربوطه به منظور جداسازی ویژگی ها و هدف، تولید batch های مناسب، کلاس روند مدل انتخابی و رسم نمودار های مربوطه آورده می شوند.

قسمت 2:

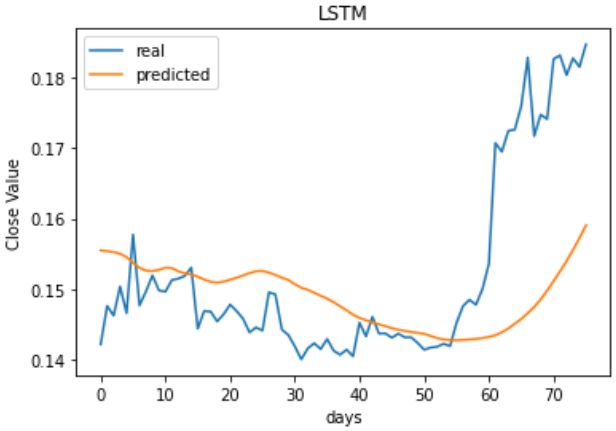
شبکه را با سلول های ساده GRU، LSTM و RNN طراحی کرده نتایج را ارائه و با یکدیگر مقایسه کنید.

2.1) نتایج حاصل از LSTM :

حال مدل مربوط به معماری شبکه را باید بسازیم. در این قسمت از شبکه‌های LSTM استفاده گردیده است. بدین صورت که دو لایه از LSTM و در لایه آخر از یک لایه Dense به منظور مشخص کردن قیمت بیت کوین در روز آینده استفاده شده است. تابع هزینه در این مدل، mean squared error اتخاذ گردیده است و برای تابع بهینه‌ساز مربوطه از Adam بهره برده‌ایم. این شبکه را با batch size برابر با 128 در طی 100 epoch ترین خواهیم کرد. نمودار خطا برای دادگان آموزش و ارزیابی در زیر مشاهده می شود:



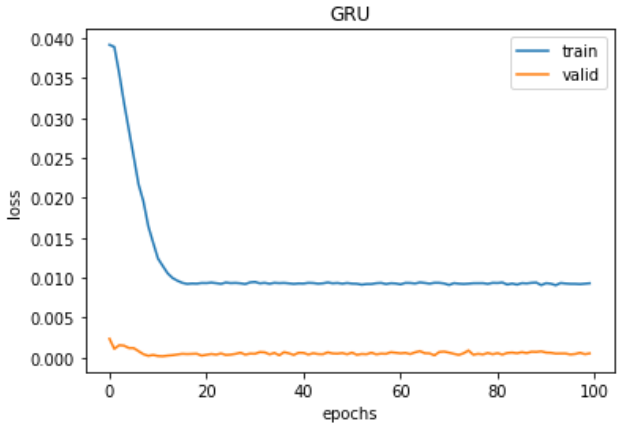
شکل ۱- نمودار تغییرات loss برای داده‌های آموزش و ارزیابی در معماری *LSTM*



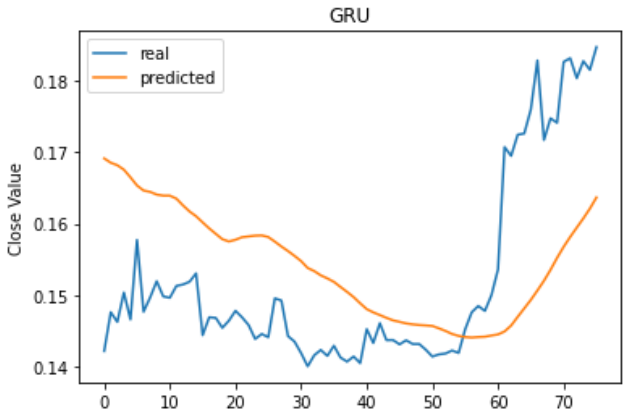
شکل ۲- نمودار مقادیر واقعی و مقادیر پیش‌بینی شده در معماری LSTM

2.2) نتایج حاصل از GRU :

در این قسمت از شبکه‌های GRU استفاده گردیده است. بدین صورت که دو لایه از GRU و در لایه آخر از یک لایه Dense به منظور مشخص کردن قیمت بیت کوین در روز آینده استفاده شده است. تابع هزینه در این مدل، mean squared error اتخاذ گردیده است و برای تابع بهینه‌ساز مربوطه از Adam بهره برده‌ایم. این شبکه را با batch size برابر با 128 در طی 100 epoch ترین خواهیم کرد. نمودار خطا برای دادگان آموزش و ارزیابی در ادامه مشاهده می شود:



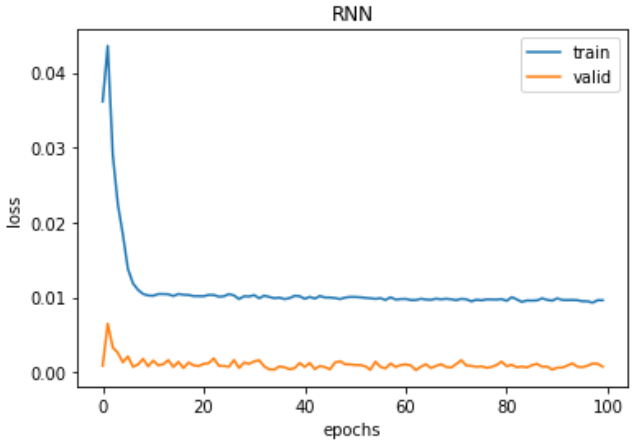
شکل ۳- نمودار تغییرات loss برای داده‌های آموزش و ارزیابی در معماری GRU



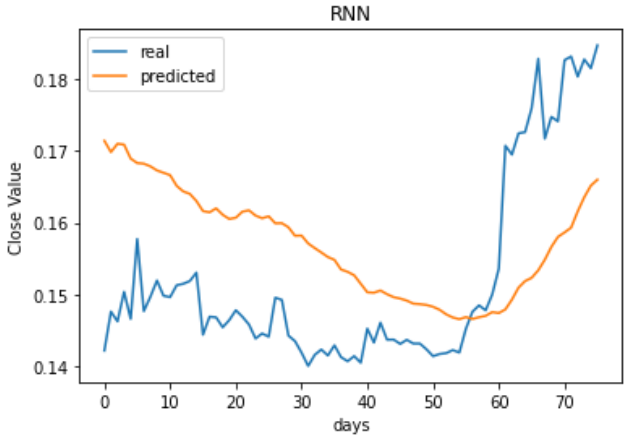
شکل ۴- نمودار مقادیر واقعی و مقادیر پیش‌بینی شده در معماری GRU

2.3) نتایج حاصل از RNN :

در این قسمت از شبکه‌های RNN استفاده گردیده است. بدین صورت که دو لایه از RNN و در لایه آخر از یک لایه Dense به منظور مشخص کردن قیمت بیت کوین در روز آینده استفاده شده است. تابع هزینه در این مدل، mean squared error اتخاذ گردیده است و برای تابع بهینه‌ساز مربوطه از Adam بهره برده‌ایم. این شبکه را با batch size برابر با 128 در طی 100 epoch ترین خواهیم کرد. نمودار خطا برای دادگان آموزش و ارزیابی در ادامه مشاهده می شود:



شکل ۵- نمودار تغییرات loss برای داده‌های آموزش و ارزیابی در معماری RNN



شکل ۶- نمودار مقادیر واقعی و مقادیر پیش‌بینی شده در معماری RNN

حال به مقایسه سه شبکه ترین شده خواهیم پرداخت که نتایج کلی در جدول 1 آورده شده است.

جدول1: مقایسه سه شبکه RNN, GRU و LSTM

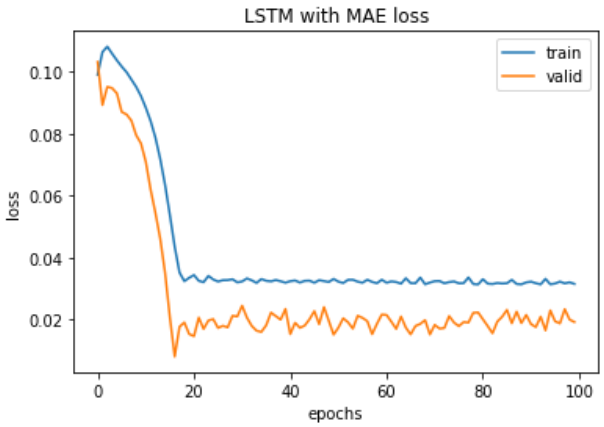
	Run Time	Train Loss	Test Loss
LSTM	8	0.01001	0.00024
GRU	7.29	0.01003	0.000405
RNN	7.32	0.0101	0.0011

با توجه به آنچه در جدول بالا مشاهده می شود از لحاظ زمانی lstm اندکی بیشتر از سایر روش ها هزینه دارد ولی در عین حال خطای بهتری را نسبت به دو روش دیگر دارد. این در حالی است که با توجه به اینکه روش GRU دارای کمترین زمان است اما خطای بالاتری را به ویژه در مورد داده های ارزیابی از خود نشان می دهد. همچنین با توجه به نمودارهای مقدار واقعی و مقدار پیش‌بینی شده، مقادیر پیش‌بینی‌شده برای داده‌های تست، در روش lstm نزدیک‌تر به مقادیر واقعی هستند. همچنین عملکرد ضعیف rnn با توجه به long-term dependency موجود بین داده‌ها قابل درک است. با توجه به اینکه اختلاف زمانی بین روش های مختلف زیاد نیست اما خطای کمتری را منتج می شود معماری lstm به عنوان برگزیده سه روش قبل در بخش های بعدی سوال مورد استفاده قرار می گیرد.

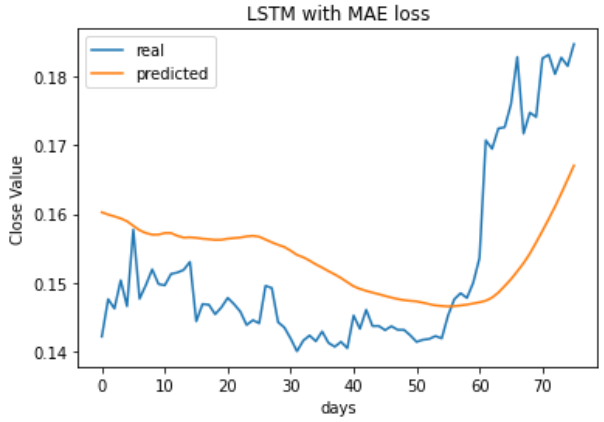
قسمت 3:

تعداد ایپاک و سایر پارامترها در مقایسه‌های زیر ثابت گرفته‌شده‌اند.

در ابتدا با استفاده از تابع خطای MAE نتایج زیر حاصل می شود:



شکل ۷- نمودار تغییرات loss برای داده‌های آموزش و ارزیابی با تابع هزینه MAE

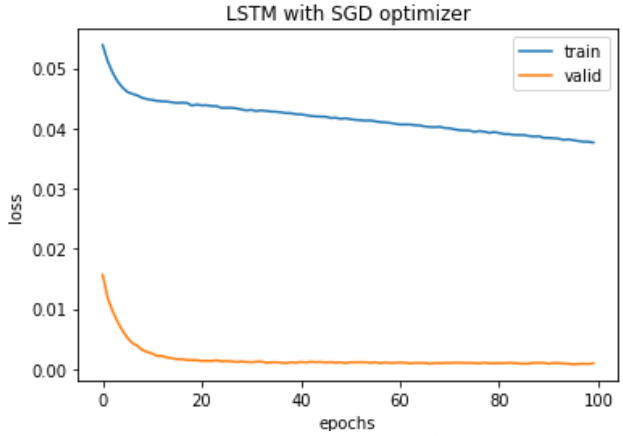


شکل ۸- نمودار مقادیر واقعی و مقادیر پیش‌بینی شده با تابع هزینه‌ی MAE

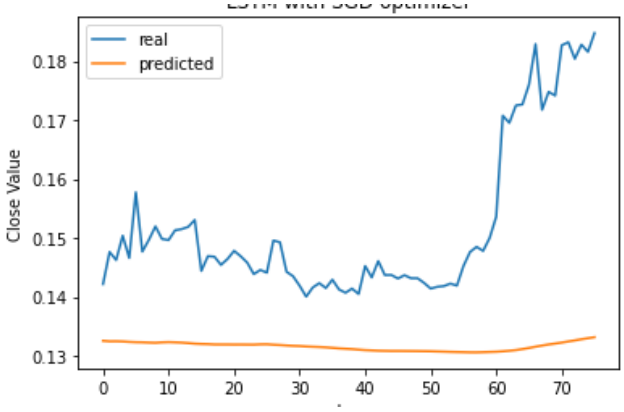
همانطور که مشخص است، در مقایسه با شکل ۱ و ۲ که همین معماری با تابع هزینه‌ی MSE را نشان می‌دهد؛ دقت مقادیر پیش‌بینی شده در دادگان تست کاهش یافته و در روند کاهش جریان loss هم شامل نوسانات بیشتری هستیم.

بهینه‌ساز استفاده شد در شبکه‌ی اصلی adam است و ما در ادامه سه بهینه‌ساز دیگر را با آن مقایسه خواهیم کرد.

با استفاده از optimizer = SGD نتایج زیر حاصل می شود:



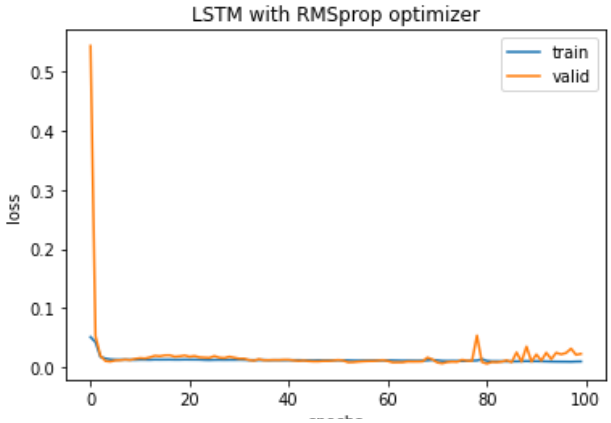
شکل ۹- نمودار تغییرات loss برای داده‌های آموزش و ارزیابی با بهینه‌ساز SGD



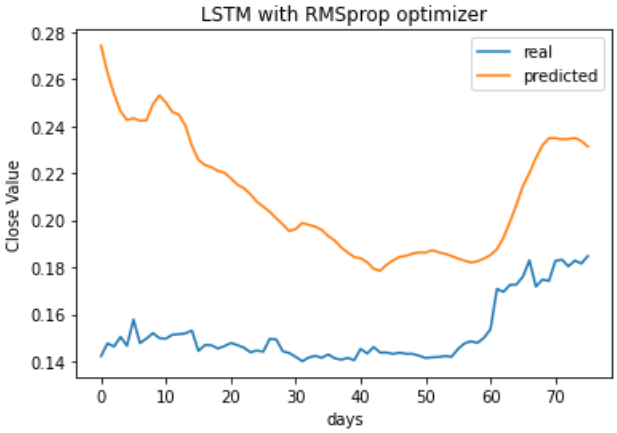
شکل ۱۰- نمودار مقادیر واقعی و مقادیر پیش‌بینی شده با تابع بهینه‌ساز SGD

مجددا در مقایسه با شکل ۱ و ۲، به نظرمی‌رسد که تا حد زیادی، یادگیری اصلا انجام نشده است و دقت پیش‌بینی مدل به صورت قابل توجهی کاهش یافته است و حتی در پیش‌بینی کاهش یا افزایش‌های کلی هم موفق عمل نکرده است.

با استفاده از optimizer = RMSPROP نتایج زیر به دست می آید:



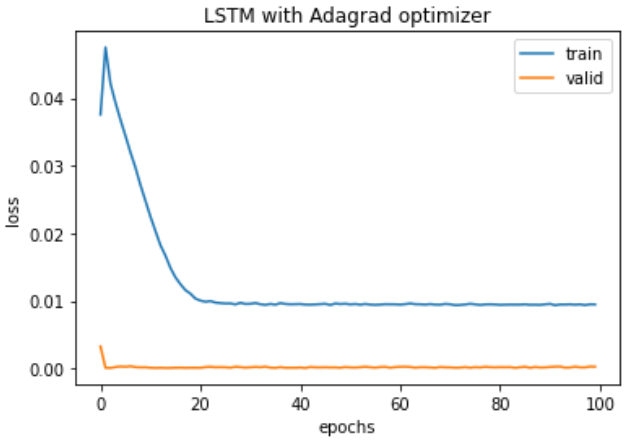
شکل ۱۱- نمودار تغییرات loss برای داده‌های آموزش و ارزیابی با بهینه‌ساز RMSprop



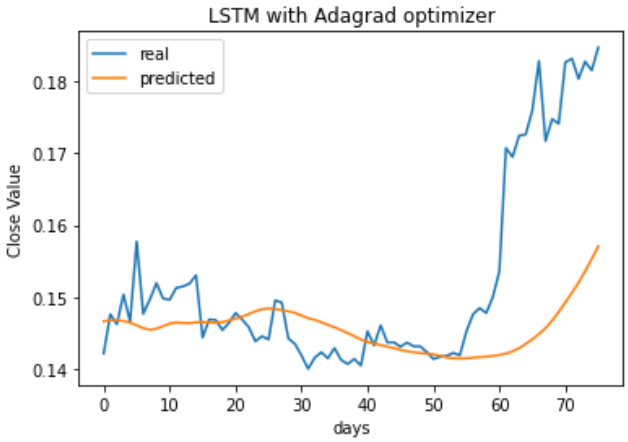
شکل ۱۲- نمودار مقادیر واقعی و مقادیر پیش‌بینی شده با تابع بهینه‌ساز RMSprop

در این بهینه‌ساز هم دقت پیش‌بینی تا حد بسیار زیادی کاهش یافته و موفق به آموزش مدل در تعداد ایپاک‌های مشخص شده نبوده.

در نهایت با استفاده از optimizer = AdaGrad خواهیم داشت:



شکل ۱۳- نمودار تغییرات loss برای داده‌های آموزش و ارزیابی با بهینه‌ساز Adagrad



شکل ۱۴- نمودار مقادیر واقعی و مقادیر پیش‌بینی شده با تابع بهینه‌ساز Adagrad

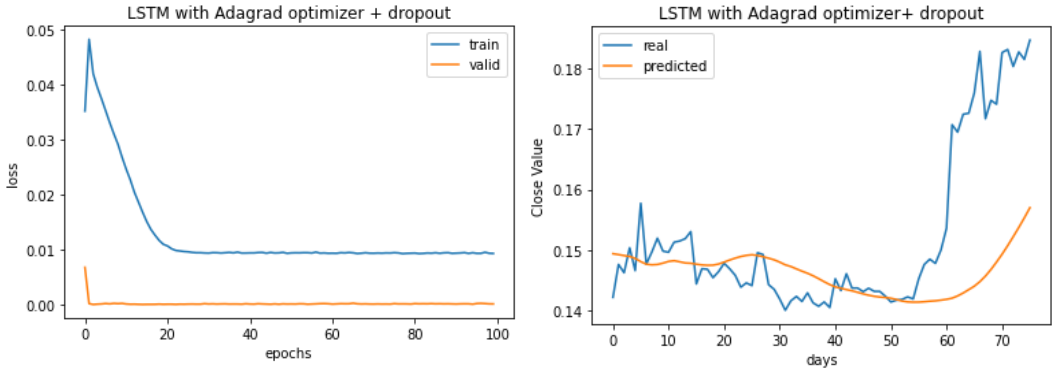
دقت پیش‌بینی در این بهینه‌ساز از نظر پیش‌بینی روند کاهش و افزایش تا حد زیادی شبیه به بهینه‌ساز adam است، در حالی که دقت آن هم کمی افزایش یافته است. در نتیجه به نظر میرسد که کارایی بهتری نسبت به adam در این مساله داشته‌باشد.

در مراحل بعدی، از معماری lstm با تابع هزینه‌ی MSE و تابع بهینه‌ساز Adagrad به عنوان معماری برگزیده استفاده می‌شود.

قسمت 4 :

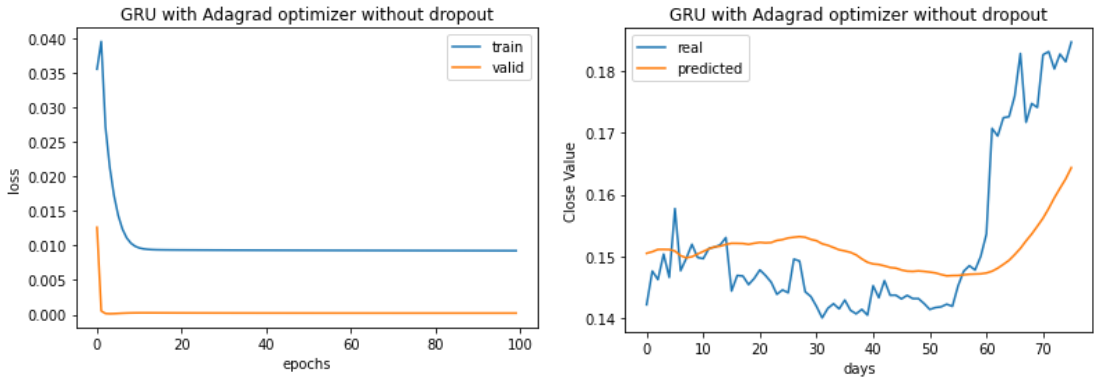
برای هر سه معماری lstm، gru و rnn تاثیر حذف dropout با توابع هزینه و بهینه‌ساز برگزیده در زیر گزارش شده است.

برای معماری lstm نتایج با اعمال dropout را مشاهده می‌کنید.



شکل ۱۵- نمودارهای عملکرد مدل lstm با اعمال dropout در لایه‌های بازگشتی

در زیر نتیجه حذف drop out مشاهده می شود.

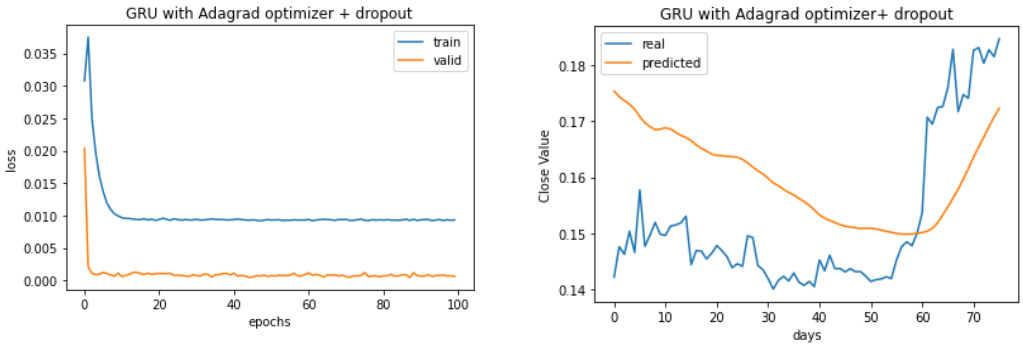


شکل ۱۶- نمودارهای عملکرد مدل lstm با حذف dropout در لایه‌های بازگشتی

با وجود اینکه به نظر میرسد روند کاهش مقدار loss و مقادیر نهایی در هر دوی آن‌ها یکی باشد، اما دقت پیش‌بینی پس از حذف dropout کاهش پیدا کرده و به نظر می‌رسد که کمی overfitting در روند یادگیری وجود داشته‌است که این اختلاف به‌وجود آمده.

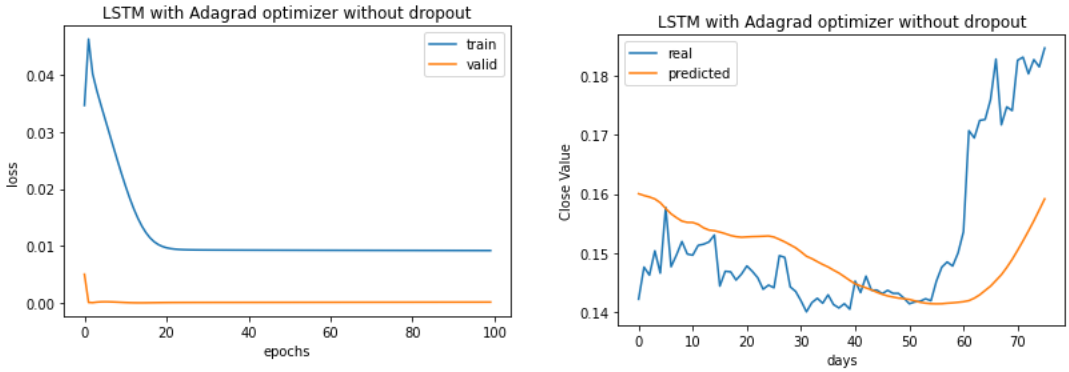
در ادامه برای دو شبکه GRU و RNN مشاهده می شود که نتایج زیر به دست می آید:

برای GRU، با اعمال dropout نتایج زیر را شاهدیم.



شکل ۱۷- نمودارهای عملکرد مدل gru با اعمال dropout در لایه‌های بازگشتی

با حذف dropout نمودارهای زیر حاصل میشوند.

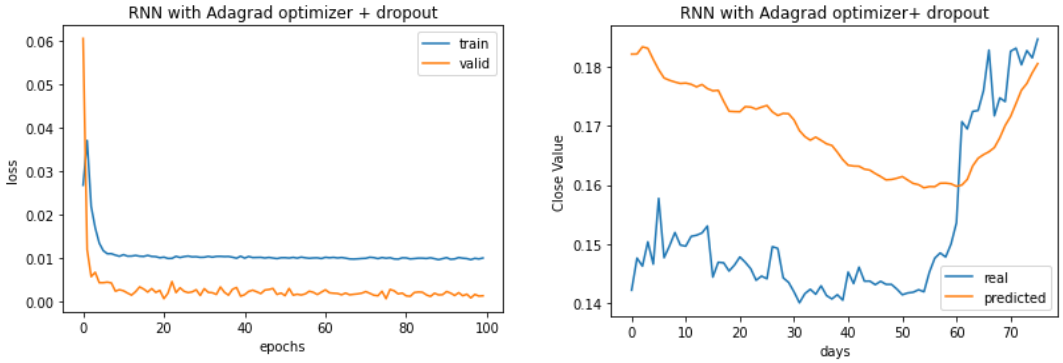


شکل ۱۸- نمودارهای عملکرد مدل gru با حذف dropout در لایه‌های بازگشتی

این بار حذف dropout باعث افزایش دقت پیش‌بینی مدل شده است و به نظر میرسد که اعمال آن‌ها روند یادگیری را تا حد مختل کرده بوده و با توجه به اینکه gru گیت‌های کمتری هم برای یادگیری دارد، می‌توان این مشاهده را توجیه کرد و شاید بهتر باشد که مقدار dropout را کمتر در نظر میگرفتیم.

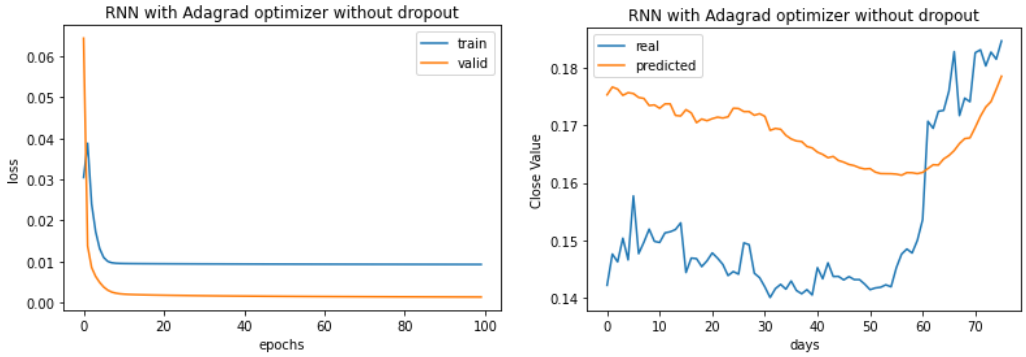
برای RNN داریم:

با افزودن drop out :



شکل ۱۹- نمودارهای عملکرد مدل rnn با اعمال dropout در لایه‌های بازگشتی

برای مشاهده تاثیر drop out در ادامه حالت بدون این تغییر را مشاهده می کنیم:

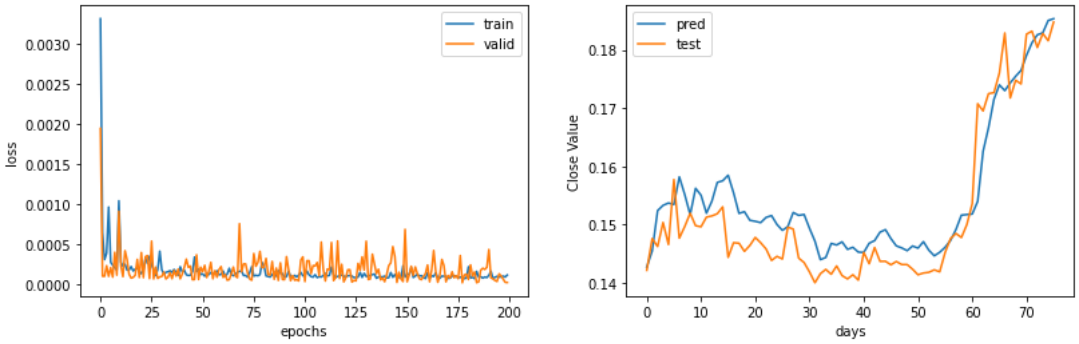


شکل ۲۰- نمودارهای عملکرد مدل rnn با حذف dropout در لایه‌های بازگشتی

در مجموع شبکه‌ی RNN عملکرد مناسبی نداشته و حذف dropput هم نمی‌توان برای آن یک بهبود و یا برعکس در نظر گرفت. چرا که این شبکه درکل موفق نبوده. اما در نمودار تغییرات loss، اینکه نوسان کمتری با حذف dropout داریم شاید نشان از آموزش بهتر مدل باشد و در این معماری هم به آن نیازی نباشد و یا مانند gru خوب باشد که از احتمال کمتری برای آن استفاده کنیم.

قسمت 5:

برای پیاده‌سازی این قسمت از یک معماری با دو لایه‌ی مخفی با ۲۵۶ و ۱۲۸ نورون استفاده کردیم که تابع فعال‌ساز آن‌ها relu، تابع هزینه mse و بهینه‌ساز ما adam است. آموزش را با ۲۰۰ ایپاک بر روی دادگان انجام دادیم. همچنین برای آموزش، داده‌های ۲۴ روز متوالی به عنوان ویژگی برای پیش‌بینی close روز بعد استفاده شده‌اند و عملکرد شبکه را می‌توانید در شکل زیر مشاهده کنید.



شکل ۲۱- نمودارهای عملکرد مدل mlp

همانطور که مشخص است، پیش‌بینی در این مدل، به مراتب بهتر از مدل‌های پیشین است. در یک lstm، ما به صورت sequential داده‌ها را به شبکه میدادیم و در اینجا به صورت یکجا این کار را کرده‌ایم. از آنجایی که بردار ویژگی ما در این قسمت به صورت یکجا به مدل داده می‌شود، تعداد این ویژگی‌ها هم زیاد نیست و نیازی به sequenceهای با طول متفاوت هم نداریم، در نتیجه مدل mlp توانسته با کاهش پیچیدگی مدل، به نتیجه‌ی بهتری دست پیدا کند. اما در صورتی که ما نیاز به sequenceهای با طول متفاوت داشتیم، این شبکه کمک چندانی نمی‌کرد.

جدول۲: مقایسه شبکه‌ی lstm و mlp

	Train Loss	Test Loss
LSTM	0.01001	0.00024
MLP	1.104e-4	2.682e-5

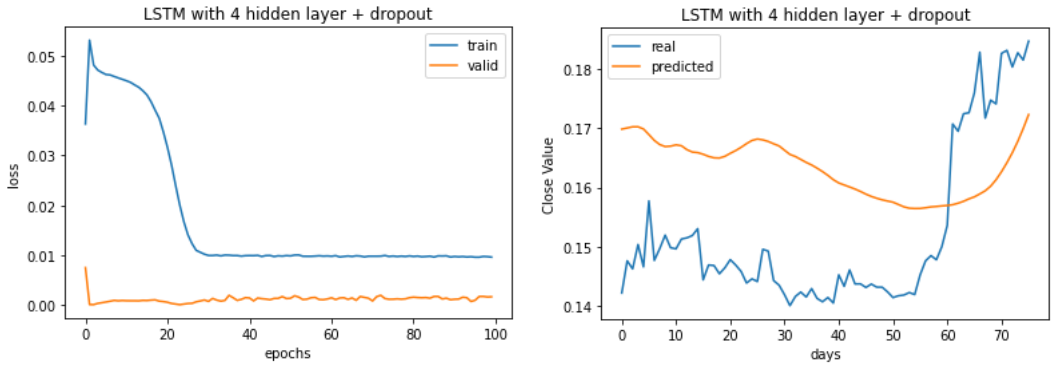
مشخص است که مقدار loss در این مدل بسیار پایین‌تر است.

قسمت6:

گاهی برخی از ویژگی‌ها تاثیر چندانی در پیش‌بینی ندارند. این به این معنا نیست که ارتباطی بین آن‌ها و پاسخ نهایی نیست؛ در برخی از حالات، بین دو ویژگی correlation قوی وجود دارد و میتوان یکی از آن‌ها را حذف کرد و... یکی از روش‌های تعیین ویژگی‌های بهتر، backward elimination است. به این صورت که ویژگی‌هایی که باعث کاهش دقت می‌شوند را یکی یکی حذف می‌کنیم تا به بهترین بردار ویژگی برسیم.

قسمت7:

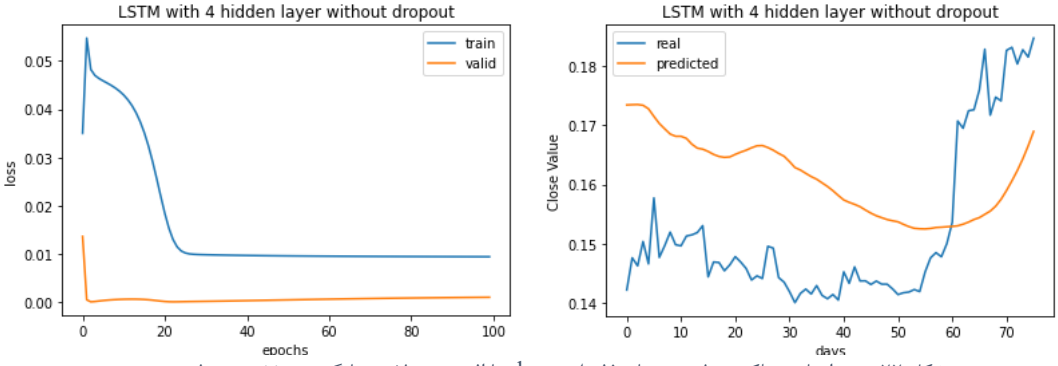
در این بخش، ابتدا نتایج افزودن دو لایه‌ی مخفی به معماری برنده (که دارای dropout برابر ۰.۳ است) بررسی می‌کنیم.



شکل ۲۲- نمودارهای عملکرد مدل برنده با dropout، با افزودن دو لایه‌ی بازگشتی مخفی به مدل

همانطور که مشخص است، دقت مدل با رساندن تعداد لایه‌های مخفی بازگشتی از ۲ به ۴ کاهش قابل توجهی داشته و افزایش لایه لزوما باعث بهبود عملکرد شبکه نمی‌شود. هرچند تا یک تعدادی لایه برای مدل کردن الگوها نیاز است؛ اما بیشتر از این تعداد ممکن است باعث overfit و کاهش دقت دادگان تست بشود.

سپس با حذف dropout تعداد لایه‌ها را به ۴ میرسانیم که نتایج به این صورت خواهد بود.



شکل ۲۲- نمودارهای عملکرد مدل برنده با حذف dropout، با افزودن دو لایه‌ی بازگشتی مخفی به مدل

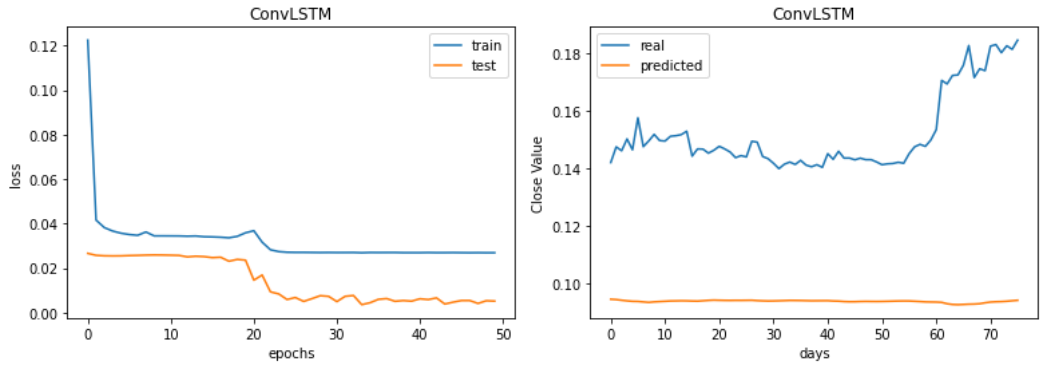
و عملکرد باز هم بدتر شده و نتیجه‌ی مطلوبی بدست نیامده است.

قسمت 8:

شبکه‌های convlstm در واقع از ترکیب دو شبکه CNN و LSTM حاصل میشوند. در حالی که CNN شبکه‌ای برای استخراج ویژگی و LSTM شبکه‌ای برای پیش‌بینی داده‌های sequential است، این شبکه LSTMای ارائه می‌کند که در سلول‌های خود convolution هم می‌گیرد.

ورودی این شبکه‌ها به صورت (batch size, seq len, num of channels, row, col) است. ورودی سلول LSTM باید به فرمت (batch size, seq len, features) باشد و ورودی CNN هم (batch size, num of channels, row, col) که با ترکیب این دو، مشخص است که چرا فرمت داده‌های ورودی ConvLSTM باید به صورت گفته شده باشد.

ما در ابتدا برای این قسمت، دادگان خود را reshape کردیم (تعداد کانال‌ها در این داده برابر با ۱ است) و سپس یک شبکه با دو لایه‌ی ConvLSTM2D و یک لایه‌ی fully connected با ۵۰ نورون پیاده‌سازی کردیم. با ترین این مدل در ۵۰ ایپاک، نتایج زیر حاصل شد:



شکل ۲۳- نمودارهای عملکرد مدل برنده پیاده‌سازی شده با لایه‌های ConvLSTM

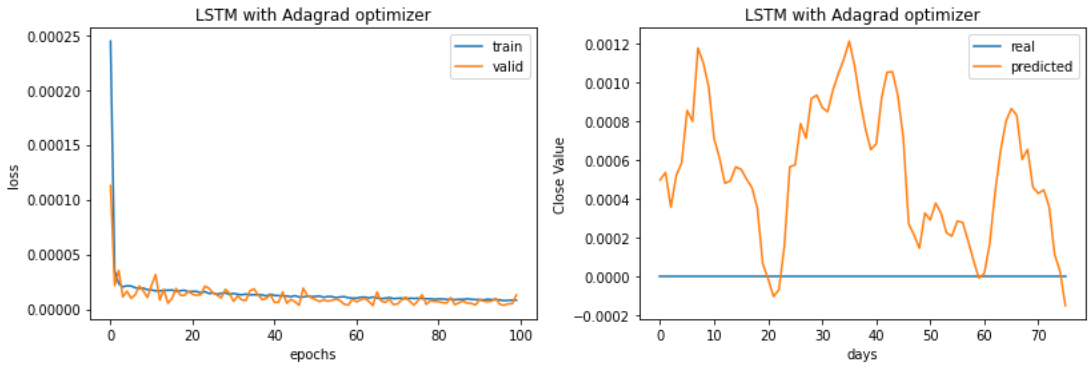
همانطور که مشخص است، مدل اصلا پیش‌بینی خوبی از روند کاهش‌ها و افزایش‌ها ندارد (اختلاف اما در حد ۰.۰۶ است). با توجه به ذات شبکه‌های CNN، به نظر می‌رسد که اگر ویژگی‌هایی به داده اضافه کنیم، مثلا نتایج تحلیل‌های تکنیکال بر روی داده، که ویژگی‌های تغییرات داده را نسبت به سایر نقاط هم وارد داده‌ی ما بکند، آنگاه این معماری عملکرد بهتری خواهد داشت. (همانند تحلیل‌هایی که در بازار سهام انجام می‌شود.)

قسمت 10:

ما در سوالات قبلی، از normalization به عنوان preprocessing داده‌ها استفاده کردیم. چرا که کوچکتر شدن داده‌ها، و قرار دادن آن‌ها بین رنج ۰ تا ۱ باعث یادگیری سریع‌تر و راحت‌تر داده‌ها خواهد شد.

در ادامه، نتایج اعمال دو preprocess دیگر را با بررسی می‌کنیم.

نتایج power transform:



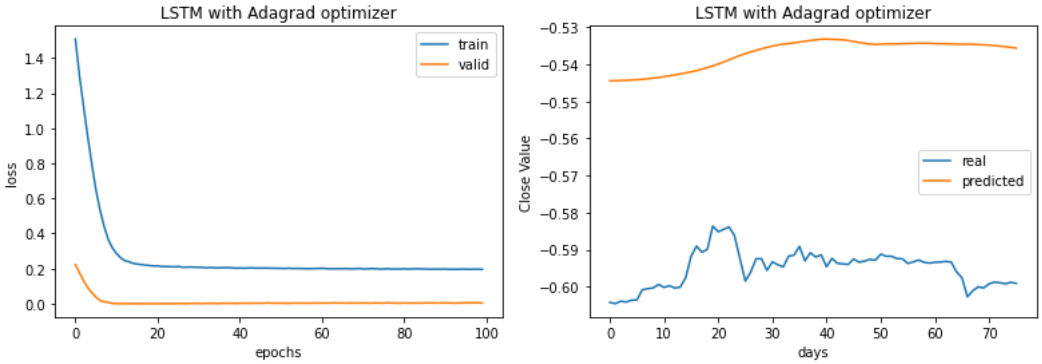
شکل ۲۴- نمودارهای عملکرد مدل آموزش دیده با تبدیل power

بعد از اعمال این تبدیل بر روی داده‌ها، باعث می‌شود که آن‌ها بیشتر شبیه به توزیع نرمال باشند. از آنجایی که داده‌های قیمت بیتکوین، اصلا نزدیک به توزیع نرمال هم نیستند، باعث می‌شود که عملکرد شبکه به شدت افت کند و در طی ۱۰۰ ایپاک تقریبا یادگیری صورت نگیرد.

قیمت بیتکوین بعد از اعمال این تبدیل صفر میشود. و تنها اعداد غیر صفر حجم معاملات در هر روز است که قطعا بیشتر از قیمت بیتکوین توزیعی شبیه به نرمال دارد. همانطور که مشخص است، این تبدیل به کلی دادگان ما را از بین میبرد و زمانی که داده‌ای نیست، یادگیری‌ای هم نیست.

نتایج standardization:

این تبدیل باعث می‌شود که واریانس داده برابر با ۱ و میانگین آن صفر باشد. بعد از اعمال این تبدیل، نتایج زیر بدست آمد:



شکل ۲۵- نمودارهای عملکرد مدل آموزش دیده با تبدیل standardization

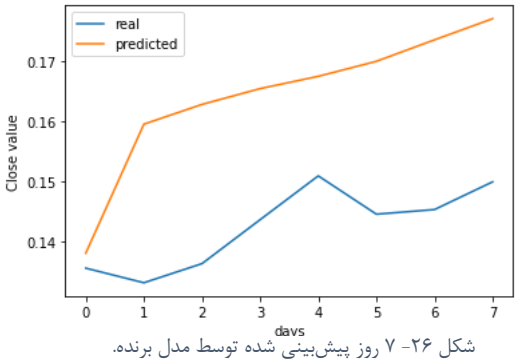
همانطور که مشخص است، نوسانات خیلی خوب پیش‌بینی نشده‌اند اما تغییرات تا حدی در این اعداد دیده می‌شود. اما بهرحال دقت پیش‌بینی قابل قبول نیست.

تا اینجا، بهترین تبدیل به نظر می‌رسد که normalization باشد.

قسمت 11:

ساختار برنده یک شبکه‌ی lstm با دو لایه‌ی بازگشتی و dropout برابر با 0.3 است که برای آموزش آن از تابع هزینه‌ی mse و الگوریتم بهینه‌سازی adagrad استفاده شده است. همچنین تبدیل MinMaxNormalization هم به عنوان بهترین تبدیل برای مرحله‌ی preprocessing داده انتخاب شده است.

نتایج حاصل از پیش‌بینی با این مدل، در تصویر زیر قابل ملاحظه است.



شکل ۲۶- ۷ روز پیش‌بینی شده توسط مدل برنده.

در اینجا هر روزی که پیش‌بینی شده، به عنوان داده‌ی روز بعد، مورد استفاده قرار می‌گیرد.

سوال3 – آشنایی با کاربرد (شبکه های عصبی بازگشتی) در متن

در این سوال به دنبال پردازش داده های متنی از جنس توئیت با استفاده از شبکه های عصبی بازگشتی هستیم. و خواسته شده است تا با استفاده از کلمات گفته شده در توئییت منفی یا مثبت بودن کلیات جمله را تشخیص داده و بر این مبنا تقسیم بندی کنیم.

پاسخ :

سوال 1:

اگر دادگان ما به صورتی باشد که تعداد آنها در کلاس های مختلف با یکدیگر متفاوت باشد شناسایی درست دادگان با توزیع کمتر سخت تر از کلاس های دیگر می باشد و احتمال پیشبینی اشتباه در آنها نیز بیشتر است. همچنین ممکن است که چون وزن دیگر کلاس ها بیشتر می باشد، شبکه نسبت به تغییرات در آنها حساسیت بیشتری نشان داده و با ایجاد اندکی نویز در آنها، در کلاس دیگر با مشکل مواجه شویم.

برای رفع این مشکل می توان از راهکارهای متفاوتی استفاده کرد. ابتدایی ترین راهی که به نظر می رسد آن است که در صورت امکان تعداد داده های موجود در کلاس کمتر را افزایش دهیم. به این منظور می توان از روش های مختلف data augmentation در مورد داده های متنی استفاده نمود. در یکی از این روش ها می توان از توابع و کتابخانه هایی استفاده کرد که با استفاده از آنها با جایگذاری کلمات هم معنی در درون جملات، داده های جدیدی تولید کرد که انتخاب جملات و کلمات کاملا به صورت تصادفی صورت گرفته تا به نحوی تعداد جملات زیاد شوند که با کلاس دیگر برابر باشند. همچنین می توان با استفاده از توابع موجود در پایتون به صورت تصادفی برخی از کلمات غیر ضروری در داخل جملات را حذف کرده و جملات جدید را تولید کرد تا عمل توازن صورت گیرد. روش‌های دیگری هم مانند حذف از کلاس با جمعیت بیشتر و یا اضافه کردن برخی کلمات تصادفی به متن داده‌های با تعداد کمتر، و اضافه کردن جملات جدید به داده‌ها هم وجود دارد.
در اینجا به علت اختلاف بسیار زیاد بین تعداد نمونه‌های دو کلاس، ما در ابتدا تعدادی از داده‌های کلاس با جمعیت بیشتر را حذف کردیم، سپس جملاتی از کلاس با جمعیت کمتر را به صورت تصادفی انتخاب کرده، دو تا از کلمات آین جمله‌ها را به طور تصادفی با واژگان هم‌معنی آن‌ها جایگزین کردیم و به عنوان جمله‌ی جدید استفاده کردیم.
همچنین در تفسیر داده ها نیز می توان در نظر گرفت که تنها دقت مد نظر قرار نگیرید. به طور مثال ممکن است که دقت در کلاسی که تعداد پایین تری دارد عدد مناسبی نباشد اما از انجا که سهم بسیار کمی را در عدد دقت نهایی دارد کم است، میزان دقت کل قابل قبول باشد در چنین مواردی می توان از معیارهای دیگر مانند ماتریس آشفتگی و f1-score و precision به تفکیک کلاس های مختلف، برای مقایسه استفاده نمود .

سوال 2:

پیش‌پردازش‌های انجام شده در این سوال شامل: ۱- lower case کردن حروف ۲- حذف stopwordها ۳- حذف علائم نگارشی ۴- حذف لینک‌ها، منشن‌ها، هشتگ‌ها و ایموجی هاست.

به طور کلی، lower case کردن حروف در تسک‌های sentiment پیشنهاد نمی‌شود. چرا که بار معنایی کلمه‌ی good و GOOD بسیار با یکدیگر متفاوت است. اما در مواردی مثل این داده که حجم دادگان ما کم است، نگه‌داشتن آن‌ها اصلا کمکی نمی‌کند. در نتیجه lower case کردن در این مورد پیشنهاد می‌شود.

حذف stopwordها از این جهت کمک می‌کند که این کلمات در جملات مثبت و منفی به یک اندازه کاربرد دارند. در نتیجه با حذف آن‌ها کاری می‌کنیم که تنها ازکلماتی که از نظر بار معنایی می‌توانند ما را به فهم sentiment نزدیک کنند، استفاده کنیم. علائم نگارشی هم که کلمه نیستند و بار معنایی‌ای نمی‌توانند منتقل کنند -البته در اکثر موارد- در نتیجه آن‌ها را حذف می‌کنیم.

لینک‌ها و منشن‌ها از آنجایی که تقریبا معنایی منتقل نمی‌کنند؛ قابل حذف هستند. اما ایموجی‌ها در این سوال با تبدیل به کلمات، نتوانستند کمکی به دقت بکنند. هشتگ‌ها هم از آنجایی که یک هشتگ ممکن است هم توسط موافق و هم توسط مخالف استفاده شود؛ گمراه‌کننده خواهند بود.

سوال 3:

در این قسمت از BERT برای تعبیه کلمات مربوطه استفاده شده است. در واقع یک لایه‌ی bert قبل از لایه‌های lstm قرار گرفته است و ورودی (که با bert tokenizer) تبدیل به ایندکس‌های شناخته‌شده توسط bert شده است، به lstm داده می‌شود. همچنین باید توکن‌هایی برای مشخص کردن جملات، قبل از استفاده از bert tokenizer به کلمات اضافه شود.

استفاده از bert می‌تواند تعداد ایپاک‌های لازم برای آموزش را تا حد بسیار زیادی کاهش دهد.

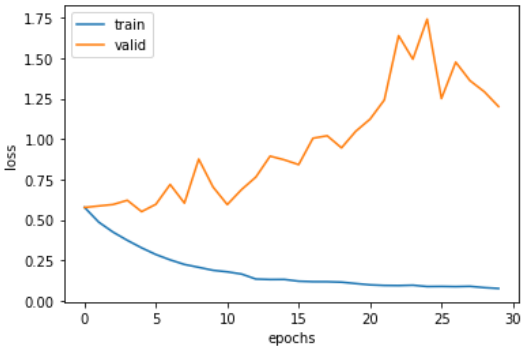
سوال 4:

برای طراحی شبکه ابتدا از یک لایه‌ی bert استفاده کردیم. بعد از آن یک lstm با دو لایه قرار دادیم که اندازه‌ی hidden آن را ۱۰۰ قرار دادیم. Bert به علت کیفیت بالایی که در embedding کلمات دارد استفاده شده. شبکه‌ی lstmای با این تنظیمات هم با کمی سعی و خطا و تجربه‌ی تسک‌های قبلی بدست آمده.

همچنین برای آموزش شبکه از تابع هزینه‌ی binary cross entropy استفاده شده که برای طبقه‌بندی‌های دو کلاسه بسیار مناسب است و سعی در جداسازی دو کلاس دارد.

سوال 5:

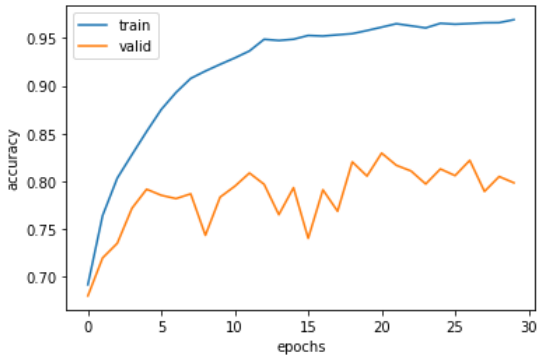
نمودار خطا برای دادگان آموزش و ارزیابی در شکل زیر مشاهده می شود.



شکل ۲۷- نمودار تغییرات خطا در فرآیند آموزش برای داده‌های آموزش و ارزیابی

همانطور که در نمودار بالا مشاهده می شود مقدار خطا برای داده های آموزش کاهش می یابد. اما مشاهده می شود که برای داده های ارزیابی از epoch = 5 به بعد مقدار خطا افزایش می یابد. دلیل اصلی از جایی ناشی می شود که در هنگامی که از BERT به منظور تعبیه کلمات استفاده می شود به طور معمول از 4 و یا

حداکثر epoch 5 به منظور آموزش استفاده می شود چرا که در تعبیه های اولیه این تابع قوی عمل کرده و نسبتا کافی میباشد. البته بررسی خطا به تنهایی کافی نیست.



شکل ۲۸- نمودار تغییرات دقت در فرآیند آموزش برای داده‌های آموزش و ارزیابی

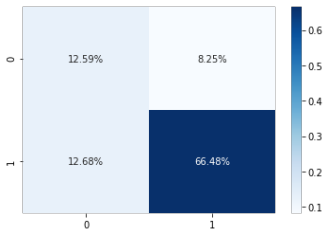
نمودار دقت نیز بعد از مراحل گفته شده در بالا رشد زیادی ندارد.

سوال 6 :

نتایج حاصل از طبقه‌بندی بر روی داده های تست به شرح زیر است:

	precision	recall	f1-score	support
0	0.50	0.66	0.57	447
1	0.90	0.82	0.86	1698
accuracy			0.79	2145
macro avg	0.70	0.74	0.71	2145
weighted avg	0.82	0.79	0.80	2145

و ماتریس آشفتگی دادگان تست هم در زیر آمده است.



شکل ۲۹- ماتریس آشفتگی داده‌های تست

در کل عملکرد مدل بر روی داده‌های Negative (کلاس ۱) که در ابتدا تعداد نمونه‌های بسیار بیشتری هم داشتند، بهتر است و هم معیار precision و هم recall در آن با اختلاف زیادی بهتر است.