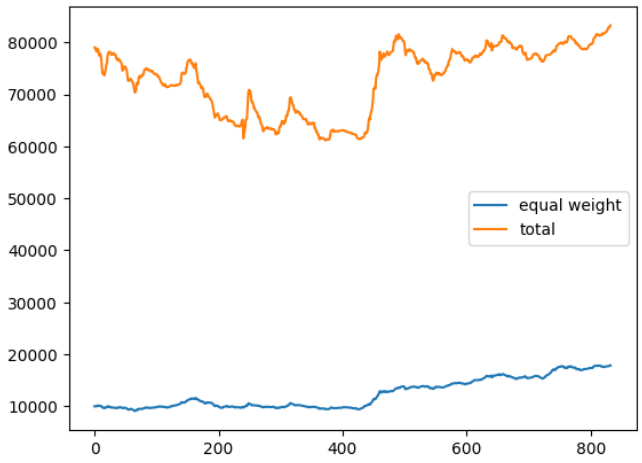
گزارش پیاده‌سازی و پاسخ به سوالات

بخش اول:

پس از دریافت داده‌های بورس از کتابخانه pytse ابتدا دو شاخص «شاخص کل» و «شاخص هم وزن» را جدا می‌کنیم. با بررسی داده‌ها مشخص شد که این دو داده مربوط به بازه‌های متفاوتی هستند که نیاز است تاریخ‌های مشترک بین آن‌ها جدا شود و از اطلاعات آن تاریخ‌های مشترک استفاده شود. پس از این کار به ۸۳۳ روز مشترک می‌رسیم که داده‌های مربوط به این دو شاخص را برای این روزها جدا می‌کنیم. همچنین برای روزهایی که شاخص کل در روز بعدی افزایش یافته برچسب یک و برای کاهش آن برچسب صفر در نظر گرفته شده است. بخشی از خروجی این قسمت به صورت زیر است:

	total	equal weight	date	label
0	79015.4	10000.0	2014-03-19	1
1	79013.5	10033.6	2014-03-25	0
2	78619.4	9998.3	2014-03-26	0
3	78239.7	9984.4	2014-03-29	0
4	78469.2	10055.5	2014-03-30	1

تغییرات این دو شاخص در این داده‌ها در نمودار زیر دیده می‌شود:



الف) در تحلیل سری‌های زمانی نیاز است که دنباله شامل تمام داده‌ها را با استفاده از پنجره‌گذاری بر روی آن‌ها و لغزاندن پنجره به دنباله‌های کوچک‌تری تبدیل کنیم که بتوان به ازای هر N نمونه در یک پنجره در خصوص

نمونه بعدی پیش‌بینی انجام داد. این عمل به دلیل آن است که داده‌های با فاصله زیاد در پیش‌بینی تاثیرگذاری کمتری دارند و درصورت استفاده از آن‌ها پارامترهای مدل بیشتر می‌شودکه ضروری نیست.

این کار را بر روی داده‌های فراهم شده انجام می‌دهیم. به این صورت که مقدار دو شاخص را برای N روز گذشته را به عنوان داده ورودی مدل جدا می‌کنیم و برچسب مربوط به داده N ام که کاهش یا افزایش شاخص کل در روز آینده است را به عنوان برچسب آن داده‌ها در نظر می‌گیریم و این فرایند را برای همه داده‌ها انجام می‌دهیم.

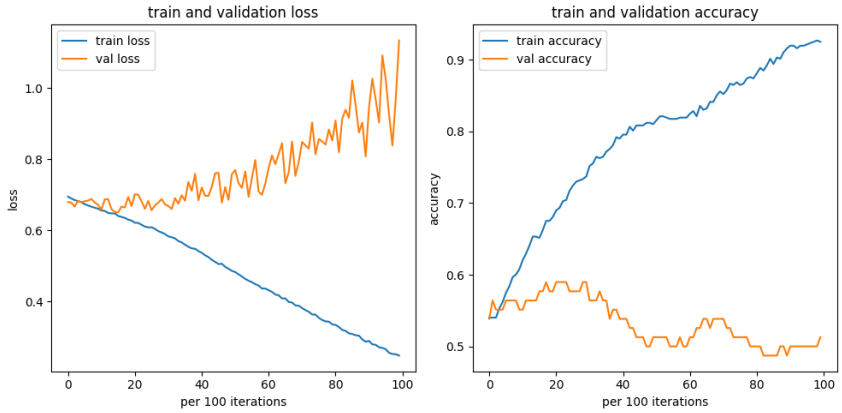
با سعی و خطا مقدار مناسب برای N را برابر ۵۰ به دست آورده‌ایم که در آموزش مدل استفاده خواهد شد.

با توجه به آن که داده‌های ورودی در بازه محدودی قرار دارند و هر مقداری در این بازه برای آن‌ها ممکن است می‌توان از این داده‌ها مستقیما به عنوان ورودی شبکه RNN استفاده کرد. ولی با بررسی که انجام شد انجام یک تغییر باعث بهبود نتایج شده است. این تغییر به این صورت است که داده‌های موجود در هر پنجره را نرمالایز می‌کنیم و به بازه ۰ تا۱۰۰ می‌بریم و به تایپ int تبدیل می‌کنیم.

همچنین ۷۰ درصد از ابتدای داده‌ها را به عنوان داده مربوط به دو شاخص را برای داده آموزشی و ۱۰ درصد برای داده ارزیابی و ۲۰ درصد انتهایی را برای داده آزمون جدا کرده‌ایم.

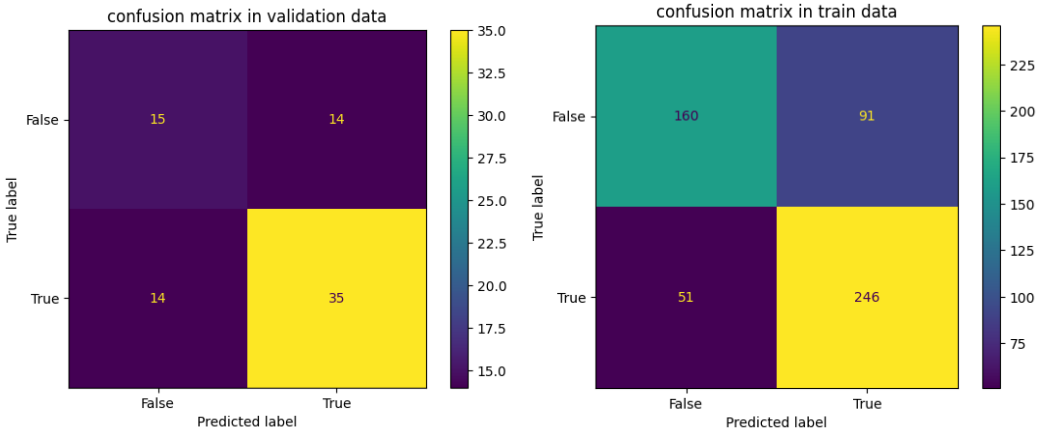
ب) مدلی بر پایه شبکه بازگشتی برای آموزش بر روی داده‌های فراهم شده پیاده‌سازی شده است. در این مدل با استفاده از چارچوب pytorch ابتدا داده‌های هر یک از دو شاخص در ۶۴ بعد تعبیه می‌شود و سپس از یک Batch Normalization عبور داده می‌شود. در ادامه برای هر یک از دو شاخص یک لایه RNN قرار داده شده است و hidden state خروجی این این دو لایه را با هم concat می‌کنیم و به یک لایه خطی Feedforward می‌دهیم که یک نورون در لایه خروجی دارد. تابع خطای استفاده شده cross entropy است و برای تابع بهینه‌ساز از Adam با نرخ یادگیری 5e-5 استفاده شده است.

پس آموزش مدل به تعداد ۱۰۰ ایپاک خروجی تغییرات loss و accuracy برای داده‌های آموزشی و ارزیابی به صورت زیر است:



همان‌طور که دیده می‌شود دقت در داده آموزشی همواره بیشتر می‌شود ولی در داده ارزیابی پس از ۲۰ ایپاک دقت کاهش یافته است و مدل overfit شده است.

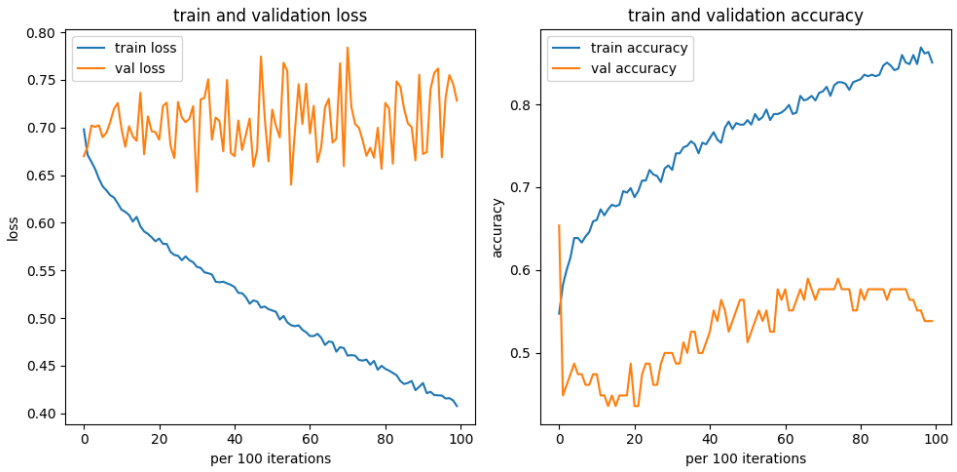
پس از آموزش مدل تا ۲۰ ایپاک استفاده می‌کنیم که در در نهایت به accuracy برابر ۰.۷۴ در داده آموزشی و ۰.۶۴ در داده ارزیابی می‌رسیم و ماتریس درهم‌ریختگی برای داده آموزشی و ارزیابی به صورت زیر است:



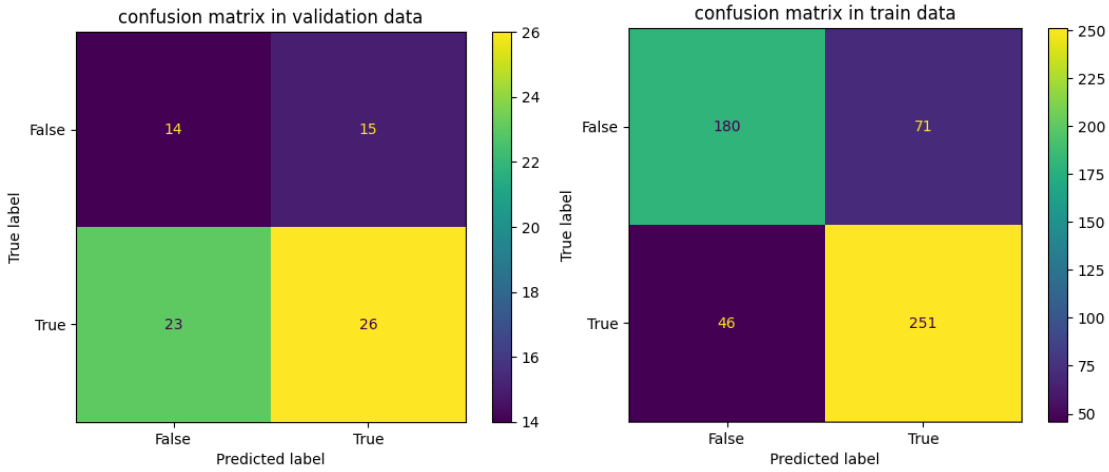
سپس مدل را بر روی داده‌های آزمون بررسی می‌کنیم که به مقدار accuracy برابر ۰.۶۲ رسیده‌ایم. ماتریس در هم‌ریختگی نتیجه پیش‌بینی به صورت زیر است:

ج) مدلی با استفاده از لایه‌های کانولوشنی نوشته شده است که در آن ابتدا دنباله داده‌های دو شاخص در کنار هم قرار گرفته و سپس بر روی آن یک کانولوشن با کرنل ۲ در ۵ زده شده است. سپس از Batch Normalization و Max pooling استفاده شده است. این فرایند سه بار تکرار شده که به ترتیب تعداد کانال‌ها از ۱ کانال اولیه به ۸، ۶۴ و سپس ۱۲۸ رسیده است. در انتها از یک لایه Feedforward استفاده شده است که یک نورون خروجی دارد. تابع خطا و بهینه‌سازی مدل مانند قسمت قبل است.

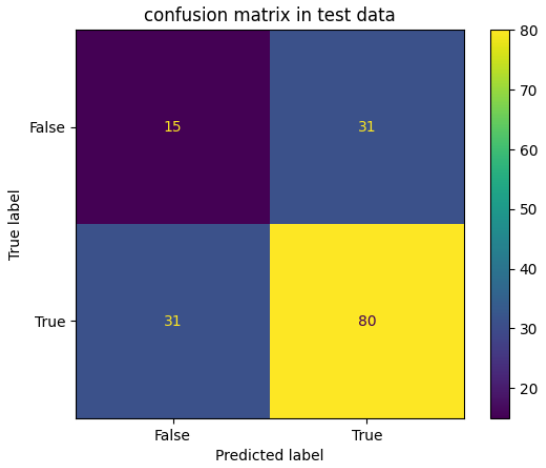
پس از آموزش مدل تا ۱۰۰ ایپاک بر روی داده‌های آموزشی تغییرات خطا و accuracy برای داده‌های آموزشی و ارزیابی به صورت زیر است:



دیده می‌شود که در ابتدا مقدار accuracy در داده‌های آموزشی و ارزیابی افزایش داشته ولی پس از مدتی دقت داده‌های ارزیابی کم شده و دقت داده‌های آموزشی همچنان رشد داشته است و مدل overfit شده است. بر این اساس مدل را تنها تا ۴۰ ایپاک آموزش می‌دهیم که در نتیجه آن به accuracy برابر ۰.۷۸ در داده‌های آموزشی و ۰.۵۱ در داده‌های ارزیابی رسیده‌ایم. ماتریس درهم‌ریختگی نتایج به صورت زیر است:



سپس مدل را بر روی داده‌های آزمون بررسی می‌کنیم که به accuracy برابر ۰.۶۰ رسیده‌ایم که ماتریس درهم‌ریختگی نهایی برای داده آزمون به صورت زیر است:



مقایسه دو مدل: در جدول زیر مقدار دقت و خطا را برای داده‌های آموزشی، ارزیابی و آزمون در دو شبکه گفته شده می‌بینیم:

	داده آموزشی		داده ارزیابی		داده آزمون	
	loss	accuracy	loss	accuracy	loss	accuracy
RNN	۰.۵۹	۰.۷۴	۰.۶۶	۰.۶۴	۰.۶۶	۰.۶۲
CNN	۰.۵۰	۰.۷۸	۰.۶۶	۰.۵۱	۰.۶۵	۰.۶۰

با مقایسه دو مدل مبتنی بر شبکه بازگشتی و شبکه کانولوشنی دیده می‌شود که استفاده از شبکه بازگشتی توانسته به accuracy بهتری برسد و با دقت بهتری کاهش یا افزایش نمونه بعدی را پیش‌بینی کند. دلیل این امر آن

است که شبکه بازگشتی می‌تواند توالی تغییرات نمونه‌ها در زمان را در نظر بگیرد در حالی که شبکه کانولوشنی تنها ویژگی‌های موجود در نمونه‌ها را به دست می‌آورد و مستقیما به تغییرات این نمونه‌ها توجهی ندارد که این تغییرات می‌تواند در سیر تغییر تا نمونه بعدی بسیار مهم باشد.

بخش دوم)

د) ناهنجاری در داده‌های سری زمانی به دلیل عدم شباهت و تغییرات فراوان یک یا تعدادی از داده‌ها نسبت به داده‌های اطرافشان است که به دلیل این تغییرات ناگهانی یا متفاوت، با استفاده از داده‌های اطرافشان قابل بازسازی و پیش‌بینی نیستند.

برای تشخیص ناهنجاری در سری زمانی با استفاده از شبکه خودکدگذار باید مراحل زیر را طی کنیم:

- ابتدا داده‌های سری زمانی را نرمالایز می‌کنیم که البته این کار در بهبود دقت مدل تاثیر گذاری است.
- با استفاده از data windowing بازه‌های N تایی از داده‌ها را جدا می‌کنیم و هر بار پنجره را یک واحد می‌لغزانیم و ماتریسی ساخته می‌شود که هر سطر آن شامل N داده متوالی از سری زمانی است.
- مدل خودکدگذاری را پیاده‌سازی می‌کنیم. در این مدل می‌توان از لایه‌های کانولوشنی و بازگشتی استفاده کرد و تابع خطا هم تابع MAE می‌باشد.
- مدل را بر روی داده‌های آموزشی، آموزش می‌دهیم.
- داده‌های تست را مانند داده‌های آموزشی نرمالایز کرده و data windowing انجام می‌دهیم.
- هر نمونه از داده‌های تست فراهم شده را به مدل می‌دهیم و خروجی مدل خودکدگذار که بازسازی ورودی است را به دست آورده و میانگین اندازه‌ی فاصله نمونه‌های واقعی از مقادیر بازسازی شده را محاسبه می‌کنیم.
- هیستوگرام خطاهای به دست‌آمده را رسم می‌کنیم که مشخص می‌کند فراوانی چه خطاهایی بیشتر بوده است.
- از روی هیستوگرام رسم شده می‌توان متوجه شد که بازسازی برخی از داده‌ها دارای خطای بسیار زیادی است که در واقع همان داده‌های پرت هستند و البته فراوانی کمی هم دارند. داده‌هایی که این خطاها را دارند را به عنوان داده پرت تشخیص داده و از داده‌های اصلی حذف می‌کنیم.

ه) در مقاله مورد نظر سعی شده است از شبکه خودکدگذار برای یک روش یادگیری خودنظارت در حوزه تصویر استفاده شود. روش اصلی این کار در این مقاله به این صورت است که در ورودی مدل خودکدگذار تصویری داده می‌شود که تعدادی از patch های تصویر به صورت تصادفی پوشیده شده است و در خروجی، تصویر اصلی باید بازسازی شود. آموزش مدل به این روش باعث می‌شود که مدل توانایی بازسازی قسمت‌های پوشیده شده از تصویر را داشته باشد. داده لازم در آموزش این مدل با استفاده از مجموعه‌ای از تصاویر بدون هیچ برچسبی فراهم می‌شود و patchهایی از داده‌ها به صورت تصادفی پوشیده می‌شود تا داده ورودی مدل فراهم شود و خود داده هم خروجی مدل است که در نتیجه از خود داده‌ها، ورودی و خروجی مورد نیاز ساخته می‌شود و یک مدل با نظارت می‌تواند از آن استفاده کند که به همین دلیل این روش یک روش خودنظارت نام دارد.