به نام خدا

دانشگاه تهران

دانشکده مهندسی برق و کامپیوتر

درس شبکه‌های عصبی و یادگیری عمیق

تمرین اول

پرسش ۱ و ۳	نام و نام خانوادگی	بابک حسینی محتشم
	شماره دانشجویی	۸۱۰۱۰۱۴۰۸
پرسش ۲ و ۴	نام و نام خانوادگی	
	شماره دانشجویی	
	مهلت ارسال پاسخ	۱۴۰۳.۱۲.۲۷

فهرست

شکل‌ها

جدول‌ها

پرسش ۱. تشخیص تقلب در کارت‌های اعتباری با استفاده از شبکه‌های عصبی چندلایه (MLP)

۱-۱. پیش‌پردازش و بررسی دادگان

۱) با مطالعه توضیحات مربوط به داده در سایت کگل، اطلاعتی از آن به دست آوردیم. یکی از کاربردهای این دادگان کمک به شرکت‌ها برای تشخیص کلاهبرداری است به طوری که خریداران برای محصولی که خرید نکرده‌اند مبلغی نپردازند.

داده ذکر شده در سال ۲۰۱۳ توسط کارت دارندگان اروپایی ایجاد شده است. تراکنش‌های دادگان در دو روز رخ داده و ۴۹۲ تراکنش از ۲۸۴۸۰۷ تراکنش موجود کلاهبرداری اعلام شده است. در نتیجه توازن بین برچسب‌ها به خوبی رعایت نشده و داده‌های با برچسب کلاهبرداری ۰.۱۷۲ درصد دادگان را تشکیل می‌دهند.

در این دادگان ۳۱ ویژگی قرار دارد که ۲۸ تای آن‌ها حاصل اعمال PCA روی ویژگی‌های هستند که برای جلوگیری از افشاء هویت در دادگان قرار داده نشده‌اند. دو ویژگی دیگر تبدیل یافته توسط PCA نیستند. یکی از آن‌ها ویژگی زمان است که نشان دهنده مدت زمان گذشته به ثانیه بین هر تراکنش با اولین تراکنش موجود در این دادگان است. ویژگی دیگر نشان دهنده میزان مبلغ تراکنش است و ویژگی آخر هم همان برچسب دادگان است که یک ویژگی دودویی است که در صورت یک بودن یعنی کلاهبرداری صورت گرفته است وگرنه تراکنش بدون کلاهبرداری است.

با توجه به ناتوازنی برچسب، پیشنهاد شده است که از معیار AUPRC برای اندازه‌گیری دقت استفاده کنیم چرا که استفاده از ماتریس درهم‌ریختگی برای دادگان با برچسب‌های نامتوازن پیشنهاد نمی‌شود چرا که ممکن است حتی در صورت خطا در برچسب‌گذاری به دلیل کمبود داده از برچسبی که به اشتباه طبقه‌بندی شده، باز هم مدل دقت بالایی بگیرد.

۲) حال ابتدا دادگان را به صورت دیتافریم پانداز وارد می‌کنیم و نمونه‌ای از داده‌گان را مشاهده می‌کنیم.

جدول ۱. نمونه‌ای از دادگان

	V1	Time	Amount	Class
43428	-16.526507	41505.0	364.19	1
49906	0.339812	44261.0	520.12	0

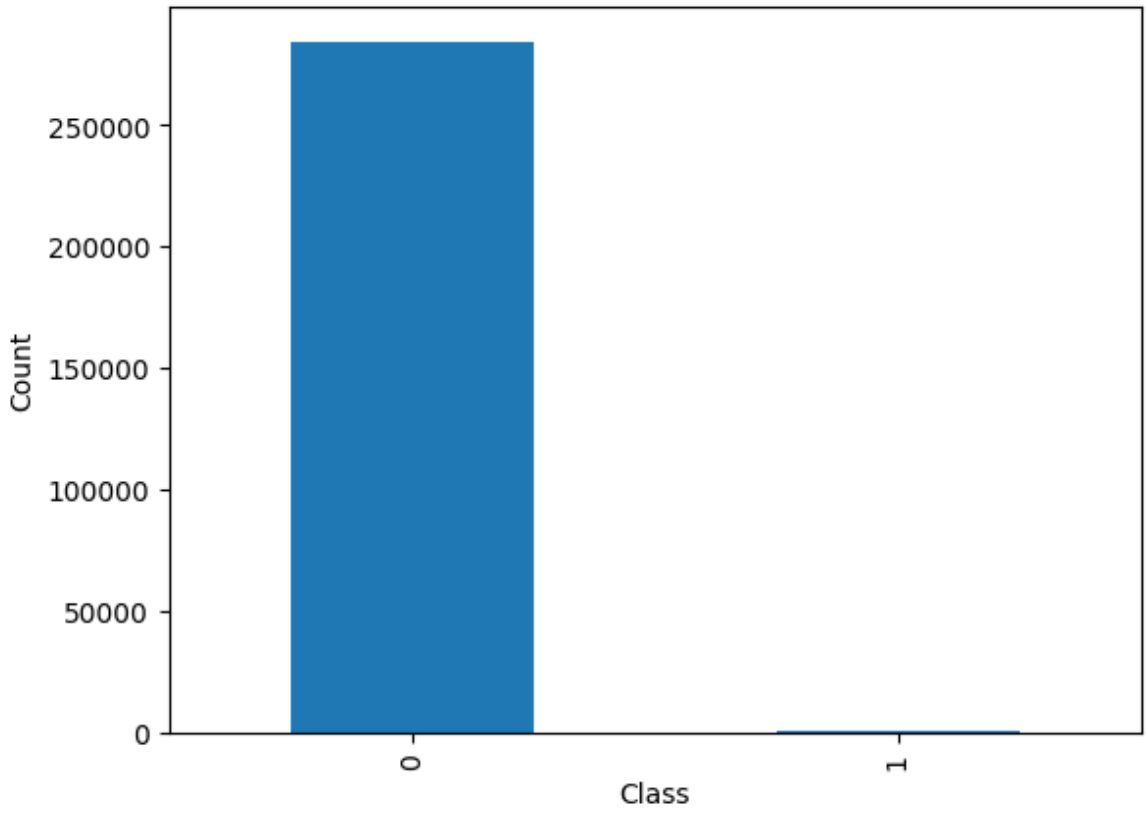
با بررسی اطلاعات جدول متوجه شدیم داده null وجود ندارد و تمامی ویژگی‌ها از جنس float هستند به جز برچسب که از جنس int است. همچنین در جدول زیر می‌توان اطلاعتی آماری از ویژگی‌ها مشاهده کرد. می‌توان دید سه چهارم دادگان مقدار تراکنشی کمتر از 77 دارند در حالیکه بشترین مقدار 25691 است که می‌توان نتیجه گرفت اکثر تراکنش‌ها مقدار کمی دارند ولی تعداد کمی از تراکنش‌ها هم با مبالغ بالا وجود دارند. همچنین می‌توان مشاهده کرد که تراکنش‌ها در پراکندگی زمانی نسبتا یکنواختی قرار گرفته‌اند.

جدول ۲. اطلاعات آماری دادگان

	V1	**Time**	**Amount**	**Class**
count	2.848070e+05	284807.000000	284807.000000	284807.000000
mean	1.168375e-15	94813.859575	88.349619	0.001727
std	1.958696e+00	47488.145955	250.120109	0.041527
min	-5.640751e+01	0.000000	0.000000	0.000000
25%	-9.203734e-01	54201.500000	5.600000	0.000000
50%	1.810880e-02	84692.000000	22.000000	0.000000
75%	1.315642e+00	139320.500000	77.165000	0.000000
max	2.454930e+00	172792.000000	25691.160000	1.000000

همچنین 1825 تراکنش مقدار صفر دارند که با بررسی کگل پرسش‌هایی در مورد آن نیز ایجاد شده است. در حالیکه پاسخ قطعی برای آن نتوانستیم پیدا کنیم ولی یک پاسخ احتمالی این است که ممکن است این تراکنش‌ها مربوط به سرویس‌های خاصی باشند که در ابتدا تراکنش صفر صورت می‌گیرد و مبلغ اصلی در آینده از حساب کم شود. قابل توجه است که 27 مورد از این تراکنش‌ها نیز به صورت کلاهبرداری برچسب‌گذاری شده‌اند.

۳) عدم تعادل میان تعداد دادگان ویژگی برچسب در نمودار میله‌ای زیر مشخص است.



شکل ۱. نمودار میله‌ای از توزیع کلاس‌ها

۴) عدم تعادل کلاس‌ها می‌تواند فرایند مدل‌سازی و به خصوص قسمت ارزیابی دقت مدل را چالش برانگیز کند. برای مثال فرض کنید که طبقه‌بند تشخیص ایمیل spam با دادگانی ایجاد می‌کنیم که تعداد ایمیل‌های spam یعنی کلاس 1 بسیار کمتر باشد. در این صورت اگر مدل تمام ایمیل‌ها را با کلاس 0 طبقه‌بندی کند، به دلیل وجود تعداد کمی خطای false negative، دقت مدل بالا حساب می‌شود در صورتی که هیچ یک از ایمیل‌های spam را به درستی تشخیص نداده است. در این موارد استفاده از معیارهای دیگری به جز دقت برای ارزیابی مدل توصیه می‌شود برای مثال می‌توان از معیار بازیابی در مثال ایمیل استفاده کرد تا به مدل را براساس true positive و false negative ارزیابی کند.

۱-۱-۱. مقایسه standardization با normalization

۵) ابتدا بررسی می‌کنیم که برای چه باید ویژگی‌ها را اسکیل کنیم. اسکیل کردن ویژگی‌ها باعث می‌شود ویژگی‌ها در محدوده مشخصی و مشابهی قرار گیرند و به همین دلیل باعث کاهش سوگیری مدل به سمت یک محدوده خاص می‌شود. برای مثال اگر یک ویژگی مقادیر بزرگ و ویژگی دیگری مقادیر کوچکتری داشته باشد، ممکن است مدل اهمیت بیشتری به ویژگی اول بدهد. از مزیت‌های اسکیل کردن می‌توان

گفت: می‌تواند باعث افزایش دقت مدل شود، می‌تواند باعث همگرایی سریع‌تر شود، تاثیر داده‌های پرت شود و هزینه محاسبات هم می‌تواند کاهش یابد. البته برای برخی مدل‌ها مانند decision tree و random forest تاثیری ندارد.

دو روش متداول برای اسکیل کردن عبارتند از normalization و standardization. در روش اول، محدوده مقادیر ویژگی‌ها به بازه مشخصی که معمولا بین 0 تا 1 است تبدیل می‌شود. در حالی که در روش دوم، پس از تبدیل، ویژگی‌ها دارای میانگین 0 و انحراف معیار 1 می‌شوند. یکی از مزیت‌های روش دوم بر روش اول این است که شکل توزیع داده‌ها را تغییر نمی‌دهد و تنها آن را اسکیل می‌کند. همچنین روش دوم برخلاف روش اول، نسبت به داده‌های پرت حساسیت کمتری دارد. معمولا روش اول برای مدل‌های آماری که فرض نرمال بودن داده‌ها را دارند روش بهتری است درحالیکه روش دوم برای مدل‌هایی که به اسکیل داده‌ها وابسته‌اند ولی وابستگی زیادی به توزیع داده‌ها ندارند مانند شبکه‌های عصبی می‌تواند بهتر باشد.

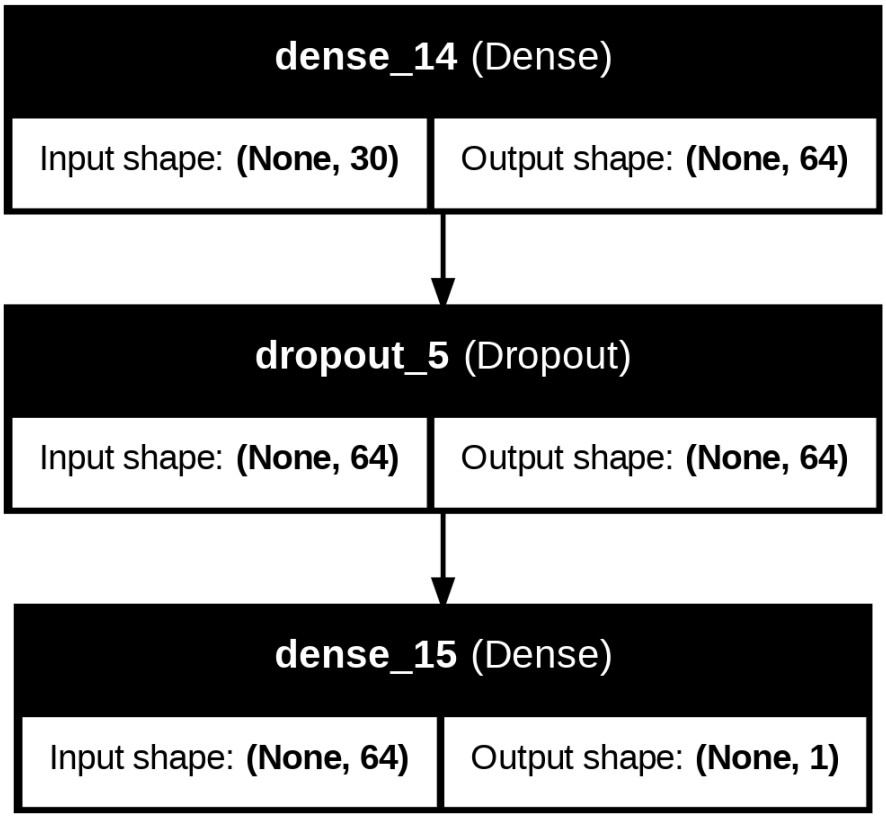
در نتیجه استفاده از هر کدام از دو روش یا ترکیبی از این دو بسته به نوع داده و مدل می‌تواند بهتر باشد، ما هر دو روش را تست خواهیم کرد ولی مبنای اصلی برای بهبود مدل را به دلیل استفاده از شبکه‌های عصبی روش normalization قرار می‌دهیم. برای مقدار Amount که داده‌های پرت دارد standard کردن ممکن است بهتر باشد درحالیکه برای Time که توزیع نسبتا یکنواختی دارد به نظر استفاده از normalization می‌تواند کمک کند.

۶) برای جلوگیری از بروز اطلاعات داده‌های ارزیابی به داده‌های آموزش، ابتدا داده‌ها را تقسیم و سپس اسکیل می‌کنیم.

۱-۲. طراحی و پیاده سازی یک شبکه MLP ساده

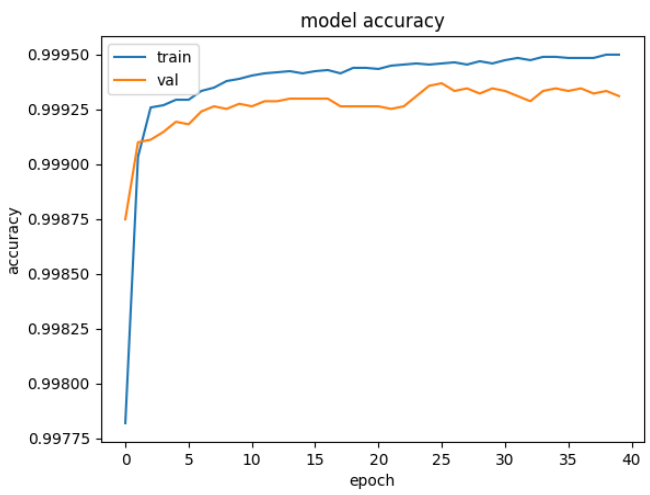
ما چهار مدل را طراحی و آموزش دادیم و نتیاج‌شان را بررسی کردیم.

مدل اول لایه Dropout ندارد و منظم‌سازی هم نمی‌کند. مدل دوم لایه Dropout دارد ولی منظم‌سازی نمی‌کند. مدل سوم لایه Dropout ندارد ولی منظم‌سازی می‌کند و نهایتا مدل آخر هم لایه Dropout دارد و هم منظم‌سازی می‌کند.

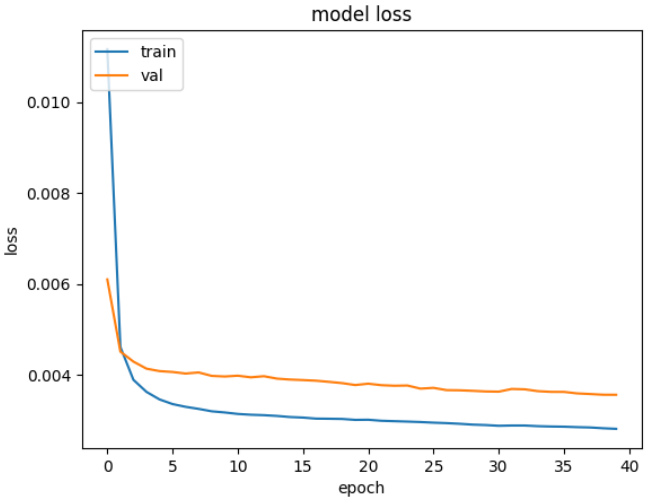


شکل ۲. معماری شبکه عصبی مدل چهارم

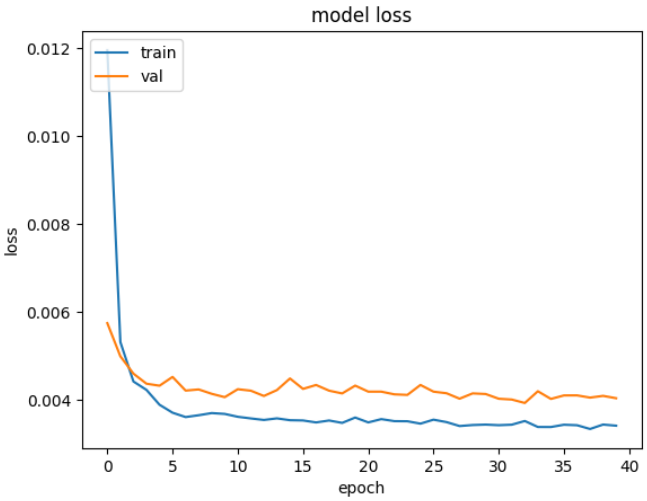
نمودار خطا و صحت مدل‌ها حین آموزش روی داده آموزش و تست به صورت زیر است.



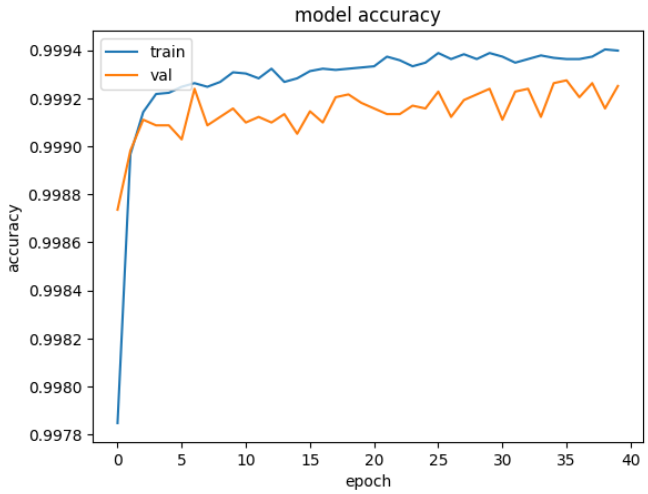
شکل ۴. صحت مدل اول حین آموزش



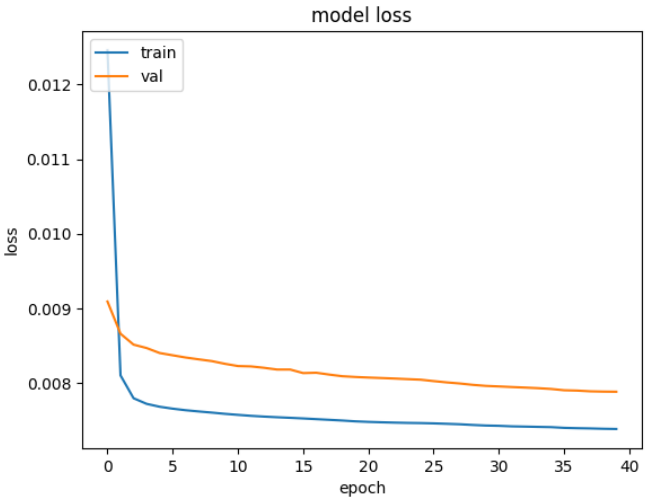
شکل ۳. خطای مدل اول حین آموزش



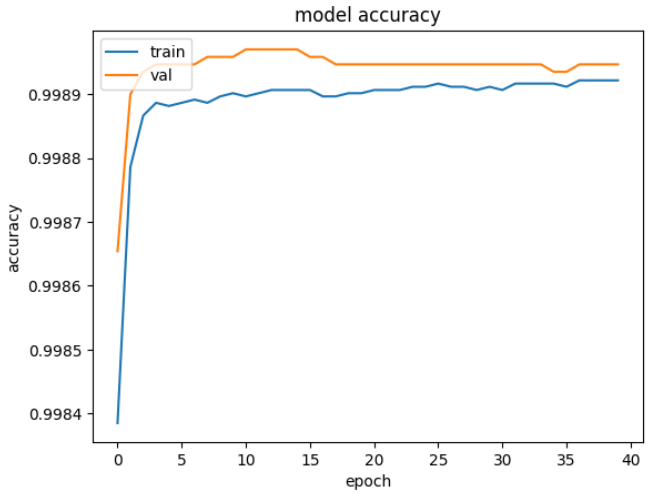
شکل ۵. خطای مدل دوم حین آموزش



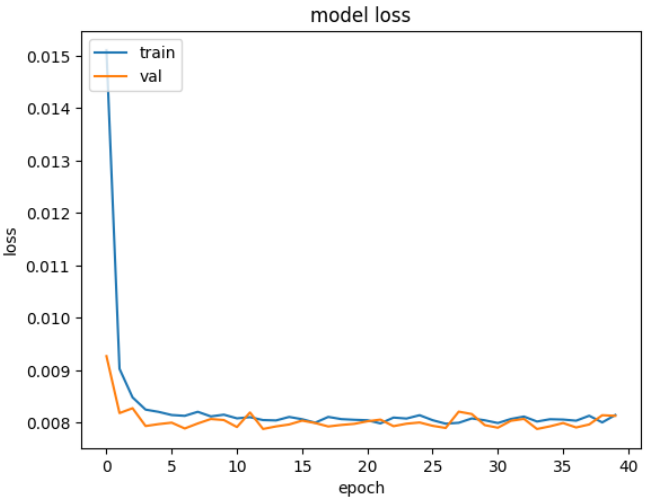
شکل ۶. صحت مدل دوم حین آموزش



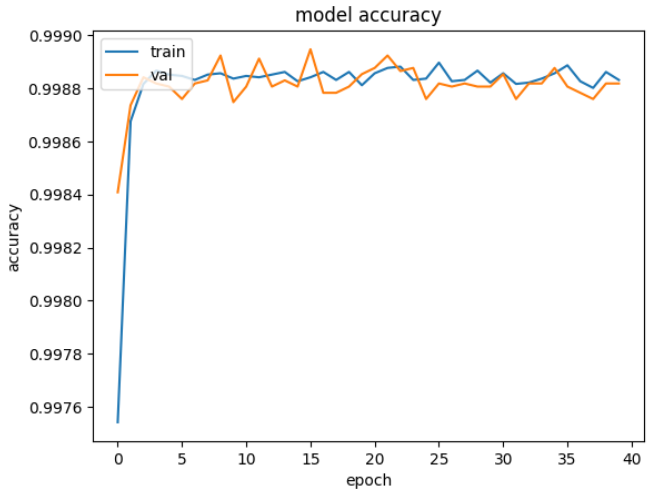
شکل ۷. خطای مدل سوم حین آموزش



شکل ۸. صحت مدل سوم حین آموزش



شکل ۹. خطای مدل چهارم حین آموزش



شکل ۱۰. صحت مدل چهارم حین آموزش

با بررسی نمودارهای بالا می‌توان نکاتی را مشاهده کرد. به نظر می‌رسد در تمام مدل‌ها پس از ده epoch، مقدار صحت مدل افزایش زیادی پیدا نمی‌کند به خصوص در مدل آخر که می‌تواند نشان دهنده سرعت بالای همگرایی مدل باشد. همچنین می‌توان دید در مدل اول فاصله بین صحت داده آموزش و داده تست بیشتر می‌شود و احتمالا با آموزش بیشتر بیش‌برازش رخ می‌دهد. همچنین در دو مدل آخر که منظم‌سازی صورت گرفته صحت داده آموزش و تست تا epoch آخر یکسان و یا بیشتر است.

حال این مدل‌ها را روی داده تست ارزیابی می‌کنیم.

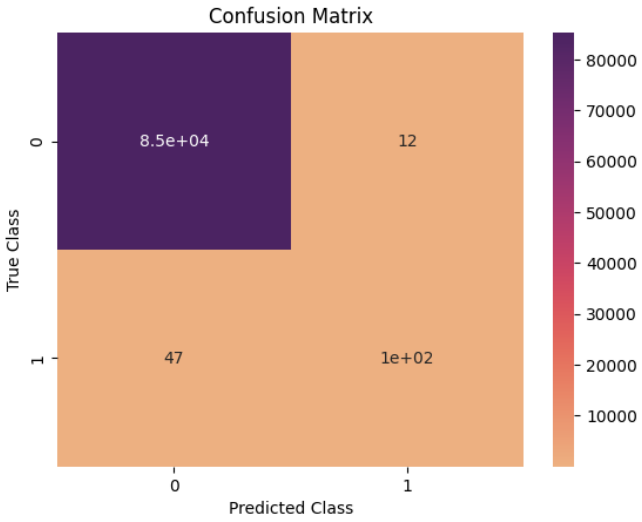
مدل اول:

Accuracy: 0.999

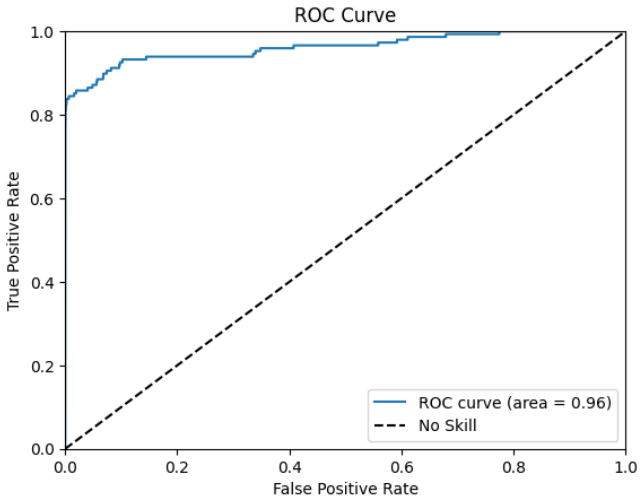
Precision: 0.894

Recall: 0.682

F1 score: 0.774



شکل ۱۱. ماتریس درهم‌ریختگی مدل اول



شکل ۱۲. منحنی ROC مدل اول

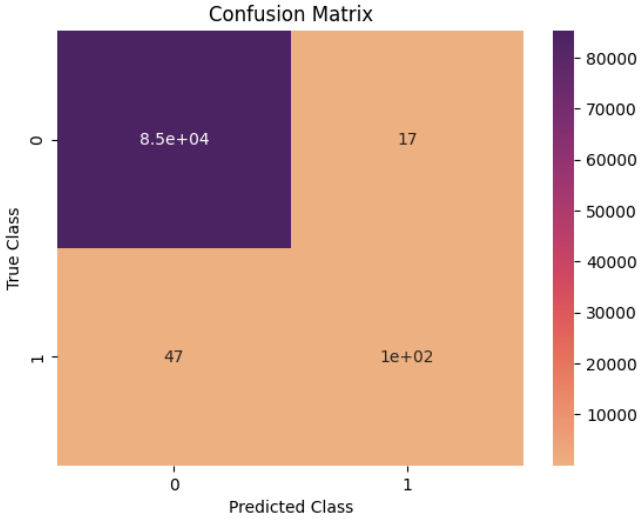
مدل دوم:

Accuracy: 0.999

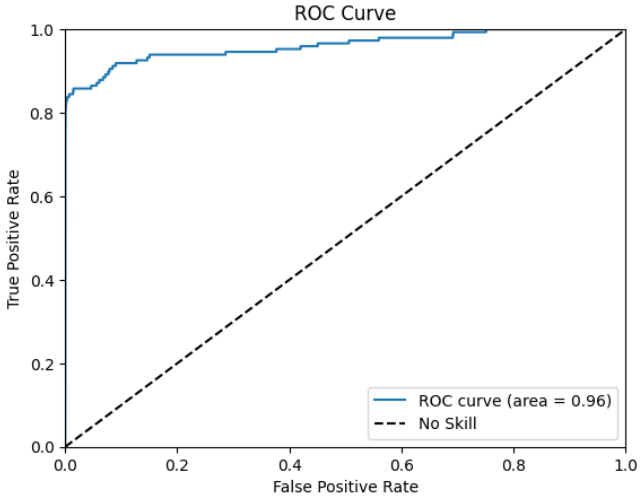
Precision: 0.856

Recall: 0.682

F1 score: 0.759



شکل ۱۳. ماتریس درهم‌ریختگی مدل دوم



شکل ۱۴. منحنی ROC مدل دوم

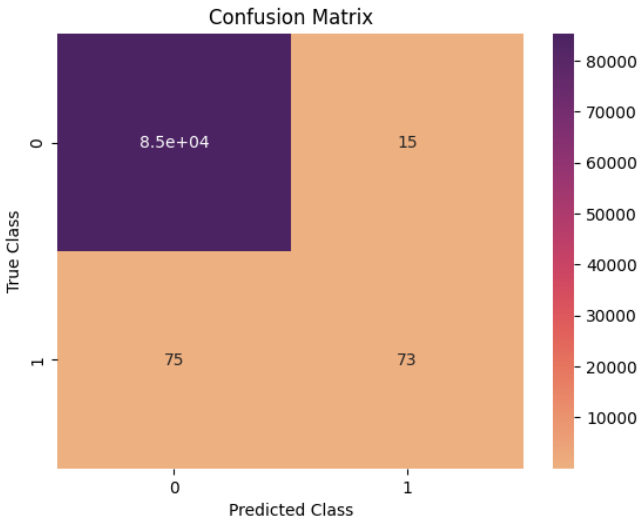
مدل سوم:

Accuracy: 0.999

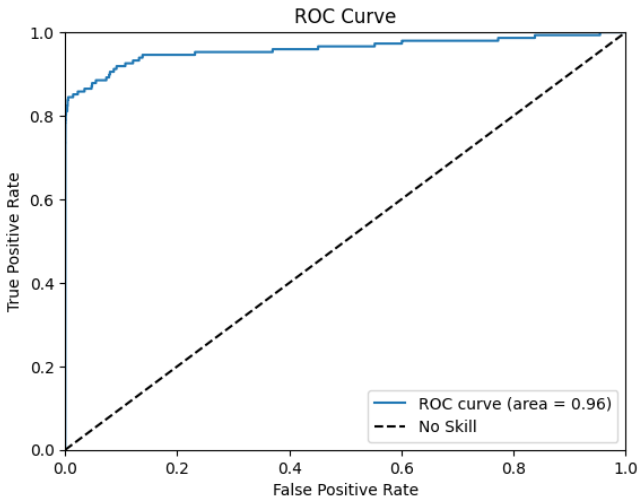
Precision: 0.83

Recall: 0.493

F1 score: 0.619



شکل ۱۵. ماتریس درهم‌ریختگی مدل سوم



شکل ۱۶. منحنی ROC مدل سوم

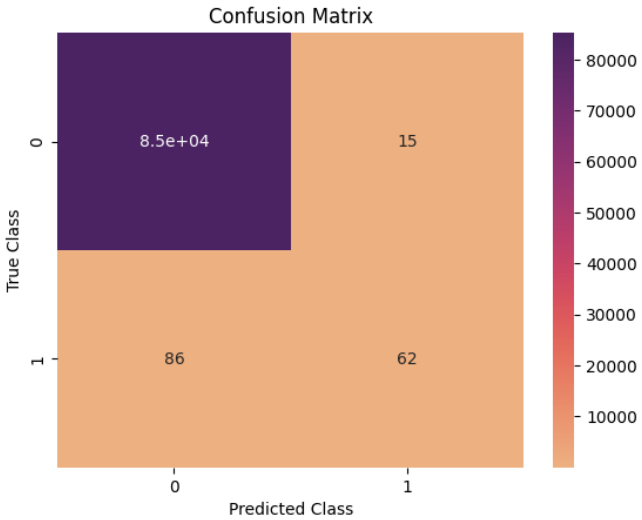
مدل چهارم:

Accuracy: 0.999

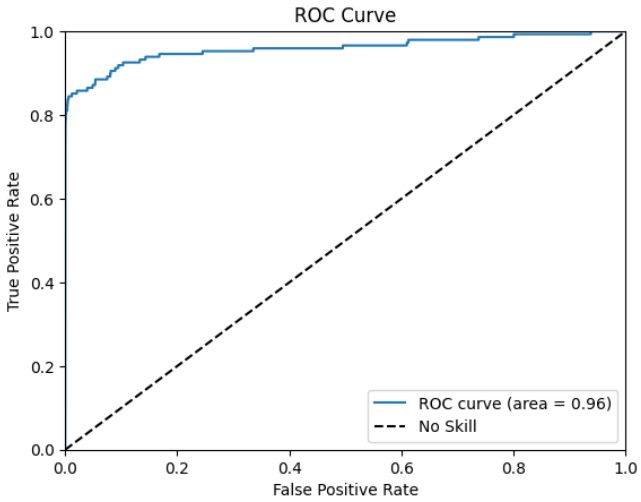
Precision: 0.805

Recall: 0.419

F1 score: 0.551



شکل ۱۷. ماتریس درهم‌ریختگی مدل چهارم

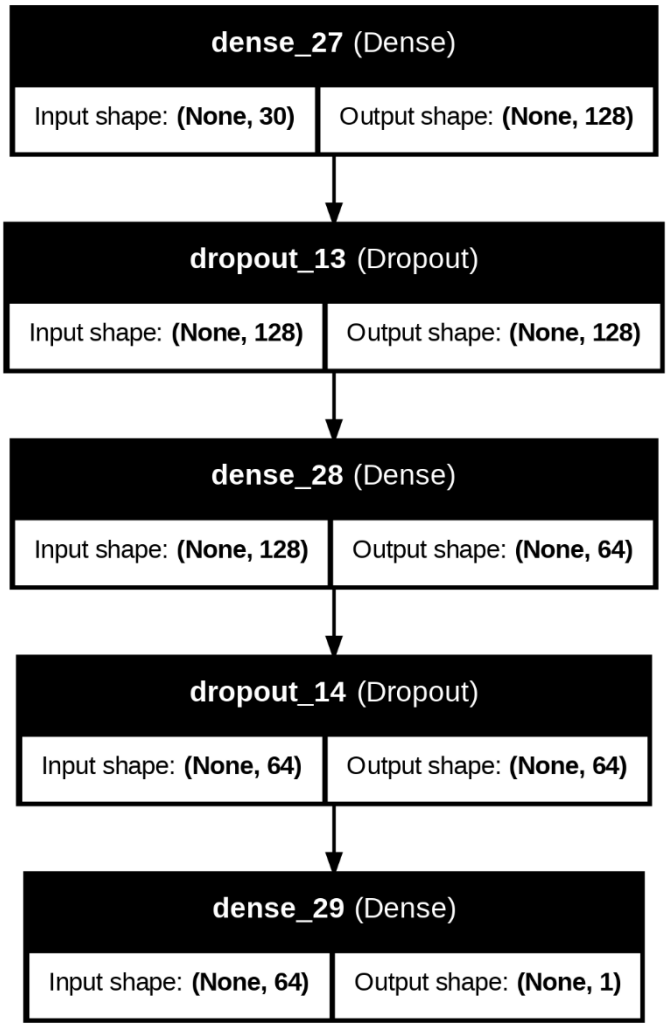


شکل ۱۸. منحنی ROC مدل چهارم

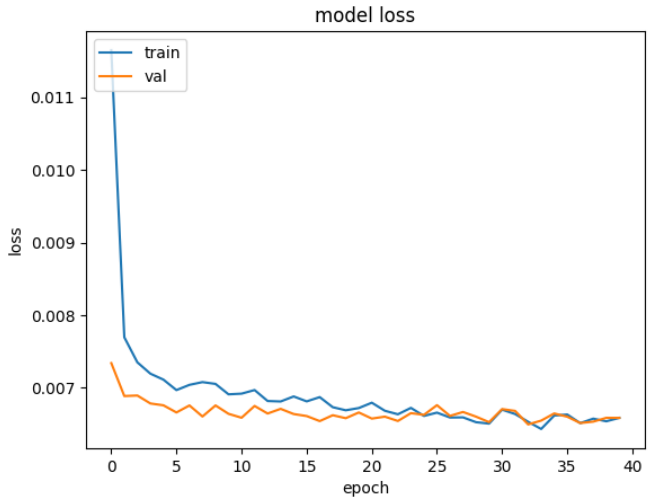
با بررسی نمودارهای بالا می‌توان دید که به ترتیب مدل‌ها خطا بیشتر می‌شود که می‌تواند نشان دهنده این باشد که مدل‌ها را برای داده موجود بیش از حد نیاز پیچیده کردیم و همان مدل اول قدرت کافی طبقه‌بندی را داشت.. همچنین می‌توان دید که مدل‌ها در برچسب‌گذاری کلاس یک که تعداد کمتری داده از آن وجود دارد، خطای بیشتری دارند. به طور کلی به نظر می‌رسد مدل اول که ساده ترین مدل بود بهترین عملکرد را پس از 40 epoch آموزش دارد البته احتمالا با آموزش بیشتر مدل اول زودتر از بقیه مدل‌ها دچار بیش‌برازش بشود.

۱-۴. طراحی یک شبکه عصبی MLP عمیق‌تر

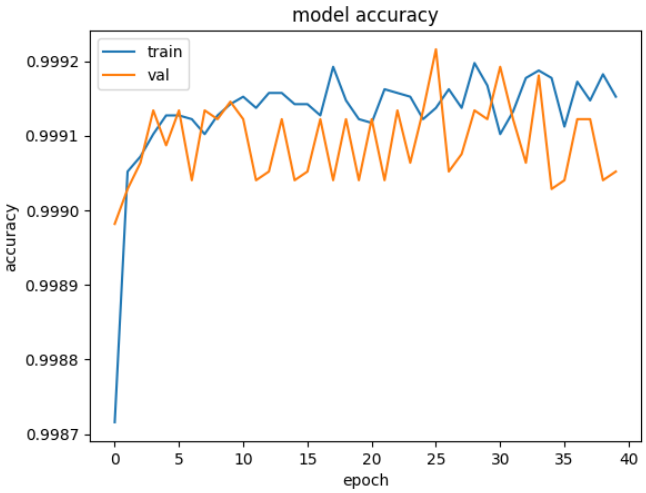
حال مدلی با دولایه پنهان با معماری زیر ایجاد می‌کنیم.



شکل ۱۹. معماری شبکه عصبی با دو لایه پنهان



شکل ۲۰. خطای مدل با دو لایه پنهان حین آموزش



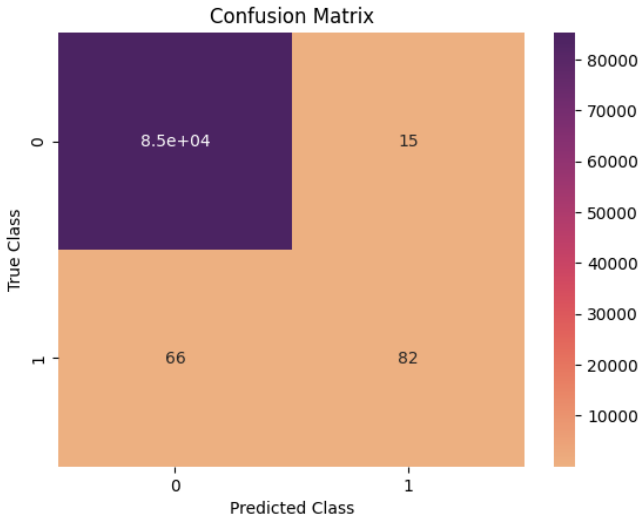
شکل ۲۱. صحت مدل با دو لایه پنهان حین آموزش

Accuracy: 0.999

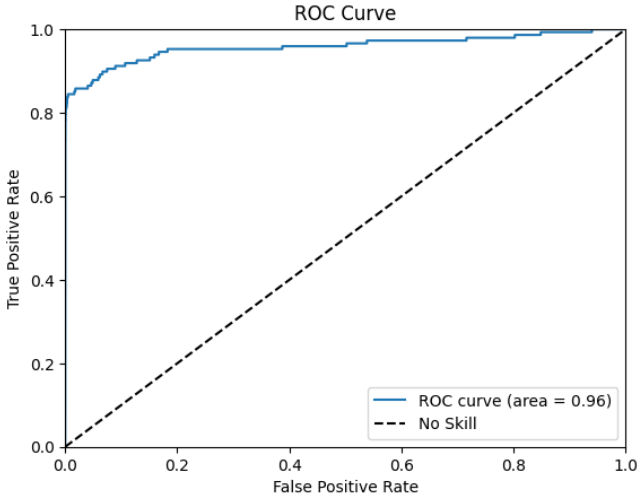
Precision: 0.845

Recall: 0.554

F1 score: 0.669



شکل ۲۲. ماتریس درهم‌ریختگی مدل با دو لایه پنهان



شکل ۲۳. منحنی ROC مدل با دو لایه پنهان

با توجه به نمودارهای بالا به نظر می‌رسد اضافه کردن یک لایه دیگر نیز تاثیر زیادی برای این دادگان ندارد و باعث ایجاد مدلی با عملکردی بین مدل دوم و سوم از بخش قبل می‌شود. درنتیجه ساده‌ترین مدل یعنی مدل اولی که آموزش دادیم بهترین عملکرد را تا به اینجا دارد.

۱-۵. تحلیل ماتریس آشفتگی و معیارهای ارزیابی

صحت یا accuracy از رابطه زیر به دست می‌آید:

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

در داده‌های نامتوازن، حتی اگر TP یا TN صفر باشند، چون مقدار دیگر بسیار زیاد است، دقت نیز مقدار بزرگی می‌شود در صورتی که شاید برای ما یکی از TP یا TN از اهمیت بیشتری برخوردار باشد ولی به دلیل نامتوازن بودن داده‌ها دقت می‌تواند بالاتر از میزان مورد نظر باشد.

معیار دقت یا precision از رابطه زیر به دست می‌آید:

$$precision = \frac{TP}{TP + FP}$$

همان طور که از رابطه بالا هم مشخص است، این معیار با افزایش TP یعنی برچسب‌گذاری کلاس یک به درستی و کاهش FP یعنی برچسب‌گذاری کلاس یک به نادرستی، افزایش میابد. در تشخیص تقلب بسیار مهم است که تا حد امکان کسی که تقلب نکرده است را به اشتباه متقلب تشخیص ندهیم و در عین حال افراد متقلبین را هم تشخیص دهیم. درنتیجه به همین دلیل اهمیت کم بودن FP، استفاده از معیار دقت، در این مورد می‌توان کمک کند.

معیار بازیابی یا recall از رابطه زیر به دست می‌آید:

$$recall = \frac{TP}{TP + FN}$$

از رابطه بالا نتیجه می‌گیریم که مانند دقت، با افزایش TP بازیابی هم افزایش میابد ولی برخلاف دقت که به FP وابسته بود، بازیابی به FN وابسته است و با کاهش آن، بازیابی افزایش میابد. معیار FN نشان دهنده تشخیص کلاس صفر به نادرستی است. اگر بازیابی بالا باشد، یعنی بیشتر مطمئنیم که مدل ما نمونه‌ای که متعلق به کلاس یک است را درست تشخیص می‌دهد و به اشتباه آن را در کلاس صفر قرار نمی‌دهد. برای مثال برای ایجاد مدلی برای تشخیص بیماری کرونا بازیابی معیار مهمی است چرا که تشخیص ندادن یک فرد بیمار می‌تواند هزینه بیشتری داشته باشد تا تشخیص اشتباه یک فرد سالم که در معیار دقت مورد تاکید است.

معیار F1-score هر دو معیار دقت و بازیابی را ترکیب می‌کند البته شاید بتوان گفت معیار صحت هم با در نظر گرفتن هر چهار معیار TP، TN، FP و FN به نوعی مشابه این معیار است. با این حال دلایلی وجود دارد که استفاده از F1-score می‌تواند مفیدتر باشد. معیار F1-score میانگین هارمونیک دو معیار دقت و

بازیابی است و اگر هر یک از این دو معیار پایین باشند، باعث می‌شوند F1-score نیز پایین بیاید. درنتیجه در مواقعی که بالا بودن دقت و بازیابی برای ما اهمیت دارد می‌توانیم از این معیار استفاده کنیم. همچنین به همین دلیل این معیار در مواجهه با کلاس‌های نامتوازن، این معیار می‌تواند بهتر از معیار صحت عمل کند. در کل شاید بتوان گفت اگر اهمیت TP و TN برایمان زیاد و کافی است و یا کلاس‌ها متاوزنند استفاده از صحت توصیه می‌شود وگرنه استفاده از F1-score بهتر است.

منحنی ROC نشان دهنده عملکرد مدل در طبقه‌بندی به ازای آستانه‌های مختلف برای میزان اطمینان مدل از خروجی خود است. برای مثال اگر آستانه را 0.7 در نظر بگیریم، در صورتی که مدل اطمینان بیشتر از 0.7 برای کلاس یک برای یک داده داشته باشد، آن داده را در کلاس یک در نظر گرفته وگرنه در کلاس صفر در نظر می‌گیریم. محور افقی نمودار نشان دهنده FPR(False Positive Rate) و محور عمودی نمودار نشان دهنده TPR(True Positive Rate) است.

رابطه این دو معیار به صورت زیر است:

$$\text{FPR} = \frac{FP}{FP + TN}$$

$$TPR = \frac{TP}{TP + FN} = recall$$

مشخص است که هر چقدر میزان FPR کم و TPR زیاد باشد به مدل مطلوب‌تری می‌رسیم پس با استفاده از این نمودار می‌توانیم آستانه را طوری تعیین کنیم که به مقادیر مورد نظر از دو معیار بالا برسیم. معیار دیگری نیز از این نمودار به نام AUC(Area Under Curve) به دست می‌آید. همان طور که از نام این معیار مشخص است، این معیار برابر مساحت زیر منحنی ROC است. هر چقدر AUC بیشتر باشد، می‌توانیم نتیجه بگیریم میزان FPR بیشتر و TPR کمتر و درنتیجه مدل عملکرد بهتری دارد.

در تمام مدل‌ها کلاس یک، بیشتر اشتباه طبقه‌بندی شده است و کلاس یک کلاسی است که تعداد بسیار کمتری نمونه از آن در داده وجود دارد و همین می‌تواند دلیلی برای خطای بیشتر در طبقه بندی این کلاس باشد.

بین دقت و بازخوانی معمولا تبادل وجود دارد به طوری که با افزایش یکی دیگری کاهش میابد زیرا با افزایش دقت مدل محتاطانه‌تر عمل می‌کند تا به اشتباه به نمونه‌ای برچسب کلاس یک ندهد در صورتی که با افزایش بازخوانی مدل در برچسب کلاس سفر دادن محتاط‌تر می‌شود پس ممکن است تعداد بیشتری از دادگان را به اشتباه کلاس یک برچسب بدهد.

۱-۶. جست و جوی بهترین هایپرپارامترهای شبکه یک لایه مخفی

ما برای جست و جوی بهترین هایپرپارامترهای شبکه، از کتابخانه optuna و با روش random search استفاده کردیم چرا که با توجه به تعداد هایپرپارامترهای مورد جستجو، روش grid search بسیار کند خواهد بود. بهترین مدل با پارامترهای زیر تولید یافت می‌شود:

{units': 256, 'regularization': 0.0001, 'dropout_rate': 0.2, 'batch_size': 32'}

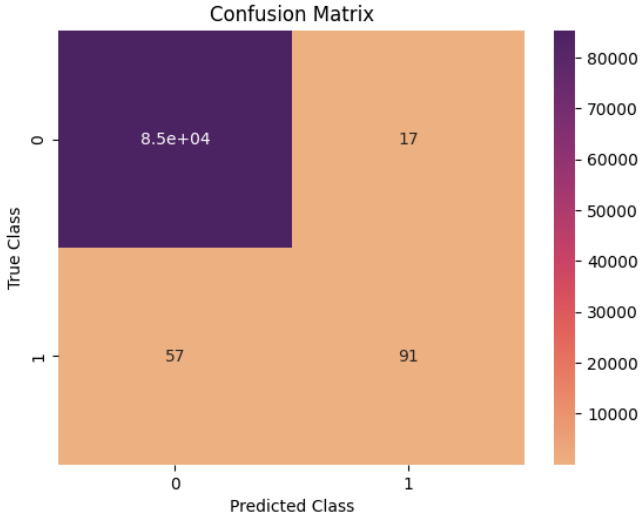
نتیجه ارزیابی این مدل پس از چهل epoch به صورت زیر است:

Accuracy: 0.999

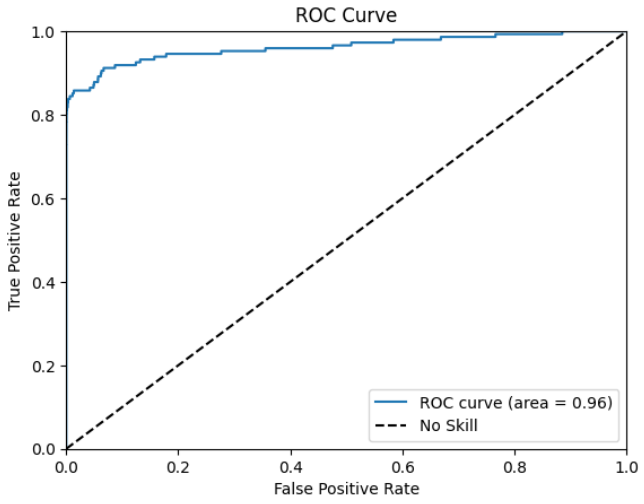
Precision: 0.843

Recall: 0.615

F1 score: 0.711



شکل ۲۴. ماتریس درهم‌ریختگی مدل یافت شده



شکل ۲۵. منحنی ROC مدل یافت شده

این مدل نیز کمی از مدل با دو لایه پنهان قوی‌تر ولی از بهترین مدل بدون لایه مخفی ضعیف‌تر است. یکی از دلایل آن می‌تواند استفاده از random search و جستجوی 10 مدل باشد احتمالا اگر تمام پارامترها را ارزیابی می‌کردیم مدل بهتری میافتیم.

۶-۱. مقایسه‌ی مدل MLP با مدل Logistic Regression

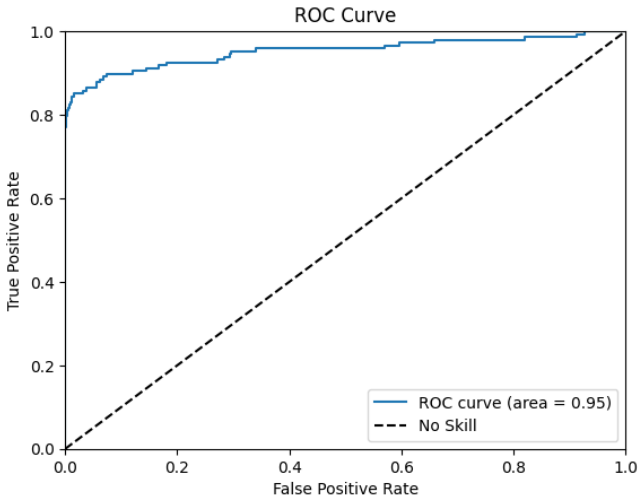
یک طبقه‌بند logistic regression را آموزش می‌دهیم و آن را ارزیابی می‌کنیم:

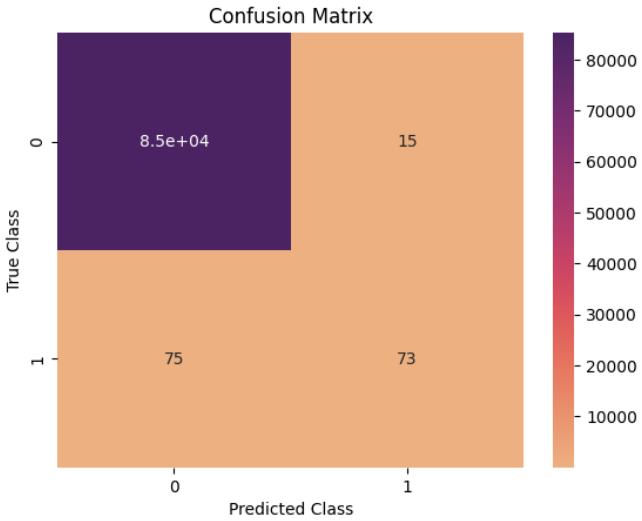
Accuracy: 0.999

Precision: 0.83

Recall: 0.493

F1 score: 0.619





شکل ۲۶. ماتریس درهم‌ریختگی مدل logistic regression شکل ۲۷. منحنی ROC مدل logistic regression

همان طور که مشخص است این مدل عملکرد ضعیف‌تری نسبت به بهترین مدل شبکه عصبی که پیدا کردیم دارد. این مدل در اصل معادل است با شبکه عصبی با لایه ورودی و خروجی بدون لایه مخفی و همچنان در یادگیری ماشین کاربرد دارد. اگر تعداد دادگان کم باشد، استفاده از طبقه‌بند logistic regression می‌تواند نتایج بهتری تولید کند. همچنین توصیف‌پذیری این مدل نسبت به شبکه‌های عصبی در حوزه‌های خاصی می‌تواند دارای اهمیت باشد. از طرفی سرعت و هزینه آموزش این مدل نسبت به شبکه‌های عصبی بهتر است ولی در مواردی مثل این‌جا که اندازه دادگان بسیار بزرگ است شبکه‌های عصبی می‌توانند بهتر عمل کنند.

۱-۷. بررسی تاثیر Standardization به جای Normalization

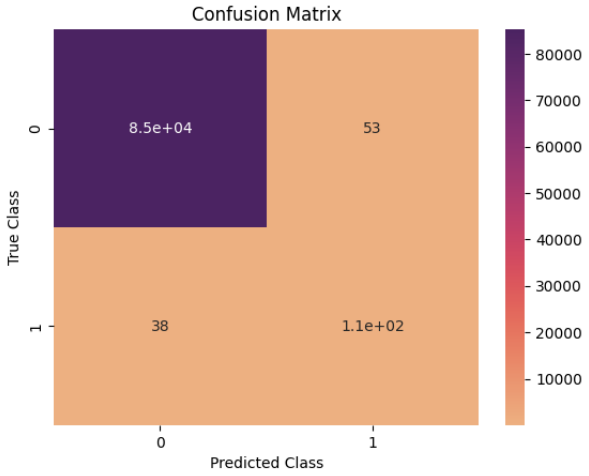
حال بهترین مدلی که ایجاد کردیم یعنی همان مدل ابتدایی با یک لایه پنهان را روی داده‌ای که Standardize شده است آموزش داده و عملکرد آن را ارزیابی می‌کنیم.

Accuracy: 0.999

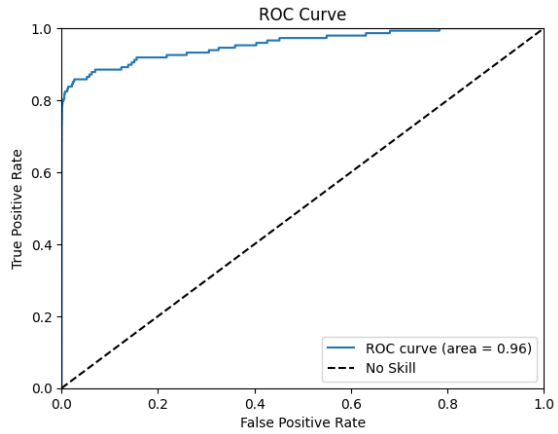
Precision: 0.675

Recall: 0.743

F1-score: 0.707



شکل ۲۸. ماتریس درهم‌ریختگی داده standardize شده



شکل ۲۹. منحنی ROC مدل داده standardize شده

مدل بالا در طبقه‌بندی کلاس صفر که داده بیشتری از آن موجود بود، ضعیف‌تر از مدل‌های بخش قبل است ولی در طبقه‌بندی کلاس یک عملکرد بهتری دارد.

۱-۸. جمع‌بندی

در این تمرین مدل‌های بسیاری را آزمایش کردیم و از بین تمام مدل‌ها، مدل اول بیشترین مقدار **F1-score** و دقت را داشت. البته مدل آخر معیار بازیابی بهتری دارد. با ارزیابی این مدل‌ها متوجه شدیم برای این دادگان، افزایش تعداد لایه‌ها و استفاده از روش‌های **Dropout** و منظم‌سازی، باعث افزایش پیچیدگی مدل و کاهش عملکرد آن می‌شوند. البته به نظر می‌رسید با افزایش تعداد **epoch** این روش‌ها می‌توانند از بیش‌برازش جلوگیری کنند.

بهینه‌سازی پارامترها مدلی با عملکرد متوسط یافت که ممکن است با جست و جوی بیشتر به مدل‌های بهتری دست پیدا می‌کردیم. البته زمان جست و جو و آموزش مدل‌های مختلف زمان نسبتا زیادی بود و با افزایش فضای جستجو حتی این زمان بیشتر هم می‌شود.

خطای اصلی مدل در طبقه‌بندی کلاس یک بود که دلیل اصلی آن هم به دلیل وجود تعداد بسیار کمتری نمونه از این نمونه در بین داده‌های آموزش بود. شاید بتوان معیار جست و جو را بازیابی یا معیارهای مشابه قرار داد که بتوان مدل‌هایی با بازیابی بهتر یافت. همچنین می‌توان با ترکیب نتایج مدل‌ها یک مدل **ensemble** از مدل‌های اول و آخر ساخت تا به دقت و بازیابی بیشتری از هر یک از تک مدل‌ها رسید. همچنین تغییر **threshold** به مقدار مورد نظر با تحلیل منحنی **ROC** نیز می‌تواند کمک کند مرز مورد نظر را برای رسیدن به دقت و بازیابی مناسب پیدا کرد. دقت و بازیابی صحت مدل در تعیین دو کلاس مختلف را تعیین می‌کنند و معمولا با افزایش یکی از آن‌ها صحت مدل در تشخیص کلاس دیگر کاهش میابد و اینکه بدانیم کلاس کلاس اهمیت بیشتری دارد به ما کمک می‌کند تا مدل‌ها را بهتر مقایسه کنیم و به مدلی که مد نظرمان است برسیم. برای مثال در تشخیص تقلب دقت و در تشخیص بیماری کرونا بازیابی اهمیت بیشتری دارد.

طبق مدل‌هایی که آموزش دادیم، با توجه به نوع دادگان، افزایش پیچیدگی مدل برای این دادگان باعث افزایش محاسبات اضافی و افت عملکرد مدل شد. شاید بتوان با انجام پیش‌پردازش‌های متفاوت به مدل‌های بهتری رسید. برای مثال استفاده از **standardization** به جای **normalization** بازیابی مدل را بیشتر کرد. همچنین با توجه به نمودارهای رسم شده، تعداد چهل **epoch** برای آموزش این مدل‌ها مقدار زیادی بود و پس از حدود ده **epoch** عملکرد مدل‌ها تقریبا تغییر زیادی نمی‌کرد. با کاهش **epoch** هزینه آموزش هم کاهش میابد و می‌توانیم مدل‌های بیشتری را آموزش دهیم تا مدل مورد نظر را بیابیم.

به نظر یکی از چالش‌های تشخیص تقلب در داده‌های واقعی که در این دادگان هم وجود داشت نامتعادل بودن دادگان هر کلاس است که باعث می‌شود مدل نتواند به خوبی کلاسی که داده کمتر دارد که کلاس تقلب است را بیابد. افزایش داده معمولا به افزایش عملکرد مدل می‌انجامد. البته جست و جوی بین مدل‌های مختلف برای یافتن مدل بهتر نیز و استفاده از معیارهای درست می‌تواند کمک کند.

اگر مدل را دوباره قرار بود طراحی کنیم، تعداد **epoch** را کاهش می‌دادیم تا با افزایش سرعت آموزش، مدل‌های بیشتری را ارزیابی کنیم. همچنین از پیش‌پردازش‌های مختلف روی داده‌ها استفاده می‌کردیم. یکی از نکات مهم دیگر که باید در دنیای واقعی رعایت کنیم جدا کردن بخشی از داده به عنوان داده تست نهایی است. زیرا جست و جو ارزیابی عملکرد روی داده فعلی می‌تواند باعث بیش‌برازش شود. البته استفاده از روش‌های **cross fold validation** نیز می‌تواند کمک کند.

.

پرسش ۲ – طراحی شبکه عصبی چندلایه در مسئله رگرسیون مقاومت بتن

۲-۱. بررسی دادگان

۲-۱-۱. بررسی ویژگی‌ها

با بررسی منبع دادگان یعنی سایت کگل، اطلاعاتی از دادگان به دست می‌آوریم.

دادگان شامل اطلاعاتی در مورد مقاومت بتن است که نقشی کلیدی در ساخت و ساز دارد. دادگان ویژگی‌های مختلفی دارد که مربوط به مواد تشکیل دهنده بتن هستند که روی مقاومت کلی بتن تاثیر می‌گذارند. همچنین اطلاعات جزئی‌تری نیز در مورد هر یک از ویژگی‌ها در صفحه نوشته شده است. توضیحات ویژگی برچسب بدین صورت است: Strength نشان دهنده مقاومت نهایی بتن است. مقاومت، نشان دهنده میزان تحمل بار و عملکرد کلی در ساخت‌وساز است.

نمونه ای دو داده موجود را رد جدول زیر مشاهده می‌کنیم:

جدول ۳. نمونه‌ای از دادگان

Cement Component	BlastFurnaceSlag	FlyAsh Component	Water Component	Superplasticizer Component	CoarseAggregate Component	FineAggregate Component	AgeIn Days	Strength
266.0	114.0	0.0	228.0	0.0	932.0	670.0	365	52.91
362.6	189.0	0.0	164.9	11.6	944.7	755.8	7	55.90

حال اطلاعات آماری دادگان را نیز به دست آورده و در صفحه بعد مشاهده می‌کنیم.

از جداول صفحه بعد اطلاعاتی در مورد ویژگی‌ها به دست می‌آید. برای برخی از ویژگی‌ها، مقادیر به طور تقریبا یکنواختی توزیع شده‌اند که این ویژگی‌ها عبارتند از: CementComponent، WaterComponent، CoarseAggregateComponent و FineAggregateComponent. همچنین تمام اعداد موجود نامنفی هستند که چون نشان دهنده میزان مواد تشکیل دهنده بتن هستند منطقی است. از طرفی برخی از ویژگی‌ها مقادیر صفر هم دارند که یعنی می‌توان از آن مواد در ساخت بتن استفاده نکرد. همچنین این

ویژگی‌ها اکثرا مقدار صفر دارند و توزیع‌شان به نظر چوله به راست است. این ویژگی‌ها عبارتند از: BlastFurnaceSlag، FlyAshComponent و SuperplasticizerComponent. در نهایت ویژگی AgeInDays هم نشان دهنده تعداد روزهایی است که بتن گذاشته می‌شود تا خشک شود. و طبق توضیحات سایت، مقاومت بتن در گذر زمان افزایش میابد. این ویژگی حداقل مقدار یک روز و حداکثر مقدار یک سال دارد ولی این ویژگی هم چوله به راست است.

جدول ۴. اطلاعات آماری دادگان

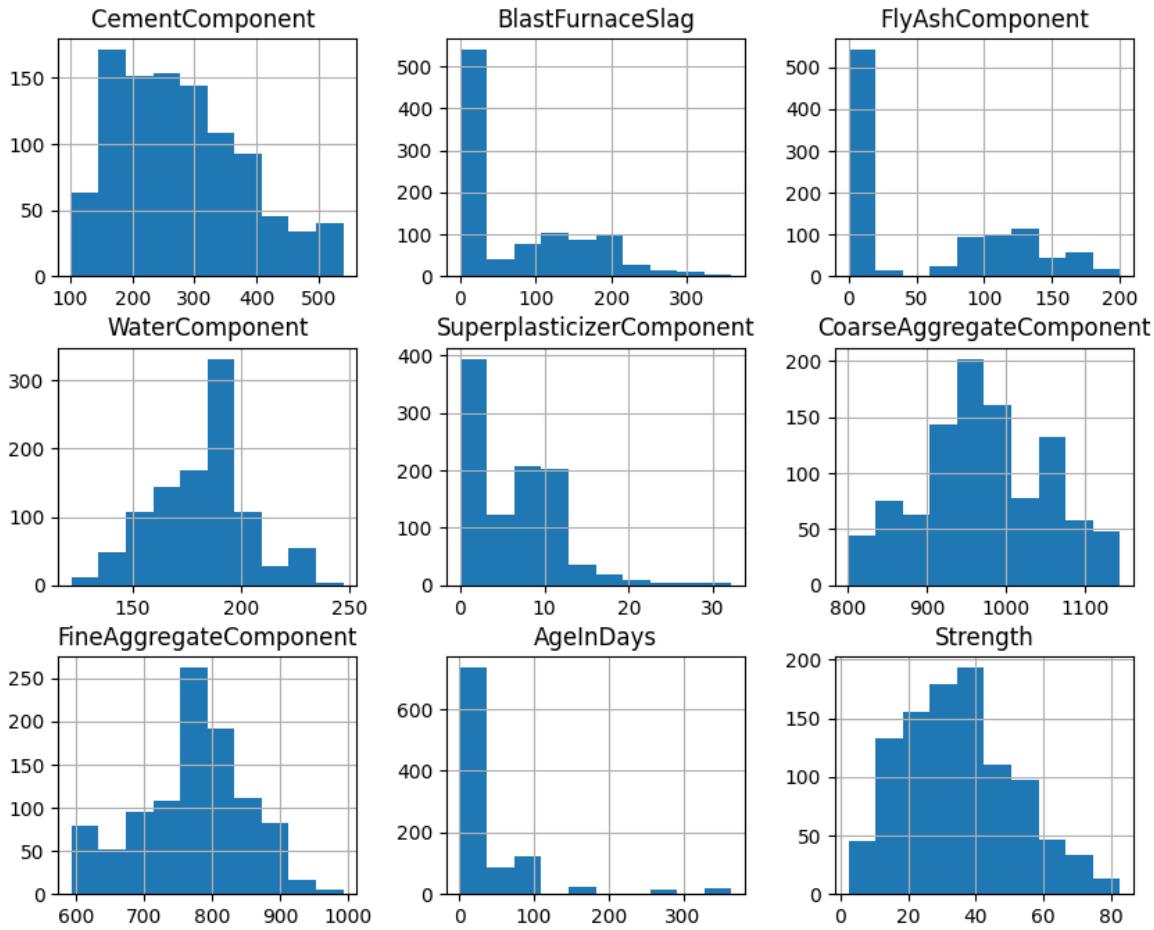
	Cement Component	BlastFurnaceSlag	FlyAsh Component	Water Component	Superplasticizer Component
count	1030.000000	1030.000000	1030.000000	1030.000000	1030.000000
mean	281.167864	73.895825	54.188350	181.567282	6.204660
std	104.506364	86.279342	63.997004	21.354219	5.973841
min	102.000000	0.000000	0.000000	121.800000	0.000000
25%	192.375000	0.000000	0.000000	164.900000	0.000000
50%	272.900000	22.000000	0.000000	185.000000	6.400000
75%	350.000000	142.950000	118.300000	192.000000	10.200000
max	540.000000	359.400000	200.100000	247.000000	32.200000

جدول ۵. ادامه اطلاعات آماری دادگان

	CoarseAggregateComponent	FineAggregateComponent	AgeInDays	Strength
count	1030.000000	1030.000000	1030.000000	1030.000000
mean	972.918932	773.580485	45.662136	35.817961
std	77.753954	80.175980	63.169912	16.705742
min	801.000000	594.000000	1.000000	2.330000
25%	932.000000	730.950000	7.000000	23.710000
50%	968.000000	779.500000	28.000000	34.445000
75%	1029.400000	824.000000	56.000000	46.135000
max	1145.000000	992.600000	365.000000	82.600000

همچنین با گرفتن اطلاعات دادگان متوجه می‌شویم در کل 1030 نمونه داریم که هیچ یک null نیست و تمام ویژگی‌ها از جنس float هستند به جز ویژگی روز که از جنس int است. همچنین 25 نمونه از دادگان دارای مقادیر یکسان هستند که آن‌ها را حذف می‌کنیم زیرا اطلاعاتی جدیدی به مدل اضافه نمی‌کنند و ممکن است هم در داده آموزش باشند و هم در داده تست و باعث افزایش نادرست مقدار صحت مدل شوند. البته اگر این داده‌ها به درستی در چندین نمونه‌برداری جمع شده‌اند و به دلیل خطا در نمونه‌برداری ایجاد نشده‌اند، بهتر است که حذف‌شان نکنیم ولی از آن‌جایی که تعداد ویژگی‌ها زیاد و اکثرا اعداد حقیقی هستند و با توجه به تعداد کم داده‌های تکراری نسبت به تعداد کل دادگان، به نظرمان احتمال اینکه اشتباهی صورت گرفته باشد بالا است و به همین ترتیب این دادگان را حذف می‌کنیم.

حال نمودار هیستوگرام برای هر ویژگی را رسم می‌کنیم.

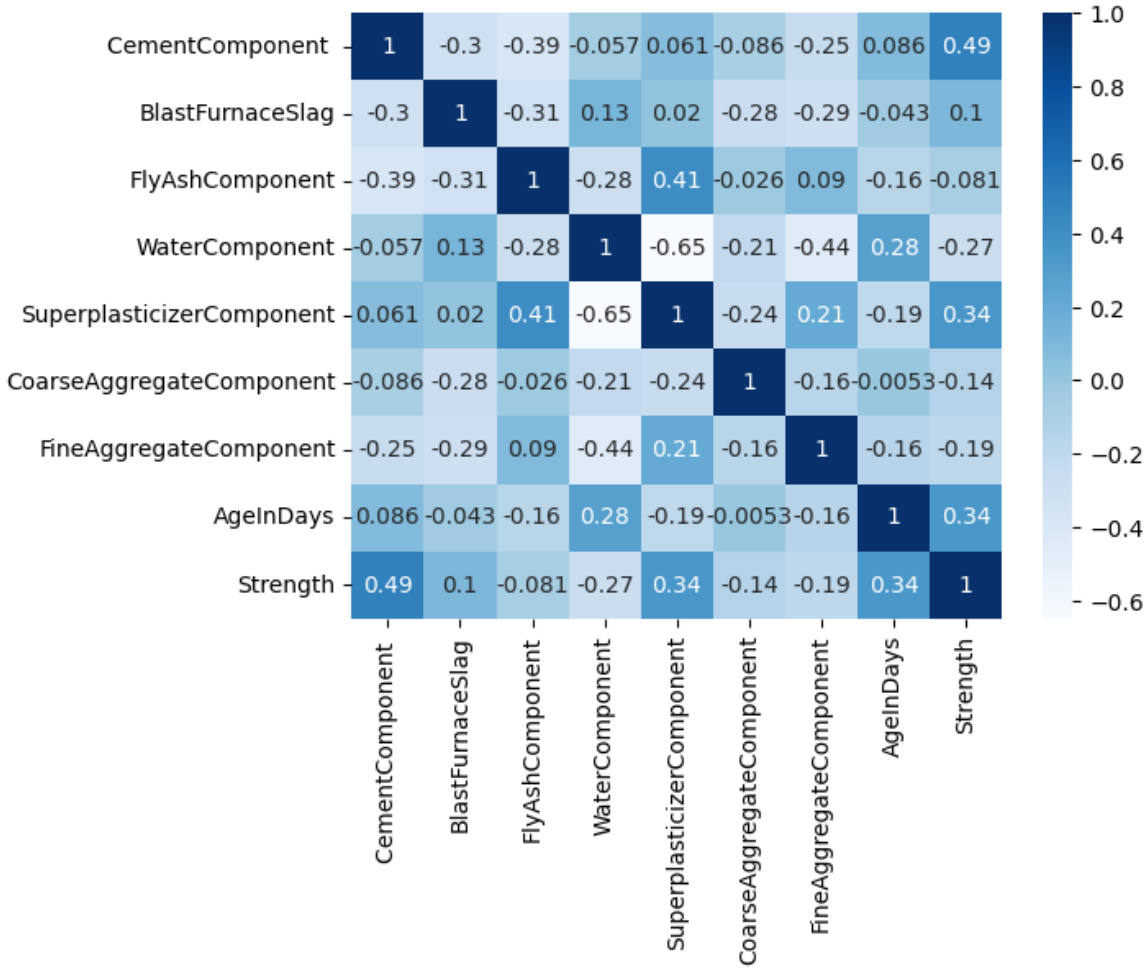


شکل ۳۰. هیستوگرام ویژگی‌های موجود در دادگان

اطلاعاتی که از روی اطلاعات آماری به دست آوردیم در مورد ویژگی‌ها، در نمودارهای هیستوگرام نیز مشخص است. به جز چهار ویژگی ذکر شده که چوله به راست هستند، بقیه ویژگی‌ها شکل زنگوله‌ای و نرمال دارند.

۲-۱-۲. بررسی همبستگی

حال ماتریس همبستگی را بین ویژگی‌ها رسم می‌کنیم تا بدانیم کدام ویژگی‌ها همبستگی بیشتری با ویژگی برچسب دارند و از طرفی کدام ویژگی‌ها با هم همبستگی دارند.



شکل ۳۱. ماتریس همبستگی ویژگی‌ها

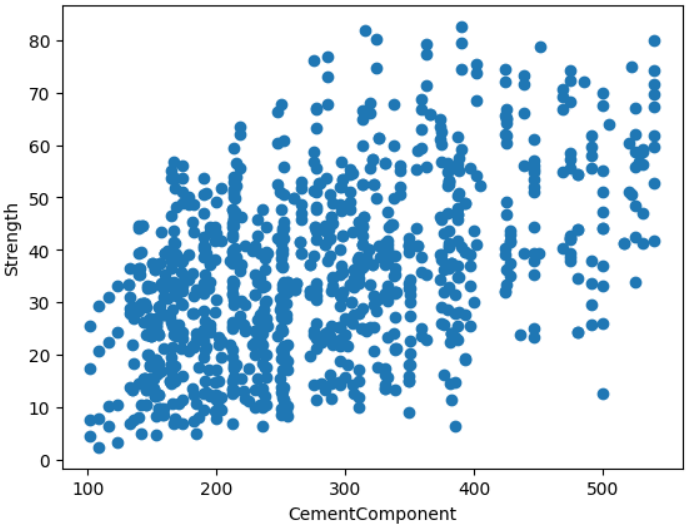
از ماتریس رسم شده می‌توان اطلاعات زیر را به دست آورد:

- بیشترین همبستگی مقاومت به ترتیب با ویژگی‌های سیمانس سپس superplasticizer و روز است و با دو ویژگی BlastFurnaceSlage و FlyAshComponent تقریبا همبستگی ندارد.
- همبستگی میان دو ویژگی superplasticizer و میزان آب زیاد و در خلاف جهت هم است. پس از جست و جو در مورد مواد superplasticizer اطلاعات زیر را از ویکی پدیای فارسی میابیم که می‌تواند دلیل این همبستگی بالا را توضیح دهد: "فوق روان‌کننده‌ها (SPs) که به عنوان کاهنده قوی آب نیز شناخته می‌شوند، افزودنی‌هایی هستند که در ساخت بتن با مقاومت بالا استفاده می‌شوند. روان‌کننده‌ها ترکیبات شیمیایی هستند که امکان تولید بتن را با تقریباً ۱۵ درصد آب کمتر فراهم می‌کنند. فوق روان‌کننده‌ها امکان کاهش مقدار آب را تا ۳۰ درصد یا بیشتر فراهم می‌کنند."

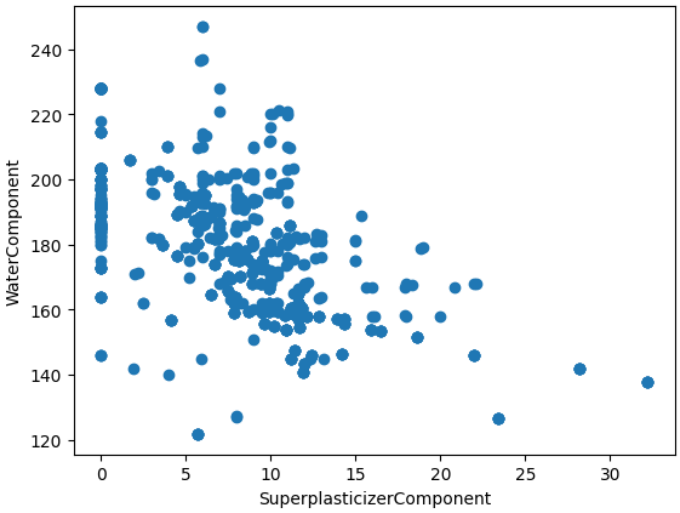
- همچنین برخی دیگر از ویژگی‌ها هم همبستگی نسبتا زیادی دارند از جمله: FineAggregate با میزان آب و superplasticizer با FlyAsh.

با توجه به ماتریس همبستگی، می‌توانیم از بین ویژگی‌هایی که همبستگی بالایی با یکدیگر دارند، تنها ویژگی که همبستگی بالاتری با ویژگی برچسب دارد را نگه داریم. بدین ترتیب، باعث افزایش استقلال ورودی‌های مدل می‌شویم که می‌تواند باعث عملکرد آن شود. بدین ترتیب دو ویژگی میزان آب و FlyAsh را به دلیل وجود همبستگی بالا با superplasticizer و همبستگی نسبتا پایین با مقاومت حذف می‌کنیم. البته در نهایت یک بار هم مدلی با وجود تمام ویژگی‌ها آموزش خواهیم داد تا تاثیر حذف کردن ویژگی‌ها را بهتر درک کنیم.

برای مشاهده همبستگی بین ویژگی‌ها، دو نمودار scatter یکی بین مقاومت و میزان سیمان که بیشترین همبستگی را با ویژگی هدف یعنی مقاومت دارد و دیگری بین میزان آب و superplasticizer که بیشترین میزان همبستگی بین دو ویژگی است.



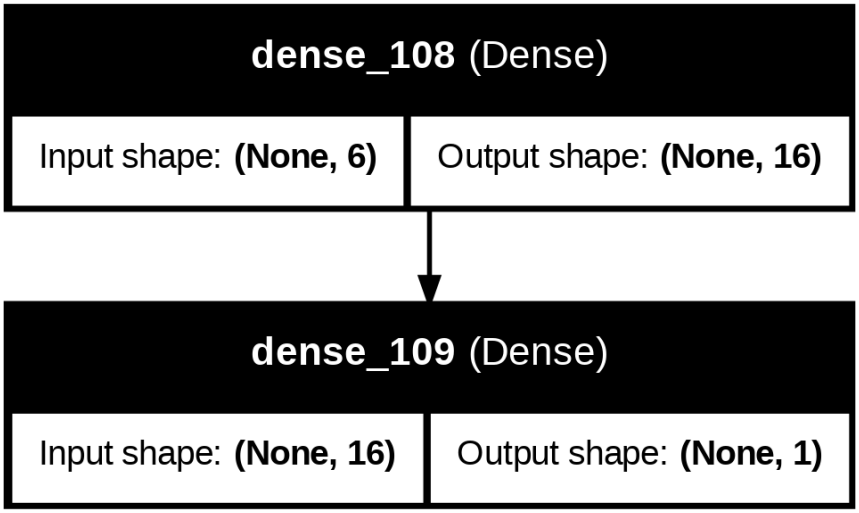
شکل ۳۲. نمودار scatter مقاوت و میزان سیمان



شکل ۳۳. نمودار scatter میزان آب و superplasticizer

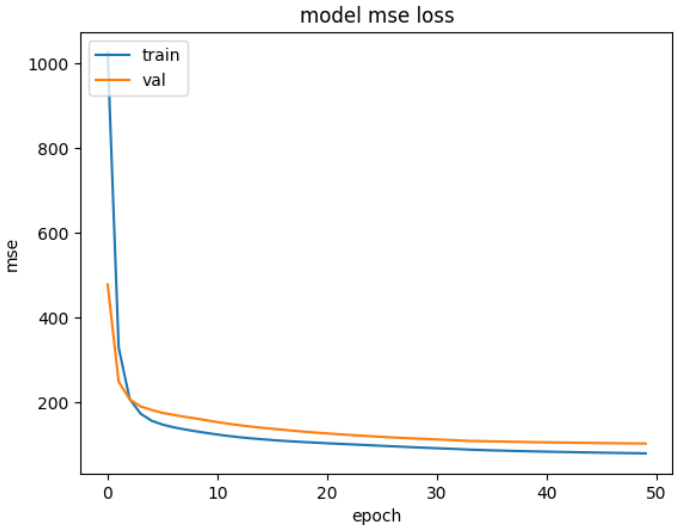
۲-۲. پیاده سازی مدل شبکه عصبی چندلایه

دو مدل با معماری زیر ایجاد می‌کنیم که یکی 16 و دیگری 32 نورون در لایه مخفی خود دارد.

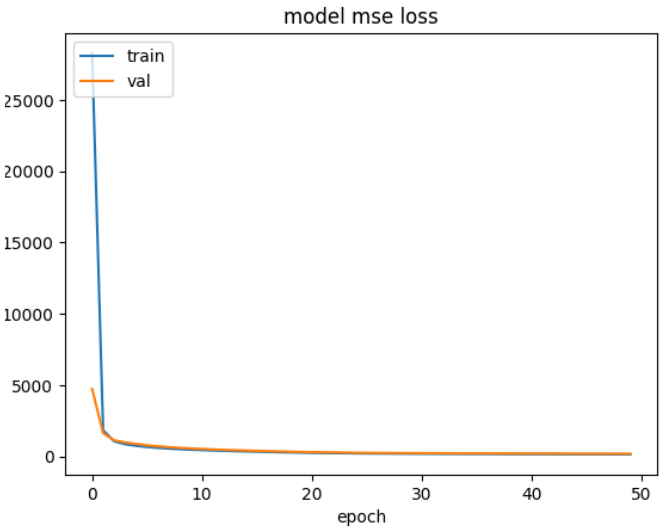


شکل ۳۴. معماری مدل اول

نمودار خطای دو مدل بدین صورت می‌شود.



شکل ۳۶. معماری شبکه با 32 نورون لایه پنهان



شکل ۳۵. معماری شبکه با 16 نورون لایه پنهان

همان طور که از دو نمودار بالا مشخص است، هر دو مدل به خوبی آموزش دیده و بیش‌برازش هم رخ نداده است. حال دو مدل را روی داده تست ارزیابی می‌کنیم.

جدول ۶. خطای دو مدل با تعداد نورون متفاوت روی داده تست

	Model 1 with 16 neurons	Model 2 with 32 neurons
MSE	184.24888610839844	103.38710021972656
MAE	10.689740180969238	7.641387939453125

مشخص است که مدلی که نورون‌های بیشتری داشته به خطای کمتری رسیده است.

۲-۲-۱. آموزش با تمام ویژگی‌ها

در بخش اول دو ویژگی را حذف کردیم و مدل‌های بالا را آموزش دادیم. حال دو مدل با معماری مشابه ولی با تمامی ویژگی‌ها آموزش می‌دهیم تا تاثیر حذف ویژگی‌ها را درک کنیم.

جدول ۷. خطای دو مدل آموزش دیده با تمام ویژگی‌ها روی داده تست

	Model 1	Model 2
MSE	149.39707946777344	114.64889526367188
MAE	9.604386329650879	8.275128364562988

می‌توان دید که حذف ویژگی‌ها با همبستگی بالا باعث بهبود عملکرد مدل دوم که مدل بهتر است شده است پس در ادامه هم از همین معماری و بدون آن دو ویژگی استفاده خواهیم کرد.

۲-۳. بررسی تغییرات تنظیمات مدل

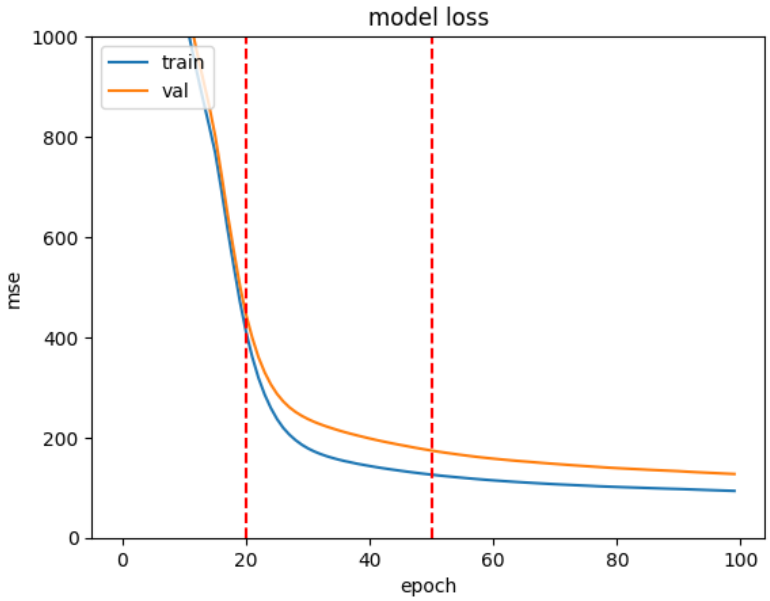
۲-۳-۱. تاثیر ایپاک‌ها

ابتدا تاثیر تعداد epoch را بررسی می‌کنیم. مدل قسمت قبل را سه بار به epoch های 20، 50 و 100 آموزش می‌دهیم و سه مدل به دست آمده را مقایسه می‌کنیم. خطای mse مدل‌ها به ازای epochهای مختلف بدین صورت می‌شود.

جدول ۸. خطای مدل به ازای آموزش با epochهای مختلف

	MSE
20 epochs	502.2999572753906
50 epochs	176.07669067382812
100 epochs	127.23625183105469

مشخص است که با افزایش تعداد epoch خطای مدل کاهش یافته است. البته قابل ذکر است که ما چندین بار این تنظیمات را تست کردیم و گاهی برای مدل با 16 نورون، بیش‌برازش رخ می‌داد و خطای مدل در صد epoch نسبت به پنجاه epoch افزایش میافت.



شکل ۳۷. خطا به ازای آموزش با epochهای مختلف

۲-۳-۲. مقایسه توابع هزینه

ابتدا هر یک از سه تابع خطا را به طور مختصر توضیح خواهیم داد.

تابع خطای اول خطای میانگین مربعات خطا یا mean squared error(MSE) است. رابطه این خطا به صورت زیر است.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(Y_i - \hat{Y}_i)^2$$

که N برابر تعداد نمونه‌های داده تست، Y_i برابر مقدار واقعی ویژگی برچسب برای نمونه iام است و $\hat{Y}_i$ هم نشان دهنده خروجی مدل برای نمونه iام است.

از مزیت‌های این تابع خطا می‌توان گفت: یک تابع خطای ساده است و همچنین مشتق‌پذیر است در نتیجه هزینه آموزش کاهش میابد و تابع خطای محدب است که باعث می‌شود یک نقطه بهینه مطلق داشته باشد که باعث می‌شود آموزش و هم‌گرایی آسان‌تر شود.

البته این تابع خطا ایراداتی دارد مثل اینکه واحد آن مربع واحد ورودی است و باعث کاهش توصیف‌پذیری آن می‌شود به همین دلیل گاهی از آن جذر می‌گیریم و استفاده می‌کنیم. همچنین به دلیل

توان دو رساندن، باعث می‌شود که نسبت به داده‌های پرت مقاوم نباشد و تعداد کمی خطا هم می‌تواند باعث افزایش زیاد مقدار خطا شود.

تابع خطای بعدی میانگین قدرمطلق خطا یا mean absolute error(mae) است. رابطه این خطا به صورت زیر است.

$$MAE = \frac{1}{N}\sum_{i=1}^{N} |Y_i - \widehat{Y}_i|$$

این تابع خطا نیز قابل درک و توصیف است. از طرفی مشکل متفاوت بودن واحد خطا را مثل mse ندارد. و برخلاف mse نسبت به داده‌های پرت مقاوم است. مشکل اصلی این تابع خطا نداشتن مشتق حول نقطه صفر است.

تابع خطای آخر زیان هوبر Huber loss است که رابطه آن به صورت زیر است.

$$\begin{cases} Huber = \frac{1}{N}\sum_{i=1}^{N}(Y_i - \widehat{Y}_i)^2 = MSE & |Y_i - \widehat{Y}_i| \le \delta \\ Huber = \frac{1}{N}\sum_{i=1}^{N}|Y_i - \widehat{Y}_i| = MAE & |Y_i - \widehat{Y}_i| > \delta \end{cases}$$

همان طور که از رابطه آن مشخص است، این تابع خطا قصد دارد با ترکیب دو تابع خطای قبلی، مشکلات هر یک را حل کند. این تابع خطا می‌تواند برای خطاهای کم برابر خطای mae باشد و برای خطاهای بزرگ برابر mse شود و بدین ترتیب، نسبت به داده‌های پرت نیز مقاوم می‌شود.

مشکل اصلی این تابع خطا تعیین هایپرپارامتر δ است که مرز بین رفتار به صورت mae یا mse را مشخص می‌کند و در خطای کلی و میزان حساسیت نسبت به داده‌های پرت نقش اساسی دارد.

حال سه مدل با معماری یکسان را به مدت 100 epoch آموزش می‌دهیم و میزان خطا به صورت جدول زیر به دست می‌آید.

جدول ۹. خطای مدل به ازای آموزش با epochهای مختلف

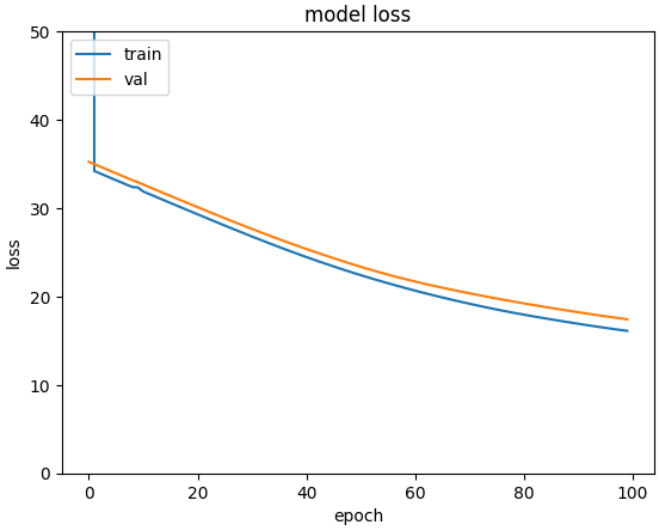
	Trained with MSE Model 1	Trained with MAE Model 2	Trained with HUBER Model 3
MSE Loss	94.3820	151.5988	92.4613
MAE Loss	7.7803	9.5310	7.1066
HUBER Loss	7.2999	9.0580	6.5812

اینکه کدام معیار خطا مناسب است را نمی‌توان با آموزش مدل‌ها متوجه شد چرا که برای مقایسه مدل‌ها هم نیاز به معیاری هست و اینکه کدام معیار را به عنوان خطا در نظر بگیریم روی انتخاب‌مان اثر دارد. ما چندین بار مدل‌ها را اجرا کردیم و به طور کلی می‌توان گفت هر مدل سعی می‌کند معیار خطای خود را کاهش دهد ولی در عین حال به دلیل ماهیت مشابه خطاها باعث کاهش بقیه معیارها هم می‌شود. پس برای انتخاب خطا شاید بهتر باشد با توجه به هدف و نوع دادگان معیار خطای مناسب را انتخاب کنیم اما با توجه به جدول بالا، مدل با خطای huber در این اجرا، هر سه خطا را بیش از دو مدل دیگر کاهش داد پس در قسمت بعد از آن استفاده خواهیم کرد.

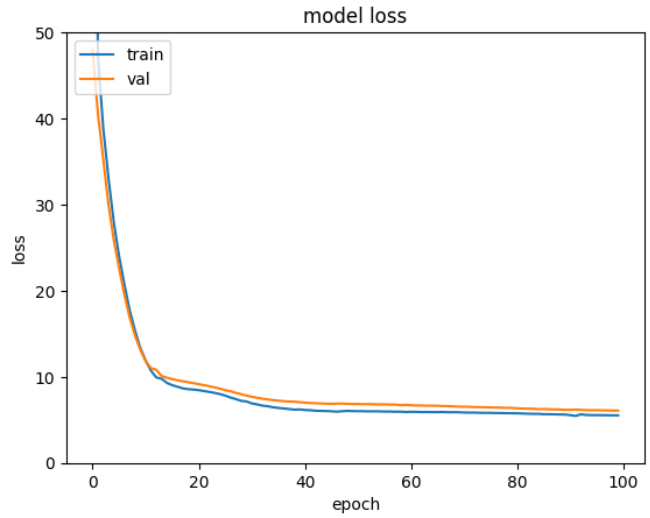
۲-۳-۳. مقایسه توابع بهینه‌ساز

سه تابع بهینه‌ساز متداول عبارتند از SGD، ADAM و RMSprop. برای مقایسه این سه تابع بهینه‌ساز، سه مدل با معماری یکسان ایجاد کردیم و با سه تابع بهینه‌ساز مختلف آموزش دادیم.

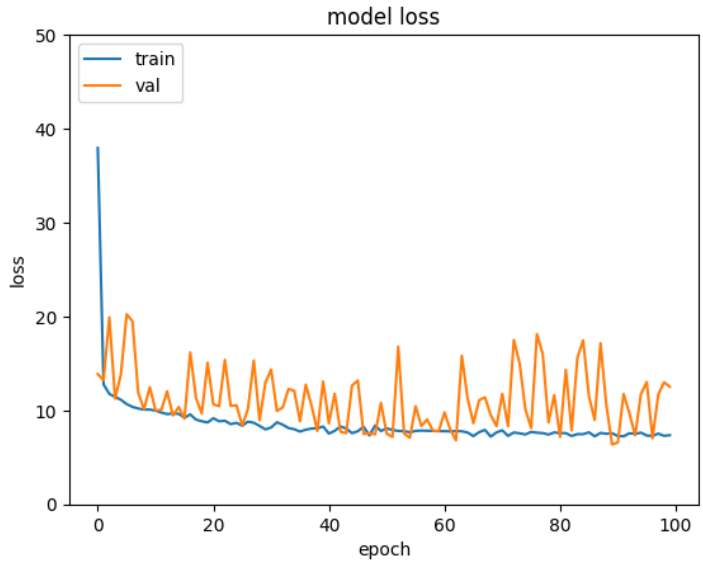
ابتدا نمودار خطای سه مدل را بررسی می‌کنیم.



شکل ۳۸. خطا برای تابع بهینه ساز sgd



شکل ۳۹. خطا برای تابع بهینه ساز adam



شکل ۴۰. خطا برای تابع بهینه ساز rmsprop

نکته قابل توجه که در چندین بار آموزش هم مشخص بود، نویزی و ناپایدار بودن مدل با تابع بهینه‌ساز rmsprop نسبت به دو تابع دیگر است که می‌تواند باعث ایجاد مدل‌هایی با عملکرد گوناگون شود. همچنین قابل مشاهده است که خطای مدل با تابع بهینه‌ساز adam، بسیار سریع‌تر از خطای دو مدل دیگر کاهش میابد و به خطای کمتری می‌رسد.

همچنین میزان خطا روی داده تست برای سه تابع بهینه‌ساز تفاوت نسبتا زیادی دارد. با این که ممکن است این تفاوت‌ها به دلیل اعداد تصادفی در محاسبات یا نوع داده باشند و لزوما به معنی بهتر بودن یک تابع بهینه‌ساز نیستند ولی با چندین بار آموزش و ارزیابی نتایج مشابهی گرفتیم.

SGD: 17.42364501953125

ADAM: 6.080770492553711

RMSprop: 12.513409614562988

مشخص است که خطای مدل با تابع بهینه‌ساز adam بسیار کمتر از دو مدل دیگر است. با چند بار آموزش مدل‌ها هم تغییر زیادی در خطاها مشاهده نکردیم و به نظر برای این داده تست و معماری شبکه، تابع بهینه‌ساز مناسب adam است.

۲-۴. جمع‌بندی

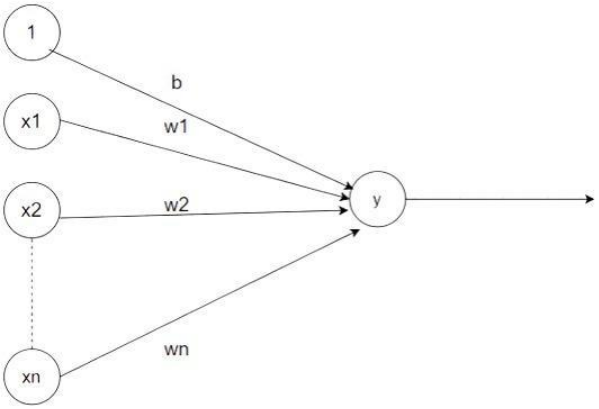
از ماتریس همبستگی دریافتیم که میزان سیمان موجود در بتن بیشترین همبستگی را با مقاومت بتن دارد که چون می‌دانیم سیمان از اجزای اصلی بتن است به نظر نتیجه‌ای منطقی است.

ما با آموزش مدل‌های مختلف به نتایجی رسیدیم. افزایش تعداد نورون در بعضی اجراها باعث افزایش عملکرد می‌شد در حالیکه در برخی اجراها این افزایش عملکرد آن‌قدر زیاد نبود. از طرف دیگر، افزایش تعداد ایپاک از ۵۰ به ۱۰۰ در برخی اجراها باعث بیش‌برازش و بعضی اجراها باعث افزایش عملکرد می‌شد. مقایسه توابع هزینه با صرفا آموزش مدل‌ها زیاد امکان‌پذیر نیست چرا که بسته به معیار ارزیابی می‌توان یک تابع خطا را برتر از دیگری دانست. نهایتا طبق اجراهای مختلف به نظر ما تابع هزینه adam عملکرد بهتری نسبت به sgd و rmsprop دارد. در کل مدلی که در نهایت از نتیجه بخش‌های مختلف به آن رسیدیم بدین صورت است: ۳۲ نورون در لایه مخفی، با ۱۰۰ ایپاک آموزش با تابع خطای هوبر و تابع بهینه‌ساز adam.

در مدل‌سازی رگرسیون، معیار ارزیابی متفاوت می‌شود و باید انتخاب معیار مناسب می‌تواند کمک کند. همچنین همبستگی بین ویژگی‌ها با یکدیگر و با ویژگی مورد نظر نیز باید در نظر گرفته شود و در صورت نیاز برخی ویژگی‌ها حذف شوند.

پرسش ۳ – پیاده سازی Adaline برای دیتاست IRIS

۳-۱. آشنایی با Adaline



شکل ۴۱. تصویری از یک شبکه adaline ازکتاب فاست

به شبکه‌ای با یک واحد خطی Adaline گوییم. در چنین شبکه‌ای تنها یک واحد خروجی قرار دارد و مقدار خروجی ۱+ یا ۱- است. همچنین وزن‌های بین واحد ورودی و خروجی قابل تغییر است. الگوریتم Adaline به نام delta بدین صورت است:

۰) مقداردهی اولیه وزن‌ها و ترم بایاس به مقادیر غیر صفر کوچک و انتخاب مقدار کوچکی برای learning rate.

۱) اجرای مراحل ۲ تا ۴ به ازای هر داده آموزش.

۲) برابر قرار دادن $s_i \rightarrow x_i$ به ازای i از ۱ تا n

۳) حساب کردن net برابر جمع وزن‌دار ورودی‌ها بدین صورت:

$$net = \sum_{i=1}^{n} w_i x_i + b$$

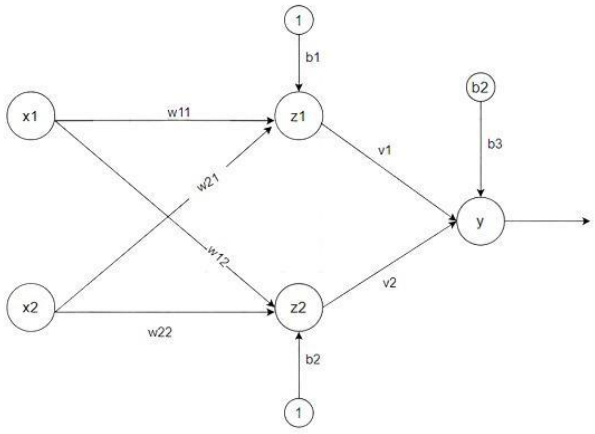
۴) آپدیت کردن وزن‌ها و بایاس با استفاده از رابطه زیر که α نرخ یادگیری است:

$$w_i^+ = w_i^- + \alpha(t - net)x_i \qquad b^+ = b^- + \alpha(t - net)$$

۵) اگر تابع هزینه حساب شده برای تمام داده‌های آموزشی از یک آستانه‌ای کمتر شد می‌رویم به مرحله ۶ وگرنه به مرحله ۱.

۶) پایان

مدل madaline از تعدادی Adaline موازی و یک واحد خروجی تشکیل شده است به طوری که وزن‌های بین لایه ورودی و لایه پنهان قابل تغییر ولی وزن‌های لایه آخر ثابتند.



شکل ۴۲. تصویری از یک شبکه madaline از کتاب فاست

الگوریتم آپدیت وزن‌های مدل Madaline بدین صورت است:

۰) مقداردهی اولیه وزن‌ها و ترم بایاس به مقادیر غیر صفر کوچک و انتخاب مقدار کوچکی برای learning rate و مقداردهی وزن‌های لایه خروجی به ۰.۵ یعنی:

$$v_1 = v_2 = b = 0.5$$

۱) اجرای مراحل ۲ تا ۶ به ازای هر داده آموزش.

۲) برابر قرار دادن $s_i \rightarrow x_i$ به ازای i از ۱ تا n

۳) حساب کردن net برابر جمع وزن‌دار واحدهای Adaline بدین صورت:

$$z_{in1} = b_1 + x_1 w_{11} + x_2 w_{21}$$
$$z_{in2} = b_2 + x_1 w_{12} + x_2 w_{22}$$

۴) حساب کردن خروجی واحدهای Adaline بدین صورت:

$$f(z) = 1 if z \geq 0 and (-1) if z < 0$$
$$z_1 = f(z_{in1}) \qquad z_2 = f(z_{in2})$$

۵) حساب کردن خروجی:

$$y_{in} = b_3 + z_1 v_1 + z_2 v_2$$

با عبور مقدار بالا از تابع فعال‌سازی خروجی کلی به دست می‌آید.

۶) آپدیت کردن وزن‌ها و بایاس با استفاده از رابطه زیر که α نرخ یادگیری است:

اگر t=y باشد وزن‌ها تغییری نمی‌کنند وگرنه اگر t=1 باشد، وزن‌ها را آپدیت می‌کنیم.

$$w_{ij}^{+} = w_{ij}^{-} + \alpha(t - z_{inj})x_i \qquad b_j^{+} = b_j^{-} + \alpha(t - z_{inj})$$

۷) اگر تابع هزینه حساب شده برای تمام داده‌های آموزشی از یک آستانه‌ای کمتر شد می‌رویم به مرحله ۸ وگرنه به مرحله ۱.

۸) پایان

مدل MLP تفاوت‌هایی با مدل Madaline دارد. برخلاف Madaline که تعداد واحد Adaline را کنار هم قرار می‌دهد، مدل MLP تعدادی واحد Preceptron را کنار هم قرار می‌دهد. در نتیجه در MLP خطا تابعی از خروجی واقعی و خروجی مدل پس از گذشتن از تابع فعال‌سازی است در حالیکه در Madaline خطا تابعی از خروجی واقعی و خروجی مدل پیش از اعمال تابع فعال‌سازی است. همچنین MLP برخلاف Madaline محدودیتی روی تعداد لایه‌ها ندارد و می‌تواند به تعداد دلخواه لایه پنهان داشته باشد. همچنین در Madaline تابع فعال‌ساز تابع sign است که تابعی سخت است ولی در MLP توابع نرم مشتق‌پذیر مانند tanh جایگزین می‌شوند. با داشتن تابع فعال‌ساز مشتق‌پذیر، MLP روشی سیستماتیک براساس گردیان برای یادگیری مدل می‌دهد. همچنین به دلیل وجود توابع نرم در MLP می‌توان خروجی غیر گسسته برخلاف Madaline ایجاد کرد و برای حل مسائل رگرسیون از آن استفاده کرد. بدین ترتیب می‌توان برخلاف مدل Madaline با افزایش تعداد لایه‌ها و استفاده از توابع فعال‌سازی نرم، مسائل پیچیده‌تری مثل طبقه‌بندی غیرمحدب هم انجام داد. همچنین با ایجاد تغییراتی در مدل می‌توان همبستگی و اغتشاش موجود در ورودی را نیز برخلاف Madaline از بین برد.

۳-۲. آماده‌سازی دادگان

پس از لود کردن دادگان با کمک کتاب‌خانه sklearn، توضیحات مربوط به این دادگان را می‌خوانیم تا اطلاعاتی در مورد آن به دست آوریم. دادگان iris یکی از معروف‌ترین دادگان در شناسایی الگو به شمار می‌رود. دادگان دارای سه کلاس برچسب است که گونه گل را نشان می‌دهد و این کلاس‌ها متعادل هستند به طوریکه ۵۰ نمونه از هر کلاس وجود دارد. همچنین دو کلاس اول از هم به طور خطی قابل جداسازی هستند.

جدول ۱۰. نمونه‌ای از دادگان

sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	species
6.1	2.8	4.7	1.2	1
5.7	3.8	1.7	0.3	0

در این دادگان چهار ویژگی طول و عرض کاسبرگ و گلبرگ به واحد سانتی‌متر قرار دارد. همان‌طور که خواسته شده، دو ویژگی طول و عرض گلبرگ را به همراه دو کلاس setosa و versicolor نگه می‌داریم. هم چنین به گل‌های کلاس setosa مقدار ۱- و به گل‌های کلاس versicolor مقدار ۱ را می‌دهیم تا بتوانیم از مدل Adaline استفاده کنیم.

پس از اعمال تغییرات ذکر شده، ۴۲ داده با مقادیر یکسان در دادگان وجود دارد ولی ما این‌ها را حذف نمی‌کنیم زیرا می‌دانیم این پیش از اعمال تغییرات داده تکراری نداشتیم و پس از حذف دو ویژگی، باعث بوجود آمدن داده تکراری شدیم در صورتی که در حقیقت این داده‌ها متعلق به دو گل متفاوت هستند.

حال داده آموزش و تست را جدا می‌کنیم و بعد ویژگی‌ها را نرمالایز می‌کنیم. جدا کردن داده آموزش و تست پیش از نرمالایز کردن ضروری است چرا که باعث می‌شود اطلاعاتی از داده تست وارد داده آموزش نشود.

۳-۳. پیاده‌سازی Adaline

حال تابعی ایجاد می‌کنیم که الگوریتم ذکر شده در بالا را اجرا کند. البته در قسمت‌هایی که می‌توانیم، از خواص جبرخطی و آرایه‌های numpy برای افزایش سرعت و خوانایی بیشتر استفاده می‌کنیم. تابعی که پیاده کردیم، دادگان آموزش، مقدار نرخ یادگیری و تعداد ایپاک‌های یادگیری را دریافت می‌کند. ابتدا به بردار ویژگی‌های X مقدار یک را به عنوان یک ویژگی اضافه می‌کنیم. بدین ترتیب می‌توانیم با بایاس هم به عنوان یک وزن رفتار کنیم. سپس وزن‌ها و دو آرایه خطا و صحت را ایجاد و ادامه الگوریتم را پیاده می‌کنیم که به ازای هر داده آموزش وزن‌ها را آپدیت و مقدار خطا و دقت را محاسبه کند.

همان طور که خواسته شده این تابع باید بتواند برای چندین نورون در لایه خروجی نیز کار کند پس در صورتی که آرایه y بیش از یک بعد داشت، الگوریتم را به ازای هر یک از خروجی‌ها حساب می‌کنیم و در نهایت بردارها را ترکیب و برمی‌گردانیم.

بدین ترتیب تابع Adaline به صورت زیر می‌شود.

```
def adaline(X, y, learning_rate, num_epochs):
  X = np.hstack((X, np.ones(X.shape[0]).reshape(-1, 1)))

  def delta(X,y,learning_rate, num_epochs):
    W = np.random.random((num_epochs+1, X.shape[1]))
    Loss = np.zeros(num_epochs+1)
    Accuracy = np.zeros(num_epochs+1)

    for i in range(1, num_epochs+1):
      W[i, :] = W[i-1, :]
      for j,x in enumerate(X):
        net = x @ W[i,:]
        W[i,:] += learning_rate * (y[j] - net) * x
        Loss[i] += (y[j] - net)**2
        Accuracy[i] += (y[j] == (1 if net >= 0 else -1))

      Loss[i] /= X.shape[0]
      Accuracy[i] /= X.shape[0]
    return W[1:,:], Loss[1:], Accuracy[1:]

  if y.ndim==1:
    return delta(X,y,learning_rate, num_epochs)

  W, Loss, Accuracy = delta(X,y[:,0],learning_rate, num_epochs)
  W = W.reshape((1,-1,W.shape[1]))

  for i in range(1, y.shape[1]):
    W_new, Loss_new, Accuracy_new = delta(X, y[:,i], learning_rate,
num_epochs)
    W = np.vstack((W, W_new.reshape((1, -1, W_new.shape[1]))))
    Loss = np.vstack((Loss, Loss_new))
    Accuracy = np.vstack((Accuracy, Accuracy_new))
  return W, Loss, Accuracy
```

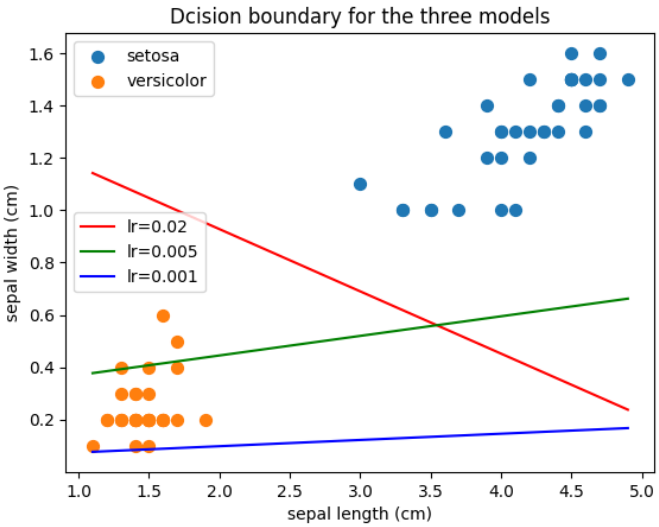
حال سه مدل با نرخ یادگیری ۰.۰۲، ۰.۰۰۵ و ۰.۰۰۱ ایجاد می‌کنیم و آموزش می‌دهیم.

۳-۴. نمایش و تحلیل نمایش

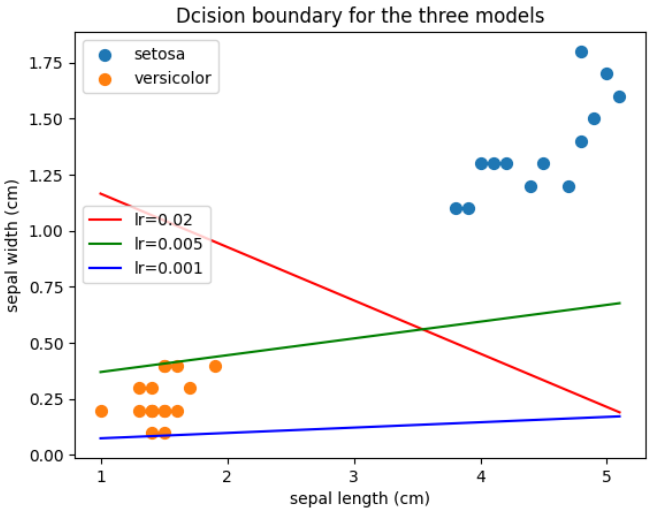
قصد داریم مرز تصمیم‌گیری مدل‌ها را رسم کنیم. می‌دانیم Adaline به صورت خطی تصمیم‌گیری می‌کند. همچنین مرز تصمیم Adaline بدین صورت است که اگر مقدار net پس از گذر از تابع فعال‌سازی، بزرگتر از صفر بود، کلاس یک وگرنه کلاس ۰ را تشخیص می‌دهد. بدین ترتیب مرز تصمیم این مدل برای دو ویژگی و یک خروجی بدین صورت حساب می‌شود:

$$w_1x_1 + w_2x_2 + b = 0 \rightarrow x_2 = \frac{-w_1x_1 - b}{w_2}$$

بدین ترتیب می‌توانیم مرز تصمیم را رسم کنیم. به دلیل خطی بودن مرز تصمیم، دو نقطه ابتدا و انتها از ویژگی اول را در نظر می‌گیریم و با توجه به معادله بالا خط مرز تصمیم را رسم می‌کنیم. حال یک بار نقاط داده آموزش و یک بار هم نقاط داده تست را همراه با مرز تصمیم سه مدل رسم می‌کنیم تا عملکرد مدل‌ها را مشاهده کنیم.

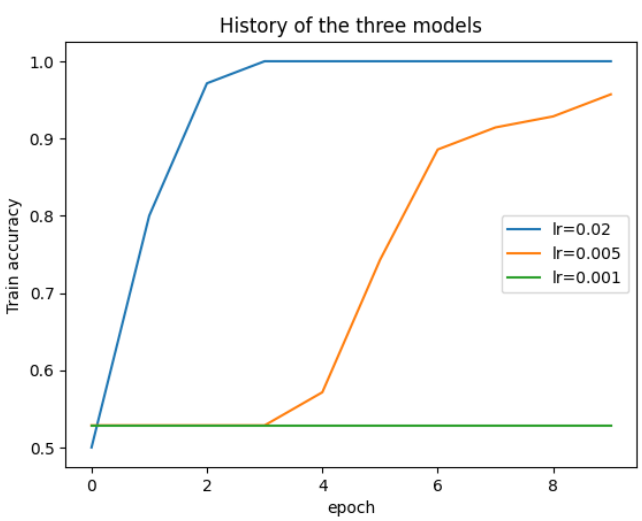


شکل ۴۳. مرز تصمیم سه مدل برای داده آموزش

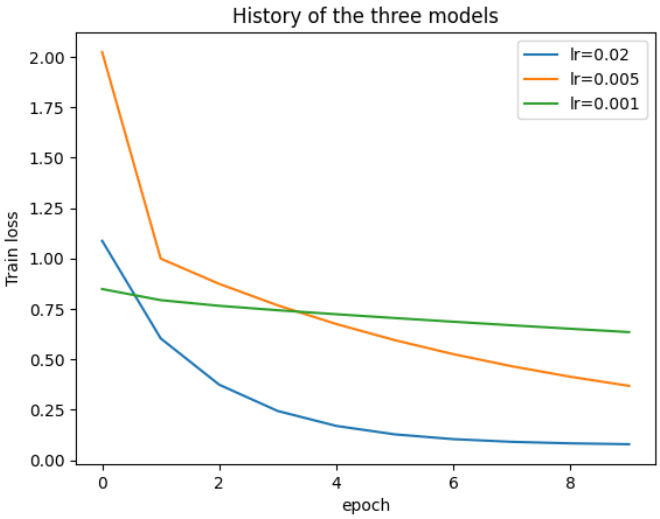


شکل ۴۴. مرز تصمیم سه مدل برای داده تست

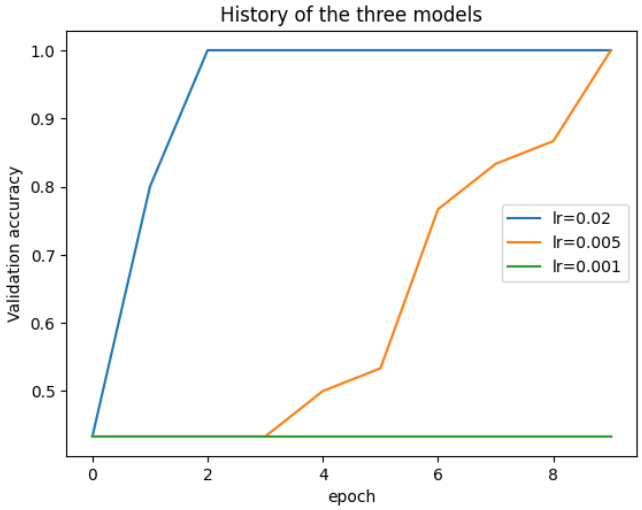
همان طور که در دو نمودار بالا مشخص است، با کاهش نرخ یادگیری مرز تصمیم در جدا کردن دو کلاس ضعیف‌تر عمل می‌کند. حال نمودارهای خطا و صحت برحسب epoch برای دو داده آموزش و تست را برای هر سه مدل رسم می‌کنیم.



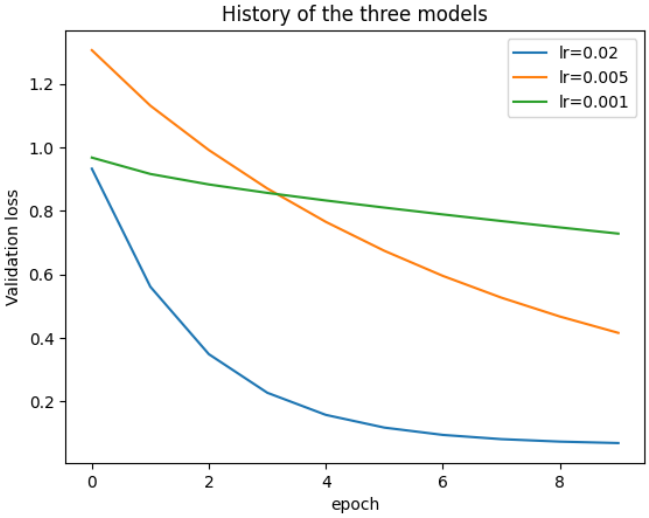
شکل ۴۵. خطای مدل‌ها برحسب epoch برای داده آموزش



شکل ۴۶. صحت مدل‌ها برحسب epoch برای داده آموزش



شکل ۴۷. خطای مدل‌ها برحسب epoch برای داده تست



شکل ۴۸. صحت مدل‌ها برحسب epoch برای داده تست

در نمودارهای تغییرات خطا و صحت و همچنین نمودارهای مرز تصمیم‌گیری، مشخص است که با کاهش نرخ‌یادگیری عملکرد مدل‌ها کاهش میابد. البته همین طور که از نمودارهای تغییرات خطا مشخص است، خطای هر سه مدل در حال کاهش است ولی خطای مدل سوم با سرعت کمتری کاهش میابد. به همین دلیل با ده epoch، مدل‌های دوم و سوم نمی‌توانند همگرا شوند یا به نقاط بهینه محلی همگرا می‌شوند.. البته اگر میزان نرخ یادگیری را بسیار بزرگ دهیم هم ممکن است مدل‌ها هیچ‌گاه همگرا نشوند و ناپایدار شوند.

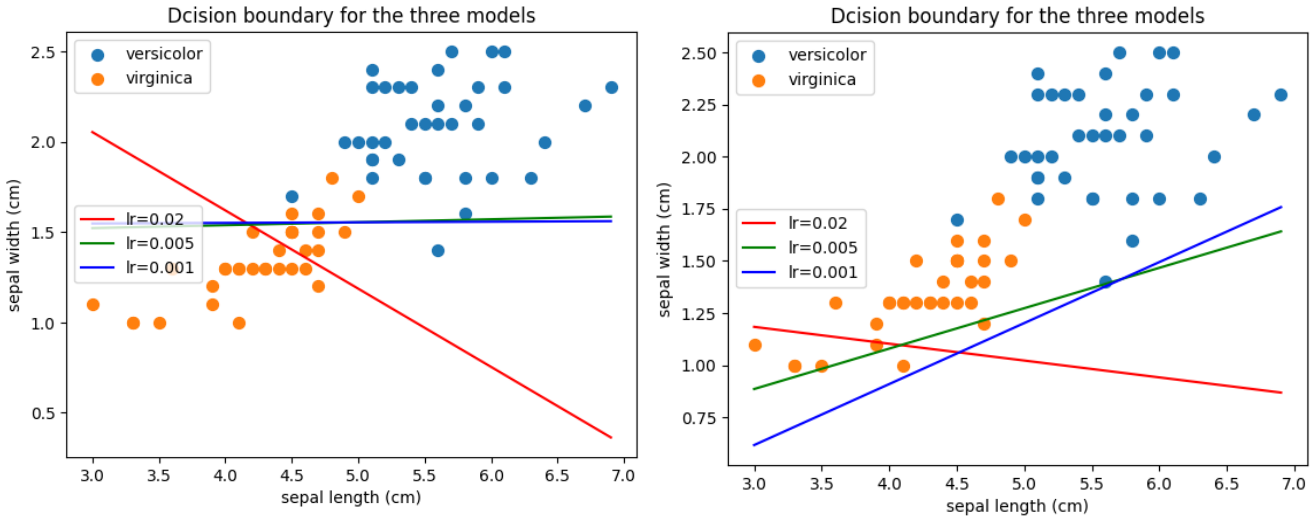
می‌دانیم مدل Adaline یک جداساز خطی است و احتمالا در صورتی که کلاس‌ها به طور خطی قابل جداسازی نباشند، این مدل ضعیف عمل خواهد کرد. برای تست این موضوع، مدل‌مان را روی ویژگی دوم و سوم آموزش خواهیم داد.

نرخ یادگیری با تاثیر گذاشتن در میزان تغییر وزن‌ها، می‌تواند تاثیر زیادی در عملکرد مدل داشته باشد. از طرفی اگر مقدار آن خیلی کم باشد، همگرایی کند می‌شود و به ایپاک‌های بیشتری برای یادگیری نیز است. از طرف دیگر نرخ یادگیری بالا می‌تواند باعث گذر از مقادیر بهینه مطلق شود و باعث ناپایداری و ناهمگرا شدن مدل شود. نرخ یادگیری مناسب، باید تا حد لازم کم باشد تا باعث همگرایی دقیق شود و از طرفی باید به اندازه کافی بزرگ باشد تا زمان یادگیری مناسب باشد. همچنین نرخ یادگیری کم به ایپاک‌های بیشتری نیاز دارد و می‌تواند وزن‌های بهتری دهد در حالی که نرخ یادگیری بزرگ می‌تواند از وزن‌های بهینه عبور کند.

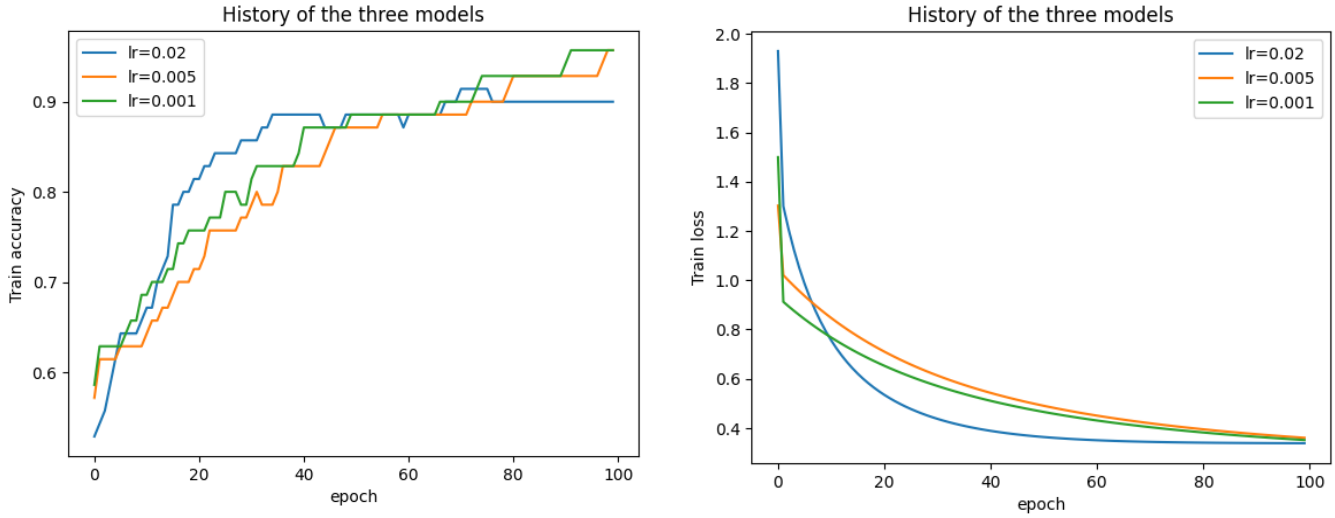
توابع بهینه‌ساز مختلف از روش‌های مختلفی برای تنظیم نرخ یادگیری اعمال استفاده می‌کنند. یکی از روش‌ها این است که ابتدا نرخ یادگیری را بزرگ قرار دهیم و با هر ایپاک نرخ یادگیری را کمتر کنیم تا مدل بتواند در ابتدا در جهت کلی کاهش خطا برود و با کاهش نرخ یادگیری نقاطه بهینه بهتری را بیابد.

۳-۵. بررسی عملکرد مدل‌ها برای کلاس‌های غیرخطی

برای بررسی عملکرد مدل‌ها برای کلاس‌های غیرخطی، پس از بررسی کلاس‌ها متوجه می‌شویم که دو کلاس versicolor و virginica با هم رابطه غیرخطی دارند پس سه مدل قبل را برای این تشخیص این دو کلاس آموزش می‌دهیم.



شکل ۴۹. مرزتصمیم داده‌آموزش کلاس غیرخطی ۱۰ایپاک شکل ۵۰. مرزتصمیم داده‌آموزش کلاس غیرخطی ۱۰۰ایپاک



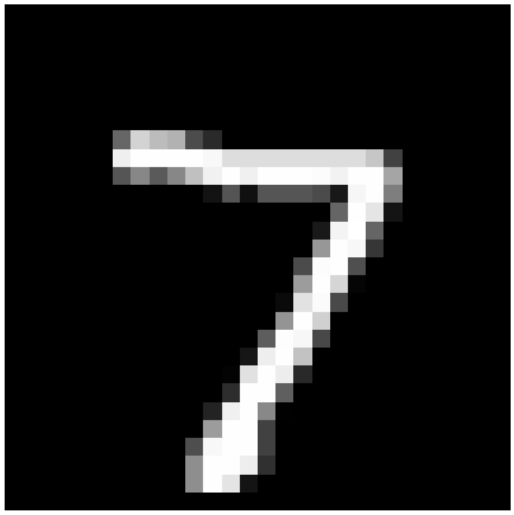
شکل ۵۱. خطا برحسبepoch برای داده آموزش غیرخطیشکل ۵۲. صحت برحسبepoch برای داده آموزش غیرخطی

همان طور که از مرز تصمیم و نمودار تغییرات مشخص است، تمام مدل‌ها عملکرد ضعیفی در ۱۰ ایپاک دارند. پس مدل‌ها را ۱۰۰ ایپاک آموزش می‌دهیم. می‌توان دید که در نهایت تمام مدل‌ها همگرا می‌شوند ولی هیچ یک نمی‌توانند کلاس‌ها را کاملا درست از هم جدا کنند که دلیل اصلی آن غیرخطی بودن کلاس‌ها است. با این‌حال مدل‌ها خطهایی را ایجاد کرده‌اند که سعی کنند کمترین خطا را داشته باشد.

پرسش ۴ – آموزش اتوانکودر و طبقه بندی با دیتاست MNIST

۴-۱. دانلود و پیش پردازش داده ها

دادگان MNIST شامل ۷۰۰۰۰ تصویر دست‌نوشته اعداد ۰ تا ۹ است. هر تصویر سیاه‌وسفید و دارای ابعاد ۲۸ در ۲۸ پیکسل است. ابتدا دادگان را لود می‌کنیم و سپس همان طور که خواسته شده، اعداد تصاویر را به مقادیر بین صفر و یک نرمالایز می‌کنیم و همچنین به آرایه‌هایی یک بعدی با اندازه ۷۸۴ تبدیل می‌کنیم.



شکل ۵۳. یک نمونه از دادگان MNIST

۴-۲. طراحی و پیاده سازی مدل

۴-۲-۱. اتوانکودرها

مدل‌های انکودر و رمزگشا را به صورت خواسته شده تشکیل می‌دهیم و با ترکیب آن‌ها دو مدل اتوانکودر ایجاد می‌کنیم. مدل اول لایه‌هایی با ۴ نورون در انکودر و رمزگشا دارد و مدل دوم به جای آن لایه‌هایی با ۸ نورون دارد. حال هر دو مدل را آموزش می‌دهیم و حواسمان هست که داده خروجی مورد انتظار همان مقادیر ورودی است.

۴-۲-۲. طبقه‌بندی با انکودر

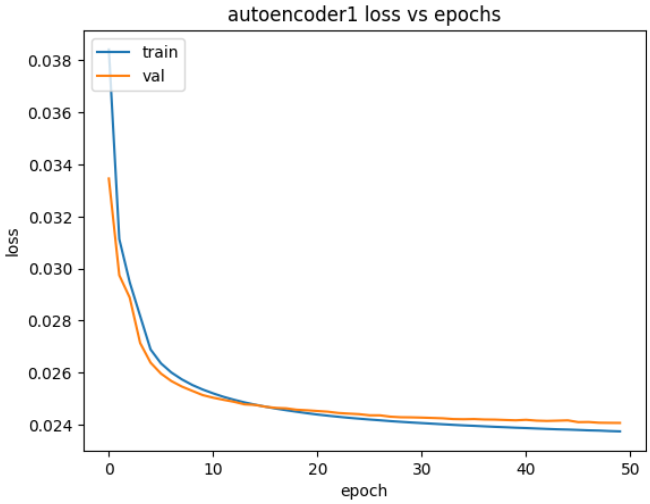
حال دو شبکه fully connected ایجاد می‌کنیم. شبکه اول ورودی ۸ تایی دریافت می‌کند و سپس یک لایه با ۴ نورون دارد ولی شبکه دوم مستقیم ورودی ۴ تایی دریافت می‌کند و لایه با ۴ نورون را ندارد. خر یک از انکودرهای قسمت قبل را فریز می‌کنیم و با ترکیب انکودر اول با شبکه fully connected اول،

طبقه‌بند اول و با ترکیب انکودر دوم با شبکه fully connected دوم طبقه‌بند دوم را تشکیل می‌دهیم. حال این دو طبقه‌بند را آموزش می‌دهیم البته این بار از برچسب‌های داده شده برای آموزش مدل استفاده می‌کنیم. همچنین در صورت پروژه خواسته شده که در شبکه‌های fully connected لایه خروجی شامل ۲ نورون باشد ولی از آن جاییکه برچسب‌ها شامل اعداد ۰ تا ۹ است، منطقی است که در لایه خروجی هم ۱۰ نورون قرار بگیرد بدین ترتیب در لایه‌های خروجی ۱۰ نورون قرار دادیم.

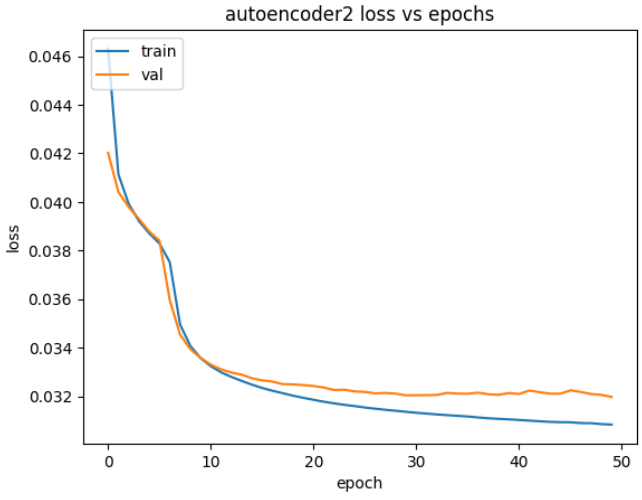
۵-۳. نتایج و تحلیل

۵-۳-۱. تغییرات خطا و صحت مدل حین آموزش

ابتدا نمودارهای مربوط به تغییرات خطای اتوانکودرها حین آموزش را برای دادگان آموزش و تست بررسی می‌کنیم.



شکل ۵۴. خطا برحسب ایپاک برای اتوانکودر اول

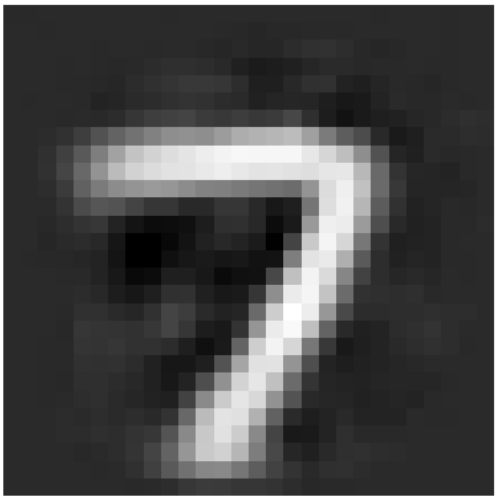


شکل ۵۵. خطا برحسب ایپاک برای اتوانکودر دوم

پس از اتمام آموزش خطای mse مدل اول روی داده تست برابر ۰.۰۲۴ و خطای مدل دوم برابر ۰.۰۳۲ شده است. با اینکه مدل اول تعداد پارامترهای بیشتری دارد، امکان بیش‌برازش در آن بیشتر است ولی در نهایت خطای آن کمتر از مدل دوم شده است که این می‌تواند نشان دهنده این موضوع باشد که ۴ نورون برای فشرده‌سازی تصاویر بسیار کم بوده و باعث از دست رفتن اطلاعات می‌شود و به طور می‌توانیم انتظار داشته باشیم با کاهش اندازه لایه پنهان کیفیت تصاویر بازتولید شده نیز افت پیدا کند.

۵-۳-۲. تصاویر بازتولید شده توسط اتوانکودرها

حال نمونه مشاهده شده از داده در شکل ۵۳ را که توسط دو اتوانکودر بازتولید شده است را در صفحه بعد مشاهده می‌کنیم.



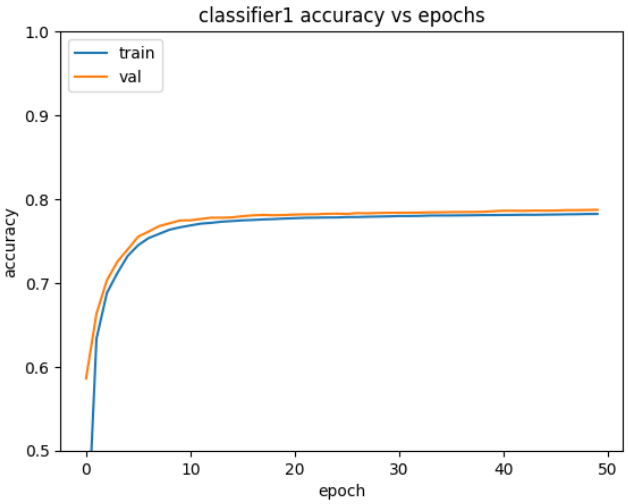
شکل ۵۶. تصویر بازتولید شده توسط اتوانکودر اول

شکل ۵۷. تصویر بازتولید شده توسط اتوانکودر دوم

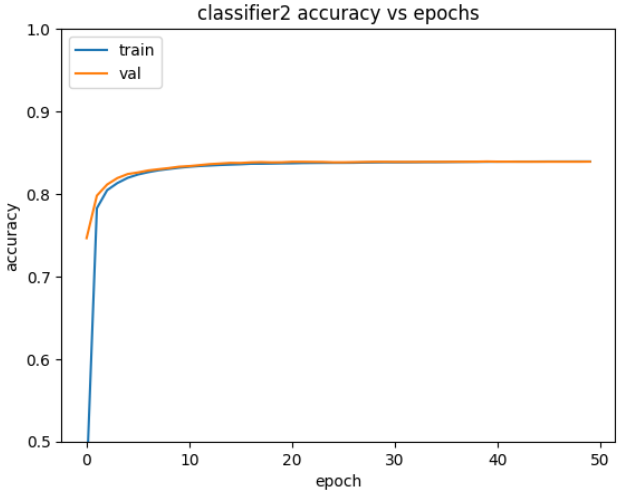
تفاوت بین تصاویر بازتولید شده توسط دو مدل با تصویر اول و با هم قابل رویت است. به نظر می‌رسد مدل اول تصویر تارتری تولید کرده با این‌حال ویژگی‌های اصلی عدد هفت در آن قابل تشخیص و نمایان است در حالیکه مدل دوم تصویر شارپ‌تر ولی با نویز بیشتر تولید کرده است.

۵-۳-۳. عملکرد طبقه‌بندها

حال به بررسی تغییرات صحت مدل‌های طبقه‌بند حین آموزش می‌پردازیم.



شکل ۵۸. صحت برحسب ایپاک برای طبقه‌بند اول



شکل ۵۹. صحت برحسب ایپاک برای طبقه‌بند دوم

با توجه به نمودارهای رسم شده، به نظر می‌رسد صحت هر دو مدل به طور یکنواخت افزایش یافته و در نهایت هم به مقادیر نزدیک به هم همگرا شده‌اند. در نهایت صحت مدل اول روی دادگان تست ۰.۰۷۸۷۱ و صحت مدل دوم روی دادگان تست ۰.۸۳۹۱ شده است. با اینکه اتوانکودر اول در بازتولید تصاویر خطای کمتری داشت ولی می‌بینیم که در طبقه‌بندی باعث کاهش عملکرد شده است. یکی از دلایل می‌تواند

افزایش تعداد پارامترهای مدل باشد. با افزایش تعداد پارامترها، سرعت آموزش کندتر و از طرف احتمال رخ دادن بیش‌برازش بیشتر می‌شود. در ۵۰ ایپاک نخست در نمودارهای بالا رخ دادن بیش‌برازش مشاهده نمی‌شود ولی انتظار می‌رود با افزایش ایپاک‌ها بیش‌برازش رخ دهد. البته افزایش پارامترهای مدل همیشه بد نیست و تا حدی لازم است ولی افزایش بیش از حد آن باعث افت عملکرد می‌شود.

از روش‌های کاهش بیش‌برازش می‌توان به کاهش تعداد پارامترها، اضافه کردن دراپ اوت، توقف زودهنگام آموزش مدل، اضافه کردن نویز به داده‌ها، منظم‌سازی وزن‌ها اشاره کرد.

۵-۳-۴. تعداد پارامترها

حال نتایج مدل‌ها را مشاهده و مقایسه می‌کنیم. حال مدل‌ها را از نظر تعداد پارامتر مقایسه می‌کنیم.

جدول ۱۱. خلاصه پارامترهای مدل اتوانکودر اول

Layer (type)	Output Shape	Param #
Encoder	(None, 8)	101,512
Decoder	(None, 784)	102,288

Total params: 203,800 (796.09 KB)

Trainable params: 203,800 (796.09 KB)

Non-trainable params: 0 (0.00 B)

جدول ۱۲. خلاصه پارامترهای مدل اتوانکودر دوم

Layer (type)	Output Shape	Param #
Encoder	(None, 4)	100,996
Decoder	(None, 784)	101,776

Total params: 202,772 (792.08 KB)

Trainable params: 202,772 (792.08 KB)

Non-trainable params: 0 (0.00 B)

جدول ۱۳. خلاصه پارامترهای مدل طبقه‌بند اول

Layer (type)	Output Shape	Param #
Encoder	(None, 4)	101,512
FeedForward	(None, 10)	86

Total params: 101,598 (396.87 KB)

Trainable params: 86 (344.00 B)

Non-trainable params: 101,512 (396.53 KB)

جدول ۱۴. خلاصه پارامترهای مدل طبقه‌بند دوم

Layer (type)	Output Shape	Param #
Encoder	(None, 4)	100,996
FeedForward	(None, 10)	50

Total params: 101,046 (394.71 KB)

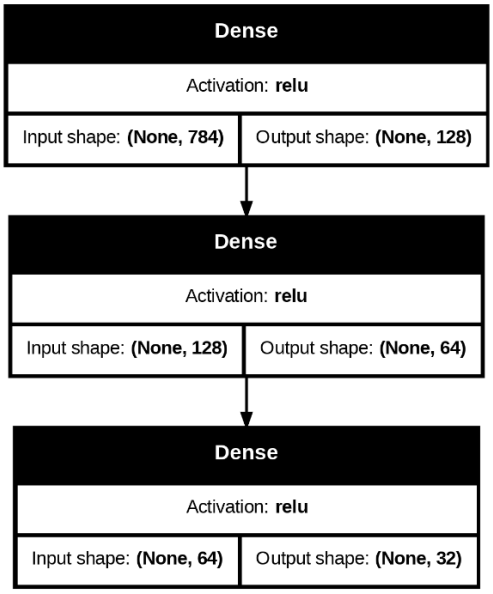
Trainable params: 50 (200.00 B)

Non-trainable params: 100,996 (394.52 KB)

البته تعداد پارامترهای دو مدل تفاوت زیادی ندارند ولی با این‌حال در معیار صحت تفاوت‌هایی را مشاهده کردیم. یکی از دلایل دیگر که می‌تواند تفاوت در عملکرد مدل‌ها را توجیه کند این است که با کاهش ابعاد، مدل مجبور می‌شود همبستگی بین ویژگی‌ها را از بین ببرد و تا حد امکان بدون از دست دادن اطلاعات مفید، اطلاعاتی که مهم نیستند را حذف کند و بدین ترتیب ویژگی‌های با کیفیت‌تری به دست می‌آید که می‌تواند به طبقه‌بندی کمک کند. احتمالا با تغییر در معماری بتوانیم حتی با افزایش تعداد پارامترها، به مدل‌های بهتری برسیم که در قسمت بعد بررسی خواهیم کرد.

۵-۳-۵. بهبود عملکرد(امتیازی)

ایده اصلی ما برای این بهبود عملکرد بدین صورت است که از طرفی با افزایش پارامترها توانایی مدل را افزایش دهیم و از طرف دیگر، با افزایش تعداد لایه‌ها و پخش کردن متوازن تعداد نورون‌ها فرصت یادگیری آسان‌تری برای مدل فراهم کنیم. بدین ترتیب معماری انکودر که طراحی کردیم به شکل زیر است. همچنین دیکودر نیز معماری مشابه دارد.

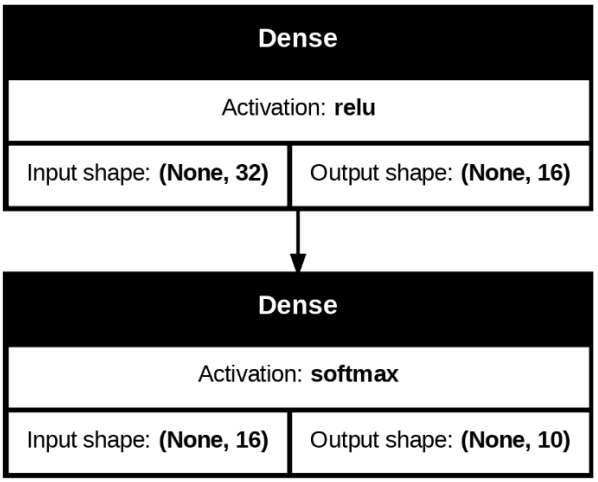


شکل ۶۰. معماری انکودر

خطای مدل اتوانکودر روی دادگان تست برابر ۰.۰۱۴۵ می‌شود که از خطای دو اتوانکودر دیگر کمتر است.

شکل ۶۱. تصویر بازتولید شده توسط اتوانکودر سوم

پس از فریز کردن وزن‌های انکودر، به آن شبکه با معماری زیر را اضافه می‌کنیم.



شکل ۶۲. معماری شبکه feed forward طبقه‌بند سوم

بدین ترتیب کل طبقه‌بند مانند یک انکودر عمل می‌کند چرا که تعداد نورون‌های آن در هر لایه کاهش میابد.

جدول ۱۵. خلاصه پارامترهای مدل طبقه‌بند سوم

Layer (type)	Output Shape	Param #
Encoder	(None, 32)	110,816
FeedForward	(None, 10)	698

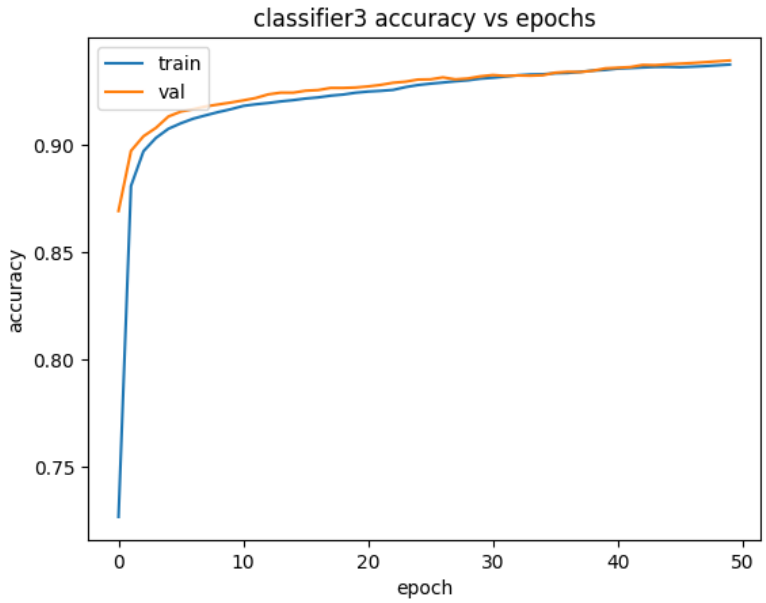
Total params: 112,912 (441.07 KB)

Trainable params: 698 (2.73 KB)

Non-trainable params: 110,816 (432.88 KB)

Optimizer params: 1,398 (5.46 KB)

طبقه‌بند ایجاد شده پارامترهای بیشتری نسبت به دو طبقه‌بند دیگر دارد ولی به دلیل لایه‌های بیشتر و تعداد نورون‌های مناسب عملکرد بهتری دارد.



شکل ۶۳. صحت برحسب ایپاک برای طبقه‌بند سوم

این مدل روی دادگان تست صحت 0.9393 دارد که بیشتر از ده درصد از دو طبقه‌بند دیگر عملکرد بهتری دارد.