به نام خدا

دانشگاه تهران

دانشکده مهندسی برق و کامپیوتر

شبکه های عصبی و یادگیری عمیق

تمرین سری دوم

نام و نام خانوادگی	محمدحسین بدیعی
شماره دانشجویی	810199106
تاریخ ارسال گزارش	1 آذر 1400

فهرست گزارش سوالات

سوال 1 – شبکه پرسپترون چندلایه در کاربرد طبقه‌بندی

بخش الف)

در ابتدا پس از استخراجِ داده‌ها از فایل ionosphere.data با استفاده از دستور open، اقدام به تقسیمِ داده‌های آموزش، تست و ارزیابی می‌نماییم.

این تقسیم بندی به روش‌های گوناگونی که به طور عمده وابسته به تعداد سمپل‌ها از مجموعه داده (number of samples in dataset) و نوع مدل (actual model) است می‌تواند صورت پذیرد که در ادامه شرح می‌دهیم. به عنوان مثال به جهت انتخاب validation set، برخی از مدل‌ها با تعدادِ کمی هایپرپارامتر، طبیعتا به آسانی تیون می‌شوند و لذا در این موارد می‌توانیم سایزِ مجموعه‌ی ارزیاب (validation set) را کاهش دهیم. ولی در مقابل اگر مدلِ انتخابی دارای هایپرپارامترهای زیادی باشند در این‌صورت از مجموعه‌ی ارزیابی با سایزِ بالا استفاده می‌کنند که البته در این دسته از موارد نیز متدِ cross validation به کثرت مورد استفاده قرار می‌گیرد.

در روش‌هایی ابتدا دیتاست را به دو بخشِ تست و آموزش تقسیم می‌کنند که می‌تواند این تقسیم به صورت تصادفی یا با الگویی خاص (مثلا تعداد مشخصی با شروع از اولین داده‌ی موجود در دیتاست) باشد و سپس از مجموعه‌ی داده‌های آموزشِ بدست آمده، قسمتی را به عنوانِ validation و بقیه را به عنوان actual train data در نظر می‌گیرند. اغلب در این روش‌ها نسبت‌های تست و آموزش در ابتدا به ترتیب به صورت 20 درصد و 80 درصد انتخاب می‌شود و سپس از این 80 درصد داده‌های آموزش، 20 درصد را به عنوان مجموعه‌ی validation و مابقی را به عنوان actual train data در نظر می‌گیرند. در روش‌های دیگر نیز در ابتدای کار این تقسیم بندی را انجام می‌دهند. به عنوان مثال یکی از رایج‌ترین تقسیم‌بندی‌ها در این روش 70 درصد داده‌ی آموزش و 15 درصد داده‌ی تست و 15 درصدِ مابقی نیز داده‌های ارزیاب را تشکیل می‌دهد. که این روش‌ها نیز مشابه با روش‌های قبل می‌توانند به صورت رندوم یا مطابق الگویی خاص به انتخابِ داده‌های اموزش و تست و ارزیابی بپردازند. این دو مجموعه روشِ آخر در مواقعی رایج است که از غنایِ کافی در نمونه‌های داده برخوردار هستیم. و لذا برای زمانیکه هایپرپارامترهای کمی داریم، این درصد از validation set نیز کاهش می‌یابد (مطالب ذکر شده در روش‌های قبلی برای نمونه‌های غنی از داده بود). به گونه‌ای که حتی اگر هایپرپارامترهای مسأله بسیار کم (مثلا هیچ یا یکی) بود، می‌توان مجموعه‌ی validation را کنار گذاشت و با نسبتی در حدود 80 درصد داده‌ی آموزش و 20 درصد تست یا 75 درصد آموزش و 25 درصد تست، به تقسیم بندیِ مجموعه‌ی نمونه‌ها (dataset) پرداخت. با این حال باید توجه داشت که بین دادگان آموزش و تست نیز یک trade-off وجود دارد و بدین صورت است که

هرچه میزان داده‌ی آموزش در ساختِ مدل کمتر باشد، شاهد تغییراتِ زیاد در پارامترهای تخمین هستیم و از طرفی اگر تعداد نمونه‌های تست کم باشد نیز شاهد تغییرات زیاد در تخمین عملکردِ شبکه خواهیم بود. لذا موضوعی که این روش‌ها را از یکدیگر متمایز می‌کند، نوع مدل، مجموعه‌ی داده‌گان و تعداد هایپرپارامترهای قابل کنترل است.

در رابطه با مسأله‌ی پیش‌رو با توجه به اینکه تعداد داده‌ها دارای غنای نسبتا کافی می‌باشد و تعداد هایپرپارامترهای مسأله نیز کم نیستند، ما از روش تقسیم‌بندی داده‌ها به صورت 15 درصد داده‌ی تست و 85 درصد داده‌ی آموزش، و نیز با توجه به هایپرپارامترهای مسأله از این 85 درصد نیز 20 درصد را به عنوان validation set و مابقی را به عنوانِ actual training data در نظر می‌گیریم. بدین صورت در این تقسیم‌بندی، 15 درصد مجموعه‌ی داده‌گان تست و 68 درصد مجموعه داده‌گان حقیقی آموزش و 17 درصد نیز validation set خواهند بود.

سپس طبق خواسته‌ی سوال پیش‌پردازش‌های مناسب را اعمال می‌کنیم. پیش‌پردازش‌های در نظر گرفته شده standardization و normalization می‌باشد.

معماری شبکه را طبق خواسته‌ی مسأله با دو لایه و تعداد 15 نورون در لایه‌ی میانی اول و 6 نورون در لایه‌ی میانی دوم و یک نورون خروجی با تابعِ فعالسازِ sigmoid در نظر گرفتیم. با توجه به این معماری به هر دسته‌ی وارد شده متناسب با حرف g یا b به صفر و یک‌هایی نگاشت کردیم تا بتوانیم به عنوانِ خروجی مورد انتظار از شبکه‌ی MLP به مدلِ خود ارائه کنیم. در حالتِ دیفالت توابعِ فعالساز را مورد ارزیابی قرار دادیم و به این نتیجه رسیدیم که برای نورون‌های لایه‌ی میانی اول از تابع فعالساز ریلو و برای نورون‌های لایه‌ی میانی دوم از تابعِ softplus استفاده کنیم. Loss function را hinge در نظر گرفته و برای تعدادِ 220 ایپاک به آموزش مدل پرداختیم.

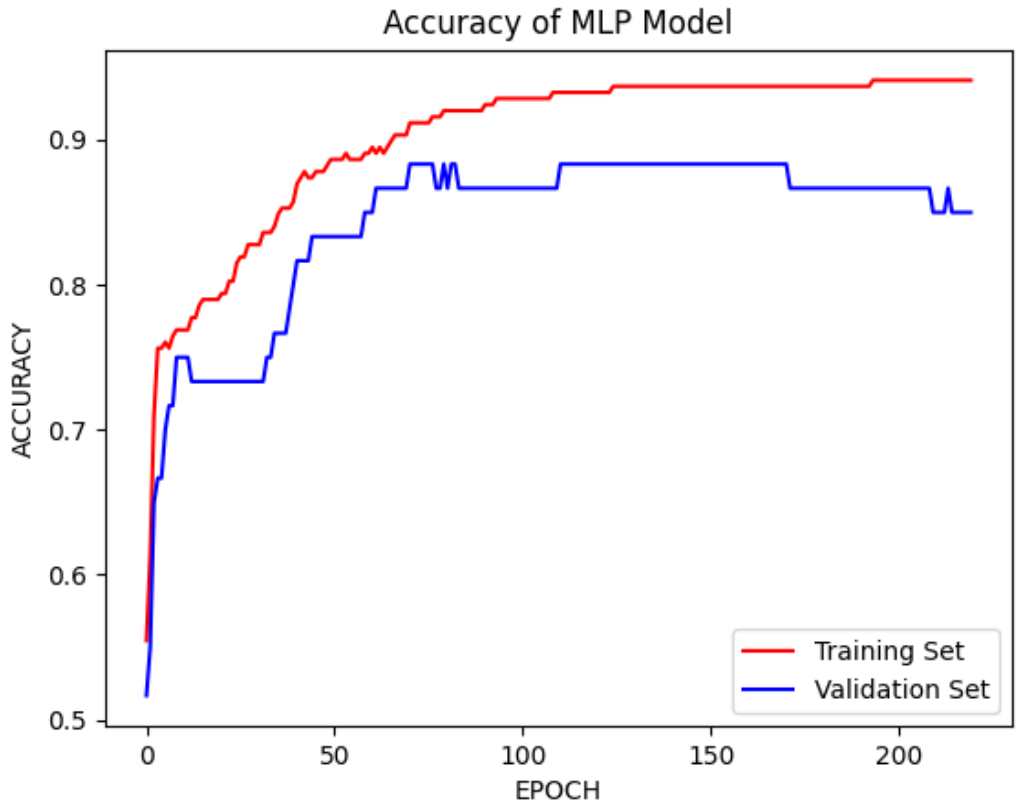
```
15 relu Hidden-Layer-No-1
6 softplus Hidden-Layer-No-2
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
Hidden-Layer-No-1 (Dense)    (None, 15)                525
_________________________________________________________________
Hidden-Layer-No-2 (Dense)    (None, 6)                 96
_________________________________________________________________
Output-Layer (Dense)         (None, 1)                 7
=================================================================
Total params: 628
Trainable params: 628
Non-trainable params: 0
_________________________________________________________________
```

بخش ب)

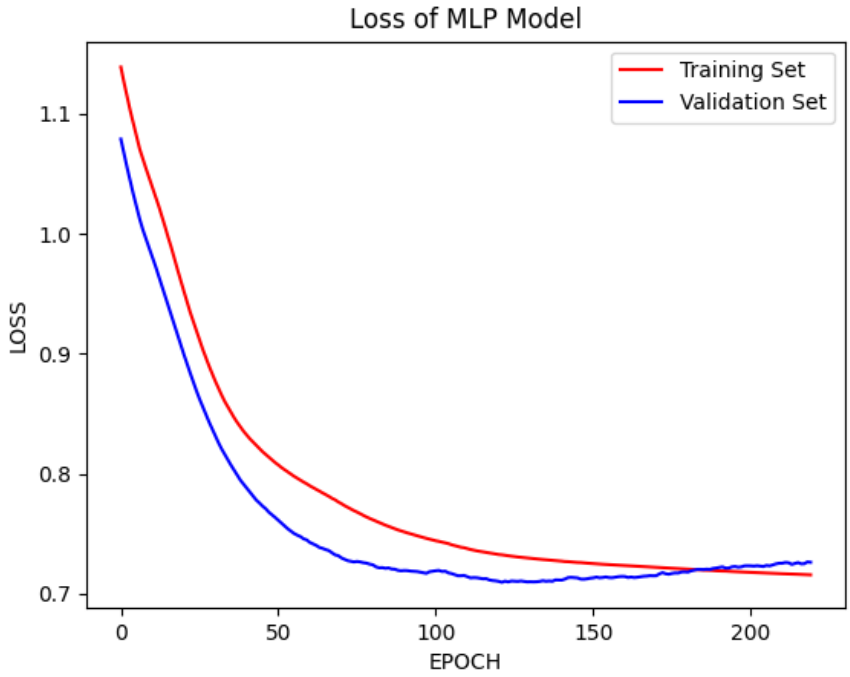
سه حالت در نظر گرفته شده در این بخش با تعداد ایپاک 220 و تابع hinge loss و مشابه با مواردی است که در ساخت مدل در حالتِ الف ذکر کردیم. با این تفاوت که تعداد نورون‌های لایه‌های مخفی را مطابق با خواسته‌ی سوال در سه نوبت تغییر دادیم. نتایج به صورت زیر بدست آمد.

تعداد 6 نورون در لایه‌ی پنهان اول و 2 نورون در لایه‌ی پنهان دوم:

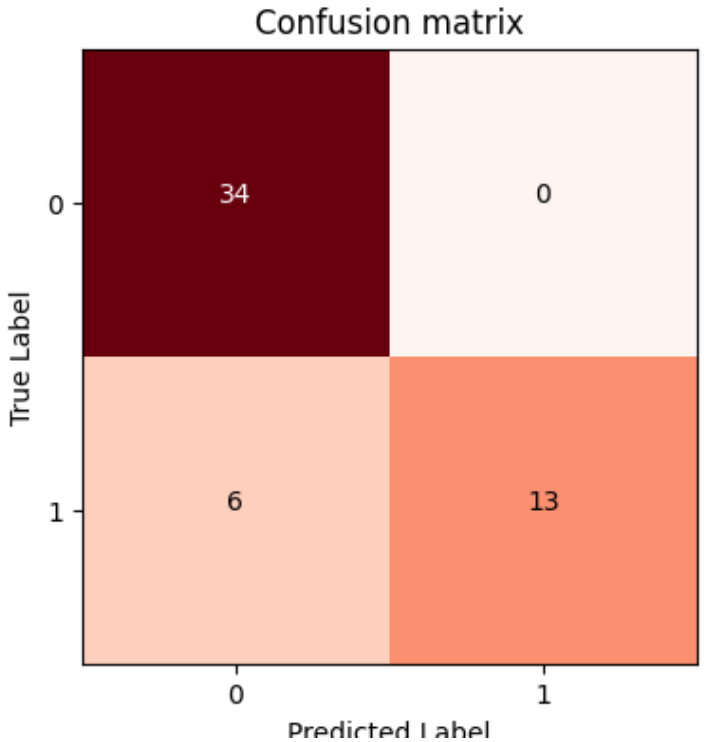
نمودار صحت برای داده‌گان آموزش و ارزیابی به صورت زیر است. مثلا هر چه تعداد ایپاک‌ها افزایش می‌یابد، صحت داده‌گان آموزش افزایش یافته ولی برای مجموعه‌ی ارزیابی، این افزایشِ صحت تا زمانی اتفاق می‌افتد که overfitting رخ نداده ندهد و لذا با افزایشِ بی رویه‌ی تعداد ایپاک‌ها و رخ دادنِ overfitting شاهدِ کاهشِ صحت در مجموعه‌ی ارزیابی خواهیم بود.



برای خطای مدل نیز مشابه با مطالبِ قبلی شاهدِ کاهشِ خطا برای داده‌های آموزش خواهیم بود و نیز شاهدِ کاهش خطای مدل برای مجموعه‌ی ارزیابی البته تا زمانیکه overfitting رخ ندهد خواهیم بود. نتایج خطای مدل برای داده‌گان آموزش و مجموعه‌ی ارزابی نیز به صورت زیر می‌باشد.



خواسته‌ی بعدیِ مسأله ترسیمِ ماتریس آشفتگی Confusion Matrix است که به صورت زیر بدست آوردیم.



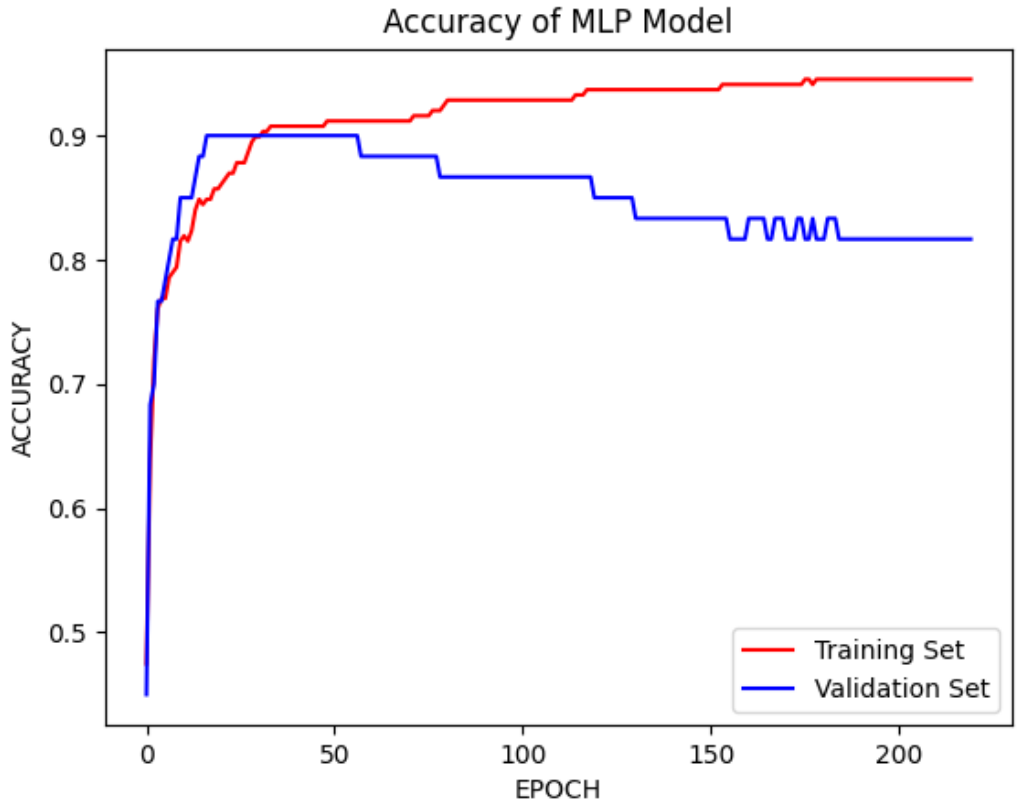
نهایتا خطا و صحتِ مدل بر روی مجموعه‌ی تست نیز به صورت زیر است:

صحت مدل برای داده‌های تست	0.8867924809455872
خطای مدل برای داده‌های تست	0.7539919018745422

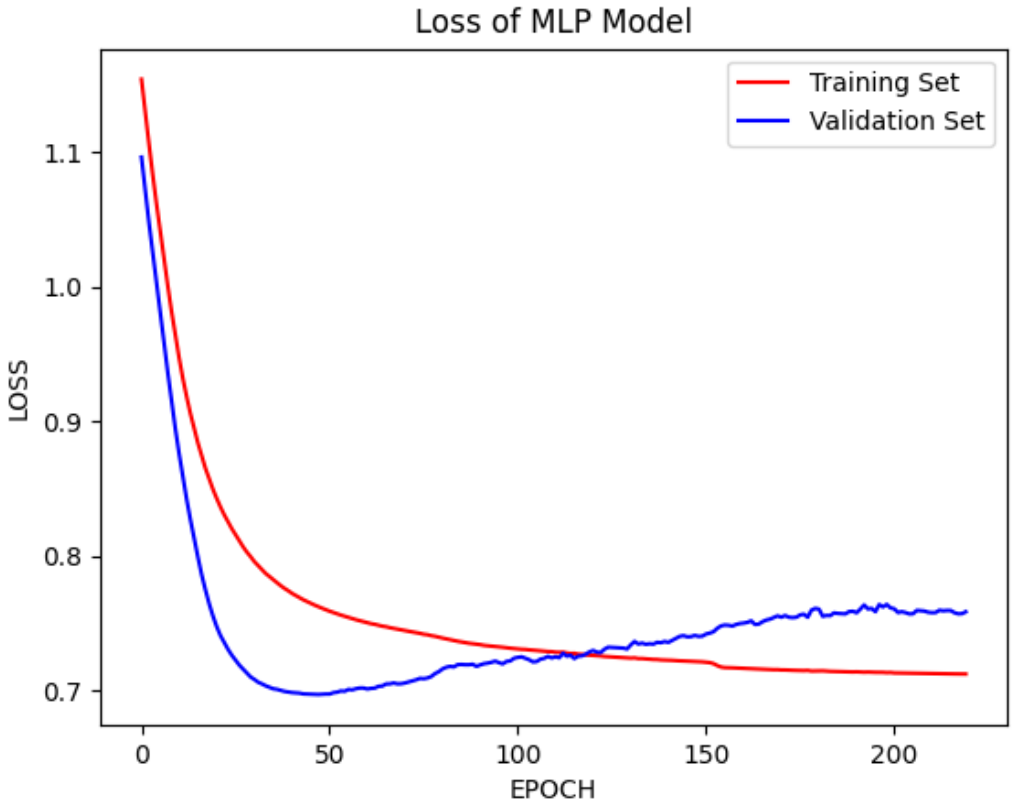
زمان آموزش: **17.25135064125061**

تعداد 15 نورون در لایه‌ی پنهان اول و 6 نورون در لایه‌ی پنهان دوم:

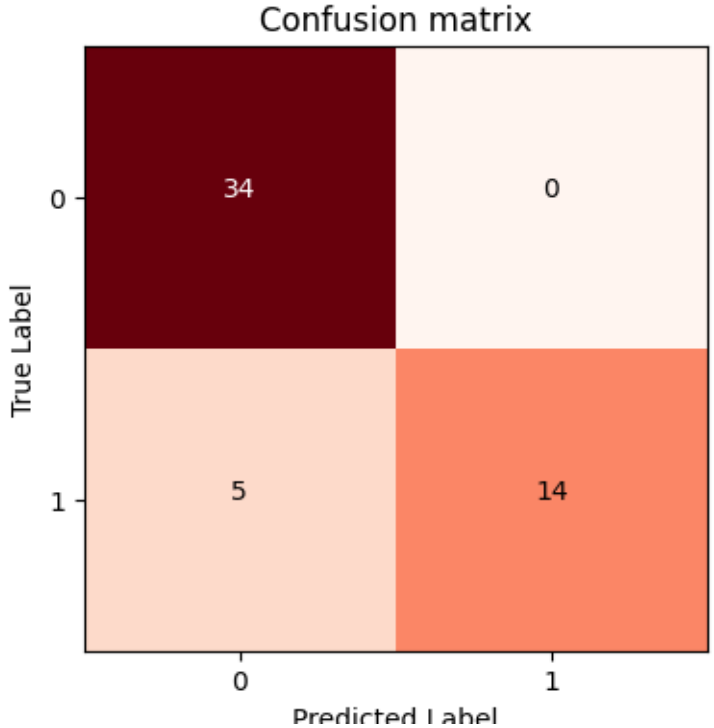
نمودار صحت برای مجموعه‌ی آموزش و ارزیابی به صورت زیر است. عملکرد مشابه با آن مطالبی است که برای مرحله‌ی قبل ذکر نمودیم.



نمودار خطا نیز به صورت زیر بدست آمده است:



ماتریس آشفتگی نیز به ازای این تعداد نورون‌ها به صورت زیر است:



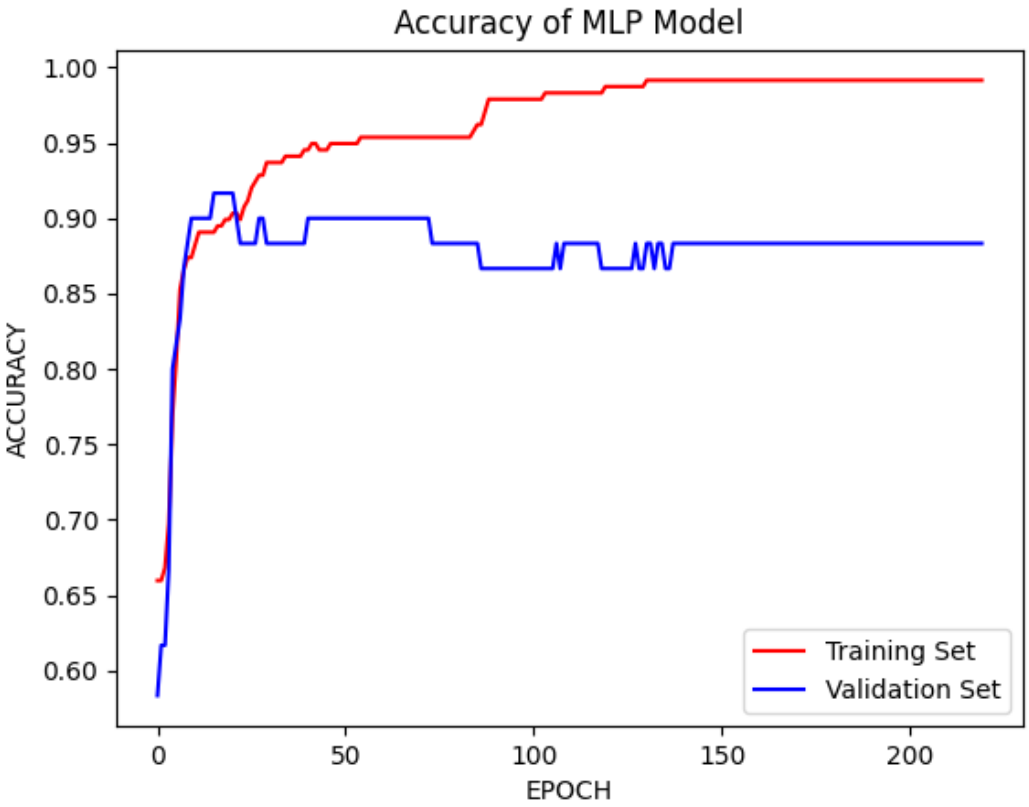
خطا و صحتِ مدل بر روی مجموعه‌ی تست نیز به صورت زیر است :

0.9056603908538818	صحت مدل برای داده‌های تست
0.7397675514221191	خطای مدل برای داده‌های تست

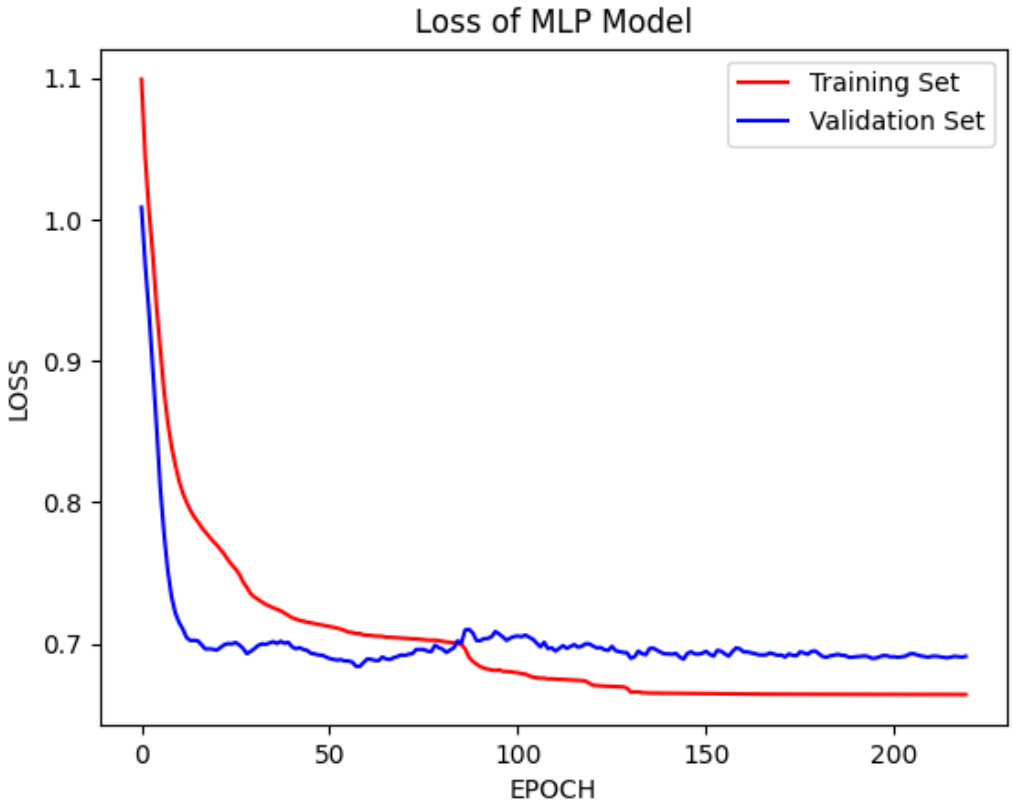
زمان آموزش: **17.36235547065735**

تعداد 55 نورون در لایه‌ی پنهان اول و 40 نورون در لایه‌ی پنهان دوم:

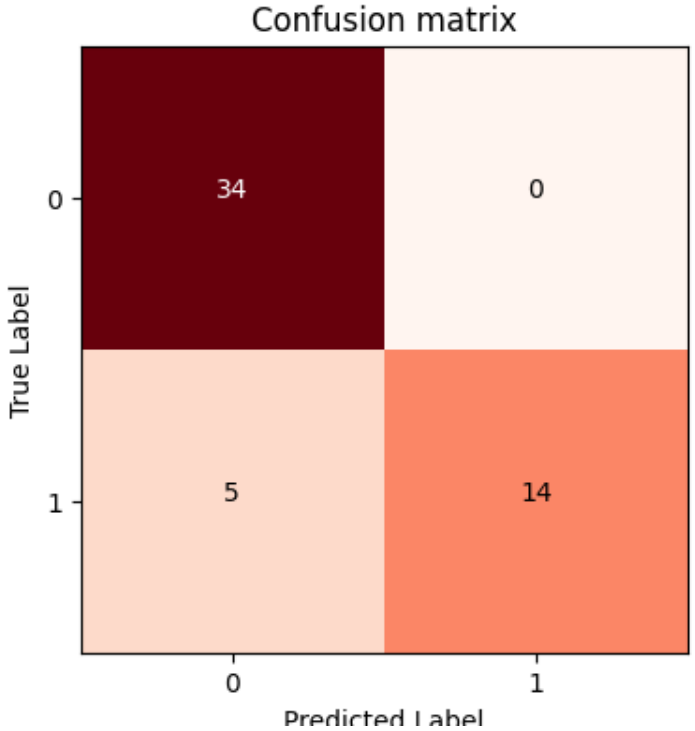
نمودار صحت با استدلالی که برای حالت اولی کردیم و البته با شهودِ بیشتری در اینجا به صورت زیر است.



نمودار خطا نیز به صورت زیر بدست آمده است:



ماتریس آشفتگی نیز به ازای این تعداد نورون‌ها به صورت زیر است:



خطا و صحتِ مدل بر روی مجموعه‌ی تست نیز به صورت زیر است :

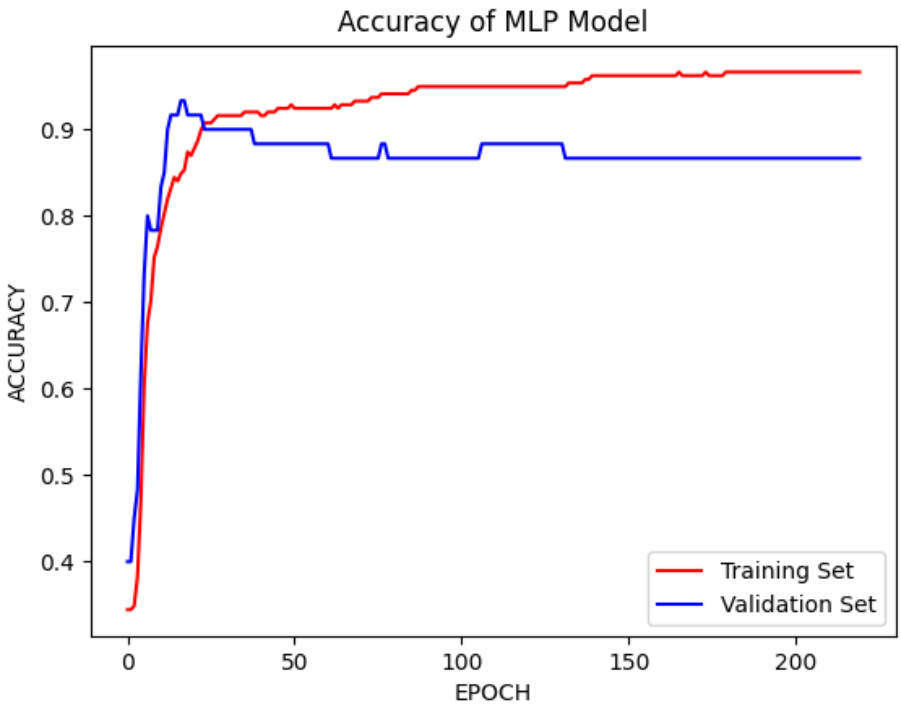
0.9056603908538818	صحت مدل برای داده‌های تست
0.735938310623169	خطای مدل برای داده‌های تست

زمان آموزش: **18.179588079452515**

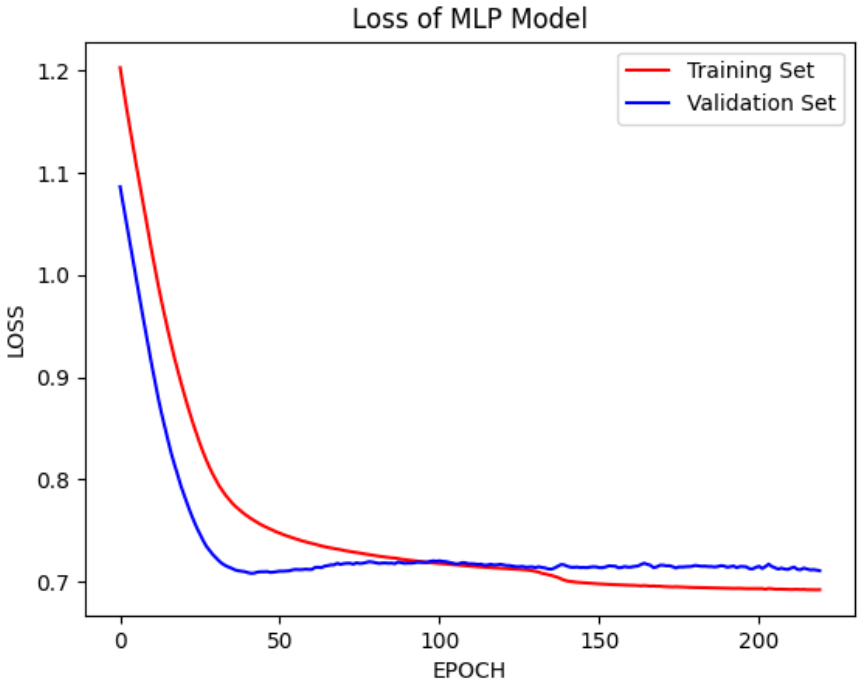
همانطور که مشاهده می نمایید با افزایشِ تعداد نورون‌های لایه‌ی مخفی صحت تست افزایش یافته است که از سویی نیز باید توجه کنیم که با در نظر گرفتنِ تعداد ایپاک ها، شبکه را به سمتِ overfit شدن هدایت نکنیم. از سویی مشاهده می‌کنیم که زمان آموزش نیز با افزایشِ تعداد نورون‌ها افزایش یافته است.

بخش ج)

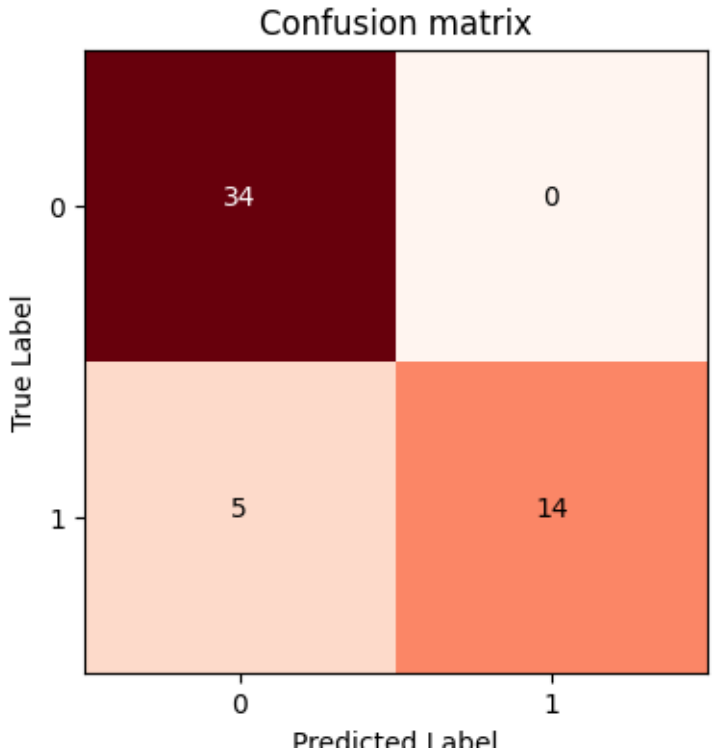
در مرحله‌ی قبل با توجه به دقت و زمانِ آموزش بهترین نتیجه برای 15 نورون در لایه‌ی پنهان اول و 6 عدد در لایه‌ی پنهان دوم بدست آمد. . حالت اول به ازای batch-size=32 نتایج به صورت زیر است.



خطای مدل به ازای batch-size=32 به صورت زیر است.



ماتریس آشفتگی برای داده‌های تست به صورت زیر است:



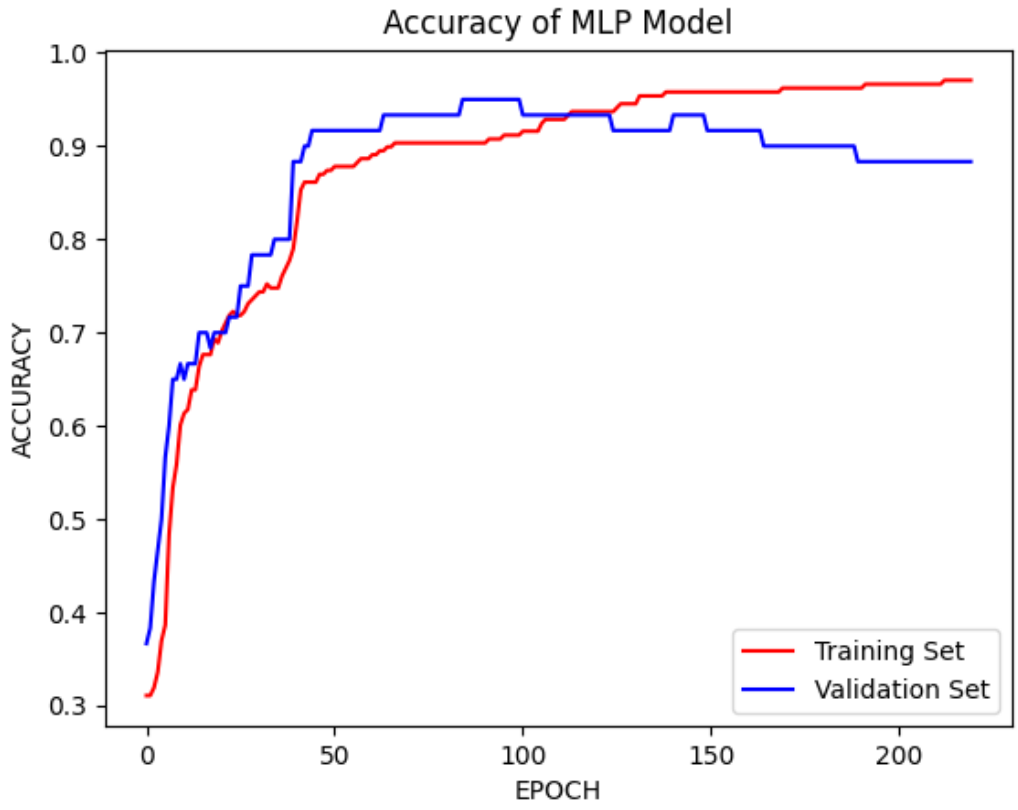
صحت و خطای مدل آموزش دیده برای داده‌های تست به صورت زیر بدست آمد.

صحت مدل برای داده‌های تست	0.9056603908538818
خطای مدل برای داده‌های تست	0.736475944519043

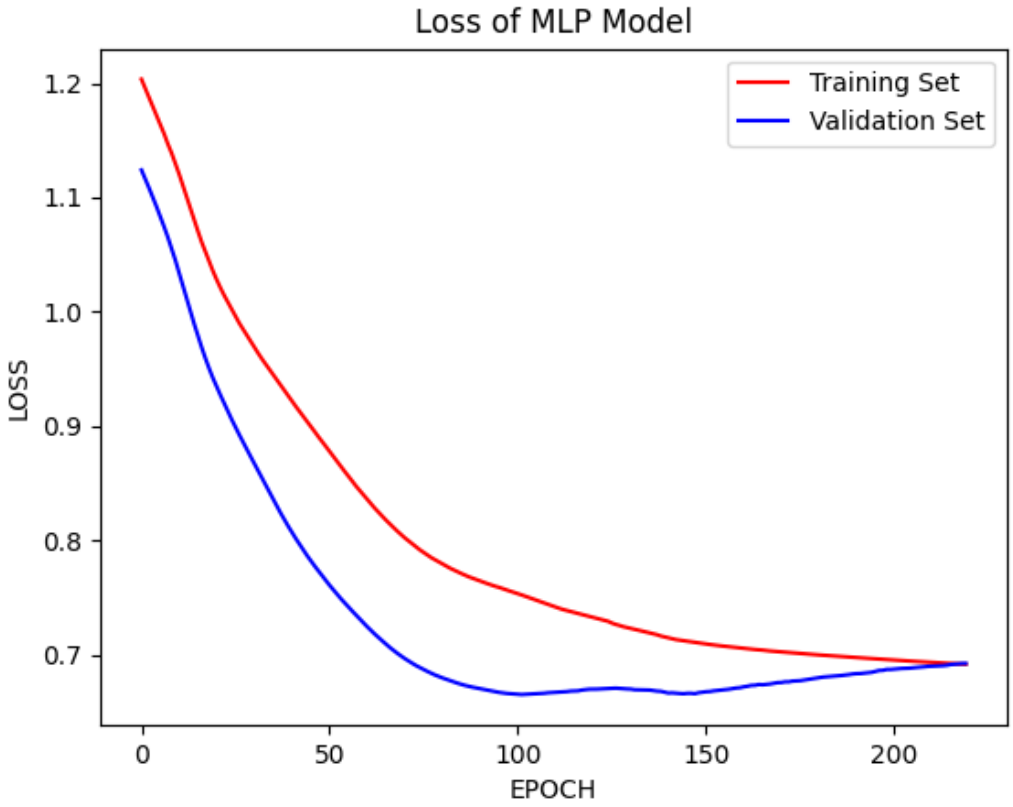
زمان آموزش نیز در این حالت برابر است با:

Time of 1st Hidden Layers
19.899946689605713

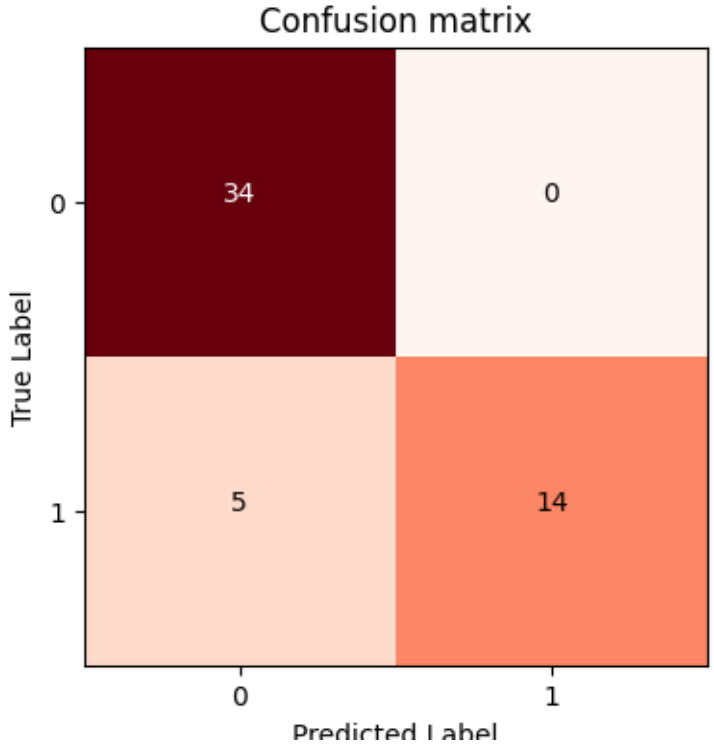
حالت دوم به ازای batch-size=64 نتایج به صورت درآمدند.



خطای مدل به ازای batch-size=64 به صورت زیر است.



ماتریس آشفتگی برای داده‌های تست نیز به صورت زیر است:



صحت و خطای مدل آموزش دیده برای داده‌های تست به صورت زیر بدست آمد.

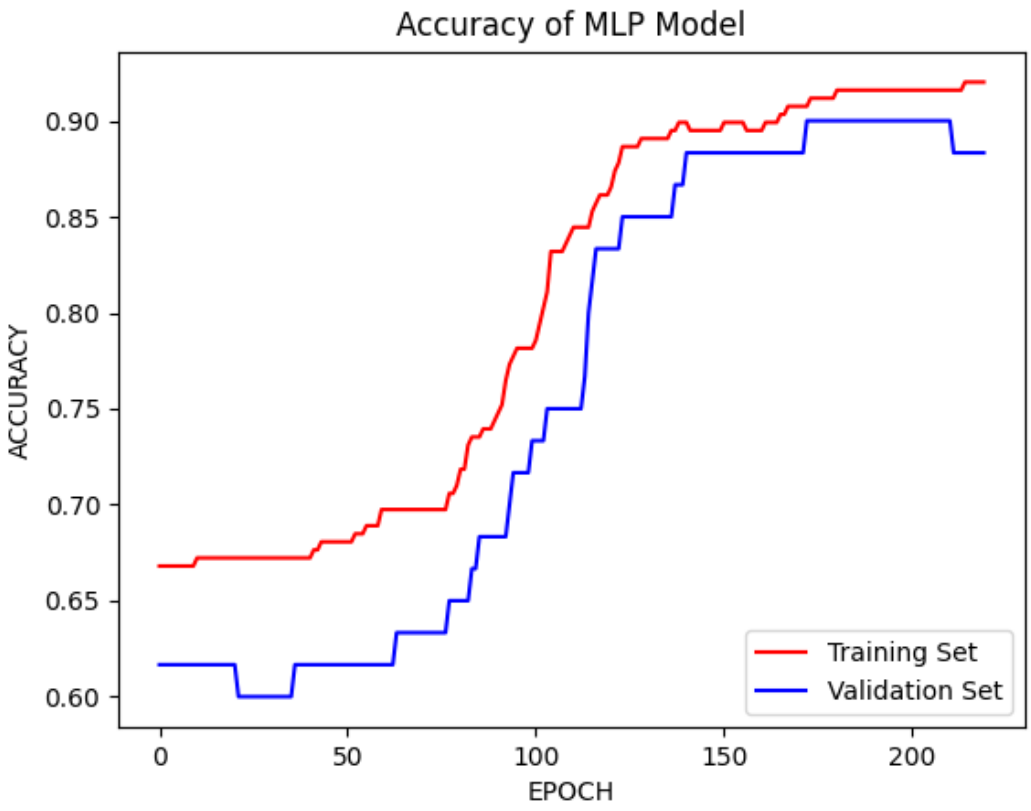
صحت مدل برای داده‌های تست	0.9056603908538818
خطای مدل برای داده‌های تست	0.7436994314193726

زمان آموزش نیز در این حالت برابر است با:

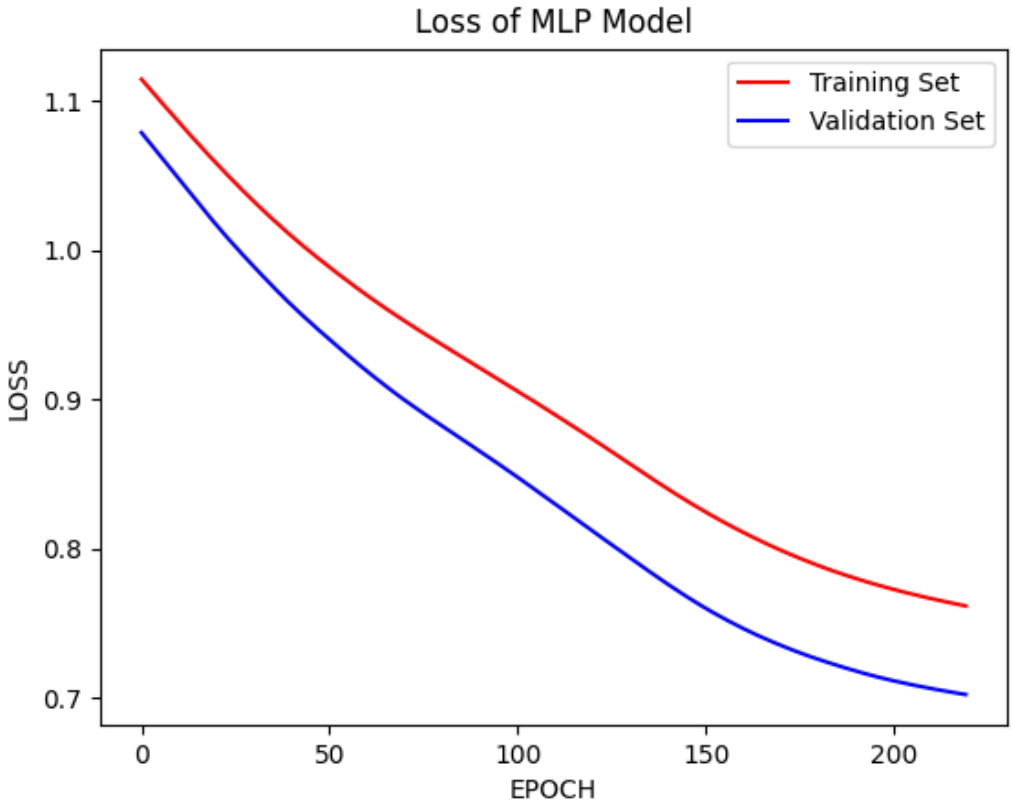
Time of 2st Hidden Layers
16.5401394367218

نتایج در حالت سوم به ازای batch-size=256 به صورت درآمدند.

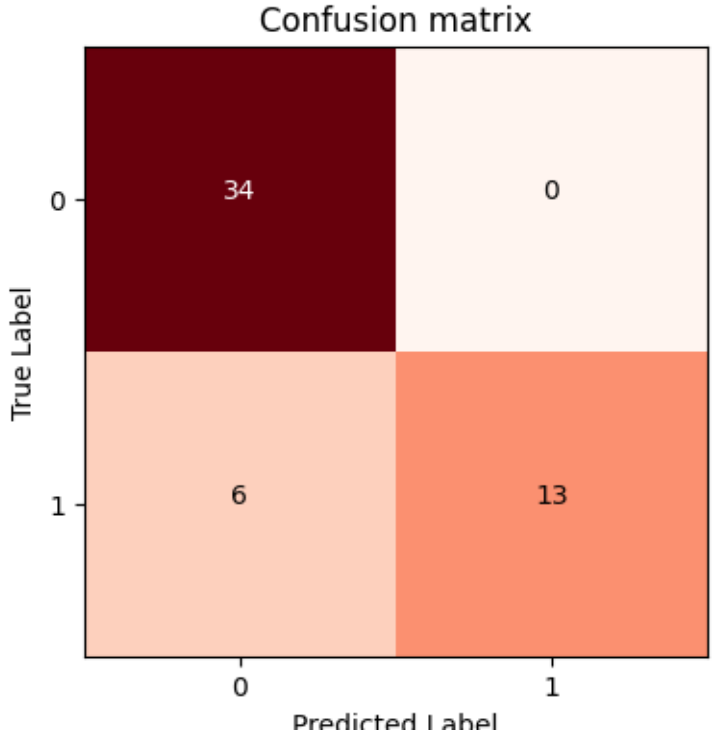
نمودار صحت به صورت زیر است:



خطای مدل به ازای batch-size=256 به صورت زیر است.



ماتریس آشفتگی برای داده‌های تست نیز به صورت زیر است:



صحت و خطای مدل آموزش دیده برای داده‌های تست به صورت زیر بدست آمد.

صحت مدل برای داده‌های تست	0.8867924809455872
خطای مدل برای داده‌های تست	0.7786927819252014

زمان آموزش نیز در این حالت برابر است با:

Time of 3st Hidden Layers
16.301080465316772

همانطور که مشاهده می‌نمایید در این آزمایش صحت برای bathc-size=32 با صحت برای batch-size=64 برابر و بیشتر از حالتی است که batch-size بزرگ و برابر با 256 است. این نتایج کاملا سازگار با انتظار ما از تاثیرِ batch-size است چرا که با افزایشِ اندازه‌ی batch-size اگرچه سرعتِ آموزشِ مدل افزایش یافته (که از نتایج جدول زیر مشخص است) ولی با افزایشِ بیش از حد بدلیلِ گیر افتادن در local min ها، صحت مدل کاهش می‌یابد. در واقع هر چه سایز بچ را افزایش می‌دهیم، مجموعه داده‌گان بیشتری را برای ارائه‌ی گرادیان و نتیجتا کاهشِ خطا ارائه می‌دهیم و اگر این تعداد داده‌گان بسیار زیاد باشد، طبیعتا شبکه با سرعت بالایی به نقطه‌ی مینیمم اطراف خود همگرا شده و مشکلِ گیر کردن در مینیمم‌های محلی local min problem را بوجود می‌آورد. و از طرفی هر چه هم بچ سایز کوچک‌تر انتخاب شود، شبکه برای همگرا شدن، تناوب‌های زیادی را قبل از رسیدن به نقطه‌ی همگرایی (با لحاظ اینکه قبل از رسیدن به این نقطه واگرا نشود) باید طی کند و در واقع مسأله‌ی optimization نه برای منتخبی از دیتاست بلکه برای تعداد کمی داده تعریف می‌شود که چه بسا واگرایی را نیز به همراه بیاورد. لذا نتیجه می‌گیریم که هر اندازه سایز بچ افزایش یابد، زمان آموزش کم می‌شود و برای دقت نیز به نظر می‌رسد بچ سایزِ 64 عدد مناسبی است تا نه مشکلِ گیر کردن در مینیمم‌های محلی را به ازای افزایشِ زیادِ بچ سایز داشته باشیم و نه مشکلِ واگرایی و کثرتِ تناوب و سرعتِ آموزشِ کم به ازای بچ سایزهای کم (مثلا بچ سایزِ 32 و حتی کمتر)

جدول زمان آموزش را به ترتیب از چپ به راست برای 32 و 64 و 256 نیز در کد رسم نمودیم. و مشاهده می‌فرمایید ظبق استدلال‌های فوق و به طور شهودی از جدولِ زیر با افزایشِ سایزِ بچ، زمان آموزش کم می‌شود.

Time of 1st Hidden Layers	Time of 2st Hidden Layers	Time of 3st Hidden Layers
19.899946689605713	16.5401394367218	16.301080465316772

بخش د)

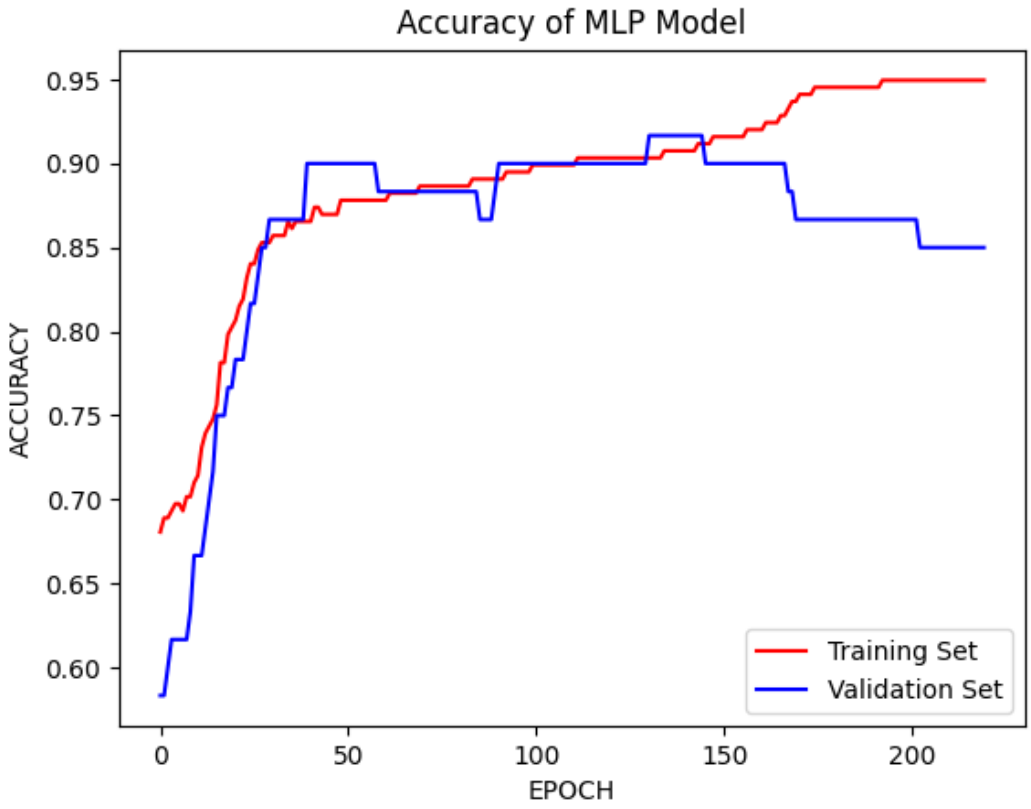
همانطور که استدلال کردیم تا این مرحله بهترین نتایج را برای batch-size=64 و 15 عدد نورون در لایه‌ی پنهان اول و 6 عدد نورون در لایه‌ی پنهان دوم بدست آمد. . حال نتایج سه تابع فعالساز دیگر را بررسی می‌کنیم.

لذا با در نظر داشتنِ activation function ای که در مراحل پیش استفاده کردیم، مجموعا چهار activation function را به صورت زیر بررسی می‌کنیم.

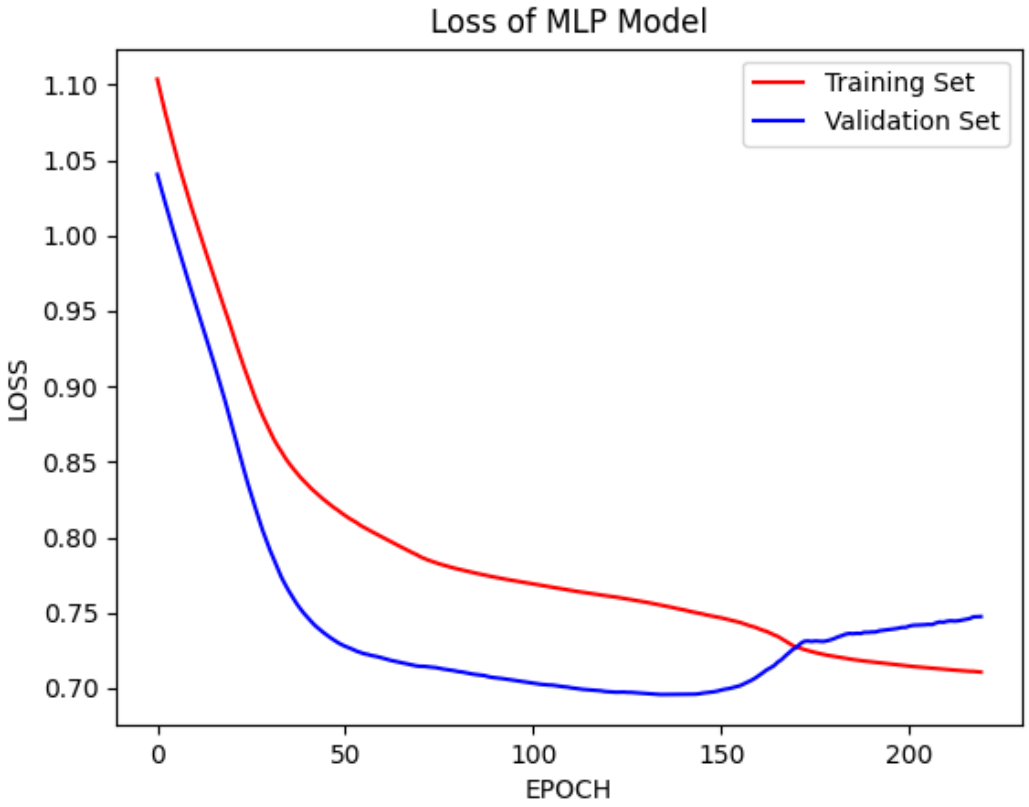
```
secondActivationFuncList = ['softplus','sigmoid', 'relu', 'tanh']
```

نتایج تابع softplus :

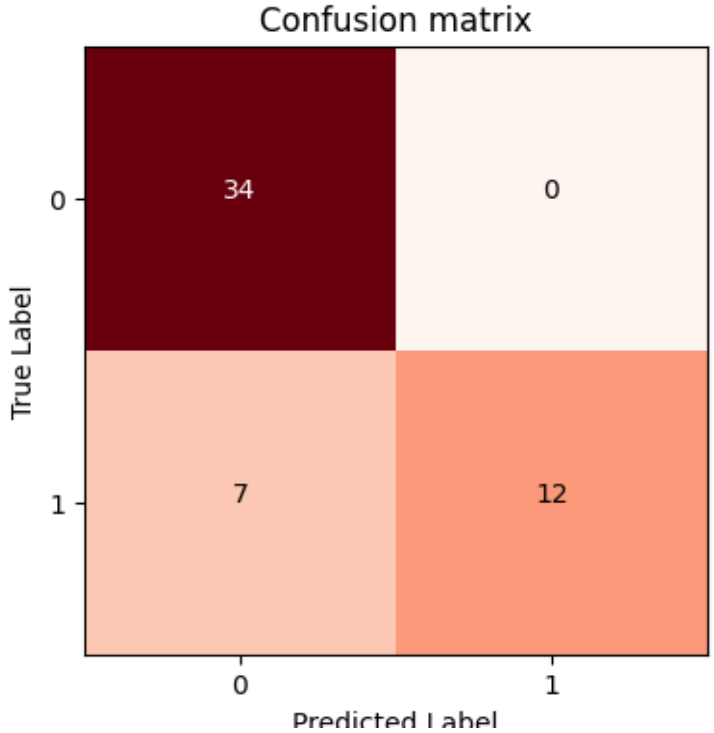
صحت مدل برای داده‌های آموزش و ارزیابی به ازای ایپاک‌های طی شده به صورت زیر است:



نمودار خطای مدل برای مجموعه‌ی آموزش و ارزیابی به صورت زیر است:



و ماتریس آشفتگی نیز به صورت زیر است:

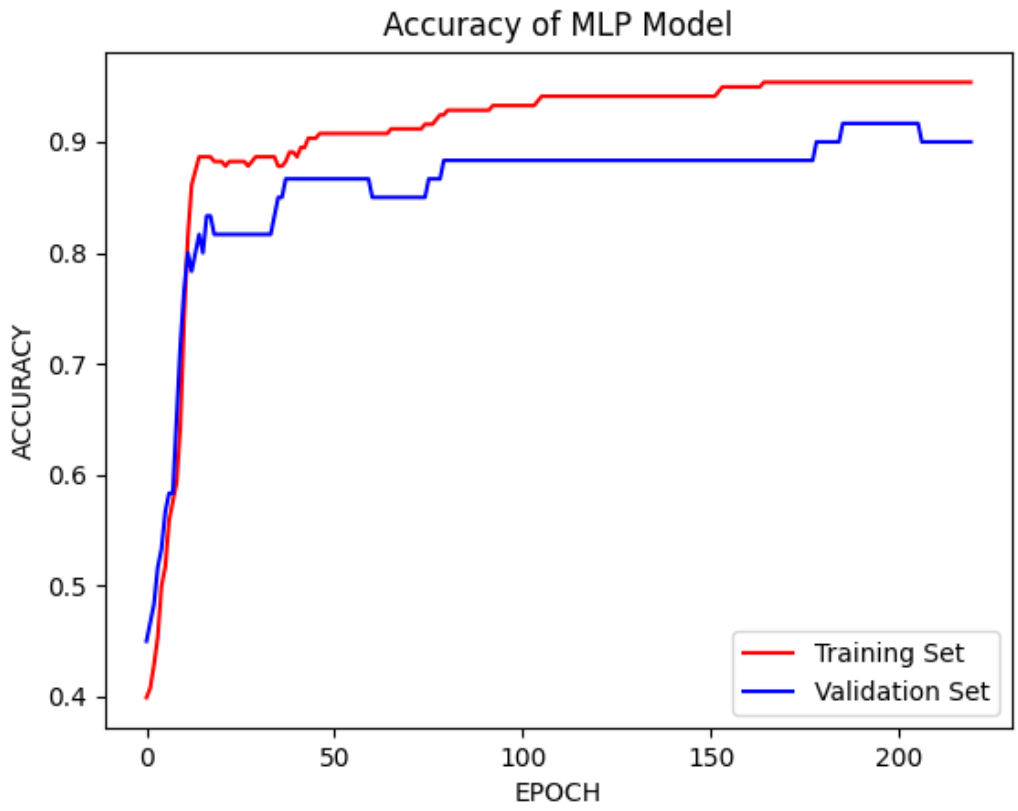


صحت و خطای مدل آموزش دیده برای داده‌های تست به صورت زیر بدست آمد.

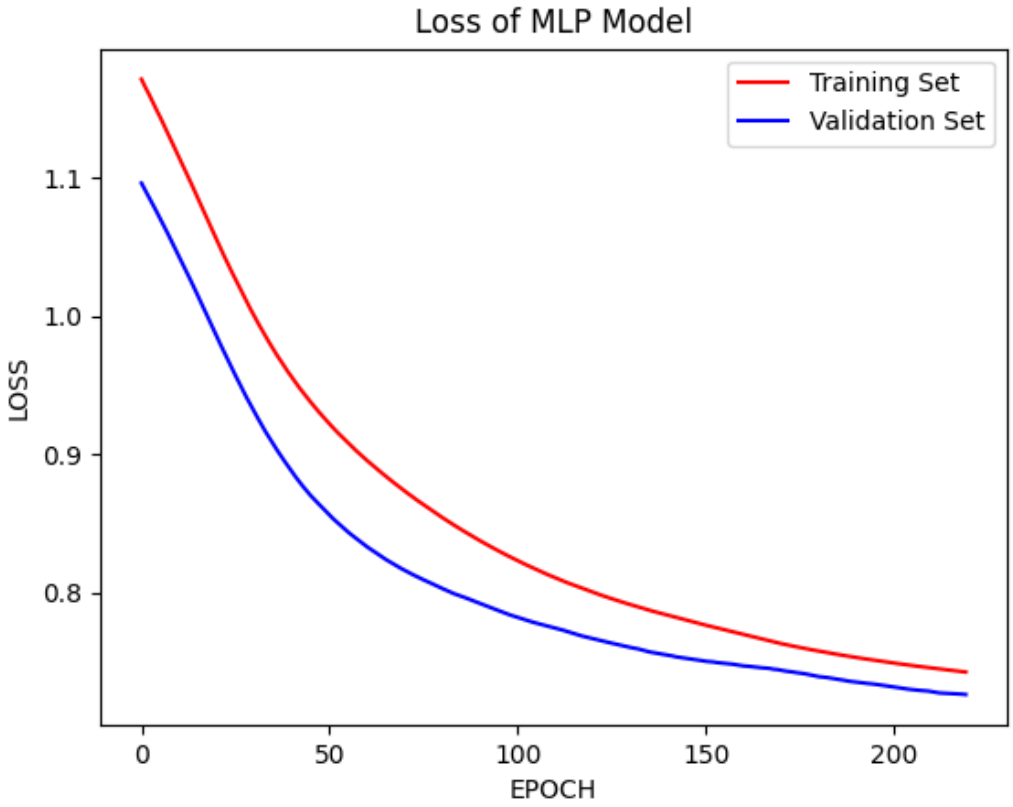
صحت مدل برای داده‌های تست	0.8679245114326477
خطای مدل برای داده‌های تست	0.7786893844604492

نتایج تابع sigmoid :

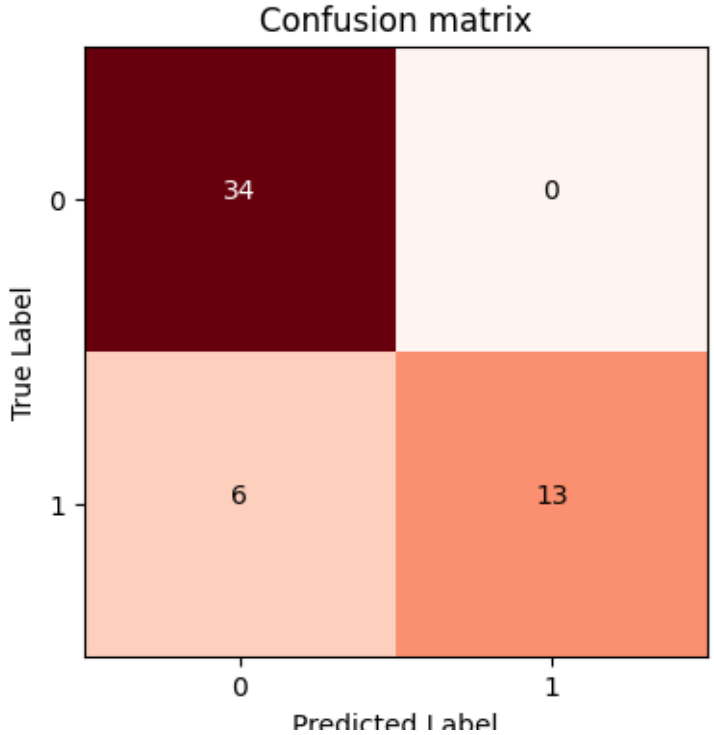
صحت مدل برای داده‌های آموزش و ارزیابی به ازای ایپاک‌های طی شده به صورت زیر است:



نمودار خطای مدل برای مجموعه‌ی آموزش و ارزیابی به صورت زیر است:



و ماتریس آشفتگی نیز به صورت زیر است:

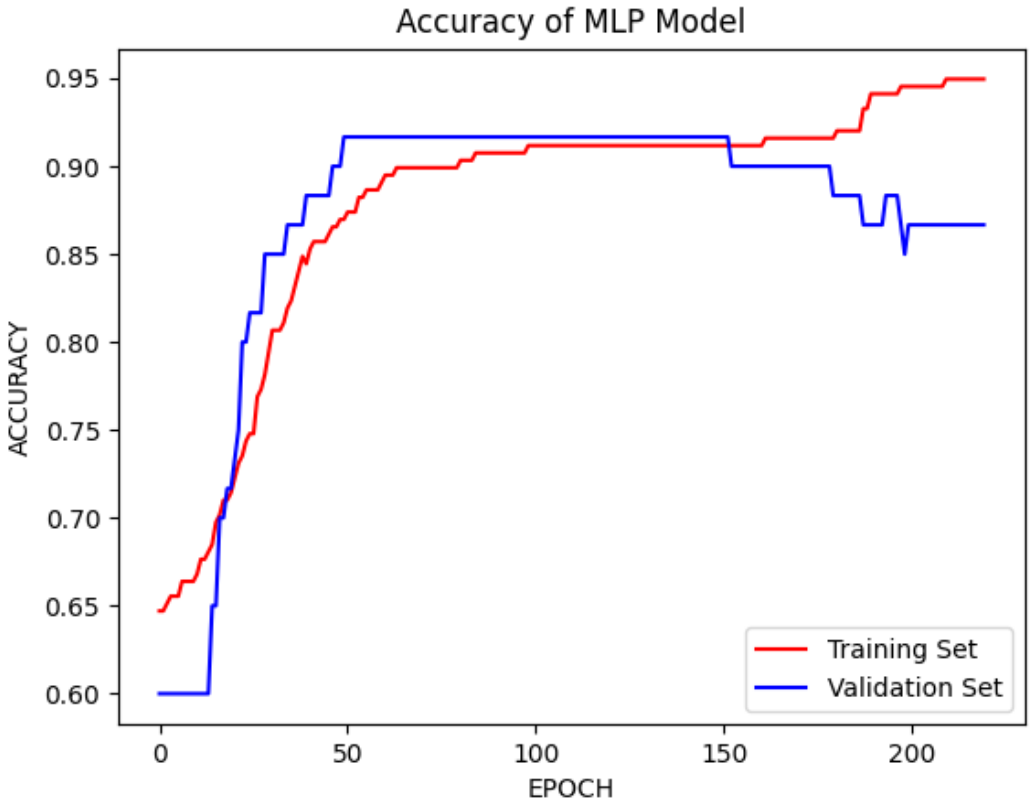


صحت و خطای مدل آموزش دیده برای داده‌های تست به صورت زیر بدست آمد.

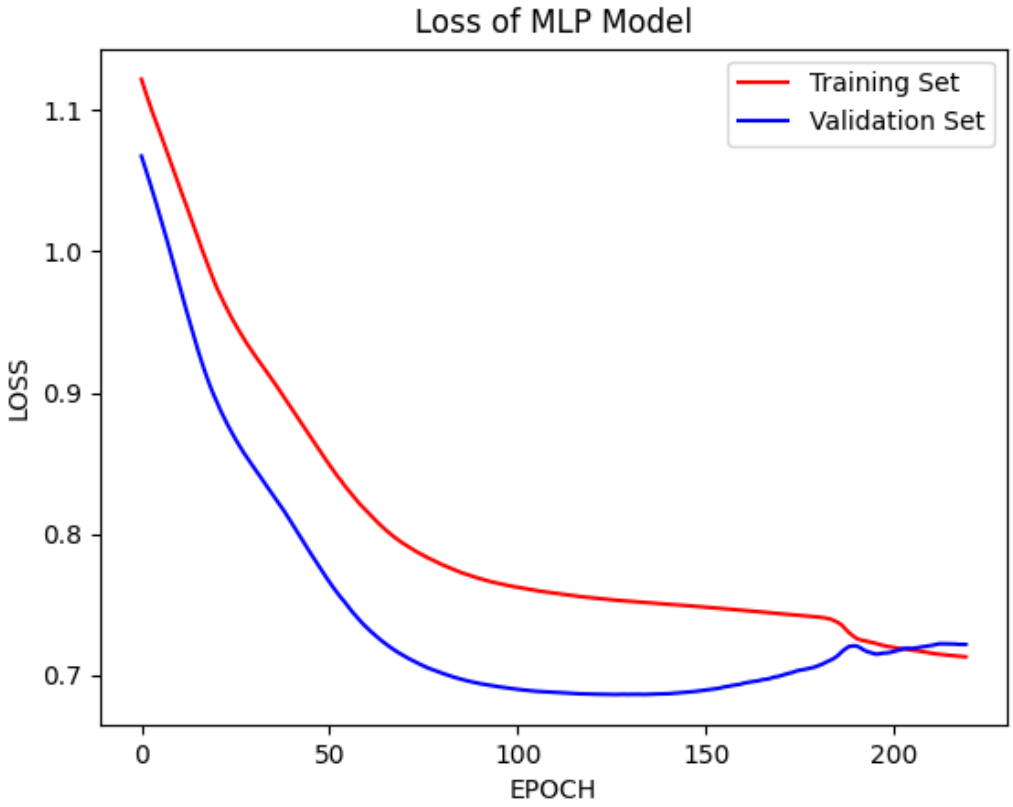
صحت مدل برای داده‌های تست	0.8867924809455872
خطای مدل برای داده‌های تست	0.7790065407752991

نتایج تابع relu :

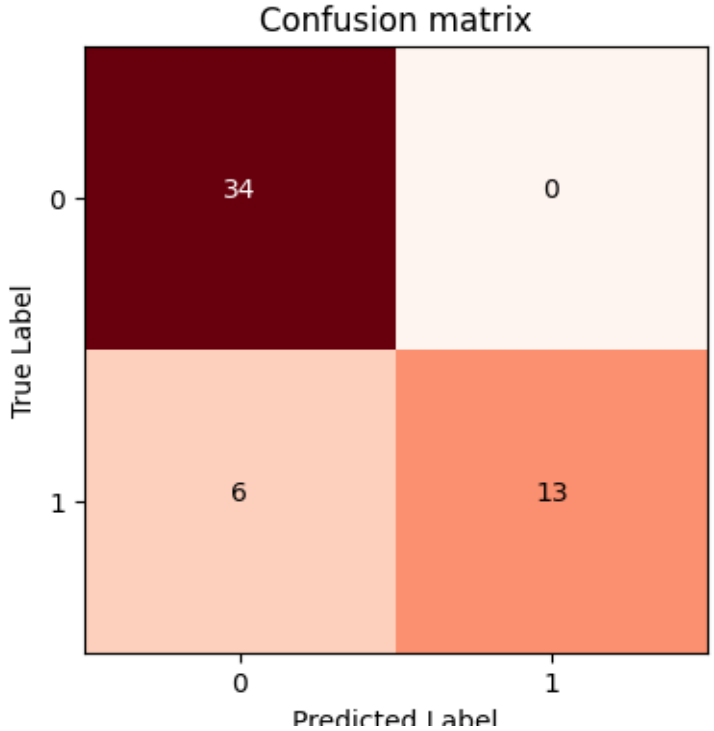
صحت مدل برای داده‌های آموزش و ارزیابی به ازای ایپاک‌های طی شده به صورت زیر است:



نمودار خطای مدل برای مجموعه‌ی آموزش و ارزیابی به صورت زیر است:



و ماتریس آشفتگی نیز به صورت زیر است:

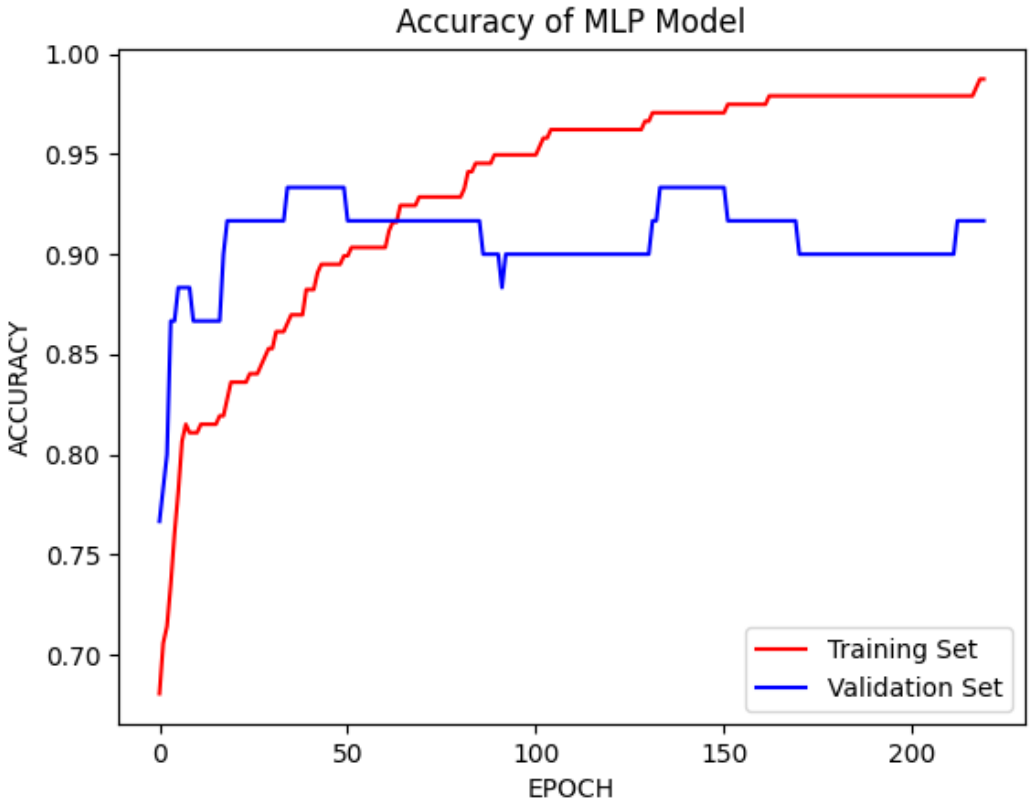


صحت و خطای مدل آموزش دیده برای داده‌های تست به صورت زیر بدست آمد.

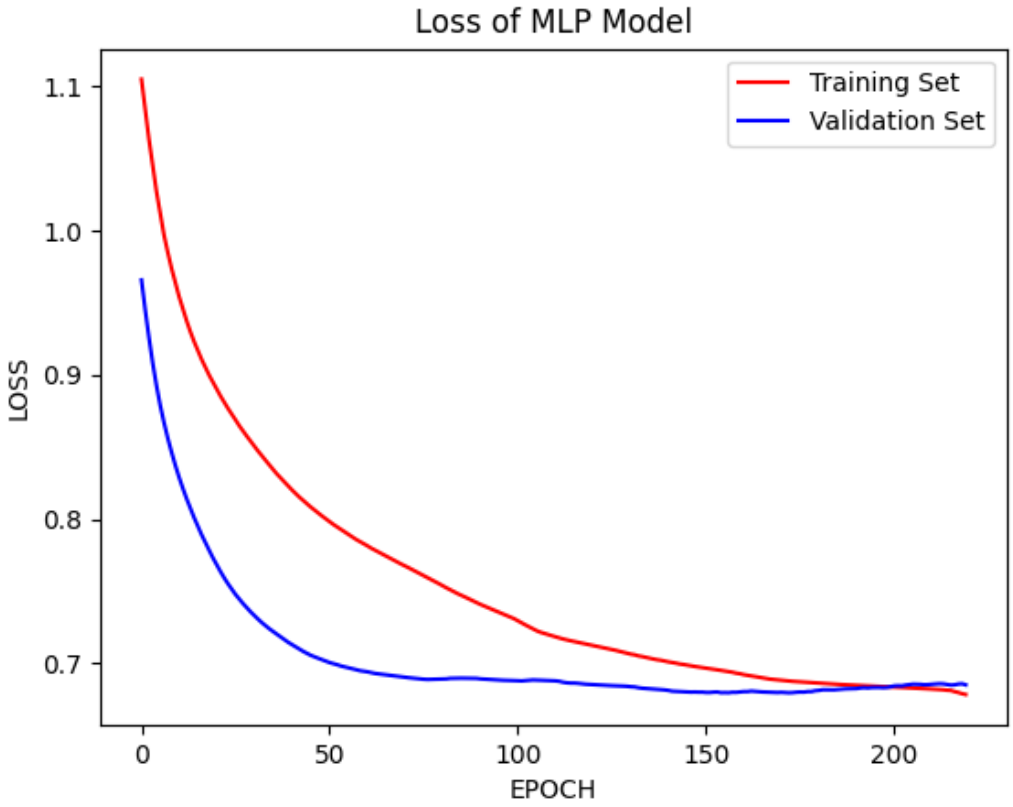
صحت مدل برای داده‌های تست	0.8867924809455872
خطای مدل برای داده‌های تست	0.7740792036056519

نتایج تابع tanh :

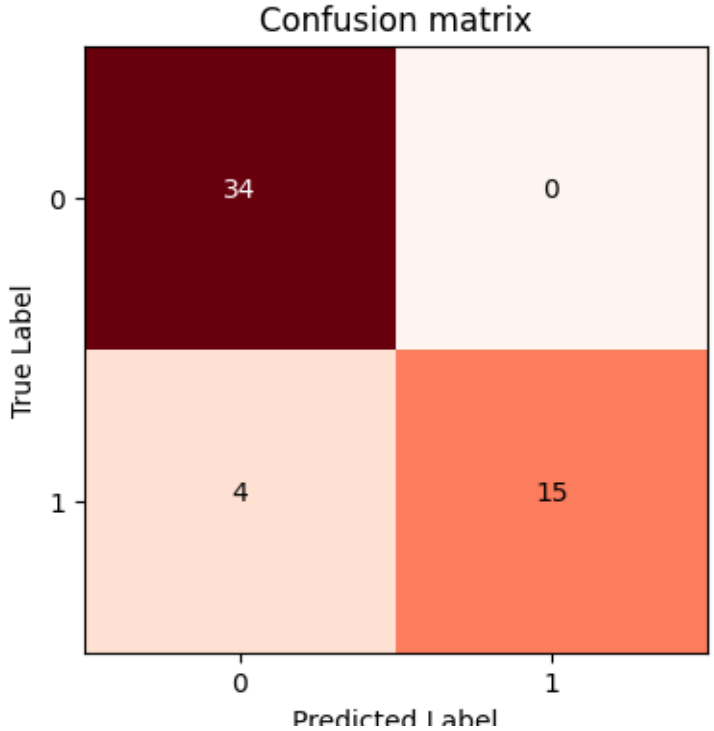
صحت مدل برای داده‌های آموزش و ارزیابی به ازای ایپاک‌های طی شده به صورت زیر است:



نمودار خطای مدل برای مجموعه‌ی آموزش و ارزیابی به صورت زیر است:



و ماتریس آشفتگی نیز به صورت زیر است:



صحت و خطای مدل آموزش دیده برای داده‌های تست به صورت زیر بدست آمد.

صحت مدل برای داده‌های تست	0.9245283007621765
خطای مدل برای داده‌های تست	0.727911114692688

در ابتدا صحت مدل را به ازای این توابع فعالساز مقایسه می‌کنیم.

Model Accuracy(tanh) > Model Accuracy(Relu) = Model Accuracy(Sigmoid) > Model Accuracy(softplus)

حال خطای مدل را بررسی می‌کنیم.

Model Loss(tanh) < Model Loss(Relu) < Model Loss(softplus) < Model Loss(Sigmoid)

از نتایج بالا مشخص است که تابعِ فعالساز برای مدل ما tanh می‌باشد چراکه هم صحت مدل بالاتر و هم خطای مدل کمتری نسبت به سایر توابعِ فعالساز دارد.

حال طبق خواسته‌ی سوال به بررسی مزایا و معایت هر یک از این توابع فعالساز می‌پردازیم:

- ❖ Softplus
  - مزایا: همگراییِ گرادیان بسیار اسموث تر از ریلو است. و نیز می‌تواند مشکل محوشدگی گرادیان را هندل کند.
  - معایب: از لحاظ محاسباتی پیچیدگی بیشتری (مثلا نسبت به ریلو) دارد.
- ❖ Sigmoid
  - مزایا: در هنگام همگرایی، گرادیان اسموثی از خود نشان می‌دهد. و نیز یکی بهترین توابعِ نرمالایزد شده است.
  - معایب: احتمال رخ دادنِ مشکل محوشدگی گرادیان در این روش وجود دارد.
- ❖ Relu
  - مزایا: می‌تواند مشکل محوشدگی گرادیان را هندل کند. پیچیدگی محاسباتی کمی دارد.
  - معایب: یک تابع zero-centric نیست.
- ❖ tanh

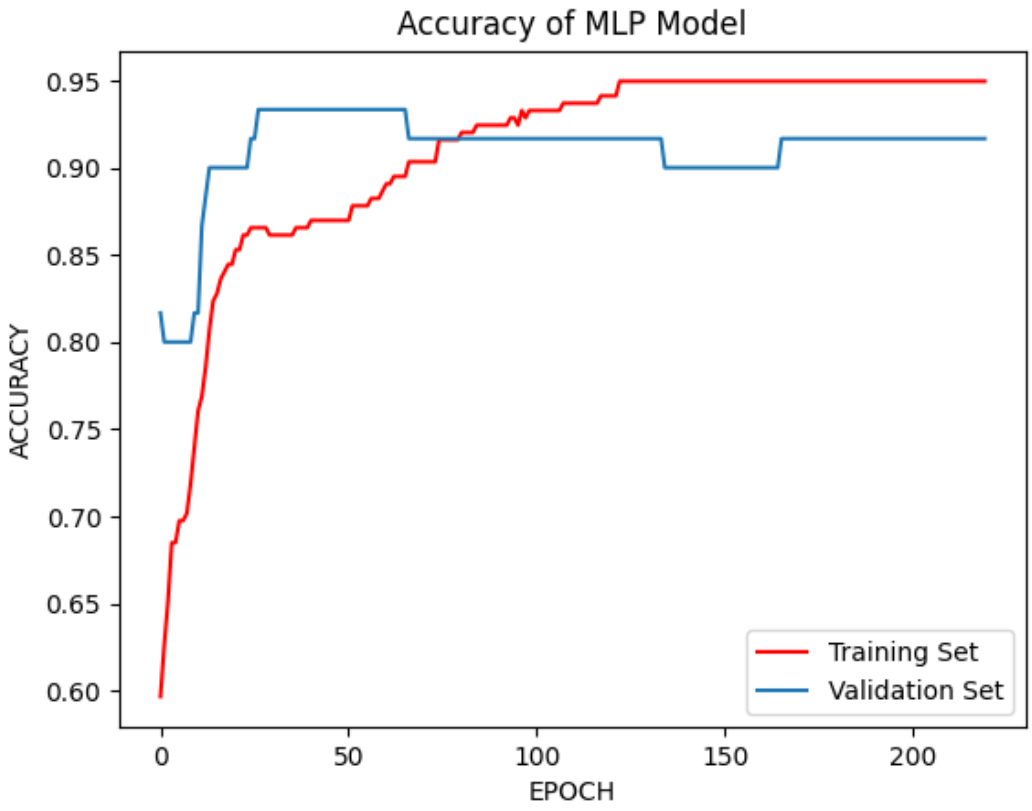
- مزایا: به طور معمول به عنوان یک تابع احتمالاتی برای دسته بندیِ خروجی‌های باینری (مشابه مسألله ما) استفاده می‌شود. و نیز یک تابع zero-centric است. شیب همگراییِ اسموثی دارد.
- معایب: احتمال رخ دادنِ مشکل محوشدگی گرادیان در این روش وجود دارد. و نیز یک تابع با هزینه‌ی محاسباتی بالایی است.

بخش ه)

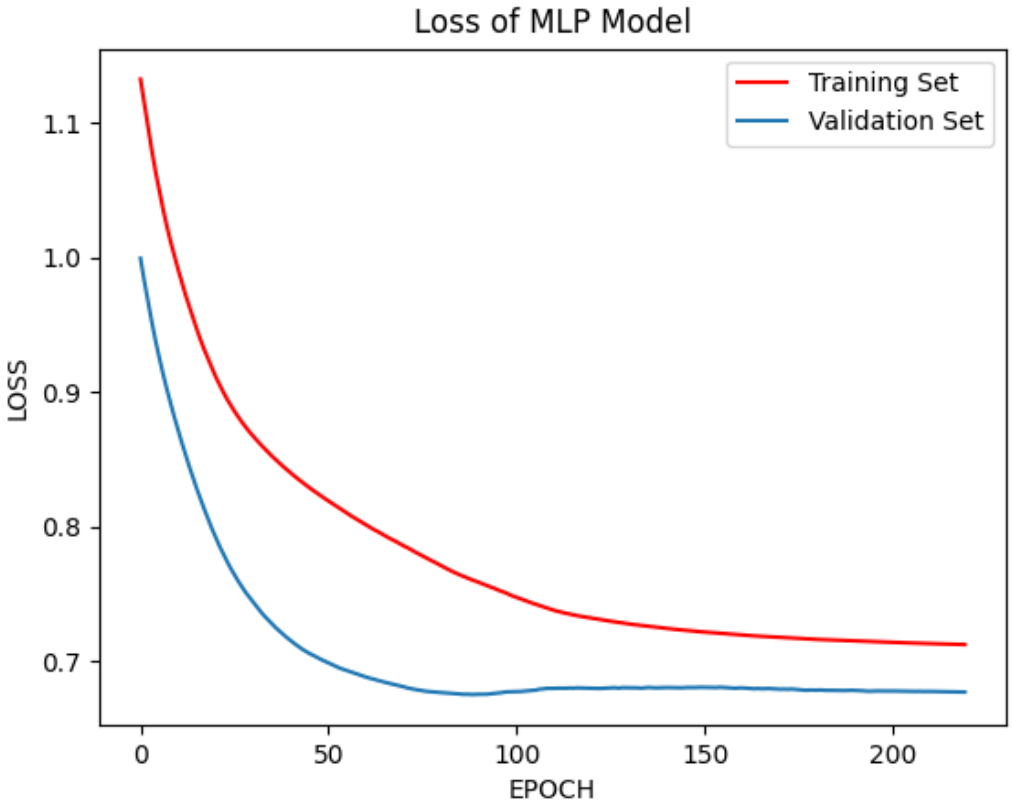
این مرحله را مطابق با خواسته‌ی سوال با بهترین نتایج در مراحلِ قبلی پیش می‌گیریم.

Hinge:

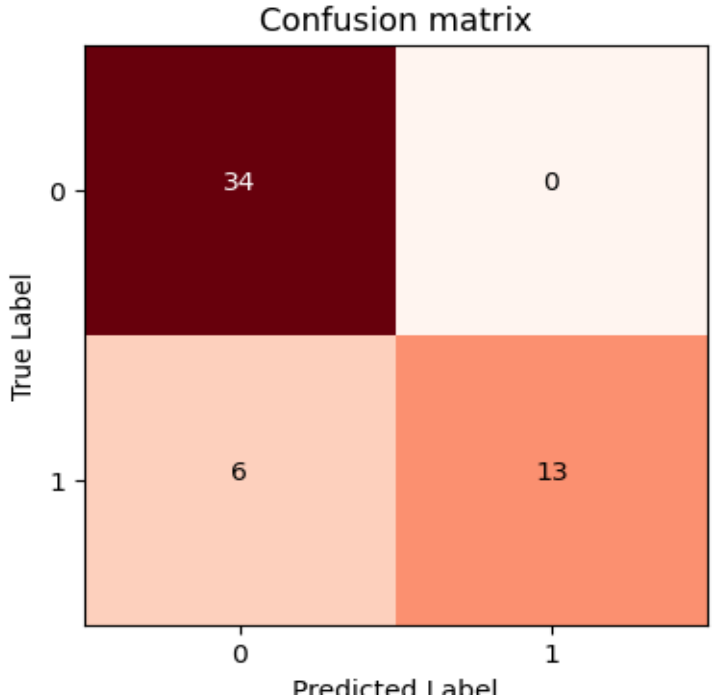
صحت مدل برای داده‌های آموزش و ارزیابی به ازای ایپاک‌های طی شده به صورت زیر است:



نمودار خطای مدل برای مجموعه‌ی آموزش و ارزیابی به صورت زیر است:



و ماتریس آشفتگی نیز به صورت زیر است:

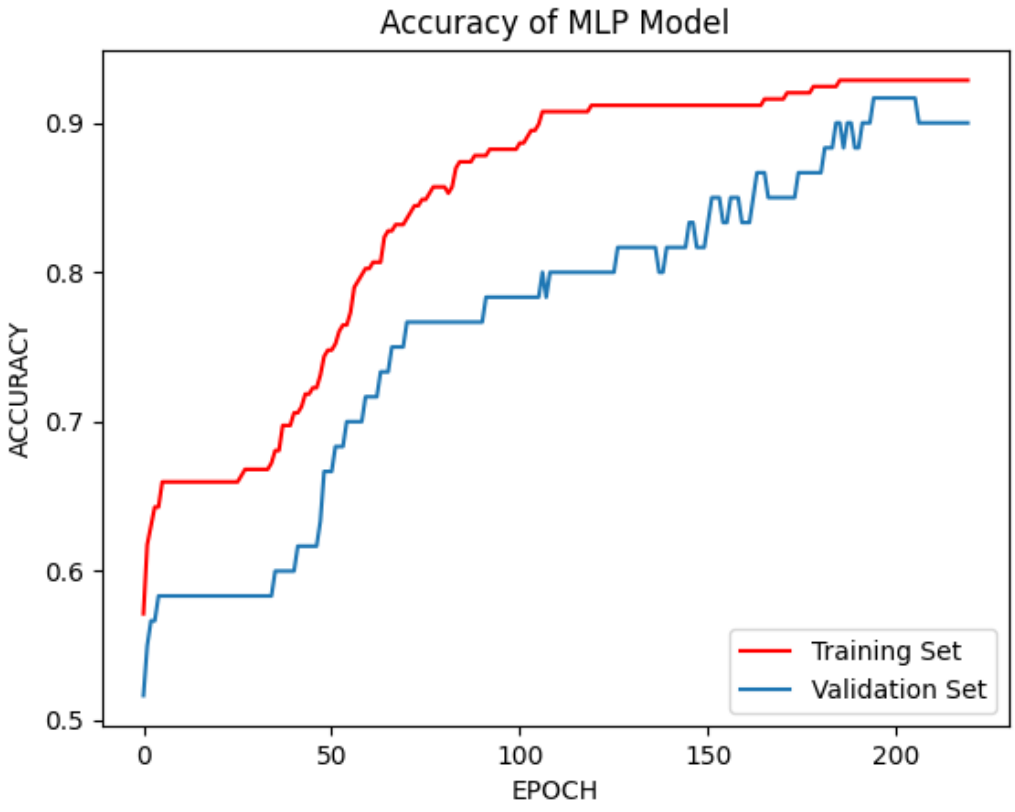


صحت و خطای مدل آموزش دیده برای داده‌های تست به صورت زیر بدست آمد.

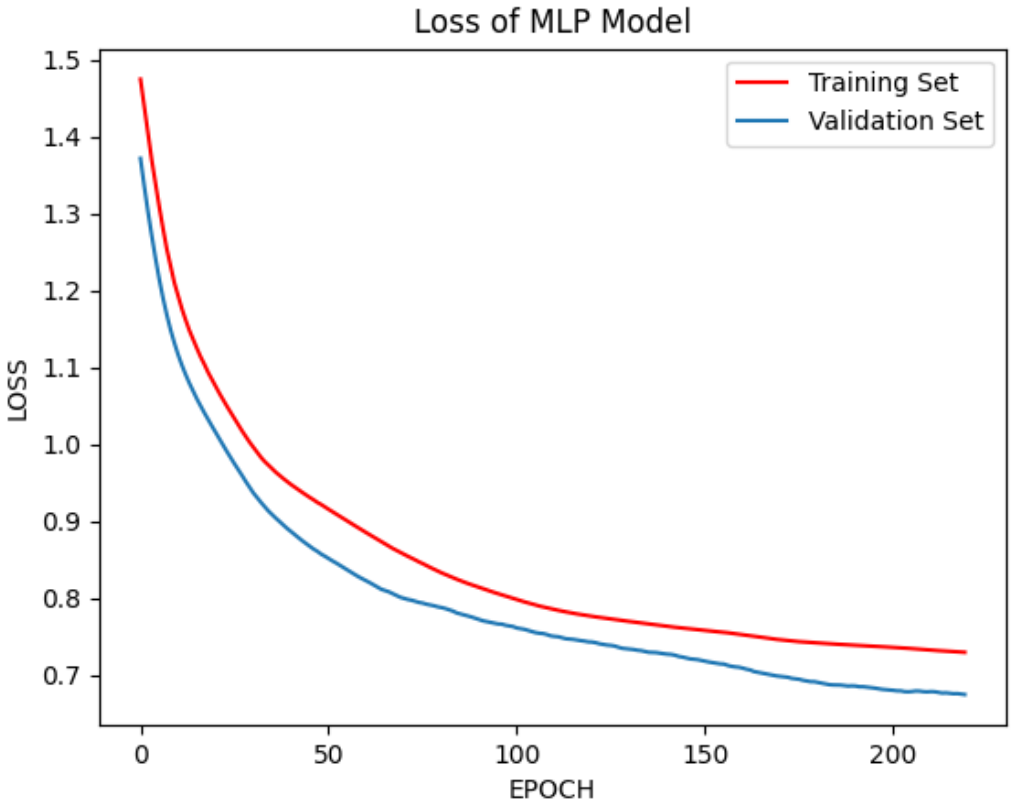
صحت مدل برای داده‌های تست	0.8867924809455872
خطای مدل برای داده‌های تست	0.7614867091178894

Squared hing:

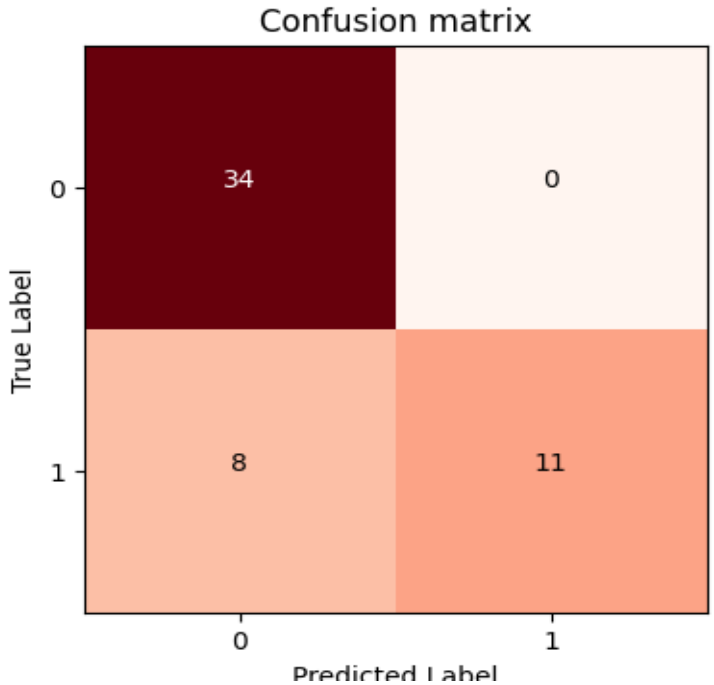
صحت مدل برای داده‌های آموزش و ارزیابی به ازای ایپاک‌های طی شده به صورت زیر است:



نمودار خطای مدل برای مجموعه‌ی آموزش و ارزیابی به صورت زیر است:



و ماتریس آشفتگی نیز به صورت زیر است:

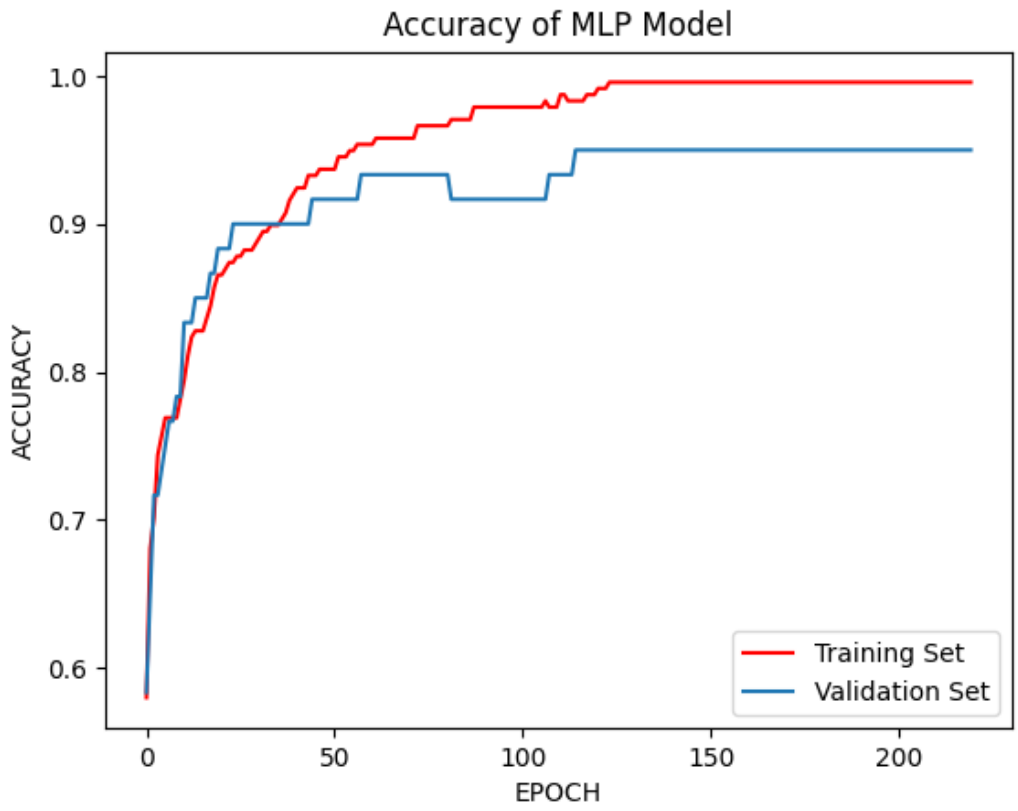


صحت و خطای مدل آموزش دیده برای داده‌های تست به صورت زیر بدست آمد.

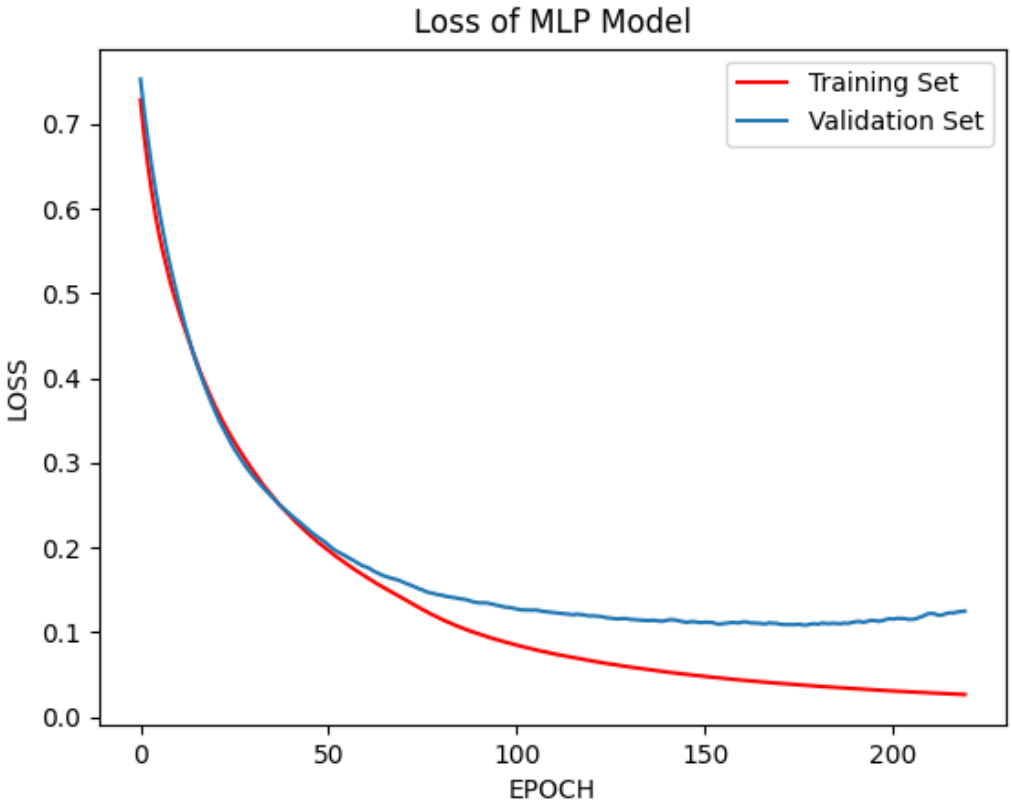
صحت مدل برای داده‌های تست	0.849056601524353
خطای مدل برای داده‌های تست	0.7706219553947449

:binary_crossentropy

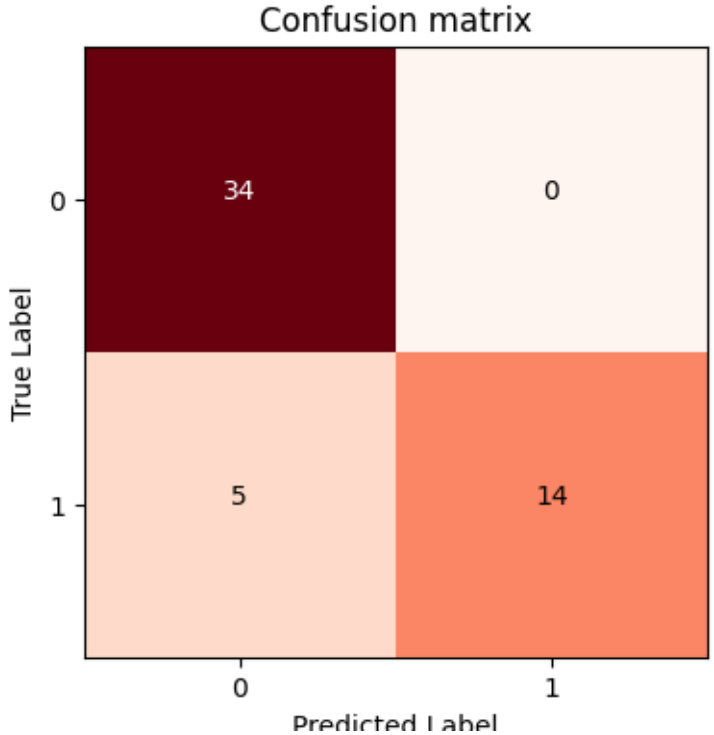
صحت مدل برای داده‌های آموزش و ارزیابی به ازای ایپاک‌های طی شده به صورت زیر است:



نمودار خطای مدل برای مجموعه‌ی آموزش و ارزیابی به صورت زیر است:



و ماتریس آشفتگی نیز به صورت زیر است:



صحت و خطای مدل آموزش دیده برای داده‌های تست به صورت زیر بدست آمد.

صحت مدل برای داده‌های تست	0.9056603908538818
خطای مدل برای داده‌های تست	0.21897006034851074

همانطور که مشاهده می‌کنیم بهترین نتیجه برای binary crossentropy چه از منظرِ دقت و چه میزان خطای مدل بدست آمد.

برای خطای binary crossentropy داریم:

$$\text{Loss} = -\frac{1}{\text{output size}} \sum_{i=1}^{\text{output size}} y_i \cdot \log \hat{y}_i + (1 - y_i) \cdot \log (1 - \hat{y}_i)$$

برای hinge نیز داریم:

$$\ell(y) = \max(0, 1 - t \cdot y)$$

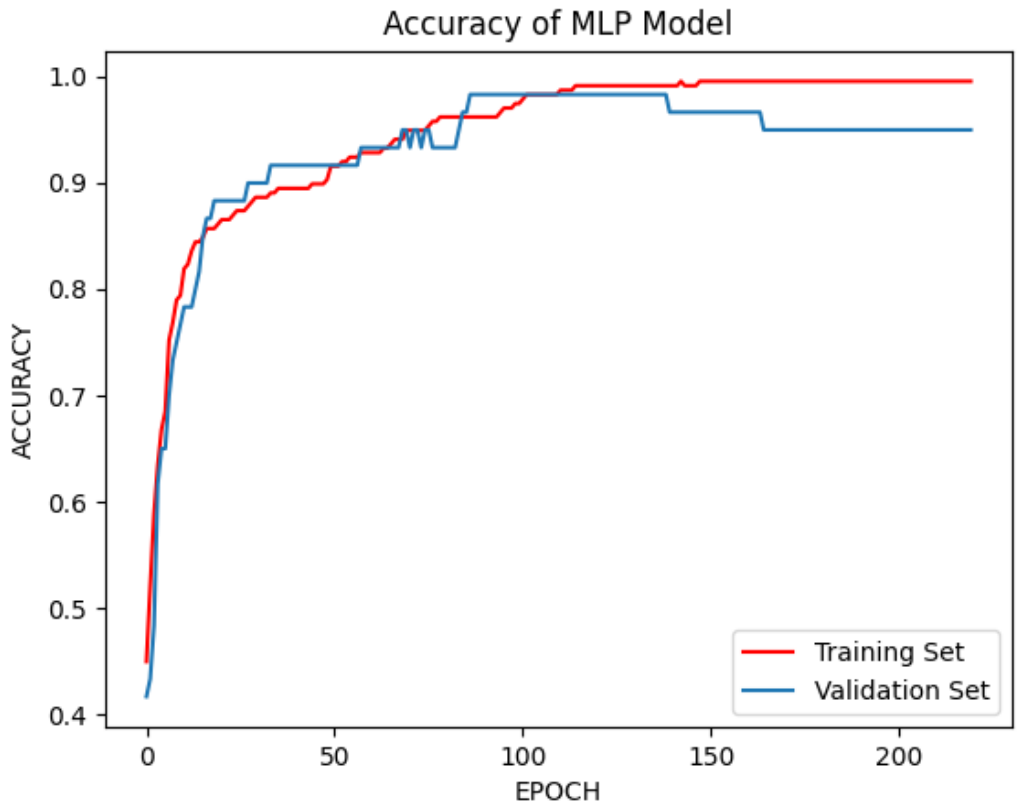
و برای square hinge نیز مولفه‌های فورمول فوق در max به توان دو می‌رسند. از هر کدام از ساختارهای فوق نتیجه می‌گیریم که binary cross entropy یک log loss و نیز hinge و square hinge یک SVM loss هستند و نیز می‌دانیم log loss در بحث مربوط به طبقه‌بندی مانند مسأله‌ی ما، یک logistic regression را بروز می‌دهد. از طرفی مشاهده می‌نمایید که hinge تابعی non-smooth و نیز non-differentiable است و این درحالی است که Logistic Regression یک تابعی smooth و differentiable می‌باشد و ایم مسأله انعطاف حضور این تابع را در گستره‌ی زیادی از مسائل مانند مسأله‌ی ما فراهم می‌کند. از طرفی SVM نمی‌تواند به مدل کردنِ مسائل احتمالاتی در binary classification به صورت مستقین وارد شود و این در حالی است که در مسأله‌ی ما از تابع فعالساز tanh به عنوان یک تابع احتمالاتی برای 0 و 1 ها استفاده شده است و این خود نشان‌دهنده‌ی برتری binary cross entropy نسبت به دو تابع هزینه‌ی دیگر است.

بخش و)

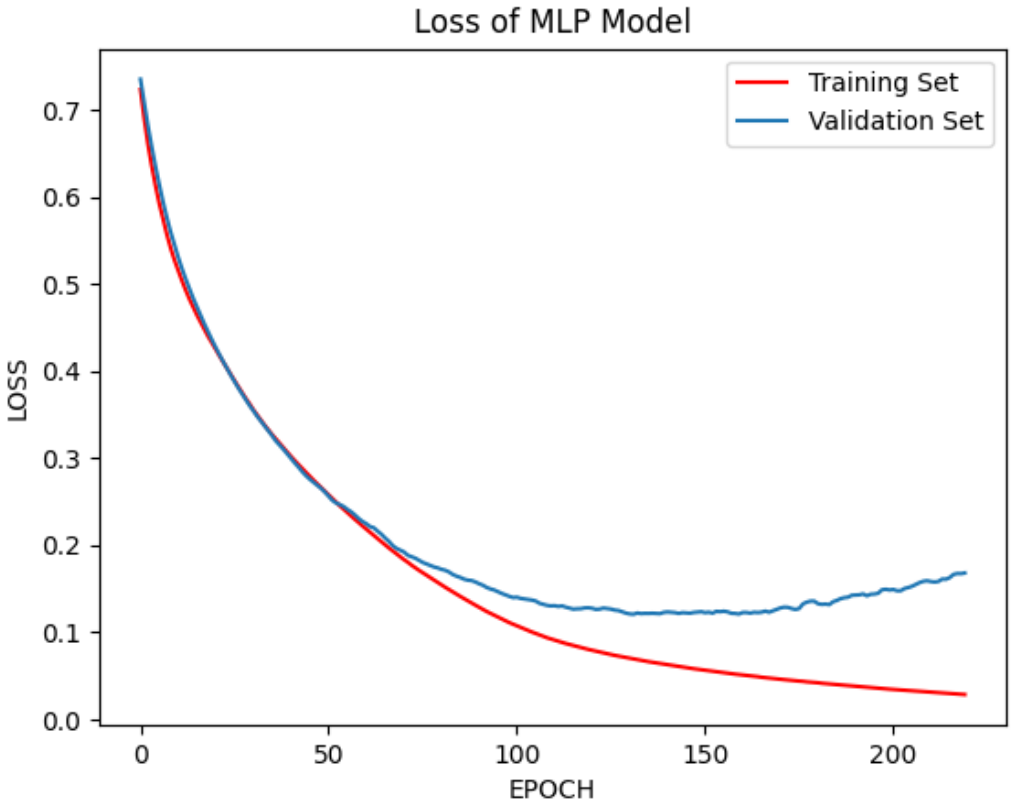
این مرحله را نیز مطابق با خواسته‌ی سوال با بهترین نتایج در مراحلِ پیشین پیاده کردیم و نتایج به صورت زیر است. البته لازم به ذکر است که توابع زیادی برای بهینه‌سازی وجود دارد ولی سه تابع ذکر شده در این بخش مخصوصِ binary classification است و با مسأله‌ی ما کاملا تطابق دارد.

بهینه ساز Adam:

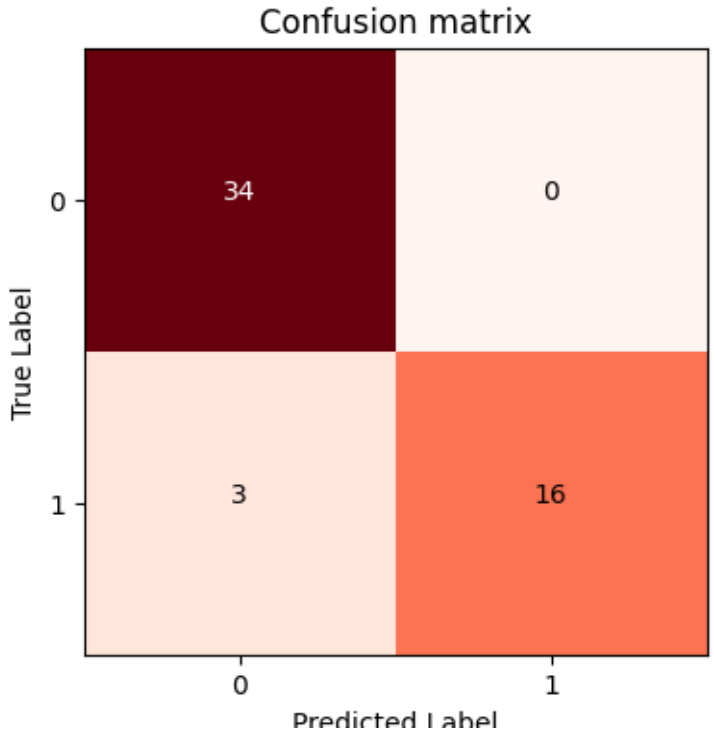
صحت مدل برای داده‌های آموزش و ارزیابی به ازای ایپاک‌های طی شده به صورت زیر است:



نمودار خطای مدل برای مجموعه‌ی آموزش و ارزیابی به صورت زیر است:



و ماتریس آشفتگی نیز به صورت زیر است:

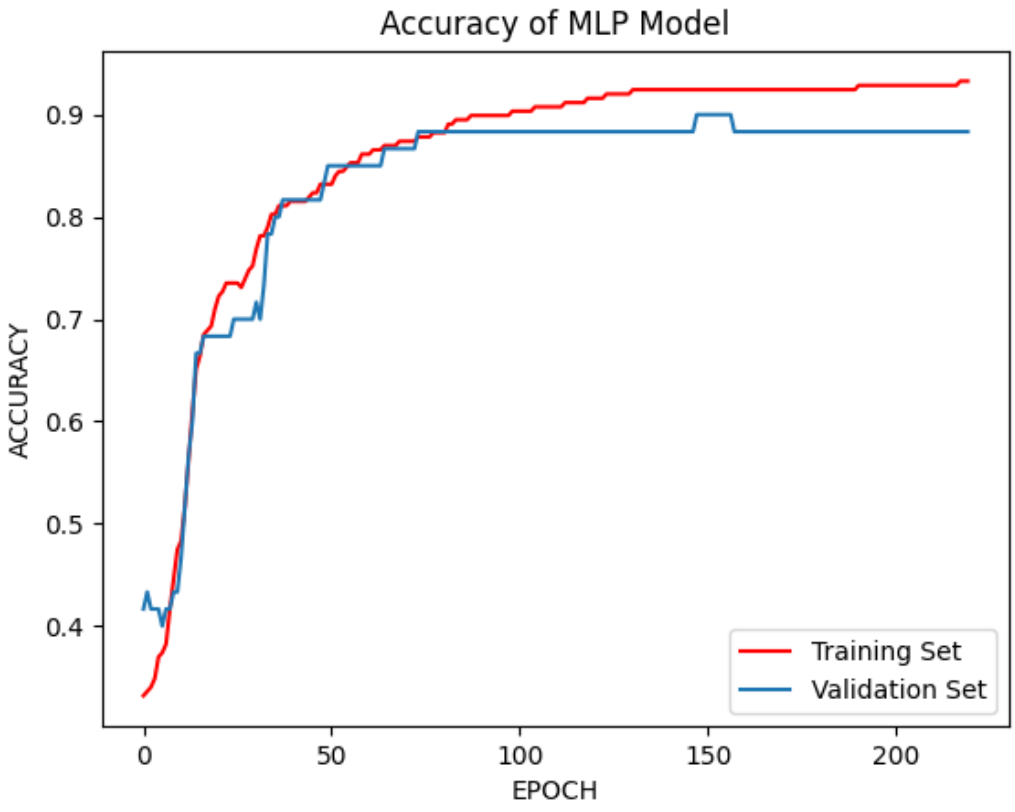


صحت و خطای مدل آموزش دیده برای داده‌های تست به صورت زیر بدست آمد.

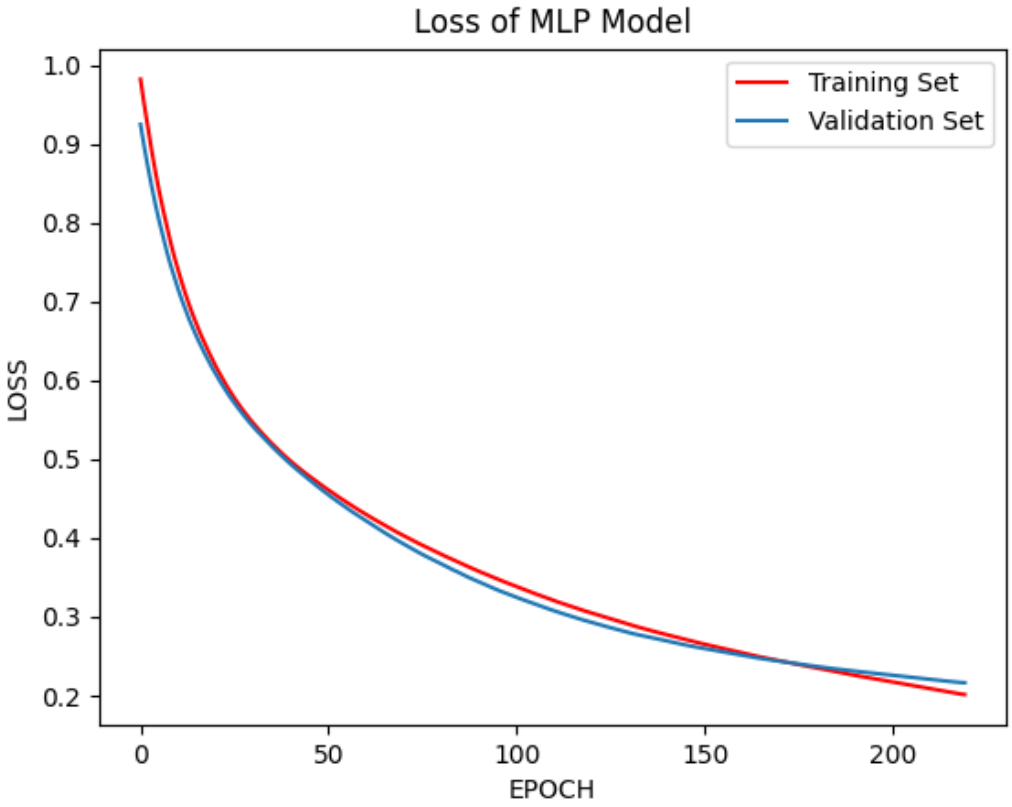
صحت مدل برای داده‌های تست	0.9433962106704712
خطای مدل برای داده‌های تست	0.23269927501678467

بهینه ساز SGD:

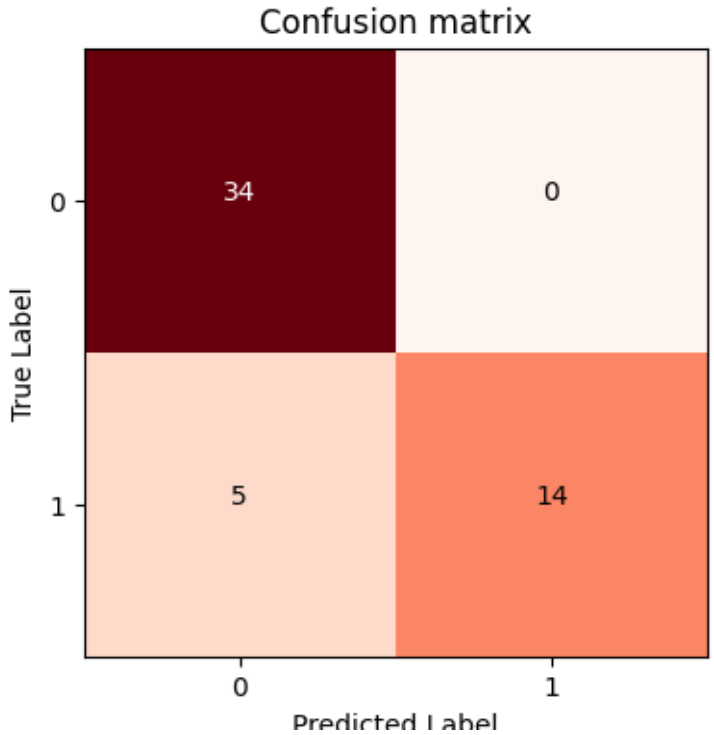
صحت مدل برای داده‌های آموزش و ارزیابی به ازای ایپاک‌های طی شده به صورت زیر است:



نمودار خطای مدل برای مجموعه‌ی آموزش و ارزیابی به صورت زیر است:



و ماتریس آشفتگی نیز به صورت زیر است:

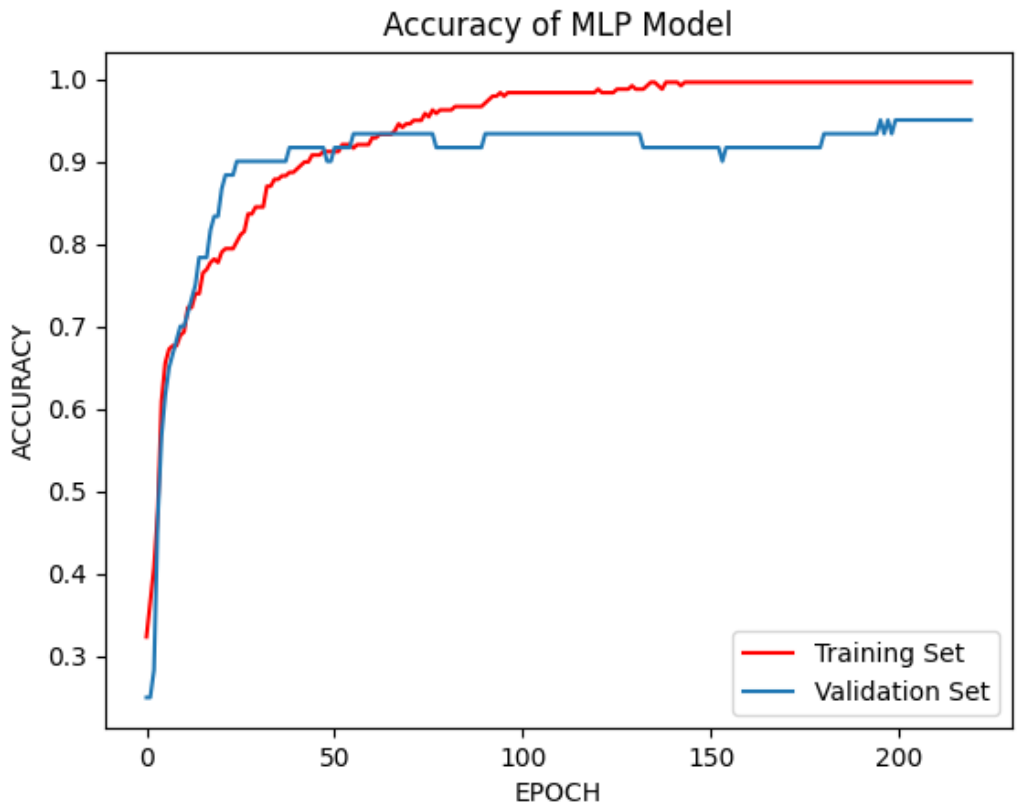


صحت و خطای مدل آموزش دیده برای داده‌های تست به صورت زیر بدست آمد.

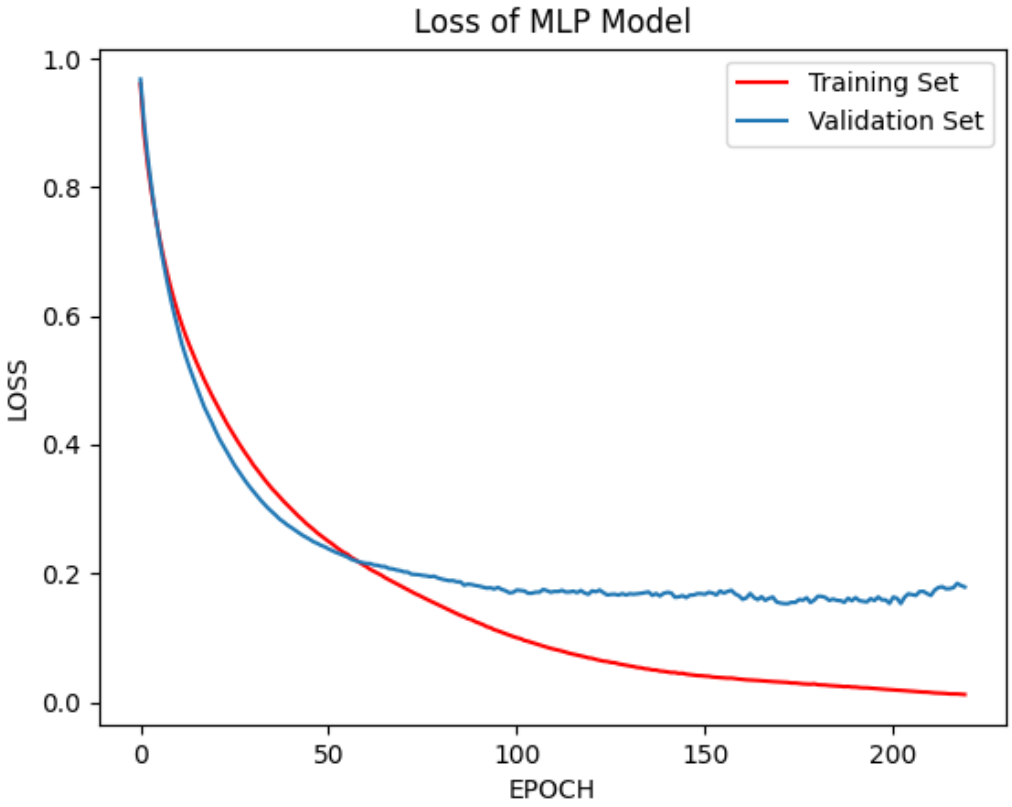
صحت مدل برای داده‌های تست	0.9056603908538818
خطای مدل برای داده‌های تست	0.2958594858646393

بهینه ساز RMSProp:

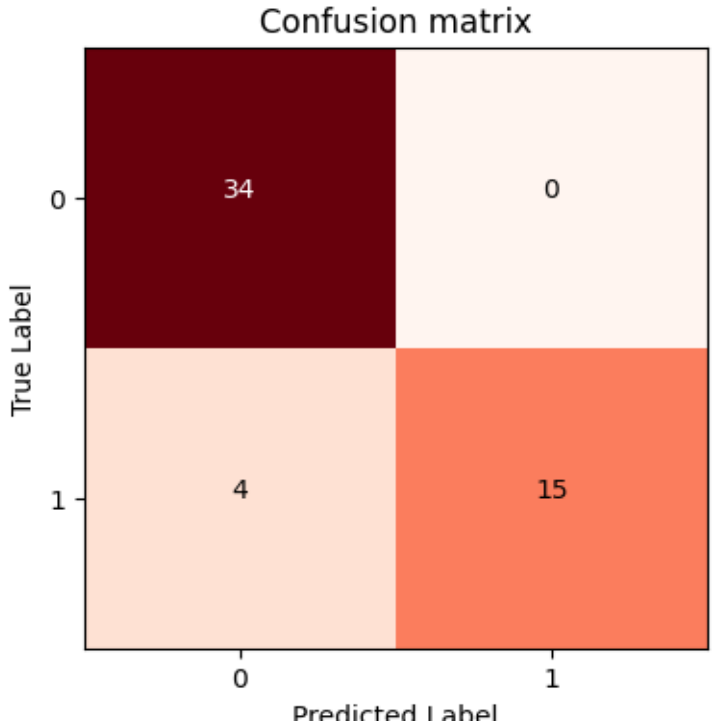
صحت مدل برای داده‌های آموزش و ارزیابی به ازای ایپاک‌های طی شده به صورت زیر است:



نمودار خطای مدل برای مجموعه‌ی آموزش و ارزیابی به صورت زیر است:



و ماتریس آشفتگی نیز به صورت زیر است:



صحت و خطای مدل آموزش دیده برای داده‌های تست به صورت زیر بدست آمد.

صحت مدل برای داده‌های تست	0.9245283007621765
خطای مدل برای داده‌های تست	0.25393983721733093

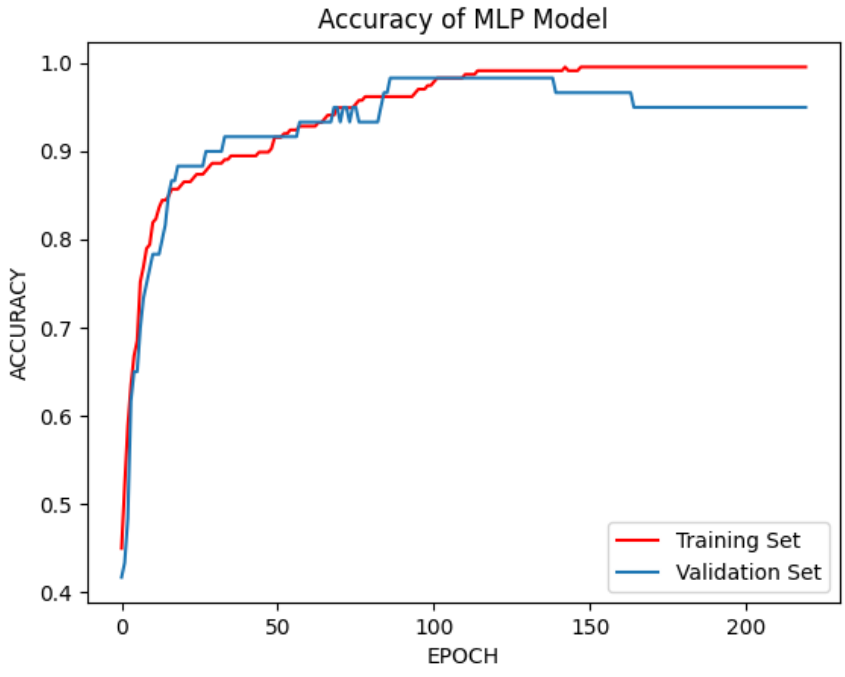
همانطور که مشاهده می‌کنیم، نتایج نشان می‌دهد که بهترین بهینه‌ساز در مسأله‌ی ما adam با صحت 0.9433962106704712 و دقت 0.23269927501678467 می‌باشد. پس از adam، بهترین نتیجه را Root Mean Squared Propagation از خود نشان داد و در آخر هم SGD.

بخش ح)

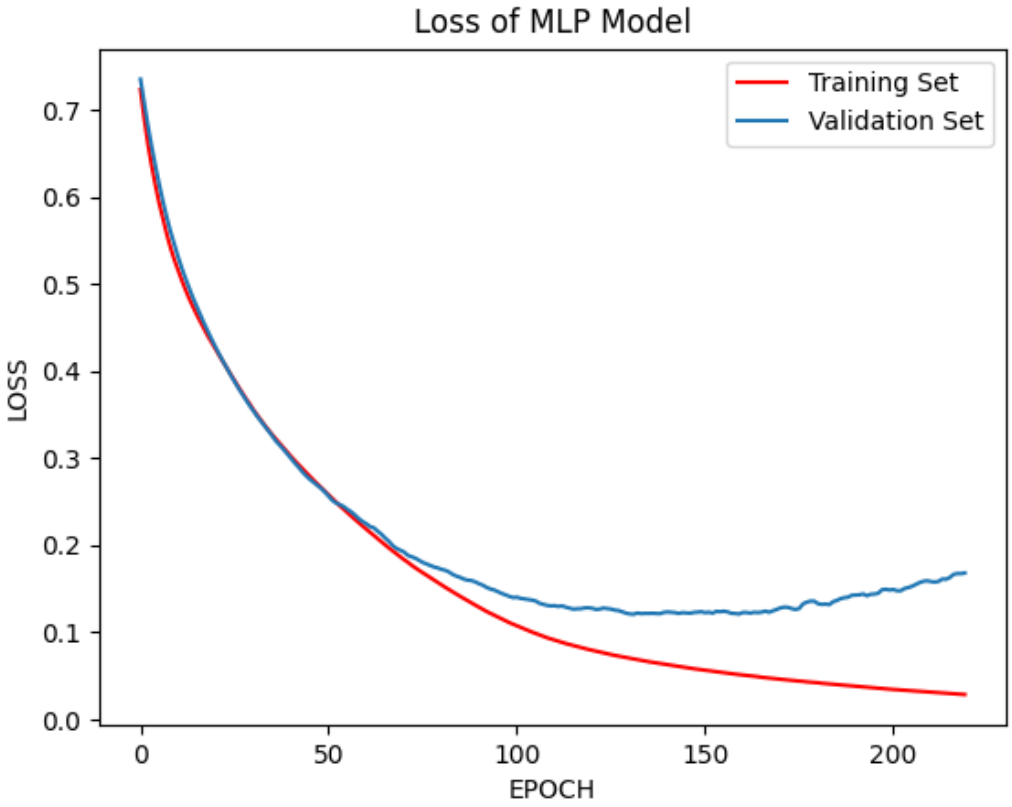
در این سوال در ابتدا یک لایه اضافه با تعداد نورون4 اضافه کردیم و سپس یک لایه‌ی دیگر به مدل قبلی با تعداد نورون 3 و در نهایت یک لایه‌ی دیگر به مدل قبلی با تعداد نورون 2 را اضافه کردیم. برای توابع فعالساز هم از ریلو در لایه‌های پنهان جز لایه‌ی پنهان آخر که tanh است استفاده نمودیم که برامده از بهترین نتایجمان می‌باشد.

تعداد نورون‌ها در لایه‌ی پنهان [15, 6, 4]:

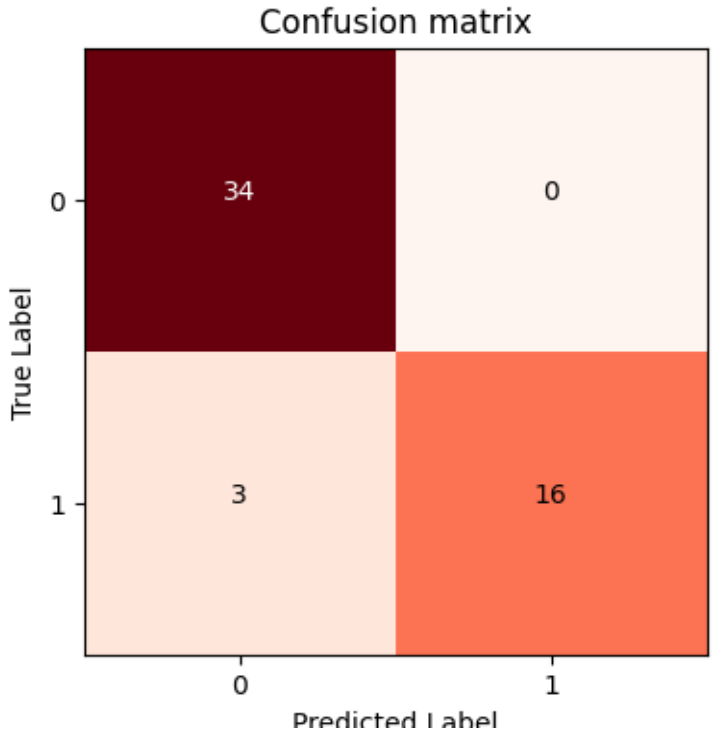
صحت مدل برای داده‌های آموزش و ارزیابی به ازای ایپاک‌های طی شده به صورت زیر است:



نمودار خطای مدل برای مجموعه‌ی آموزش و ارزیابی به صورت زیر است:



و ماتریس آشفتگی نیز به صورت زیر است:

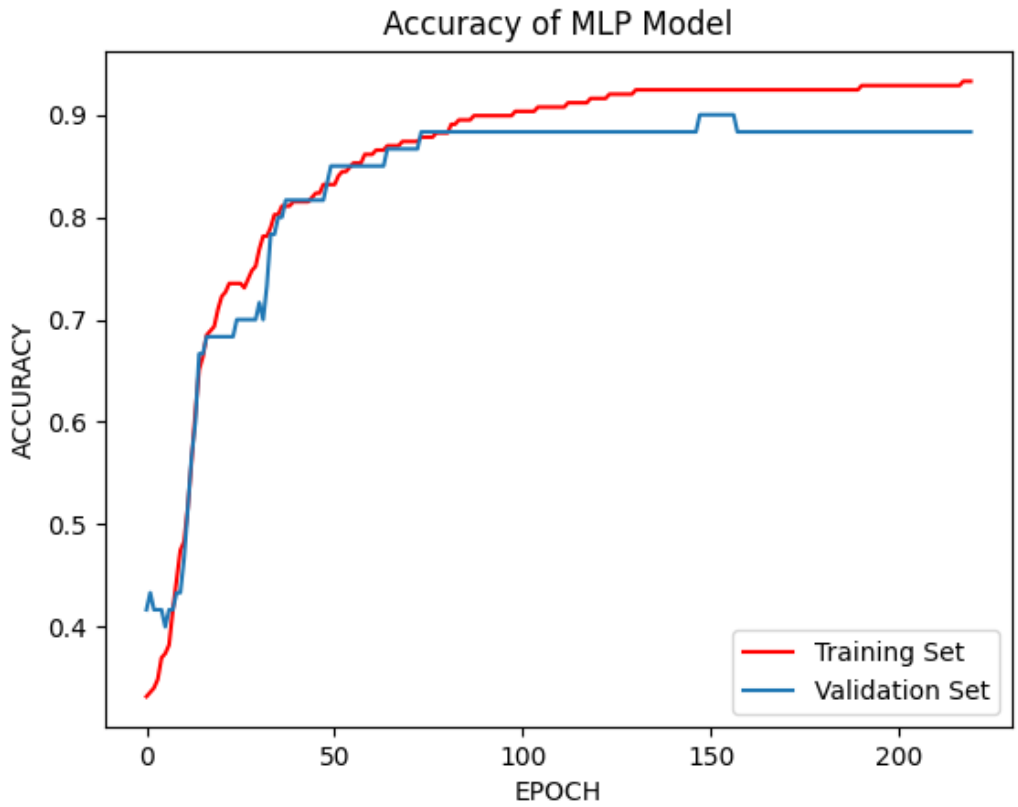


و در نهایت، نتایج مدل بر روی داده‌های تست به صورت زیر می‌باشد.

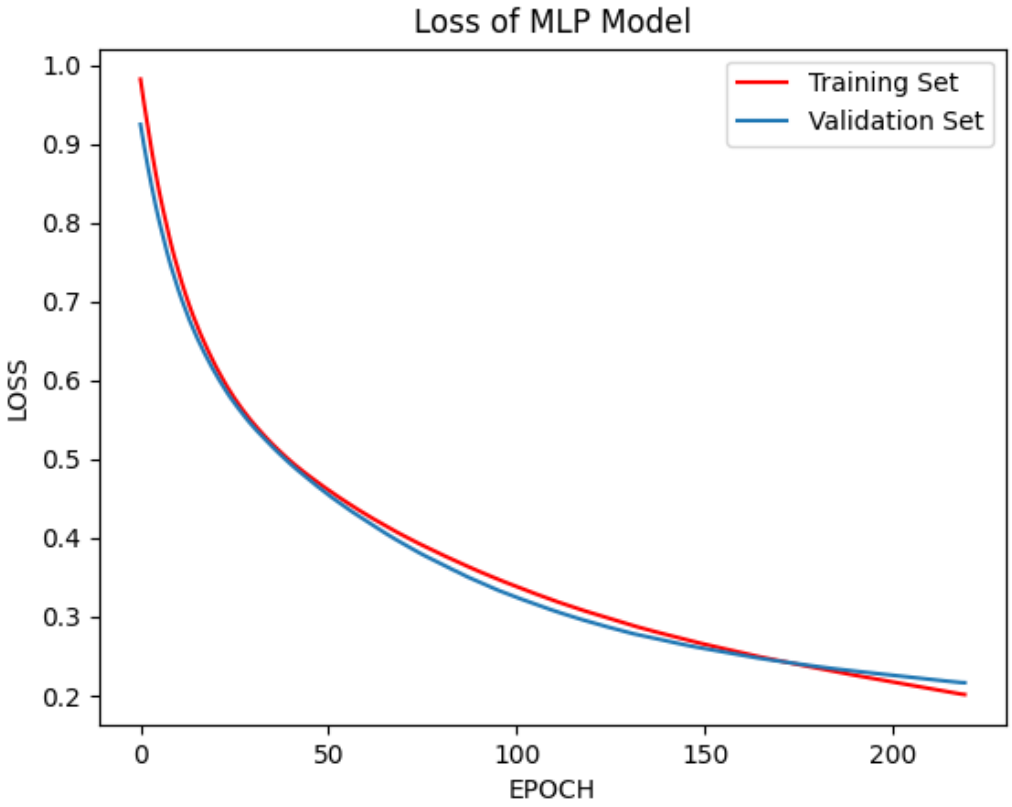
Accuracy	0.9433962106704712
Loss	0.14462608098983765
Precision	1.0
Recall	0.8421052631578947
Fscore	0.9142857142857143

تعداد نورون‌ها در لایه‌ی پنهان [15, 6, 4, 3]:

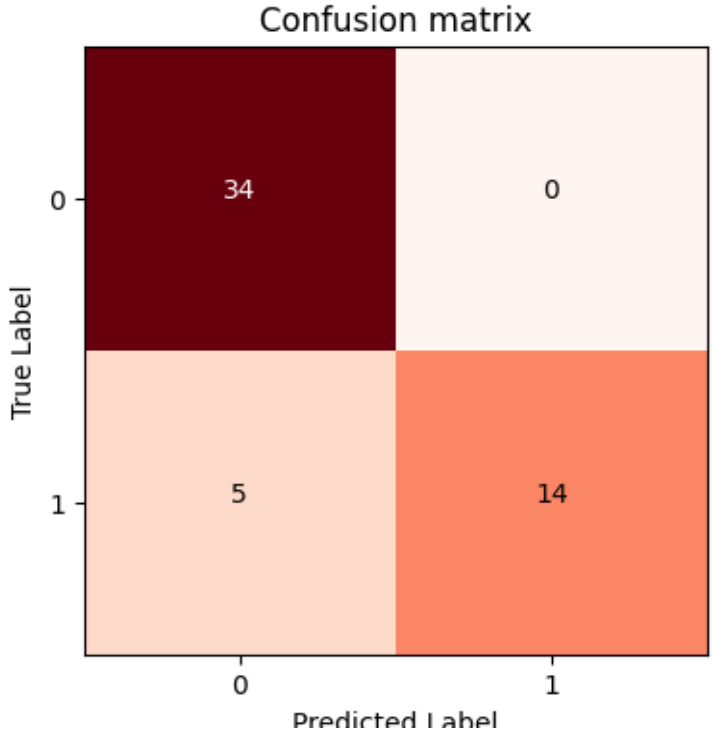
صحت مدل برای داده‌های آموزش و ارزیابی به ازای ایپاک‌های طی شده به صورت زیر است:



نمودار خطای مدل برای مجموعه‌ی آموزش و ارزیابی به صورت زیر است:



و ماتریس آشفتگی نیز به صورت زیر است:

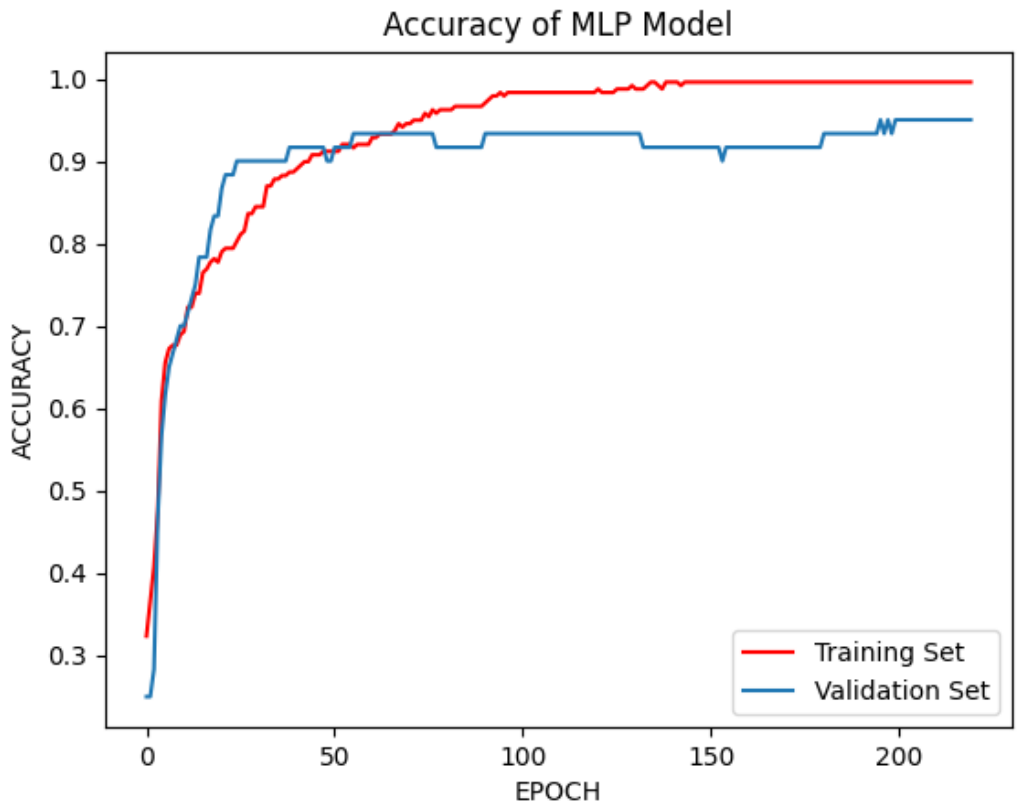


و در نهایت، نتایج مدل بر روی داده‌های تست به صورت زیر می‌باشد.

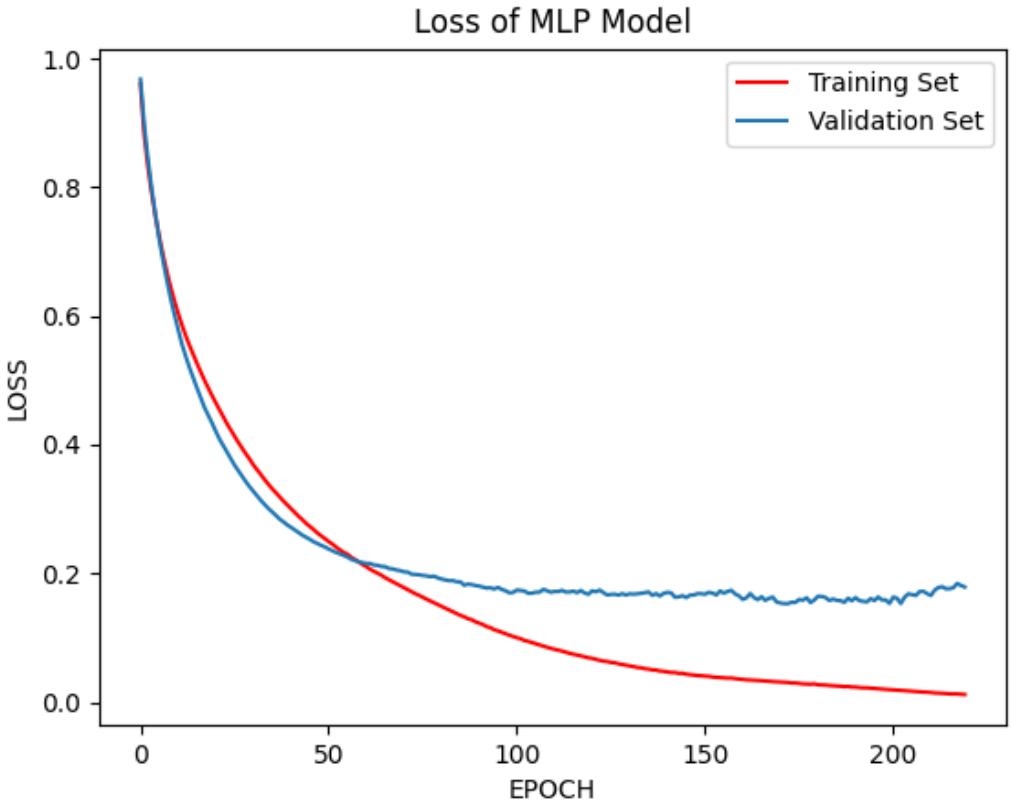
Accuracy	0.9056603908538818
Loss	0.28122007846832275
Precision	0.9375
Recall	0.7894736842105263
Fscore	0.8571428571428572

تعداد نورون‌ها در لایه‌ی پنهان [15, 6, 4, 3, 2]:

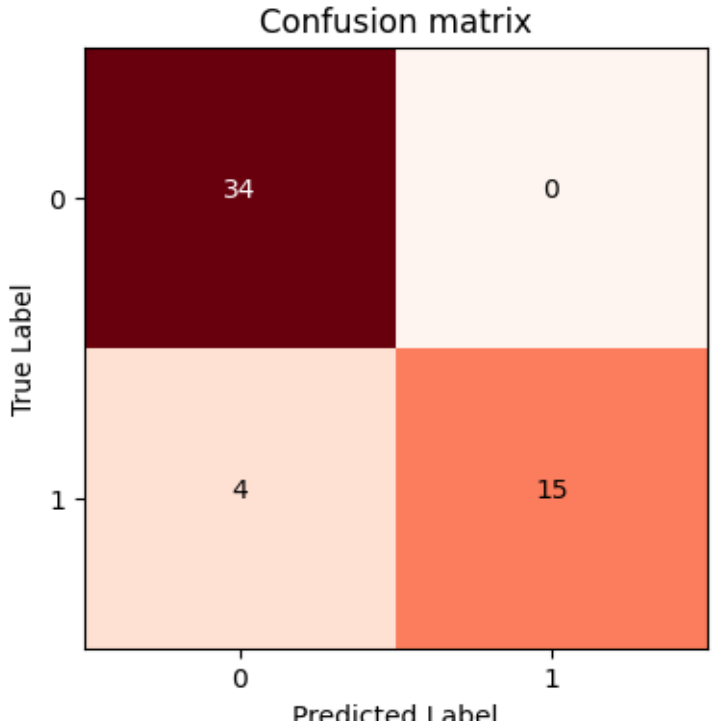
صحت مدل برای داده‌های آموزش و ارزیابی به ازای ایپاک‌های طی شده به صورت زیر است:



نمودار خطای مدل برای مجموعه‌ی آموزش و ارزیابی به صورت زیر است:



و ماتریس آشفتگی نیز به صورت زیر است:



و در نهایت، نتایج مدل بر روی داده‌های تست به صورت زیر می‌باشد.

Accuracy	0.9433962106704712
Loss	0.2694690525531769
Precision	1.0
Recall	0.8421052631578947
Fscore	0.9142857142857143

همانطور که مشاهده می‌کنیم با افزایش یک لایه با تعداد نورون 4 به صحت بالاتر و خطای کمتر رسیدیم و با افزایش یک لایه‌ی دیگر صحت کاهش یافت و در اخر نیز با افزایش یک لایه‌ی دیگر مجددا افزایش یافت. لذا می‌توان از مشاهدات فوق نتیجه گرفت که با افزایشِ تعداد لایه‌ها بیشتر از سه لایه تغییر قابل ملموسی در نتایج ایجاد نمی‌شود و در واقع شبکه‌ی MLP با داشتن همین سه لایه می‌تواند بهترین جواب را از منظر صحت و خطا به ما بدهد و نیازی به افزایشِ تعداد لایه‌های پنهان بیش از 3 تا نیست.

بخش ط)

بهترین نتایج برای هایپرپارامترهای زیر بدست آمد:

تعداد لایه‌های پنهان :3 لایه با نورون‌های [15, 6, 4]

توابع فعالسازی : ['relu', 'relu', 'softplus', 'sigmoid']

Batch-size: 64

Loss: binary cross entropy

بهینه ساز : adam

در رابطه با تعداد لایه نشان دادیم که شبکه‌ی MLP با داشتن سه لایه می‌تواند بهترین جواب را چه از منظر دقت و چه خطا به ما بدهد و نیازی به افزایشِ بیشتر تعداد لایه‌های پنهان نیست. همچنین در رابطه با توابع فعاساز برای بهترین نتیجه که tanh بود ذکر کردیم که این تابع فعالساز به طور معمول به عنوان یک تابع احتمالاتی برای دسته بندیِ خروجی‌های باینری (مشابه مسأله ما) استفاده می‌شود. و نیز یک تابع zero-centric است. و شیب همگراییِ اسموثی دارد که از جمله مزایای این تابع بود و صحت بالاتری را نتیجه داد. و نیز برای batch-size استدلال کردیم که هر اندازه سایز بچ افزایش یابد، زمان آموزش کم

می‌شود و برای دقت نیز به نظر می‌رسد بچ سایزِ 64 عدد مناسبی است تا نه مشکلِ گیر کردن در مینیمم‌های محلی را به ازای افزایشِ زیادِ بچ سایز داشته باشیم و نه مشکلِ واگرایی و کثرتِ تناوب و سرعتِ آموزشِ کم به ازای بچ سایزهای کم (مثلا بچ سایزِ 32 و حتی کمتر). همچنین برای تابع خطا نیز binary cross entropy را قرار دادیم زیرا binary cross entropy یک تابعی smooth و differentiable می‌باشد و این مسأله انعطاف حضور این تابع را در گستره‌ی زیادی از مسائل مانند مسأله‌ی ما فراهم می‌کند. از طرفی SVM نمی‌تواند به مدل کردنِ مسائل احتمالاتی در binary classification به صورت مستقیم وارد شود و این در حالی است که در مسأله‌ی ما از تابع فعالساز tanh به عنوان یک تابع احتمالاتی برای 0 و 1 ها استفاده شده است و این خود نشان‌دهنده‌ی برتری binary cross entropy نسبت به دو تابع هزینه‌ی دیگر است. در نهایت برای بهینه‌ساز نیز با توجه به نتایج صحت و loss، adam را انتخاب کردیم.

بخش ی)

هنگامیکه داده‌های کلاس‌ها از توازنِ عددیِ یکسان یا حدودا یکسانی برخوردار نیستند و یک یا چند کلاس دارای داده‌های کمی نسبت به سایرِ کلاس‌ها می‌باشند، مشکلِ imbalanced classification بوجود می‌اید که در اینصورت مدل قادر نیست پیش‌بینی درستی از داده‌ها بویژه داده‌هایی که در کلاس‌های اقلیت قرار دارند، انجام دهد. همانطور که ذکر کردیم imbalanced classification ناشی از imbalanced dataset است که البته راهکارها و تکنیک‌هایی برای مقابله و کم کردنِ تاثیرِ این مشکل وجود دارد که ذکر می‌کنیم.

روش اول : Random undersampling (RUS)

در این روش تعدادی از داده‌های کلاس‌هایی که اکثریتِ نمونه‌ها را دارند، به صورت تصادفی حذف می‌کنیم تا به طور تقریبی همه‌ی کلاس‌ها از یک توازن در تعدادِ دادهگان برخوردار شوند. این روش از جمله روش‌های Undersampling محسوب می‌شود.

روش دوم : Adaptive synthetic sampling approach (ADASYN)

این روش از جمله روش‌های Data level methods است که در آن از یک تابعی استفاده می‌شود که به محاسبه‌ی چگالی نمونه‌ها به عنوان معیاری خودکار جهتِ برای تعداد نمونه‌هایی که می‌توانند برای هر کلاس اقلیت تولید شوند تا تعداد نمونه‌ها را متوازن کند، استفاده می‌نماید.

روش سوم: Random oversampling

این روش یکی از روش‌های الگوریتم‌های مبتنی بر oversampling است که که با ترکیبِ نمونه‌های کلاس‌های اقلیت به صورت تصادفی، اقدام به تولید نمونه‌های جدید می‌نماید و از این طریق تعداد نمونه‌های کلاس‌های اقلیت را افزایش می‌دهد تا در دیتاست بین مجموعه‌ی کلاس ها از لحاظ تعداد نمونه، توازن بوجود آید.

سوال ۲ – شبکه پرسپترون چندلایه در کاربرد رگرسیون

در ابتدا normalization و standardization را همانند آنچه در سوال یک انجام دادیم و به صورت زیر بر روی مجموعه‌ی داده‌گان اعمال می‌کنیم.

Standardization **Normalization**

$$x = \frac{x - mean(x)}{standard\ deviation(x)} \qquad x = \frac{x - min(x)}{max(x) - min(x)}$$

در واقع هدف از نرمالیزه کردن این است که تمامی متغیرهای ورودی رفتار مشابهی را در مدل داشته باشند و نیازی نباشد مدل خود را با واحد (یکه‌ی) ورودی‌ها سازگار کند. هدف از استانداردسازی هم این است که میانگین را بر روی 0 و واریانس را بر روی 1 نگاشت کند و بدین صورت تمامی مجموعه‌های داده‌گان ورودی از یک ساختار مشابه برای آموزشِ شبکه استفاده می‌کنند و انعطاف شبکه را به مجموعه داده‌گان دیگر بالاتر برده و نتیجتا پیش‌بینی بهتری را ارائه می‌نماید.

1)

مطابق با خواسته‌ی صورت سوال از مدل رگرسیون خطی استفاده نمودیم و نتایج به صورت زیر بدست آمدند:

: Linear Regression

در ابتدا نتایج مربوط به خطاها را ارائه می‌کنیم که به صورت زیر می‌باشد:

MAE	MSE	RMSE
0.7094307439520315	0.8504541404665655	0.9222007050889549

و زمان آموزش نیز به صورت زیر است.

```
PS D:\dars\شبکه عصبی\تمرین ها\HW2_AbbasBadiei_810199106\Codes> python3 .\Q2_Linear_Regression.py
training time : 0.020999908447265625
```

زمان آموزش : 0.020999908447265625

قسمت امتیازی

مدل رگرسیون خطی به صورت زیر می‌باشد.

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ \dots + \beta_i X_i$$

Y : Dependent variable
β_0 : Intercept
β_i : Slope for X_i
X = Independent variable

در این مدل هر ورودی i بعدی وارد می‌شود و با داشتنِ ورودی و خروجیِ سیستم، اقدام به تخمین ضرایبِ بتا در معادله‌ی فوق می‌کنیم. به عنوان نمونه در مدل رگرسیون از تابعی مانند least square error به عنوان loss function استفاده می‌شود ولی این در حالی است که در ridge مینیم سازی خطا از طریق تابع زیر صورت می‌گیرد.

$$\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 + \lambda\sum_{j=1}^{p}\beta_j^2 = \mathrm{RSS} + \lambda\sum_{j=1}^{p}\beta_j^2,$$

در نهایت طبق خواسته‌ی بخش امتیازی اقدام به پیاده‌سازی مسأله رگرسیون با استفاده از ridge کردیم و نتایج به صورت زیر درآمد.

:Ridge

در ابتدا نتایج مربوط به خطاها را ارائه می‌کنیم که به صورت زیر می‌باشد:

MAE	MSE	RMSE
0.7094050377004908	0.8501810773922034	0.9220526435037228

و زمان آموزش نیز به صورت زیر است.

```
PS D:\dars\شبکه عصبی\تمرین ها\HW2_AbbasBadiei_810199106\Codes> python3 .\Q2_Rige.py
training time : 0.004996776580810547
```

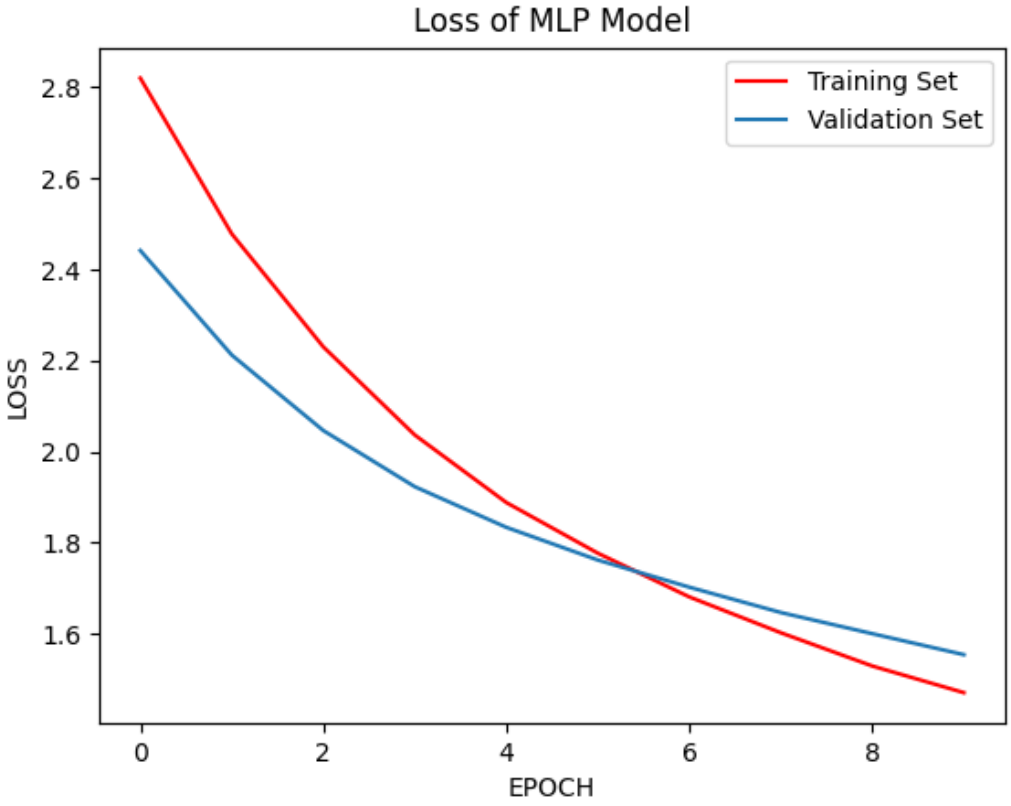
زمان آموزش : 0.004996776580810547

همانطور که مشاهده می‌کنیم از نظر زمانی بسیار نسبت به رگرسیون خطی سریعتر است و مقدار بسیار کمی هم خطای مدل را نسبت به رگرسیون خطی کاهش داده است.

(2

ابتدا نتایج را به تفکیک ذکر می‌کنیم و سپس مدلِ برنده در هر ایپاک را اعلام می‌نماییم.

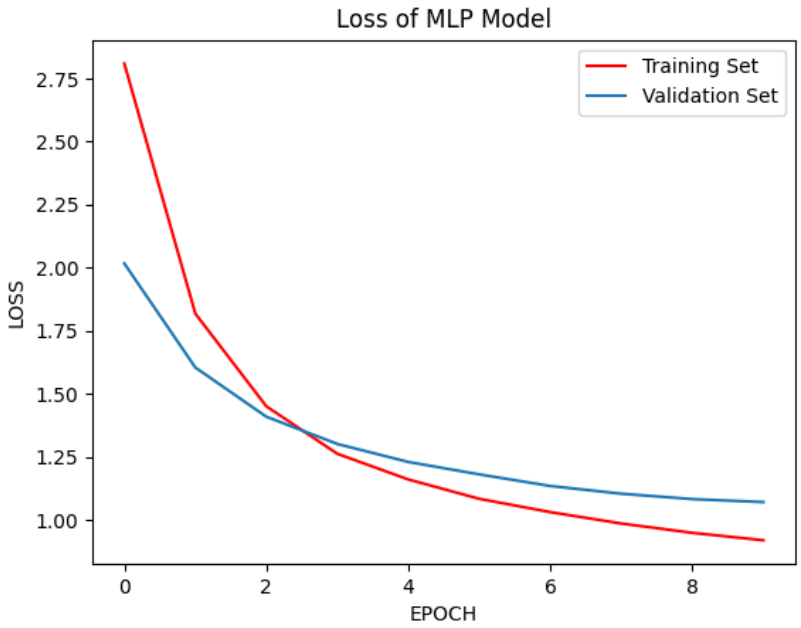
Epochs: 10 loss: mean squared error optimizer: Adam



MAE	MSE	RMSE
1.022081811113588	1.6289250373318747	1.2762934761769624

Time	1.7214252948760986

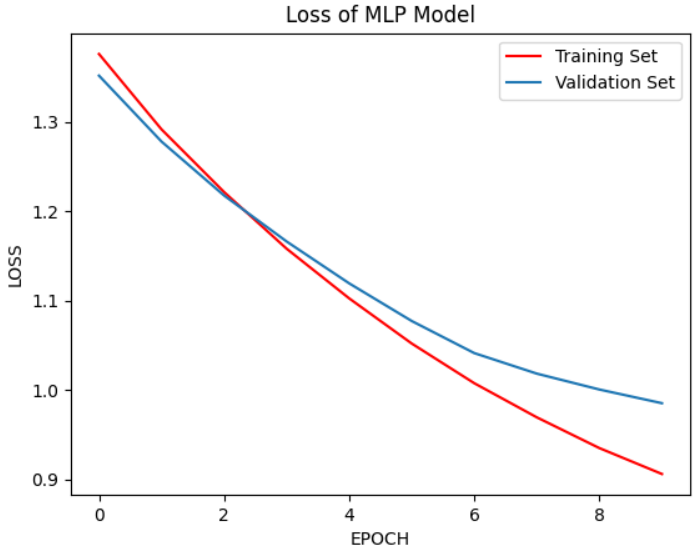
Epochs: 10 loss: mean squared error optimizer: SGD



MAE	MSE	RMSE
0.7952965028658998	1.0017483398965885	1.000873788195389

Time	1.596402883529663

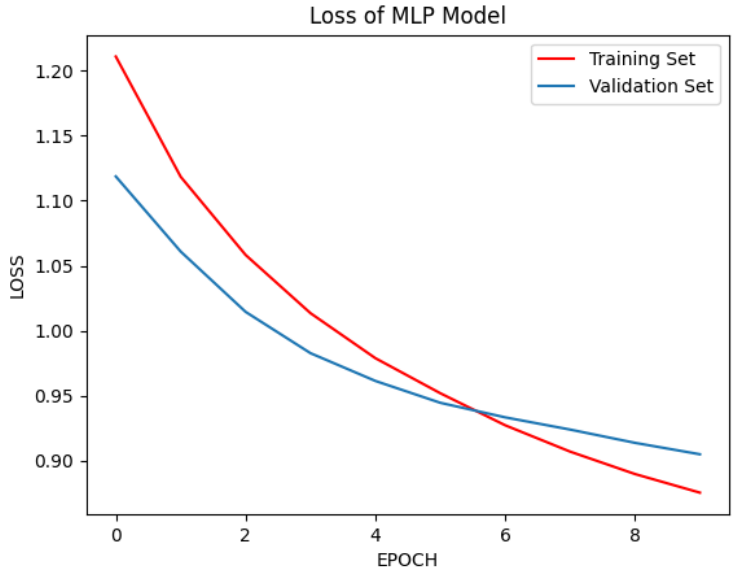
Epochs: 10 loss: mean absolute error optimizer: Adam



MAE	MSE	RMSE
0.9867619746187823	1.636308668610396	1.2791828128185572

Time	1.8223283290863037

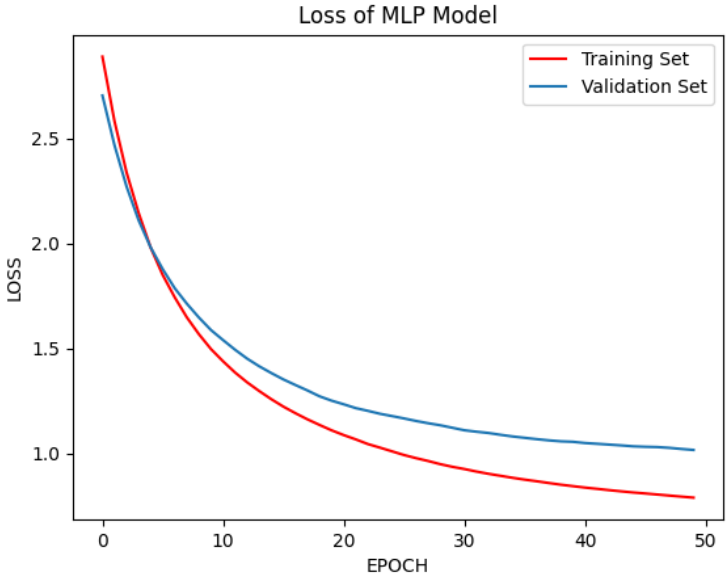
Epochs: 10 loss: mean absolute error optimizer: SGD



MAE	MSE	RMSE
0.8750038186066829	1.2658150607750362	1.125084468284509

Time	2.2265636920928955

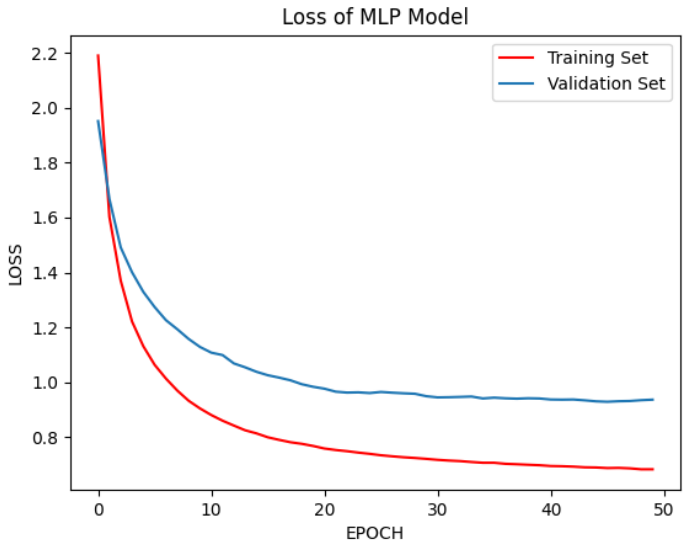
Epochs: 50 loss: mean squared error optimizer: Adam



MAE	MSE	RMSE
0.7884283636366693	0.9819478964385433	0.990932841537984

Time	5.393349885940552

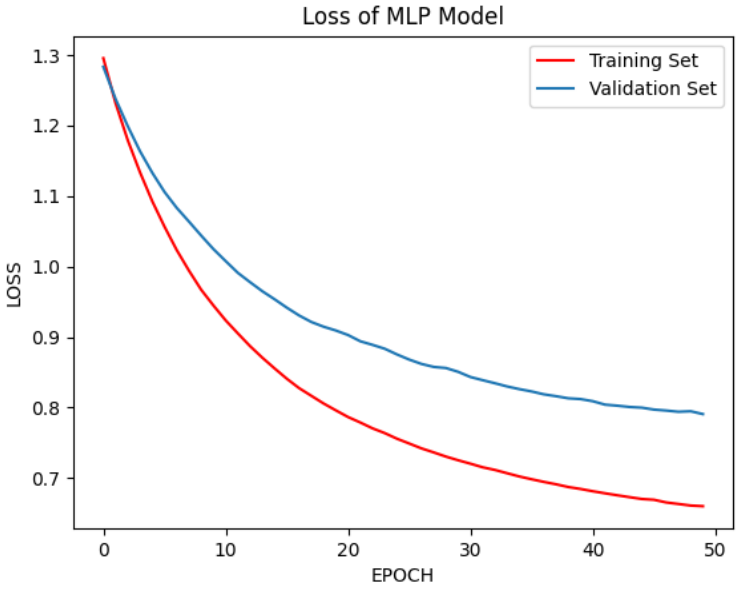
Epochs: 50 loss: mean squared error optimizer: SGD



MAE	MSE	RMSE
0.7005624700105122	0.8191064658454639	0.9050450076352358

Time	5.292332887649536

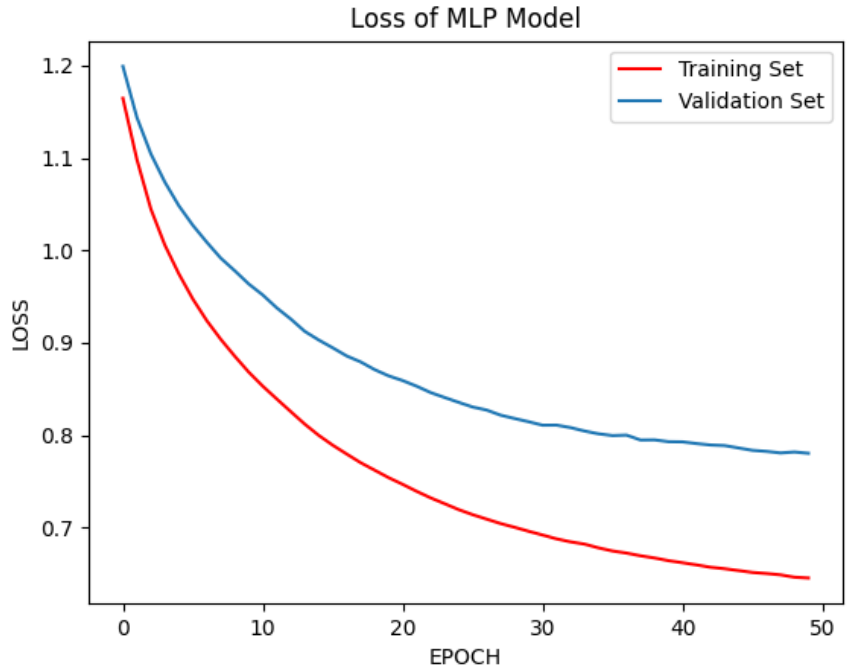
Epochs: 50 loss: mean absolute error optimizer: Adam



MAE	MSE	RMSE
0.7239272363183094	0.9253459744414516	0.9619490498157642

Time	5.373349189758301

Epochs: 50 loss: mean absolute error optimizer: SGD



MAE	MSE	RMSE
0.721958736508334	0.871489139409797	0.9335358265271864

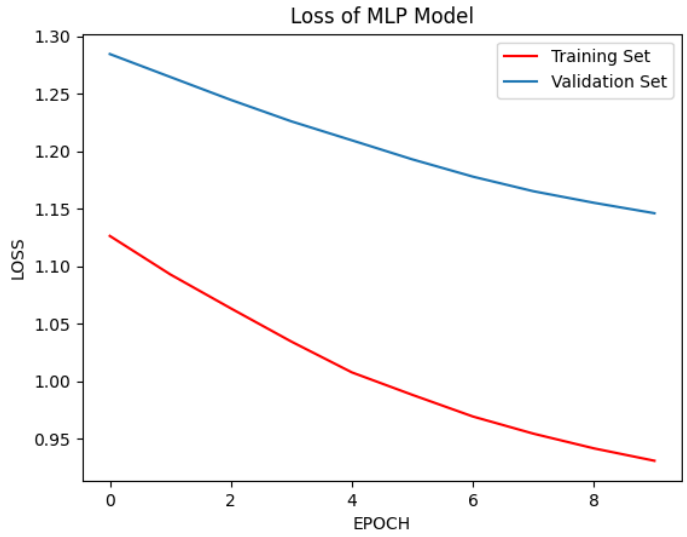
Time	5.299809217453003

همانطور که مشاهده می‌نمایید بهترین نتایج در این بخش چه برای 10 ایپاک و چه برای 50 ایپاک به ازای تابع لاس mean square error و بهینه‌ساز SGD حاصل شد. مسلما از نظر زمانی نیز 50 ایپاک زمان بیشتری را نسبت به 10 ایپاک صرف میکند. از لحاظ مقایسه با بخش الف که رگرسیون خطی بود خطاها به طور کلی مشابه با مدل رگرسیون و البته کمی بیشتر نیز بودند و همچنین مدل رگرسیون خطی نیز از نظر زمانی بسیار مناسب تر برای آموزش مدل بود. با این حال اگر بحث زمان را کنار بگذاریم به نظر می‌رسد که اگرچه مدل شبکه عصبی کمی ضعیف تر عمل کرده و چه بسا بدلیل نداشتنِ لایه‌ی پنهان باشد ولی به طور کلی بهترین نتیجه در مدل شبکه عصبی تواسته بهبودی کمی نسبت به رگرسیون حاصل کند و این موضوع کارایی مدل‌های شبکه عصبی در مباحث رگرسیون را به خوبی نشان می‌دهد.

3)

تابع فعالسازی که مورد استفاده قرار دادیم softsign است. این تابع را با توجه به بیش‌برازشهایی که بر روی داده‌گان انجام دادیم انتخاب نمودم. نتایج به صورت زیر است.

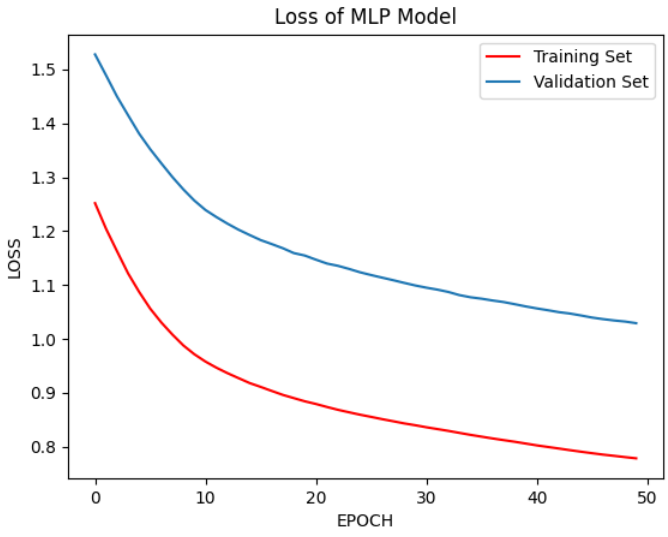
به ازای 10 ایپاک:



MAE	MSE	RMSE
0.7571599651187197	0.9083784490091988	0.9530888987965387

Time	2.881725549697876

به ازای 50 ایپاک نیز نتایج به صورت زیر است:



MAE	MSE	RMSE
0.7166973755126269	0.8326418977855825	0.9124921357390334

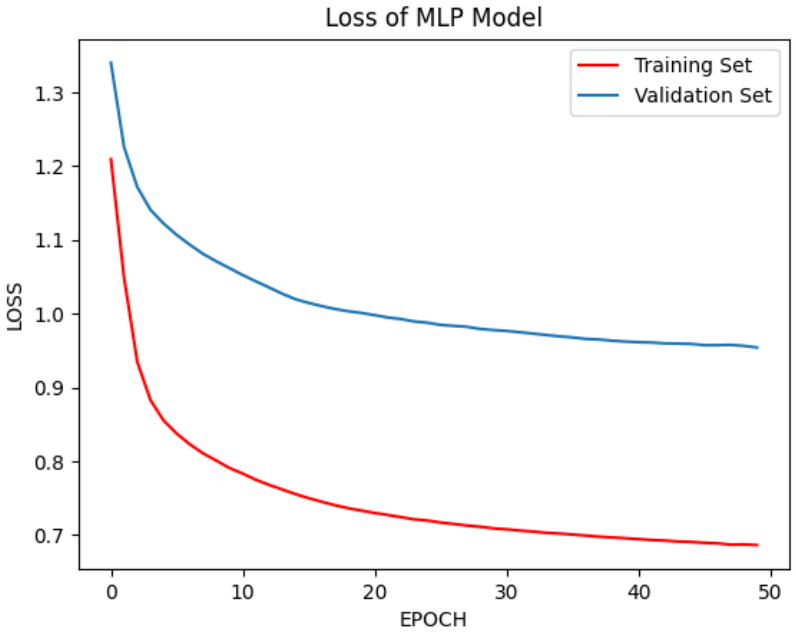
Time	6.86773157119751

همانطوری که مساهده می‌کنید به ازای 10 ایپاک بهبودی داشتیم و به ازای 50 ایپاک هم تقریبا همان نتایج را گرفتیم ولی چیزی که مسلم است این می‌باشد که تغییر زیادی با بهترین نتایج در مرحله قبل ایجاد نشده و از طرفی زمان آموزش نیز که بخاطر غیر خطی بودنِ تابع فعالساز است افزایش یافته. لذا به نظر می رسد مدل‌های مرحله قبل مدل‌های بهتری در ایپاک‌های بالا می‌باشند.

4)

این بخش را با سه سایز بچِ 16، 64 و 256 مورد ارزیابی قرار می‌دهیم. نتایج به صورت زیر است.

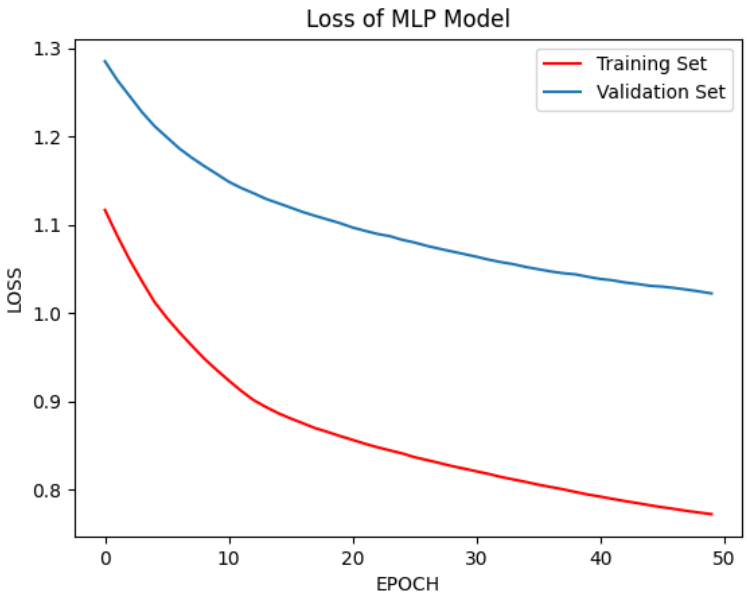
به ازای 16 ایپاک:



MAE	MSE	RMSE
0.706021674317693	0.8188837481913276	0.904921956961664

Time	11.969998598098755

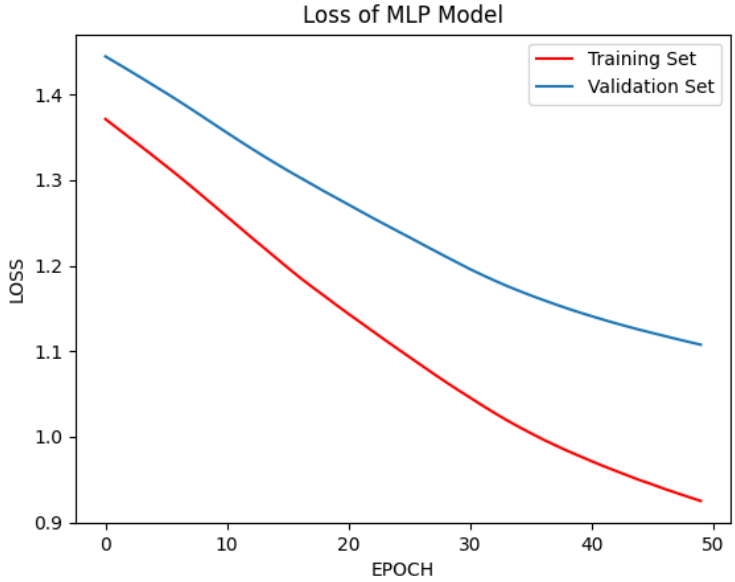
به ازای 64 ایپاک:



MAE	MSE	RMSE
0.718845912313919	0.8015927205182738	0.8953171061240112

Time	5.846472978591919

به ازای 256 ایپاک:



MAE	MSE	RMSE
0.729715697001697	0.8607828495547905	0.9277838377309612

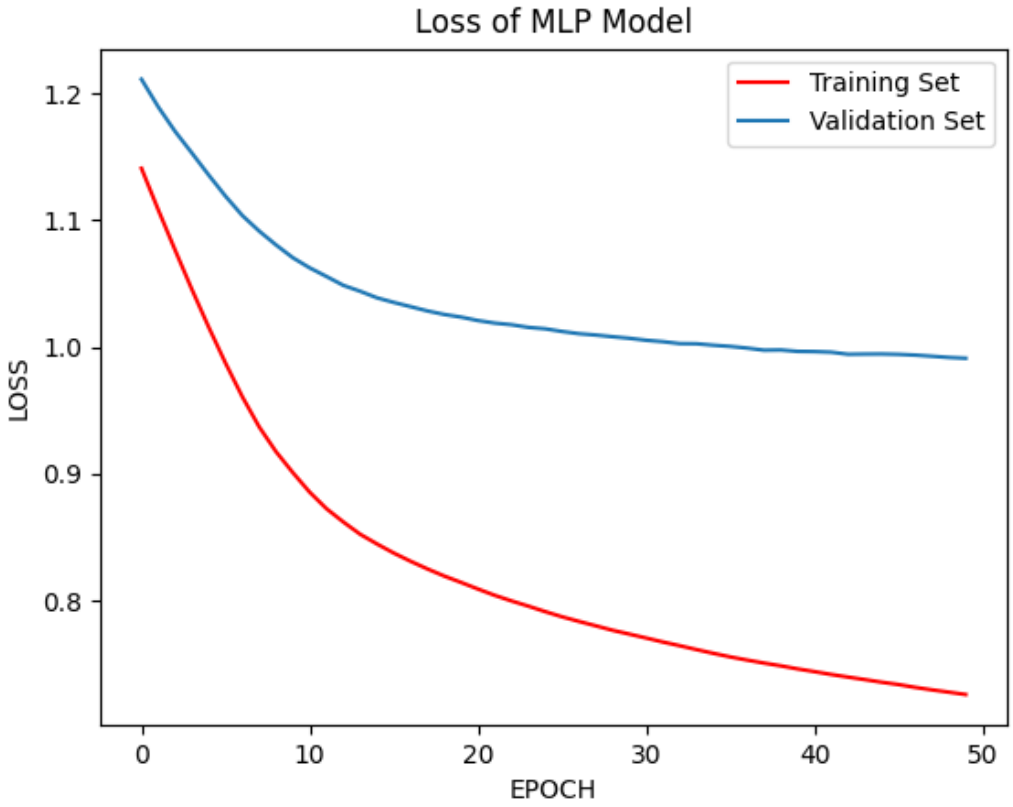
Time	4.793204307556152

بله اندازه‌ی سایز بچ موثر بوده و همانطور که مشاهده می‌نمایید با افزایشِ batch size توابع خطای مدل کاهش یافته و البته از قبل هم دیده بودیم که زمانِ آموزش مدل کاهش می‌یابد.

5)

برای یک لایه‌ی پنهان نتایج به صورت زیر است:

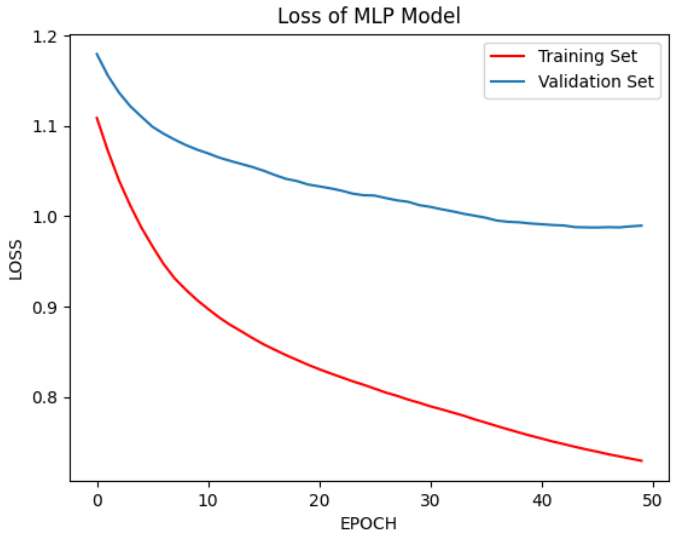
Loss: mean squared error optimizer: Adam



MAE	MSE	RMSE
0.6775157985665372	0.7613731199343249	0.8725669716040855

Time	5.128291130065918

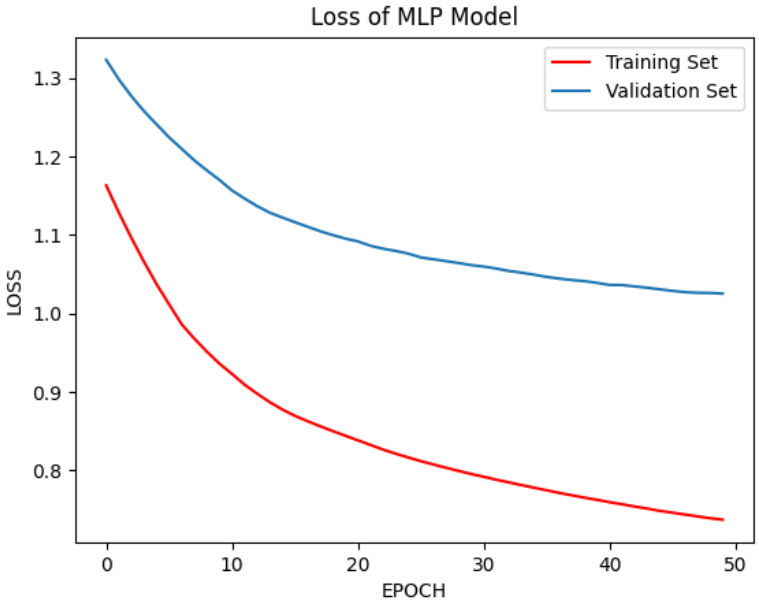
Loss: mean squared error optimizer: SGD



MAE	MSE	RMSE
0.7134932951092396	0.8276804009129237	0.9097694218388106

Time	5.234323978424072

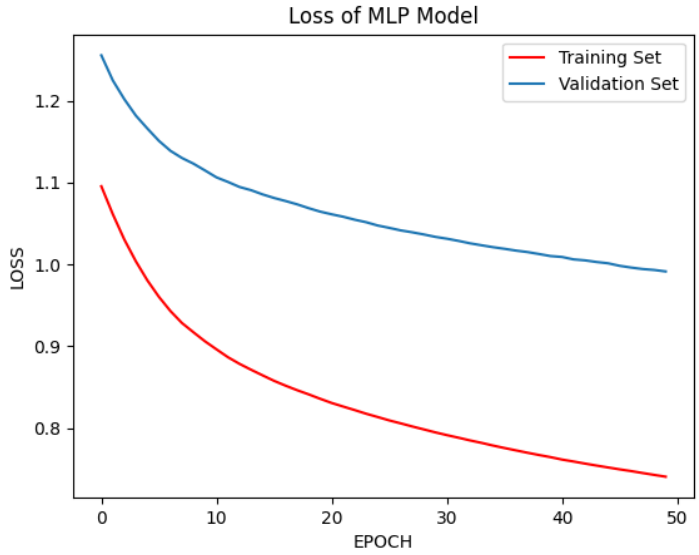
Loss: mean absolute error optimizer: Adam



MAE	MSE	RMSE
0.6953376358746601	0.8141145242986072	0.9022829513509646

Time	5.547208309173584

Loss: mean absolute error optimizer: SGD

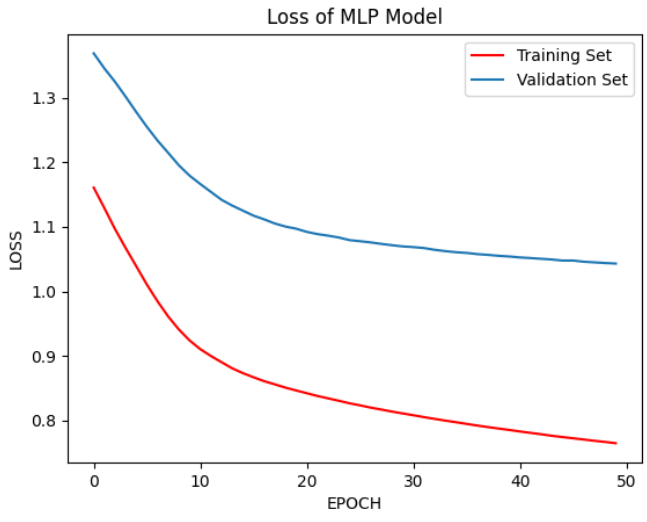


MAE	MSE	RMSE
0.7005629949424501	0.7823322593250365	0.8844954829308268

Time	5.235307455062866

و برای دو لایه‌ی پنهان نیز نتایج به صورت زیر درامدند:

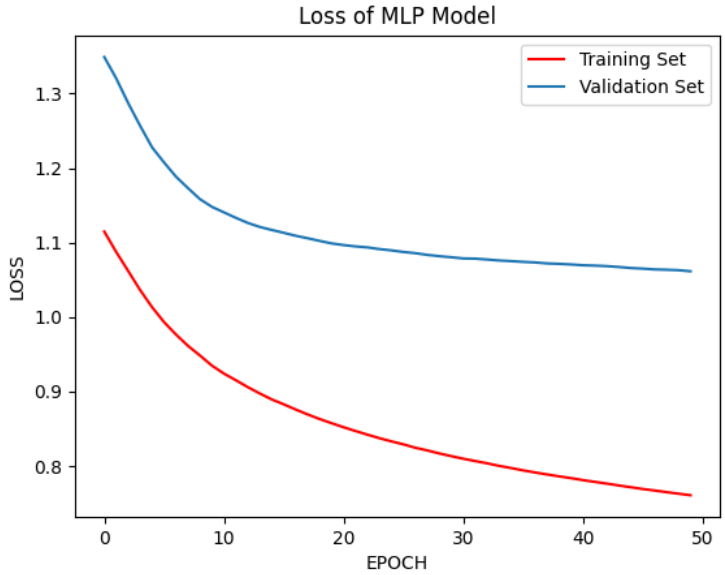
Loss: mean squared error optimizer: Adam



MAE	MSE	RMSE
0.715050866670079	0.8059572437007853	0.8977512148144302

Time	5.511540412902832

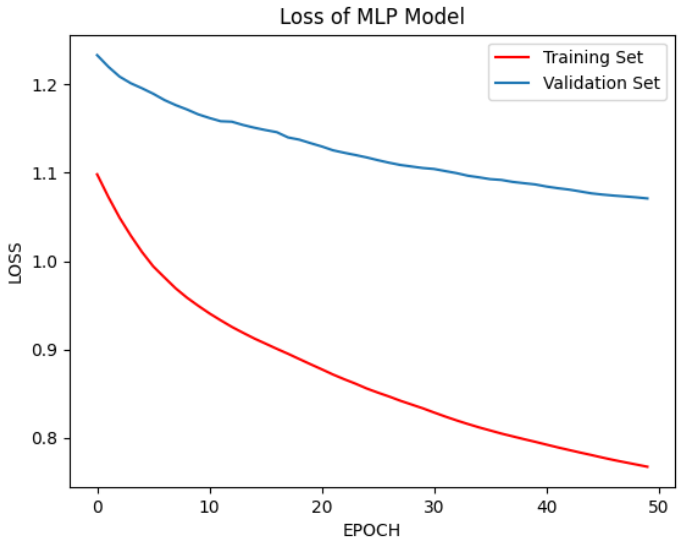
Loss: mean squared error optimizer: SGD



MAE	MSE	RMSE
0.7181826498722128	0.8034934308622596	0.8963779509014373

Time	5.061149835586548

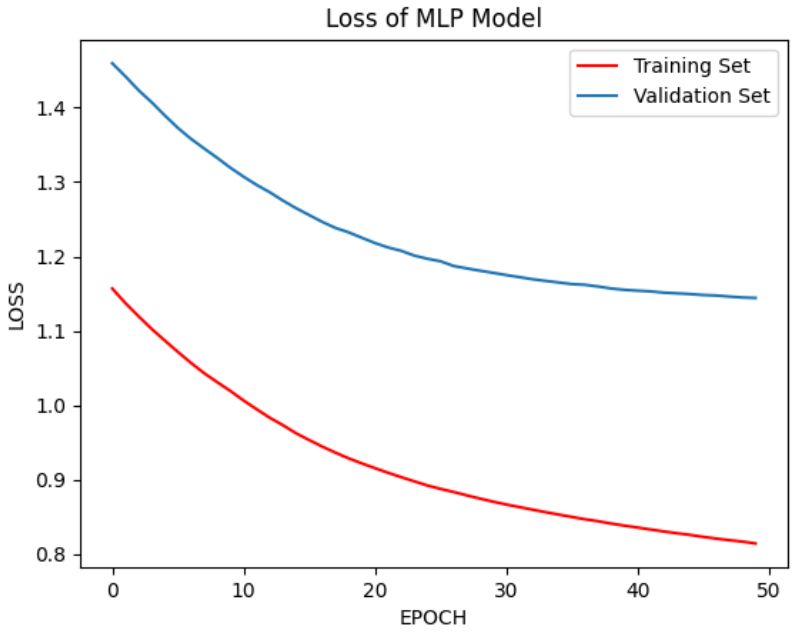
Loss: mean absolute error optimizer: Adam



MAE	MSE	RMSE
0.7002523814721737	0.7504337749910599	0.8662758076912109

Time	5.026263475418091

Loss: mean absolute error optimizer: SGD



MAE	MSE	RMSE
0.7210696924418537	0.8201046529612395	0.9055962969012403

Time	5.158216714859009

مشاهده می‌نماییم که با اضافه کردنِ یک لایه، مدل برنده به ازای Loss: mean squared error و optimizer: Adam بهترین خروجی را از منظر کاهش خطا نشان داد و با اضافه کردن دو لایه نیز مدل برنده با انتخاب Loss: mean absolute error و optimizer: Adam حاصل شد. همچنین مشاهده می‌کنیم که هر اندازه تعداد لایه‌ها بیشتر، نتیجتا تعداد پارامترهای مسأله افزایش می‌یابد و زمان آموزش نیز بیشتر می‌شود. تفاوتی که با مدل پیشین مشاهده می‌نماییم این است که با اضافه کردن لایه‌ی پنهان خطا کاهش یافته و نیز مدل برنده نیز در بهینه‌ساز برای یک لایه و در بهینه ساز و تابعِ خطا برای دو لایه با مدل پیشین

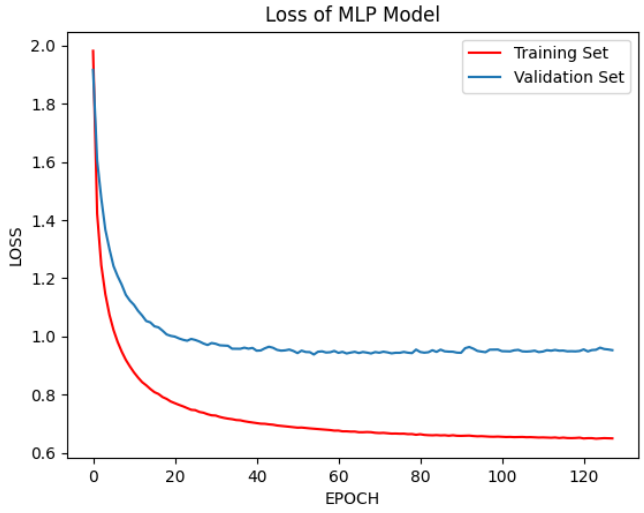
تفاوت دارد. نتیجتا اضافه کردنِ لایه‌ها هر چند پیچیدگی زمانی را افزایش داده ولی به کاهشِ خطای مدل کمک نموده است و این یک مزیت است.

برای تعیین تعداد نورون‌های لایه‌ی پنهان الگوی ثابتی وجود ندارد و گاها بر حسب تجربه و یا استفاده از الگوریتم‌های سرچ هیوریستیک یا شبکه‌ای و یا حریصانه و حتی تصادفی اقدام به یافتنِ یک جواب مناسب برای تعداد نورون‌های لایه‌ی مخفی می‌نمایند. با این حال طبق خواسته‌ی مسأله ما یک روش را از یکی از مقالات در آورده ایم و اراده می‌کنیم. در مقاله‌ی مذکور نشان داده است که یکی از راه‌های مناسب برای تعداد لایه‌ها و تعداد نورن‌های لایه‌های مخفی استفاده از یک شبکه‌ی MLP دولایه است که لایه‌ی پنهان اول به تعداد ورودی‌ها نورون دارد و لایه‌ی پنهان دوم به تعداد دو برابر ورودی علاوه‌ی دو، نورون را دارا می‌باشد و سپس از توابع فعالساز سیگموید استفاده کردند که محققان مکتشفِ این روش اثبات نمودند که چنین شبکه‌ای قادر است هر تابع پیوسته‌ی چند جمله‌ای را با هر میزان دقت دلخواه، تخمین بزند.

6)

ابتدا نتایج را بررسی می‌کنیم و سپس به تحلیل می‌پردازیم. طبق خواسته‌ی مسأله تعداد ایپاک‌ها را افزایش دادیم و به 128 ایپاک رساندیم و نهایتا مدل‌های برنده را در سه حالت بدون لایه پنهان و یک لایه پنهان و دو لایه‌ی پنهان به مدل دادیم. ئ نیز طبق خواسته‌ی سوال یک dropout(0.25) را برای سیستم قرار دادیم.

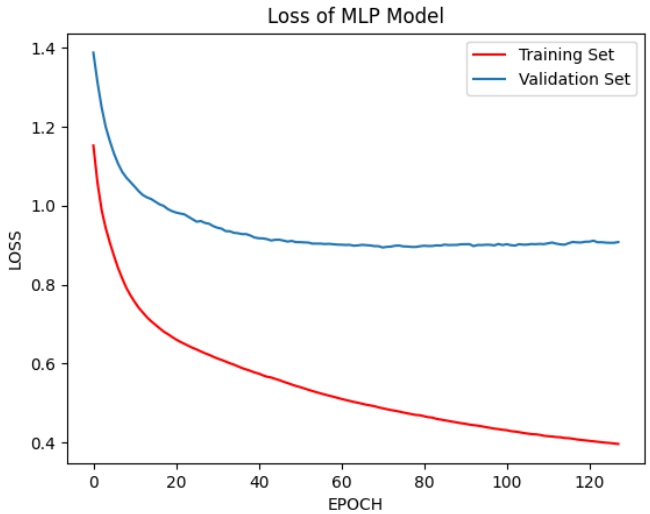
نتایج برای مدل برنده‌ی بدون لایه‌ی پنهان:



MAE	MSE	RMSE
0.7039255636077715	0.8266500047359633	0.9092029502459631

Time	11.564171314239502

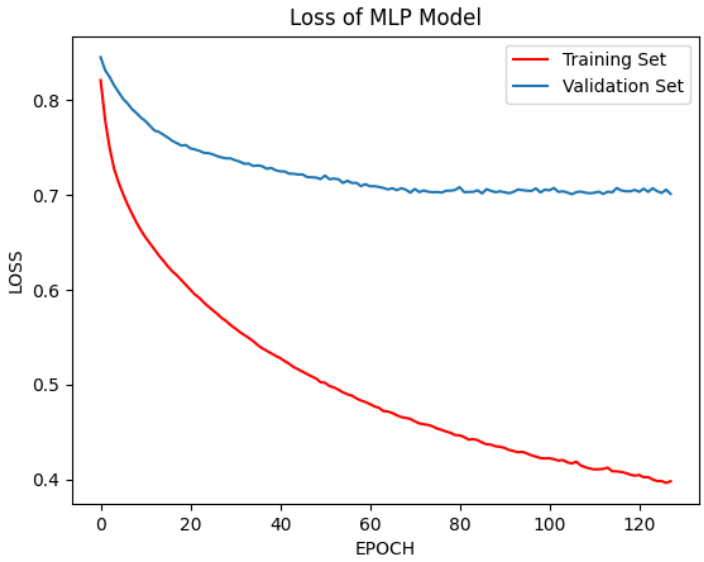
نتایج برای مدل برنده‌ی با یک لایه‌ی پنهان:



MAE	MSE	RMSE
0.6790856144575269	0.8023515839539996	0.8957408017691276

Time	11.608914613723755

نتایج برای مدل برنده‌ی با دو لایه‌ی پنهان:



MAE	MSE	RMSE
0.6658892994260462	0.8256575579618133	0.9086570078758064

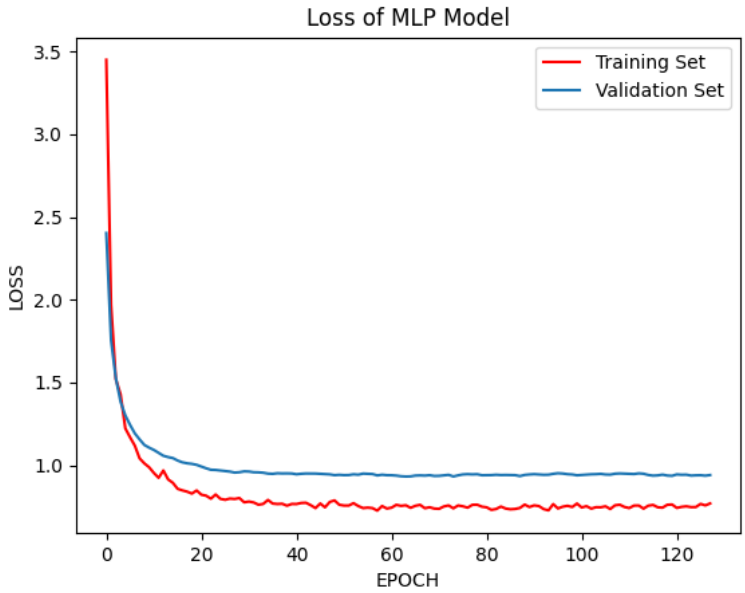
Time	12.373781681060791

همانطور که مشاهده می‌کنید در مدل‌های برنده‌ی هر بخش با اضافه نمودنِ dropout بهبودی حاصل شده و خطا کاهش یافته است. دلیل این امر هم به dropout مربوط می‌شود که در هر مرحله از آموزش نورون‌های لایه‌هایی که به آنها اضافه شده است را با یک احتمالِ خاصی حذف می‌کند. بدین ترتیب همه‌ی پارامترها با یکدیگر آپدیت نشده و نسبت به ورودی‌های جدید انعطاف نشان می‌دهند و برخی آپدیت شده و برخی نیز آپدیت نمی‌شوند. نتیجتا همین عامل باعث می‌شود که شبکه را از over fitting دور کند که از مهم‌ترین مزایای dropout است.

7)

ابتدا نتایج را بررسی می‌کنیم و سپس به تحلیل می‌پردازیم. طبق خواسته‌ی سوال در ادامه‌ی بخش قبلی BatchNormalization را اضافه می‌نماییم. نتایج برای هر قسمت به صورت زیر است.

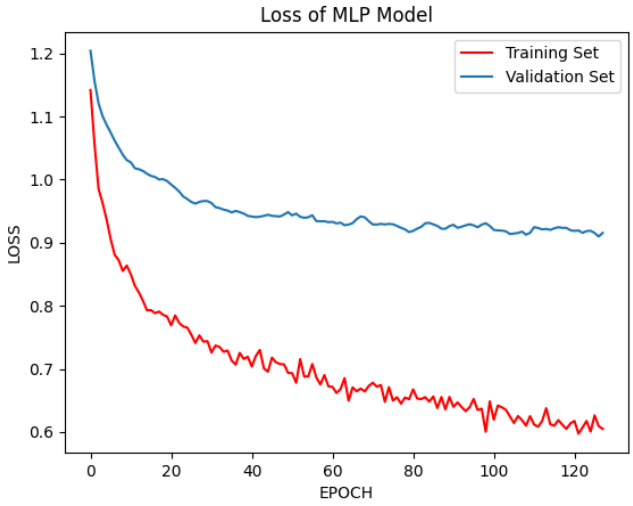
نتایج برای مدل برنده‌ی بدون لایه‌ی پنهان در قسمت قبل:



MAE	MSE	RMSE
0.6991318770739916	0.7726866121862179	0.8790259451155114

Time	12.698858976364136

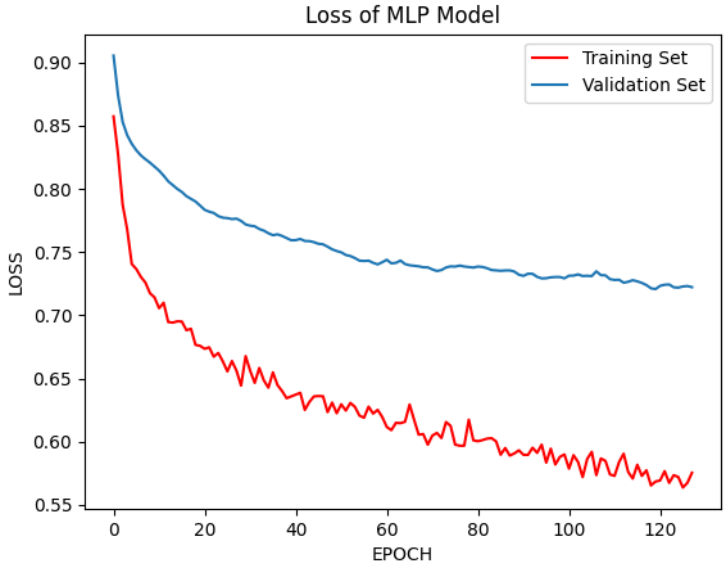
نتایج برای مدل برنده با یک لایه‌ی پنهان در قسمت قبل:



MAE	MSE	RMSE
0.6445254015568336	0.7073156897059389	0.84102062382913

Time	14.11127495765686

نتایج برای مدل برنده با دو لایه‌ی پنهان در قسمت قبل:



MAE	MSE	RMSE
0.6327385493613061	0.7305723141678968	0.8547352304473572

Time	13.719094514846802

همانطور که مشاهده نمودیم با اضافه کردن BatchNormalization به مدل توانستیم خطا را مجددا تا حد خوبی در حدود چند صدم کاهش دهیم. لذا هر سه مدل برنده در مسأله‌ی ما در این بخش بهبودی پیدا کردند. حال اندازه دسته را کمی بیشتر کردیم و به ترتیب نتایج زیر برای خطا بدست آمده. با توجه به این نتایج می‌توان به طور کلی نتیجه گرفت که افزایش سایز دسته‌ها می‌تواند عملکرد BatchNormalization را تا حد کمی موثرتر کرده و بهبودی کمی را برای شبکه حاصل کند. (البته نباید خیلی بزرگ انتخاب کنیم که در بهینه‌های محلی گیر بیافتد)

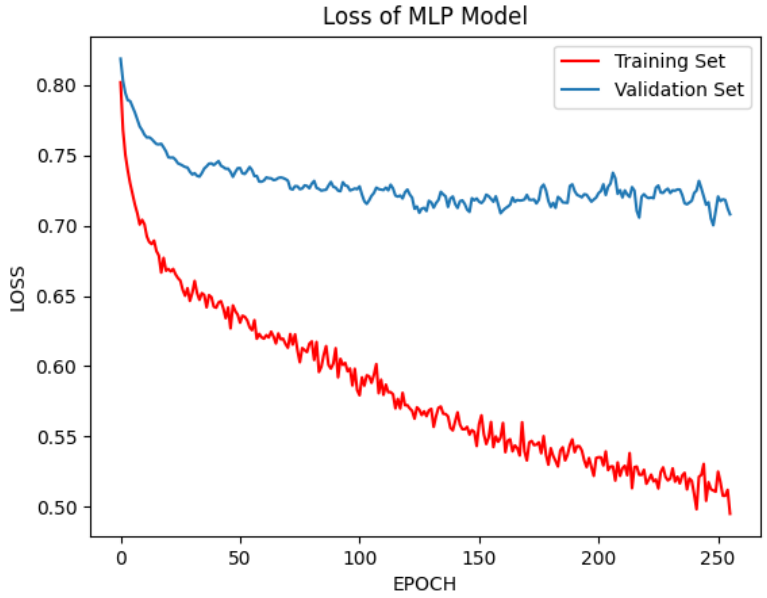
MAE	MSE	RMSE
0.7002774229944957	0.7828718581762906	0.8848004623508572

MAE	MSE	RMSE
0.6371981965448709	0.709969327084363	0.842596776094214

MAE	MSE	RMSE
0.6233804379282633	0.7406298933606175	0.8605985669059747

8)

با توجه به خواسته‌ی مسأله در ابتدا سه لایه پنهان برای مدل خود در نظر می‌گیریم. سپس با توجه به نتایجی که تابع فعالساز softsign از خود بروز داد، همه‌ی توابع فعالساز را به صورت softsign در نظر می‌گیریم و از طرفی به مدل dropout(0.2) و نیز BatchNormalization را اضافه می‌کنیم. بهینه‌ساز را adam در نظر گرفته و loss function را MAE انتخاب می‌کنیم. همچنین تعداد ایپاک را مشابه با مرحله‌ی قبل یعنی 256 در نظر می‌گیریم زیرا تاثیر مثبت این تعداد ایپاک را در کاهش خطای مدل دیدیم. میزان batch-size را همان 64 که نتیجه‌ی قسمت‌های قبل بود لحاظ کرده و شبکه را با لحاظ هایپرپارامترهای فوق آموزش می‌دهیم. نتایج که بهترین نتیجه بین تمامی بخش‌ها است به صورت زیر در می‌آید.



MAE	MSE	RMSE
0.6162198475358884	0.7548687140298582	0.8688318099781213

Time	26.37018847465515

نتایج مدل انتخابی با بهترین پارامترهای نتیجه گرفته شده در مراحل قبل

همانطور که مشاهده می‌فرمایید این مدل در بین تمامی بخش‌ها بهترین عملکرد را از خود نشان داده است. در نهایت متوجه شدیم که یک مدل مبتنی بر استدلال‌هایی که در این سوال داشتیم می‌تواند برای انجام عملیات‌ها رگرسیون خوب و بلکه عالی تر از مدل‌هایی مانند linear regression یا ridge عمل کند که این کارایی شبکه‌های عصبی در کاربردهای رگرسیون را نشان می‌دهد.

سوال 3 – آشنایی با کاهش بُعد

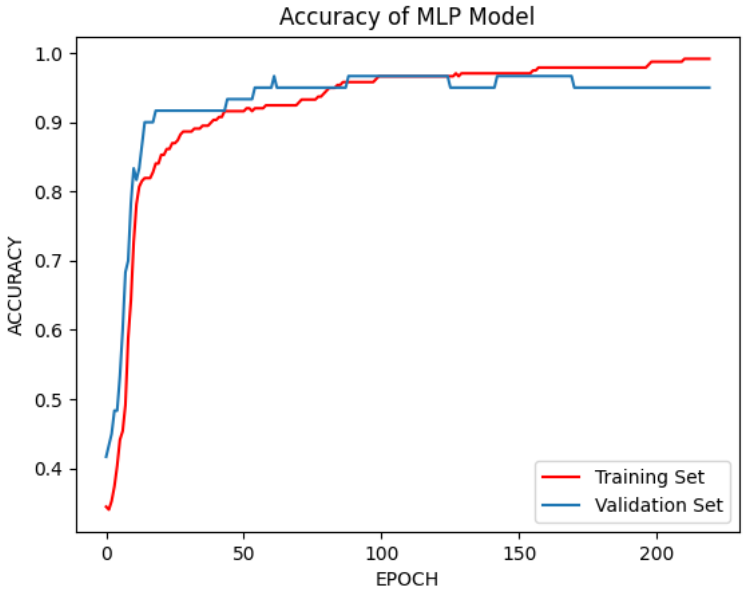
طبق صورت سوال می‌بایست این دو بخش را ابتدا برای مدل برنده‌ی سوال یک و سپس مدل برنده‌ی سوال دو پیاده‌کنیم.

طبقه بندی داده‌های سوال یک به همراه autoencoder:

ابتدا به پیاده‌سازی autoencoder پرداختیم و مدل برنده‌ای را که برای سوال یک انتخاب کرده بودیم را به همراه این کاهنده‌ی ابعاد و با شبکه‌ی MLP بر روی مجموعه داده‌گان این سوال اعمال نمودیم.

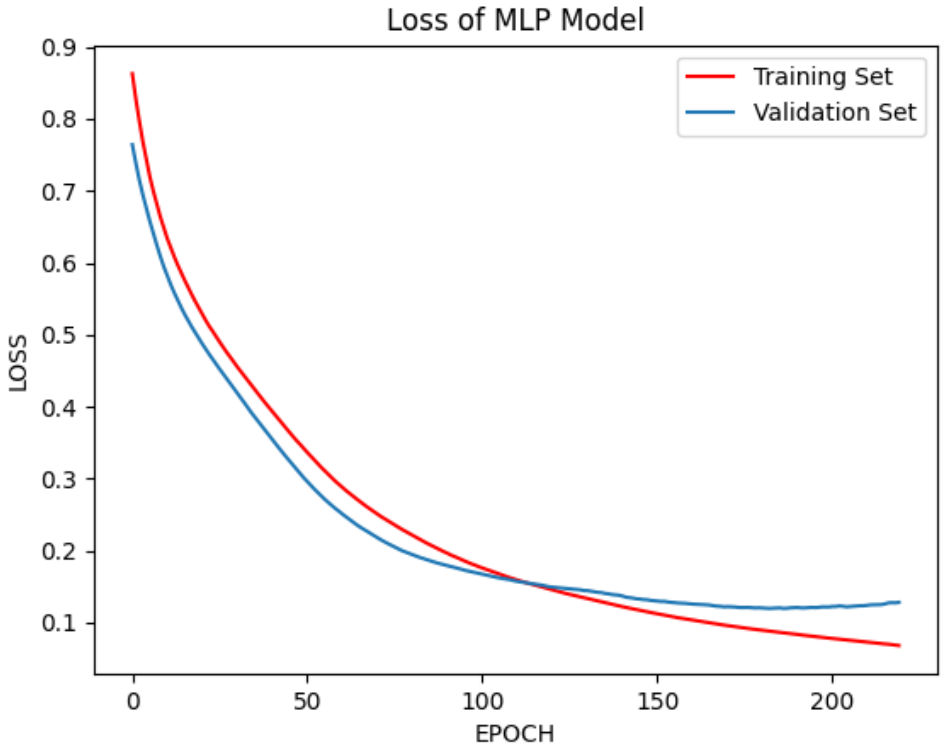
برای کاهش ابعاد نیز راه‌حل های متفاوتی در این روش وجود دارد که با توجه به اینکه این الگوریتم dimension reduction مبتنی بر مدل شبکه عصبی است می‌توانیم تعداد نورون‌های لایه‌ی پنهان را (لایه‌ی پنهان وسط ابعاد جدید داده‌گان را نشان می‌دهد) را تغییر می‌دهند که به کمترین خطای تست برسند. البته هایپرپارامترهای دیگر نیز از جمله متغیرهای موثر در کاهش خطا و حفظ اطلاعات مفید در شبکه autoencoderها هستند. بدین ترتیب تعداد نورون (به عنوان یکی از هایپرپارامترها) برای لایه‌ی پنهان با در نظر داشتن هایپرپارامترهای دیگر و خروجی‌های های شبکه (صحت و دقت و ...) انتخاب می‌شود. طبیعتا این تعداد نورون لایه‌ی پنهان وسط، ابعاد جدید داده‌گان هستند.

نتایج به صورت زیر بدست آمد:



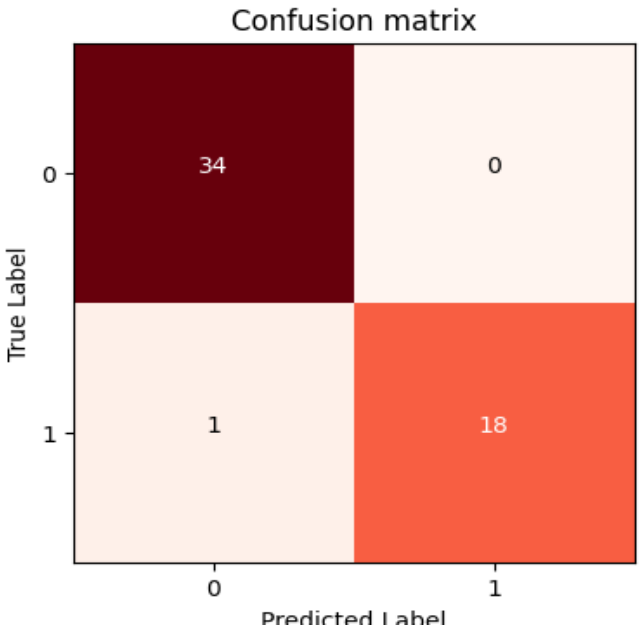
نمودار صحت مدل با عملیات کاهش ابعاد با autoencoder

سپس نمودار خطای مدل را مورد ارزیابی قرار دادیم که به صورت زیر درامد:



نمودار خطای آموزش و ارزیابی مدل با عملیات کاهش ابعاد با autoencoder

سپس ماتریس آشفتگی را رسم نمودیم:



ماتریس آشفتگی مدل با عملیات کاهش ابعاد با autoencoder

زمان آموزش:

Time	15.964470386505127

و نتایج مدل بر روی داده‌گان تست در مقایسه با نتایج بخش اول برای مدل برنده به صورت زیر است:

	with autoencoder	without autoencoder
Accuracy	0.9811320900917053	0.9433962106704712
Loss	0.11537490040063858	0.14462608098983765
Precision	1.0	1.0
Recall	0.9473684210526315	0.8421052631578947
Fscore	0.972972972972973	0.9142857142857143
Time	15.964470386505127	21.051527738471167

همانطور که مشاهده می‌کنید نتایج برای مدل برنده در این بخش نسبت به سوال اول بهبودی داشته است. دلیل این بهبودی می‌تواند حذف اطلاعات غیر مفید از داده‌گانی باشد که همبستگی بالایی دارند و نتیجتا خطای مدل نیز کاهش داشته است. همچنین زمان نیز در این روش کاهش یافته است که بدلیل کاهش تعداد نورون‌های ورودی و کاهش تعداد پارامترهایی است که قرار است مورد آموزش قرار گیرند.

طبقه بندی داده‌های سوال یک به همراه PCA:

در رابطه با روش pca همانطور که می‌دانیم، داده‌گان از مختصات ویژگی‌هایشان به یک مختصات جدیدی که شامل ابعاد (ویژگیهای) کمتر است نگاشت می‌کند. مبنای این روش بدین صورت است که مولفه‌هایی از داده‌گان که بیشترین تاثیر را در واریانس دارند حفظ میکند و مابقی ستون ویژگی‌ها را وابسته به سایز ابعاد از ستون‌های نمونه‌ها حذف می‌نماید. در ابتدا ماتریس داده‌گان را به صورت X در نظر می‌گیریم و تجزیه منفرد آن را محاسبه می‌نماییم.

$$Y^T = X^T W = V\Sigma$$

سپس میانگین تجربی را محاسبه کرده و بر روی مجموعه داده‌گان به صورت زیر اعمال می‌کنیم.

$$u[m] = \frac{1}{N}\sum_{i=1}^{۷۲} X[m,i]$$

$$B = X - uh$$

که h برداری با اندازه با مقدار ۱ در هرکدام از درایه‌ها است. سپس ماتریس کواریانس را به صورت زیر محاسبه می‌نماییم:

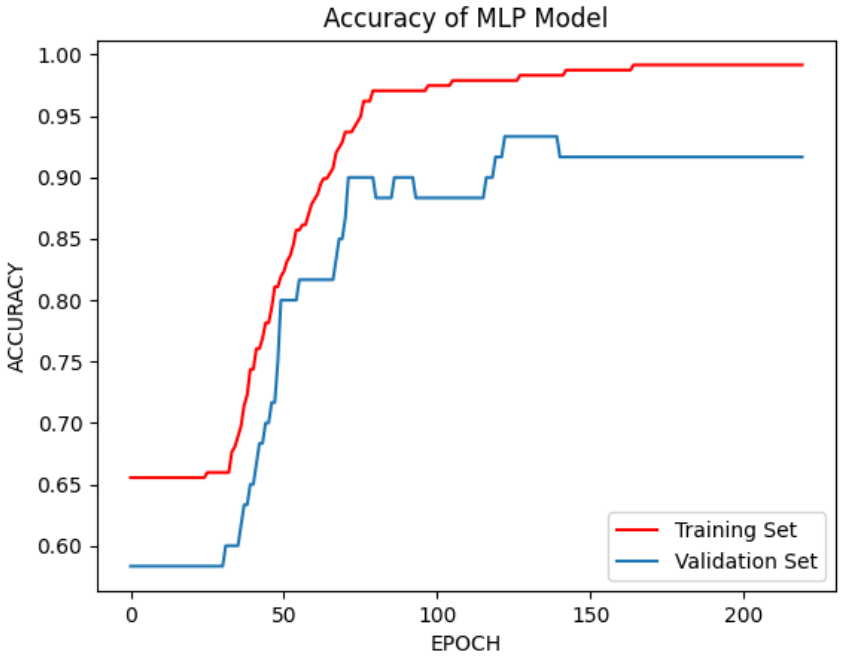
$$C = \mathbb{E}[B \otimes B] = \mathbb{E}[B \cdot B^*] = \frac{1}{N} B \cdot B^*$$

سپس مقادیر ویژه و بردار ویژه را محاسبه می‌نماییم.

$$V^{-1}CV = D$$

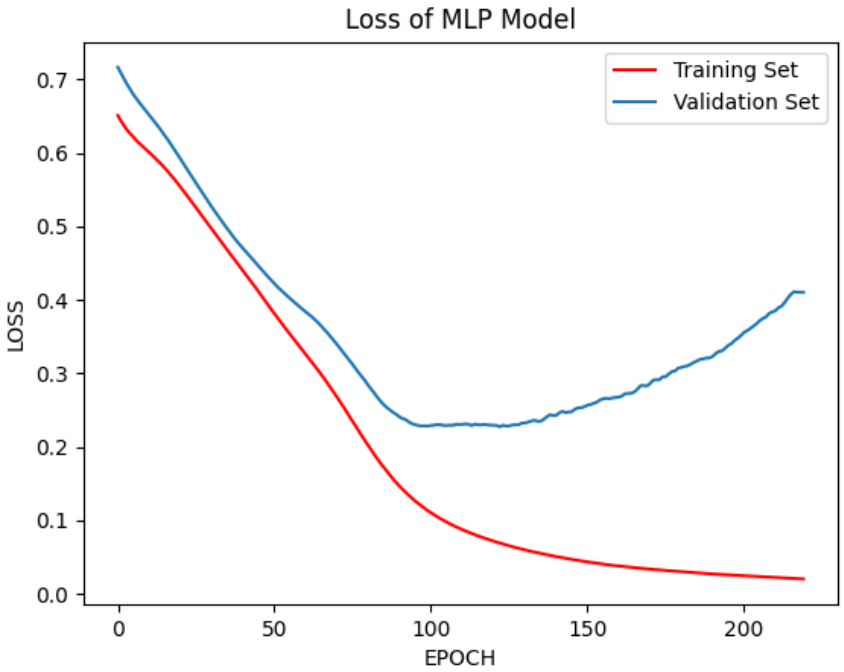
سپس زیرمجموعه‌ای از بردارهای ویژه را با در نظر داشتن مقدار ویژه مربوطه حفظ خواهیم کرد و بدین صورت ستون‌های حاوی اطلاعات انتخاب خواهند شد. تعداد این ستون‌ها می‌تواند بستگی به سایز تعیین شده و حد مشخصی واریانس در بین مقادیر یک مولفه‌ی مجموعه‌ی داده‌گان باشد. در رابطه با کاهش ابعاد همانطور که گفتیم، می‌بایست همبستگی بین مجموعه‌ی مولفه‌های دیتاهای موجود در دیتاست را بررسی کرده و اطلاعاتی که مفید نیستند را از مجموعه حذف کنیم که در واقع این کار با بررسی واریانس مولفه‌های داده‌گان انجام می‌پذیرد. در واقع با اینکار ستون‌های ویژگی‌هایی را انتخاب می‌نماییم که یا از یکدیگر مستقل هستند یا بیشترین واریانس را در مجموعه‌ی خودشان (همان ویژگی) دارند.

حال خروجی‌های مدل برنده‌ی سوال یک را که در این بخش با PCA ابعادش کاهش یافته ارائه می‌کنیم و سپس با سوال یک مقایسه می‌کنیم.



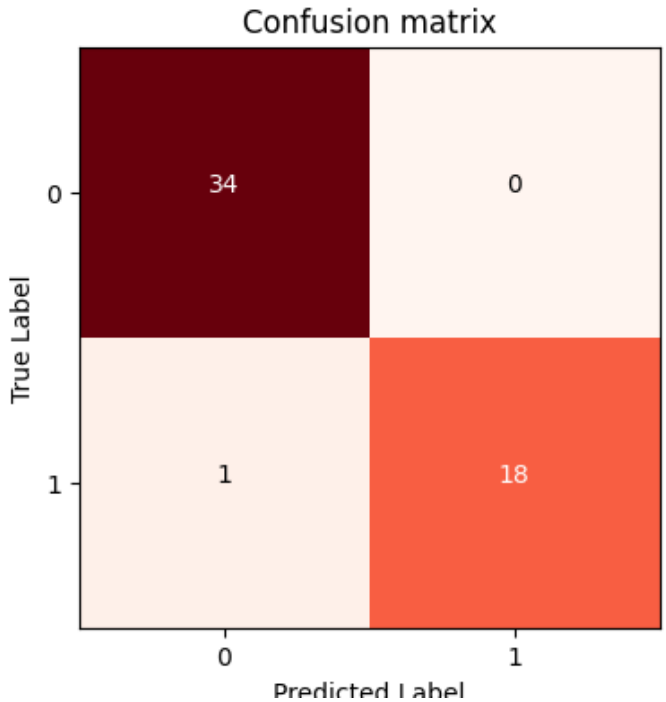
نمودار صحت مدل با عملیات کاهش ابعاد با PCA

سپس نمودار خطای مدل را مورد ارزیابی قرار دادیم که به صورت زیر درامد:



نمودار خطای آموزش و ارزیابی مدل با عملیات کاهش ابعاد با PCA

سپس ماتریس آشفتگی به صورت زیر است.



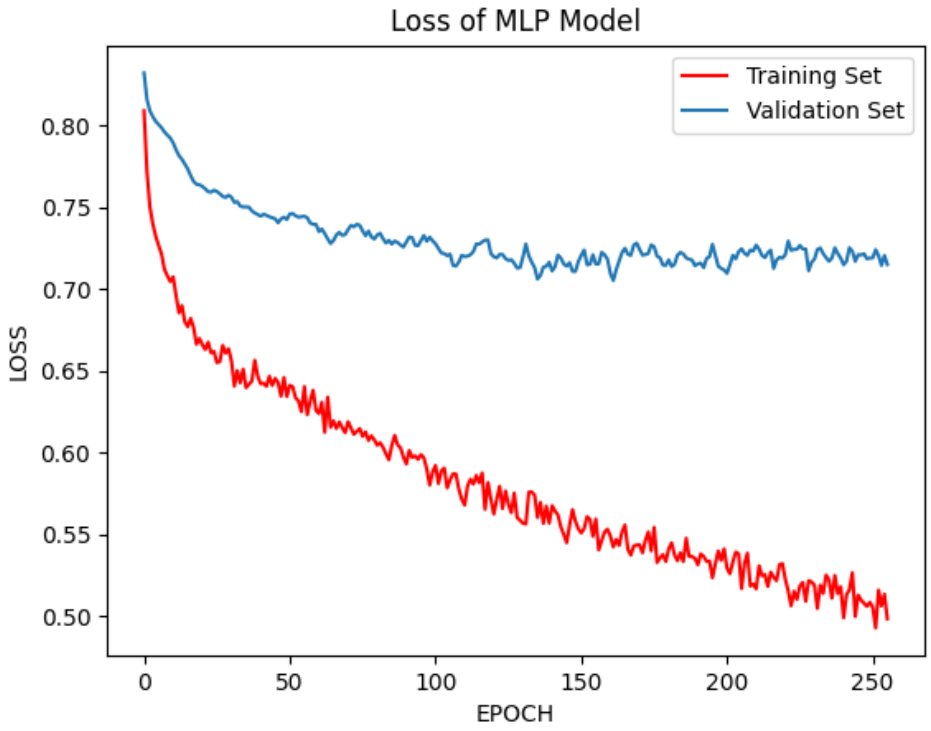
ماتریس آشفتگی مدل با عملیات کاهش ابعاد با PCA

و نتایج مدل بر روی داده‌گان تست در مقایسه با نتایج بخش اول برای مدل برنده به صورت زیر است:

	with PCA	without PCA
Accuracy	0.9811320900917053	0.9433962106704712
Loss	0.1247323751449585	0.14462608098983765
Precision	1.0	1.0
Recall	0.9473684210526315	0.8421052631578947
Fscore	0.972972972972973	0.9142857142857143
Time	19.868239402770996	21.051527738471167

بهبودی نتایج مشابه با قبل از و مشاهده می‌نمایید در اینجا نیز PCA مشابه با autoencoder عمل کرده و جواب یکسانی برای نتایج داده که در ماتریس آشفتگی نیز مشهود است. تنها تفاوت در زمان است که میبینیم مجددا PCA باعث کاهش زمان شده که نتیجه‌ی کاهش تعداد نورون ورودی است.

مسأله رگرسیون داده‌های سوال دوم به همراه autoencoder:



نمودار خطای آموزش و ارزیابی مدل با عملیات کاهش ابعاد با autoencoder

جدول خطای تست مدل با عملیات کاهش ابعاد با autoencoder:

MAE	MSE	RMSE
0.626389818770667	0.7410455802741467	0.8608400433728363

حال برای مقایسه جدول خطای تست مدل در سوال دوم را می‌اوریم.

MAE	MSE	RMSE
0.6162198475358884	0.7548687140298582	0.8688318099781213

همانطور که مشاهده می‌نمایید در مسأله‌ی رگرسیون مطرح شده در سوال دوم نتایج مربوط به autoencoder کمی دارای خطای بیشتر است که البته اگر مشاهده بفرمایید قابل اغماض می‌باشد.

همچنین برای زمان‌های الگوریتم‌ها نیز داریم.

زمان مسأله رگرسیون برای مدل برنده با autoencoder:

Time	18.373626232147217

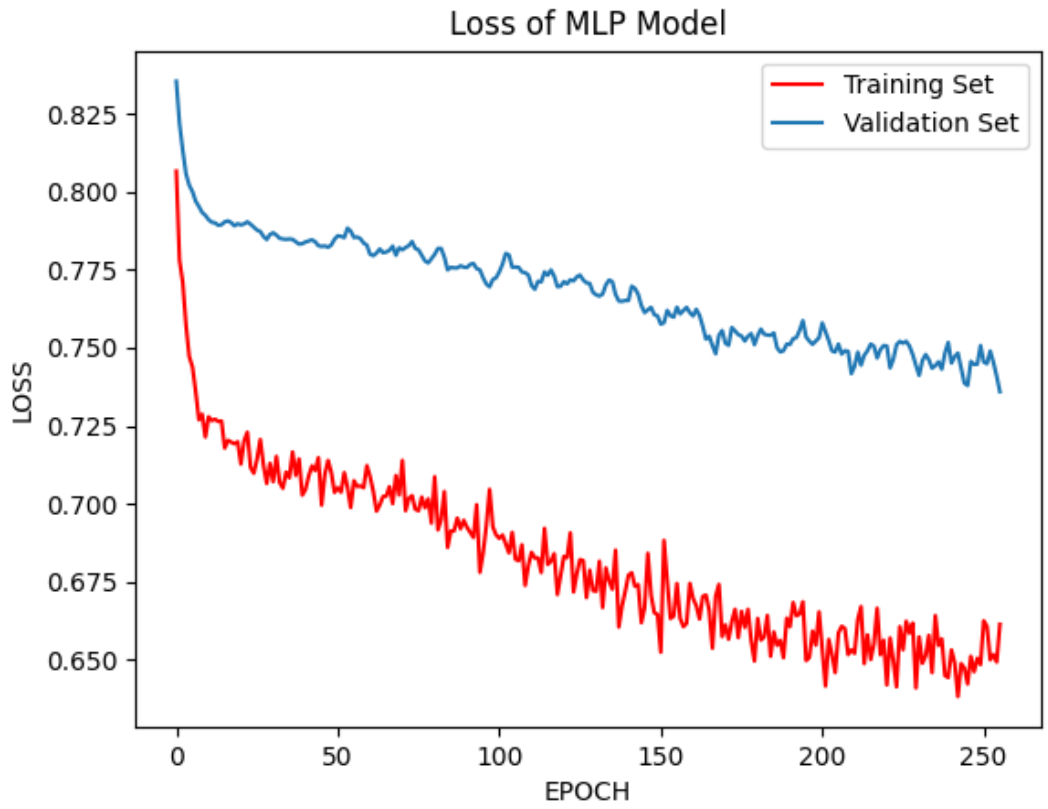
زمان مسأله رگرسیون برای مدل برنده بدون autoencoder:

Time	26.37018847465515

از نظر زمانی نیز همانطور که انتظار داشتیم، در مدل برنده با autoencoder بدلیل تعداد نورون کمی که در ورودی خواهیم داشت و نتیجتا تعداد کم پارامترها سریع‌تر شبکه آموزش می‌بیند.

حال نتایج PCA را برای مدل برنده‌ی سوال دوم ارائه می‌نماییم.

مسأله رگرسیون داده‌های سوال دوم به همراه PCA:



نمودار خطای آموزش و ارزیابی مدل با عملیات کاهش ابعاد با PCA

جدول خطای تست مدل برنده در سوال دوم با عملیات کاهش ابعاد با PCA:

MAE	MSE	RMSE
0.6593069003536757	0.7829699710422808	0.884855904112235

در مقایسه‌ی خطای این بخش با سوال دوم نتایجی مشابه با autoencoder و البته با خطای کمی بیشتر دریافت کردیم. این مسأله بواسطه‌ی حذف برخی اطلاعات مفید می‌تواند رخ دهد. حال به بررسی زمان‌ها می‌پردازیم:

زمان مسأله رگرسیون برای مدل برنده با PCA:

Time	18.20957851409912

زمان مسأله رگرسیون برای مدل برنده بدون PCA:

Time	26.37018847465515

مجددا زمان مربوط به بهترین مدل سوال دوم را با و بدون PCA ارائه کردیم تا مشاهده نمایید که کاهش زمان داریم و این مربوط به کاهش تعداد نورون ورودی و نتیجتا کاهش پارامترهایی که آموزش خواهند دید و تسریع در روند آموزش است.

در حالت کلی از سوال سوم به تاثیر بسیار خوب autoencoder و سپس PCA در کاهش ابعاد و تسریع در فرایند آموزش مدل رسیدیم. همچنین مشاهده کردیم که در مسائل طبقه بندی این مدل‌ها لااقل در این تمرین بهتر خروجی دادند و البته خروجی‌های مسائل رگرسیون هم قابل قبول و نسبت به مدل بهترین اختلاف کمی از نظر خطا داشتند. تمامی این موارد نشان از قابلیت این دو کاهنده‌ی ابعاد بویژه در مسائل طبقه بندی می‌دهد.

با تشکر - بدیعی