|  |  |  |
| --- | --- | --- |
|  | **به نام خدا**  **دانشگاه تهران**  **دانشکده‌ مهندسی برق و کامپیوتر** |  |
| **درس شبکه‌های عصبی و یادگیری عمیق**  **تمرین اول** | | |

|  |  |
| --- | --- |
| نام و نام خانوادگی | **سارا رستمی – محمدامین شاهچراغی** |
| شماره دانشجویی | **810۱00۳۵۵ - 810۱۹۹۱۹۶** |
| تاریخ ارسال گزارش | **07/08/1401** |

**فهرست**

[**پاسخ 1**. **شبکه عصبی** **Mccullotch-Pitts** 1](#_Toc117420798)

[۱-۱. ضرب‌کننده‌ی باینری دوبیتی 1](#_Toc117420799)

[**پاسخ ۲** **- AdaLine and MAdaLine** 3](#_Toc117420800)

[۱-۲. **AdaLine** 3](#_Toc117420801)

[۲-۲. **MadaLine** 7](#_Toc117420802)

[**پاسخ ۳** **–Restricted Boltzmann Machine** 14](#_Toc117420803)

[۱-۳. **سیستم توصیه‌گر** 14](#_Toc117420804)

[**پاسخ ۴** **–MLP** 21](#_Toc117420805)

[۱-۴. **Multi-Layer Perceptron** 21](#_Toc117420806)

**شکل‌ها**

شکل 1 – شبکه‌عصبی یک ضرب‌کننده‌ی باینری دوبیتی با استفاده از نورون M&P 1

شکل 2 - تمامی حالات ضرب‌کننده‌ی باینری دو بیتی M&P 2

شکل 3 - نمودار پراکندگی دو دسته‌ داده‌ بخش الف 3

شکل 4 - خط جداساز یادگیری شده توسط شبکه AdaLine برای داده‌های الف 4

شکل 5 - نمودار تغییرات خطای mean square error برحسب epoch برای داده‌های الف 4

شکل 6 - نمودار پراکنندگی دو دسته داده بخش ج 5

شکل 7 - خط جداساز یادگیری شده توسط شبکه 6

شکل 8 - نمودار تغییرات خطای mean square error برحسب epoch برای داده‌های ج 6

شکل 9 - داده های ورودی با برچسب 1 و -1..................................................................................................7

شکل 10 - نمودار پراکندگی‌ داده‌های دیتاست Madaline..........................................................................8

شکل 11 - نمودار loss/epochبا 3 نورون.........................................................................................................9

شکل 12 - تفکیک نقاط با 3 نورون.....................................................................................................................9

شکل 13 - نمودار loss/epochبا 4 نورون.....................................................................................................10

شکل 14 - تفکیک نقاط با 4 نورون..................................................................................................................11

شکل 15 - نمودار loss/epochبا 8 نورون......................................................................................................12

شکل 16 - تفکیک نقاط با 8 نورون..................................................................................................................12

شکل 17 - مقدار خطا در هر ایپاک 18

شکل 18 - نمودار خطا بر حسب epoch در سیستم توصیه‌گر 19

شکل 19 - خروجی تابع info() روی دیتاست houses 21

شکل 20 - تعداد مقادیر nan در هر یک از ستون‌های دیتاست houses 22

شکل 21- ماتریس correlation ویژگی‌های دیتاست houses 23

شکل 22- ویژگی با بیشترین میزان correlation با ویژگی price 23

شکل 23 - نمودار توزیع قیمت خانه‌ها 24

شکل 24 - نمودار پراکندگی دو ویژگی price و sqft\_living 25

شکل 25- دو ستون year و month 26

شکل 26- تقسیم دیتاست به داده‌های train و test 26

شکل 27 - MinMax Scalar کردن داده‌ها 27

شکل 28- آموزش شبکه‌ی MLP سه لایه 28

شکل 29 - آموزش شبکه با Adadelta optimizer 29

شکل 30- آموزش شبکه با RMSprop optimizer 30

شکل 31 - آموزش شبکه با Mean Absolute Error loss function 31

شکل 32 - آموزش شبکه با Binary Cross-Entropy loss function 32

شکل 33 - آموزش شبکه MLP با داده‌ی train و validation 33

شکل 34 - نمودار train loss و validation loss بر حسب epoch 33

شکل 35 - اختلاف قیمت پیش‌بینی شده توسط شبکه با قیمت واقعی 34

**جدول‌ها**

جدول 1 - دقت پیشبینی با 3 نورون................................................................................................................10

جدول 2 - دقت پیشبینی با 4 نورون................................................................................................................11

جدول 3 - دقت پیشبینی با 8 نورون................................................................................................................13

جدول 4 - مقایسه در 3 حالت............................................................................................................................13

[جدول 5 - ۵ مورد نخست فیلم‌ها 14](#_Toc117600072)

[جدول 6- ۵ مورد آخر فیلم‌ها 14](#_Toc117600073)

[جدول 7 - ۵ مورد اول امتیازها 15](#_Toc117600074)

[جدول 8 - ۵ مورد آخر امتیازها 15](#_Toc117600075)

[جدول 9 - اضافه کردن ستون List Index 16](#_Toc117600076)

[جدول 10 - ادغام دو مجموعه داده 16](#_Toc117600077)

[جدول 11 - حذف ستون‌های اضافی 17](#_Toc117600078)

[جدول 12 - دسته‌بندی بر اساس UserId 17](#_Toc117600079)

[جدول 13 - 15 فیلم با بیشترین امتیاز پیشنهادی مدل برای کاربر 27 20](#_Toc117600080)

# **پاسخ 1**. **ش**بکه عصبی **Mccullotch-Pitts**

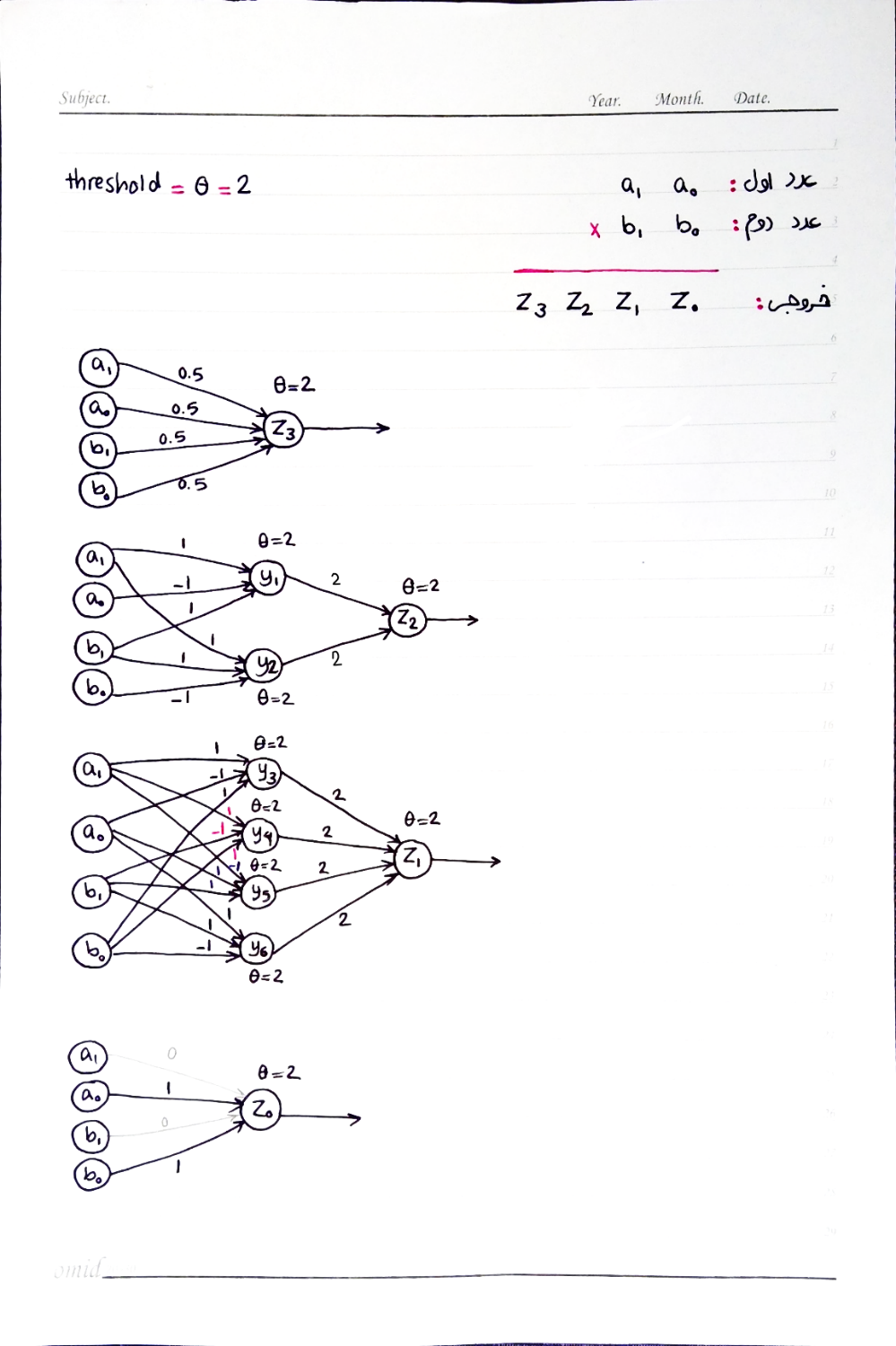
۱-۱. ضرب‌کننده‌ی باینری دوبیتی

**الف)** ابتدا جدول درستی یک ضرب‌کننده‌ی دوبیتی را رسم کرده و معادله‌ی خروجی‌ها را از طریق جدول کارنو بدست می‌آوریم. معادله‌ی هر یک از بیت‌های خروجی به صورت زیر می‌باشد:

*و*

و

با استفاده از معادلات بالا شبکه را رسم کرده و وزن یال‌ها را طبق شکل ۱ قرار می‌دهیم.



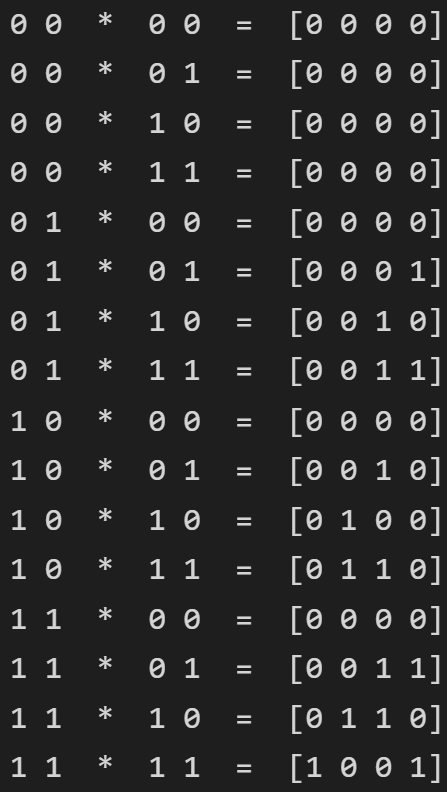
شکل 1 – شبکه‌عصبی یک ضرب‌کننده‌ی باینری دوبیتی با استفاده از نورون M&P

در شکل ۱ ، وزن یال‌هایی که رسم نشده اند(یا کمرنگ رسم شده‌اند مثل دو یال متصل به نورون ) برابر

با صفر است. به طور مثال از ورودی یالی به نورون رسم نشده که منظور همان یالی با وزن صفر است.

**ب)** تابعی(به نام mp\_neuron) برای پیاده‌سازی یک نورون M&P تعریف می‌کنیم که ورودی‌های شبکه و وزن‌های شبکه را به صورت numpy array و همچنین threshold نورون را به عنوان آرگومان‌های ورودی می گیرد و طبق تابع پله با میزان threshold خروجی ۰ و ۱ بر‌می‌گرداند. سپس یک تابع برای ضرب‌کننده‌ی دوبیتی پیاده‌سازی کردیم که ورودی شبکه (یعنی دو عدد دوبیتی) را به صورت یک آرایه با اندازه‌ی ۴ به عنوان آرگومان ورودی ‌می ‌گیرد. در داخل این تابع ۱۰ بار تابع mp\_neuron با ورودی ها و وزن‌های مشخص شده در شکل ۱ فراخوانی می‌شود. هر فراخوانی این تابع خروجی یکی از نورون های y و z را محاسبه می‌کند. در نهایت تابع two\_bit\_multiplier خروجی های شبکه (یعنی ) را بر‌می‌گرداند.

نتیجه‌ی فراخوانی تابع با تمام حالت جدول درستی برای دو عدد دوبیتی را در شکل ۲ مشاهده می‌کنید.



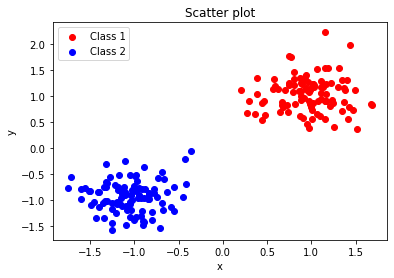
شکل 2 - تمامی حالات ضرب‌کننده‌ی باینری دو بیتی M&P

# **پاسخ ۲** **- AdaLine and MAdaLine**

## ۱-۲. **AdaLine**

**الف)** ابتدا داده‌ها را به طور تصادفی از توزیع‌های نرمال با میانگین و انحراف معیار مذکور generate می‌کنیم و برچسب ۱ و ۱- را به ترتیب به داده‌های کلاس ۱ و ۲ assign‌ می‌کنیم.

نمودار پراکندگی دو دسته داده را در شکل ۳ مشاهده می‌کنید.

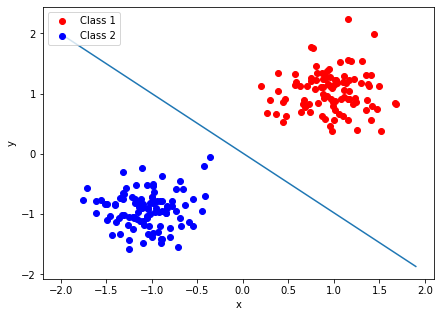


شکل 3 - نمودار پراکندگی دو دسته‌ داده‌ بخش الف

**ب)** آموزش شبکه را تا زمانی ادامه می‌دهیم که یا خطای همه‌ی نمونه‌های train کمتر از یک مقدار مشخصی باشد(مقدار default ما در اینجا threshold = 0.5 می‌باشد) یا تعداد epochها به ۵۰ برسد. آرگومان‌های ورودی تابع Adaline، داده‌ی ورودی شبکه، آلفا (learning rate) و threshold برای شرط خاتمه‌ی شبکه می‌باشد. این تابع به عنوان خروجی میزان خطا در هر epoch، وزن‌های یالها و bias را در انتهای یادگیری را بر‌می‌گرداند.

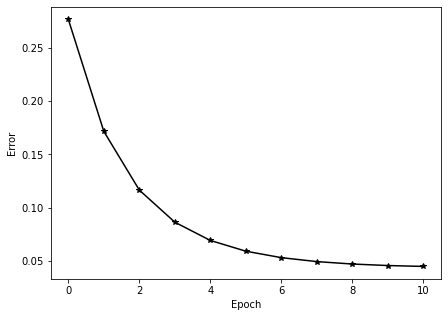
به ازای مقادیر مختلف برای پارامتر‌های آلفا و threshold ، شبکه را با داده را train کردیم و در نهایت مقدار alpha = 0.001 و threshold = 0.33 را برای آموزش شبکه بر‌گزیدیم که در نتیجه‌ی آن، شبکه‌ی ما طی ۱۱ تا epoch آموزش یافت. معادله‌ی خط جداساز بدست آمده توسط شبکه به صورت زیر می‌باشد:

این خط جداساز را در شکل ۴ مشاهده می‌کنید.



شکل 4 - خط جداساز یادگیری شده توسط شبکه AdaLine برای داده‌های الف

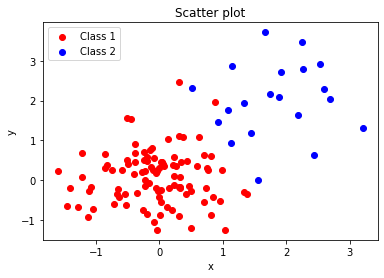
نمودار تغییرات خطا بر حسب epoch را در شکل ۵ مشاهده می‌کنید.



شکل 5 - نمودار تغییرات خطای mean square error برحسب epoch برای داده‌های الف

همانطور که در نمودار شکل ۵ مشاهده می‌کنید، میزان خطا با گذشت epochها، از حدود ۰.۳ به حدود ۰.۰۴ (عددی نزدیک به صفر) کاهش پیدا می‌کند. پس این‌طور به نظر می‌رسد که شبکه به درستی آموزش یافته. دقت شود که این نمودار نشان‌دهنده‌ی خطای آخرین نمونه‌ در هر epoch می‌باشد.

**ج)** نمودار پراکندگی دو دسته داده‌ی جدید را در شکل۶ مشاهده می‌کنید.

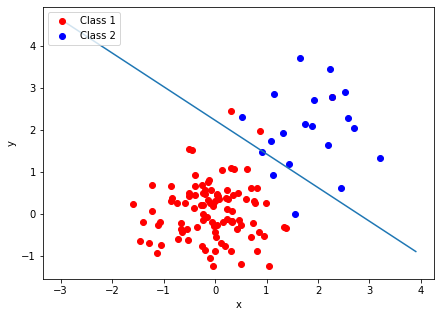


شکل 6 - نمودار پراکنندگی دو دسته داده بخش ج

این بار الگوریتم AdaLine (حتی با تغییر مقادیر پارامتر های آلفا و threshold) با شرط خاتمه‌ی threshold همگرا نمی‌شود و ۵۰ تا epoch را می‌گذراند. همانطور که انتظار می‌رفت از آنجایی که تعداد داده‌های دو کلاس نامتوازن(unbalanced) می‌باشد، روش AdaLine نتوانست به درستی دو دسته را از هم جدا سازد و حاوی تعدادی خطاست. همچنین از آنجایی که داده‌ها پراکنندگی زیادی دارند و قابل جداسازی با یک خط نمی‌باشند، AdaLine در چنین شرایطی به خوبی قادر به جداسازی داده‌ها نمی‌باشد.

معادله‌ی خط جداساز یادگرفته شده توسط شبکه به صورت زیر می‌باشد:

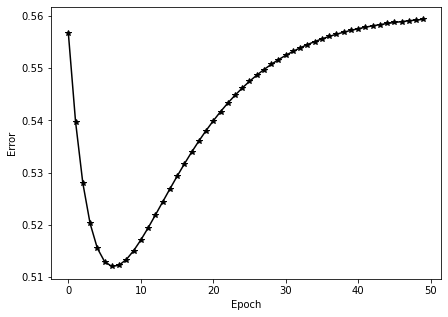
با مشاهده‌ی شکل ۷ می‌توان خط جداساز یادگرفته توسط شبکه را دید.



شکل 7 - خط جداساز یادگیری شده توسط شبکه

نمودار تغییرات خطا بر حسب epoch را در شکل ۸ مشاهده می‌کنید. میزان خطا کلا بالاست و روند نزولی ندارد.

به دلایل ذکر شده در بالا، نمودار تغییرات خطا بر حسب epoch، مانند بخش الف نزولی نمی‌شود چرا که AdaLine نمی‌تواند خطی پیدا کند که قادر به جداسازی داده‌ها باشد.

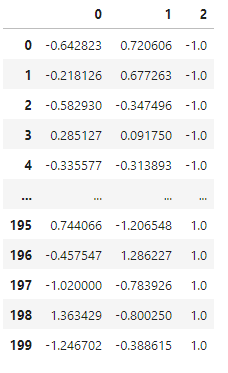


شکل 8 - نمودار تغییرات خطای mean square error برحسب epoch برای داده‌های ج

## ۲-۲. **MAdaLine**

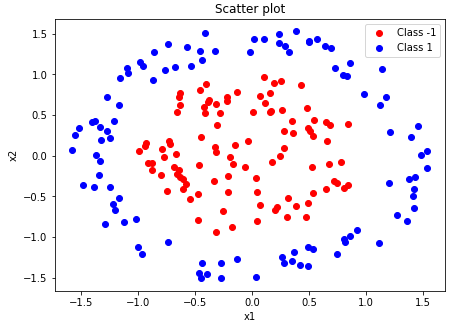
**الف)** الگوریتم MRI (که در واقع همان ورژن اصلی MAdaLine می‌باشد) یکی از الگوریتم‌های Madaline است.  
در این الگوریتم در لایه خروجی وزن ها ثابت هستند و تنها وزن ها و بایاس های لایه‌های میانی تنظیم  
میشوند. در این الگوریتم ابتدا مقدار اولیه‌ی کوچک برای وزن‌ها در نظر میگیریم. همچنین نرخ یادگیری  
مقدار کوچکی دارد. سپس تا زمانی که شرط توقف (ثابت شدن تقریبی وزن ها در دو تکرار متوالی یا طی  
تعداد مشخصی Epoch) برقرار نباشد مراحل تکرار میشود.  
حال مقدار شبکه برای هر نورون Adaline را حساب میکنیم و خروجی را پس از عبور از تابع فعالساز به  
دست می‌آوریم. اگر خروجی محاسبه شده با تارگت متناظر برابر بود، وزن ها آپدیت نمی شوند در غیر این صورت اگر تارگت برابر یک باشد وزن های مربوط به نورونی که مقدار net ورودی از همه به صفر نزدیک تر است با استفاده از الگوریتم گرادیان کاهشی آپدیت می‌شود. اما اگر تارگت مربوطه برابر با ۱-باشد، وزن های تمام نورون هایی که مقدار net آنها مثبت است آپدیت می‌شود. وزن و بایاس لایه نهایی را هم بر اساس تعداد نورون لایه میانی طوری تنظیم میکنیم که گیت OR تشکیل دهد .

**ب)** ابتدا داده ها را میخوانیم و در دیتافریم ذخیره میکنیم سپس به آنهایی متعلق به گروه 1 هستند برچسب -1 و به آنهایی که متعلق به گروه 2 هستند برچسب 1 میدهیم .شکل زیر مجموعه داده را پس از این عملیات نشان میدهد .



شکل 9 - داده های ورودی با برچسب 1 و -1

نمودار پراکندگی داده‌ها را در شکل زیر مشاهده می‌کنید.



شکل 10- نمودار پراکندگی‌ داده‌های دیتاست MadaLine

برای آموزش شبکه و جداسازی کلاسی به نام MadalineNetwork را در پایتون تعریف کردیم که شامل متد هایی برای موارد زیر است :

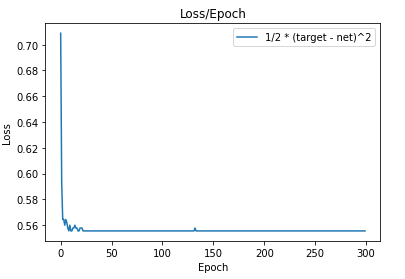
* تنظیم متغیر ها و وزن دهی اولیه (مقادیر کوچک و تصادفی)
* محاسبه مقادیر خروجی لایه های پنهان و خروجی نهایی
* تابع فعالساز
* تصحیح وزن ها و بایاس ها بر اساس الگوریتم mri
* محاسبه خطا بر اساس تابع ½ \*(𝑡 − 𝑛𝑒𝑡)^ 2
* نمایش خطا ها بر اساس ایپاک ها
* آموزش مدل
* پیشبینی بر اساس مدل به دست آمده و محاسبه دقت مدل

.

حال مدل را با ترین میکنیم و با به ترتیب 3و4و8 نورون جداسازی را انجام میدهیم . (ماکسیمم epoch را 300 و نرخ یادگیری را 0.1 قرار داده ایم )

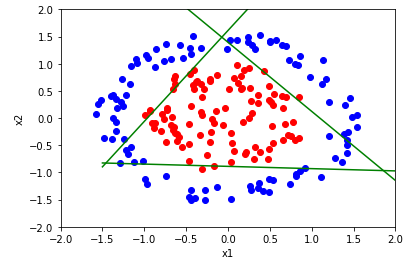
MadaLine با سه نورون:

300 epoch انجام میشود .



شکل 11- نمودار loss/epochبا 3 نورون

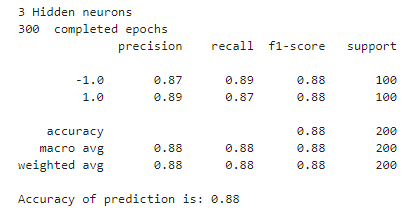
حالا جداسازی میکنیم :



شکل 12- تفکیک نقاط با 3 نورون

سپس دقت جدا سازی را نمایش میدهیم که accuracy اینجا 88% شد . مدل با 3 نورون به دقت کامل نمیرسد.

جدول 2 – دقت پیشبینی با 3 نورون



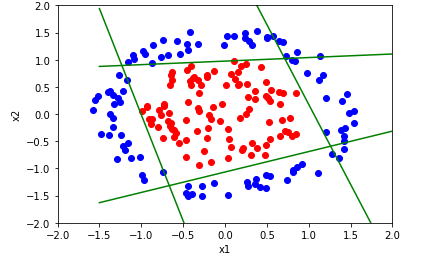
MadaLine با چهار نورون:

57 epoch انجام میشود و به دقت کامل میرسیم .



شکل 13- نمودار loss/epochبا 4 نورون

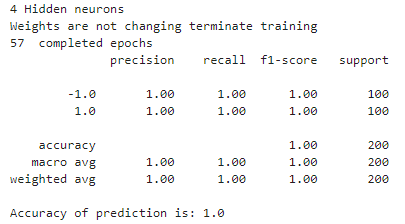
حالا جداسازی میکنیم :



شکل 14- تفکیک نقاط با 4 نورون

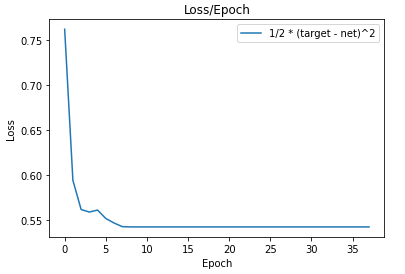
سپس دقت جدا سازی را نمایش میدهیم که accuracy اینجا 100% شد و مدل کاملا نقاط را از هم تفکیک کرد .

جدول 2 – دقت پیشبینی با 4 نورون



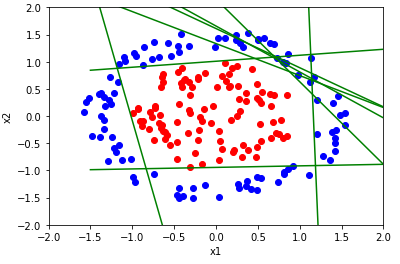
MadaLine با هشت نورون:

37 epoch انجام میشود و به دقت کامل میرسیم .



شکل 15- نمودار loss/epochبا 8 نورون

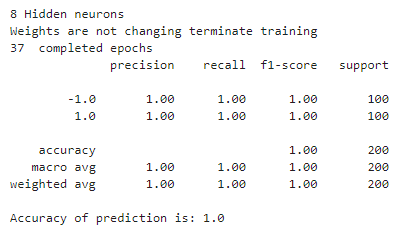
حالا جداسازی میکنیم :



شکل 16- تفکیک نقاط با 8 نورون

سپس دقت جدا سازی را نمایش میدهیم که accuracy اینجا 100% شد و مدل کاملا نقاط را از هم تفکیک کرد .

جدول 3 – دقت پیشبینی با 8 نورون



ج**)** حال به مقایسه در دقت و ایپاک در حالت های 3،4و8 نورون میپردازیم .

جدول 4 – مقایسه در 3 حالت

|  |  |  |  |
| --- | --- | --- | --- |
| 8 | 4 | 3 | تعداد نورون |
| 37 | 57 | 300 | تعداد ایپاک |
| 100% | 100% | 88% | دقت |

با 3 نورون به تفکیک کامل نرسیدیم و حداقل 4 نورون نیاز است تا دقت 100% داشته باشیم. وقتی تعداد نورون ها را بالاتر میبریم دقت افزایش میابد ؛ در واقع تعداد نورون ها در اینجا همان خط های جدا کننده هستند بنابراین با نورون های بیشتر نواحی به شکل جامع تر از هم جدا میشوند پس اگر نقاطی در آینده اصافه شوند مدل آنها را هم بهتر تفکیک خواهد کرد .

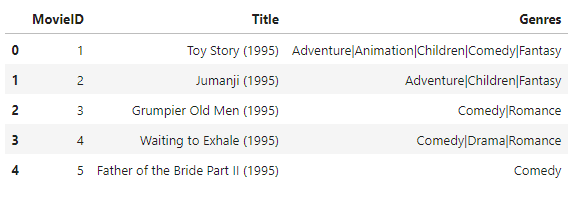
همچنین هر قدر تعداد نورون ها بالاتر باشد سریعتر به همگرایی میرسیم و با تعداد ایپاک کمتری میتوانیم تفکیک را انجام دهیم . وقتی تعداد نورون ها 8 تا میشود همچنان که در شکل 16 مشخص است 5 خط داده ها را کاملا از هم جدا میکنند و 3 خط دیگر بلا استفاده میمانند در واقع مدل در هنگام ترین شدن دیگر وزن های آنها را بروزرسانی نکرده است چرا که 5 ضلعی برای جداسازی کفایت میکند .

# **پاسخ ۳** **–Restricted Boltzmann Machine**

## ۱-۳. **سیستم توصیه‌گر**

**A)** داده های مربوط به فیلم ها و امتیازات را میخوانیم و در 2 دیتا فریم ذخیره میکنیم ، سطر شماره صفر که نام ستون ها است که بعنوان داده خوانده شده را حذف میکنیم و برای ستون ها نامگذاری مناسب انجام میدهیم . سپس 5 مورد اول و انتهایی فیلم ها و امتیازات را نمایش میدهیم .

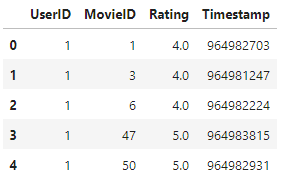
جدول 5 - ۵ مورد نخست فیلم‌ها



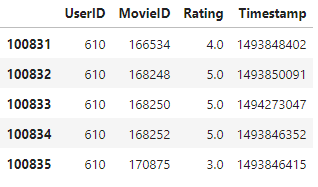
جدول 6- ۵ مورد آخر فیلم‌ها



جدول 7 - ۵ مورد اول امتیازها



جدول 8 - ۵ مورد آخر امتیازها



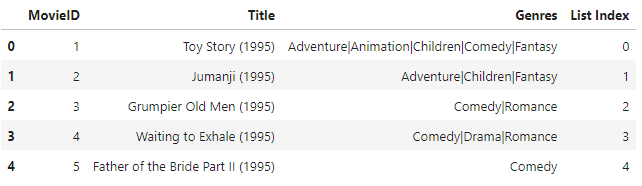
ابعاد هر دو مجموعه داده را نمایش میدهیم :

ابعاد فیلم ها : (9742, 3)

ابعاد امتیازات : (100836, 4)

سپس ستون جدیدی ایجاد میکنیم و شماره هر سطر را در آن ذخیره میکنیم :

جدول 9 - اضافه کردن ستون List Index



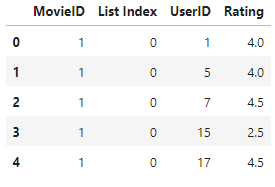
**B)** بر اساس ستون شناسه فیلم دو مجموعه داده را با هم ادغام میکنیم .

جدول 10 - ادغام دو مجموعه داده



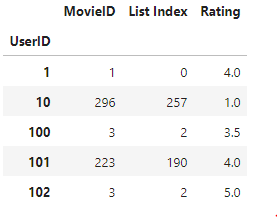
**C)** ستون های Title ، Genres ، Timestamp در محاسبات مورد نظر برای این روش تاثیر گذار نیستند چرا که در این روش وقتی کاربران مختلف به یک فیلم امتیاز مشابهی داده باشند احتمال اینکه در آینده نیز به یک فیلم امتیاز یکسانی بدهند بالا است . ولی بین 3 ستون ذکر شده در میان کاربران ارتباطی وجود ندارد و کاربر آنها را تعیین نمیکند و مقادیر ثابتی دارند که ویژگی های خود فیلم است برای اشاره به فیلم ها فقط شناسه آنها کافی است بنابراین آنها را حذف میکنیم تا راحتتر با دیتا فریم کار کنیم .

جدول 11 - حذف ستون‌های اضافی



**D)**در دیتافریم به دست آمدهداده ها را بر اساس شناسه هر کاربر گروه بندی میکنیم .

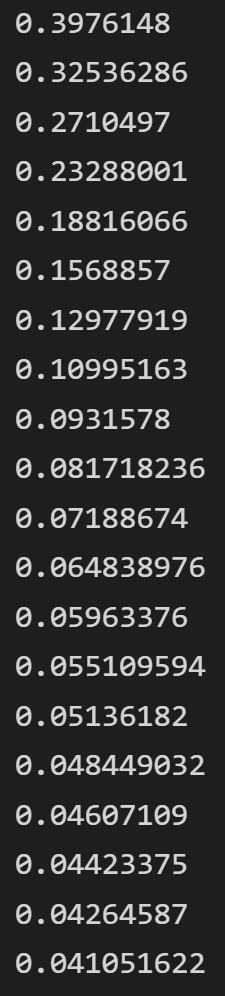
جدول 12 - دسته‌بندی بر اساس UserId



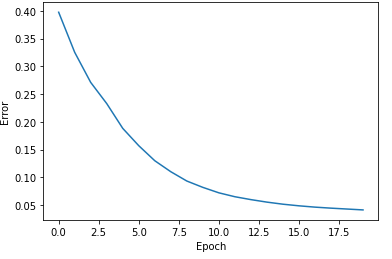
**E) چون تمام امتیازات فیلم ها بین 1 تا 5 است با تقسیم همه به 5 نرمالسازی انجام میشود و همه امتیازات بین 0 تا 1 قرار میگیرند . لیست** train\_X **را میسازیم که هر سطر آن مربوط به یک کاربر است و طول آن سطر به اندازه ی تعداد فیلم ها است که در هر خانه امتیاز مربوط به یک فیلم ذخیره شده است .**

**F) برای ساخت مدل از کتابخانه** tensorflow **کمک میگیریم تعداد** hidden layer **ها را 20 نورون و برای** visible layer **ها به تعداد فیلم ها در نظر میگیریم .** bias **و وزن هایی که دو لایه را به یکدیگر متصل میکنند را در** placeholder **ذخیره میکنیم . لایه ها را ایجاد میکنیم و توابع فعالساز آنها را** relu  **و** sigmoid **قرار میدهیم ،** learning rate **را برابر 0.1 قرار میدهیم . گرادیان ها را میسازیم در اینجا باید** Contrastive Divergence **را حداکثر کنیم .**

**G) مدل را در 20** epoch **ترین میکنیم ، مقدار خطا در هر ایپاک را در شکل 17 مشاهده می کنید. همانطور که نمودار خطا بر حسب ایپاک را در شکل زیر می‌بینید مقدار خطا در هر ایپاک کاهش پیدا می‌کند. پس شبکه به خوبی آموزش دیده شد.**

****

شکل 17 - مقدار خطا در هر ایپاک

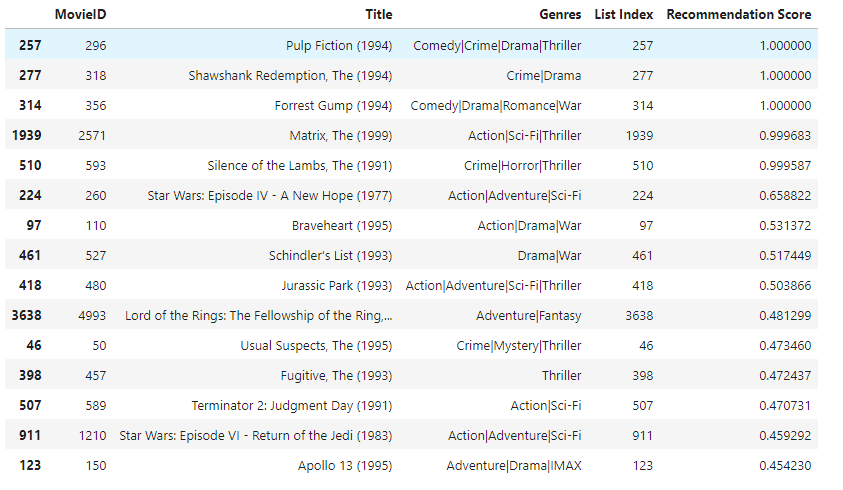


شکل 18 - نمودار خطا بر حسب epoch در سیستم توصیه‌گر

**H) حالا مدل** RBM **را داریم برای اینکه فیلم های مورد علاقه یک کاربر خاص را پیدا کنیم اطلاعات مربوط به امتیاز فیلم های او که در** train\_X **ذخیره کرده بودیم را به مدل وارد(**feed**) میکنیم و بر اساس آن ورودی را بازسازی میکنیم . کاربر شماره 27 را به شکل تصادفی انتخاب میکنیم و** train\_X[27]  **را وارد میکنیم . مدل امتیاز تمام فیلم ها را برای او پیشبینی میکند . دیتا فریم را بر اساس امتیازات** sort **میکنیم و ۱۵ فیلمی که برای کاربر 27 بیشترین امتیاز پیشنهاد داده شده توسط مدل را دارند نشان میدهیم.**

**این ۱۵ فیلم پیشنهادی را در جدول 13 مشاهده می کنید.**

جدول 13 - 15 فیلم با بیشترین امتیاز پیشنهادی مدل برای کاربر 27



.

# **پاسخ ۴** **–MLP**

## ۱-۴. **Multi-Layer Perceptron**

**A)** دیتاست را در دیتافریمی به نام houses لود می‌کنیم. نتیجه‌ی اجرای تابع info() بر روی دیتاست خوانده شده را در شکل ۱9 مشاهده می‌کنید.



شکل 19 - خروجی تابع info() روی دیتاست houses

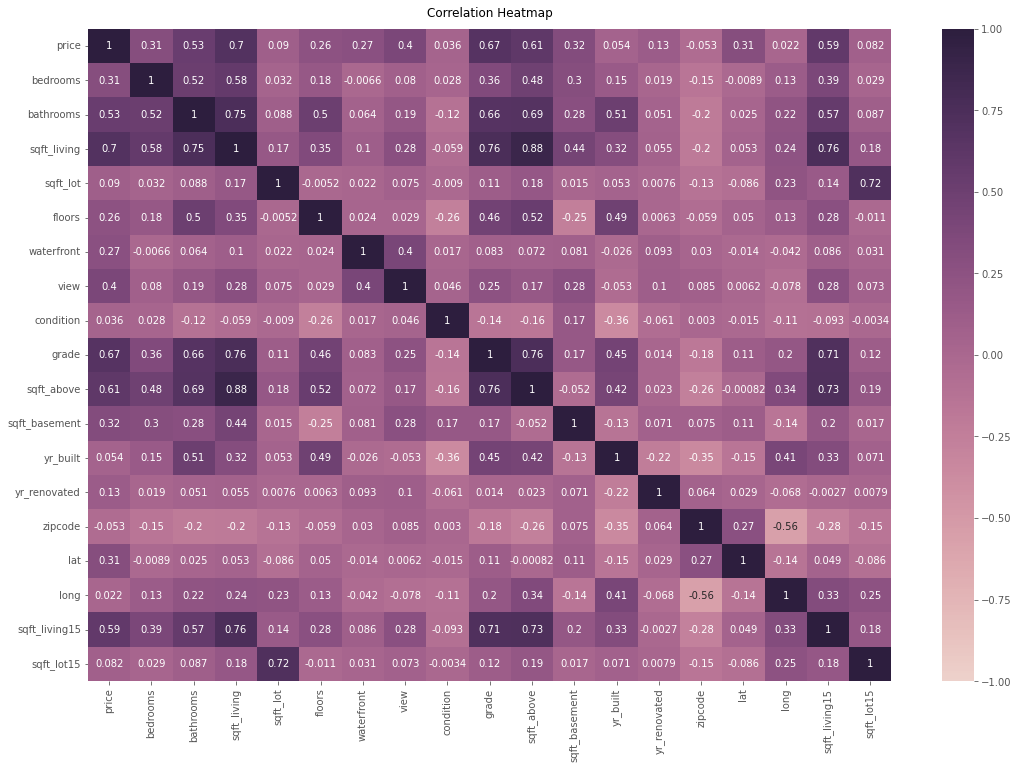
**B)** با تابع اجرای isna() روی هر ستون می‌توان nan یا non-nan بودن داده‌های موجود در دیتافریم را فهمید. پس کافیست تابع sum() را هم روی houses.isna() اعمال کنیم. houses.isna.sum() تعداد مقادیر nan هر ستون را به ما می‌دهد که آن را در شکل 20 مشاهده می‌کنید.



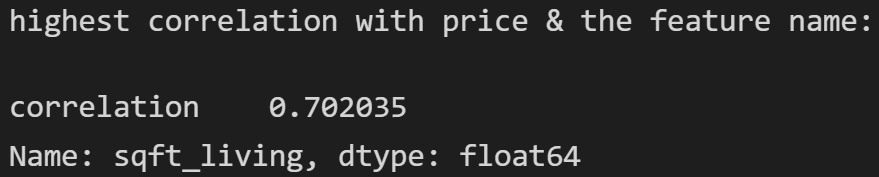
شکل 20 - تعداد مقادیر nan در هر یک از ستون‌های دیتاست houses

همانطور که در شکل 20 می‌بینید ما در این دیتاست مقدار nan نداریم.

**C)** ماتریس correlation ویژگی‌ها را در شکل 21 مشاهده می‌کنید. همانطور که در ماتریس قابل مشاهده است، ویژگی sqft\_living بیشترین همبستگی را با ویژگی price داشته که مقدار این همبستگی برابر با 0.702035 می‌باشد.

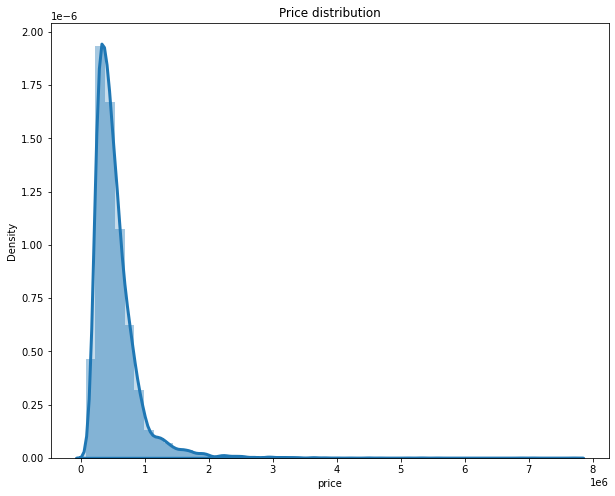


شکل 21- ماتریس correlation ویژگی‌های دیتاست houses



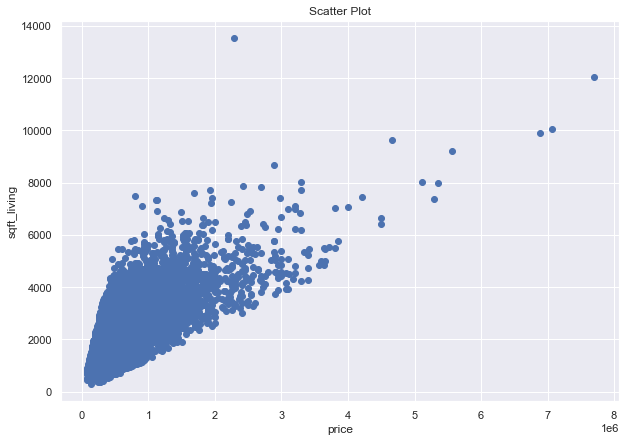
شکل 22- ویژگی با بیشترین میزان correlation با ویژگی price

**D)** نمودار توزیع ویژگی price را در شکل 23 مشاهده می‌کنید. این ویژگی یک توزیع با چولگی راست می‌باشد. یعنی فراوانی خانه‌ها با قیمت‌های کم نسبت به خانه های با قیمت بالا خیلی بیشتر است.



شکل 23 - نمودار توزیع قیمت خانه‌ها

نمودار پراکندگی دو ویژگی price و sqft\_living (که بیشترین میزان همبستگی را با price دارد) را در شکل 24 مشاهده می‌کنید. مانند انتظار نمودار این دو ویژگی همبستگی بالای آنها را نشان می‌دهد.



شکل 24 - نمودار پراکندگی دو ویژگی price و sqft\_living

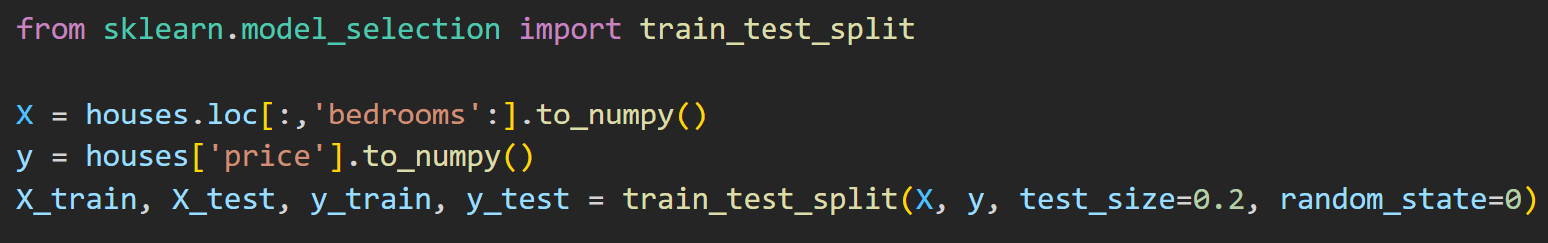
به طور کلی با افزایش price (قیمت)، مقدار sqft\_living (مساحت اتاق پذیرایی) هم افزایش می‌یابد و بلعکس که چنین اتفاقی منطقی می‌باشد.

**E)**  با parse کردن ستون date، مقدار سال و ماه را استخراج کرده date، به int تبدیل کرده و این دو ستون(month و year) را به دیتافریم اضافه می‌کنیم. خود ستون date را هم drop می‌کنیم. نمایش دو ستون جدید را در شکل 25 مشاهده می‌کنید.



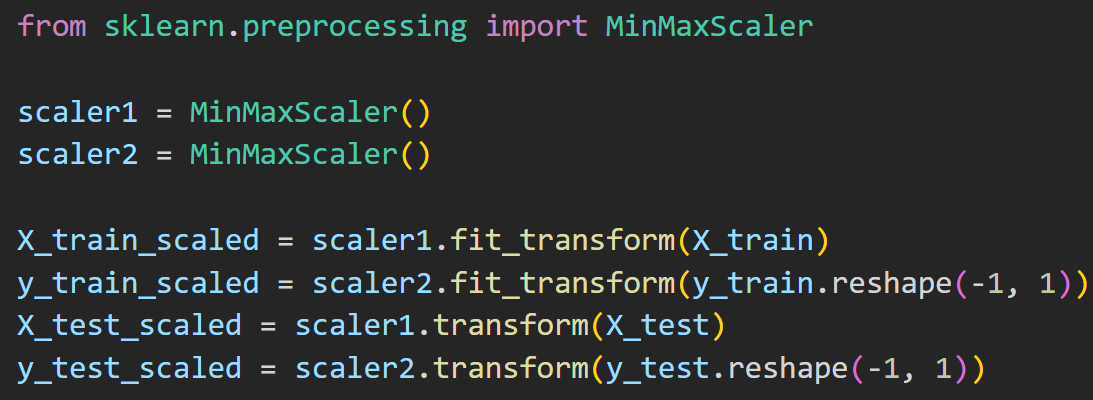
شکل 25- دو ستون year و month

**F)** همه‌ی ستون ها به جز id و price را به عنوان مجموعه‌ی x و ستون price را به عنوان مجموعه‌ی y (target ) انتخاب می‌کنیم. سپس با استفاده از متد train\_test\_split() از کتابخانه‌ی sklearn date، داده‌ها را به شیوه‌ی گفته شده به مجموعه‌های train و test تقسیم می کنیم. برای reproducibility این تقسیم، random sate را set کردیم. این کد را در شکل 26 مشاهده می کنید.



شکل 26- تقسیم دیتاست به داده‌های train و test

**G)** در کد شکل 27، با استفاده از تابع fit\_transform() دیتای train را center می‌کنیم. سپس با استفاده از همان میانگین و انحراف معیار که برای داده‌ی train استفاده شد داده‌ی test را transform می‌کنیم.



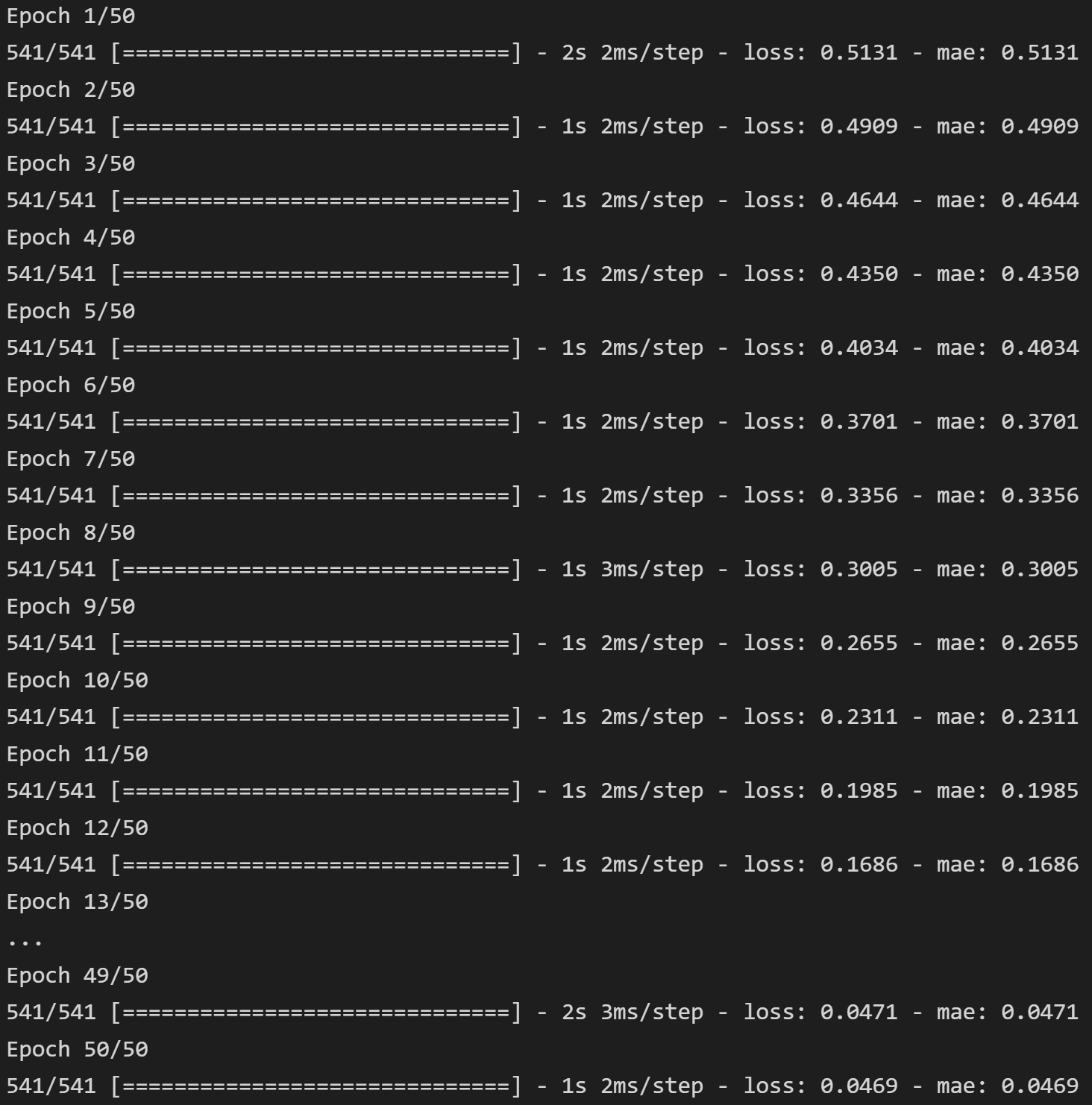
شکل 27 - MinMax Scalar کردن داده‌ها

**H)** از کتابخانه‌‌ی tensorflow استفاده می‌کنیم. شبکه‌ای با سه لایه می‌سازیم. لایه‌ی پنهان اول ۱۵ نورون، لایه‌ی پنهان دوم ۷ نورون و لایه‌ی خروجی ۱ نورون دارد. از Relu function به عنوان تابع فعالساز نورون‌های هر سه لایه استفاده می‌کنیم. از (MAE)Mean Absolute Error به عنوان loss function استفاده می‌کنیم . optimizer را هم Stochastic Gradient Descent قرار می‌دهیم. تعداد epochها را هم ۵۰ قرار می‌دهیم. در نهایت با x و y آموزش(train) که به روش Minmax، scale شده‌اند، مدل را آموزش می‌دهیم. همانطور که در شکل 28 می بینیم، مقدار mae در هر epoch کاهش پیدا می‌کند. کاهش چشمگیری در ایپاک چهارم می‌بینیم (مقدار خطای mae از 0.0569 به 0.0291 کاهش می‌یابد) و پس از آن مقدار کاهش کمتر می‌باشد تا اینکه در ایپاک ۵۰م مقدار خطا به 0.0172 می‌رسد. پس شبکه به درستی آموزش می‌بیند.

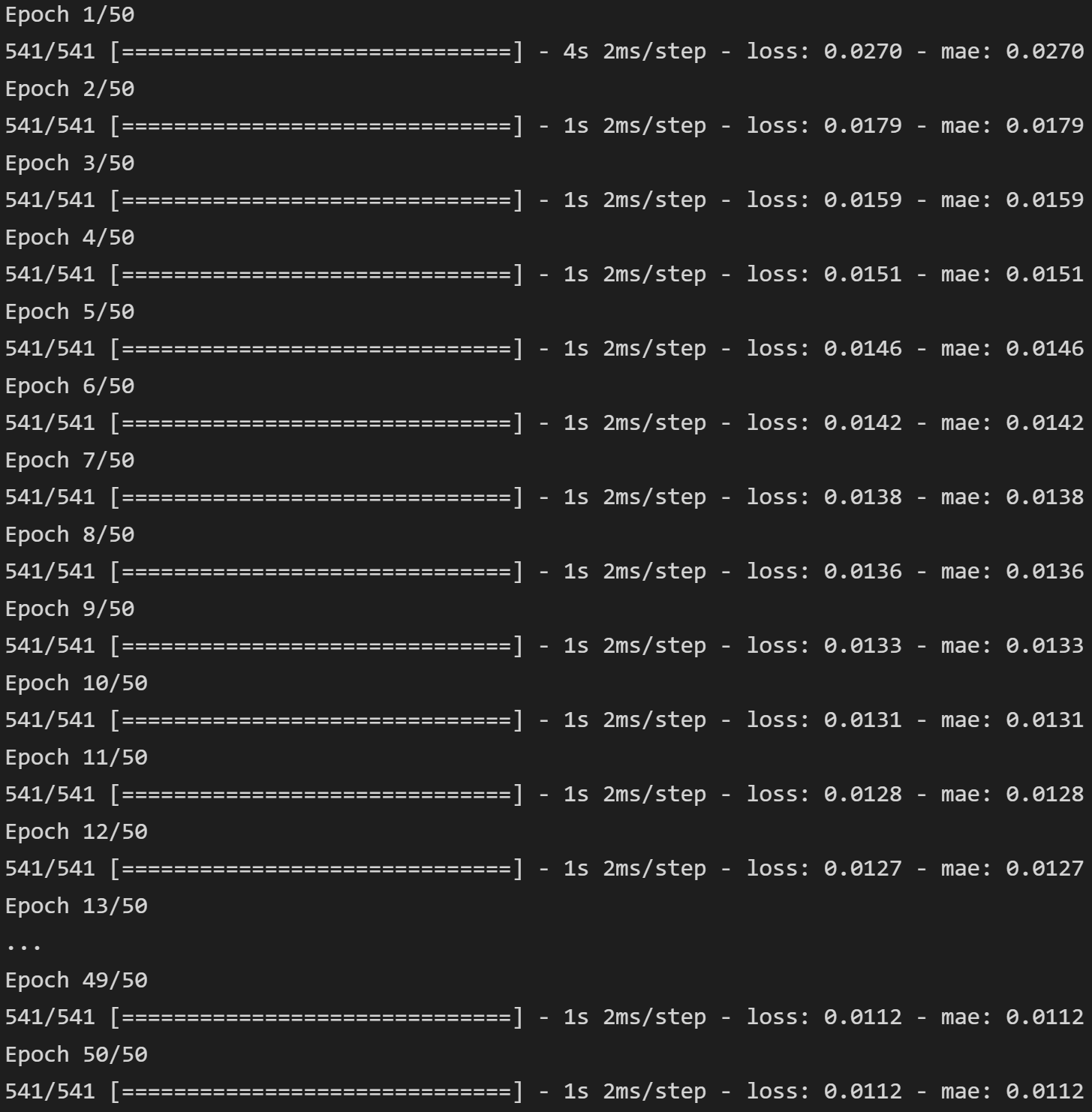


شکل 28- آموزش شبکه‌ی MLP سه لایه

**I)** در شکل ۲9 نتیجه‌ی آموزش شبکه با Adadelta optimizer و در شکل 30 نتیجه‌ی آموزش شبکه با RMSprop optimizer را مشاهده می‌کنید. در هردو این حالات loss function انتخابی mean absolute error می‌باشد. می‌بینید میزان خطا شبکه با RMSprop کمتر از شبکه با Adadelta‌ می‌باشد. در هردو این optimizerها نیازی به دستی تغییر دادن نرخ یادگیری نیست چرا که نرخ یادگیری به طور adaptive تغییر می‌کند. RMSprop شباهت زیادی به Adadelta دارد. تنها تفاوت آنها در نحوه مدیریت گرادیان‌های گذشته می‌باشد.

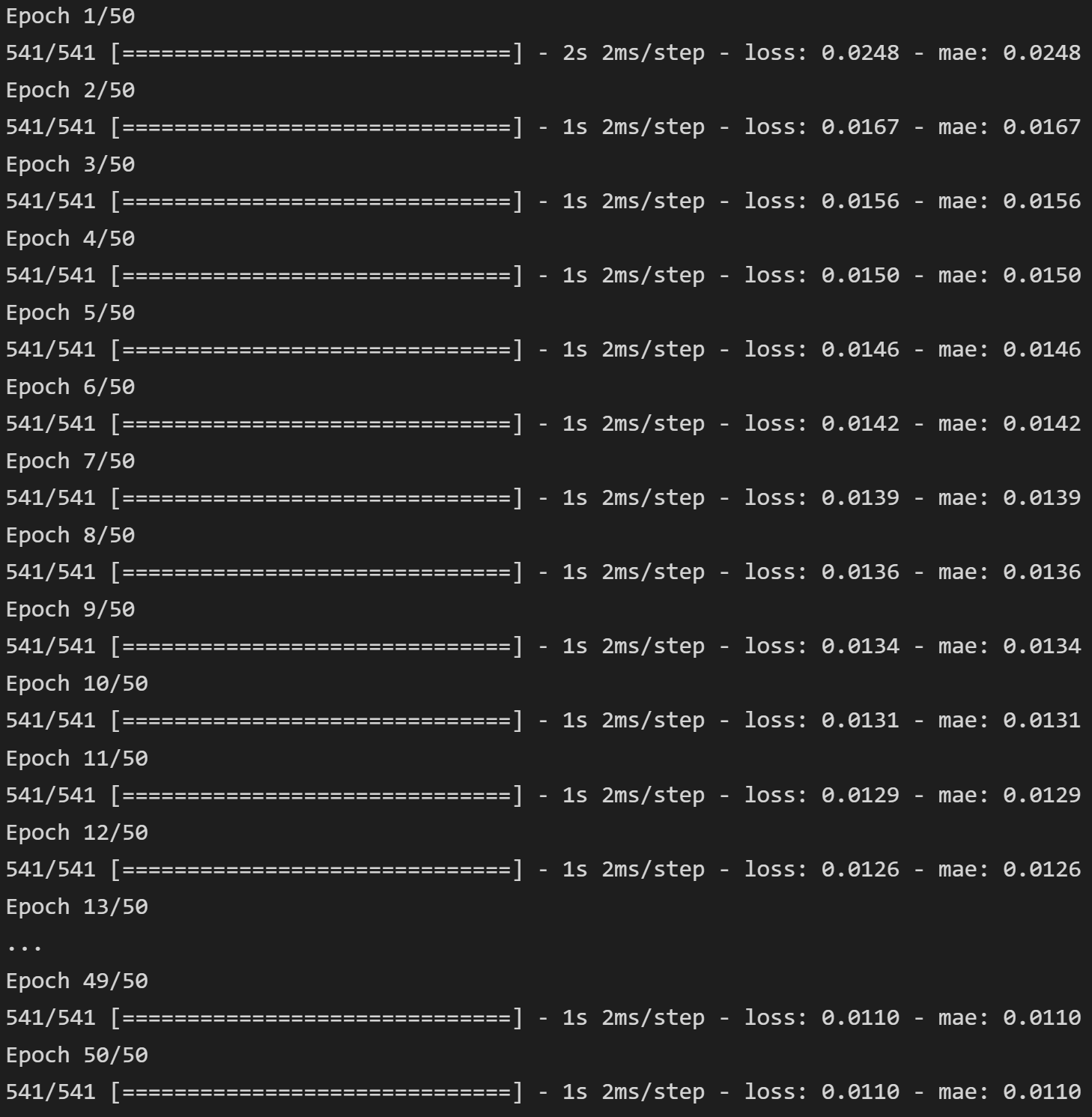


شکل 29 - آموزش شبکه با Adadelta optimizer

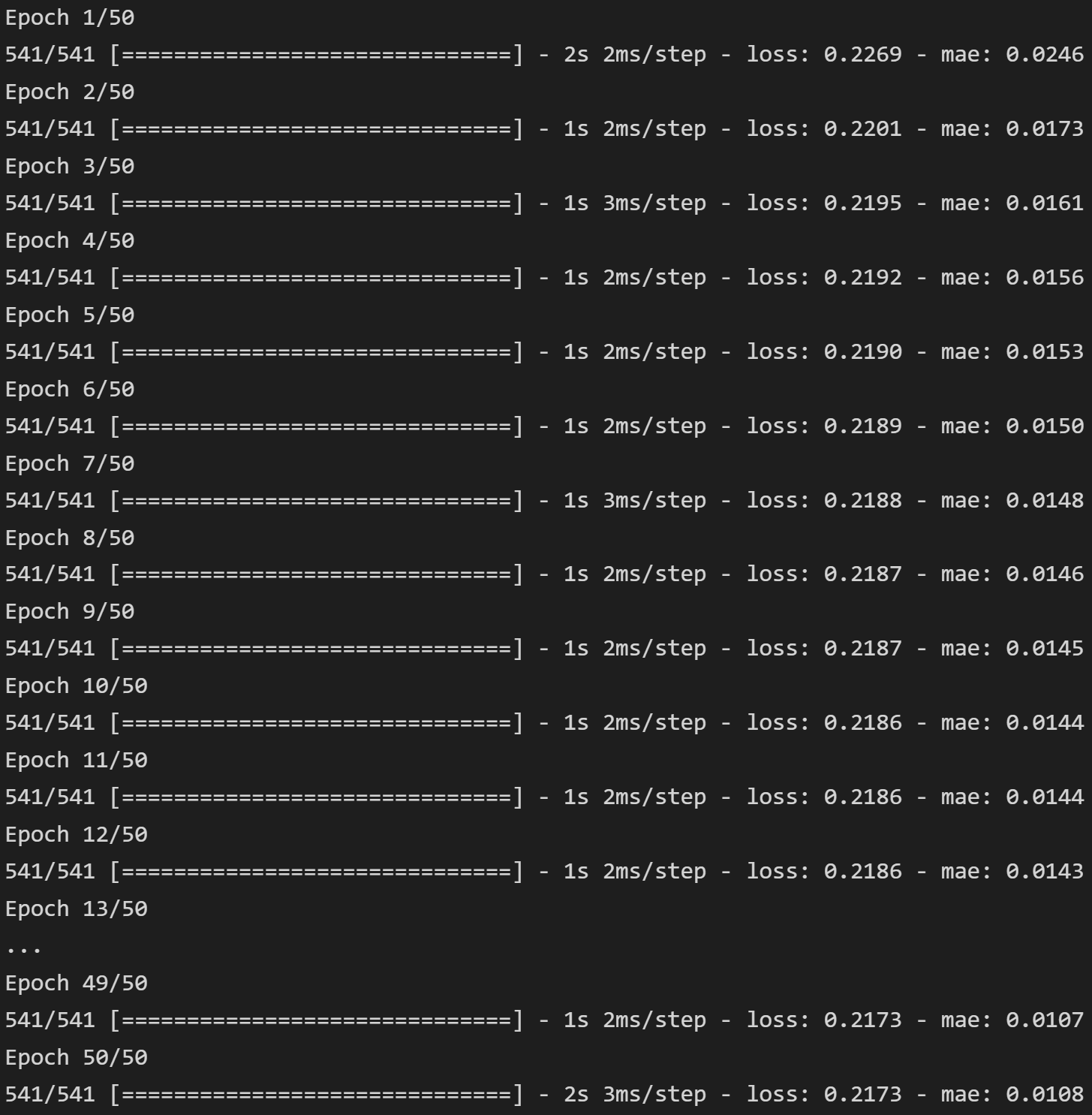


شکل 30- آموزش شبکه با RMSprop optimizer

در شکل31 نتیجه‌ی آموزش شبکه با Mean Absolute Error loss function و در شکل 32 نتیجه‌ی آموزش شبکه با Binary Cross-Entropy loss function را مشاهده می‌کنید.

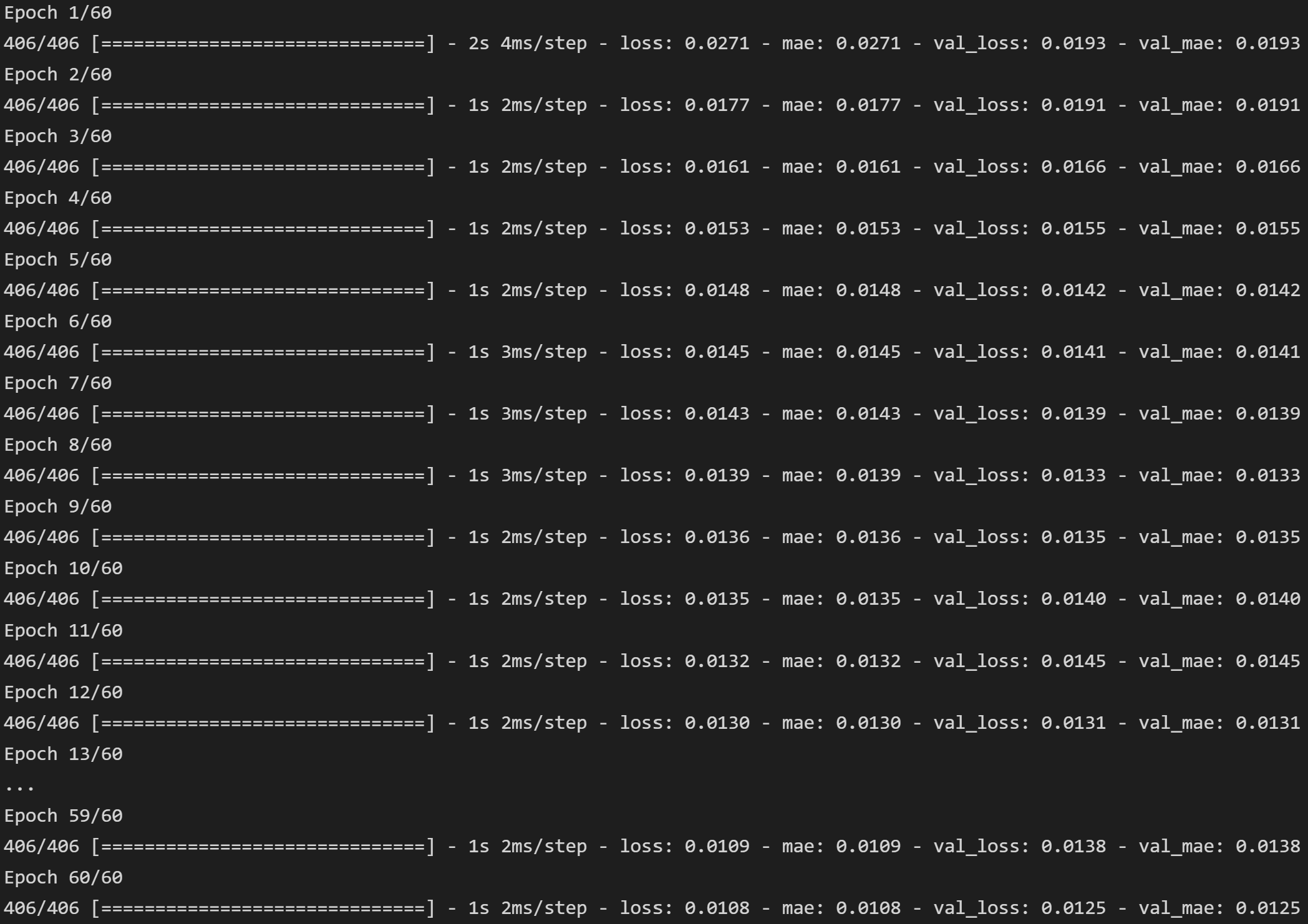


شکل 31 - آموزش شبکه با Mean Absolute Error loss function



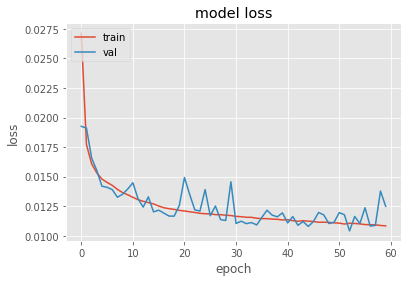
شکل 32 - آموزش شبکه با Binary Cross-Entropy loss function

**J)** ۲۵ درصد از داده‌های train را به عنوان validation در نظر گرفتیم و شبکه را آموزش دادیم که نتیجه‌ی آن را در شکل 33 مشاهده می‌کنید. از آنجایی که بهترین نتایج را با RMSprop optimizer و MAE loss function گرفته بودیم، شبکه را با آنها و با ۶۰ ایپاک train می‌کنیم.



شکل33 - آموزش شبکه MLP با داده‌ی train و validation

نمودار loss و validation loss را در شکل 34 مشاهده می کنید. نوسان میزان loss داده‌های validation با گذر ایپاک‌های بیشتر، کمتر می‌شود. پس به نظر می‌رسد که مدل، نسبتاً قابل قبول آموزش یافته است.



شکل 34 - نمودار train loss و validation loss بر حسب epoch

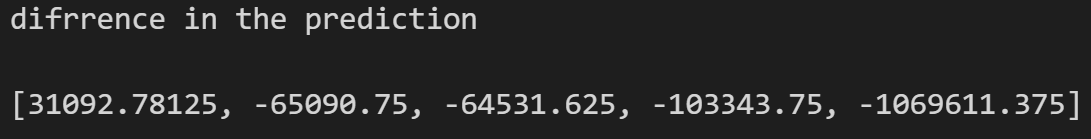
**K)** ۵ داده‌ی تصادفی انتخاب شده از مجموعه‌ی test عبارتند از:

[485000, 340000, 335606, 425000, 490000]

پیش‌بینی مدل به دادن این ۵ داده به صورت زیر است:

[453907.22, 405090.75, 400137.62, 528343.75, 1559611.4]

اختلاف قیمت پیش‌بینی شده توسط مدل با قیمت واقعی را در شکل 35 مشاهده می‌کنید:



شکل 35 - اختلاف قیمت پیش‌بینی شده توسط شبکه با قیمت واقعی