

دانشگاه صنعتي امیرکبیر  
(پلی تکنیک تهران)

دانشكده مهندسی برق

ﮔﺰﺍﺭﺵکار ﺁﺯﻣﺎﻳﺶ ﺩﻭﻡ

ﺁﺯﻣﺎﻳﺸﮕﺎﻩ مقدمه ای بر ﻫﻮﺵ ﻣﺤﺎﺳﺒﺎﺗﻲ

پیاده سازی پِرسِپترون

نگارش

ارشیا اسمعیل طهرانی

علی بابالو

پویا ابراهیمی

استاد راهنما

سرکار خانم موسوی

آبان ماه 1401

|  |  |
| --- | --- |
| فهرست مطالب | صفحه |

**پیش گزارش** ۱

نقش perceptron در شبکه عصبی ........................................................................................................................ ۱

انواع perceptron ....................................................................................................................................................... ۱

مدل ساده از perceptron ........................................................................................................................................ ۲

شباهت مدل artificial perceptron با ساختار perceptron در بدن انسان ............................................ ۳

**مقدمه ۴**

الگوریتم یادگیری perceptron .............................................................................................................................. ۴

**شرح آزمایش ۶**

محیط پایتون ............................................................................................................................................................... ۶

تابع فعال ساز tanh ................................................................................................................................................. ۱۹

تابع فعال ساز ReLU.............................................................................................................................................. ۲۰

تمرین مربوطه ........................................................................................................................................................... ۲۱

محیط matlab ........................................................................................................................................................ ۲۱

**بررسی به کار گیری الگوریتم Perceptron برای داده های زیر**  **۲۸**

**پیاده سازی گیت های And, Or, XOR ۲۹**

پیاده سازی And ..................................................................................................................................................... ۲۹

پیاده سازی Or ........................................................................................................................................................ ۳۴

پیاده سازی XOR .................................................................................................................................................. ۳۹

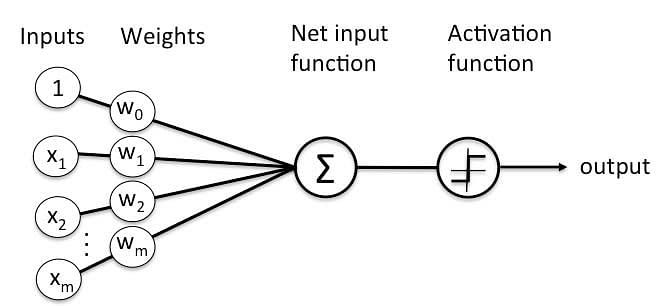
دیتاست Iris ............................................................................................................................................................ ۴۲

منابع و مواخذ .......................................................................................................................................................... ۴۶

# پیش گذارش

## نقش Perceptron در شبکه عصبی

پرسپترون توسط فرانک روزنبلات در سال 1957 معرفی شد. او قانون یادگیری پرسپترون را بر اساس نورون اصلی MCP پیشنهاد کرد. پرسپترون الگوریتمی برای Supervised Learning از طبقه بندی کننده های باینری است. این الگوریتم نورون‌ها را قادر می‌سازد تا عناصر مجموعه آموزشی را در یک زمان یاد بگیرند و پردازش کنند.



شکل 1) یک مدل ساده از Perceptron

### انواع Perceptron

تک لایه: پرسپترون تک لایه فقط می تواند الگوهای قابل جداسازی خطی را یاد بگیرد.

چند لایه: پرسپترون های چندلایه می توانند در مورد دو یا چند لایه که دارای قدرت پردازش بیشتری هستند یاد بگیرند.

الگوریتم پرسپترون وزن سیگنال های ورودی را برای ترسیم یک مرز تصمیم گیری خطی می آموزد.

توجه: Supervise Learning نوعی از یادگیری ماشینی است که برای یادگیری مدل‌ها از داده‌های آموزشی برچسب‌گذاری شده استفاده می‌شود. پیش‌بینی خروجی برای داده‌های آینده یا دیده نشده را ممکن می‌سازد. اجازه دهید در بخش بعدی روی قانون یادگیری پرسپترون تمرکز کنیم.

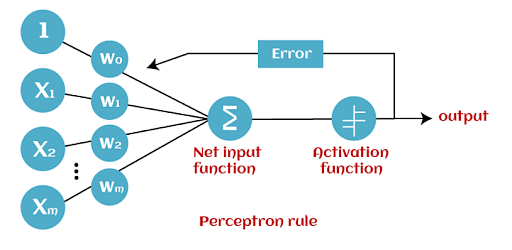
### Perceptron دریادگیری ماشین

رایج ترین اصطلاح در هوش مصنوعی و یادگیری ماشین (AIML) Perceptron است. این مرحله ابتدایی یادگیری کدنویسی و فناوری‌های یادگیری عمیق است که شامل مقادیر ورودی، امتیازها، آستانه‌ها و وزن‌های پیاده‌سازی گیت‌های منطقی است. پرسپترون مرحله پرورش یک پیوند عصبی مصنوعی است. در قرن 19، آقای فرانک روزنبلات Perceptron را اختراع کرد تا محاسبات سطح بالا را برای شناسایی قابلیت های داده های ورودی یا هوش تجاری انجام دهد. با این حال، اکنون از آن برای اهداف مختلف دیگری استفاده می شود.

### مدل پرسپترون در یادگیری ماشینی چیست؟

یک الگوریتم مبتنی بر ماشین که برای Supervised Learning، وظایف مرتب‌سازی باینری مختلف استفاده می‌شود Perceptron نام دارد. علاوه بر این، Perceptron همچنین نقش اساسی به عنوان یک نورون مصنوعی یا پیوند عصبی در تشخیص محاسبات داده های ورودی خاص در هوش تجاری دارد. مدل پرسپترون نیز به عنوان یکی از بهترین و خاص ترین انواع شبکه های عصبی مصنوعی طبقه بندی می شود. به عنوان یک الگوریتم یادگیری نظارت شده از طبقه‌بندی‌کننده‌های باینری، می‌توانیم آن را یک شبکه عصبی تک لایه با چهار پارامتر اصلی در نظر بگیریم: مقادیر ورودی، وزن‌ها و بایاس، مجموع خالص و تابع فعال‌سازی.

## مدل ساده از Perceptron

پرسپترون یک پیوند عصبی تک لایه با چهار پارامتر اصلی در نظر گرفته می شود. مدل پرسپترون با ضرب تمام مقادیر ورودی و وزن آنها شروع می شود، سپس این مقادیر را برای ایجاد مجموع وزنی اضافه می کند. علاوه بر این، این جمع وزنی برای به دست آوردن خروجی مورد نظر به تابع فعال سازی 'f' اعمال می شود. این تابع فعال سازی به عنوان تابع گام نیز شناخته می شود و با "f" نشان داده می شود.

شکل 2) یک مدل دیگر ساده از Perceptron

## شباهت مدل Artificial Perceptron با ساختار perceptron در بدن انسان

|  |  |
| --- | --- |
| Biological Neuron | Artificial Neuron |
| Cell Nucleus (Soma) | Node |
| Dendrites | Input |
| Synapse | Weights or interconnections |
| Axon | Output |

جدول 1) نورون بیولوژیکی در مقابل نورون مصنوعی

ورودی های پرسپترون مانند دِندریت ها عمل می‌کنند و اطلاعات را دریافت می‌کنند. بدنه ی اصلی نورون ها مشابه تابع فعال ساز عمل می‌کند که پس از رسیدن به حد آستانه (Threshold) فعال شده و سیگنال خروجی تولید می‌شود که معادل آکسون ها در نورون های بیولوژیکی می‌باشد. و سپس خروجی آکسون ها توسط سیناپس ها به نورون بعدی منتقل می‌شود.

# مقدمه

در این آزمایش قصد داریم به پیاده سازی Perceptron در محیط های Python و MATLAB بپردازیم تا عمل Classification بر روی داده هایی انجام شود. مدل کلی Perceptron، به همراه رابطه خروجی آن، در ذیل آورده شده است.



شکل 3) مدل پرسپترون در شبکه عصبی

خروجی پرسپترون بر اساس ورودی های آن در رابطه زیر توضیف شده است.

وزن های مدل پرسپترون توسط الگوریتم یادگیری پرسپترون آپدیت می‌شوند که در ادامه به بررسی این الگوریتم پرداخته می‌شود.

## الگوریتم یادگیری Perceptron

هدف این الگوریتم، پیدا کردن بهترین وزن های شبکه برای تشخیص صحیح کلاس داده های ورودی است.

|  |
| --- |
| **ورودی شبکه:**  P تا زوج دو تایی *{*(*x1, d1*)*...*(*xp, dp*)*} که بردار x مقادیر ورودی هر نمونه و بردار با ابعاد واحد d، خروجی مورد نظر متناظر با آن بردار است.*  **خروجی شبکه:**  مدل آموزش دیده (وزن های w مناسب) Perceptron می‌باشد.  **مراحل:**   1. مقادیر انتخاب می‌شوند. 2. وزن ها به صورت مقادیر رندوم کوچک مقداردهی می‌شوند. 3. شمارنده ها و متغیر خطا مقداردهی می‌شوند. 4. آغاز سیکل یادگیری؛ 5. وزن ها بر اساس رابطه مقابل، بروز رسانی می‌شوند: 6. مقدار خطا بر اساس رابطه ی روبرو محاسبه می‌شود. 7. اگر باشد، آن‌گاه: 8. ، برو به مرحله 4 9. در غیر این صورت: 10. به مرحله 12 برو. 11. پایان شرط. 12. اگر باشد، آنگاه: 13. یادگیری پایان یافته است. 14. در غیر این صورت: 15. مرحله بعدی یادگیری را شروع کن و به مرحله 17 برو. 16. پایان شرط. 17. اگر باشد، آن گاه: 18. در غیر این صورت: 19. یادگیری پایان یافته است 20. پایان شرط |

شکل 4) الگوریتم یادگیری Perceptron

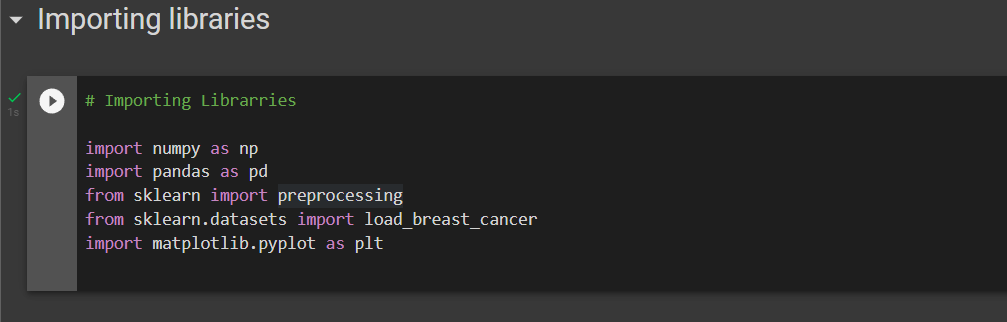
در این الگوریتم وزن های شبکه در ابتدا با استفاده از اعداد تصادفی مقدار دهی اولیه می‌شوند. سپس تا هنگامی که وزن ها همگرا نشده اند، وزن ها بر اساس تفاوت بین کلاس پیش بینی شده توسط شبکه، و کلاس اصلی داده ورودی آپدیت می‌شوند.

# شرح آزمایش

در این قسمت به پیاده سازی Perceptron در محیط Python و MATLAB پرداخته می‌شود.

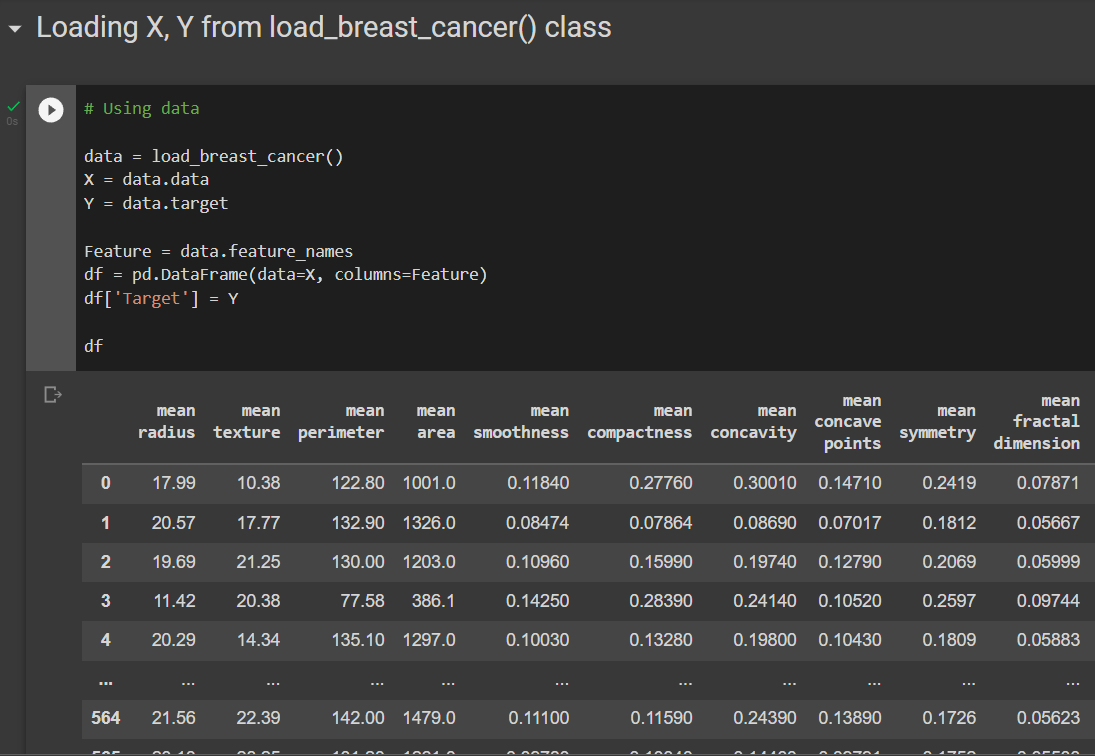
## محیط Python

توضیحات ذیل مرتبط با ابتدای آزمایش تا تمرین اول می‌باشد. توجه شود که snippet های داخل گزارشکار مربوط به کد نویسی functional می‌باشد. بخش ماژولار و به صورت OOP در کد آمده است.



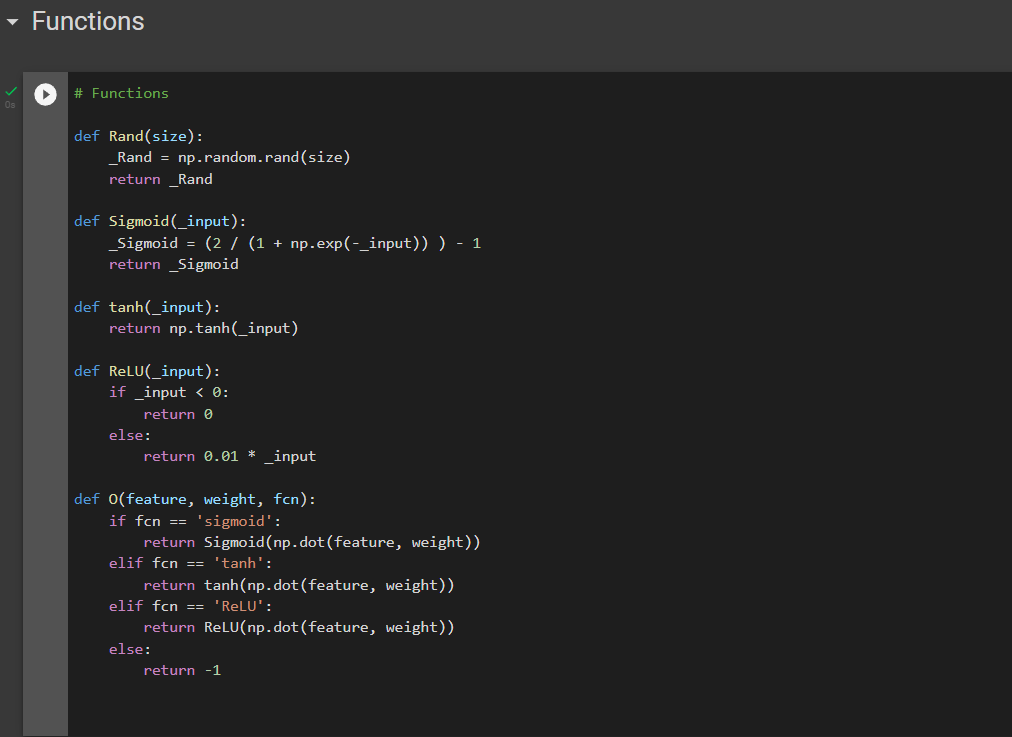
شکل 5) ماژول ها و کتابخانه های مربوطه را اضافه می‌کنیم.

توجه شود که از pandas برای dataframe استفاده می‌کنیم.



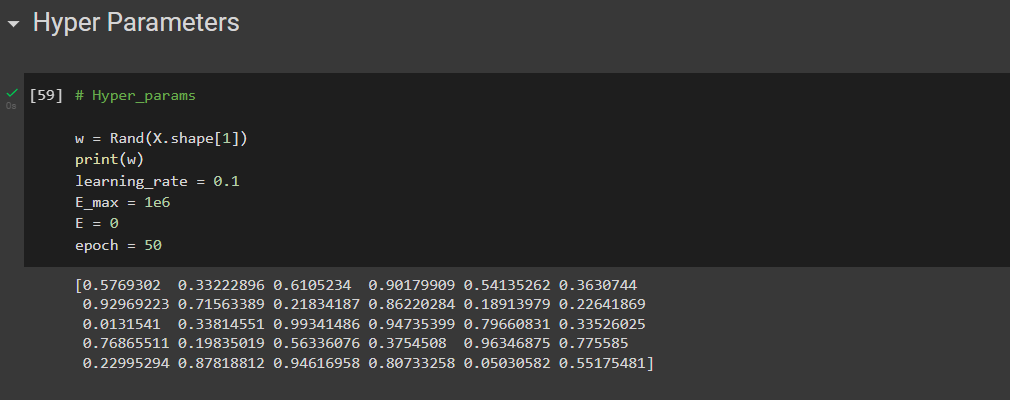
شکل 6) ورودی ها و خروجی ها از دیتاست استخراج می‌شوند.

همانطور که مشاهده می‌شود، در آخر در خروجی واضح است که چگونه دیتا استخراج شده است.

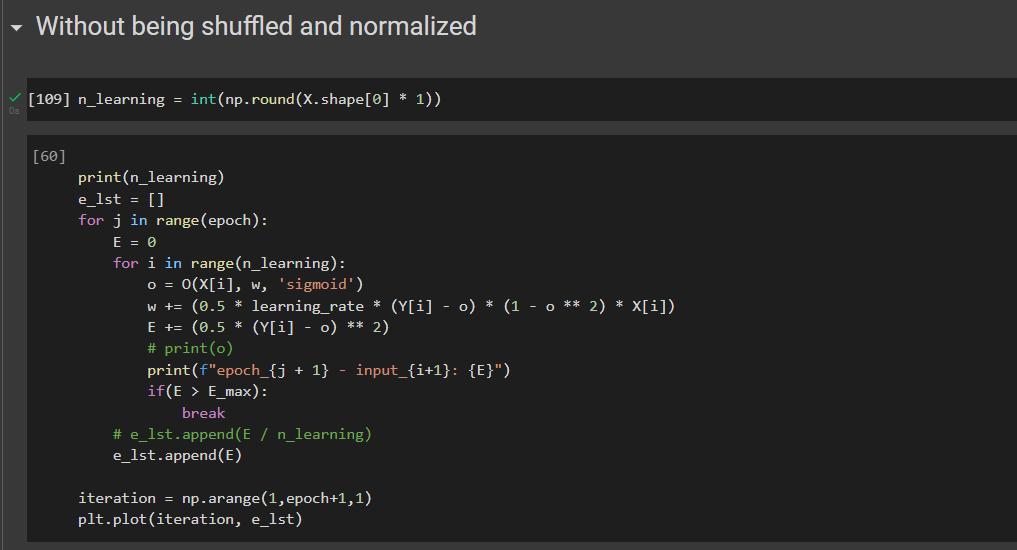


شکل 7) در ادامه تمامی توابع مربوطه را مشاهده میکنید.

به ترتیب توابع مربوط به تولید مقادیر رندم( در ادامه برای مقدار دهی رندم به وزن ها)، توابع فعال ساز Sigmoid، tanh، ReLU و همچنین تابع خروجی O نیز می‌باشد.



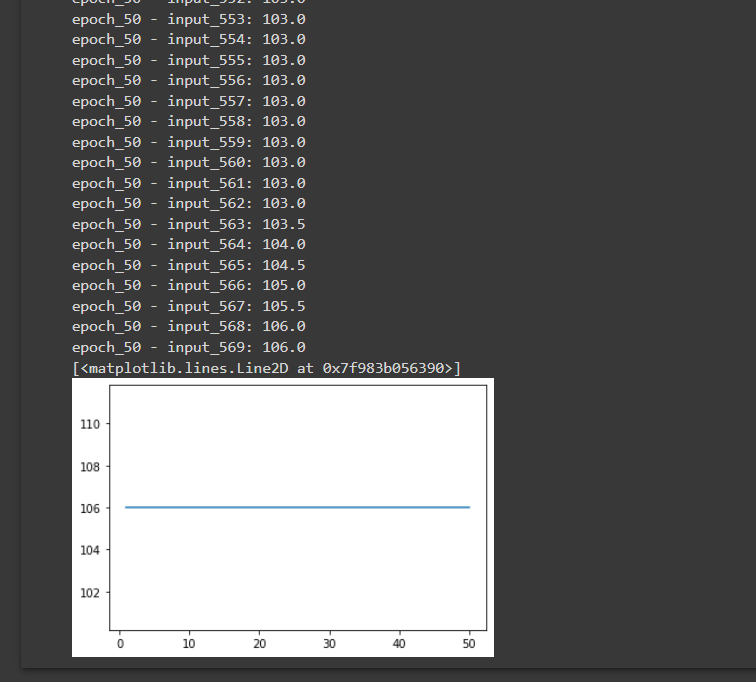
شکل 8) در این قسمت Hyperparameter ها مقدار دهی می‌شوند.



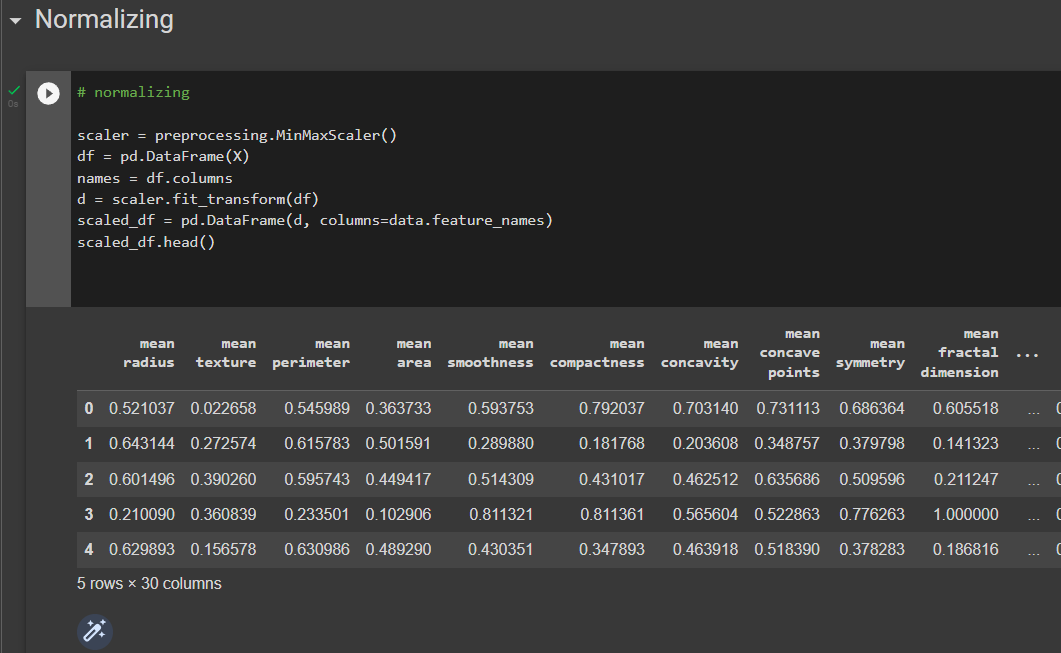
شکل 9) در این قسمت، سیستم بدون در نظر گرفتن بایاس با همه ورودی ها آموزش داده می‌شود.

ملاحظه می‌کنید که با تابع فعال ساز sigmoid سیستم آموزش داده شده است.

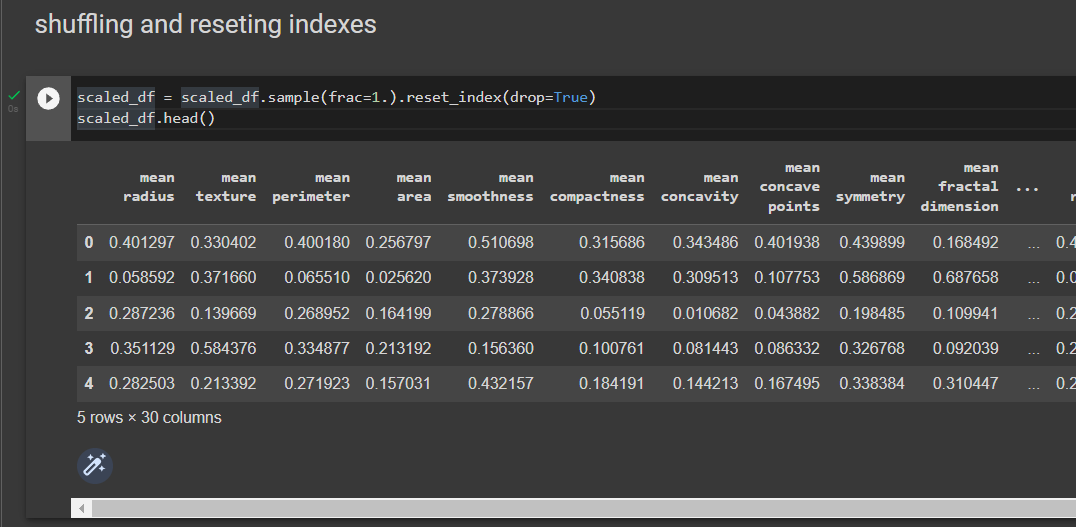
در ادامه پس از epoch=50، ارور را مشاهده می‌کنید که بر روی مقدار 106 همگرا شده است.



شکل 10) پس از 50 بار آموزش دیدن بر روی همه ورودی ها، نمودار خطا بر حسب تکرار یا epoch به صورت فوق مشاهده می‌شود.

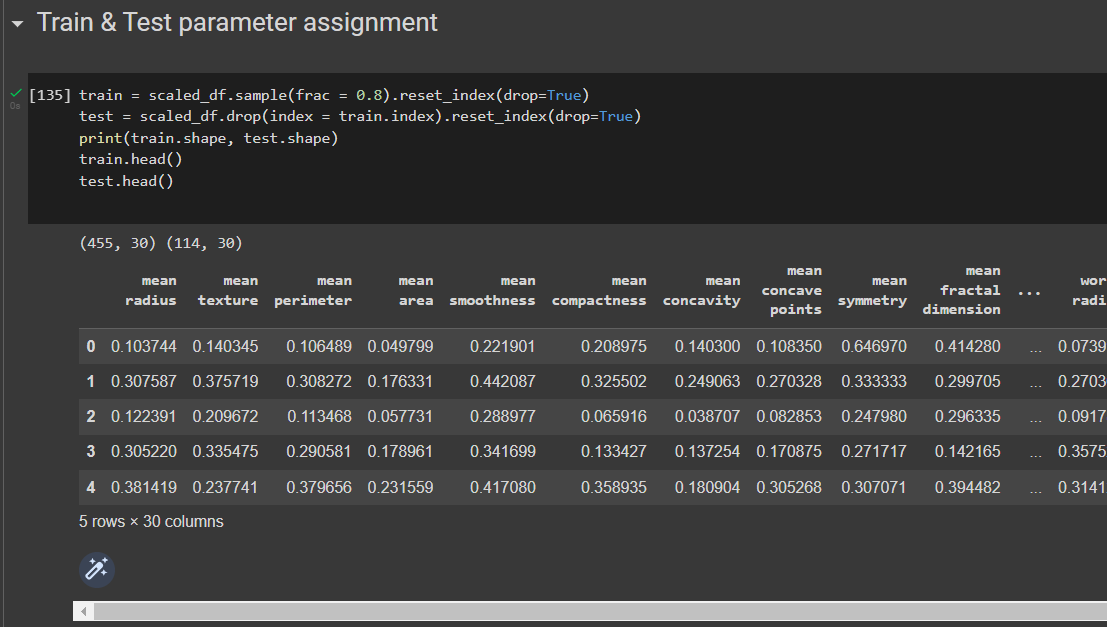


شکل 11) در این تکه کد، با استفاده از توابع موجود در pandas و باقی توابع، ورودی ها نرمال سازی شده اند.

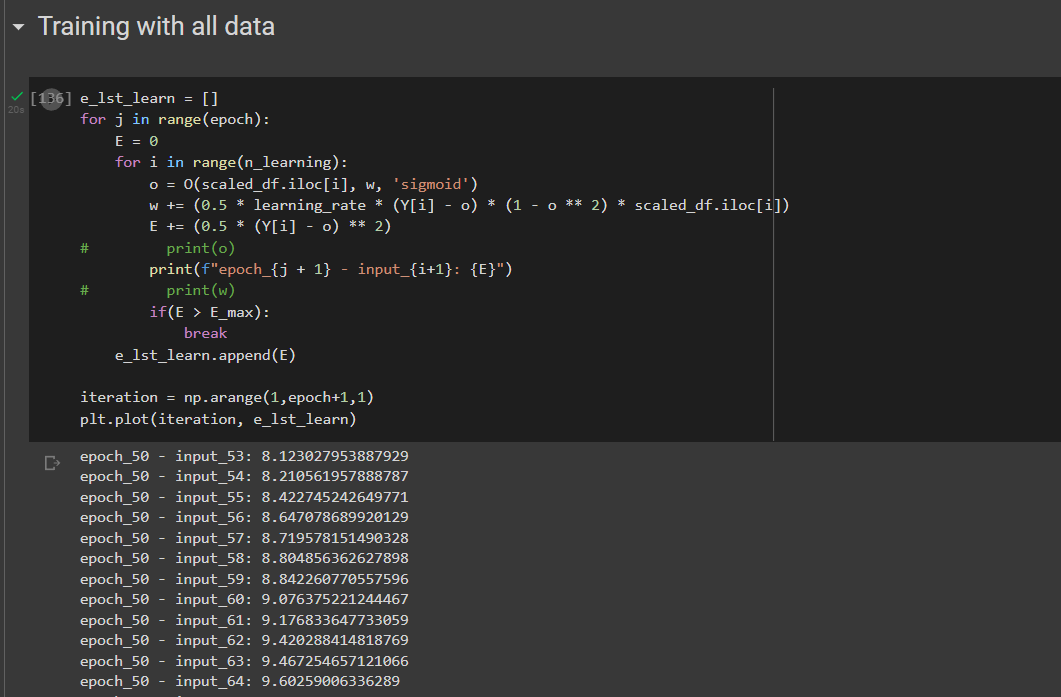


شکل 12) در این تکه کد، عمل shuffling و reset کردن اندیس ها اتفاق افتاده است.

متد sample با frac=1، همه داده ها را به صورت تصادفی انتخاب کرده و متد reset\_index با drop=true، اندیس جابه جا شده به علت تصادفی انتخاب کردن اندیس ها را reset میکند.

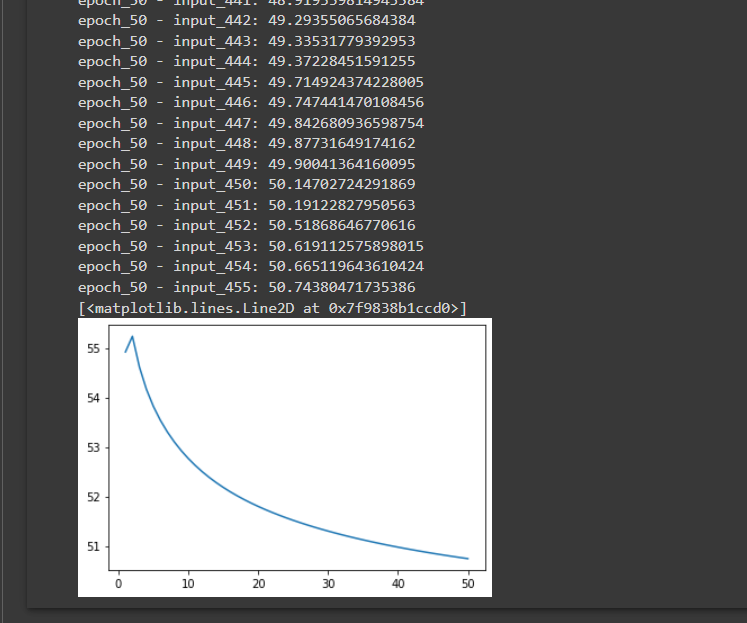


شکل 13) در این قسمت همانند شکل 12، داده های train و test به گونه ای که 80% کل داده ها با train و باقی داده ها به test اختصاص داده شده است، مقدار دهی می‌شوند.

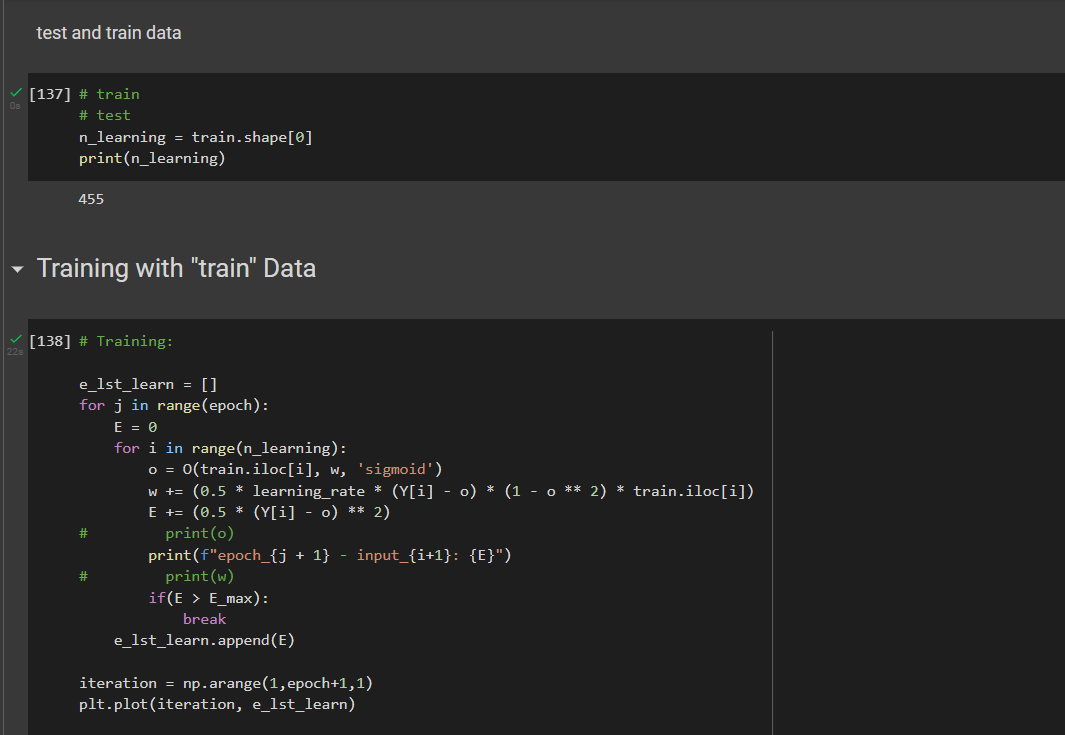


شکل 14) در این قسمت، پس از نرماله شدن و شافل شدن متغیر ها، دوباره آموزش داده می‌شوند.

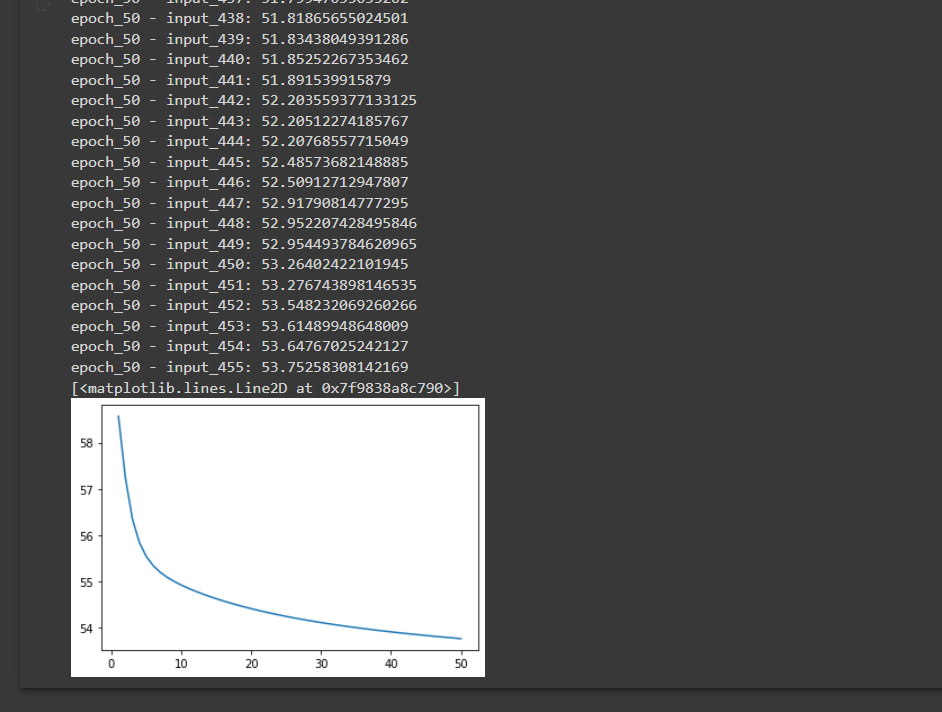
در نمودار شکل بعد ملاحظه می‌شود که با همان hyperparameter ها و همان تابع فعال ساز، به طور کلی بدون دست خوردن تنظیمات، اما وقتی ورودی ها نرماله سازی شده باشند، چقدر تفاوت در MSE ملاحظه می‌شود.



شکل 15) ملاحظه می‌شود که ارور به حد قابل توجه ای کاهش پیدا کرد.

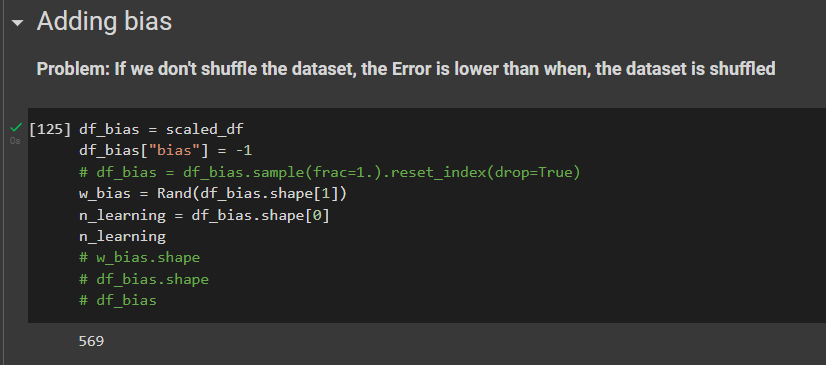


شکل 16) حال در این مرحلطه آموزش با ورودی داده های train انجام می‌شود و نتیجه را در شکل بعدی ملاحظه خواهید نمود.



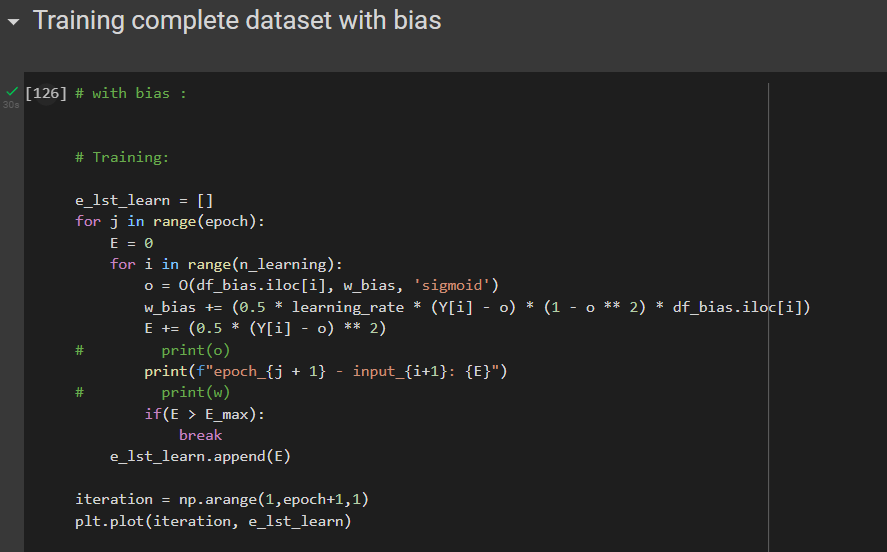
شکل 17) نتیجه به صورت فوق قابل مشاهده است.

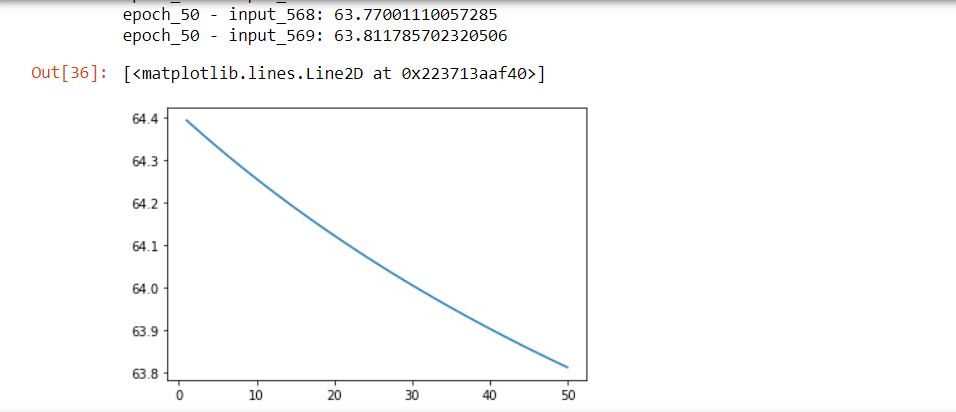
در مراحل بعد، به سیستم بایاس هم اضافه می‌شود و نتایج با توجه به آن هم محاسبه می‌شود.



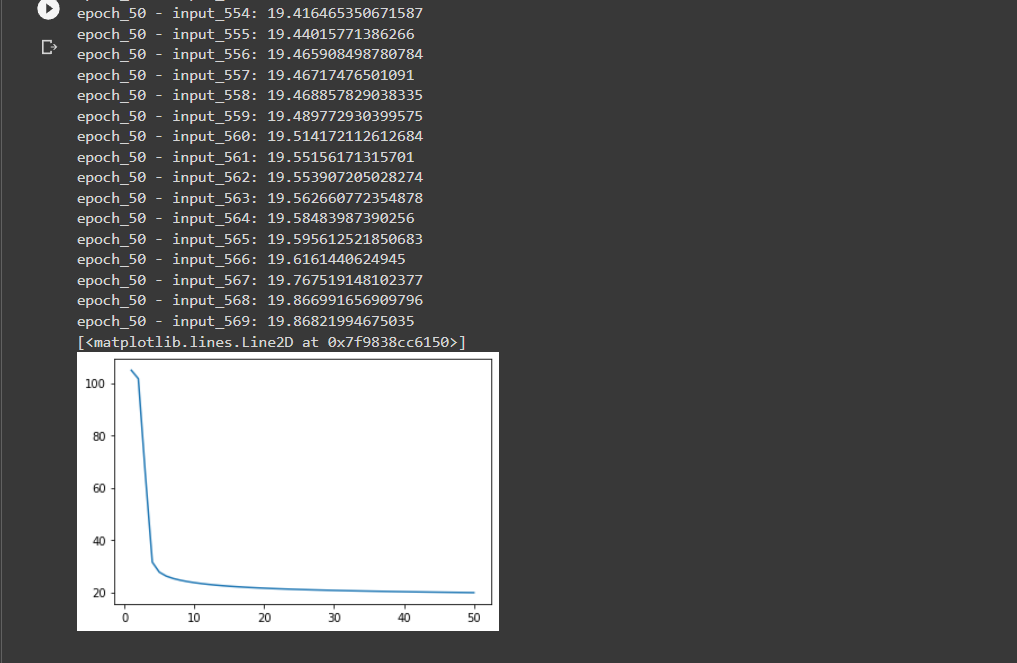
شکل 18) دیتا فریم با بایاس، به صورت فوق ساخته می‌شود.

نکته ای که حائز اهمیت است این است که در با همان تنظیمات و هایپر پارامتر ها، در صورت عدم شافل کردن داده ها، ارور کمتری به دست می‌آید تا زمانی که داده ها را شافل کنیم.

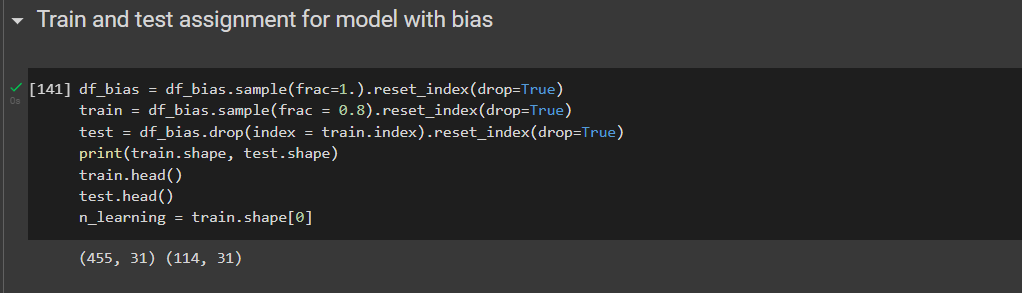


شکل 19) در این قسمت تمامی دیتا ست را به همراه بایاس آموزش داده ایم که نتایج به صورت شکل زیر قابل مشاهده است.

**شکل 20-1 خروجی MSE با دیتاست نرمال شده و بایاس با eta = 1**

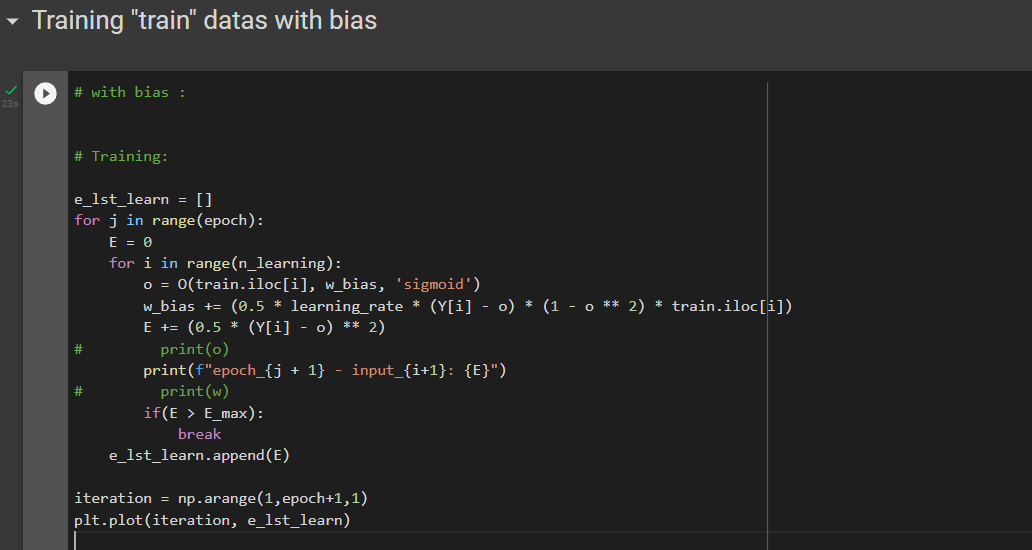


شکل 20-2) نتایج تحلیل MSE نشان میدهد که با همان تنظیمات و فقط به علت وجود پارامتر بایاس در شبکه، MSE چقدر کاهش پیدا کرد و تقریبا به 1/3 مقدار بدون بایاس میل کرد.

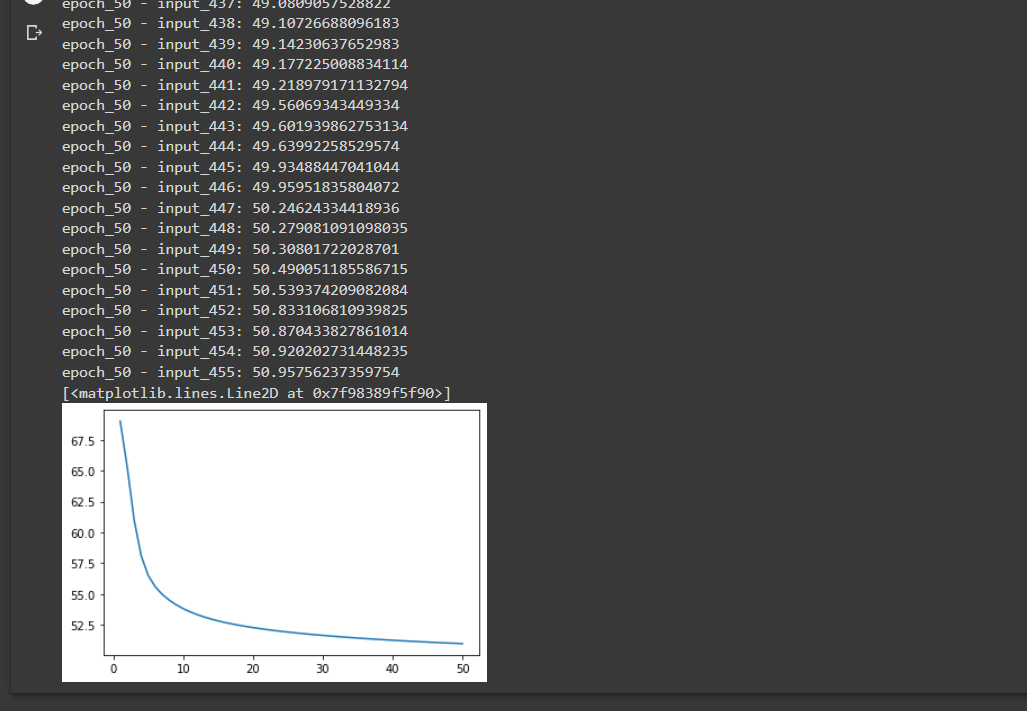


شکل 20) در این قسمت داده های train و test نیز تخصیص داده می‎‌شوند

روند کار به این صورت است که ابتدا کل داده ها شافل می‌شوند، سپس همانطور که پیش تر ذکر شده بود، با کمک توابع sample و reset\_index، ترتیب اندیس ها به خورده و همینطور ورودی هایی به train و test تخصیص داده می‌شود.



شکل 21) سپس در مرحله ی آخر، شبکه فقط با داده های train آموزش داده می‌شود و در ادامه نتایج قابل مشاهده است.



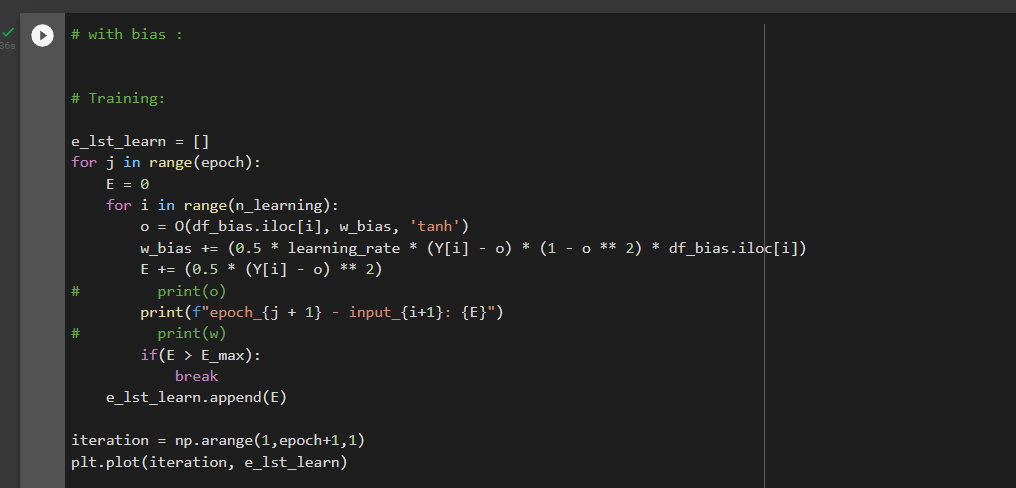
شکل 22) نتایج قابل مشاهده است.

بدیهی است که به علت کمتر بودن داده های train نسبت به کل دیتاست، ارور بیشتری (MSE) قابل مشاهده است.

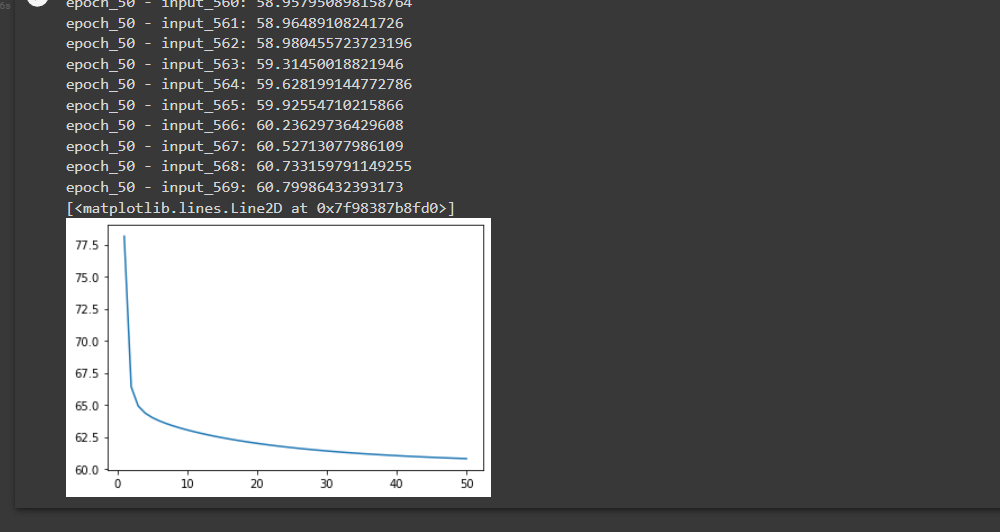
در تحلیل اعمال انجام شده، شایان ذکر است که درست است که روی کل داده اگر شبکه آموزش داده شود دقت بسیار بالایی برای همان داده ها حاصل می‌شود. اما، ممکن است در configuration ای از تنظیمات، در صورتی که داده جدید به سیستم وارد شود، به علت overfitting روی داده ها، آن داده های جدید با خطای بسیار زیادی پیش بینی شوند. و راه حل در همین استفاده از داده های train برای آموزش و تست فرضی داده های test روی سیستم است.

حال برای حالت شبکه با بایاس، با همان پارامتر ها اما با توابع activation دیگر خواهیم داشت:

### TANH



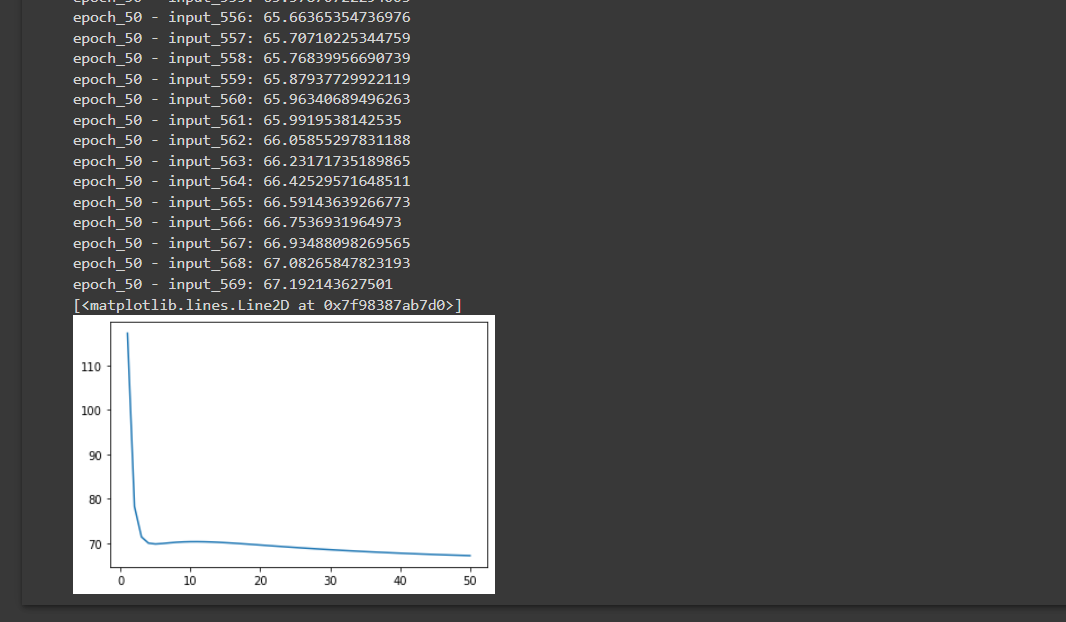
شکل 23) به صورت قابل مشاهده در شکل، پارامتر را به tanh تغییر میدهیم



شکل 24) خروجی را ملاحظه می‌کنید که دقت و خطای بیشتری را نسبت به حالت تابع فعال ساز sigmoid ارائه می‌کند.

### ReLU

خروجی برای تابع ReLU به صورت زیر می‌باشد



شکل 25) مشاهده می‌شود که MSE از دو حالت قبلی بیشتر است و نشان دهنده این است که با تنظیمات پارامتر های یکسان، MSE در حالت تابع فعال ساز sigmoid از همه آن ها کمتر است.

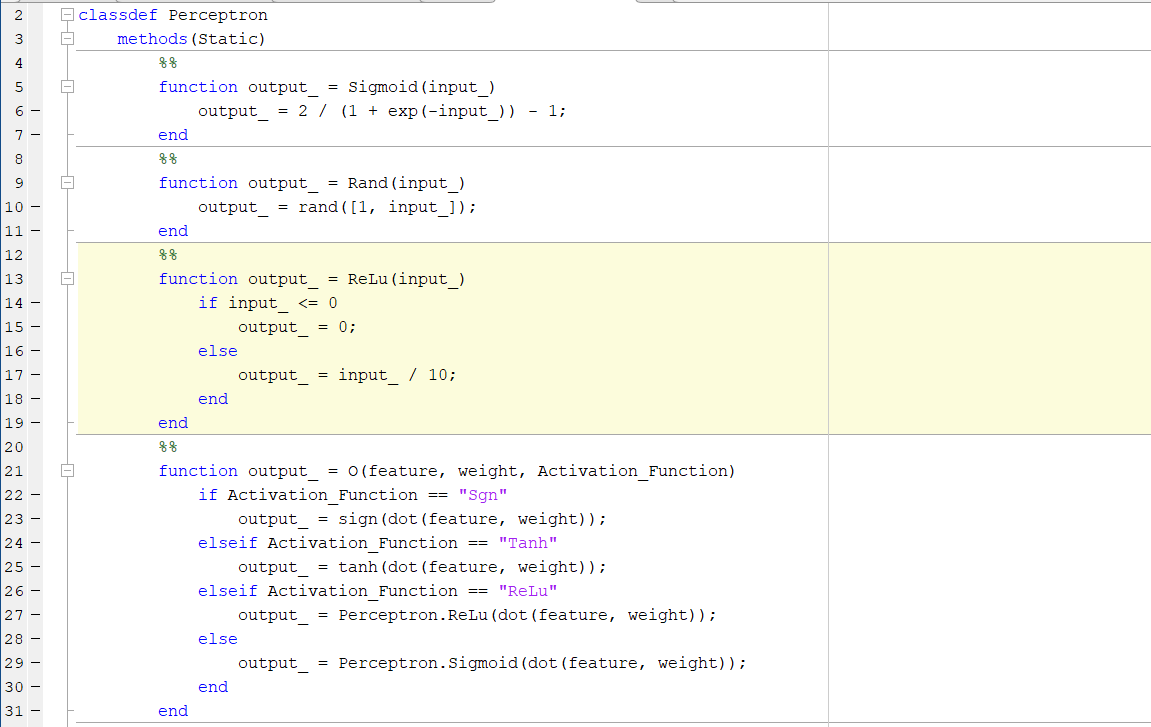
## تمرین مربوطه

اثر اضافه کردن بایاس، در این است که سرعت همگرایی نیز بیشتر می‌شود و مدل به دقت بهتری می‌رسد.

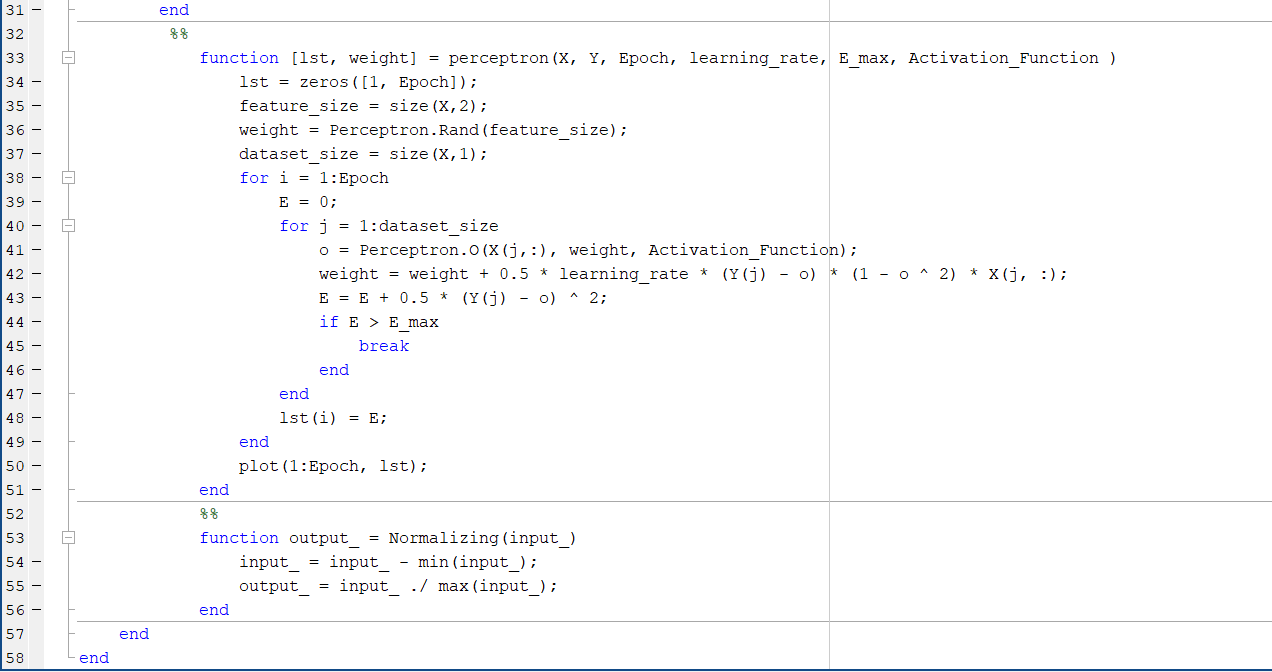
## محیط MATLAB

در MATLAB از همان ابتدا script به صورت Class نوشته شده است و از OOP استفاده شده است.

شایان ذکر است که در MATLAB صرفا syntax و برخی نکات جزئی با Python متفاوت است، و الگوریتم و روند کلی کار مشابه است. بنابراین توضیحات اضافه آورده نمی‌شود.



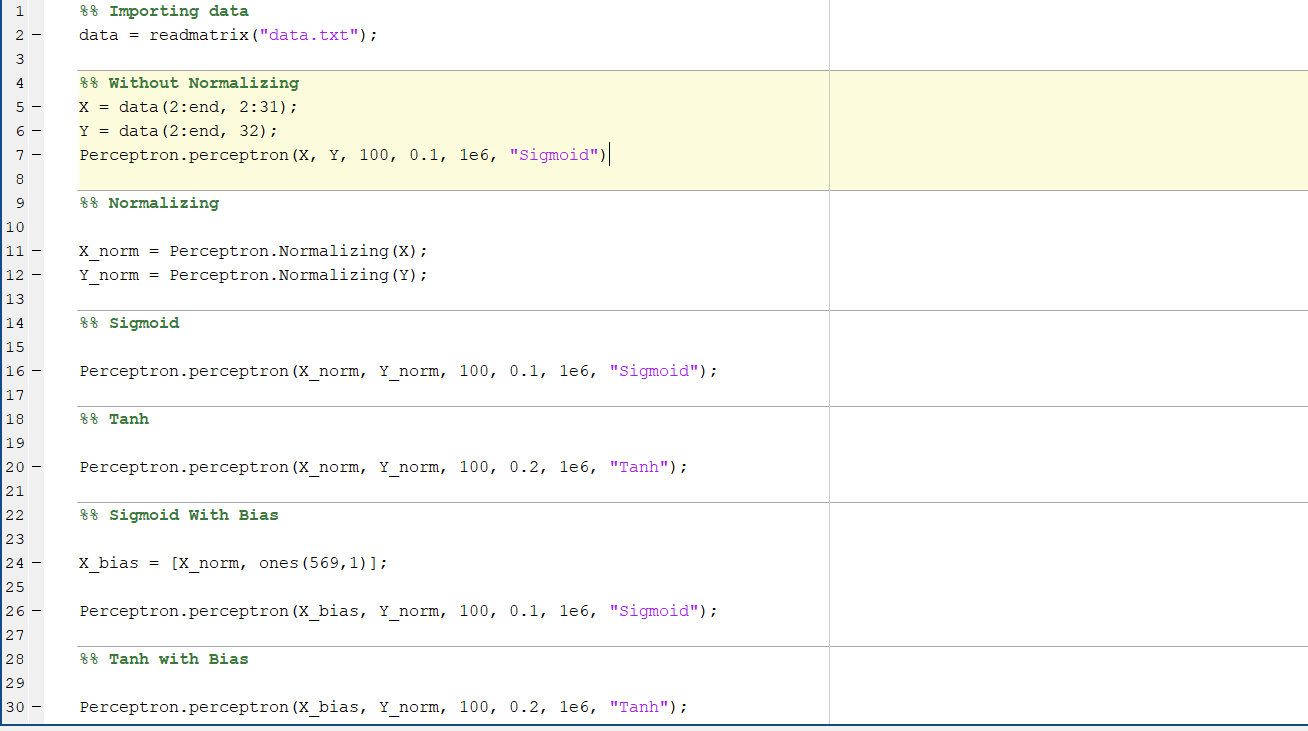
شکل 26) در این شکل مشاهده می‌شود که شبکه در داخل کلاس Perceptron نیز پیاده سازی شده است.



شکل 27) در ادامه شکل قبل، این شکل تکمیل کننده فایل "Perceptron.m" نیز می‌باشد.

پس از پیاده سازی class، در فایل "Breast\_Cancer.m" باقی Script آورده شده است و خواسته ها نیز پیاده سازی گردیده است.

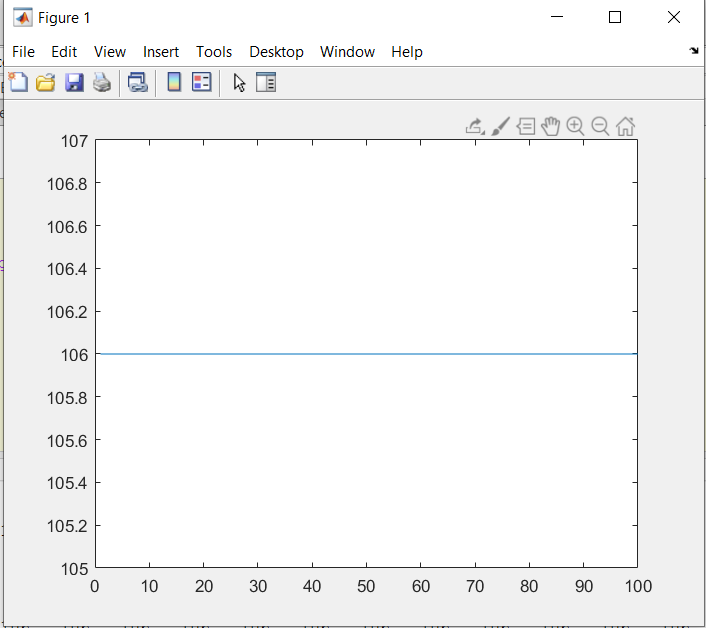
در ادامه محتویات داخل فایل ذکر شده قابل مشاهده است.



شکل 28) Script های استفاده از کلاس Perceptron

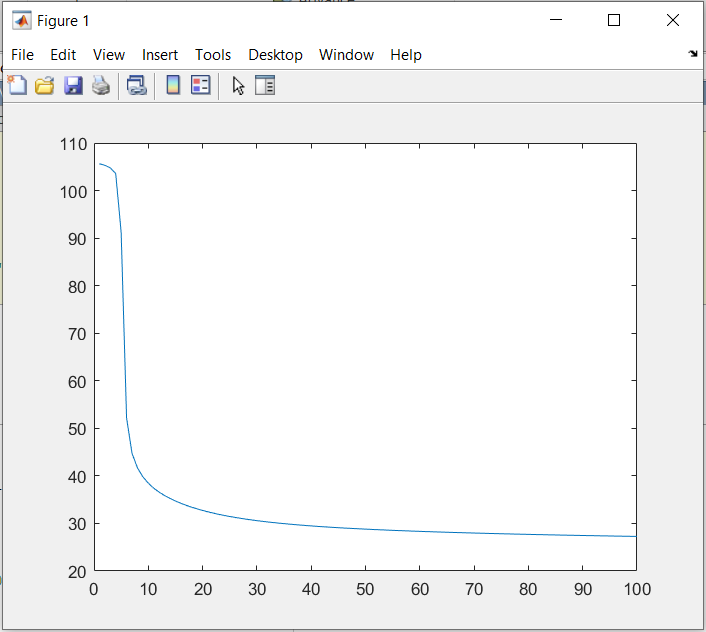
### نتایج

#### MSE بدون نرمال سازی



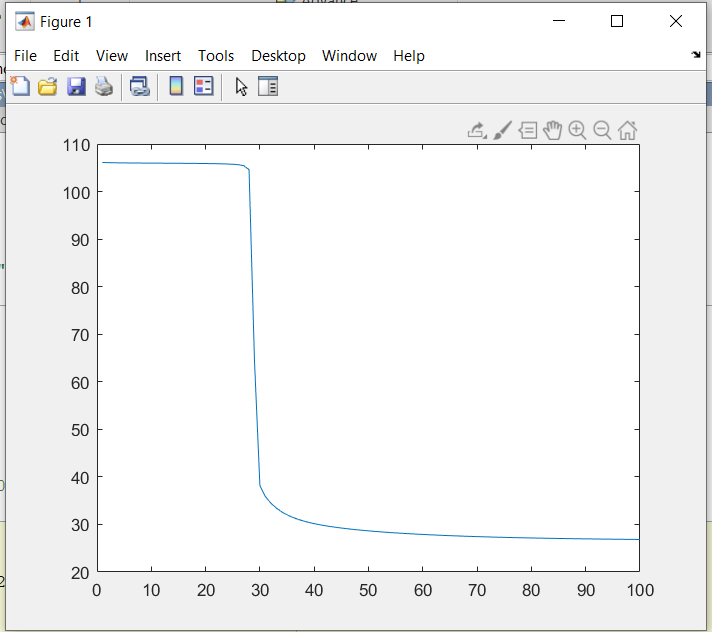
شکل 29) Perceptron بدون نرماله سازی

#### فعال سازی Sigmoid با نرماله سازی



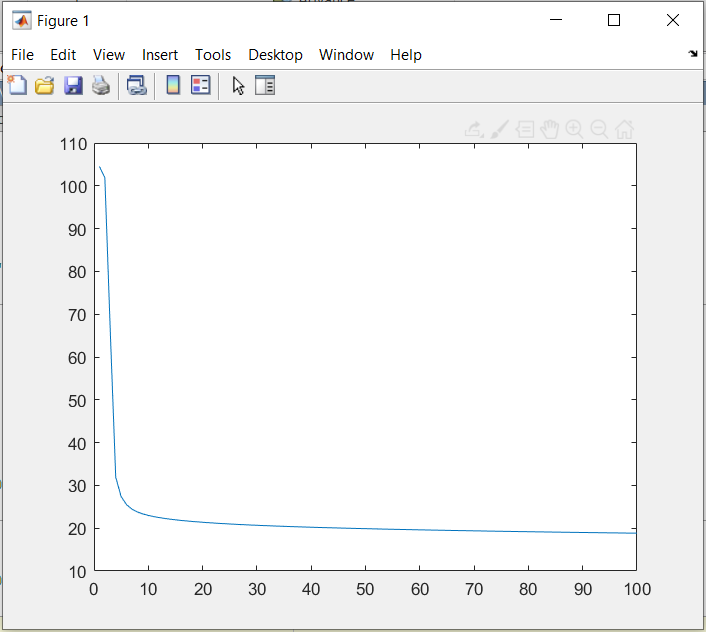
شکل 30) نمودار MSE برای فعال سازی sigmoid با نرمال سازی

#### فعال سازی tanh با نرماله سازی



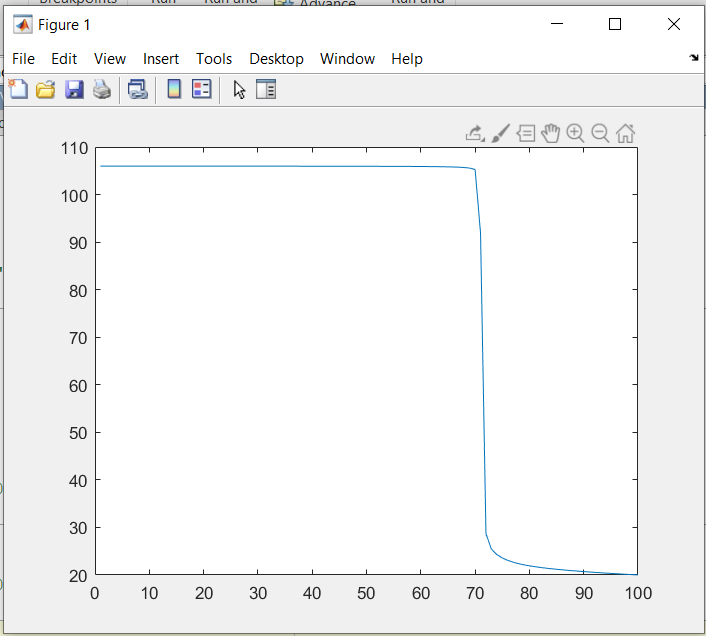
شکل 31) نمودار MSE برای فعال ساز tanh با نرماله سازی

#### فعال ساز sigmoid با بایاس و نرماله شده



شکل 32) نمودار MSE برای فعال ساز Sigmoid با بایاس و نرماله شدن

#### فعال سازی tanh با بایاس و نرماله شدن



شکل 33) نمودار MSE برای فعال سازی tanh با بایاس و نرماله شدن

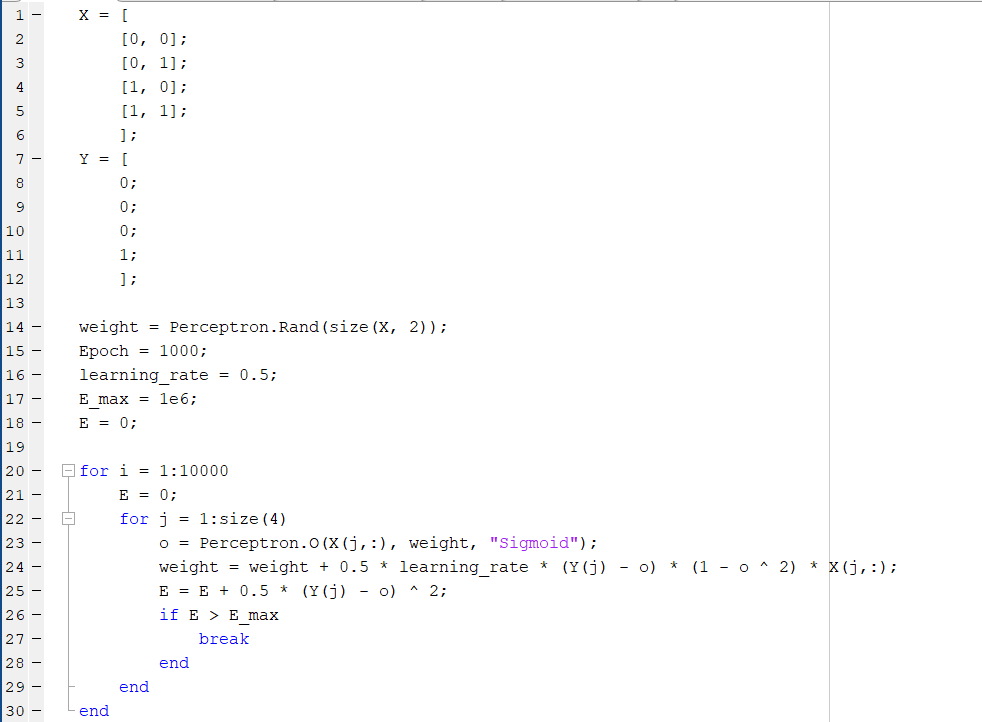
## بررسی به کار گیری الگوریتم Perceptron برای داده های زیر

شکل 34) داده های گلبهی رنگ گروه a و آبی رنگ گروه b

اگر بخواهيم تنها يك لايه Perceptron را به كار بگيريم، جداسازي داده ها در اين حالت ناممكن است چرا كه يك لايه ي Perceptron امكان جداسازي توابع غيرخطي را ندارد و مدل همگرا نمي شود. اما با به كارگيري چندين لايه، جداسازي اين داده ها به سادگي قابل انجام است.

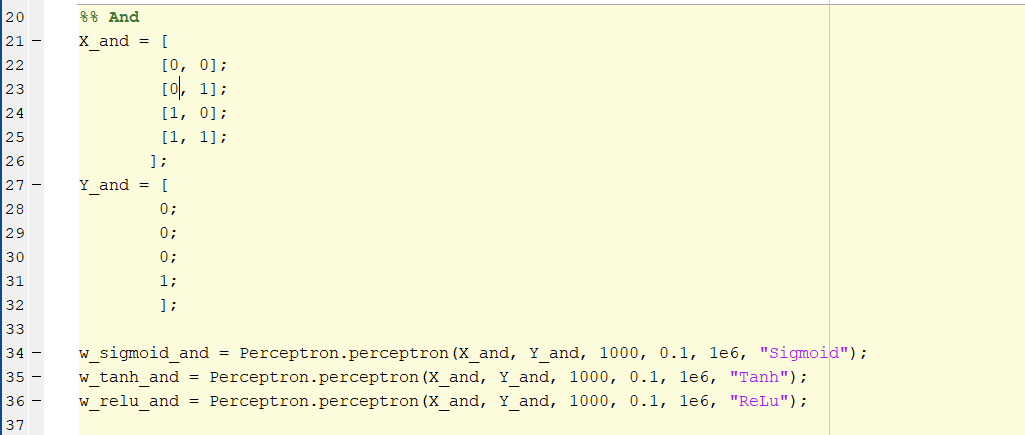
# پیاده سازی توابع XOR، OR، و AND با Perceptron در MATLAB

## پیاده سازی AND



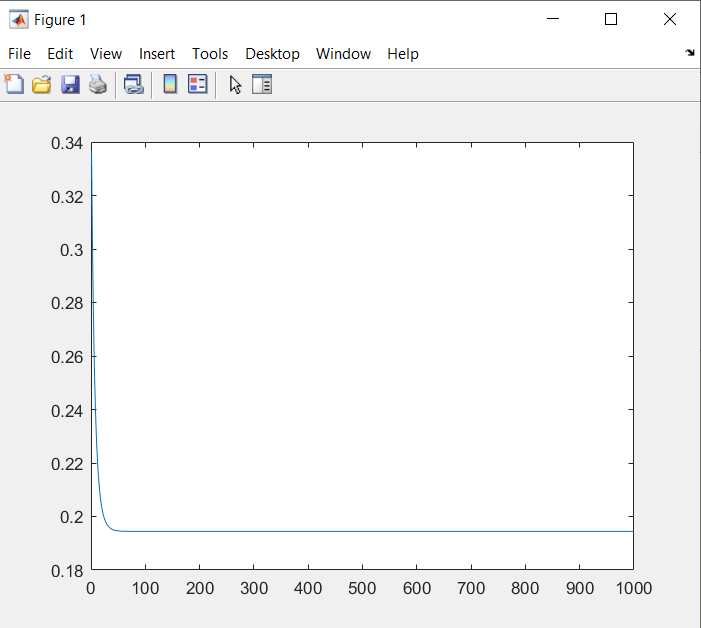
شکل 35) در فایل “And.m” نیز Script فوق موجود است و در آن از کلاس Perceptron نیز استفاده شده است.

همچنین در فایل "Modularized.m" نیز از این فایل ها استفاده شده است.



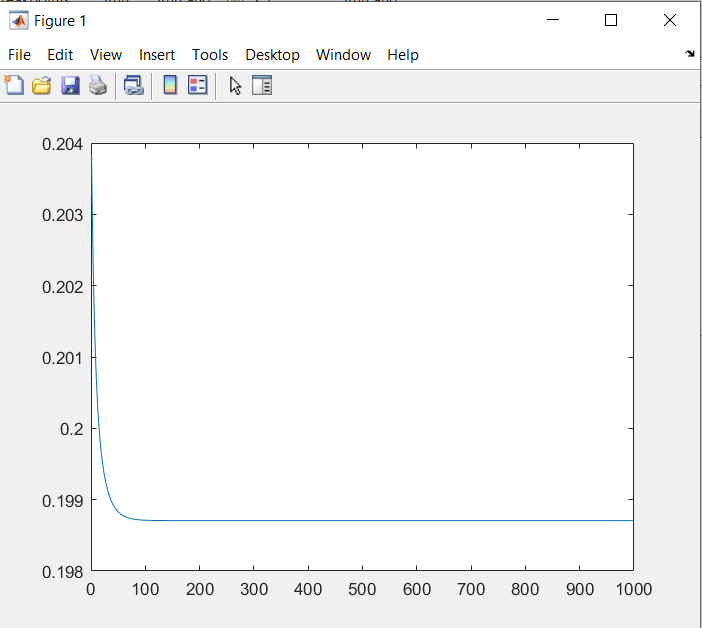
شکل 36) snippet مربوط به AND داخل کد

### تابع فعال سازی sigmoid



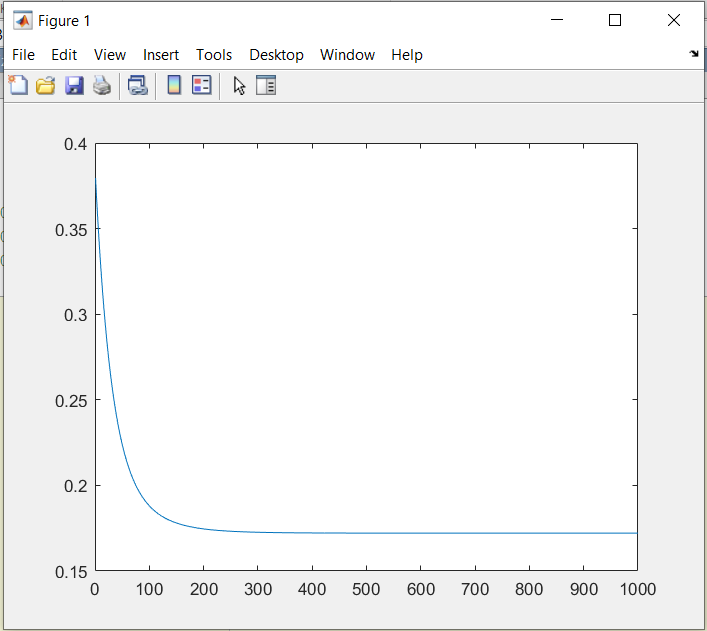
شکل 37) MSE با تابع فعال ساز sigmoid

### تابع فعال سازی Tanh



شکل 38) MSE با تابع فعال ساز Tanh

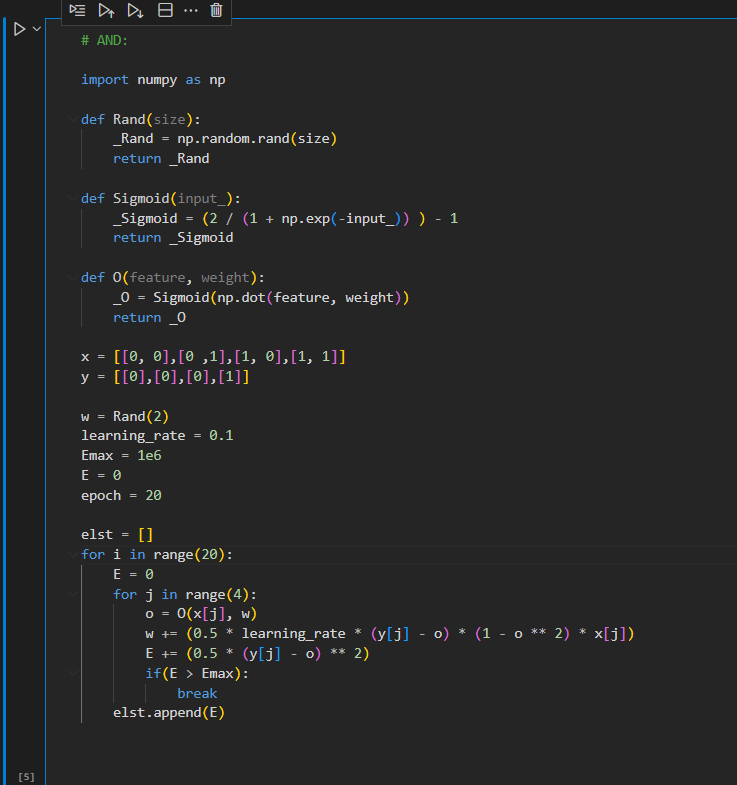
### تابع فعال ساز ReLU



شکل 39) MSE با تابع فعال ساز ReLU

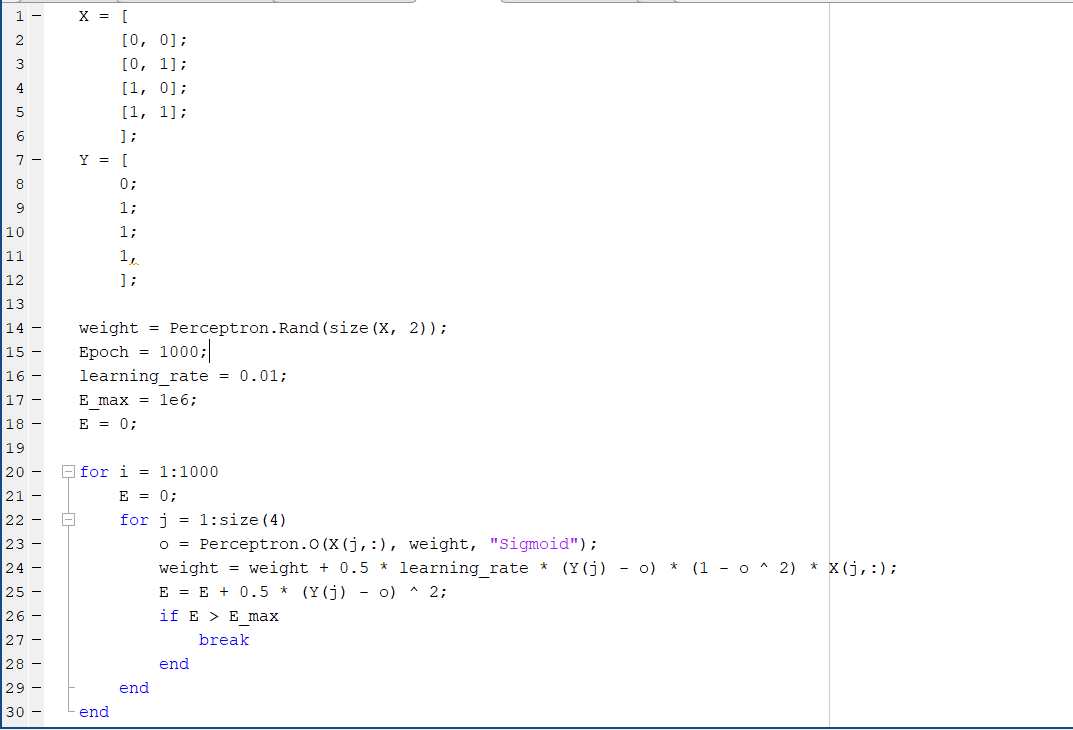
### پیاده سازی در Python

به طور کلی تمرکز بر روی Script داخل MATLAB می‌باشد.



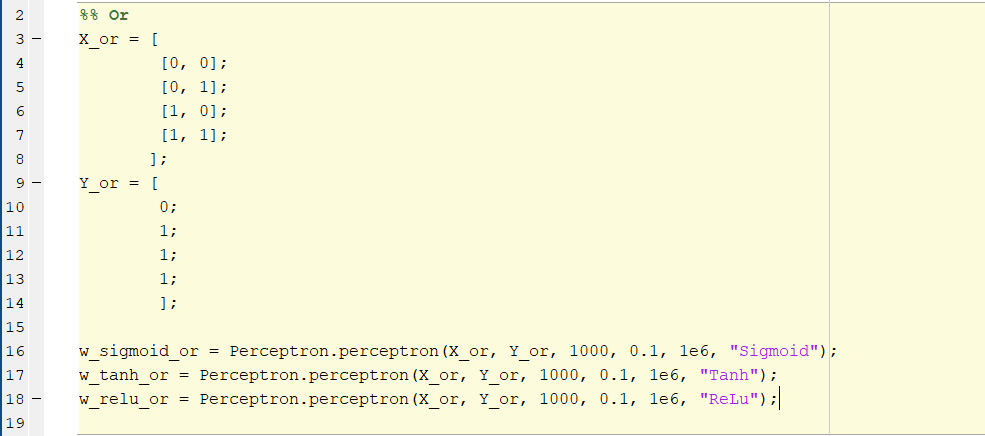
شکل 40) پیاده سازی AND در Python، شایان ذکر است که به علت تکرار مفاهیم، از توضیح بیشتر پرهیز شده است.

* 1. **پیاده سازی Or**



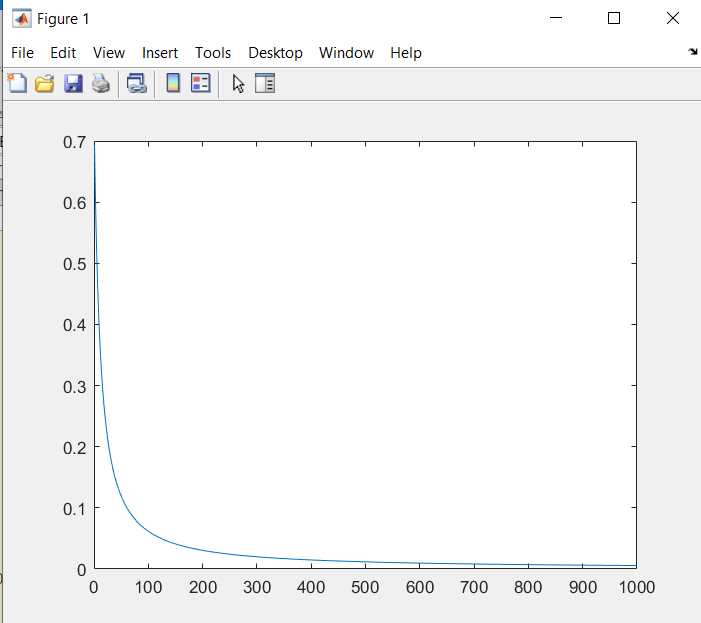
**شکل 42) در فایل “Or.m” نیز Script فوق موجود است و در آن از کلاس Perceptron نیز استفاده شده است.**

همچنین در فایل "Modularized.m" نیز از این فایل ها استفاده شده است.



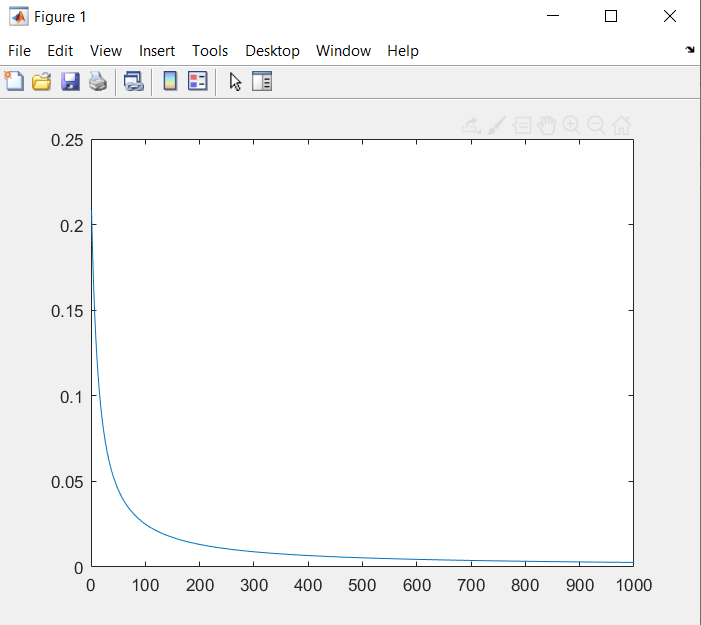
**شکل 43) snippet مربوط به Or داخل کد**

* + 1. **تابع فعال سازی sigmoid**



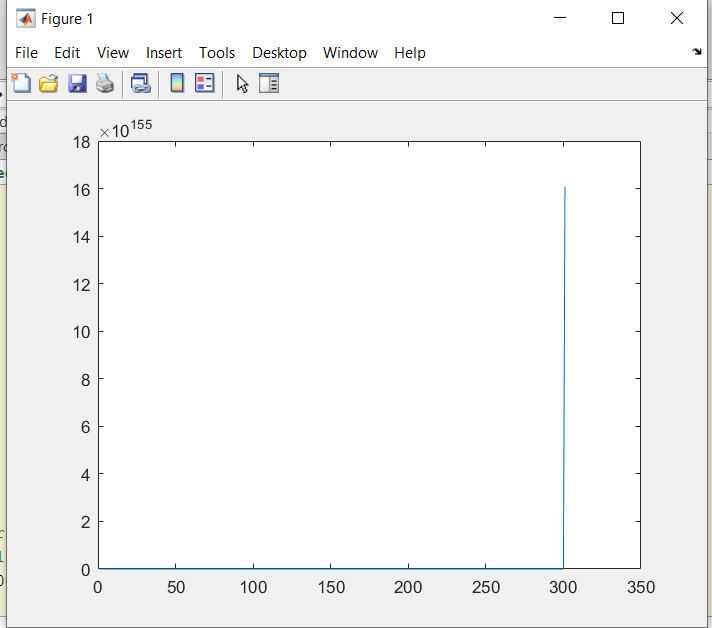
**شکل 44) MSE با تابع فعال ساز sigmoid**

* + 1. **تابع فعال سازی Tanh**



**شکل 45) MSE با تابع فعال ساز Tanh**

* + 1. **تابع فعال ساز ReLU**

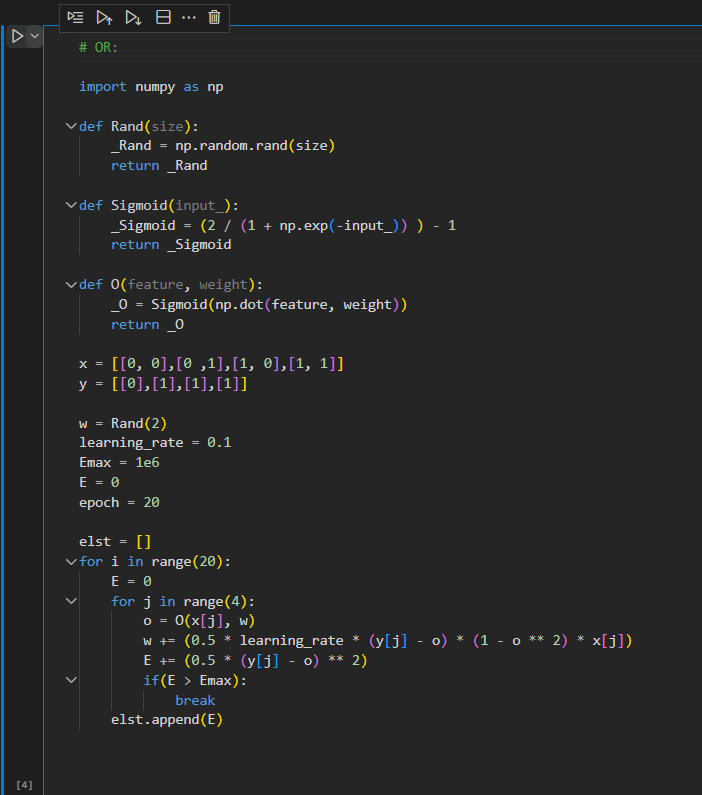


**شکل 46) MSE با تابع فعال ساز ReLU**

با توجه به نمودار بالا تابع ReLU فعال ساز مناسبی برای گیت OR نیست.

### پیاده سازی در Python

به طور کلی تمرکز بر روی Script داخل MATLAB می‌باشد.

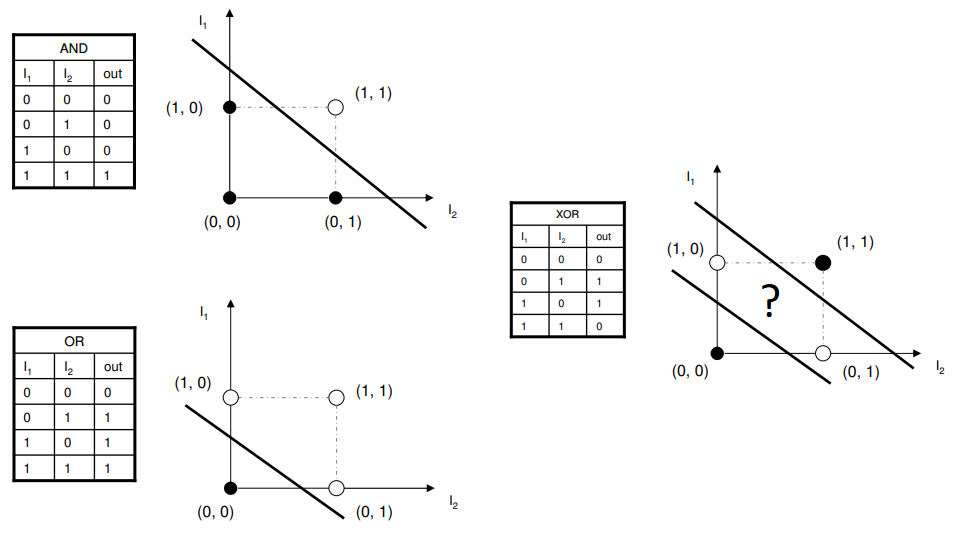


شکل 47) پیاده سازی Or در Python، شایان ذکر است که به علت تکرار مفاهیم، از توضیح بیشتر پرهیز شده است.

## پیاده سازی XOR

در پیاده سازی XOR توسط Perceptron به چالش جدیدی بر می‌خوریم.

همانطور که در شکل زیر قابل مشاهده است، پس از تشکیل جدول درستی هر یک، و در نهایت رسم اعضای داخل ناحیه ها، ملاحظه می‌شود که:

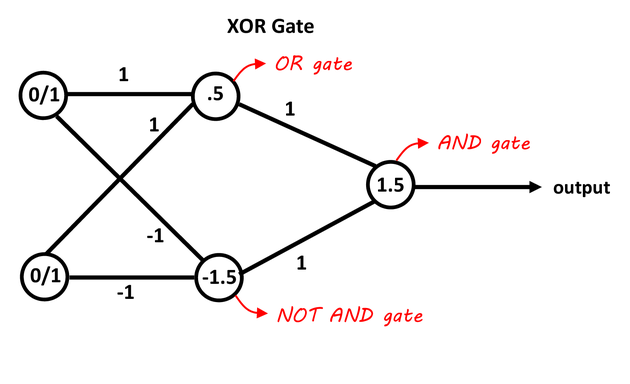


شکل 48) جدول درستی هر یکی از گیت ها و تشکیل نواحی اعضا

بنابراین یک Perceptron منفرد نمی تواند گیت XOR ما را جدا کند زیرا فقط می تواند یک خط مستقیم بکشد.

ترفند این است که متوجه شویم که می توانیم به طور منطقی دو Perceptron را روی هم قرار دهیم. دو Perceptron خطوط مستقیم را ترسیم می کنند و Perceptron دیگری که این دو سیگنال مجزا را در یک سیگنال واحد ترکیب می کند که فقط باید بین یک مرز صحیح / نادرست تفاوت قائل شود.

به طور شهودی تر، ما نمی‌توانیم از یک شبکه تک لایه استفاده کنیم و باید از تعداد لایه های بالاتر استفاده کنیم. همانند تمرین دوم، اگر بخواهيم تنها يك لايه Perceptron را به كار بگيريم، جداسازي داده ها در اين حالت ناممكن است چرا كه يك لايه ي Perceptron امكان جداسازي توابع غيرخطي را ندارد و مدل همگرا نمي شود. اما با به كارگيري چندين لايه، جداسازي اين داده ها به سادگي قابل انجام است.



شکل 49) گیت XOR را می توان با ترکیب زیر از یک گیت NOT AND و یک گیت OR ایجاد کرد

**دیتاست Iris**

ديتاست Iris مخزن UCI شامل داده هاي مربوط به گياهان است و هدف دسته بندي گونه ي گياهي بر اساس داده ها ارائه شده مي باشد. در اين ديتاست اطلاعات ۳گونه ي گياهي موجود است و براي هر نمونه ۴ ويژگي ارائه شده است:

1. Sepal length in cm – طول گلبرگ
2. Sepal width in cm – عرض گلبرگ
3. Petal length in cm – طول کاسبرگ
4. Petal width in cm – عرض کاسبرگ

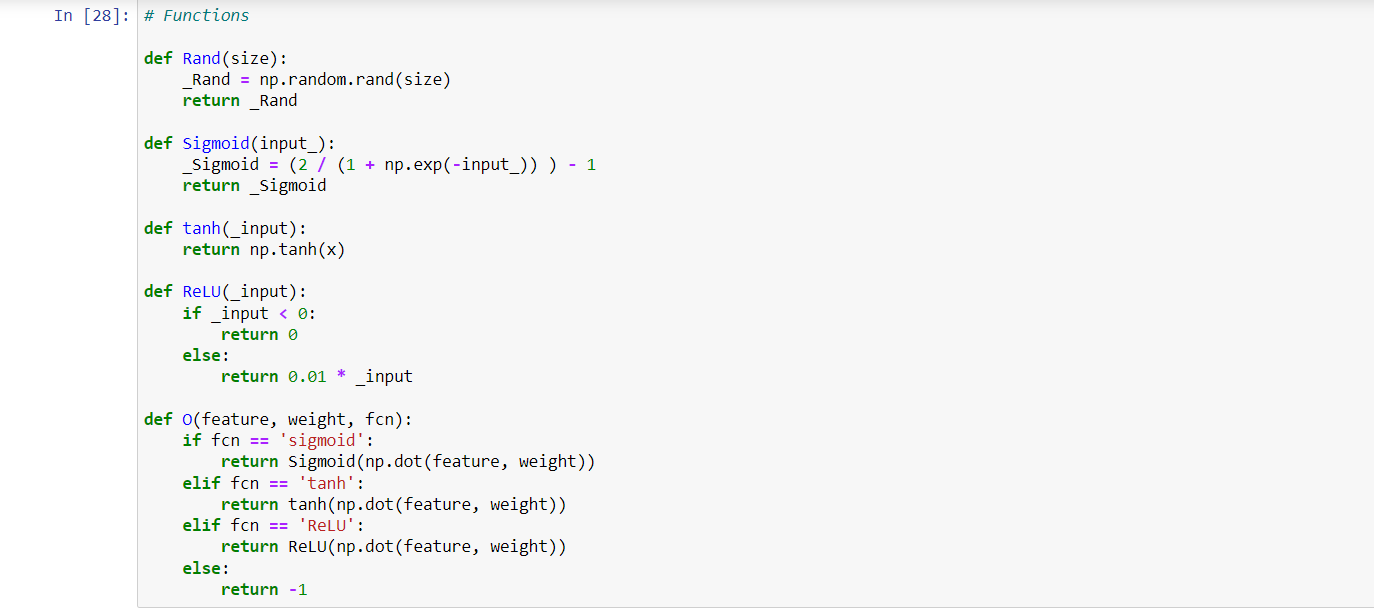
این دیتاست شامل ۳ گونه ی گیاهی است و مانند سوال دوم تمرین نمیتوان آن را با ۱ پرسپترون جداسازی کرد و برای دسته بندی داده ها به ۳ پرسپترون در یک لایه نیاز داریم ( نیازی نیست شبکه ما چند لایه باشد). به همین علت به ۳وکتور برای وزن نیاز داریم که در ادامه آن را توضیح خواهیم داد.

Text

Description automatically generated with medium confidenceاولین نکته که باید ذکر کرد در این دیتاست مانند دیتاست های معمول در سطر اول فایل .csv به نام گذاری ستون ها که همان feature های دیتای ماست نپرداخته و وقتی ما دیتاست را به عنوان ورودی میگیریم سطر اول که خود شامل دیتای ماست را به عنوان نام ستون ها درنظر میگیرد. برای حل این مشکل و از دست ندادن یک دیتا، ما میتوانیم ۲ کار انجام دهیم: راه حل اول به این صورت است که بصورت دستی دیتاست را تغییر دهیم و سطر اول آن را به نام گذاری feature بپردازیم اما چون دست بردن به دیتاست کار معقولی نیست از روش دوم استفاده کردیم. در این روش دیتا سطر اول را بصورت دستی به ابتدای دیتافریممان اضافه می­کنیم که مراحل انجام این روش را در عکس پایین می­بینیم.

**شکل 50) تصویر مربوط به اضافه کردن دیتا سطر اول و گرفتن فیچرها و خروجی**

در ادامه نیز توابع مورد نیاز که همان توابع استفاده شده در سوال مربوط به سرطان سینه و گیت ها می­باشد را اضافه می­کنیم و با توجه به این که قبلا توضیحات مربوط به این توابع را دادیم از تکرار آنها خود داری می­کنیم.

**شکل 51)‌ توابع مورد نیاز نورون ها**

در ادامه به پارامتر ها و نرمال کردن دیتاها می­پردازیم.

**شکل 52) تعیین پارامتر ها و نرمال کردن دیتاست**

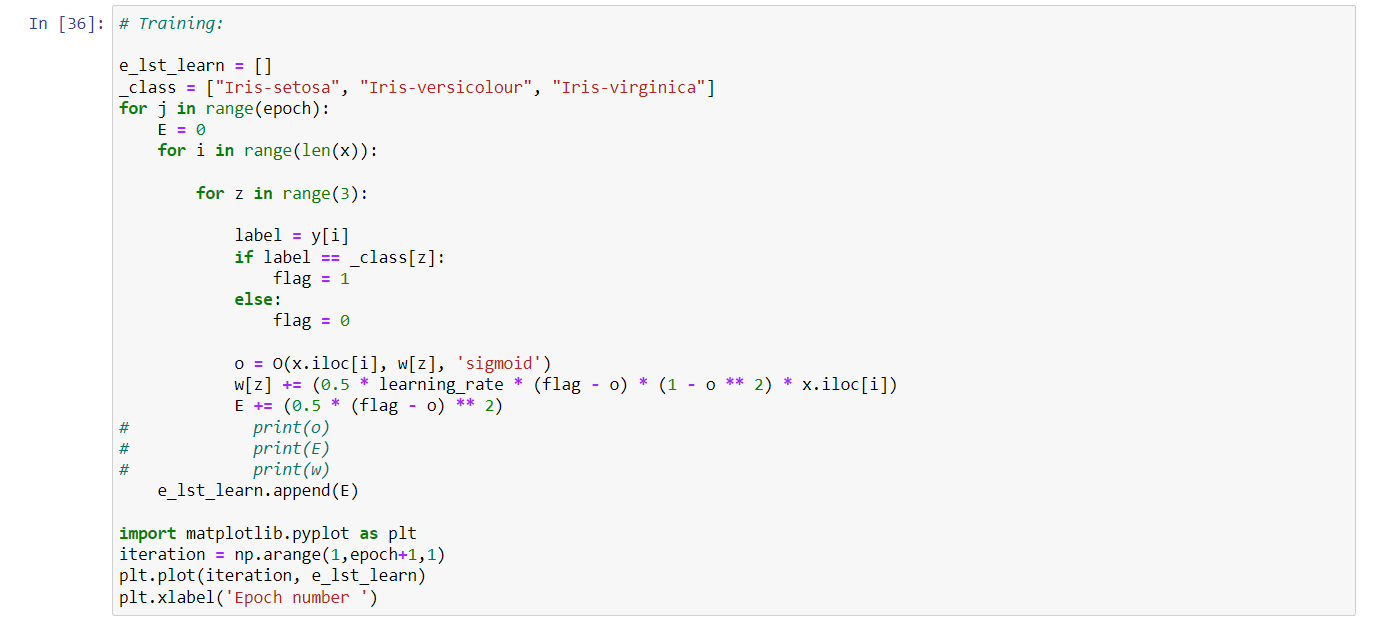
نکته ای که در رابطه به تعیین پارامترها مهم است و قبلا هم اشاره کردیم،‌ در این دیتاست برای آموزش دیتاست به ۳ نورون نیاز داریم که نتیجه میشود به ۳ وکتور وزن نیاز داریم. همانطور که سلول اول تصویر بالا مشخص است ما وزن را در یک لیست میریزن که این لیست یک ماتریس ۵\*۳ است.

۳ سطر برای اینکه ما ۳ نورون داریم و ۵ ستون به این علت که داده های ما دارای ۴ نوع دیتای مختلف است که دربالا آن ها را توضیح دادیم و یک ستون برای بایاس نیز در ادامه به آن اضافه میکنیم پس هر نورون ما نیاز به ۵ وزن دارد که تشکیل یک ماتریس ۵\*۳ می­دهند، در ادامه نیز این لیست را یک npArray تبدیل میکنیم.

برای نرمال کردن داده هایمان نیز مانند گذشته از تابع Processing در کتابخانه sklearn استفاده می­کنیم که مراحل نرمال سازی آن در تصویر بالا مشخص است و در انتها به دیتا های فیچرمان ستون بایاس را نیز اضافه می­کنیم.

برای آموزش شبکه مان نیاز به ۳ حلقه داریم. حلقه اول مربوط به تعداد iteration هایمان است و در هر تکرار حلقه مقدار خطا را باید صفر کنیم. حلقه دوم مربوط به تعداد دیتا های دیتاستمان است که برابر با ۱۵۰ دیتا است، چون تعداد دیتا ها کم است همه آن را برای اموزش استفاده میکنیم، این حلقه همانطور که قبلا توضیح داده شده برای این است که برای هر دیتا از دیتاستمان باید وزن ها و خطا را اپدیت کنیم و حلقه آخر نیز برای نورون ها است که از نورون اول شروع میکنیم دیتا هارا به نورون میدهیم آن را آموزش می­دهیم و خطا را برای آن حساب میکنیم بعد به سراغ نورون بعدی می­رویم و همان کار هارا برای آن تکرار می­کنیم.

نحوه عملکرد حلقه سوم بدین صورت است که ما چک می­کنیم که آیا لیبل آن دسته ای که در آن قرار داریم ( y[i] ) با کلاس نورونی که برای هریک از گونه های گل قرار دادیم برابر است یا خیر و در این صورت متغیر فلگ را برابر ۰ یا ۱ قرار می­دهیم. به عنوان مثال اگر در نورون مربوط به گونه versicolour قرار داشته باشیم و دیتایی که در حال برسی آن هستیم لیبل setosa خورده باشید فلگ برابر ۰ می­شود اما اگر در دیتا های مربوط به همان نوع گل versicolour قرار داشته باشیم فلگ برابر ۱ می­شود. از این فلگ در محاسبه خطا و آپدیت کردن وزن ها استفاده می­شود که در کد مشخص است.

**شکل 53) نحوه آموزش شبکه برای دیتاست Iris**

Chart, line chart

Description automatically generatedدر نهایت هم عملکرد مربوط این شبکه را مشاهده میکنیم که البته هرچه تعداد تکرارمان بیشتر باشد عملکرد مطلوب تر می­شود.

**شکل 54) ارور بر حسب تکرار**

**منابع و مواخذ**

* https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load\_breast\_cancer.html
* https://github.com/scikit-learn/scikit-learn/blob/36958fb240fbe435673a9e3c52e769f01f36bec0/sklearn/datasets/data/breast\_cancer.csv
* https://www.simplilearn.com/tutorials/deep-learning-tutorial/perceptron#:~:text=A%20Perceptron%20is%20a%20neural,value%20%E2%80%9Df(x).
* https://en.wikipedia.org/wiki/Rectifier\_(neural\_networks)
* https://www.geeksforgeeks.org/implementation-of-perceptron-algorithm-for-xor-logic-gate-with-2-bit-binary-input/
* https://flipdazed.github.io/blog/python%20tutorial/introduction-to-neural-networks-in-python-using-XOR