«*In The Name Of GOD*»

[HW-01-Report]

[NEURAL COMPUTING AND DEEP LEARNING]

Hasan Masroor | [403131030] | March 15, 2025

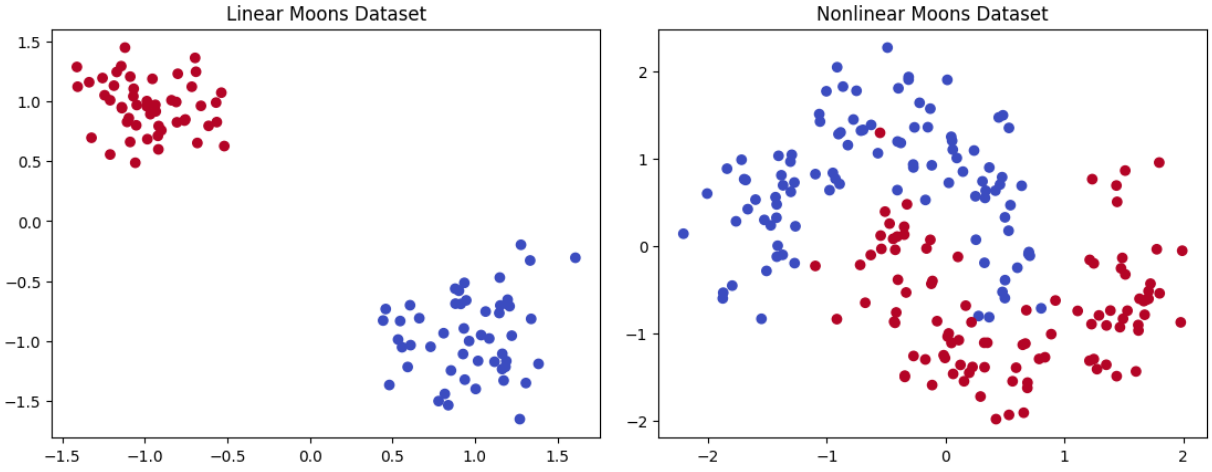
"فهرست مطالب تمرين 01"

Problem 1: Perceptron and Adaline Neural Units

Question 1:

در این سوال ما قصد داریم که دو مدل پرسپترون و آدالین خطی را برای دسته‌بندی داده‌های خطی پیاده‌سازی و روی دیتاست خطی سوال آموزش دهیم و سپس عملکرد هر دو مدل را با استفاده از نمودارهای خطای آموزشی و اعتبارسنجی، ماتریس درهم‌ریختگی و معیارهای ارزیابی مثل دقت و ... مقایسه و بررسی کنیم.

در ابتدا کتابخانه‌های مورد نظر را می‌آوریم و سپس با استفاده از قطعه کد تمرین دیتاست های خطی و غیرخطی یعنی make_blobs و make_moons را آپلود می‌کنیم، داده‌ها را استانداردسازی می‌کنیم و در نهایت هم در خروجی این دو دیتاست را نمایش می‌دهیم:



بعد از اینکه کتابخانه‌های مورد نیاز و کد پیاده‌سازی دیتاست‌ها را انجام دادیم؛ در سوال اول کلا روی دیتاست خطی و داده‌های خطی قرار هست کار کنیم. داده‌ها را به سه بخش آموزشی، اعتبارسنجی و تست تقسیم کردیم و بعد از آن داده‌ها به تنسورهای PyTorch تبدیل شدند تا برای آموزش مدل‌های پرسپترون و آدالین آماده باشند.

بعد از اینکه کارهای اولیه را برای این تمرین انجام دادیم حالا می‌خواهیم مدل **پرسپترون** خطی را برای داده‌های خطی پیاده‌سازی کنیم. این مدل شامل یک لایه خطی و تابع فعال‌سازی sigmoid برای دسته‌بندی باینری یا

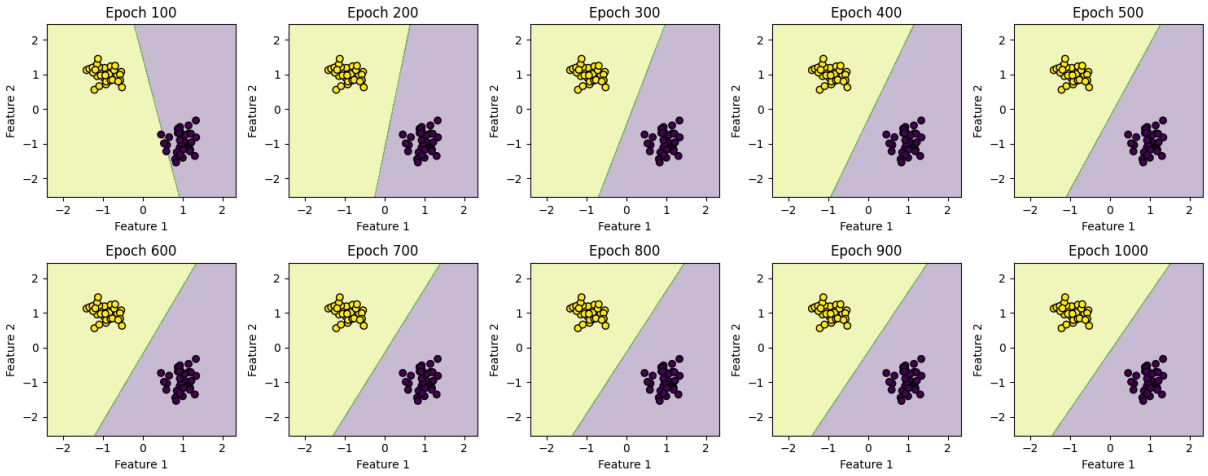
دوتایی است؛ همچنین مدل با استفاده از تابع هزینه BCELoss یا Binary Cross Entropy و بهینه‌ساز SGD آموزش داده شد.

```
Test Accuracy: 100.00%
Test Recall: 100.00%
```

این مدل روی داده‌های خطی به خوبی عمل کرد و توانست به درستی داده ها را با یک خط به دو دسته مختلف دسته بندی کند و دقت این مدل برای این داده‌های خطی 100% شده است و به این معنی است که به درستی تمامی داده‌ها را به دسته‌بندی باینری کرده است.

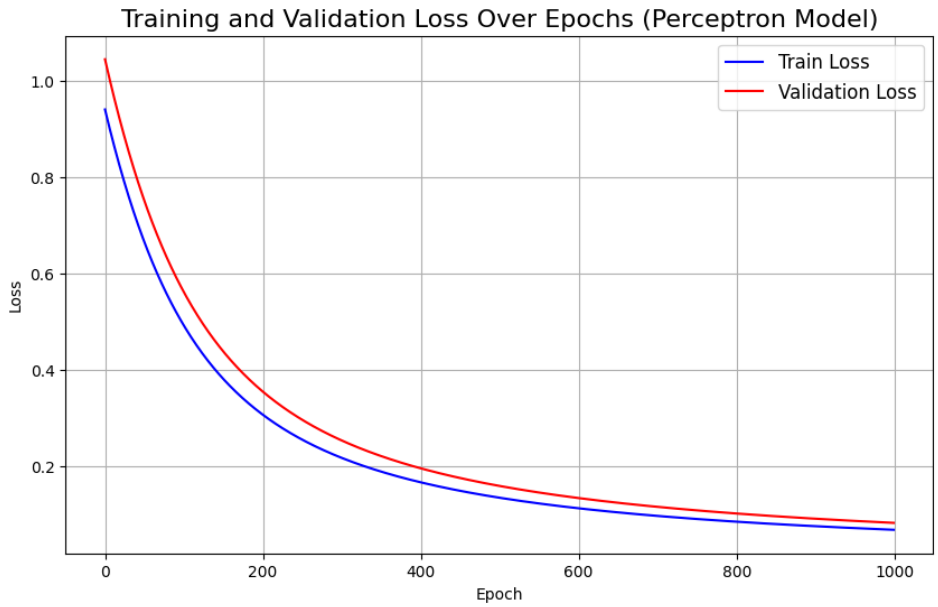
```
Epoch [100/1000], Loss: 0.4949, Validation Loss: 0.5645
Epoch [200/1000], Loss: 0.3085, Validation Loss: 0.3565
Epoch [300/1000], Loss: 0.2184, Validation Loss: 0.2546
Epoch [400/1000], Loss: 0.1675, Validation Loss: 0.1966
Epoch [500/1000], Loss: 0.1353, Validation Loss: 0.1598
Epoch [600/1000], Loss: 0.1134, Validation Loss: 0.1346
Epoch [700/1000], Loss: 0.0975, Validation Loss: 0.1164
Epoch [800/1000], Loss: 0.0855, Validation Loss: 0.1025
Epoch [900/1000], Loss: 0.0761, Validation Loss: 0.0917
Epoch [1000/1000], Loss: 0.0686, Validation Loss: 0.0830
```

همچنین ما روند آموزش این مدل را در 1000 دوره یا epoch نمایش دادیم و مقادیر **Loss** و **Validation Loss** هم در هر 100 دور نشان دادیم. همانطور در تصویر بالا هم مشخص است در ابتدای آموزش یا Epoch 100 خطای آموزشی مدل برابر 0.4949 و خطای اعتبارسنجی هم 0.5645 بوده است که با پیشرفت آموزش هر دو مقدار به طور پیوسته کاهش می‌یابند و در انتهای آموزش یا Epoch 1000، خطای آموزشی به 0.0686 و خطای اعتبارسنجی به 0.0830 رسیده است و نشان‌دهنده این است که مدل به خوبی در حال یادگیری الگوهای داده‌هاست. کاهش مداوم و پایدار هر دو خطا نشان می‌دهد که مدل به سمت همگرایی حرکت کرده است و همچنین اختلاف بین خطای آموزشی و اعتبارسنجی در طول آموزش کم است، که نشان می‌دهد مدل دچار overfittingنشده است. به طور کلی مدل پرسپترون در اینجا عملکرد خوبی داشته و توانسته‌است با کاهش خطا و همگرایی مناسب، الگوهای داده‌ها را یاد بگیرد.



برای درک بهتر علاوه بر تصویر قبل که خطا را در هر 100 دور نشان می‌داد، در تصویر بالا هم به صورت بصری نمایش دادیم که در ابتدا شاید مرز تصمیم‌گیری زیاد بهینه نباشد اما به مرور شروع به بهبود کرده و بهتر داده‌ها را جدا کرده و دذ نهایت مرز تصمیم بهینه شده و به طور درست دو دسته داده‌ها را دسته‌بندی کرده و نشان می‌دهد که مدل به خوبی آموزش دیده است.

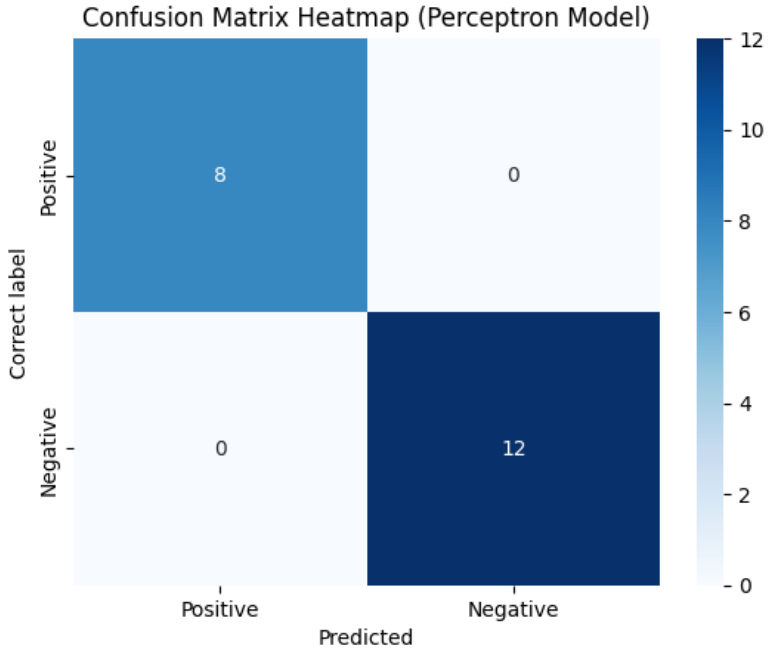
در ادامه برای اینکه بهتر بتوانیم این خطای آموزش و اعتبارسنجی را تجزیه و تحلیل کنیم و همچنین همگرایی در این مدل را بررسی کنیم نمودار مربوطه را رسم می‌کنیم:



همان‌طور که از تصویر هم مشخص است به مرور و با افزایش Epoch مقدار خطا یا Loss هم برای آموزشی و هم اعتبارسنجی کاهش پیدا می‌کند و نمودار شیب نزولی دارد و یعنی به مرور خطا کمتر می‌شود و مدل یاد می‌گیرد که بهتر عمل کن و دسته‌بندی درست و بهتری برای این دسته‌بندی باینری داشته باشد. از روی نمودار می‌بینیم که

هنوز مدل همگرا نشده است و احتمالا بعد از 1000 epoch اگر چند epoch دیگر ادامه دهیم در نقطه ای مقدار خطا تقریبا ثابت می‌شود و دیگر کاهش محسوسی ندارد و می‌توانیم روند را متوقف کنیم و دیگر بیشتر ادامه ندهیم.

حالا می‌آییم مدل آموزش‌دیده را روی داده‌های تست ارزیابی می‌کنیم و سپس ماتریس درهم‌ریختگی (CONFUSION MATRIX) را محاسبه و به صورت یک هیت‌مپ نمایش داده می‌دهیم (این ماتریس نشان می‌دهد که مدل چقدر در پیش‌بینی کلاس‌های مثبت و منفی درست عمل کرده است):



از قبل با accuracy، precision، recall و ... آشنا هستیم:

- $Accuracy = \frac{\#TP+\#TN}{\#P+\#N} = \frac{\#TP+\#TN}{\#TP+\#FN+\#TN+\#FP}$
- $Precision = \frac{TP}{TP+FP}$ و $TPR = \frac{TP}{TP+FN} = Sensitivity = Recall$

طبق این فرمول‌ها از روی ماتریس بخواهیم همان دقت و recall که اسلاید قبل مدل نمایش داده بود را بدست بیاوریم دقیقا به همان یعنی 100% برای هردو می‌رسیم.

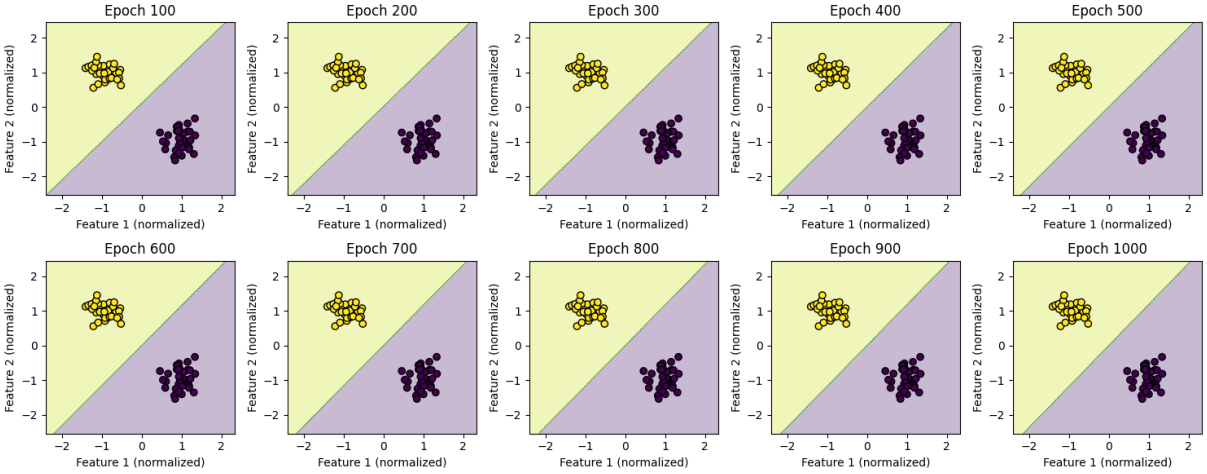
در این بخش، مدل **آدالین** را برای دسته‌بندی داده‌های خطی پیاده‌سازی کردیم. این مدل شامل یک لایه خطی بدون تابع فعال‌سازی است و از تابع هزینه **MSE (Mean Squared Error)** برای آموزش استفاده می‌کند. مدل با بهینه‌ساز **SGD** آموزش داده شد.

```
Test Accuracy: 100.00%
Test Recall: 100.00%
```

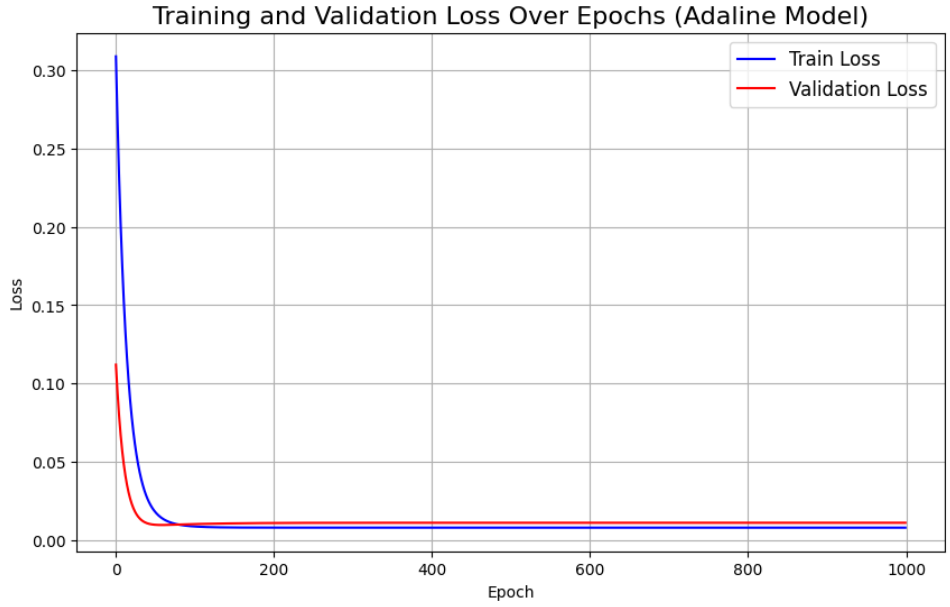
این مدل روی داده‌های خطی به خوبی عمل کرد و توانست به درستی داده ها را با یک خط به دو دسته مختلف دسته بندی کند و دقت این مدل برای این داده‌های خطی 100% شده است و به این معنی است که به درستی تمامی داده‌ها را به دسته‌بندی باینری کرده است.

```
Epoch [100/1000], Loss: 0.0088, Validation Loss: 0.0103
Epoch [200/1000], Loss: 0.0081, Validation Loss: 0.0110
Epoch [300/1000], Loss: 0.0081, Validation Loss: 0.0112
Epoch [400/1000], Loss: 0.0081, Validation Loss: 0.0112
Epoch [500/1000], Loss: 0.0081, Validation Loss: 0.0112
Epoch [600/1000], Loss: 0.0081, Validation Loss: 0.0112
Epoch [700/1000], Loss: 0.0081, Validation Loss: 0.0112
Epoch [800/1000], Loss: 0.0081, Validation Loss: 0.0112
Epoch [900/1000], Loss: 0.0081, Validation Loss: 0.0112
Epoch [1000/1000], Loss: 0.0081, Validation Loss: 0.0112
```

ما روند آموزش مدل آدالاین را در 1000 دوره (epoch) نمایش دادیم و مقادیر **Loss** و **Validation Loss** را در هر 100 دوره بررسی کردیم. در ابتدای آموزش (Epoch 100)، خطای آموزشی مدل برابر با 0.0088 و خطای اعتبارسنجی 0.0103 بود. با پیشرفت آموزش هر دو مقدار به سرعت کاهش یافتند و از دوره 300 به بعد تقریباً ثابت شدند. در انتهای آموزش(Epoch 1000) نیز خطای آموزشی به 0.0081 و خطای اعتبارسنجی به 0.0112 رسید و این کاهش سریع نشان می‌دهد که مدل به خوبی همگرا شده است و الگوهای داده‌ها را به طور موثر یاد گرفته است؛ همچنین اختلاف کم بین خطای آموزشی و اعتبارسنجی نشان می‌دهد که مدل دچار overfitting نشده است و به طور کلی مدل آدالاین عملکرد بهتری نسبت به پرسپترون نشان داد و با سرعت بیشتری به همگرایی رسید.

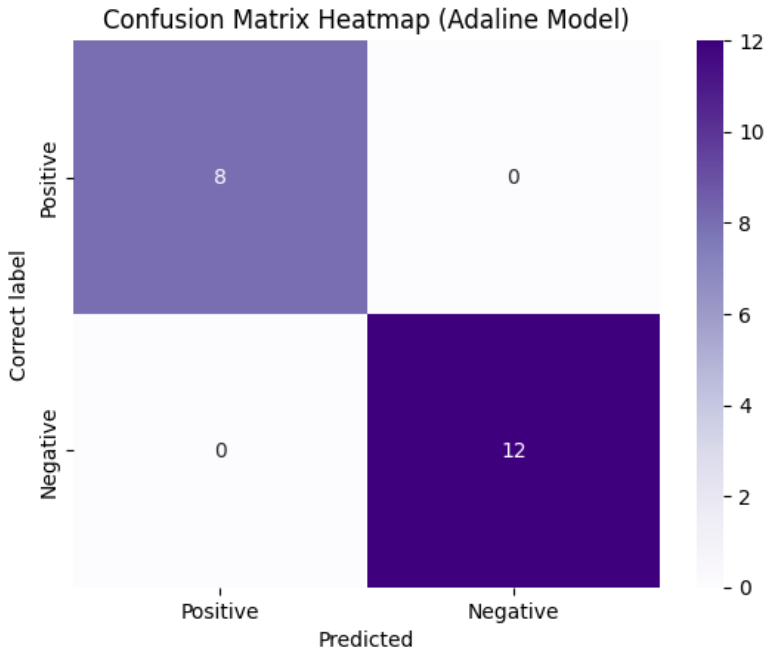


همان‌طور که مقادیر خطاها در بخش قبل را دیدیم و این تصویر نشان می‌دهند تغییرات مرز تصمیم در هر 100 Epoch خیلی کم و در حد چند ده‌هزارم است از همان دور اول آموزش هم دسته‌بندی درست و خوبی انجام داده است و در دورهای بعدی این خطا را 0.0008 کاهش داده است و برخلاف پرسپترون خیلی سریع به همگرایی رسیده است و این موضوع را در ادامه و نمودار خطاها تجزیه و تحلیل خواهیم کرد.



همان‌طور که از تصویر هم مشخص است به مرور و با افزایش Epoch مقدار خطا یا Loss هم برای آموزشی و هم اعتبارسنجی کاهش پیدا می‌کند و نمودار نزولی است و با شیب زیادی به سرعت کاهش پیدا می‌کند و مدل با سرعت بیشتری یاد می‌گیرد که بهتر عمل کند و دسته‌بندی درست و بهتری برای این دسته‌بندی باینری داشته

باشد. از روی نمودار می‌بینیم که حدودای 100 Epoch تغییرات مدل تقریبا ثابت می‌شود و مدل با سرعت بیشتری نسبت به پرسپترون همگرا می‌شود.



طبق فرمول‌های چند اسلاید قبل‌تر از روی ماتریس بخواهیم همان دقت و recall که اسلاید قبلی مدل نمایش داده بود را بدست بیاوریم دقیقا به همان یعنی 100% برای هردو می‌رسیم.

1.

نرخ یادگیری (Learning Rate) یکی از مهم‌ترین پارامترهای آموزش مدل‌های یادگیری ماشین است. این پارامتر تعیین می‌کند که مدل در هر گام به‌روزرسانی چقدر از گرادیان (gradient) استفاده کند تا وزن‌ها را به‌روز کند.

انتخاب درست نرخ یادگیری اهمیت زیادی دارد و مثلا اگر با مقادیر خیلی بالا شروع کنیم ممکن است از آن نقطه مینیمم سراسری بپرد و به همین صورت جهش‌های غیرمفید انجام دهد و ممکن است در مینیمم محلی گیر بیافتد و نتواند به نقطه بهینه برسد (به این صورت متوجه این موضوع می‌شویم که مثلا خطای آموزش و اعتبارسنجی به شدت نوسان می‌کند یا حتی افزایش می‌یابد و یا اینکه مدل ممکن است اصلاً همگرا نشود)؛ در نتیجه عملکرد مدل ضعیف می‌شود و پیش‌بینی‌ها نادرست خواهند بود.

از طرف دیگر اگر با مقادیر خیلی کوچک شروع کنیم و سرعت مدل خیلی کم می‌شود و به آرامی به روز می‌شود و باز هم امکان دارد به نقطه بهینه برسیم ولی زمان زیادی برای رسیدن به آن نیاز داریم (به این صورت متوجه این موضوع می‌شویم که خطای آموزش و اعتبارسنجی به کندی کاهش می‌یابد و یا مدل ممکن است در یک حداقل محلی گیر کند و نتواند به نقطه بهینه جهانی برسد)؛ در نتیجه آموزش مدل زمان‌بر می‌شود و ممکن است عملکرد مدل بهینه نباشد.

اگر نرخ یادگیری به درستی تنظیم شود، مدل به طور پایدار و کارآمد به نقطه بهینه همگرا می‌شود و علائم آن این است که خطای آموزش و اعتبارسنجی به طور پیوسته و بدون نوسان کاهش می‌یابد و یا مدل در زمان معقولی به همگرایی می‌رسد؛ در نتیجه عملکرد مدل بهینه می‌شود و پیش‌بینی‌ها دقیق‌تر خواهند بود.

پس به طور کلی می‌توانیم بگوییم برای نرخ یادگیری داریم:

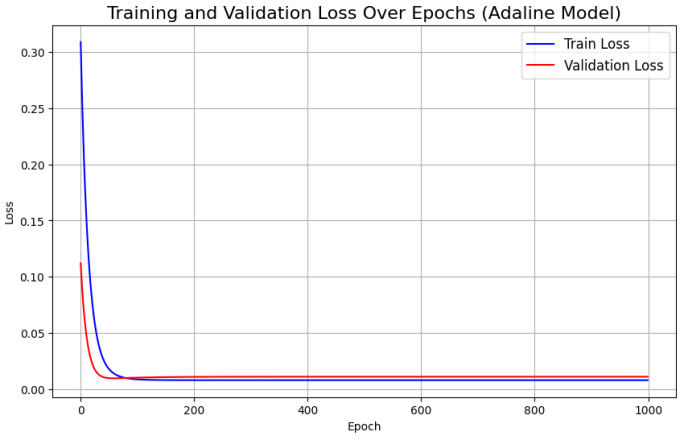
- نرخ یادگیری بزرگ :باعث ناپایداری و عدم همگرایی مدل می‌شود.
- نرخ یادگیری کوچک :باعث کندی آموزش و احتمال گیر کردن در مینیمم‌های محلی می‌شود.
- نرخ یادگیری مناسب :تعادل بین سرعت و پایداری را ایجاد می‌کند و عملکرد مدل را بهینه می‌کند.

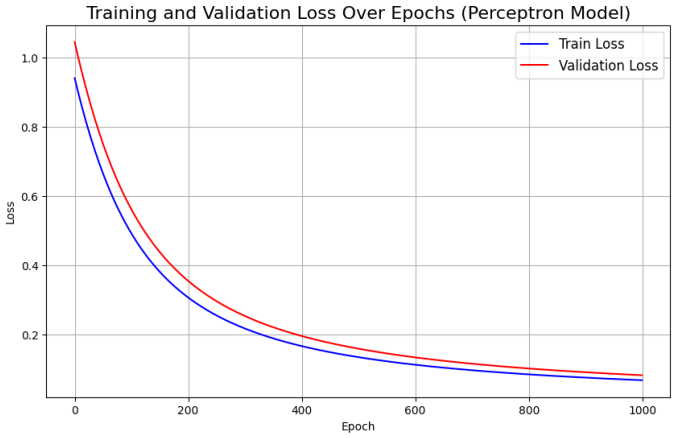
پس افزایش نرخ یادگیری ممکن است باعث شود که وزن‌های مدل به سرعت تغییر کند و می‌تواند باعث جهش‌های بزرگ، نوسانات در تابع هزینه، عدم همگرایی و در نتیجه کاهش دقت مدل شود؛ از آن طرف هم کاهش نرخ یادگیری باعث می‌شود تغییرا در وزن‌ها به آرامی انجام شود و می‌تواند سبب همگرایی کندتر و آهسته‌تر، افزایش دقت و ... شود.

از سوی دیگر اگر نرخ یادگیری ثابت باشد ممکن است مدل خیلی سریع یا خیلی کند همگرا شود؛ بنابراین تنظیم درست نرخ یادگیری اهمیت به‌سزایی دارد و یکی از روش‌های رایج این است که نرخ یادگیری در طول زمان کاهش یابد، در ابتدا نرخ یادگیری بالا است تا مدل سریع‌تر به سمت مینیمم حرکت کند و سپس به تدریج کاهش می‌یابد تا در نهایت مدل در حداقل محلی یا جهانی دقیق‌تر همگرا شود.

2.

سرعت همگرایی به این معناست که مدل چقدر سریع به یک حالت بهینه (جایی که خطای آموزش و اعتبارسنجی تقریباً ثابت می‌شوند) می‌رسد. هرچه مدل در دوره‌های کم‌تری به این حالت برسد، سرعت همگرایی آن بیشتر است. به این سوال قبلا و در بخش تحلیل نمودارهای خطای مدل پرسپترون و آدالین را کامل و دقیق بررسی کردیم و مجدد نمودار خطای این دو مدل را در ادامه می‌آوریم:





از روی نمودارها هم کاملا مشخص است سرعت همگرایی تصویر چپی که برای مدل آدالین هست نسبت به مدل پرسپترون بیشتر است. آدالین به دلیل استفاده از تابع هزینه MSE و بهینه‌سازی مستقیم خطا، سرعت همگرایی بسیار بالاتری نسبت به پرسپترون دارد و به طور کلی آدالین برای مسائل خطی گزینه بهتری از نظر سرعت همگرایی است.

Question 2:

.1

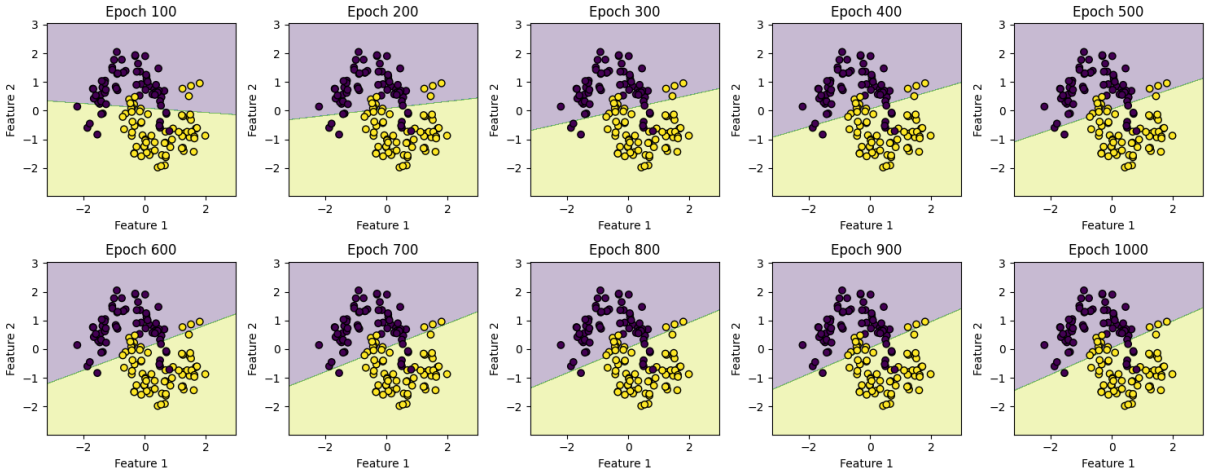
در بخش دوم قرار هست روی دیتاست غیرخطی و داده‌های غیرخطی کار کنیم. مجدد داده‌ها را به سه بخش آموزشی، اعتبارسنجی و تست تقسیم کردیم و بعد از آن داده‌ها به تنسورهای PyTorch تبدیل شدند تا برای آموزش مدل‌های پرسپترون و آدالین آماده باشند. حالا می‌خواهیم مدل **پرسپترون** خطی را برای داده‌های غیرخطی پیاده‌سازی کنیم. این مدل شامل یک لایه خطی و تابع فعال‌سازی sigmoid برای دسته‌بندی باینری یا دوتایی است؛ همچنین مدل با استفاده از تابع هزینه BCELoss یا Binary Cross Entropy و بهینه‌ساز SGD آموزش داده شد.

```
Test Accuracy: 85.00%
Test Recall: 77.78%
```

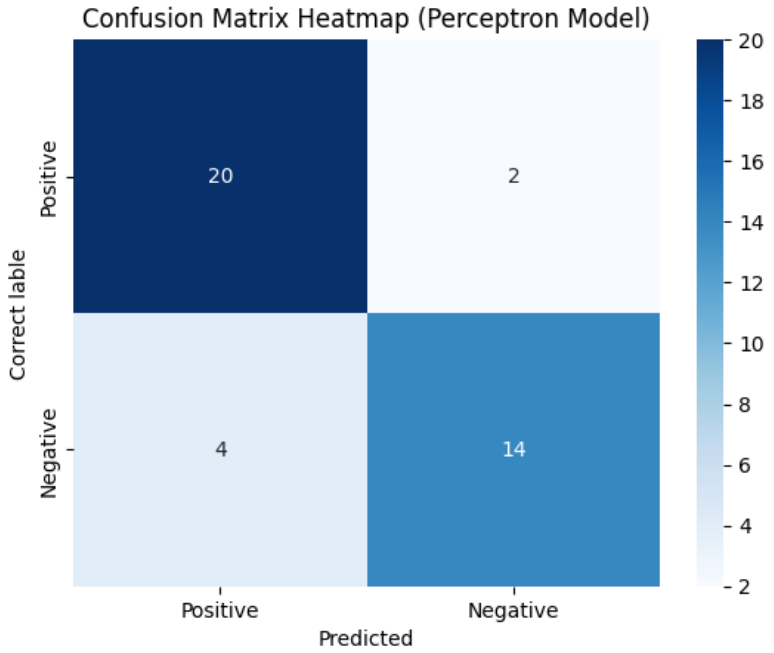
وقتی این بار پرسپترون را روی داده‌های غیرخطی اعمال می‌کنیم می‌بینیم که دقت نسبت به سری قبل کاهش پیدا پیدا کرد و این به خاطر این است که مدل‌های پرسپترون خطی و آدالین خطی نمی‌توانند زیاد خوب روی داده‌های غیرخطی عمل کنند.

```
Epoch [100/1000], Loss: 0.4890, Validation Loss: 0.5224
Epoch [200/1000], Loss: 0.4279, Validation Loss: 0.4765
Epoch [300/1000], Loss: 0.3905, Validation Loss: 0.4518
Epoch [400/1000], Loss: 0.3661, Validation Loss: 0.4381
Epoch [500/1000], Loss: 0.3493, Validation Loss: 0.4305
Epoch [600/1000], Loss: 0.3372, Validation Loss: 0.4265
Epoch [700/1000], Loss: 0.3282, Validation Loss: 0.4246
Epoch [800/1000], Loss: 0.3213, Validation Loss: 0.4241
Epoch [900/1000], Loss: 0.3159, Validation Loss: 0.4245
Epoch [1000/1000], Loss: 0.3116, Validation Loss: 0.4254
```

روند آموزش این مدل را در 1000 دوره یا epoch نمایش دادیم و مقادیر **Loss** و **Validation Loss** هم در هر 100 دور نشان دادیم. همانطور در تصویر بالا هم مشخص است در ابتدای آموزش یا Epoch 100 خطای آموزشی مدل برابر 0.4948 و خطای اعتبارسنجی هم 0.5224 بوده است که با پیشرفت آموزش هر دو مقدار به طور پیوسته کاهش می‌یابند و در انتهای آموزش یا Epoch 1000، خطای آموزشی به 0.3116 و خطای اعتبارسنجی به 0.4254 رسیده است و نشان‌دهنده این است که مدل به خوبی در حال یادگیری الگوهای داده‌هاست. اما در نهایت نتوانسته که به درستی دو دسته را دسته‌بندی کند و مدل پرسپترون برای داده‌های غیرخطی عملکرد مناسبی ندارد؛ زیرا این مدل تنها قادر به یادگیری مرزهای تصمیم‌گیری خطی است. برای بهبود عملکرد، می‌توان از مدل‌های پیچیده‌تر مثل شبکه‌های عصبی چندلایه یا SVM با کرنل غیرخطی یا ... استفاده کرد.



این تصویر هم به خوبی گویای توضیحات میزان خطاها در epoch های مختلف است و اینکه در نهایت این مدل پرسپترون خطی نتوانسته داده‌های غیرخطی را به درستی دسته‌بندی کند.



ماتریس در هم ریختگی را هم روی داده‌های تست اعمال و نمایش دادیم و در اینجا هم می‌توانیم مجدد از فرمول‌های گفته استفاده کنیم و مثلا برای دقت مجوع درست ها یعنی tp و tn را جمع کنیم که همان دو مربع آبی رنگ است و 34 می‌شود و بعد تقسیم بر محموع داده‌های این ماتریس که 40 است می‌کنیم و به همان دقت 85% قبلی خواهیم رسید.

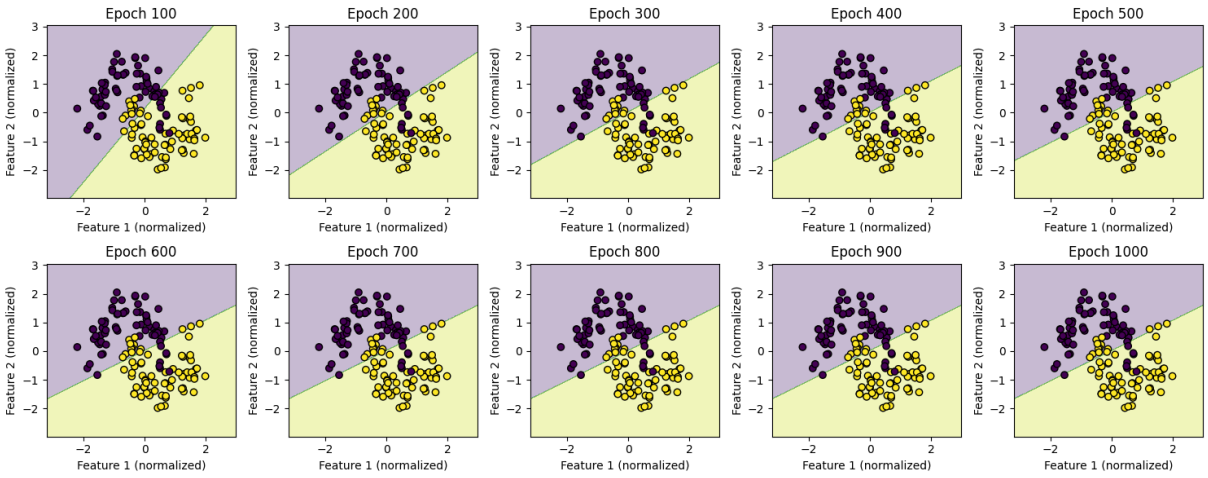
در این بخش هم مدل **آدالین** را برای دسته‌بندی داده‌های غیرخطی پیاده‌سازی کردیم. این مدل شامل یک لایه خطی بدون تابع فعال‌سازی است و از تابع هزینه **MSE (Mean Squared Error)** برای آموزش استفاده می‌کند. مدل با بهینه‌ساز **SGD** آموزش داده شد.

```
Test Accuracy: 82.50%
Test Recall: 72.22%
```

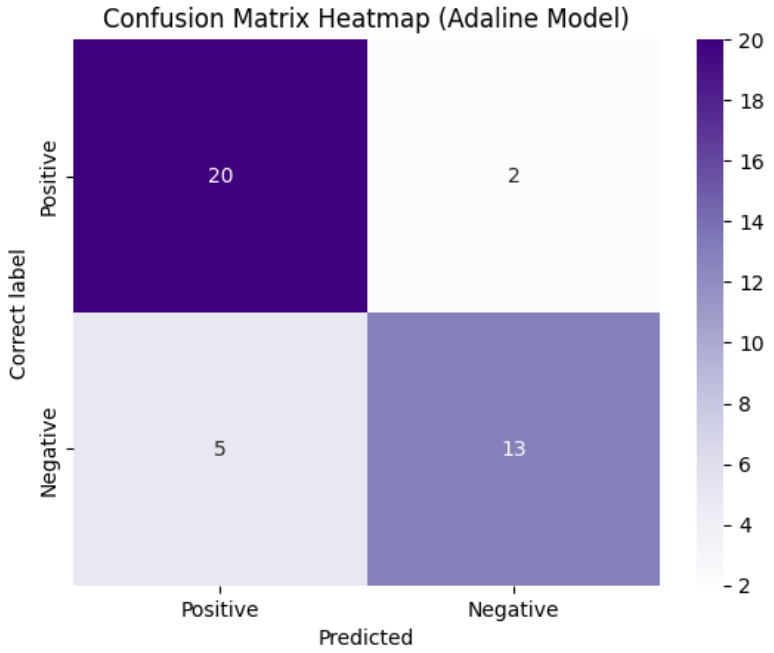
اینجا هم که آدالین را روی داده‌های غیرخطی اعمال می‌کنیم می‌بینیم که دقت نسبت به سری قبل کاهش پیدا پیدا کرد و این به خاطر این است که مدل‌های پرسپترون خطی و آدالین خطی نمی‌توانند زیاد خوب روی داده‌های غیرخطی عمل کنند.

```
Epoch [100/1000], Loss: 0.1102, Validation Loss: 0.1467
Epoch [200/1000], Loss: 0.0977, Validation Loss: 0.1370
Epoch [300/1000], Loss: 0.0967, Validation Loss: 0.1352
Epoch [400/1000], Loss: 0.0966, Validation Loss: 0.1347
Epoch [500/1000], Loss: 0.0966, Validation Loss: 0.1346
Epoch [600/1000], Loss: 0.0966, Validation Loss: 0.1346
Epoch [700/1000], Loss: 0.0966, Validation Loss: 0.1346
Epoch [800/1000], Loss: 0.0966, Validation Loss: 0.1346
Epoch [900/1000], Loss: 0.0966, Validation Loss: 0.1346
Epoch [1000/1000], Loss: 0.0966, Validation Loss: 0.1346
```

روند آموزش این مدل را در 1000 دوره یا epoch نمایش دادیم و مقادیر **Loss** و **Validation Loss** هم در هر 100 دور نشان دادیم. همانطور در تصویر بالا هم مشخص است در ابتدای آموزش یا Epoch 100 خطای آموزشی مدل برابر 0.1102 و خطای اعتبارسنجی هم 0.1467 بوده است که با پیشرفت آموزش هر دو مقدار به طور پیوسته کاهش می‌یابند و در انتهای آموزش یا Epoch 1000، خطای آموزشی به 0.0966 و خطای اعتبارسنجی به 0.1346 رسیده است و نشان‌دهنده این است که مدل به خوبی در حال یادگیری الگوهای داده‌هاست؛ اما این تغییرات و کاهش خطا آنچنان زیاد نبوده و در نهایت نتوانسته که به درستی دو دسته را دسته‌بندی کند. همچنین سرعت همگرایی خوبی داشته علی‌رغم نتیجه نهایی دلخواهی که می‌خواستیم و در Epoch 400 به همگرایی رسیده و مقدارش از اینجا به بعد تقریبا ثابت مانده است.



این تصویر هم به خوبی گویای توضیحات میزان خطاها در epoch های مختلف است و اینکه در نهایت این مدل آدالین خطی نتوانسته داده‌های غیرخطی را به درستی دسته‌بندی کند.



ماتریس در هم ریختگی را هم روی داده‌های تست اعمال و نمایش دادیم و در اینجا هم می‌توانیم مجدد از فرمول‌های گفته استفاده کنیم و اینکه دقت آدلاین در داده‌های غیرخطی کم شده و نسبت به پرسپترون که به 85% رسیده بود این مدل به 82.5% رسده است و به طور کلی این دو مدل نمی‌توانند روی داده‌های غیرخطی زیاد خوب عمل کنند و باید دنبال راهکارهای دیگری برای رسیدن به نتیجه دلخواه باشیم.

.2

برای بهبود عملکرد در دسته‌بندی داده‌های غیرخطی راهکارهای متعددی وجود دارد که می‌توانند به مدل‌های خطی مانند پرسپترون و آدالاین کمک کنند تا روابط پیچیده‌تر را یاد بگیرند. یکی از اولین راهکارها، افزودن ویژگی‌های غیرخطی به داده‌هاست. مدل‌های خطی معمولاً نمی‌توانند روابط پیچیده را یاد بگیرند، اما با اضافه کردن ویژگی‌های درجه بالاتر (مانند توان‌ها یا ترکیبات غیرخطی ویژگی‌ها) می‌توان داده‌ها را به شکلی تبدیل کرد که در فضای جدید به صورت خطی قابل تفکیک باشند. روش دیگر در همین راستا استفاده از ترفند کرنل (Kernel Trick) در ماشین بردار پشتیبان (SVM) است؛ این روش داده‌ها را به فضایی با ابعاد بالاتر منتقل می‌کند تا در آن فضا به راحتی تفکیک‌پذیر شوند و کرنل‌هایی مانند RBF یا Poly می‌توانند به مدل کمک کنند تا داده‌های غیرخطی را در یک فضای مناسب متمایز کند.

راهکار بعدی، استفاده از مدل‌های پیچیده‌تر مانند شبکه‌های عصبی چندلایه (MLP) ، درخت تصمیم و جنگل تصادفی است. شبکه‌های عصبی چندلایه با داشتن لایه‌های پنهان و استفاده از توابع فعال‌سازی غیرخطی

مثل ReLU، توانایی یادگیری روابط پیچیده را دارند. درخت تصمیم و جنگل تصادفی نیز به دلیل ساختار سلسله‌مراتبی خود، بدون نیاز به مهندسی ویژگی خاصی می‌توانند به‌خوبی داده‌های غیرخطی را دسته‌بندی کنند. این مدل‌ها به دلیل انعطاف‌پذیری بالا، گزینه‌های مناسبی برای داده‌های غیرخطی هستند. ا

گر مدل همچنان عملکرد مناسبی نداشت، ممکن است دلیل آن ظرفیت ناکافی شبکه باشد. در این صورت می‌توان با افزایش تعداد نورون‌ها در هر لایه یا افزودن لایه‌های پنهان بیشتر، قابلیت یادگیری مدل را بهبود داد. همچنین، انتخاب تابع فعال‌سازی مناسب بسیار مهم است. توابعی مانند ReLU و Leaky ReLU معمولاً عملکرد بهتری نسبت به سیگموید و تانژانت هیپربولیک دارند، زیرا مشکل اشباع‌شدن گرادیان‌ها را کاهش می‌دهند.

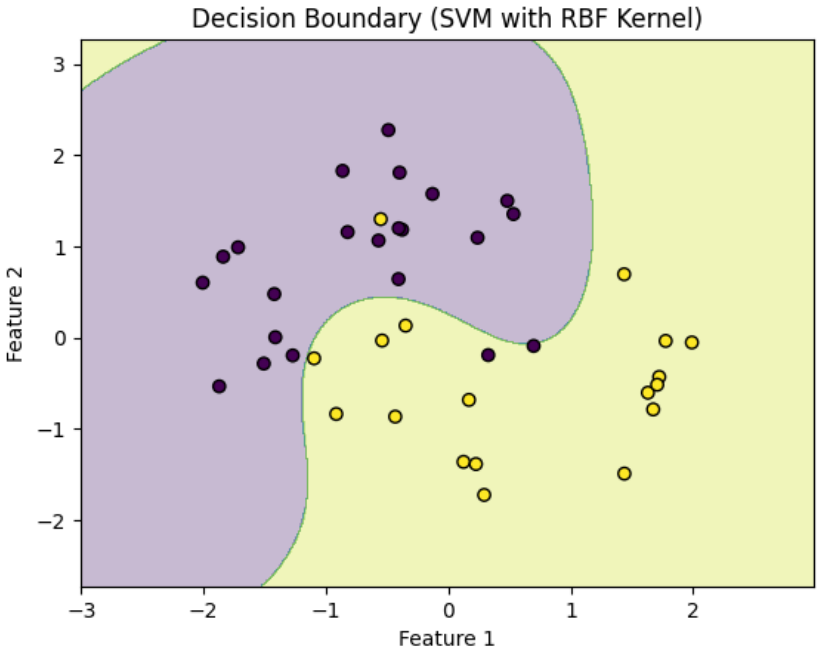
عامل دیگری که می‌تواند تأثیر زیادی بر عملکرد مدل داشته باشد تنظیم بهینه هایپرپارامترها است .نرخ یادگیری یکی از مهم‌ترین پارامترهاست؛ اگر بیش از حد بزرگ باشد مدل ممکن است در فرآیند آموزش نوسان کند و همگرا نشود و اگر بیش از حد کوچک باشد همگرایی بسیار کند خواهد بود. همچنین استفاده از بهینه‌سازهای پیشرفته مانند Adam و ... می‌تواند سرعت و دقت یادگیری را بهبود دهد. برای یافتن بهترین تنظیمات، می‌توان از روش‌هایی مانند Grid Search یا Random Search استفاده کرد.

در مجموع، بهترین روش معمولاً می‌تواند ترکیبی از این تکنیک‌ها باشد. بسته به نوع داده و میزان پیچیدگی آن می‌توان برخی از این راهکارها را اجرا کرد تا مدل بتواند به‌درستی داده‌های غیرخطی را یاد بگیرد و دقت بالاتری در دسته‌بندی آن‌ها داشته باشد.

.3

از بین روش‌های پیشنهادی برای دسته‌بندی داده‌های غیرخطی، ماشین بردار پشتیبان (SVM) با کرنل RBF را انتخاب کردیم. در این کد مدل SVM با کرنل RBF پیاده‌سازی و روی داده‌های آموزشی، آموزش داده شد و سپس مدل روی داده‌های تست ارزیابی شد و توانست دقت بهتری نسبت به موارد قبلی برای داده‌های غیرخطی داشته باشد و به دقت 92.5% رسیده است و در ادامه دقت این مدل و تصویر نهایی که مرز تصمیم رسم شده است را نمایش می‌دهیم:

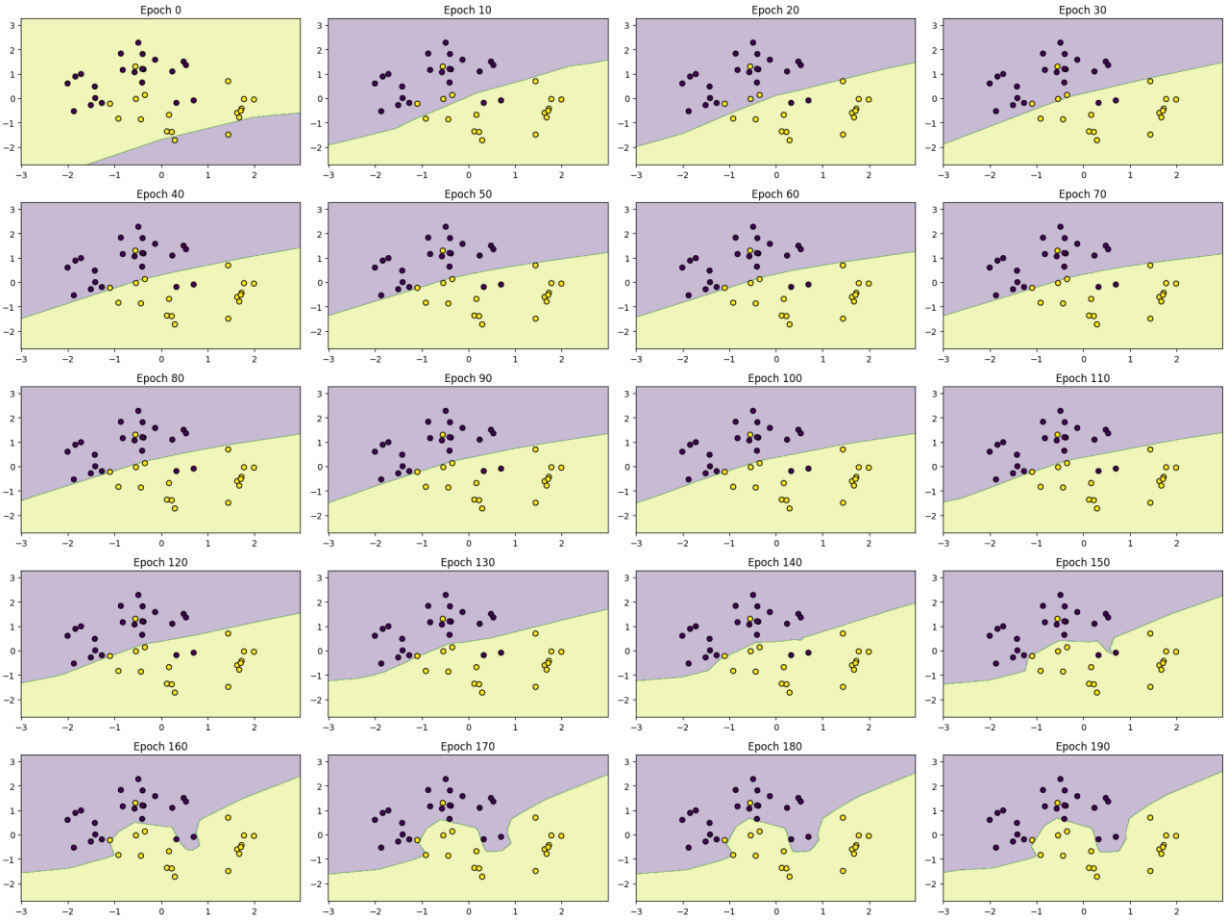
```
Test Accuracy: 92.50%
Test Recall: 94.44%
```



همچنین اگر با مدل MLP این داده‌های غیرخطی را آموزش دهیم دقت باز هم افزایش خواهد داشت و به 95% می‌رسد:

```
Epoch [0/200], Loss: 0.7520, Val Loss: 0.7429, Val Accuracy: 0.4000
Epoch [10/200], Loss: 0.5730, Val Loss: 0.5857, Val Accuracy: 0.7500
Epoch [20/200], Loss: 0.3889, Val Loss: 0.4646, Val Accuracy: 0.7500
Epoch [30/200], Loss: 0.2876, Val Loss: 0.5166, Val Accuracy: 0.7500
Epoch [40/200], Loss: 0.2816, Val Loss: 0.5817, Val Accuracy: 0.7500
Epoch [50/200], Loss: 0.2679, Val Loss: 0.4935, Val Accuracy: 0.7500
Epoch [60/200], Loss: 0.2610, Val Loss: 0.4460, Val Accuracy: 0.7500
Epoch [70/200], Loss: 0.2548, Val Loss: 0.4442, Val Accuracy: 0.7500
Epoch [80/200], Loss: 0.2486, Val Loss: 0.4365, Val Accuracy: 0.7500
Epoch [90/200], Loss: 0.2432, Val Loss: 0.4315, Val Accuracy: 0.7500
Epoch [100/200], Loss: 0.2376, Val Loss: 0.4173, Val Accuracy: 0.7500
Epoch [110/200], Loss: 0.2309, Val Loss: 0.4098, Val Accuracy: 0.7500
Epoch [120/200], Loss: 0.2190, Val Loss: 0.4073, Val Accuracy: 0.8000
Epoch [130/200], Loss: 0.2007, Val Loss: 0.3510, Val Accuracy: 0.8000
Epoch [140/200], Loss: 0.1716, Val Loss: 0.3505, Val Accuracy: 0.8000
Epoch [150/200], Loss: 0.1336, Val Loss: 0.2832, Val Accuracy: 0.8000
Epoch [160/200], Loss: 0.0924, Val Loss: 0.2633, Val Accuracy: 0.8500
Epoch [170/200], Loss: 0.0595, Val Loss: 0.2412, Val Accuracy: 0.9000
Epoch [180/200], Loss: 0.0403, Val Loss: 0.2342, Val Accuracy: 0.9000
Epoch [190/200], Loss: 0.0299, Val Loss: 0.2104, Val Accuracy: 0.9000
```

```
Test Accuracy: 95.00%
```



«. . . اسفندماه ۱۴۰۳ . . .»