به نام خدا

دانشگاه تهران

دانشکده مهندسی برق و کامپیوتر

ماشین لرنینگ

تمرین 3

نام و نام خانوادگی	محمد مشرقی
شماره دانشجویی	810199492
تاریخ ارسال گزارش	

فهرست

۱-

الف

نقش توابع هزینه

توابع هزینه در شبکه های عصبی، مانند بقیه روش های اموزشی به عنوان **یکی از اجزای اصلی آموزش و بهبود عملکرد شبکه های عصبی مورد استفاده** قرار می گیرند.

این توابع هزینه معمولاً به صورت یک تابع از خروجی شبکه و برچسب های مربوط به داده های آموزشی تعریف می شوند. هدف از تعریف تابع هزینه، مقایسه خروجی تخمینی شبکه با برچسب های واقعی داده های آموزشی و ارزیابی کیفیت عملکرد شبکه است.

تابع هزینه در شبکه های عصبی، نقش مهمی در انتخاب **روش بهینه سازی** و **پارامترهای مورد** استفاده در شبکه دارد. بهبود عملکرد شبکه بستگی به انتخاب **درست تابع هزینه** دارد.

نقش توابع فعال ساز

توابع فعال‌ساز در شبکه‌های عصبی، **بخشی اساسی** از پردازش در شبکه‌های عصبی هستند. توابع فعال‌ساز برای تبدیل ورودی هر نورون شبکه به خروجی آن نورون مورد استفاده قرار می‌گیرند. توابع فعال‌ساز به دو دسته تابع‌های **خطی** و **غیرخطی** تقسیم می‌شوند.که بیشتر توابع غیر خطی در شبکه های عصبی استفاده می شوند. مانند تابع سیگموید، تابع ReLU، تابع tanh، و تابع softmax بسیار رایج هستند. وظیفه توابع فعال‌ساز غیرخطی، تبدیل ورودی هر نورون به خروجی غیرخطی و غیریکنواخت می‌باشد.

ب

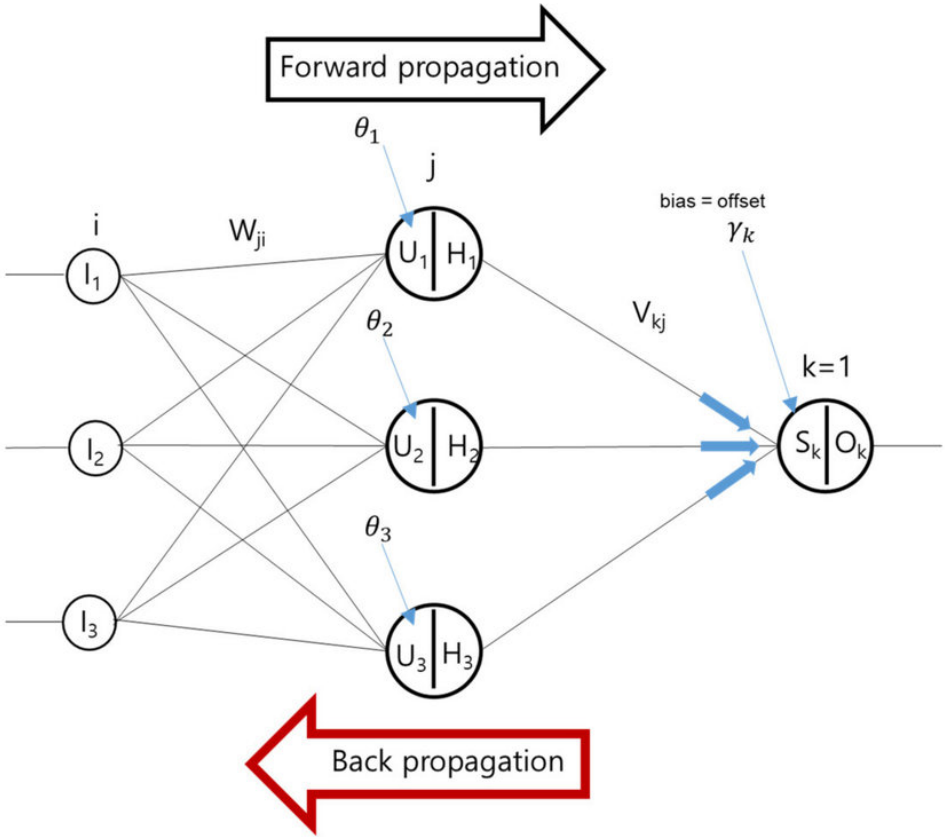
بله تاثیر زیادی دارد.

مقدار دهی اولیه مناسب باعث می شود سریع تر و دقت بیشتر به جواب نهایی برسد در حالی که مقدار دهی اولیه تصادفی باعث می شود دیر تر به جواب رسید و محاسبات بیشتری را در بر دارد حتی ممکن هست در بعضی حالات به جواب نرسد و به مشکل بخورد.

ج

پیش‌روند (Forward Propagation) : ، همچنین به عنوان فیدفوروارد شناخته می‌شود، فرایندی است که در آن داده‌های ورودی به لایه‌های شبکه‌ی عصبی به صورت پیاپی و در جهت پیشروی از لایه ورودی تا لایه خروجی رد می‌شوند. در این فرایند، داده‌های ورودی با وزن‌های نورون‌ها در هر لایه ضرب شده، و خروجی حاصل به لایه بعدی منتقل می‌شود. این فرایند تا رسیدن به لایه خروجی ادامه می‌یابد و در نهایت خروجی نهایی تولید می‌شود.

پس‌روند(Backward Propagation) : ، همچنین به عنوان بکپروپاگیشن شناخته می‌شود، فرایند محاسبه گرادیان تابع خطا نسبت به وزن‌های شبکه‌ی عصبی است که با حرکت به عقب در شبکه انجام می‌شود. این فرایند برای به‌روزرسانی وزن‌های شبکه‌ی عصبی در فرایند آموزش استفاده می‌شود، با هدف کاهش تابع خطا و بهبود عملکرد شبکه. به طور خلاصه، پیش‌روند فرایند محاسبه خروجی یک شبکه‌ی عصبی به داده‌های ورودی است، در حالی که پس‌روند فرایند به‌روزرسانی وزن‌های شبکه‌ی عصبی را براساس خروجی محاسبه‌شده و خروجی موردنظر انجام می‌دهد.



د

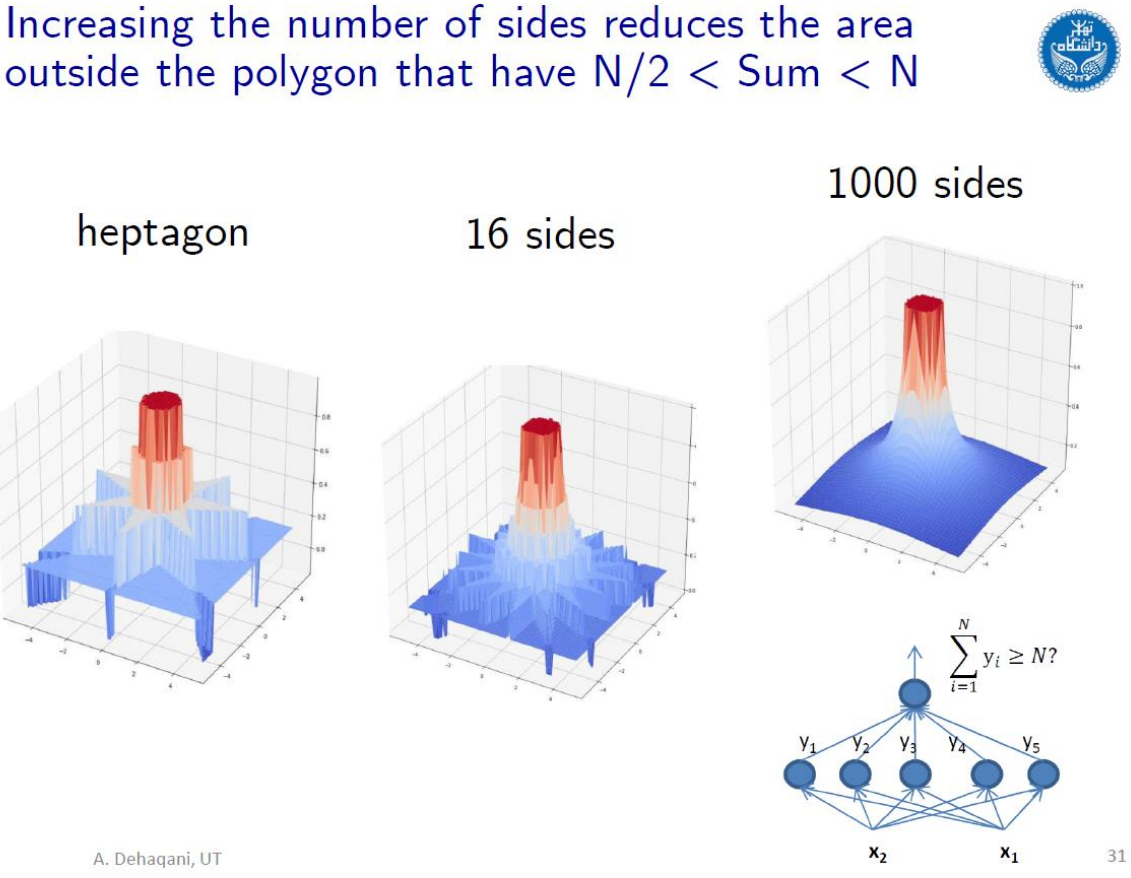
نرخ یادگیری یک هایپرپارامتر در شبکه های عصبی است که مقدار تعدادی از پارامترهای مدل مانند وزن ها و بایاس ها در هنگام آموزش تغییر می دهد. نرخ یادگیری در واقع مقداری است که اندازه گامی را که در هنگام به‌روزرسانی پارامترهای مدل به کار می‌بریم، تعیین می‌کند. این پارامتر در تعیین سرعت همگرایی مدل به جواب بهینه بسیار مهم است.

اگر نرخ یادگیری بسیار پایین باشد، ممکن است مدل برای همگرایی به یک جواب بهینه نیاز به زمان زیادی داشته باشد یا به یک جواب زیر بهینه برسد. به عبارتی دیگر، مدل به حالت دچار گیر کردن در مینیمم محلی می‌افتد. از سوی دیگر، اگر نرخ یادگیری بسیار بالا باشد، ممکن است فرایند آموزش مدل ناپایدار شود و تابع هزینه واریانس بسیار زیادی داشته باشد یا حتی منحرف شود.

به طور کلی، نرخ یادگیری باید به گونه ای انتخاب شود که مدل بتواند به سرعت و با دقت و پایداری مناسبی آموزش داده شود. با این حال، نرخ یادگیری بهینه می تواند بسته به پیچیدگی مدل، اندازه مجموعه داده و سایر عوامل مختلف، متفاوت باشد

ه

چون هرچه تعداد لایه ها بالا تر رود در انتها در از آن استفاده می کنیم و جواب های دقیق تری بدست میاریم و دقت بالاتری داریم. باتوجه به عکس هرچه لایه ها بیشتر شود محیط های مرزی دقیق تر می شوند.



و

یکی از مشکلات شایع در شبکه های عصبی، بیش‌برازش (overfitting) است که زمانی رخ می دهد که مدل به اندازه کافی به داده‌های آموزشی خود بچسبد و نویز و جزئیات از آن داده‌ها را به یاد بسپارد و در عوض برای داده‌های جدید نتواند به خوبی کار کند. این مشکل می تواند باعث کاهش عملکرد مدل در داده های آزمایشی یا در سناریوهای واقعی شود.

برای حل مشکل بیش‌برازش، روش‌های مختلفی وجود دارد که عبارتند از:

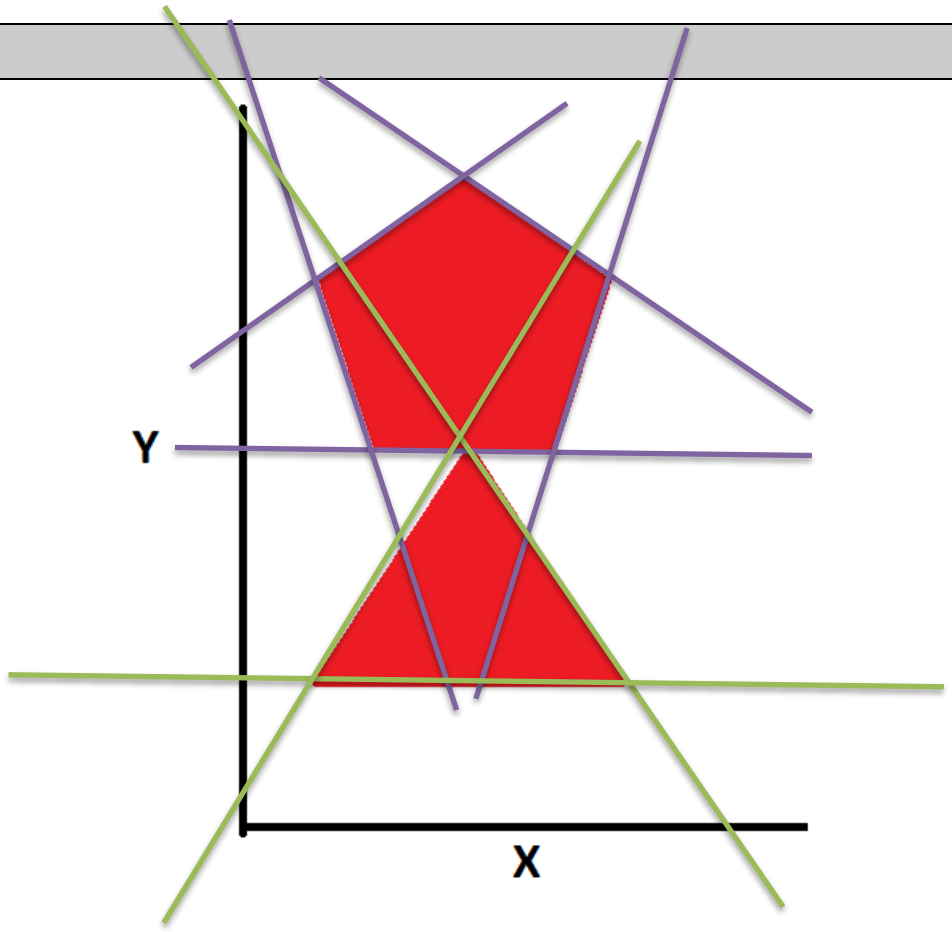
- بازتابی (رگولاریزیشن): این روش با افزودن ترمی به تابع خطا در هنگام آموزش، از بیش‌برازش جلوگیری می کند. از جمله روش‌های بازتابی می توان به رگولاریزیشن L1 یا L2، دراپ اوت (dropout) یا متوقف کردن زودهنگام (early stopping) اشاره کرد.

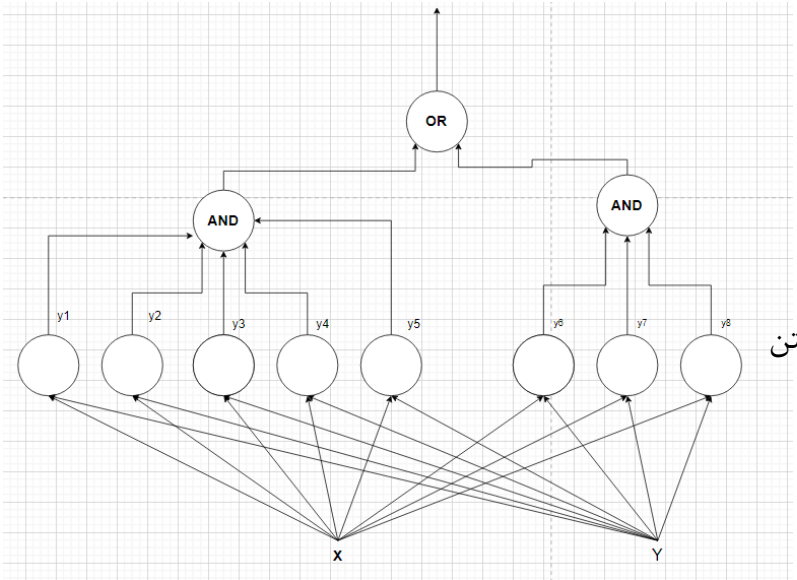
- افزایش اندازه داده: این روش شامل افزایش مصنوعی اندازه مجموعه داده آموزشی با اعمال تحولات مانند چرخش، اسکیل، یا برعکس کردن داده های موجود است که می تواند به مدل در یادگیری بهتر از داده های جدید کمک کند.

- ساختار مدل: ساختار شبکه عصبی نیز می تواند در پیشگیری از بیش‌برازش موثر باشد. از جمله روش‌های ساختاری می توان به کاهش تعداد لایه‌ها، تعداد نورون‌ها در هر لایه و یا استفاده از مدل‌های پیش آموزش دیده شده (pre-trained) اشاره کرد.

یکی دیگر از مشکلات طول کشیدن زیاد اجرای الگوریتم به صورت Batch که برای آن می توان Stochastic استفاده کرد

2





در کل ۴لایه

یه لایه نهاییی که یک نورون داره

دو لایه hidden که از بالا به پایین

به ترتیب دو نورون و ۸نورون دارد

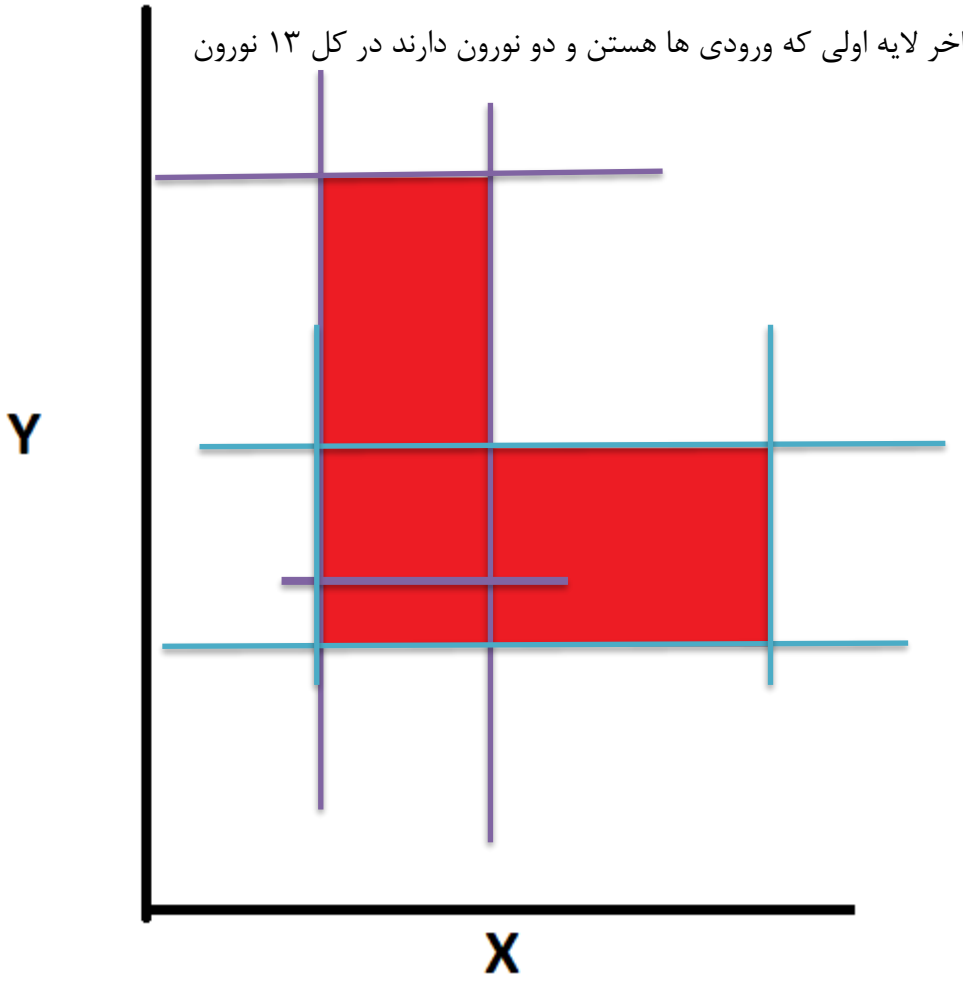
و در اخر لایه اولی که ورودی ها هستن

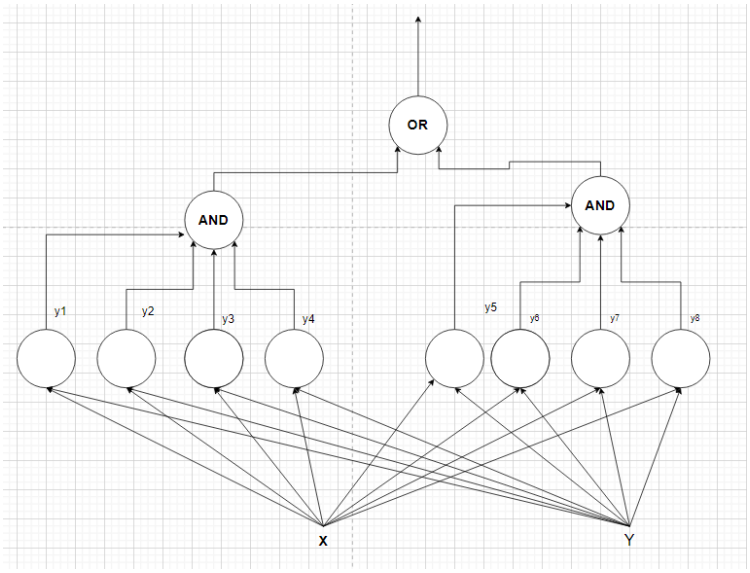
و دو نورون دارند

در کل ۱۳ نورون

در کل ۴لایه

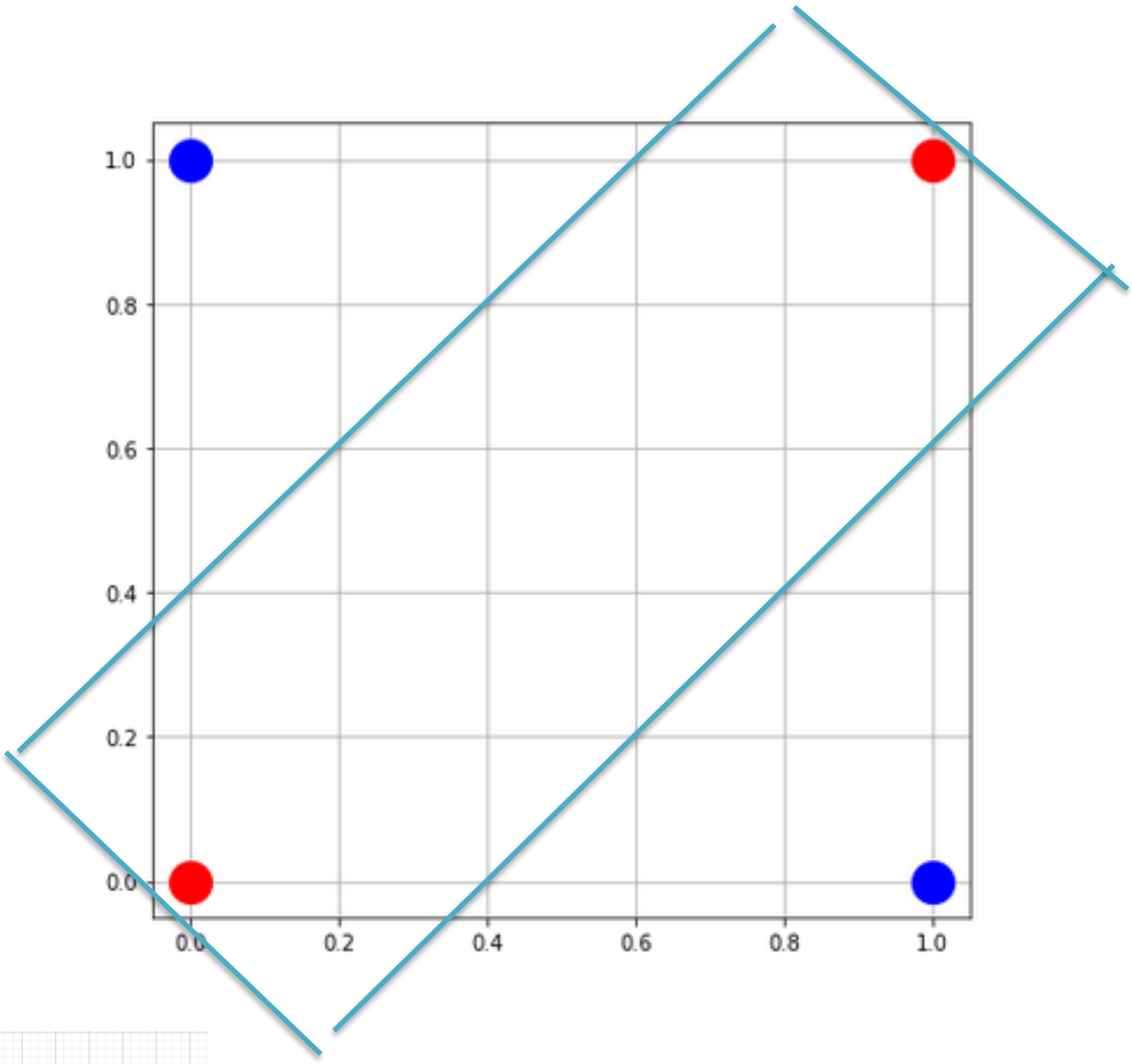
یه لایه نهاییی که یک نورون داره دو لایه hidden که از بالا به پایین به ترتیب دو نورون و ۸نورون دارد و در اخر لایه اولی که ورودی ها هستن و دو نورون دارند در کل ۱۳ نورون

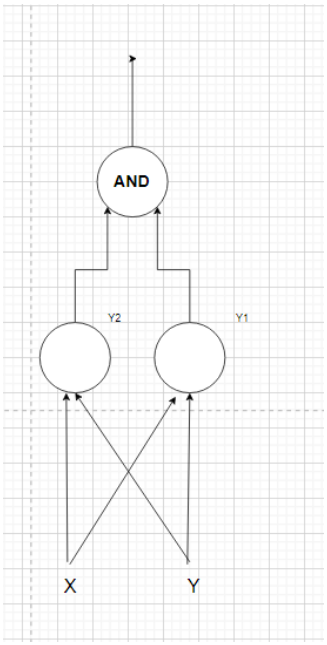




در کل ۳لایه

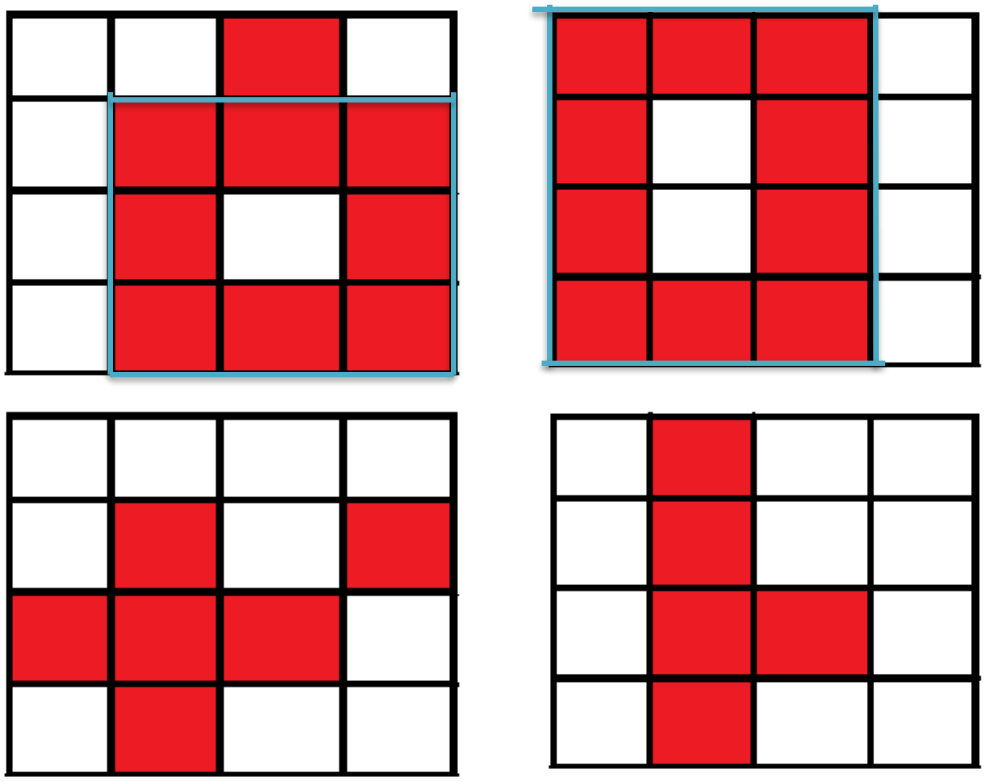
یه لایه نهاییی که یک نورون داره یک لایه hidden که و ۲نورون دارد و در اخر لایه اولی که ورودی ها هستن و دو نورون دارند در کل ۵ نورون





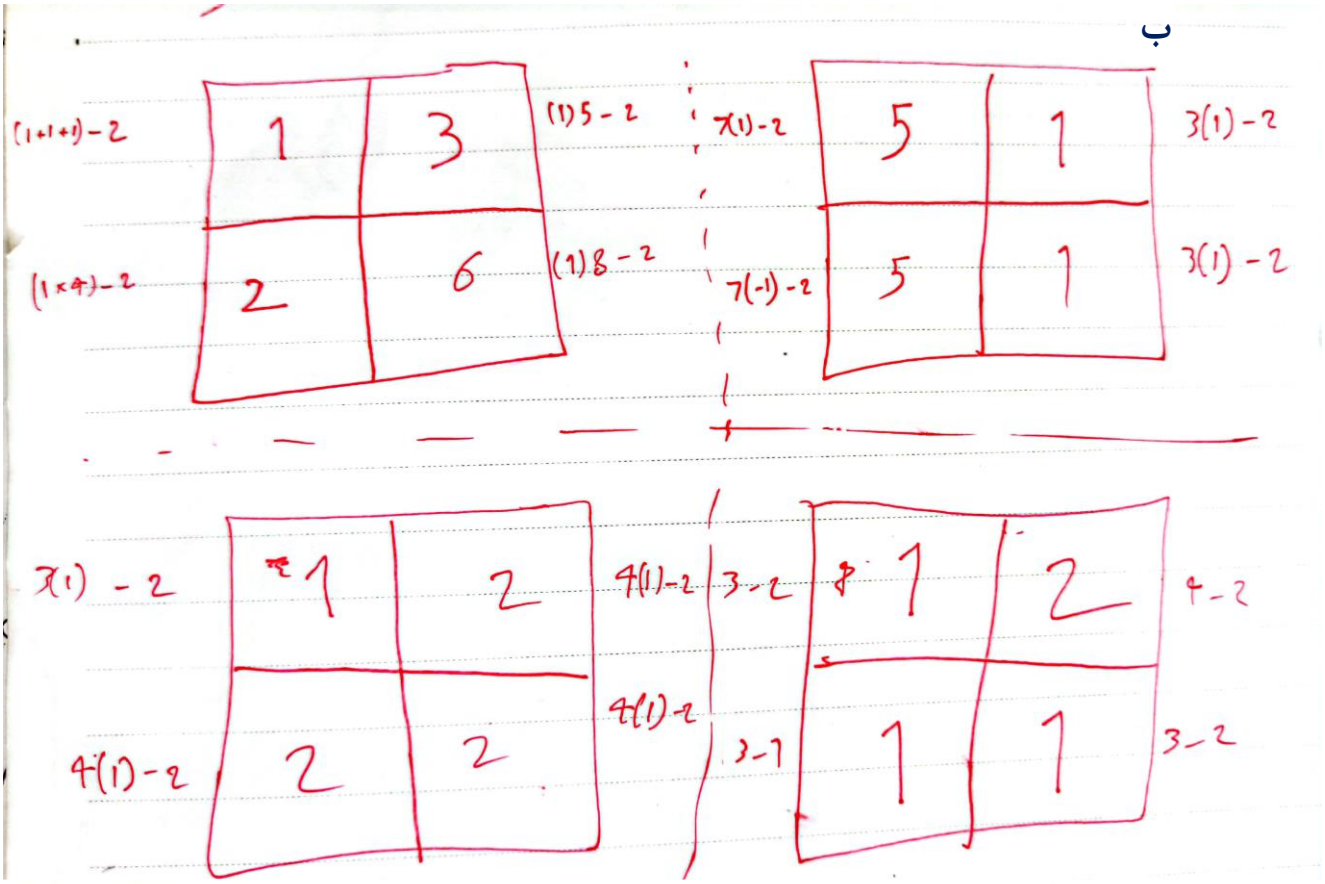
۳-

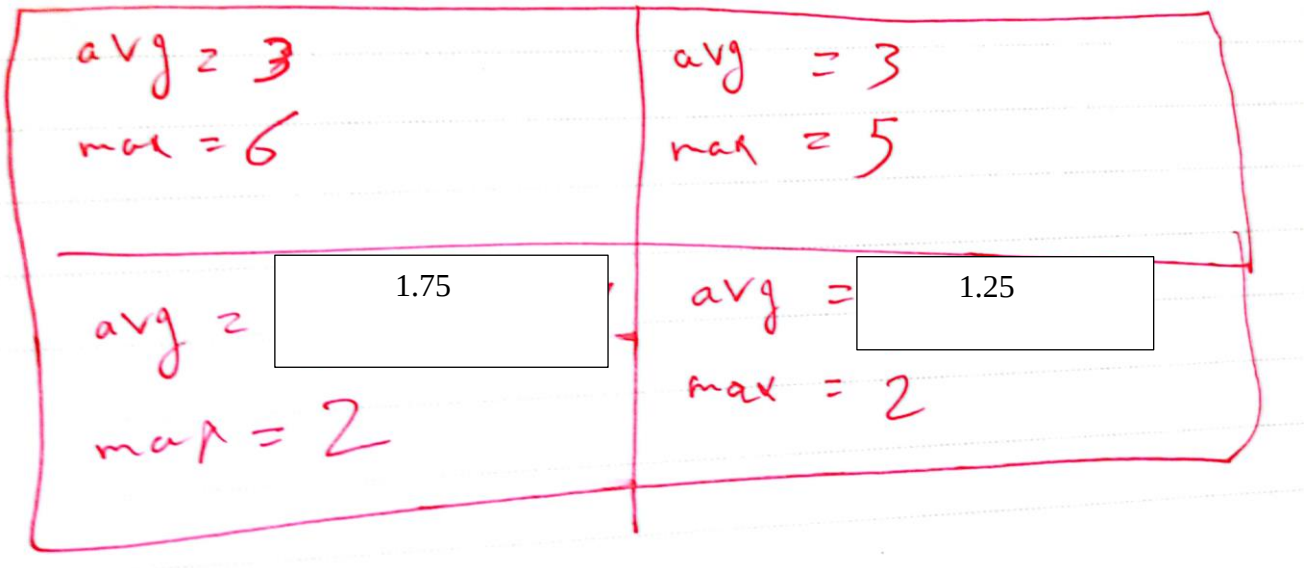
الف



ویژگی بارز که با نگاه کردن می فهمم اینه که ردیف اول یه محیط بسته تشکیل دادن در حالی که دو عکس دیگر ندادن(یعنی محیط قرمز(یک ها) محیط سفید(صفر ها) رو درون خودش جا داده یاا محیط بسته تشکیل داده)

ب



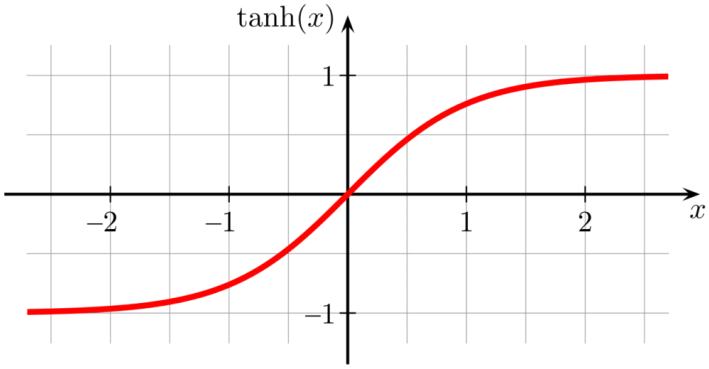


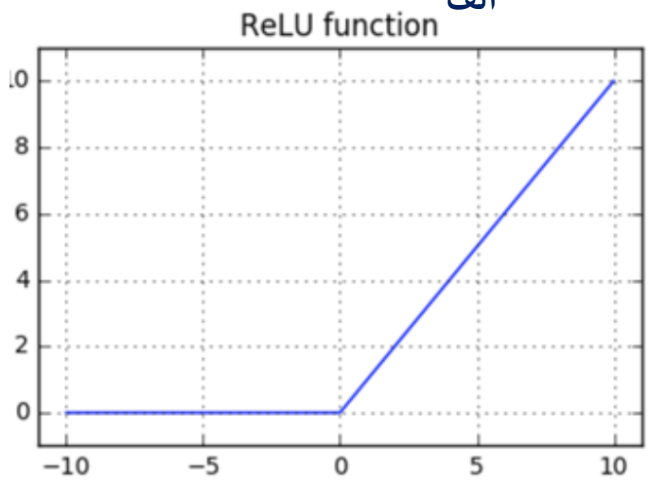
ج

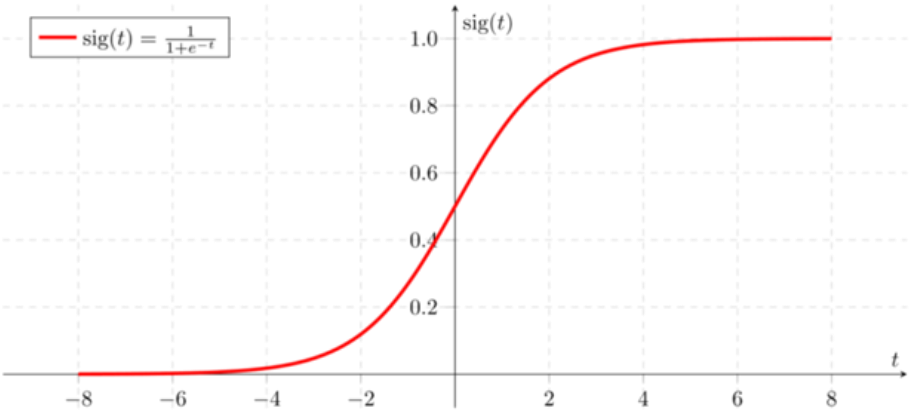
با توجه به خروجی و نتایج می تونیم بگیم که اگه میانگین خونه های بدست امده یا بیشترین مقدار یه خونه بیشتر مساوی یه حد شد اون رو در کلاس دیگر بگذاریم(گذاشتن therehold)

4-

الف







ACTIVATION FUNCTION	SIGMOID	TANH	RELU
Range	0 to 1	-1 to 1	0 to Infinity
Vanishing Gradient Problem	Yes	Yes	No
Nature	Non Linear	Non Linear	Linear
Zero Centered Activation Function	No	Yes	No
Symmetric Function	No	Yes	No
Equation	y = 1/(1+e^(-x))	y = tanh(x)	{ xi if x >=0 0 if x <=0 }
Model Accuracy	Good	Very Good	Excellent

تابع relu محاسبات کمتری دارد چون اولا تابع آن ساده است نسبت به دو تای دیگر و دوم اینکه با گرادیان گرفتن اثر آن محو می شود و دیگر در محاسبات حضور ندارد و همچنین مقدار ورودی آن همواره مثبت یا صفر است و منفی نمیشود درx منفی مقدار خروجی صفر است.

ب

مشکل گرادیان کاهشی هنگامی پیش می‌آید که گرادیان در حین برگردانده شدن به لایه‌های پایینی(back pro) شبکه به شدت کوچک شود و باعث مشکل در یادگیری شبکه شود. این مشکل در شبکه‌های عصبی عمیق با تعداد لایه‌های بیشتر رخ می‌دهد. تابع فعال‌سازی ReLU، که در صورتی که مقدار ورودی مثبت باشد مقدار ورودی را برمی‌گرداند و در غیر اینصورت صفر برمی‌گرداند، به دلیل عدم وجود مناطق اشباع کننده که گرادیان را به صفر نزدیک می‌کنند، کمتر به مشکل گرادیان کاهشی دچار می‌شود و به طور گسترده در شبکه‌های عصبی عمیق استفاده می‌شود.

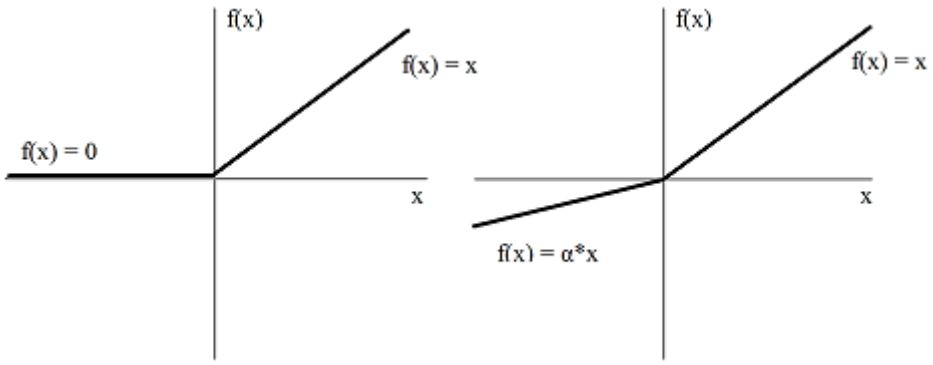
از طرف دیگر، توابع فعال‌سازی Sigma و Tanh به مشکل گرادیان کاهشی مبتلا هستند زیرا دارای مناطق اشباع کننده هستند که در صورتی که مقادیر ورودی بسیار بزرگ یا بسیار کوچک شود، گرادیان‌ها به صفر نزدیک می‌شوند. توابع Sigma و Tanh در شبکه‌های عصبی کم عمق یا در لایه‌های خروجی که مقادیر خروجی باید در یک محدوده خاص باشند، مانند مسائل طبقه‌بندی باینری، به طور معمول استفاده می‌شوند.

ج

در یک مسئله طبقه بندی، انتخاب تابع فعال سازی در آخرین لایه به الزامات و ویژگی های خاص مسئله در دست بستگی دارد. هم تابع فعال‌سازی ReLU (واحد خطی اصلاح‌شده) و هم تابع فعال‌سازی سیگموئید (که معمولاً به عنوان تابع لجستیک استفاده می‌شود) می‌توانند انتخاب‌های مناسبی باشند، اما کاربردهای آنها متفاوت است.

اگر با یک مشکل طبقه بندی باینری سر و کار داریم و خروجی های احتمالی یا تفسیرپذیری را می خواهیم، تابع فعال سازی سیگموئید می تواند انتخاب مناسبی باشد. از طرف دیگر، اگر جداسازی کلاس ها نسبتاً ساده باشد، و ما یک نمایش محاسباتی کارآمد و پراکنده را ترجیح می دهیم، تابع فعال سازی ReLU ممکن است گزینه خوبی باشد.

د



با توجه به عکس Leaky ReLU در مقادیر منفی صفر نیست و ضریب a را دارد و این باعث می شود که در گرادیان نیز حضور کم رنگی داشته باشد (برخلاف ReLU که اصلا حضوری ندارد به دلیل صفر بودن) و با این کار مشکل dying relu را حل می کنم.

انتخاب بین توابع فعال‌سازی ReLU و Leaky ReLU به مشخصات داده‌ها و مشکل مورد نظر در مدل بستگی دارد. در کل، ReLU گزینه مناسبی برای بسیاری از مسائل است زیرا یک تابع فعال‌سازی ساده و موثر است که می‌تواند نتایج خوبی تولید کند. با این حال، اگر در حین آموزش با مشکل " dying ReLU" مواجه شوید یا مشکوک به این باشید که مقادیر منفی ورودی برای مسئله شما مهم هستند، در این صورت Leaky ReLU می‌تواند گزینه بهتری باشد. استفاده از Leaky ReLU می‌تواند به پیشگیری از مشکل "dying ReLU" کمک کند و گاهاً می‌تواند عملکرد مدل را بهبود بخشد، اما با افزایش پیچیدگی یک پارامتر اضافی (مقدار آلفا) همراه است.

۵-

استخدام	تحصیلات دانشگاهی	جنسیت	سابقه کاری برحسب سال
رد	کارشناسی	مرد	1
قبول	کارشناسی	زن	2
رد	دکتری	مرد	0
قبول	کارشناسی	مرد	2
رد	کارشناسی ارشد	مرد	1
رد	دکتری	زن	1
قبول	کارشناسی ارشد	زن	0
قبول	کارشناسی	زن	1
رد	کارشناسی ارشد	مرد	0
رد	کارشناسی ارشد	مرد	2

الف :

$$H(estekhdom) = -\sum_{x} P(x)\log P(x)$$
$$= -(P(x = fail)\log P(x = fail) + P(x = pass)\log P(x = pass))$$
$$= -\left(\frac{6}{10}\log\frac{6}{10} + \frac{4}{10}\log\frac{4}{10}\right) = +0.442 + 0.528 = 0.97$$

ب:

$$H(Y \mid X_i) = \sum_{x} P(X_i = x)H(Y \mid X_i = x)$$
$$= -\sum_{x} P(X_i = x)\sum_{y} P(Y = y \mid X_i = x)\log_2 P(Y = y \mid X_i = x)$$

Information gain is difference

$$I(Y, X_i) = H(Y) - H(Y \mid X_i)$$

با توجه به فرمول بالا برای تحصیلات دانشگاهی داریم:

$$H(Y = estekhdam|X = tahsilat) =$$

$$-\big(P(PHD)(P(x = fail)\log P(x = fail) + P(x = pass)\log P(x = pass))$$
$$+ P(Master)(P(x = fail)\log P(x = fail) + P(x = pass)\log P(x = pass))$$
$$+ P(bachlor)(P(x = fail)\log P(x = fail) + P(x = pass)\log P(x = pass))\big)$$
$$= -\left(\left(\frac{3}{10}\right)(1\log 1) + \left(\frac{4}{10}\right)\left(\left(\frac{1}{4}\right)\log\left(\frac{1}{4}\right) + \left(\frac{3}{4}\right)\log\left(\frac{3}{4}\right)\right) + \left(\frac{4}{10}\right)\left(\left(\frac{1}{4}\right)\log\left(\frac{1}{4}\right) + \left(\frac{3}{4}\right)\log\left(\frac{3}{4}\right)\right)\right)$$
$$= 0.649bit$$

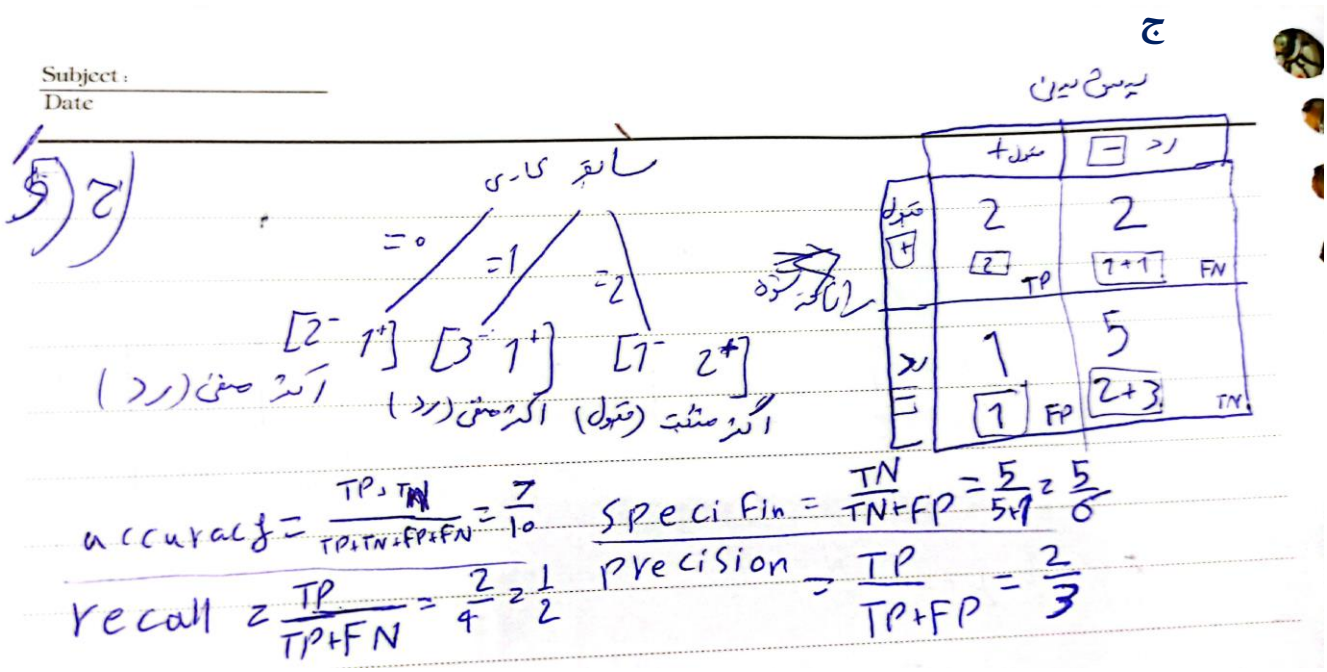
$$I(Y = estekhdam|X = tahsilat) = 0.97 - 0.65 = 0.32$$

با توجه به فرمول بالا برای سابقه کاری داریم:

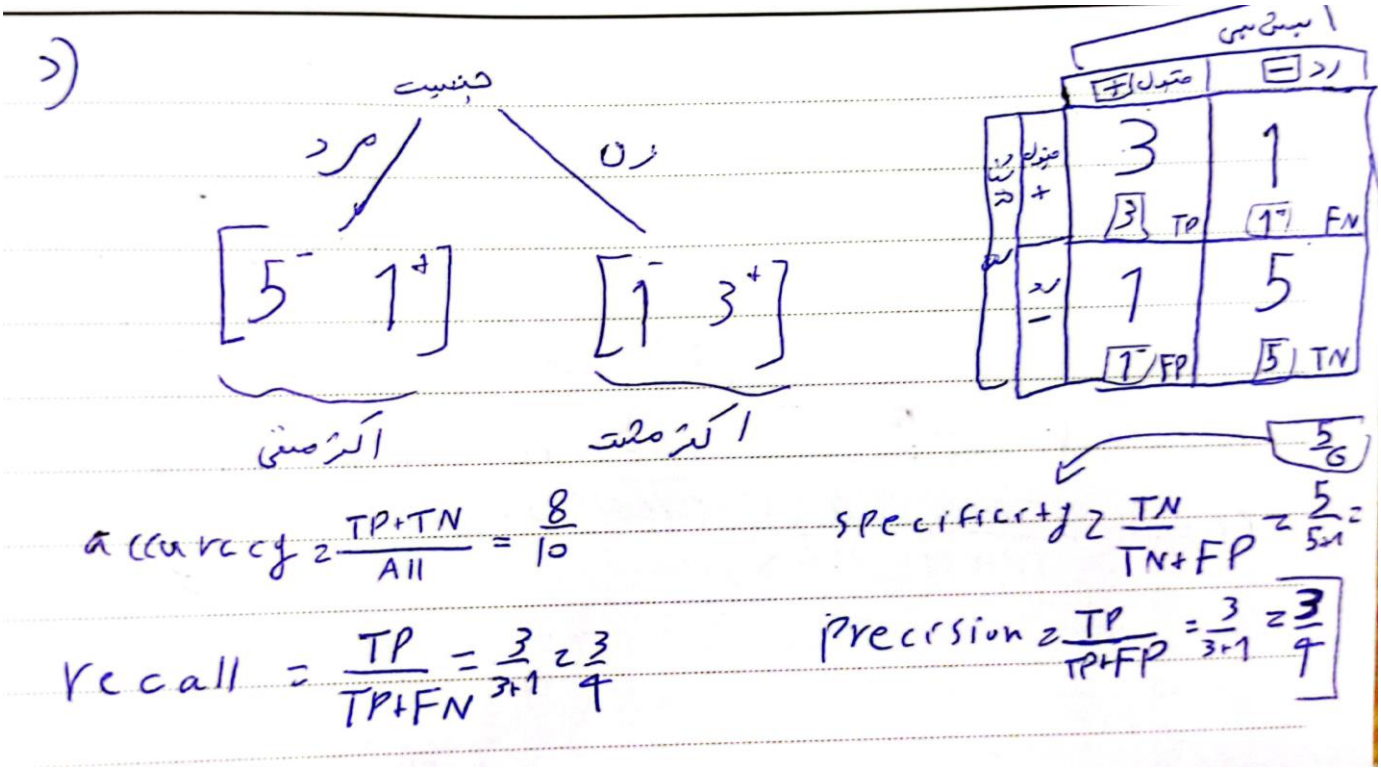
$$H(Y = estekhdam|X = sabegeh\ kari)$$
$$= -\big(P(work\ ep = 0)(P(x = fail)\log P(x = fail) + P(x = pass)\log P(x = pass))$$
$$+ P(work\ ep = 1)(P(x = fail)\log P(x = fail) + P(x = pass)\log P(x = pass))$$
$$+ P(work\ ep = 2)(P(x = fail)\log P(x = fail) + P(x = pass)\log P(x = pass))\big)$$
$$= -\left(\left(\frac{3}{10}\right)\left(\left(\frac{2}{3}\right)\log\left(\frac{2}{3}\right) + \left(\frac{1}{3}\right)\log\left(\frac{1}{3}\right)\right) + \left(\frac{4}{10}\right)\left(\left(\frac{1}{4}\right)\log\left(\frac{1}{4}\right) + \left(\frac{3}{4}\right)\log\left(\frac{3}{4}\right)\right)\right.$$
$$\left. + \left(\frac{3}{10}\right)\left(\left(\frac{2}{3}\right)\log\left(\frac{2}{3}\right) + \left(\frac{1}{3}\right)\log\left(\frac{1}{3}\right)\right)\right) = 0.875bit$$

$$I(Y = estekhdam|X = sabegeh\ kari) = 0.97 - 0.87 = 0.1$$

ج



د)


جنسیت
مرد
زن
[5⁻ 1⁺]
[1⁻ 3⁺]
اکثر صفی
اکثر ملت
پیش بینی
واقعیت
مثبت +
منفی −
3
1
3 TP
1 FN
1
5
1 FP
5 TN
5/6
accuracy = TP+TN / All = 8/10
specificity = TN / TN+FP = 5/5+1 =
recall = TP / TP+FN = 3/3+1 = 3/4
precision = TP / TP+FP = 3/3+1 = 3/4

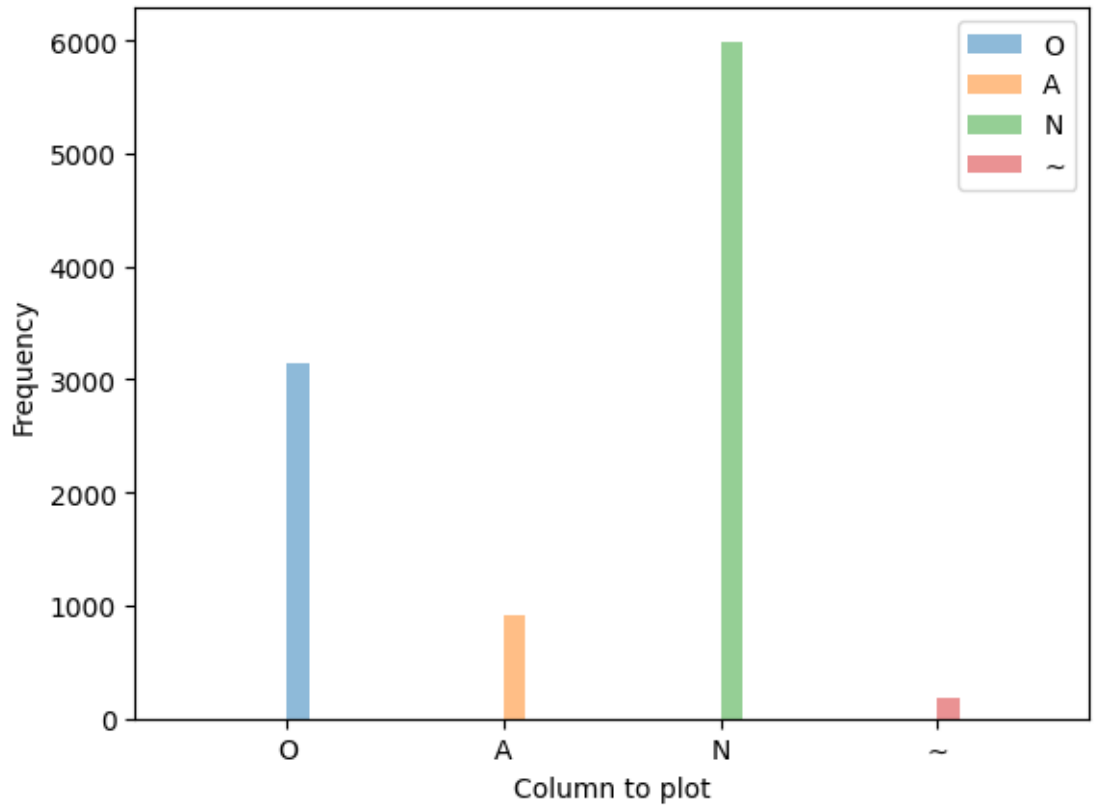
۶

الف

```
[5 rows x 171 columns]
N    5992
O    3151
A     923
~     187
Name: label, dtype: int64
```

ب

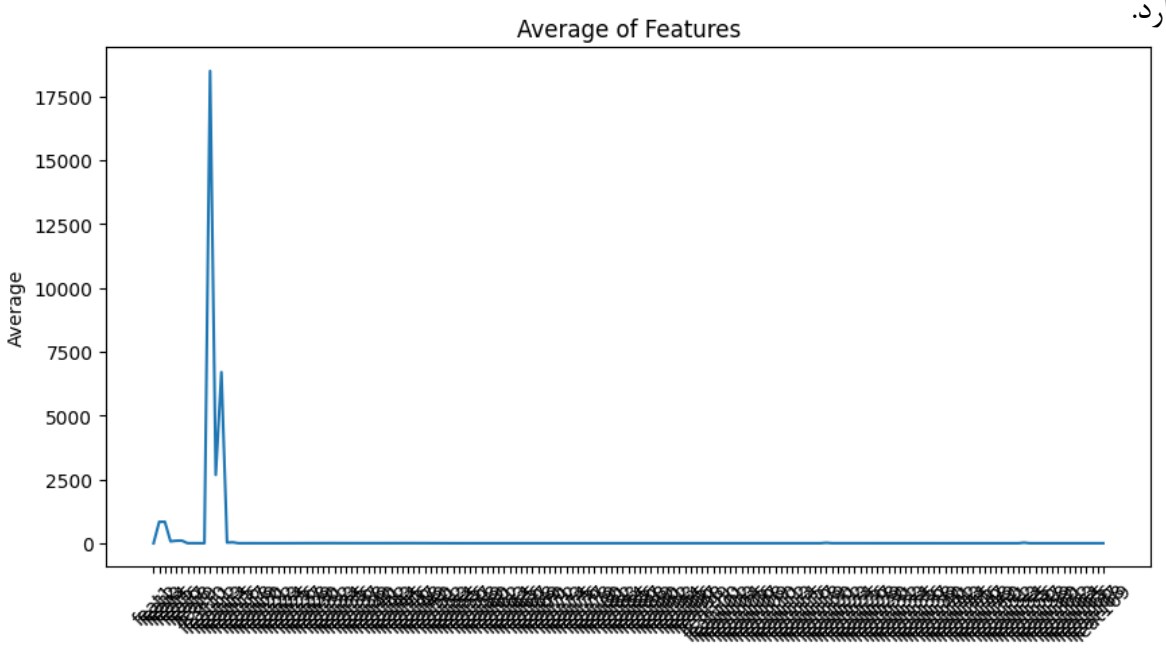
در اینجا هم تعداد داده ها و هم میانگین هر فیچر را بررسی می کنیم.



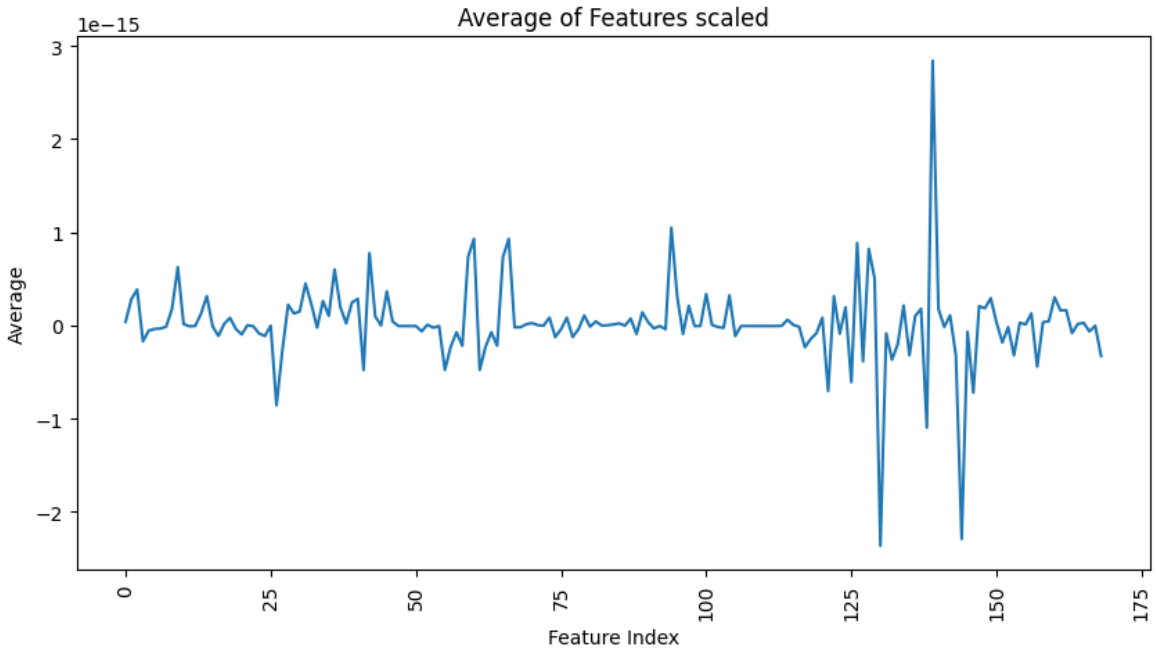
با توجه به نمودار می فهمیم تعداد نمونه های ~ و A بسیار کمتر از N و Oهست که این باعث میشه رفتار کلسیفایر عجیب باشه و شاید در ترین به دقت بالایی برسیم اما در تست جواب ها عجیب خواهد بود و ممکنه بقیه کلاس ها به خوبی تشخیص داده نشوند.

حال اگر ببینیم میانگین هر فیچر نیز یک مشکل هست که هنگام وزن دهی ممکن است اثر یه فیچر خیلی زیاد باشد و دقت را پایین بیاورد

همانطور که می بینید یکی از فیچر ها میانگینش حتی به ۱۷۰۰۰ هزار رسیده که اثر منفی زیادی دارد.



حال اگر اسکیل کنیم داریم : که تقریبا در رنج خوبی هستش



در اینجا ما درکل در حال سوگیری(bias) و دقت گمراه کننده و نادرست و ممکنه اورفیت که باعث عملکرد ضعیف کلاسیفایر می شود.

ج

```
              precision    recall  f1-score   support

           A       0.00      0.00      0.00        98
           N       0.58      1.00      0.73       590
           O       0.00      0.00      0.00       321
           ~       0.00      0.00      0.00        17

    accuracy                           0.58      1026
   macro avg       0.14      0.25      0.18      1026
weighted avg       0.33      0.58      0.42      1026

[[  0  98   0   0]
 [  0 590   0   0]
 [  0 321   0   0]
 [  0  17   0   0]]
```

داده های Nرا بهتر ار هر داده ای برازش کرده چون داده های دیگر به اندازه تعداد داده های N نبوده نتوانستند خودی نشان دهند همچنین اسکیل نبودن داده ها هم یک مشکل دیگر است. چون فیچر های بزرگ وقتی از چند تا نورون رد میشوند و چند بار در ضریب های مختلف ضرب میشوند مقادیر آن ها بسیار بزرگ می شوند و در ترین خرابکاری بدست می آید و درست ترین نمی شود.

باتوجه به داده ها N بهترین برازش رو داشته و خوب ترین شده

د

```
precision    recall  f1-score   support

A       0.86      0.85      0.85        98
N       0.89      0.90      0.90       590
O       0.81      0.80      0.80       321
~       0.42      0.47      0.44        17

accuracy                           0.85      1026
macro avg       0.74      0.75      0.75      1026
weighted avg       0.85      0.85      0.85
1026

[[ 83   6   8   1]
 [  3 529  52   6]
 [ 10  51 256   4]
 [  1   6   2   8]]
```

ه

با توجه به حرف های گفته شده در بالا و نتایج

ایندفعه چون داده ها اسکیل شدند دیگر تاثیر زیادی بر روی داده ها نداشتیم و هم رنج شدن اندازه فیچر ها دیدیم که ترین بهتر میشود در شبکه های عصبی و همچنین داده هایی که تعدادشان هم زیاد نبود در تست شناسایی شدند و ماتریس کانفیوژن و بقیه مقادیر این موضوع رو تایید می کنند.

و

بعد از پردازش داریم:

```
N    5992
O    4074
Name: label, dtype: int64
```

حال بعد از تست داریم:

```
              precision    recall  f1-score   support

           N       0.92      0.92      0.92       608
           O       0.88      0.87      0.88       399

    accuracy                           0.90      1007
   macro avg       0.90      0.90      0.90      1007
weighted avg       0.90      0.90      0.90      1007

[[560  48]
 [ 51 348]]
```

که با تقریبا برابر شدن تعداد داده ها دوباره دقت و بقیه مقادیر بهتر شدن و این نشون میده که باید در داده ها تعداد داده ها باید بین یه نسبتی باشند نه اینکه تعداد یکی خیلی خیلی بیشتر از دیگری باشه.

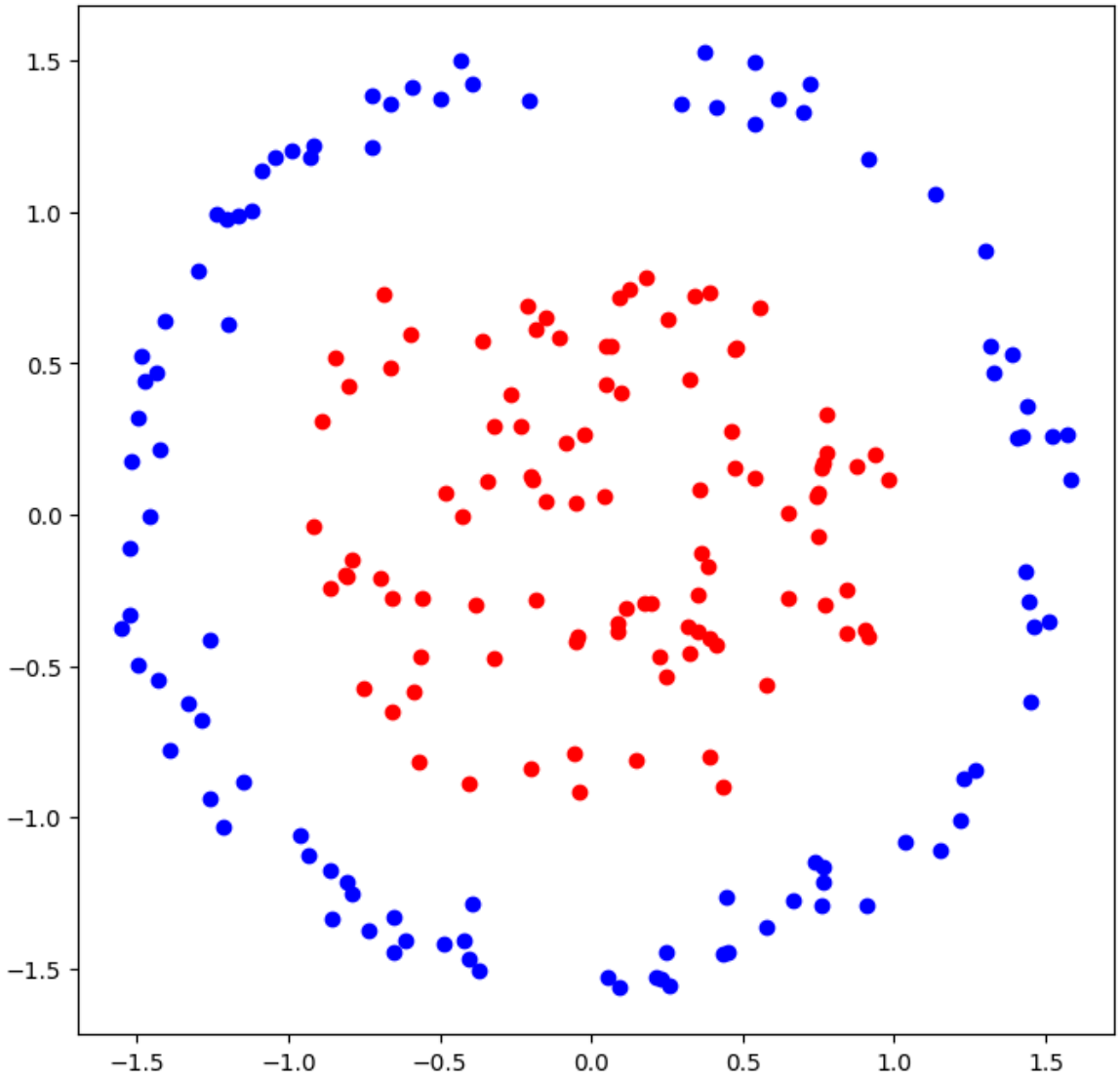
N = normal

O = abnormal

۷

الف

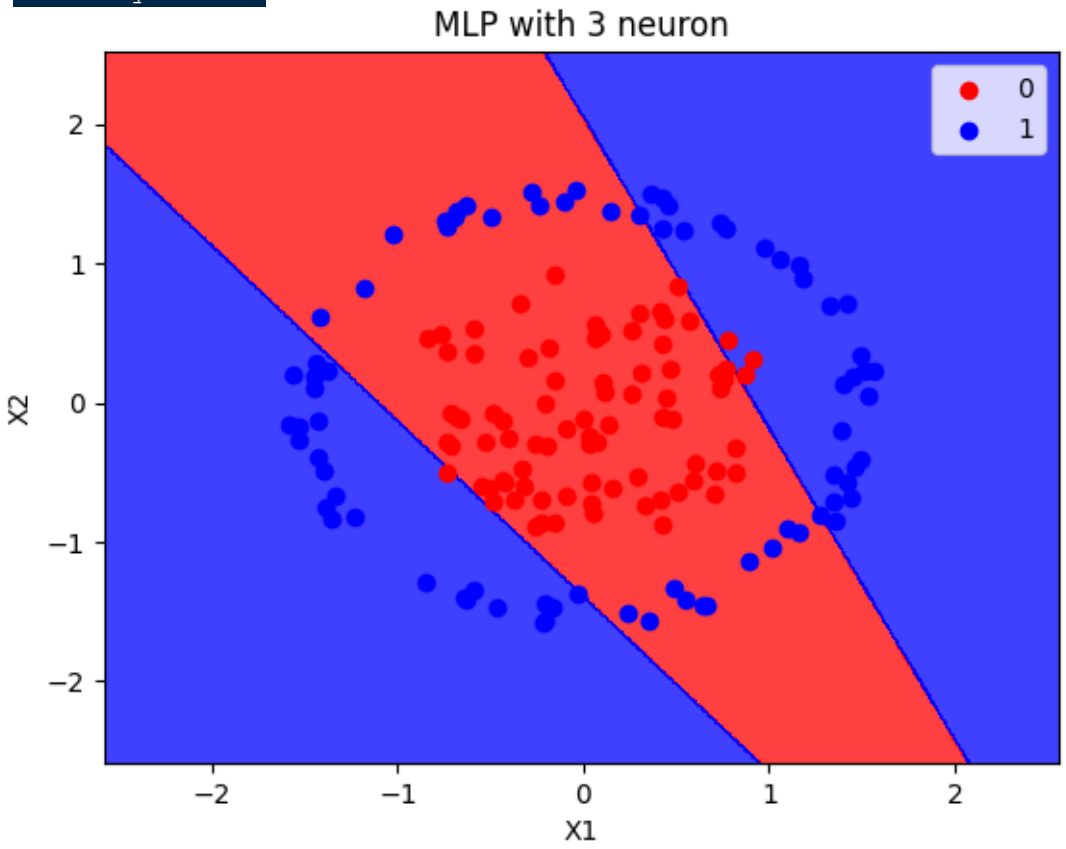
برا دیتا ست داریم:



ب

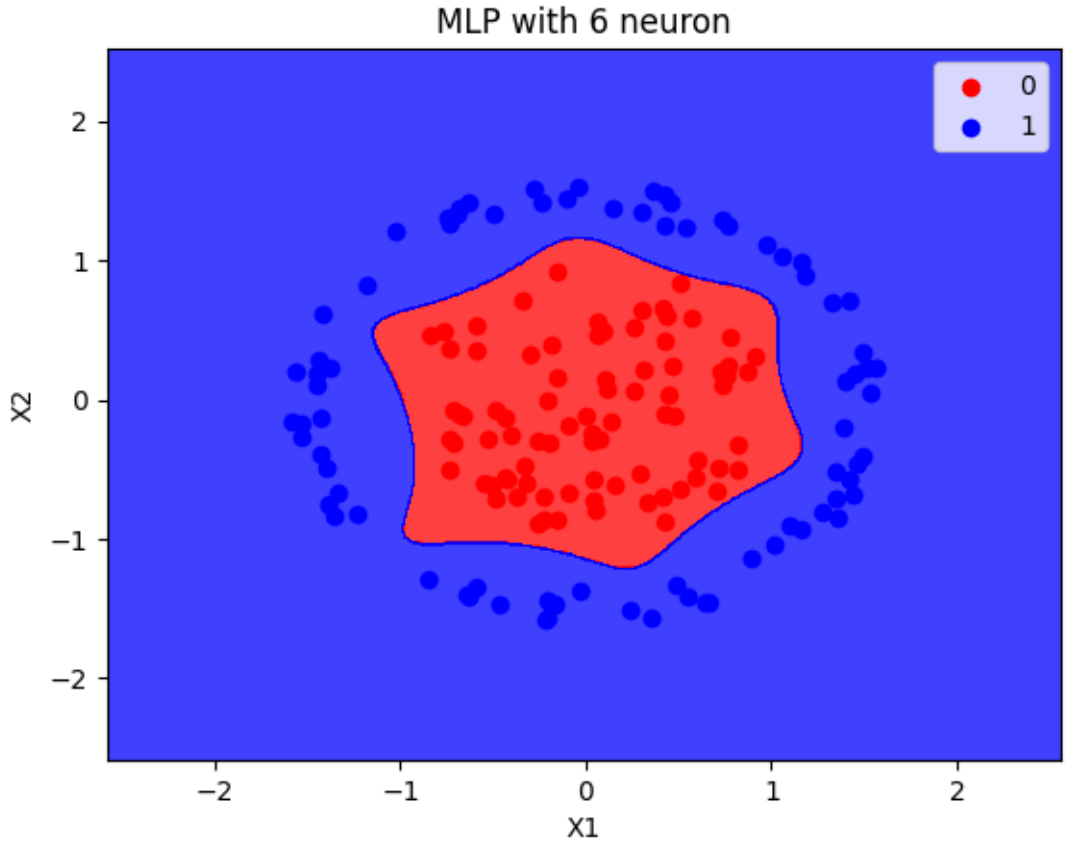
نتیجه تست با سه نورون :

```
[0 0 1 0 0 1 0 1 0 0 1 1 0 0 0 1 0 1 1 0 0 0 0 0 0 1 1 1 1 0 0 0 1 1 0
0 1
 1 0 1]
[0 1 1 0 1 1 0 1 0 1 0 1 1 1 0 0 0 1 0 0 1 1 0 1 0 1 1 1 1 0 0 0 1 1 0
1 1
 0 0 1]
Accuracy: 0.675
```



ج

```
[0 1 1 0 1 1 0 1 0 1 0 1 1 1 0 0 0 1 0 0 1 1 0 1 0 1 1 1 1 0 0 0 1 1 0
1 1
 0 0 1]
[0 1 1 0 1 1 0 1 0 1 0 1 1 1 0 0 0 1 0 0 1 1 0 1 0 1 1 1 1 0 0 0 1 1 0
1 1
 0 0 1]
Accuracy: 1.0
```



د

برای سه نورون :

البته من چند بار ران کردم و نتایج مختلفی دیدم بعضی وقت ها دقت تا ۹۰ درصد می رسید و شکل رو به خوبی نشون میداد.
بعضی وقت ها مثل دقت بین ۷۰ تا ۸۰ درصد بود و مثل یک تابع درجه دو عمل می کرد.

و همش بستگیه مقدار اولیه وزن ها به نظرم داره اونم به خاطر افتادن به مینیمم لوکال هستش فکر کنم.

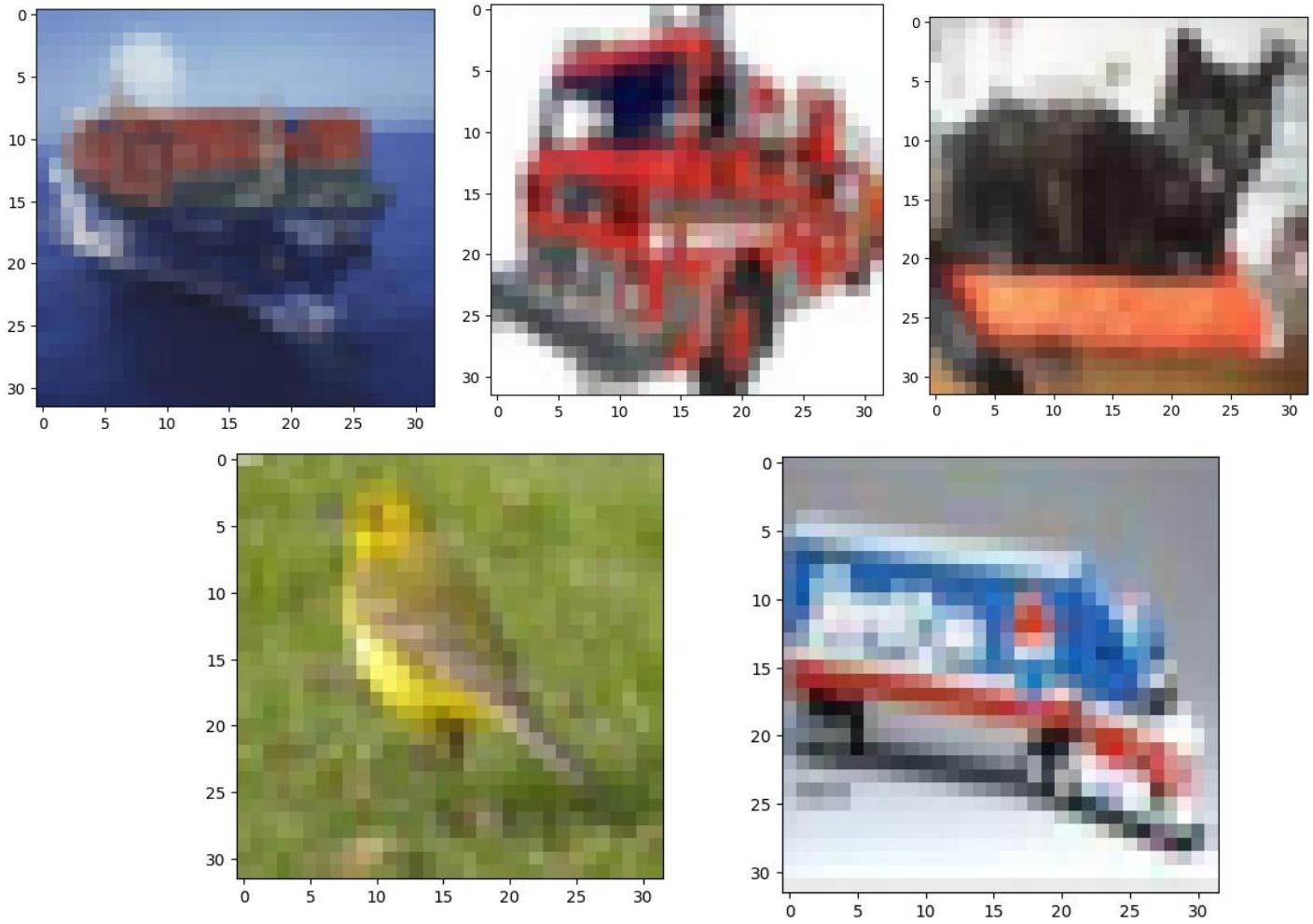
برای مقایسه می تونم بگم که ۶ نورون خیلی بهتر از ۳ نورون عمل کرده و خوب بوده اما در عوض اورفیت شد (درسته دقت تست ۱۰۰ هستش اما ممکنه داده های دیگری باشه که ما هنوز ندیده باشیم)

بعدشم نسبت به ۳ نورون زیاد از مقدار اولیه وزن ها تاثیر نمی گیره

اما نسبت به ۳ نورون هزینه پیاده کردنش بیشتره

۸

الف



ب

داده ها لیبل را ابتدا دسته بندی کرده و داده x را ابتدا فلوت کرده و اسکیل (بین ۰ و ۱)می کنیم.

و بعد جدا می کنمشون

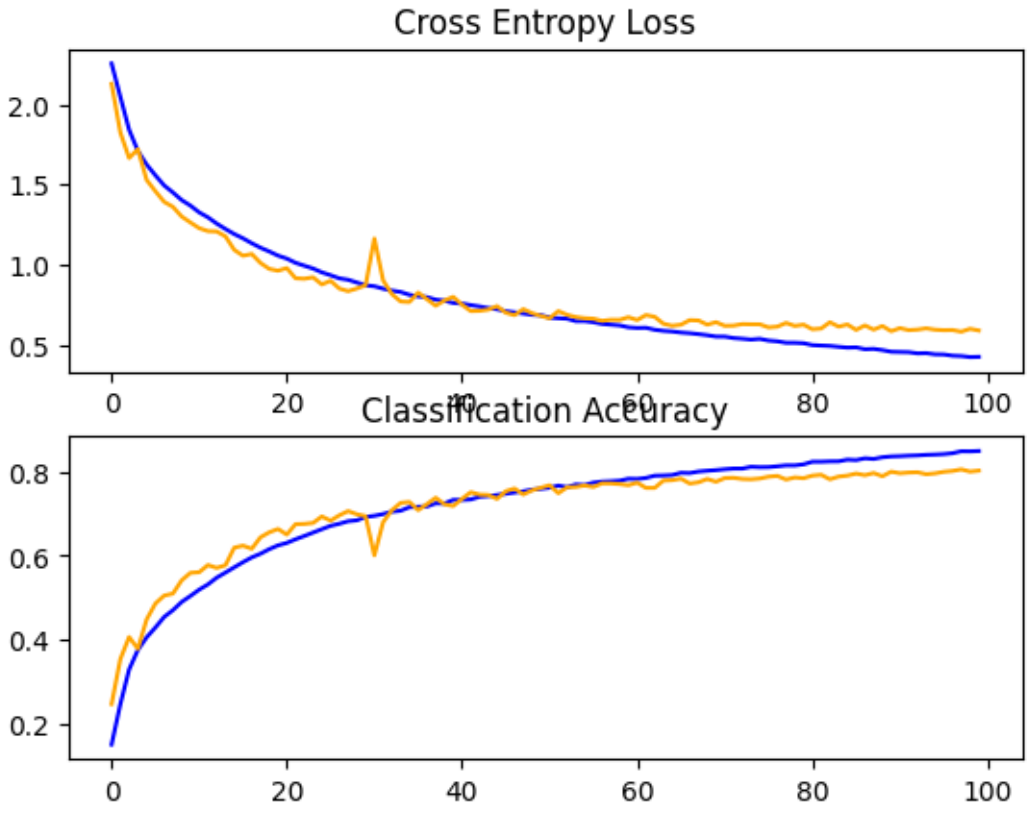
```
trainY = to_categorical(trainy)
testY = to_categorical(testy)
train_norm = trainX.astype('float32')
test_norm = testX.astype('float32')

trainX = train_norm / 255.0
testX = test_norm / 255.0
 trainX, x_val, trainY, y_val = train_test_split(trainX, trainY,
test_size=0.2, random_state=42)
```

ج و د:

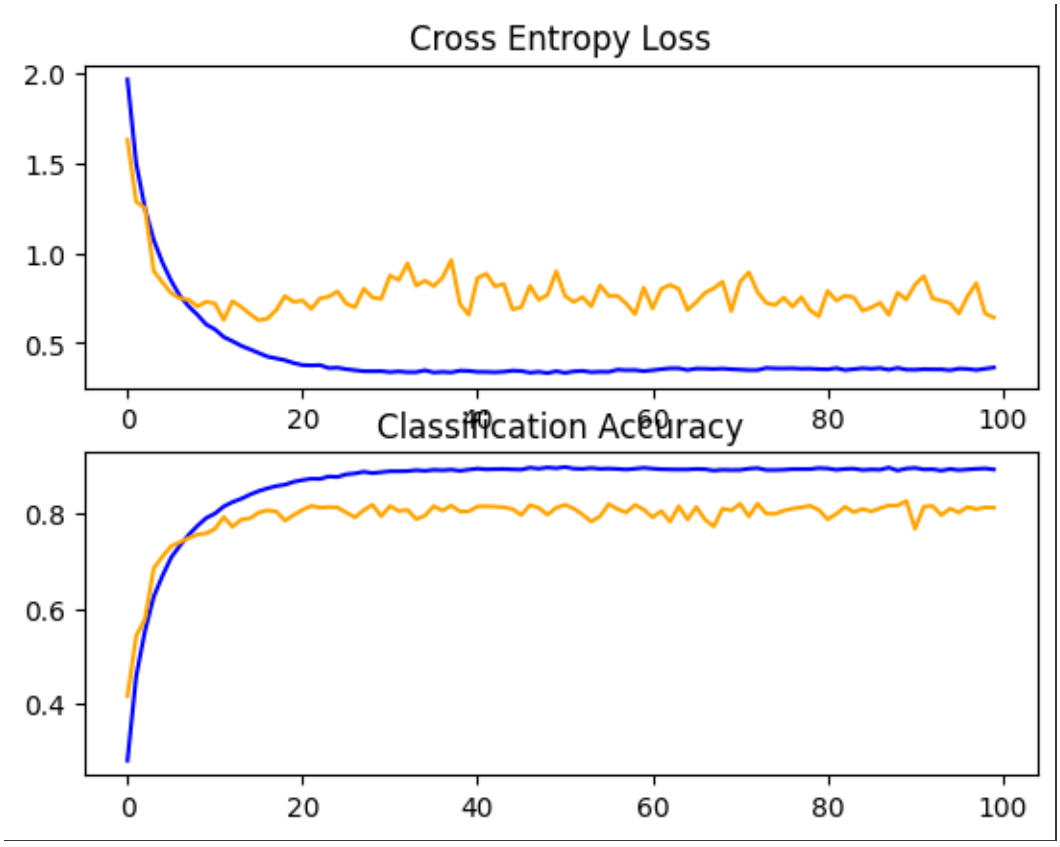
SGD

```
Accuracy: 0.800100
Precision: 0.800586
Recall: 0.800100
F1 score: 0.797951
confusion_matrix :
[[847  16  19  17  11   3   9  16  37  25]
 [ 16 900   1   3   1   4   8   1  11  55]
 [ 65   3 627  42  91  63  66  35   4   4]
 [ 16   5  41 582  71 171  48  46   4  16]
 [ 14   1  22  24 811  23  24  74   5   2]
 [ 12   3  16 115  39 729  16  60   4   6]
 [  7   2  20  41  34  15 864   7   5   5]
 [  9   0  11  28  31  34   0 881   0   6]
 [ 70  19   6   5   5   4   5   6 860  20]
 [ 19  44   2   5   3   3   1   6  17 900]]
```



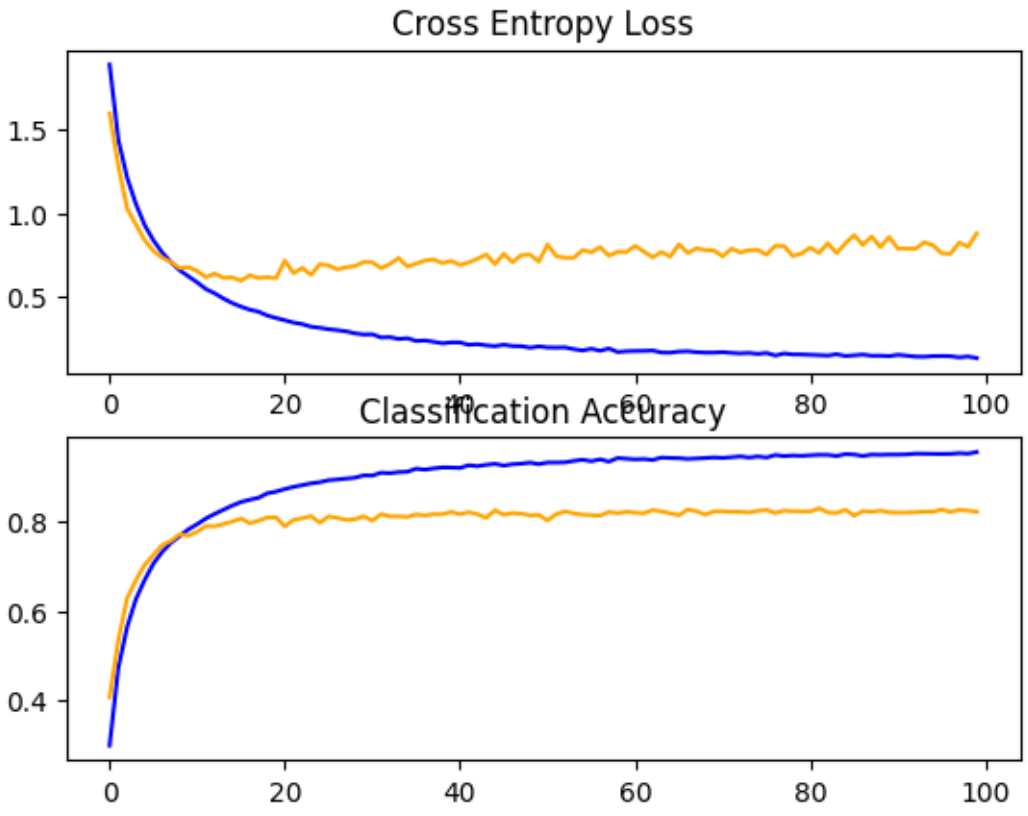
rmsprop

```
Accuracy: 0.808500
Precision: 0.816466
Recall: 0.808500
F1 score: 0.809440
confusion_matrix :
[[815  13  50  22   8   0   8   3  53  28]
 [  5 920   4   6   3   1   5   0   7  49]
 [ 51   2 744  47  49  23  58   6  12   8]
 [ 20   4  66 741  35  55  45  12  13   9]
 [  8   1  75  55 791  10  35  14  10   1]
 [  5   2  60 206  33 647  24  16   5   2]
 [  7   3  46  47   7   2 878   1   3   6]
 [ 18   3  37  58  59  16  12 778   6  13]
 [ 50  23  12   9   2   0   3   1 886  14]
 [ 15  50   3  11   4   0   4   2  26 885]]
```



Adam

```
Accuracy: 0.816400
Precision: 0.816935
Recall: 0.816400
F1 score: 0.815327
confusion_matrix :
[[883  10  19  22   8   4   9   3  29  13]
 [ 13 880   2   4   1   2  10   0  17  71]
 [ 59   2 709  36  59  47  69  11   6   2]
 [ 25   7  55 600  42 150  82  22   8   9]
 [ 15   0  55  31 798  20  43  30   5   3]
 [  4   2  28 114  38 766  26  17   2   3]
 [  5   1  29  26  22   5 904   3   3   2]
 [ 12   4  22  29  46  37  10 838   2   0]
 [ 60  16   9  10   2   1   7   3 871  21]
 [ 27  24   1   9   1   0   7   5  11 915]]
```



۹

داده ها رو می بینیم

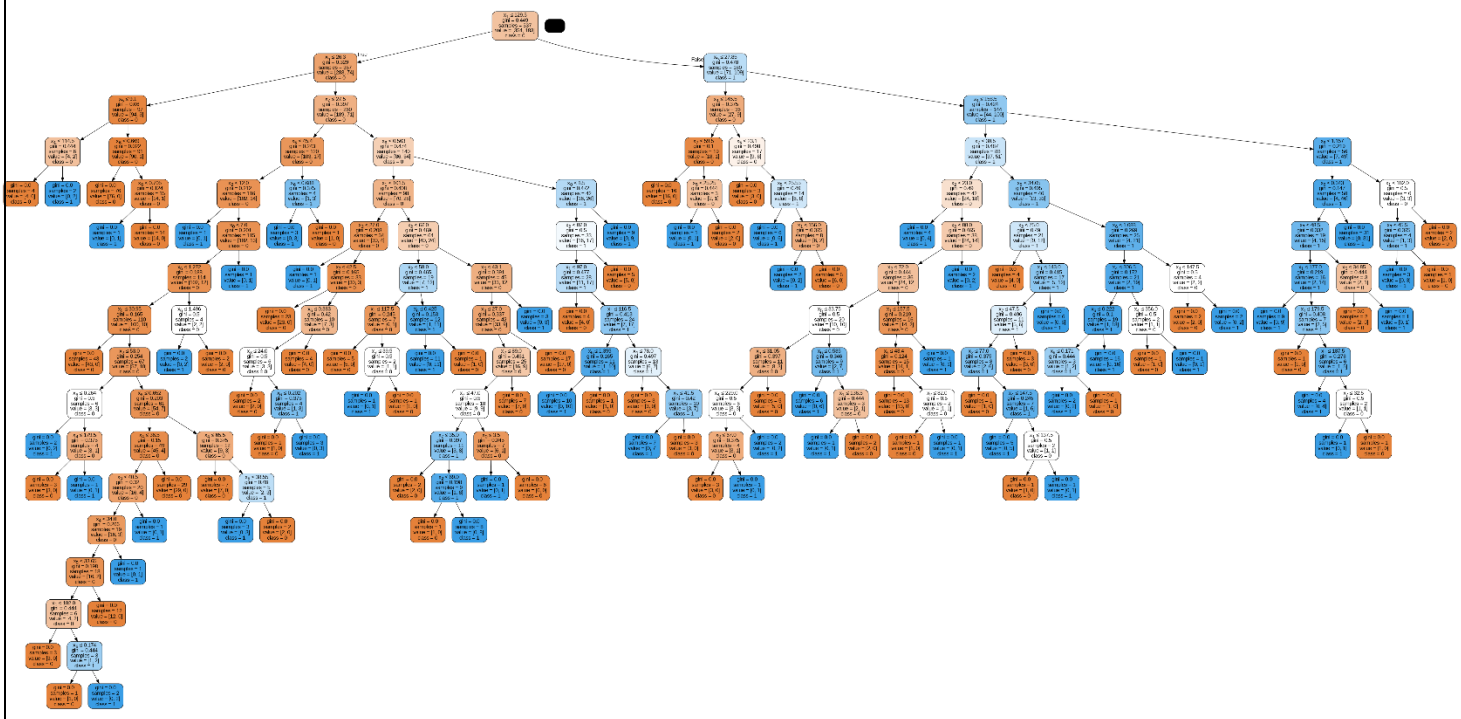
```
   pregnant  glucose  BP  skin  insulin   BMI  pedigree  age  label
0         6      148  72    35        0  33.6     0.627   50      1
1         1       85  66    29        0  26.6     0.351   31      0
2         8      183  64     0        0  23.3     0.672   32      1
3         1       89  66    23       94  28.1     0.167   21      0
4         0      137  40    35      168  43.1     2.288   33      1
0    500
1    268
Name: label, dtype: int64
```

الف

```
Accuracy for test data: 0.683982683982684
Accuracy for train data: 1.0
```

می بینیم که اورفیت شده.

ب



ج

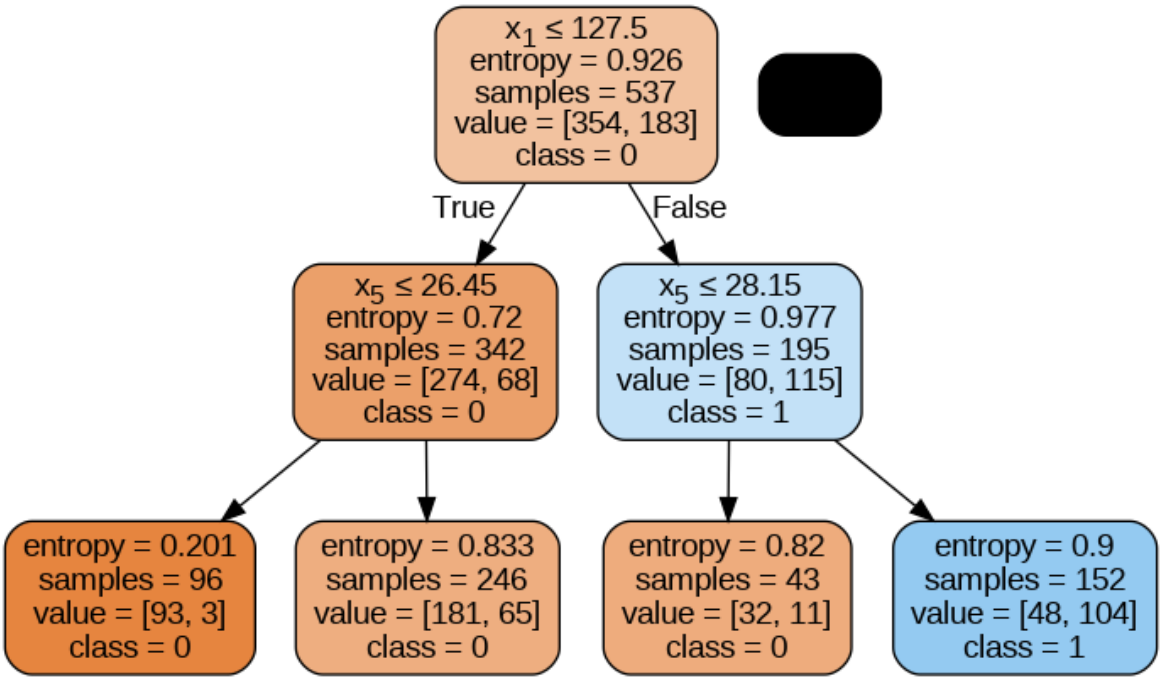
پیش‌قطعه‌گذاری (Pre-pruning) یک روش استفاده شده در الگوریتم‌های درخت تصمیم‌گیری برای محدود کردن اندازه درخت قبل از رسیدن به حداکثر عمق آن.

هدف اصلی این روش جلوگیری از بیش‌برازش مدل به داده‌های آموزشی است که ممکن است باعث کاهش کارایی در داده‌های جدید و ناشناخته شود. در پیش‌قطعه‌گذاری، الگوریتم درخت تصمیم‌گیری معیارهایی را تعیین می‌کند که با رسیدن به آن‌ها، رشد درخت را متوقف می‌کند. به عنوان مثال، این الگوریتم ممکن است رشد درخت را متوقف کند اگر تعداد نمونه‌ها در یک گره به زیر یک آستانه مشخصی برسد، یا اگر حداکثر عمق درخت را به دست آورد.

استفاده از پیش‌قطعه‌گذاری باعث می‌شود که درخت به داده‌های جدید بهتری عمل کند، زیرا به احتمال زیاد نویز داده‌های آموزشی را دربرنمی‌گیرد. با این حال، این روش نیز ممکن است باعث کم‌برازشی شود اگر درخت به صورت نامناسبی قطع شود و نتواند الگوهای مهم در داده‌ها را یاد بگیرد. بنابراین، انتخاب معیارهای مناسب برای متوقف کردن رشد درخت بسیار مهم است تا بین بیش‌برازشی و کم‌برازشی تعادلی مناسب برقرار شود.

د

```
Accuracy for test data: 0.7705627705627706
Accuracy for train data: 0.7635009310986964
```



۵

با توجه به نتایج می بینیم که دفعه اول اورفیت شده و دقت ترین ۱۰۰ و دقت تست ۶۸ درصد که قشنگ این موضوع را اثبات می کند.

اما حال برا دفعه دوم میایم شرط می زایم که ارتفاعش بیشتر از دو تا نشود اینکار باعث می شود بهترین شرط را برا ارتفاع با آنتروپی پیدا کند درست است دقت تمرین صد نمیشود اما باعث میشه دقت داده ها تست بیشتر بشه و دقت این دو به هم نزدیک شوند.