به نام خدا

تمرین سری سوم درس یادگیری ماشین

فردین عباسی 810199456

دانشکده مهندسی برق و کامپیوتر

دانشگاه تهران

بهار 1402

سوالات

سوال 1

الف)

در هر نورون یک Activation Function وجود دارد که معمولا خروجی Affine Function را به صورت باینری کد می کنند و این عملکرد در مغز انسان نیز مشهود است. اگر این تابع اصطلاحا self defferential باشد برای back propagation در فاز یادگیری بسیار مفید است.

در آخرین لایه شبکه عصبی پاسخ خروجی را می توان با پاسخ مطلوب مقایسه کرد. برای این مقایسه سنجه ها و به اصطلاح توابع هزینه ای وجود دارد تا بررسی کرد که چه قدر عملکرد شبکه بهینه است. مشتق پذیر بودن این تابع هزینه فرآیند یادگیری را سریع تر می کند.

ب)

مقدار دهی اولیه بایاس ها و وزن ها هم در دقت خروجی شبکه و هم در سرعت یادگیری آن موثر است.

انتخاب مقادیر کوچک سبب می شود شبکه در مینیمم محلی گیر کند و انتخاب مقادیر بزرگ گرادیان بزرگ می سازد که در هردو حالت فرآیند یادگیری را با اخلال مواجه می کند.

همچنین مقدار دهی اولی وزن ها باید از توزیع رندوم برگرفته شده باشد زیرا توزیع ثابت وزن ها سبب یکنواختی لایه های شبکه می شود در صورتی که با تنوع رفتار نورون ها هر یک به جنبه ای متفاوت از فضای ورودی ها می پردازند و در نهایت به پاسخ صحیح همگرا می شوند.

ج)

در فاز یادگیری شبکه، فضای ورودی X به اولین لایه شبکه وارد شده و خروجی هر لایه به ترتیب ورودی لایه بعد است. این مسیر تا لایه انتهایی ادامه یافته و اصطلاحا Forward Propagation نام دارد.

پس از طی نمودن Forward Propagation به تصحیح مقادیر وزن ها می پردازیم. با محاسبه نرخ تغیرات تابع هزینه(Gradiant Descent) جهت و مقدار تغیر وزن ها را طوری تعین می کنیم که به خروجی مطلوب نزدیک تر شود. به کل این فرآیند Back Propagation می گوییم.

همانطور که در بخش قبل اشاره شد به دلیل self differential بودن توابع فعال ساز در هر مرحله حرکت رو به عقب مشتق هر لایه را میتوان بر اساس خروجی آن لایه یافت و به ترتیب با بازگشت به لایه های قبل مشتق کامل شبکه را در اختیار داریم.

د)

نرخ یادگیری یک هایپرپارامتر است که بزرگ بودن آن سبب سریع بودن فرآیند یادگیری و کوچک بودن آن سبب کند بودن فرآیند می شود ولی در این بین باید به این trade off توجه کرد که بزرگ بودن نرخ می تواند سبب واگرایی از پاسخ اصلی و کوچک بودن آن سبب گیر افتادن در مینیمم های محلی شود.

ه)

افزایش لایه به صورت لگاریتمی نیاز به نورون ها در هر لایه را کاهش می دهد و در نتیجه ابعاد فضای مورد بررسی در هر لایه کمتر و مرتبه پیچیدگی محاسباتی به مراتب کمتر خواهد بود. همچنین افزودن لایه ها توانایی شبکه در تخمین ورودی های پیچیده تر را افزایش می دهد.

به صورت شهودی هر لایه باز نمایی جدیدی از ورودی های لایه قبل ارائه می دهد که در آن محاسبات را راحت تر می توان انجام داد.

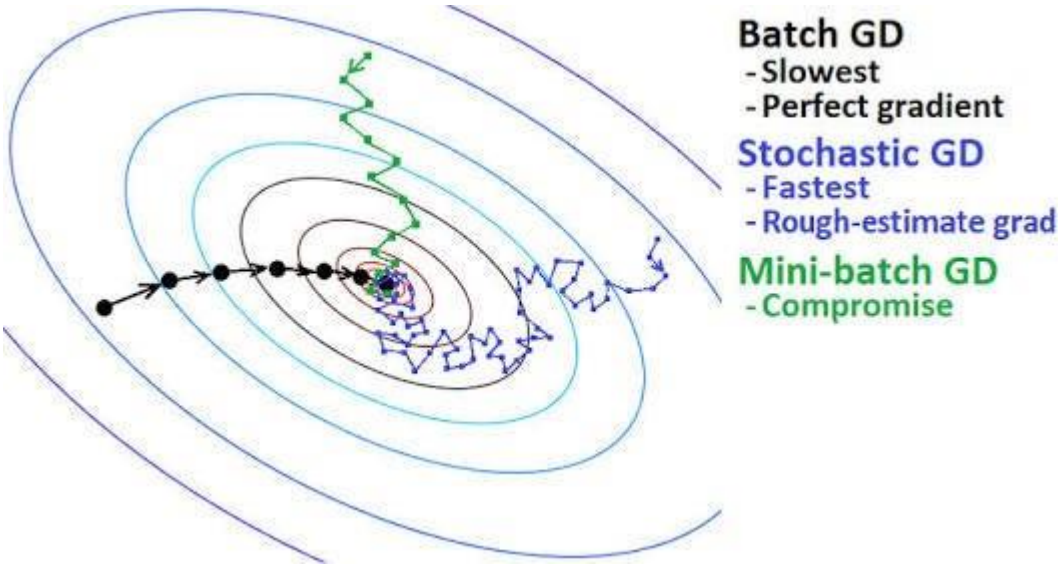
و)

مدل های شبکه عصبی در فرآیند یادگیری معمولا به حجم بالایی از داده برای آموزش نیاز دارند و این سبب می شود برای اصلاح مقادیر وزن ها هنگام Back Propagation پیچیدگی الگوریتم محاسباتی با توجه به تعداد داده ورودی بسیار بالا باشد.

برای حل این مشکل به نحوه محاسبه Gradiant Decent توجه می کنیم در حالت پیش فرض ما همه داده ها را یک جا محاسبه می کردیم که اصطلاحا Bach Gradiant است اما اگر از روش های زیر استفاده کینم سریع تر به کمینه مقدار تابع هزینه همگرا می شویم.

Mini Batch: در این روش کل n داده را به دسته های k تایی تقسیم و در هرمرحله از k تا جدید گرادیان گرفته و آن را آپدیت می کنیم.

Stochastic: در این روش در هر مرحله تنها گرادیان را به ازای یک داده محاسبه و وزن ها را اصلاح می کنیم. علی رغم غیربهینه بودن قدم ها در این روش در مقایسه با Batch اما این روش n برابر سریع تر از آن است.

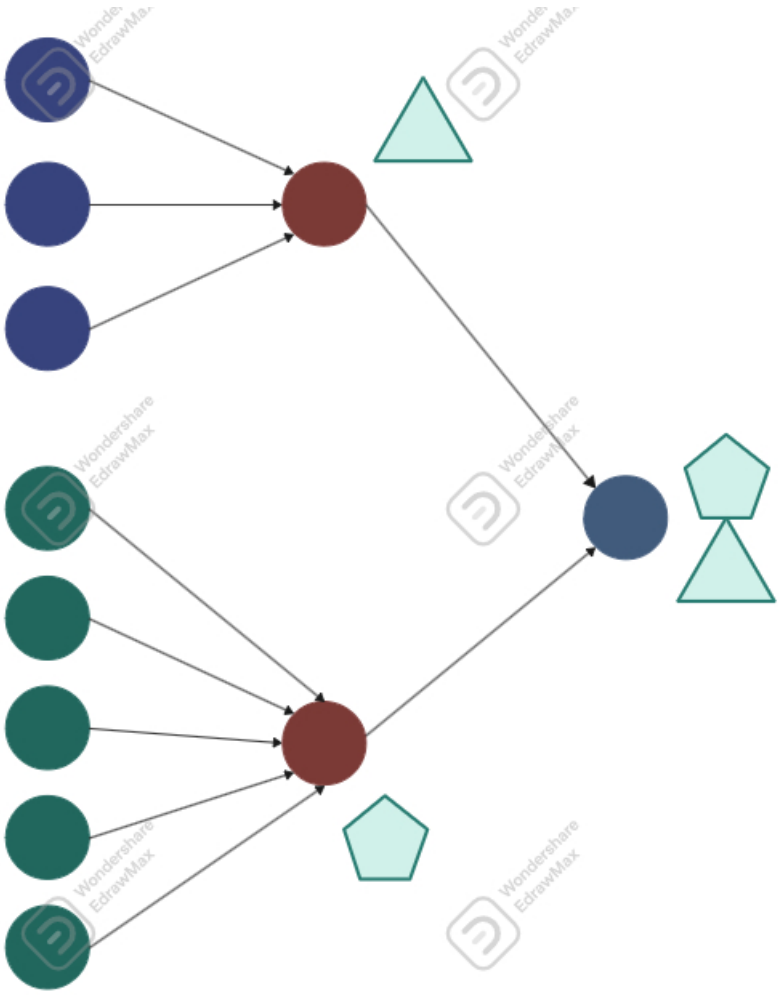


سوال 2

الف)

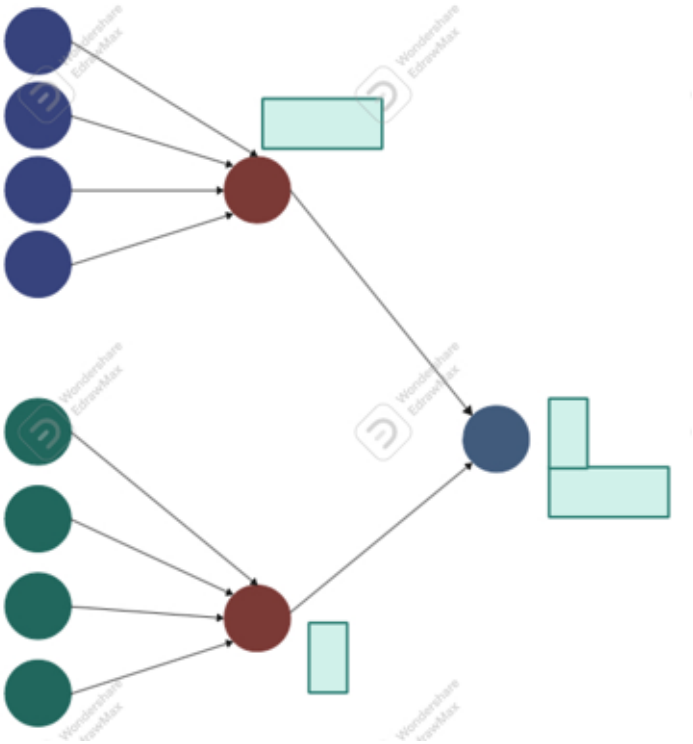
در هر لایه نورون ها محاسبات فضای خطی را انجام داده و مرز ها را می یابند سپس انتقال از هر لایه به لایه دیگر با تبدیلی غیرخطی فضا را تغیر می دهد.

در لایه اول برای مثلث به 3 نورون برای یافتن 3 مرز آن و برای 5 ضلعی به 5 نورون برای یافتن 5 مرز آن نیازمندیم. سپس در لایه بعد با اتصال مرز ها، مثلث و 5 ضلعی کامل شده و در آخرین لایه مثلث و 5 ضلعی به یکدیگر متصل می شوند.

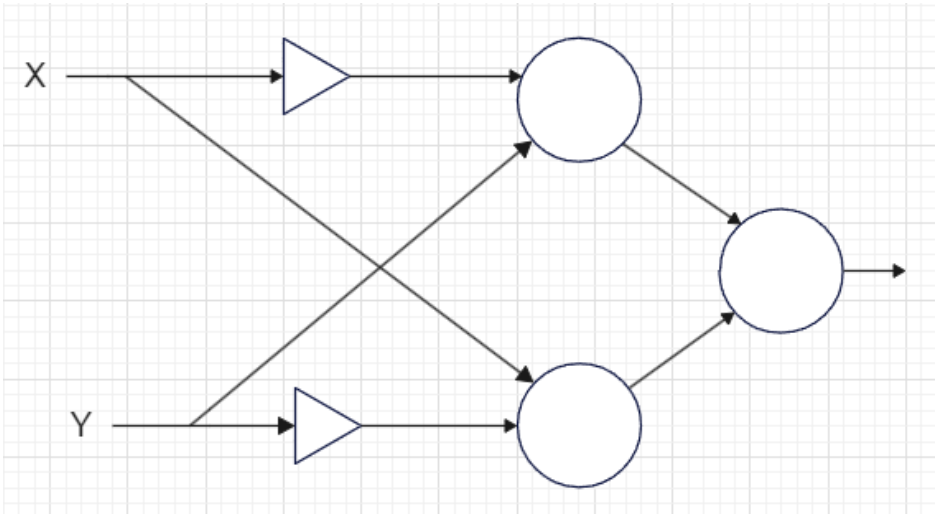


ب)

در لایه اول هر 4 نورون 4 مرز مستطیل را می یابد و سپس در لایه بعد 4 مرز با اتصال به هم مستطیل را کامل میکند و در آخرین لایه دو مستطیل به یکدیگر متصل می شوند.



ج)



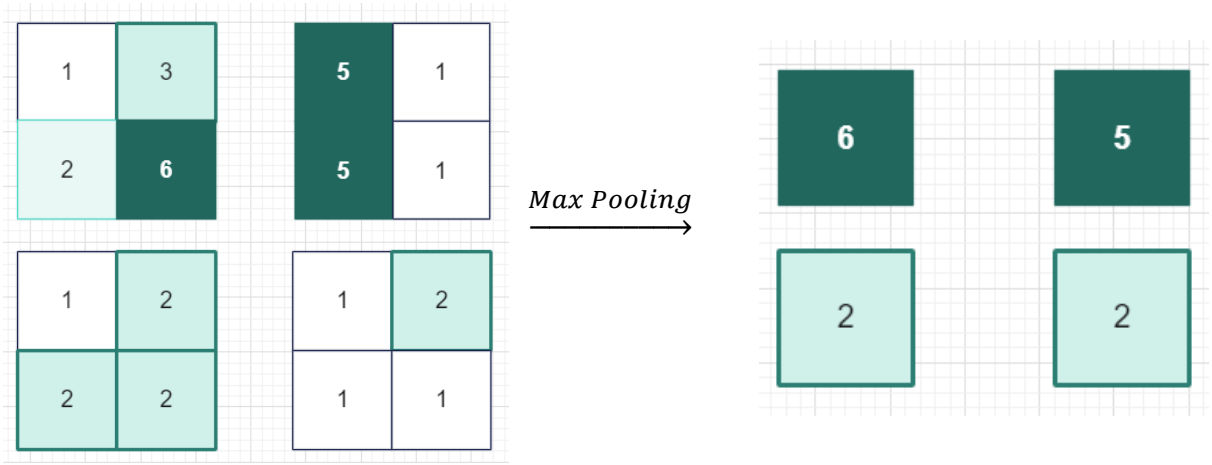
XOR فضای فیچر ها را به صورت غیر خطی تقسیم می کند بنابراین به دو لایه نیاز خواهیم داشت.

سوال 3

الف)

تفاوت بارز تصاویر کلاس اول نسبت به دوم این است که در کلاس اول فضای محصور ایجاد می کند.

ب)



ج)

با انتخاب مرزی بین 5 تا 2 (مثلا 4) می توان این دو کلاس را از هم جدا کرد.

سوال 4

الف)

توابع sigmoid و tanh به ترتیب به دلیل داشتن عملیات نمایی و تقسیم بار محاسباتی بالایی دارند در صورتی که relu صرفا مقادیر منفی را به واسطه max(x,0) صفر می کند.

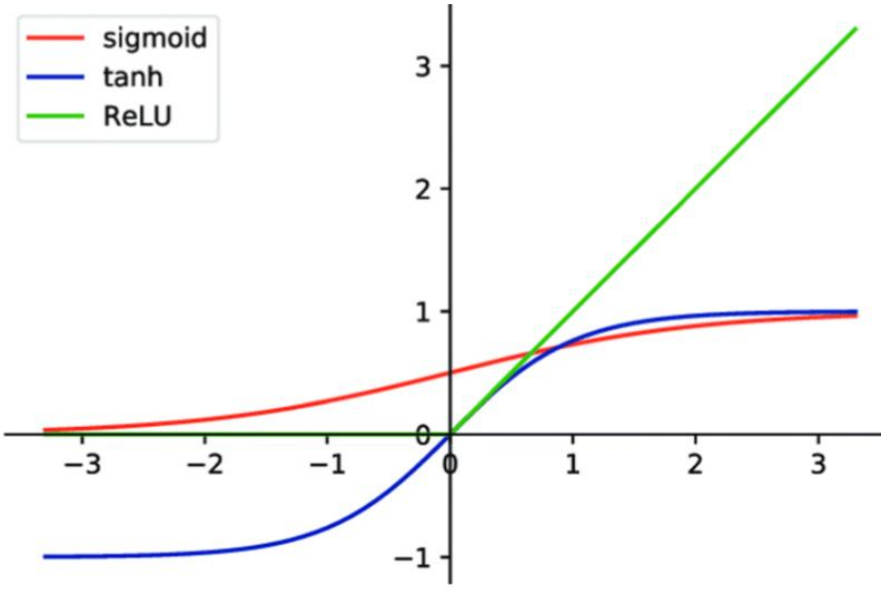
ب)

علت Vanishing Gradinat:

این مشکل زمانی اتفاق میافتد که گرادیان function error خیلی کوچک میشود و این باعث میشود که یادگیری شبکه سخت شود .این اتفاق به علت differentiation of rule chain در backpropagationرخ میدهد، به این صورت که در هر لایه گرادیانها ضرب میشوند و اگر گرادیان خیلی کوچک باشد (بین صفر و یک) در هر ضرب شدن کوچکتر نیز میشود و اینگونه است که در لایه های آخر backpropagationمیتونه به سمت صفر شدن برود و این باعث میشود که در لایههای ابتدایی یادگیری شبکه یا خیلی آهسته شود و یا اصلا نتواند یاد بگیرد .یکی از دلایلی که میتواند باعث رخ دادن این اتفاق بشود استفاده از function activation نامناسب میباشد.

علت این که relu به Vanishing Gradiant دچار نمی شود این است که مشتق خروجی آن 0 یا 1 است و در نتیجه هرچه جلوتر میرویم مقادیر کوچک تر نمی شوند.

علت این که sigmoid و tanh دچار Vanishing Gradiant می شوند: برای توضیح این مشکل در ابتدا باید function saturation را توضیح دهیم .به توابعی که ماکزیمم و مینیمم آنها خیلی خیلی بزرگ و خیلی خیلی کوچک است، function saturationمیگوییم .جفت توابع sigmoid و tanh یک function saturationهستند. (با توجه عکس نمودار sigmoid و tanh نیز میتوان این گزاره را تایید کرد.)



Saturation function ها خیلی راحت میتوانند دچار vanishing gradiant شوند زیرا مثلا برای مقادیر کوچک پس از رد شدن از function activation های مثل sigmoid و tanh به ما مقادیر خیلی کوچکتر ممکن است بدهند که منجر به vanishing gradiant میشود.

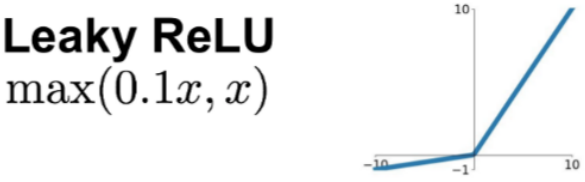
ج)

مسائل طبقه بندی اغلب باینری است و در نتیجه استفاده از sigmoid می تواند خروجی را بین 0 تا 1 کد کند که حکم احتمال تعلق به کلاس را نشان دهد.

در صورتی که مسئله چند کلاسه باشد استفاده از Relu به دلیل اینکه خروجی را از 0 تا بی نهایت کد می کند می تواند کارا باشد ولی به دلیل یادگیری سخت آن شاید استفاده از روش های دیگر مانند1Vs. All مفید تر باشد.

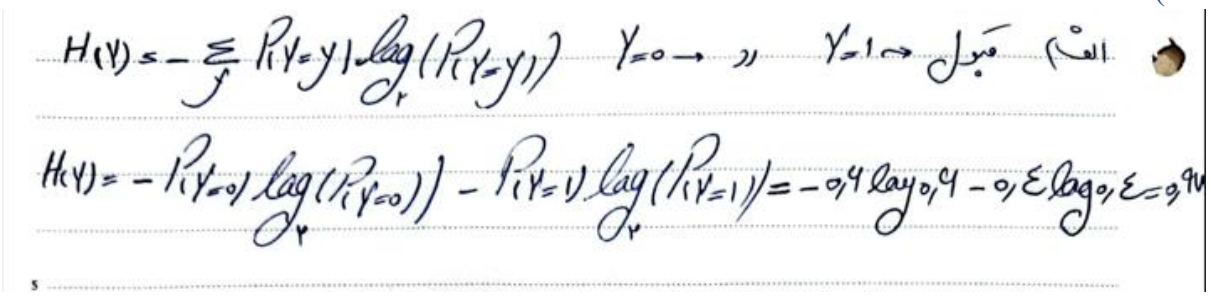
د)

در Leaky relu گرادیان ورودی های منفی دیگر صفر نیست بلکه مقدار کوچکی است که مانع مشکل Dying Relu می شود به همین دلیل هنگامی که احتمال بروز ورودی های منفی بالاست Leaky Relu را به Relu ترجیح می دهیم.

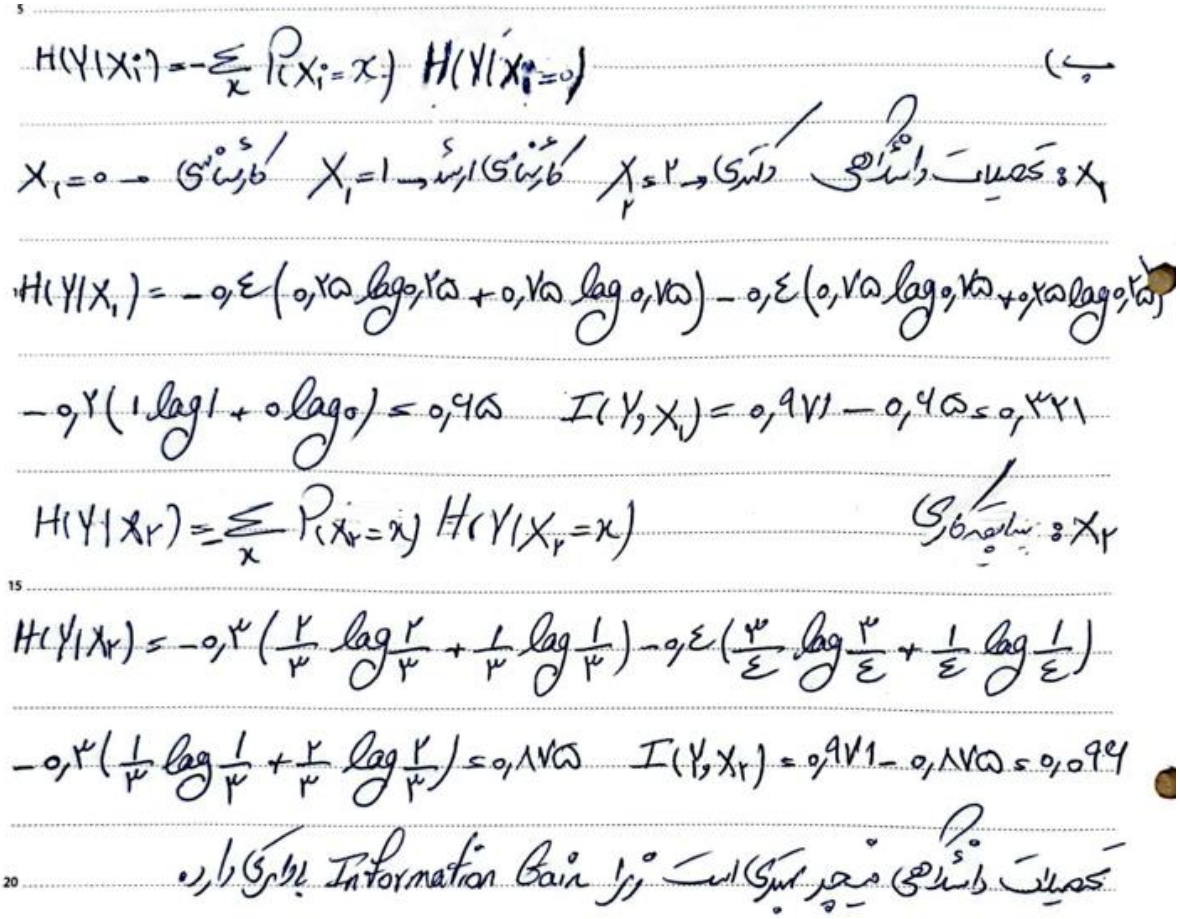


سوال 5

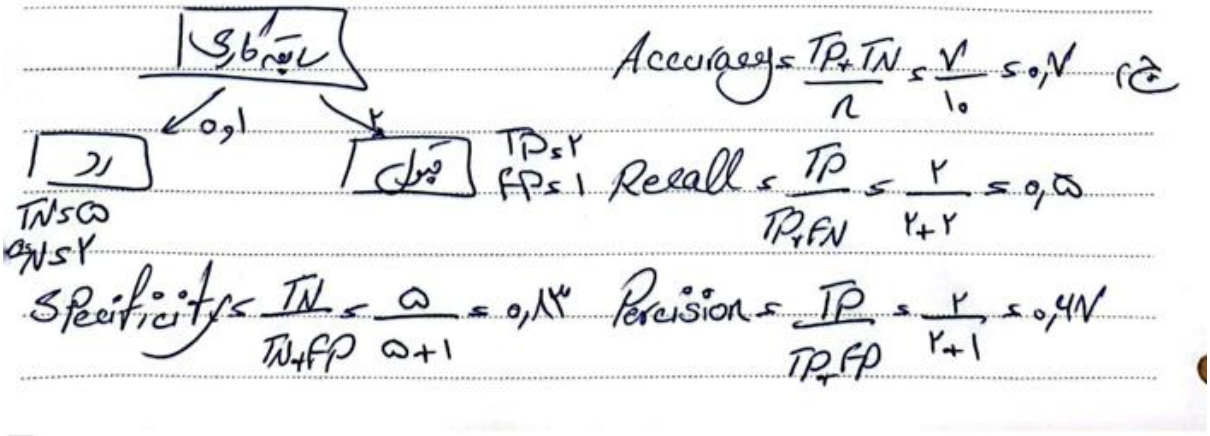
الف)


الف) قبول ← Y=1 ، Y=0 ← رد
$H(Y) = -\sum_y P(Y=y)\log_2(P(Y=y))$
$H(Y) = -P(Y=0)\log_2(P(Y=0)) - P(Y=1)\log_2(P(Y=1)) = -0,4\log 0,4 - 0,6\log 0,6 = 0,97$

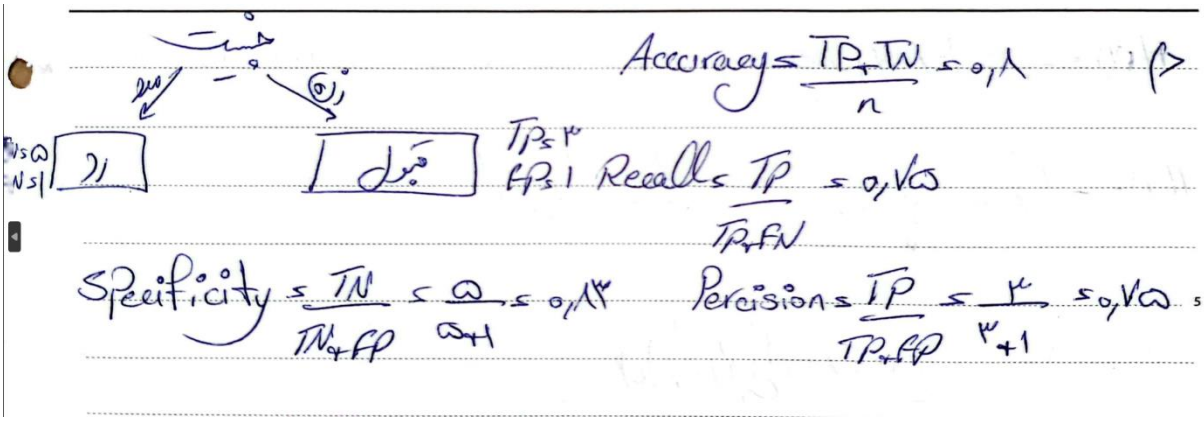
ب)


ب)
$H(Y|X_i) = -\sum_x P(X_i=x)\, H(Y|X_i=x)$
X_1: تحصیلات دانشگاهی، $X_1=0$ ← کارشناسی، $X_1=1$ ← کارشناسی ارشد، $X_2=2$ ← دکتری
$H(Y|X_1) = -0,5(0,25\log 0,25 + 0,75\log 0,75) - 0,5(0,75\log 0,75 + 0,25\log 0,25)$
$-0,2(1\log 1 + 0\log 0) = 0,75$ $I(Y,X_1) = 0,971 - 0,75 = 0,221$
X_2: سابقه کاری
$H(Y|X_2) = \sum_x P(X_2=x)\, H(Y|X_2=x)$
$H(Y|X_2) = -0,3(\frac{2}{3}\log\frac{2}{3} + \frac{1}{3}\log\frac{1}{3}) - 0,5(\frac{3}{5}\log\frac{3}{5} + \frac{2}{5}\log\frac{2}{5})$
$-0,2(\frac{1}{2}\log\frac{1}{2} + \frac{1}{2}\log\frac{1}{2}) = 0,875$ $I(Y,X_2) = 0,971 - 0,875 = 0,096$
تحصیلات دانشگاهی فیچر بهتری است زیرا Information Gain بالاتری دارد.

ج)


بیمار
۰,۱
۲
بله
خیر
TP=۲
FP=۱
TN=۳
FN=۲
Accuracy = TP+TN / n = ۷/۱۰ = ۰,۷
Recall = TP / TP+FN = ۲/۲+۲ = ۰,۵
Specificity = TN / TN+FP = ۳/۳+۱ = ۰,۷۵
Percision = TP / TP+FP = ۲/۲+۱ = ۰,۶۷

د)


خیر
بله
بله
خیر
TP=۳
FP=۱
TN=۳
FN=۱
Accuracy = TP+TN / n = ۰,۸
Recall = TP / TP+FN = ۰,۷۵
Specificity = TN / TN+FP = ۳/۳+۱ = ۰,۷۵
Percision = TP / TP+FP = ۳/۳+۱ = ۰,۷۵

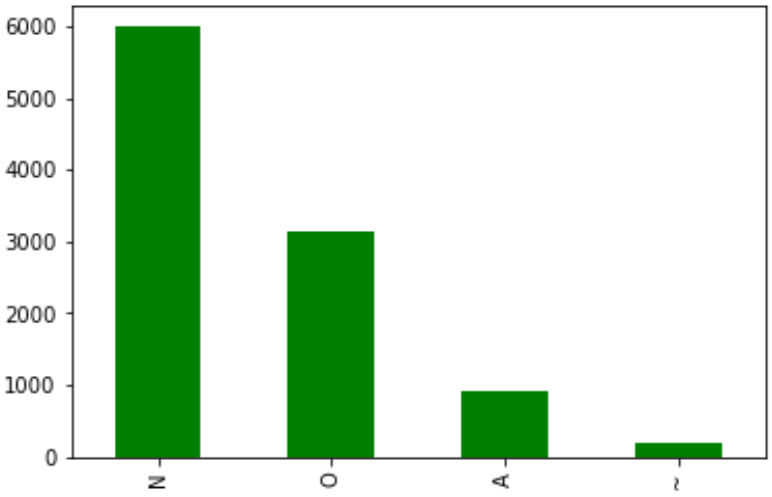
سوال 6

الف)

توزیع هر کلاس به شرح روبه رو است:

```
N    5992
O    3151
A     923
~     187
Name: label, dtype: int64
```

ب)

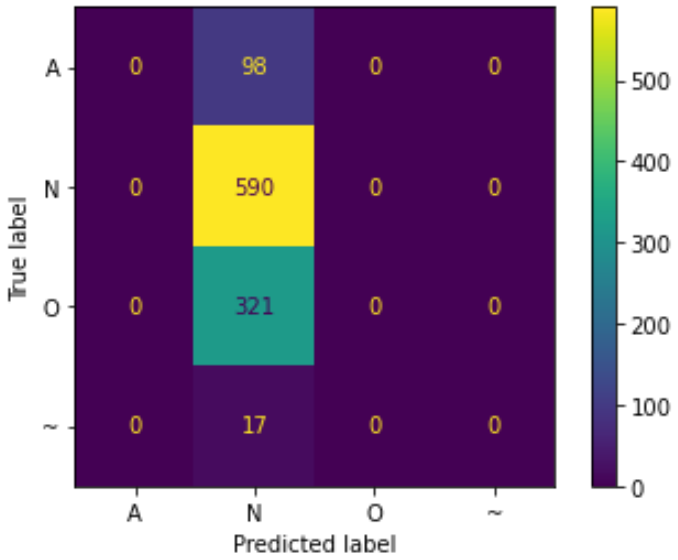


علی رغم اینکه عدم توازن در مسائل واقعی طبیعی است ولی در اینجا توزیع کلاس ها کاملا نا متوازن است به نحوی که در عمل مدل شبکه عصبی فقط یک کلاس و با یک سری بهبودات نهایتا دو کلاس را یاد میگیرد و هر داده وروی را به این دو لیبل می زند.

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

مطابق این منبع در سطح micro مقدار total precision, total recall, total f1-score با هم برابر و برابر با accuaracy آن ها است، در نتیجه مطابق جدول بالا دقت کل برابر است با 0.58



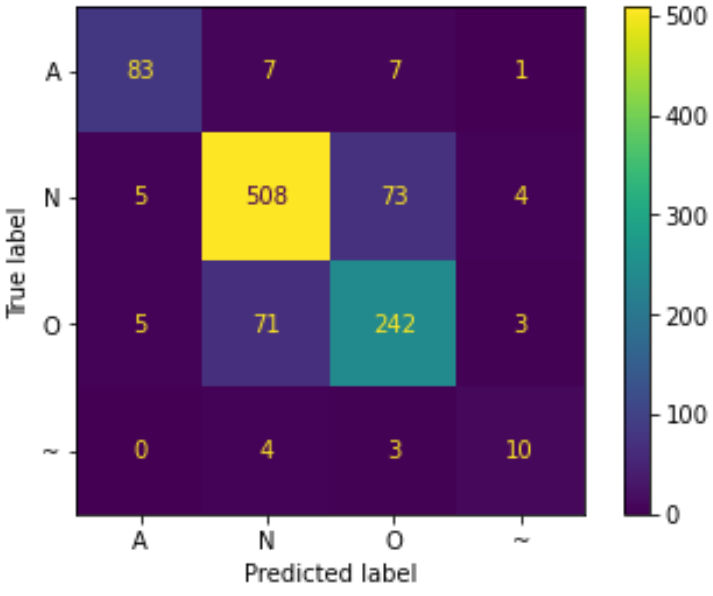
همانطور که مشاهده می شود مدل تمامی ورودی ها را به کلاس N طبقه بندی می کند که ناشی از عدم توازن دیتاست و بایاس شدید آن به کلاس N است.

همچنین عدم نرمالایز کردن فیچر ها مزید برعلت شده است.

د)

	precision	recall	f1-score	support
A	0.89	0.85	0.87	98
N	0.86	0.86	0.86	590
O	0.74	0.75	0.75	321
~	0.56	0.59	0.57	17
accuracy			0.82	1026
macro avg	0.76	0.76	0.76	1026
weighted avg	0.82	0.82	0.82	1026

Total Precision = 0.82



همانطور که مشاهده می کنید مقادیر precision,recall,f1-score پس از نرمالایز فیچر ها به مراتب بهبود یافته است.

همچنین مدل در طبقه بندی دیگر کلاس ها نیز توانا است.

ه)

بازه بعضی فیچر ها در مرتبه صدم و بازه بعضی دیگر فیچر ها در مرتبه 100 تا 1000 است به همین دلیل هنگامی که فیچر ها نرمالایز نشدند فیچر ها در مرتبه کوچک در مقابل مرتبه های بزرگ تر ارزش خود را از دست می دهند و در عمل طبقه بند برای تصمیم گیری به آنها توجه نمی کند.

هنگامی که فیچر ها نرمالایز می شوند به بازه ای مشخص و یکسان نگاشت می شوند که سبب می شود فیچر ها نسبت به هم از ارزش یکانی برخوردار باشند همچنین هزینه های محاسباتی نیز به مراتب در اعداد کوچکتر کاهش می یابد، این موضوع مخصوصا در فرآیند back propagation تاثیر خود را می گذارد.

نرمالایز به روش استاندارد 1 :

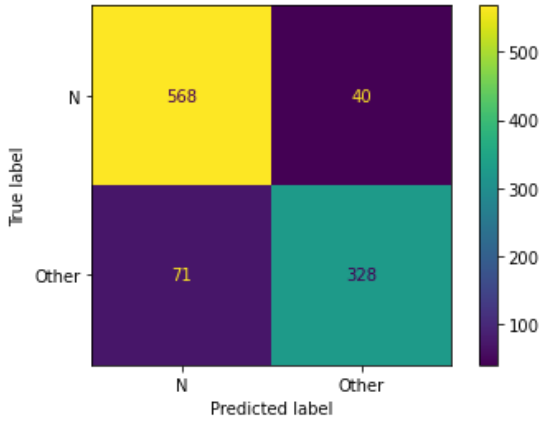
$$mean = \frac{1}{n_{train}} \sum X_{train}$$

$$std = \sqrt{\frac{1}{n_{train}} \sum (X_{train} - mean)^2}$$

$$X_{scaled} = \frac{X - mean}{std}$$

و)

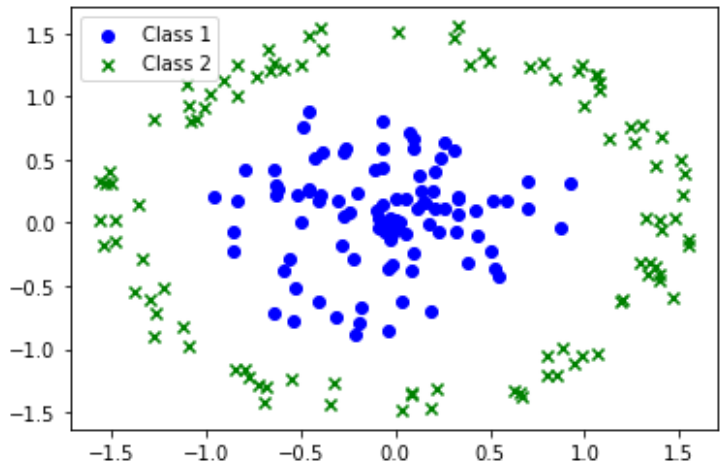
	precision	recall	f1-score	support
N	0.89	0.93	0.91	608
Other	0.89	0.82	0.86	399
accuracy			0.89	1007
macro avg	0.89	0.88	0.88	1007
weighted avg	0.89	0.89	0.89	1007



حذف کلاس با تعداد سمپل ناچیز و ترکیب کلاس های مغلوب سبب می شود تا حدودی عدم توازن دیتاست بهبود یابد و عملکرد مدل بهبود یابد.

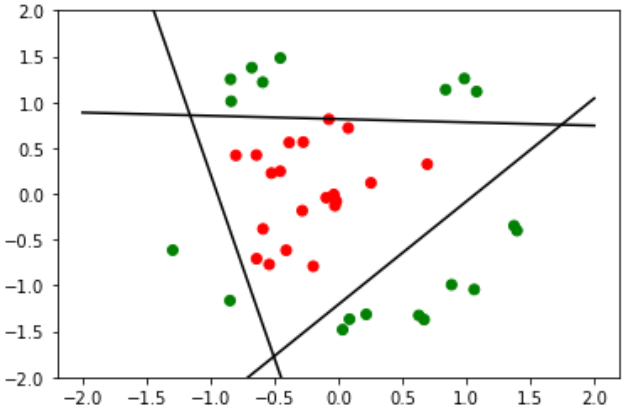
سوال 7

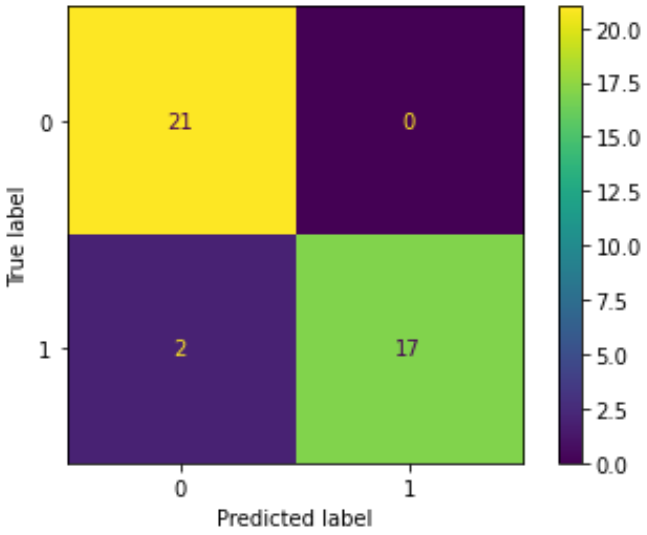
الف)



ب)

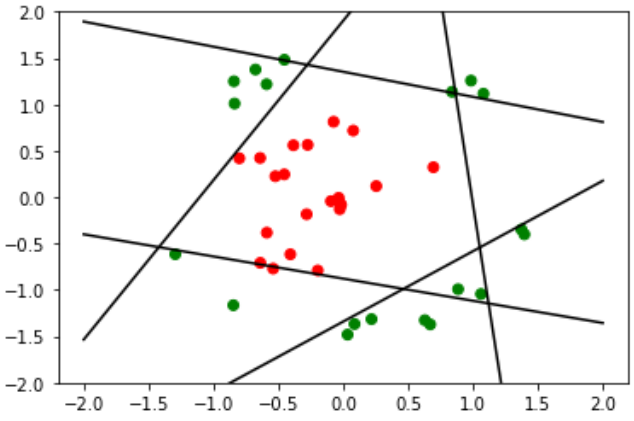
مرز بندی داده تست مطابق تصویر روبه رو می باشد.



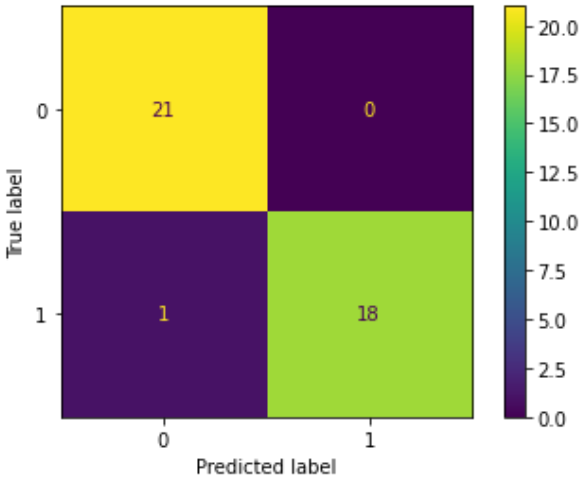


	precision	recall	f1-score	support
0.0	0.95	1.00	0.98	21
1.0	1.00	0.95	0.97	19
accuracy			0.97	40
macro avg	0.98	0.97	0.97	40
weighted avg	0.98	0.97	0.97	40

ج)



	precision	recall	f1-score	support
0.0	0.91	1.00	0.95	21
1.0	1.00	0.89	0.94	19
accuracy			0.95	40
macro avg	0.96	0.95	0.95	40
weighted avg	0.95	0.95	0.95	40



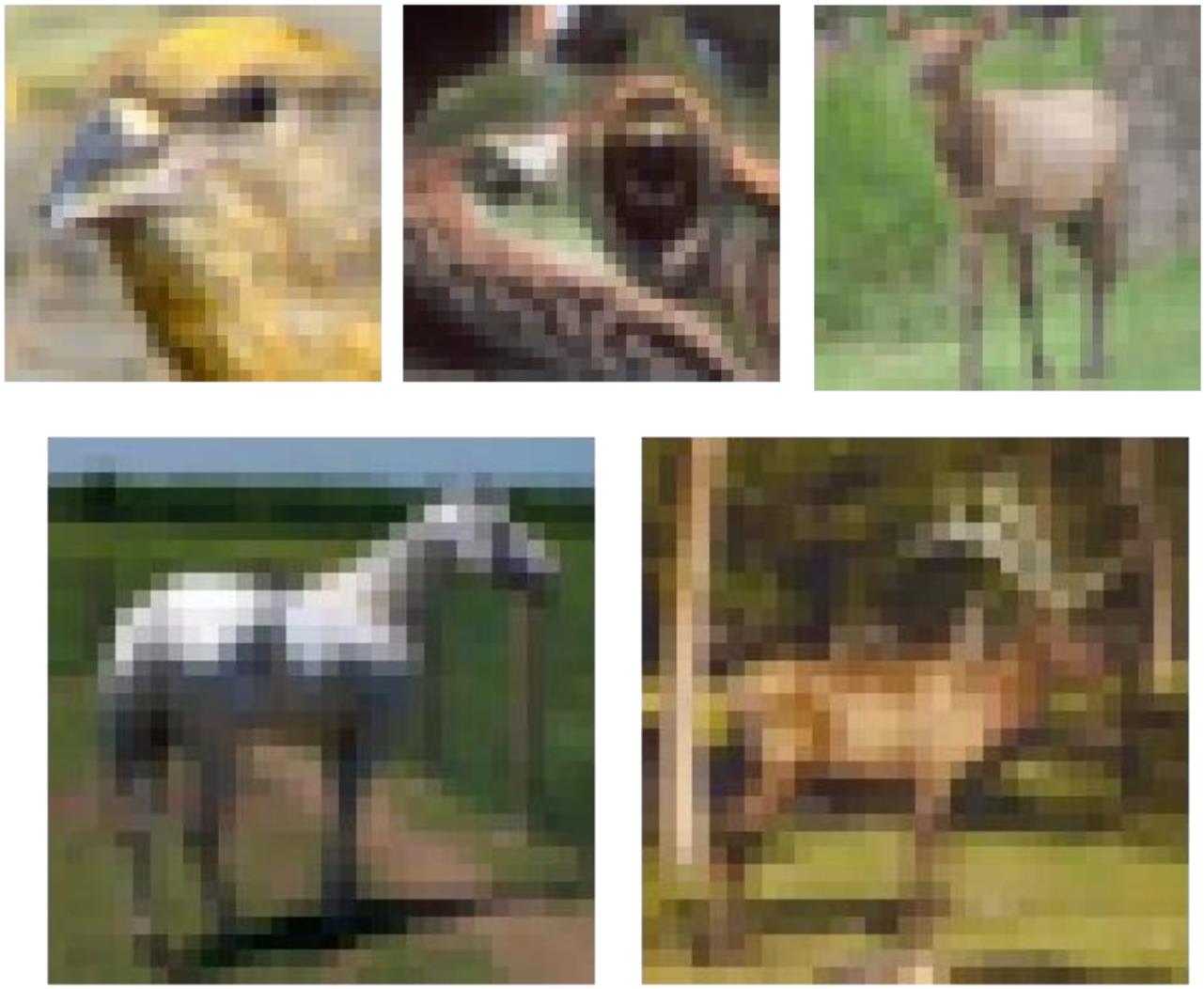
د)

به ازای 6 لایه میانی مدل عملکرد نسبتا بهتری نسبت به 3 لایه میانی داشته است یعنی فرم توزیع داده ها با 6 خط بهتر افراز می شود که با توجه به توزیع دایروی داده ها قابل پیش بینی بود.

سوال 8

الف)

5 تصویر رندوم از دیتاست cifar10:



ب)

3.2 Part B

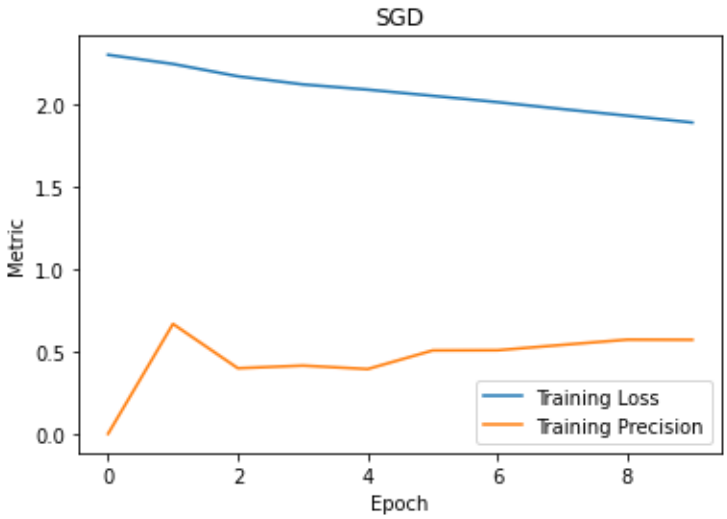
```
X_train = X_train.astype("float32") / 255.0
X_test = X_test.astype("float32") / 255.0
Y_train = to_categorical(Y_train)
Y_test = to_categorical(Y_test)

x_train, x_valid, y_train, y_valid = train_test_split(X_train, Y_train, test_size=0.8, random_state=42)
```

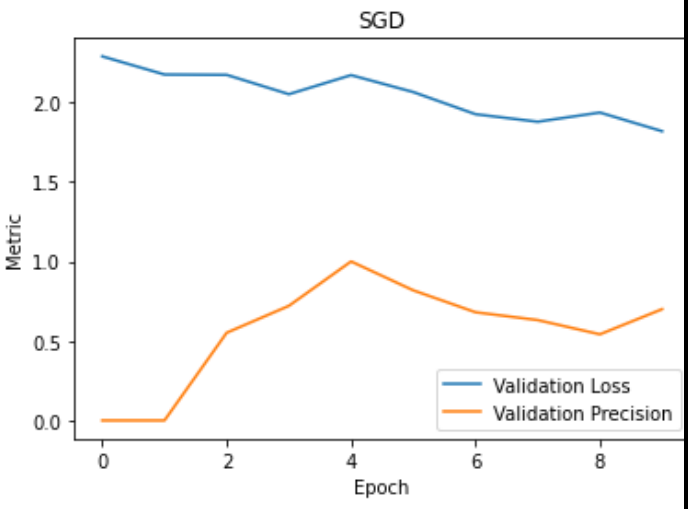
ج)

لایه های شبکه مطابق خواسته سوال:

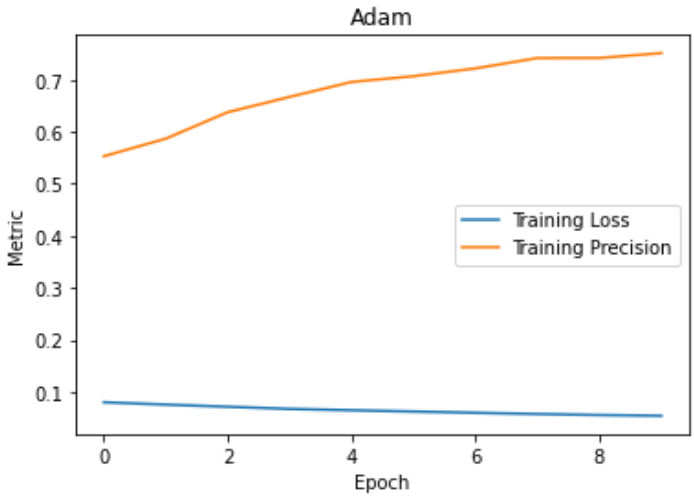
```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 30, 30, 32)        896
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 28, 28, 32)        9248
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 26, 26, 32)        9248
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 13, 13, 32)        0
_________________________________________________________________
dropout (Dropout)            (None, 13, 13, 32)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 11, 11, 64)        18496
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 9, 9, 64)          36928
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 7, 7, 64)          36928
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 3, 3, 64)          0
_________________________________________________________________
dropout_1 (Dropout)          (None, 3, 3, 64)          0
_________________________________________________________________
flatten (Flatten)            (None, 576)               0
_________________________________________________________________
dense (Dense)                (None, 512)               295424
_________________________________________________________________
dropout_2 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_1 (Dense)              (None, 10)                5130
=================================================================
Total params: 412,298
Trainable params: 412,298
Non-trainable params: 0
```

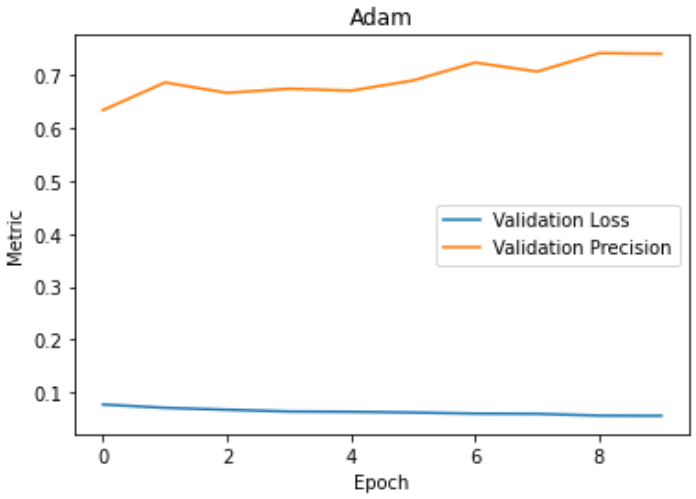
SGD:



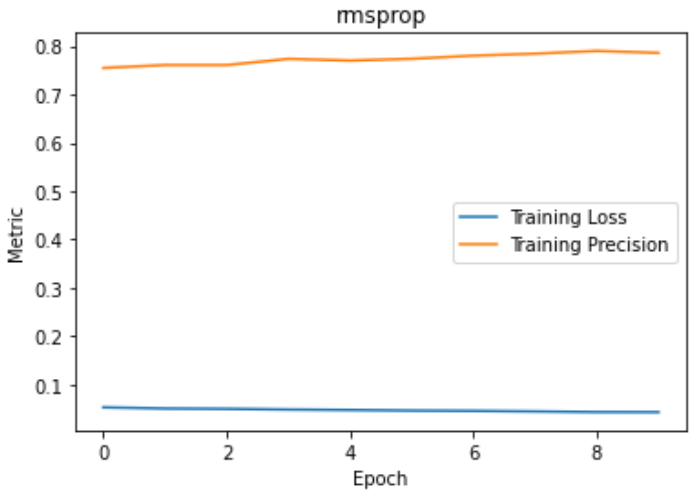


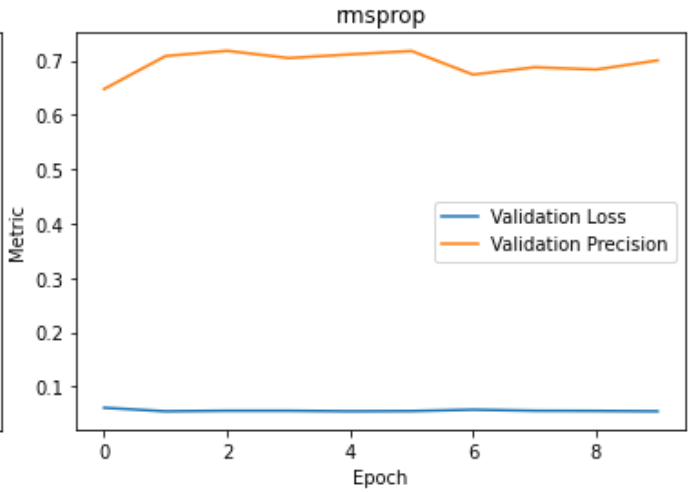
Adam:





Rmsprop:





(د

SGD:

	precision	recall	f1-score	support
0	0.41	0.43	0.42	1000
1	0.36	0.63	0.46	1000
2	0.37	0.07	0.11	1000
3	0.30	0.06	0.09	1000
4	0.19	0.04	0.07	1000
5	0.35	0.24	0.29	1000
6	0.28	0.69	0.40	1000
7	0.29	0.41	0.34	1000
8	0.37	0.41	0.39	1000
9	0.38	0.39	0.39	1000
accuracy			0.34	10000
macro avg	0.33	0.34	0.30	10000
weighted avg	0.33	0.34	0.30	10000

Adam:

	precision	recall	f1-score	support
0	0.55	0.59	0.57	1000
1	0.66	0.79	0.72	1000
2	0.42	0.43	0.42	1000
3	0.38	0.31	0.34	1000
4	0.53	0.27	0.36	1000
5	0.46	0.58	0.51	1000
6	0.64	0.69	0.67	1000
7	0.65	0.60	0.63	1000
8	0.61	0.73	0.67	1000
9	0.66	0.61	0.63	1000
accuracy			0.56	10000
macro avg	0.56	0.56	0.55	10000
weighted avg	0.56	0.56	0.55	10000

Rmsprop:

	precision	recall	f1-score	support
0	0.66	0.56	0.61	1000
1	0.71	0.78	0.74	1000
2	0.54	0.40	0.46	1000
3	0.43	0.34	0.38	1000
4	0.43	0.57	0.49	1000
5	0.54	0.55	0.54	1000
6	0.60	0.76	0.67	1000
7	0.69	0.60	0.64	1000
8	0.76	0.65	0.70	1000
9	0.61	0.73	0.66	1000
accuracy			0.59	10000
macro avg	0.60	0.59	0.59	10000
weighted avg	0.60	0.59	0.59	10000

سوال 9

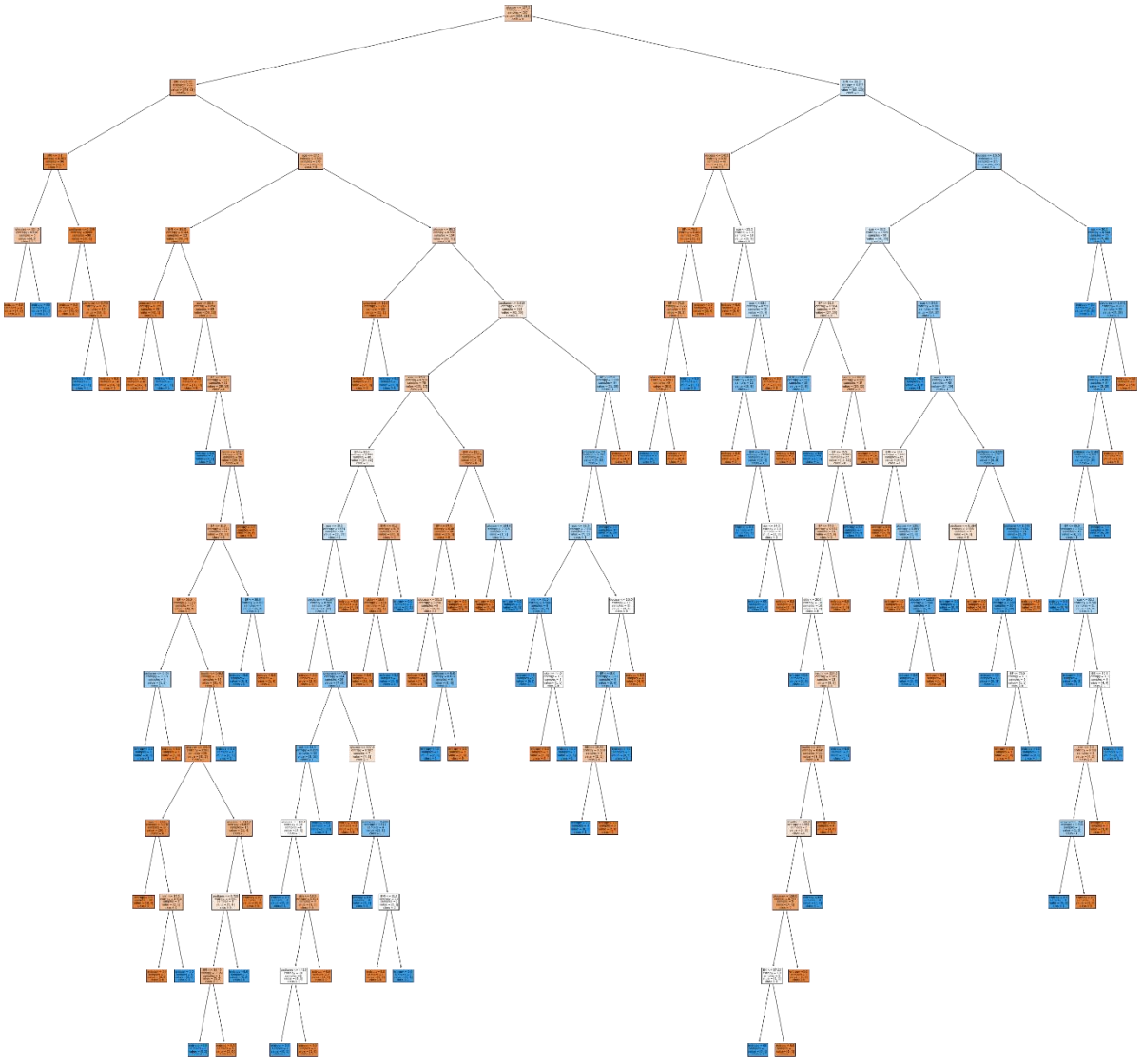
الف)

```
print("Percision for test data = "+str(round(precision_score(Y_test,y_pred_test)*100,3))+"%")
print("Percision for train data = "+str(precision_score(Y_train,y_pred_train)*100)+"%")
```

```
Percision for test data = 66.279%
Percision for train data = 100.0%
```

در این حالت عمق درخت تا جایی ادامه می یابد که داده ها در برگ ها به صورت کاملا خالص تقسیم بندی شود. همانطور که مشاهده می شود با توجه به دقت نسبتا پایین 66 درصدی برای داده تست اما دقت بالای 100 درصدی برا داده آموزش، مدل دچار بیش برازش(Overfitting) شده است که مطلوب نیست.

ب)



* اگر شرط درست به سمت چپ و اگر غلط به راست حرکت می کند.

ج)

مدل های تصمیم گیری درختی به طور کلی مستعد بیش برازش(Overfitting) می باشند برای جلوگیری از این مشکل از تکنیک pruning استفاده می شود.

$$Pruning \rightarrow \begin{cases} Pre\ Pruning/Hyperparameter\ Pruning \\ Post\ Pruning/Backward\ Pruning \end{cases}$$

در این سوال به بررسی Pre Pruning می پردازیم ولی برای مطالعه بیشتر در موردPost Pruning توصیه می شود به این منبع مراجعه نمایید.

از pre pruning برای مقابله با بیش برازش قبل از ساختن درخت استفاده می شود. به طور کلی قبل از ساختن درخت ما اطلاعی از مقدار یا حالت بهینه برای Hyperparameter هایی از جمله max_depth، min_sample_split، min_samples_leaf نداریم.

با استفاده از دادهvalidation می توانیم مقدار بهینه برای این هایپرپارامتر ها را تعین کرد تا درخت دچار بیش برازش نشود.

د)

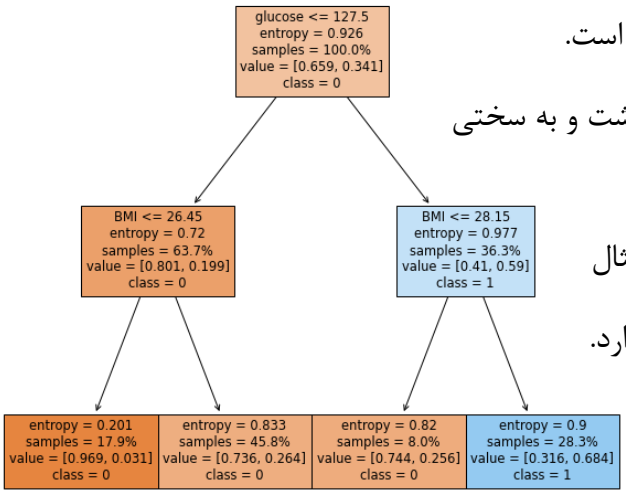
```
print("Percision for test data = "+str(round(precision_score(Y_test,y_pred_test)*100,3))+"%")
print("Percision for train data = "+str(round(precision_score(Y_train,y_pred_train)*100,3))+"%")
```

```
Percision for test data = 71.053%
Percision for train data = 68.421%
```

علی رغم کاهش دقت داده آموزش همانطور که مشاهده می شود دقت داده تست افزایش یافته است یعنی مدل دیگر دچار بیش برازش نیست و به عمق بهینه نزدیک شده است.

باید به این نکته توجه داشت که عمق کمتر هم ممکن است مدل را دچار underfitting کند، در این مثال هم دقت 71 درصدی دقت قابل قبولی است ولی به ازای عمق برابر 4 به حداکثر دقت خود می رسد.

ه)



تفسیر این درخت به مراتب نسبت به درخت قبلی راحت تر است.

درخت بخش ب به دلیل عمیق بودن آن تحلیل دشواری داشت و به سختی می توان از آن اطلاعات معنا داری یافت درحالی که در این درخت با نگاهی گذرا اطلاعات معنی داری یافت به عنوان مثال بالا بودن گلوکز و BMI رابطه مستقیمی با احتمال دیابت دارد.