به نام خدا

دانشگاه تهران
دانشکده مهندسی برق و کامپیوتر

شبکه های عصبی و یادگیری عمیق

تمرین سری دوم

نام و نام خانوادگی	امیر محمد رنجبر پازکی
شماره دانشجویی	۸۱۰۱۹۹۳۴۰
تاریخ ارسال گزارش	۱۴۰۰/۲/۴

فهرست گزارش سوالات

سوال ۱ – MLP (Regression)

الف) ابتدا داده‌ی مورد نظر با استفاده از کتابخانه pandas خوانده شد و با استفاده از describe و info اطلاعات کلی آن مورد بررسی قرار گرفت. در این بررسی تعداد ستون حاوی null شناسایی شدند. با مراجعه به توضیحات داده مشخص شد، null در این ستون‌ها معنی‌دار است. (به معنای عدم وجود) این ستون‌ها در یک آرایه نوشته‌شدند تا مقدار null آن‌ها با مقدار 'NO' جایگزین شود چرا که این مقادیر یک دسته تازه را در این ستون‌ها تشکیل می‌دادند. پس از این کار تعداد null ستون‌ها به شکل زیر درآمد.

```
Columns nulls count:
LotFrontage    259
GarageYrBlt     81
MasVnrArea       8
Electrical       1
SalePrice        0
ExterCond        0
RoofStyle        0
RoofMatl         0
Exterior1st      0
Exterior2nd      0
dtype: int64
```

شکل ۱ – تعداد **null** ستون‌ها بعد از جایگذاری **null**های معنادار

سپس، میزان missing value در هر ستون به درصد شناسایی شد و خروجی به شکل زیر بود.

```
LotFrontage    17.739726
GarageYrBlt     5.547945
MasVnrArea      0.547945
Electrical      0.068493
SalePrice       0.000000
ExterCond       0.000000
RoofStyle       0.000000
RoofMatl        0.000000
Exterior1st     0.000000
Exterior2nd     0.000000
dtype: float64
```

شکل ۲ – درصد **null** در هر ستون

ستون‌هایی که بیش از ۵ درصد missing value داشتند حذف شدند زیرا اطلاعات آن‌ها بسیار ناقص بود.

```
Columns nulls count:
MasVnrArea     8
Electrical     1
SalePrice      0
Foundation     0
RoofMatl       0
Exterior1st    0
Exterior2nd    0
MasVnrType     0
ExterQual      0
ExterCond      0
dtype: int64
```

شکل ۳ – تعداد **null** ستون‌ها پس از حذف ستون‌ها با **null** زیاد

ستون‌های عددی که باقی ماندند با استفاده از میانه پر شدند زیرا میانه نسبت به outlier حساس نیست و عدد پرتی به این مقادیر نمی‌دهد.

```
Columns nulls count:
Electrical     1
SalePrice      0
Foundation     0
RoofMatl       0
Exterior1st    0
Exterior2nd    0
MasVnrType     0
MasVnrArea     0
ExterQual      0
ExterCond      0
dtype: int64
```

شکل ۴ – تعداد **null** ستون‌ها پس از جایگزینی مقادیر خالی عددی

مقادیر خالی ستون‌های categorical با استفاده از مد یا همان بیشترین دسته پر شدند.

```
Columns nulls count:
SalePrice      0
Foundation     0
RoofMatl       0
Exterior1st    0
Exterior2nd    0
MasVnrType     0
MasVnrArea     0
ExterQual      0
ExterCond      0
BsmtQual       0
dtype: int64
```

شکل ۵ – تعداد **null** ستون‌ها پس از جایگزینی مقادیر **null** ستون‌های دسته‌ای

ویژگی هدف به صورت لگاریتمی درآمد. معمولا برای پیش‌بینی کردن بهتر قیمت از لگاریتم آن استفاده می‌کنند که برگشت‌پذیر است.

همچنین، ویژگی‌های عددی نیز نرمالیزه شدند. (mean normalization: میانگین صفر و انحراف معیار ۱) ویژگی‌های categorical نیز به صورت one hot به چند ویژگی تقسیم شدند تا بتوانیم از آن‌ها در regression استفاده کنیم. دلیل استفاده از one hot encoding آن است که طبقه‌های این ویژگی‌ها ترتیبی ندارند و هر کدام یک ویژگی محسوب می‌شوند.

در انتها نیز ستون Id که معنایی ندارد از داده‌ها حذف شد.

ب) در تابع get_train_test_loaders داده‌ها به نسبت ۸۰ به ۲۰ بین آموزش و تست تقسیم شده‌اند. همچنین، تبدیل به dataloader شده‌اند تا بتوانند در آموزش شبکه‌ی عصبی مورد استفاده قرار بگیرند. در این dataloader ها داده‌ها به صورت ۱۲۸ تایی دسته‌بندی شده‌اند.

برای این سوال یک کلاس Net نوشته‌شده‌است که در constructor داده‌ها تعداد لایه‌ها، سایز هر لایه و تابع فعالساز لایه‌ها را می‌گیرد و شبکه‌ی مورد نظر را خروجی می‌دهد. این شبکه‌ها یک dropout ده درصدی نیز دارند که از overfitting آن جلوگیری کند.

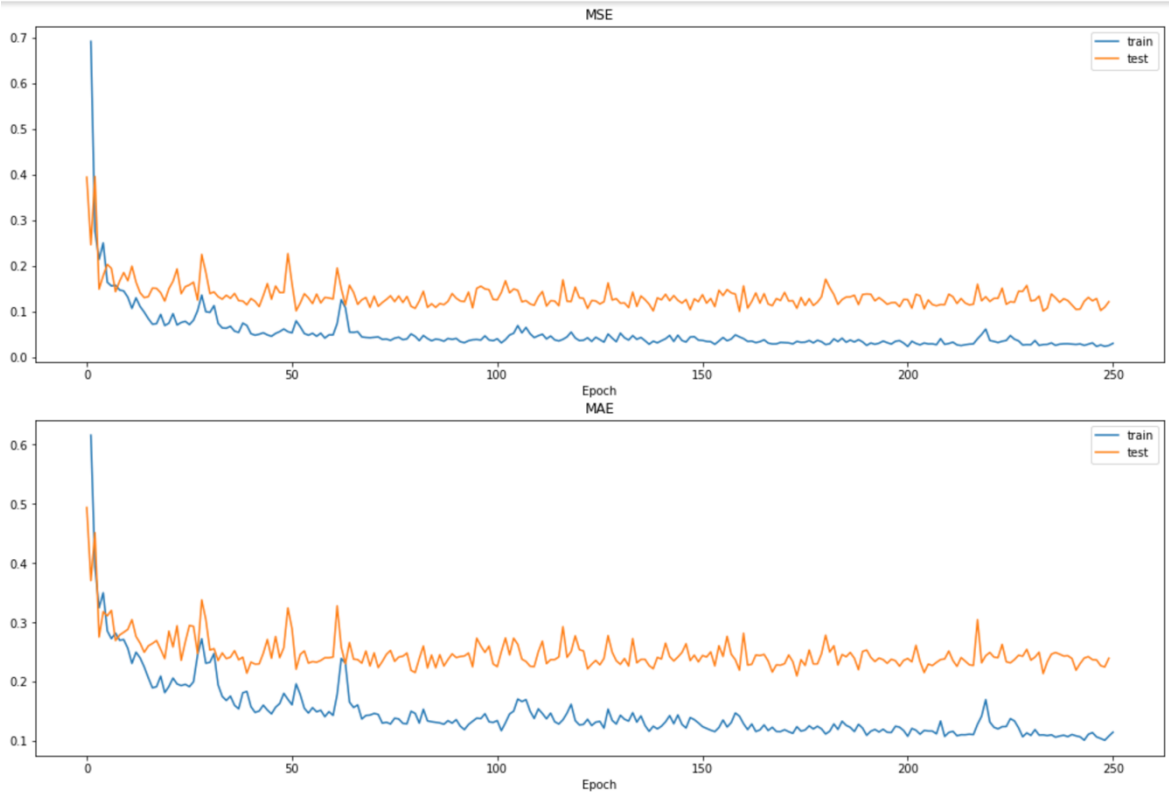
برای آموزش و تست شبکه‌ها نیز یک تابع train_and_test نوشته‌شده‌است که با گرفتن شبکه و داده‌ها آن را آموزش می‌دهد و مورد آزمون قرار می‌دهد و میزان loss را در هر epoch خروجی می‌دهد.

دو تابع نیز برای رسم loss و معیار خطای دیگر و همچنین، رسم پیش‌بینی برحسب مقدار واقعی زده‌شده‌است.

برای مدل پایه این سوال یک شبکه ۸ لایه با تابع فعالسازی ReLU مورد استفاده قرار گرفته‌است که تعداد نورون‌های این ۸ لایه به صورت زیر است:

Input_size, ۵۱۲, ۲۵۶, ۱۲۸, ۶۴, ۳۲, ۱۶, ۱

نمودارهای خطا و پیش‌بینی به صورت زیر شد.

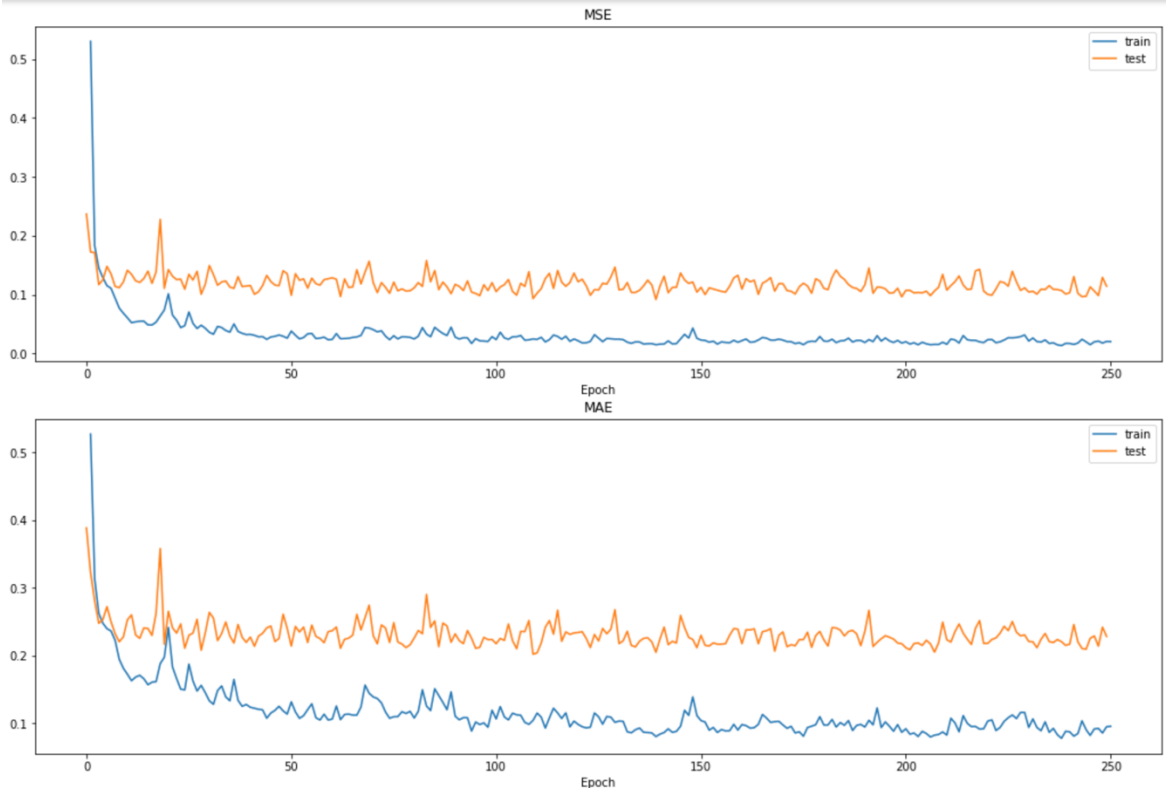


شکل ۶ – نمودار خطاهای **MAE** و **Loss** برای شبکه‌ی پایه

شکل ۷ – نمودار مقادیر پیش‌بینی شده بر مبنای واقعی

همانطور که مشاهده می‌کنید، این شبکه عملکرد نسبتا خوبی داشته‌است.
حال تعداد لایه‌ها را ۵ عدد قرار می‌دهیم. تعداد نورون‌ها به صورت زیر است.
Input size, ۵۱۲, ۱۲۸, ۳۲, ۱

نمودارهای خطا و پیش‌بینی به صورت زیر شد.



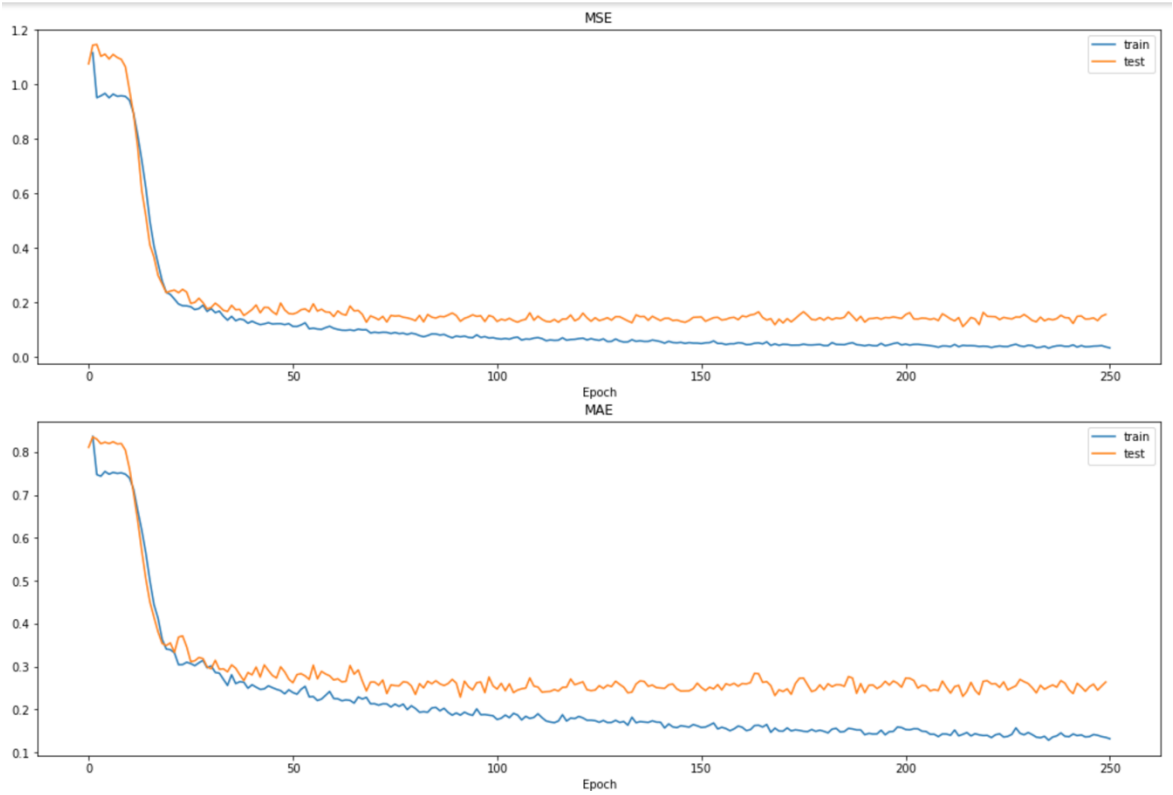
شکل ۸ – نمودار خطاهای **MAE** و **Loss** برای شبکه‌ی با ۵ لایه

شکل ۹– نمودار مقادیر پیش‌بینی شده بر مبنای واقعی

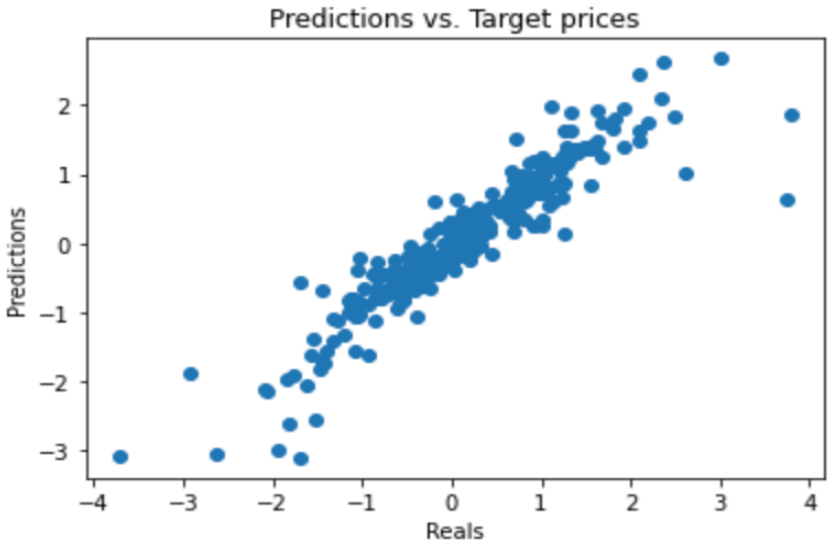
تعداد لایه‌ها چندان موثر نبوده‌است زیرا پیچیدگی مسئله به حدی نیست که تعداد لایه‌ی بالا بخواهد. کمی معماری ۸ لایه بهتر عمل کرده‌است. البته تعداد نورون‌ها نیز ممکن است اثرگذار باشد که در اینجا بررسی نشده‌است.

حال تابع فعال‌ساز را از ReLU به Sigmoid تغییر می‌دهیم. تعداد لایه‌ها و نورون‌های هر لایه مانند مدل پایه است.

نمودارهای خطا و پیش‌بینی به صورت زیر شد.



شکل ۱۰ – نمودار خطاهای **MAE** و **Loss** برای شبکه‌ی با **Sigmoid**

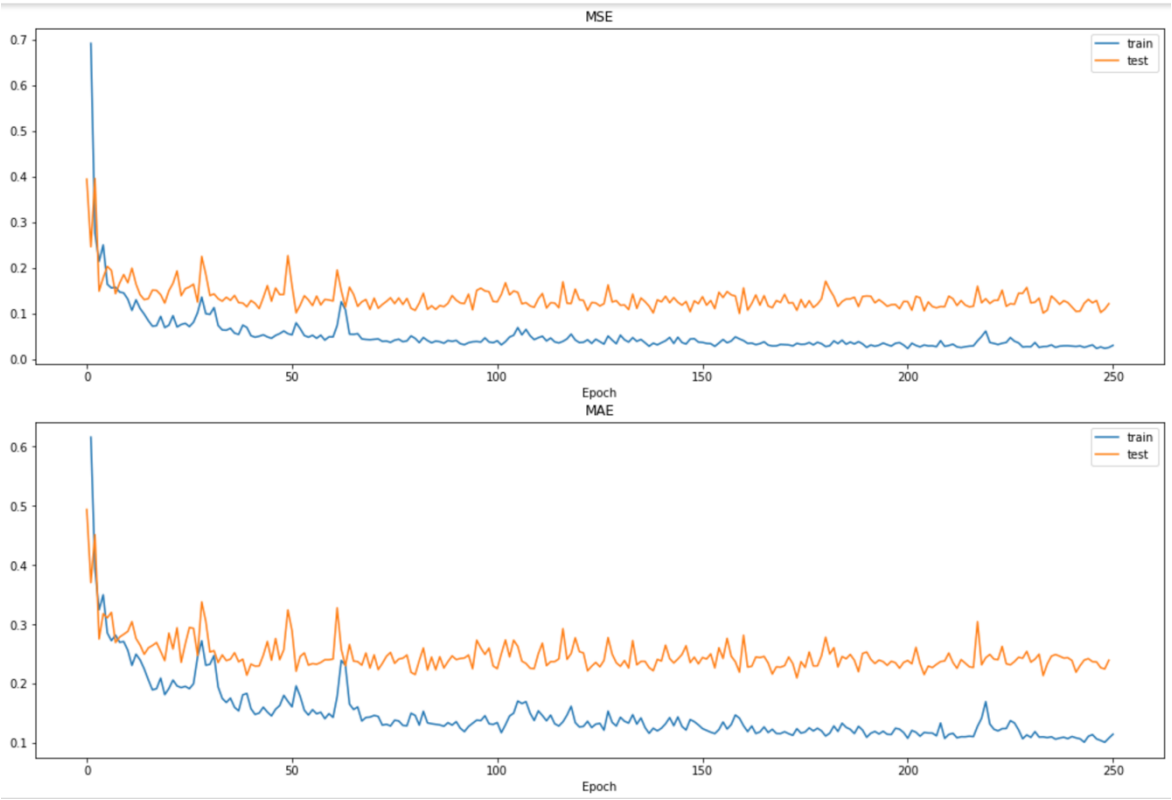


شکل ۱۱– نمودار مقادیر پیش‌بینی شده بر مبنای واقعی

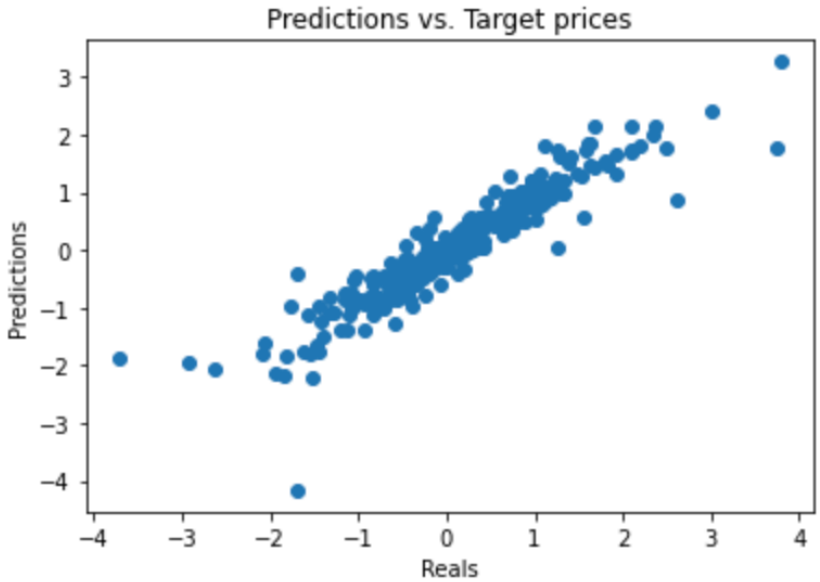
همانطور که مشاهده می‌کنید، تابع فعالسازی ReLU از نظر سرعت اجرا و همگرایی سریعتر از توابع دیگر است. همچنین، این تابع مشکل vanishing gradient را ندارد.

البته تابع Sigmoid نیز عملکرد قابل قبولی داشته‌است.

ج) مدل پایه به عنوان به بهترین مدل بخش قبل انتخاب شد. نمودارهای خطا و پیش‌بینی آن در زیر آمده‌است.



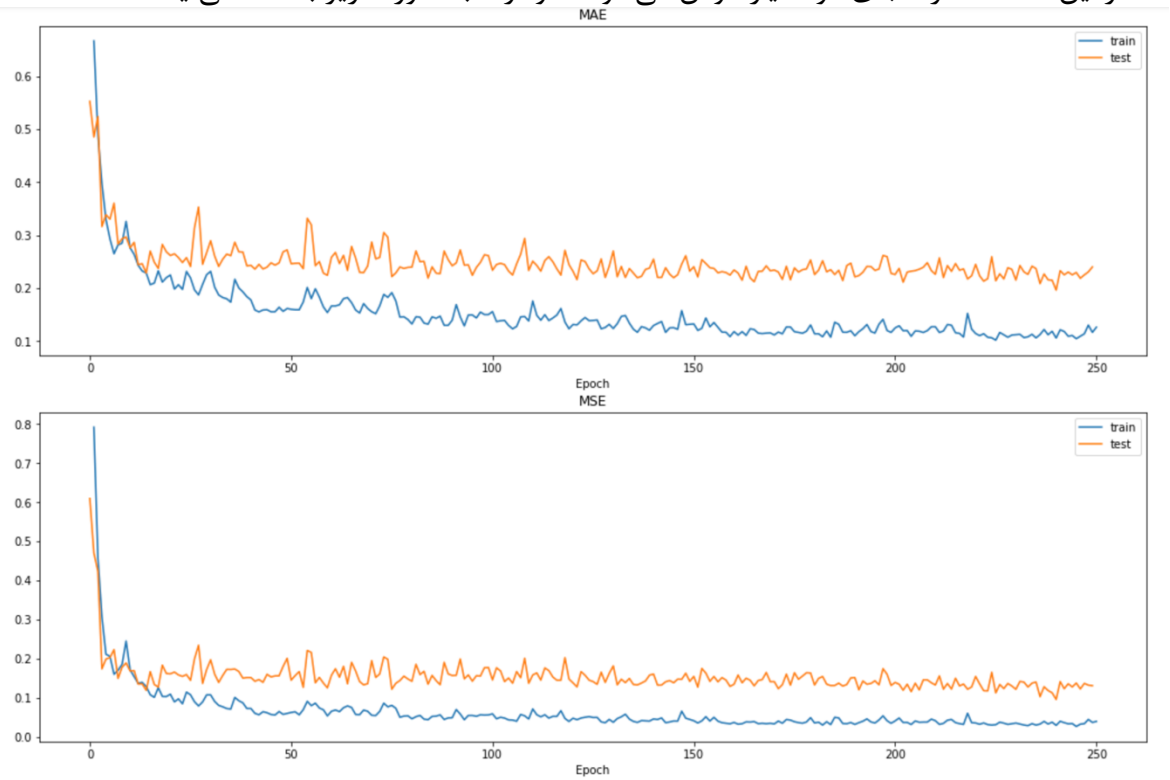
شکل ۱۲ – نمودار خطاهای **MAE** و **Loss** برای شبکه‌ی پایه



شکل ۱۳ – نمودار مقادیر پیش‌بینی شده بر مبنای واقعی

بهترین تعداد epoch جایی است که دقت validation یا همان test در این سوال(به علت نداشتن validation) کاهش نمی‌یابد. برای این کار یک تابع best_epoch نوشته‌شده‌است که با بررسی خطاها در epochهای مختلف تعداد بهینه epoch را می‌یابد. در این جا ۱۴ به عنوان تعداد بهینه epoch به دست‌آمده است. همانطور که در نمودار می‌بینید، این نقطه یک کمینه است که پس از آن تابع افزایش می‌یابد و در کاهش بعدی در همان حدود می‌رسد. پس از این چند epoch جلوتر فرآیند آموزش وارد overfitting می‌شود. البته می‌شود با کمی سختگیری کمتر حوالی ۷۰ را نیز به عنوان epoch بهینه در نظر گرفت.

د) در این قسمت صرفا جای دو معیار عوض می‌شود. نمودارها به صورت زیر به‌دست‌می‌آیند.



شکل ۱۴ – نمودار خطاهای **MSE** و **Loss** برای شبکه‌ی پایه

شکل ۱۵ – نمودار مقادیر پیش‌بینی شده بر مبنای واقعی

در این جا ۴۸ به عنوان تعداد بهینه epoch به دست‌آمده است. همانطور که در نمودار می‌بینید، این نقطه یک کمینه است که پس از آن تابع افزایش می‌یابد و در کاهش بعدی در همان حدود می‌رسد. پس از این چند epoch جلوتر فرآیند آموزش وارد overfitting می‌شود.

ه) روابط ریاضی MSE و MAE به صورت زیر است.

$$MAE = \frac{\sum_{i=1}^{n} |y_i - \hat{y}_i|}{n}$$

$$MSE = \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{n}$$

اولین تفاوتی که مشاهده می‌شود کوچک‌تر بودن اعداد MSE است که دلیل آن توان دو رساندن عدد کوچک‌تر از ۱ است. اگر از رابطه‌ی MSE به عنوان خطا استفاده کنیم، به دلیل وجود توان دو جریمه‌ی بیشتری برای خطا در نظر می‌گیرد و برای همین از مقادیر پرت دورتر می‌مانیم. البته از این دید اگر به مسئله نگاه کنیم که با استفاده از MSE به مقادیر پرت اهمیت بیشتری می‌دهیم و به این دلیل، MAE می‌تواند معیار بهتری باشد. از طرفی گرادیان MAE برای مقادیر کوچک هم مقدار بزرگی است و به همین دلیل، برای یادگیری معیار مناسبی نیست. در این داده به دلیل اینکه به نظر outlierهای کمی داریم می‌توانیم از MAE به عنوان معیار خطا استفاده کنیم.

سوال ۲ – MLP (Classification)

الف) ابتدا داده‌ی موردنظر با استفاده از کتابخانه‌ی pandas خوانده می‌شود. سپس، با استفاده توابع describe و info ویژگی‌های کلی آن مورد بررسی قرار می‌گیرند. این مجموعه‌ی داده هیچ missing value ای ندارد.

برای پیش‌پردازش تمام ستون‌ها را نرمالیزه می‌کنیم. (mean normalization) این کار باعث هم مقیاس شدن داده‌ها می‌شود که برای تعمیم‌پذیری مدل و عدم حساسیت آن به مقادیر بزرگ نیاز است. سپس، متغیر هدف که دودویی است را به صفر و یک تبدیل می‌کنیم تا بتوانیم با استفاده از شبکه‌ی عصبی آن را پیش‌بینی کنیم.

پس، get_train_val_test_loaders مجموعه داده‌ها را به سه قسمت آموزش، اعتبارسنجی و آزمون تقسیم می‌کنیم. همچنین، این تابع داده‌ها را به tensor و dataloader تقسیم می‌کند تا برای مدل شبکه‌ی عصبی مناسب باشند. این تقسیم با استفاده از تابع random_split در کتابخانه torch انجام می‌شود که به صورت تصادفی این کار را انجام می‌دهد تا داده‌ها ترتیب خاصی نداشته باشند و کاملا تصادفی باشند.

ابتدا داده‌ها به نسبت ۸۰ به ۲۰ به داده‌ها تست و آموزش تقسیم می‌شوند. دلیل این کار این است که در این مسئله تعداد داده‌ها کم است و به همین بخش زیادی به آموزش اختصاص می‌یابد تا مقدار کافی داده برای آموزش داشته‌باشیم.

از داده‌های آموزش ۲۰ درصد برای ارزیابی اختصاص می‌یابد و مابقی برای آموزش استفاده می‌شوند. دلیل این کار برای تعیین بهترین تعداد epoch و سایر پارامترهای مدل بدون دیدن داده‌ی تست است تا بتوانیم واقعا قدرت تعمیم مدل خود را بسنجیم. با این روش عددی که برای داده‌ی تست گزارش می‌کنیم بدون هیچ اطلاعی است و به همین دلیل، قدرت تعمیم مدل برای یک داده‌ی ندیده را نشان می‌دهد.

برای این داده‌ها شبکه‌ی SonarNet نوشته‌شده‌است که یک مدل general است که در constructor داده‌ها تعداد لایه‌ها، سایز هر لایه و تابع فعالساز لایه‌ها را می‌گیرد و شبکه‌ی مورد نظر را خروجی می‌دهد. این شبکه‌ها یک dropout ده درصدی نیز دارند که از overfitting آن

جلوگیری کند. همچنین، توابع فعالسازی مختلف برای لایه‌ی آخر نیز تست شده‌است که در انتها sigmoid به عنوان بهترین آن‌ها انتخاب شده‌است.

تابع train مسئولیت آموزش مدل و خروجی دادن دقت و خطای مدل در هر epoch برای داده‌ی آموزش و ارزیابی را دارد. تابع plot_plots برای رسم نمودار در حالت کلی نوشته‌شده‌است.

معماری شبکه‌ی پایه‌ی مورد استفاده به صورت زیر است.

۸ لایه با تعداد نورون‌های زیر: (به ترتیب)

Input size, ۵۱۲, ۲۵۶, ۱۲۸, ۶۴, ۳۲, ۱۶, ۱

تابع فعالساز لایه‌های ماقبل آخر: ReLU

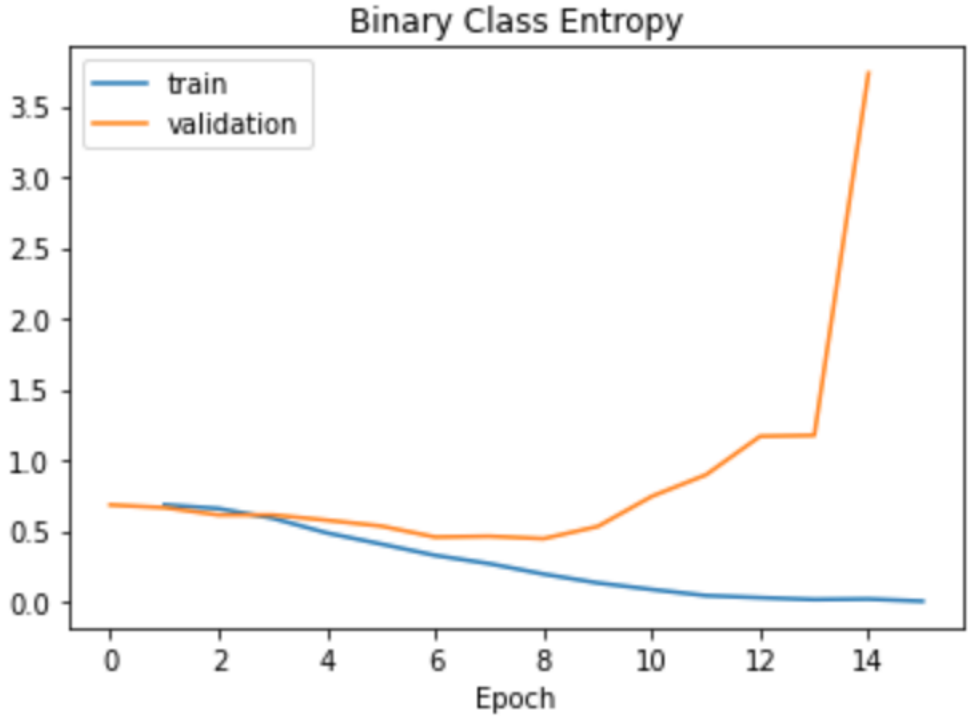
بهینه ساز: Adam با نرخ یادگیری ۰.۰۰۲

تابع loss: BCELoss یا همان Binary Class Entropy

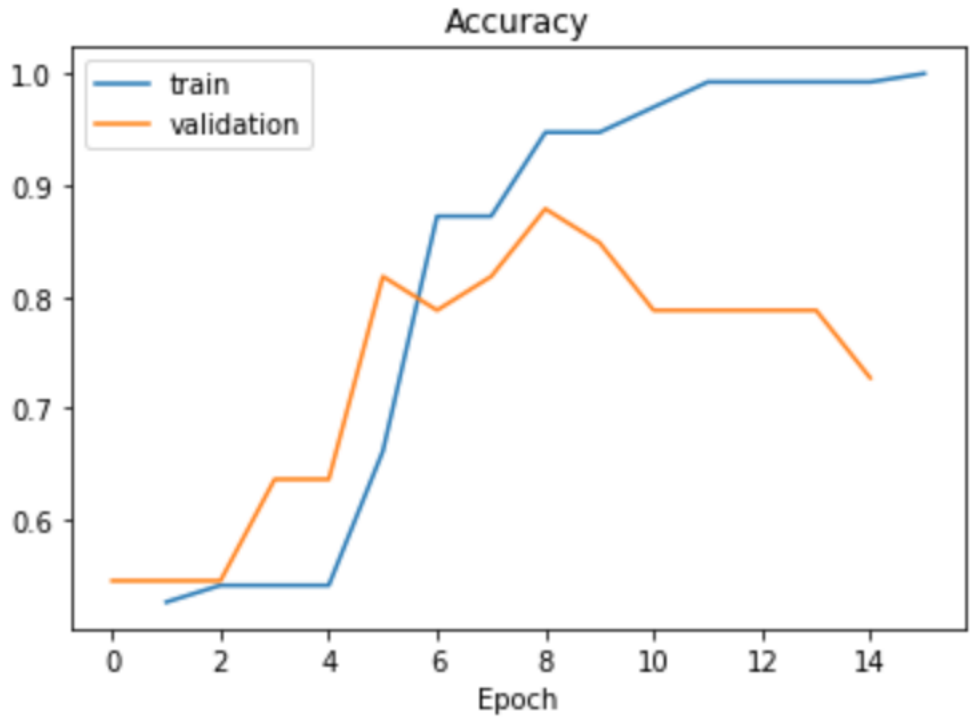
تابع فعالساز لایه‌ی آخر: Sigmoid

با dropout ۱۰ درصدی برای جلوگیری از overfitting

ب) نمودار تغییرات دقت و خطای مدل در هر epoch برای دادگان آموزش و ارزیابی به صورت زیر است.



شکل ۱۶ – نمودار **Loss** برای شبکه‌ی پایه در هر **epoch**



شکل ۱۷ – نمودار دقت برای شبکه‌ی پایه در هر **epoch**

ج) مقادیر خطا، دقت و confusion matrix دادگان تست به صورت زیر است.
دقت: ۹۲.۸۵ درصد
گزارش دسته‌بندی:

Classificaiton report:

	precision	recall	f1-score	support
0	1.00	0.86	0.92	21
1	0.88	1.00	0.93	21
accuracy			0.93	42
macro avg	0.94	0.93	0.93	42
weighted avg	0.94	0.93	0.93	42

شکل ۱۸ – گزارش دسته‌بندی شبکه‌ی پایه

میزان خطا: ۰.۲۱۱۶

Confusion matrix:

```
[[18  3]
 [ 0 21]]
```

د) معیار خطای مورد استفاده در آموزش شبکه BCELoss است که یک نوع cross entropy loss است که در مسائل دودویی مورد استفاده قرار می‌گیرد. در اینجا نیز با مسئله‌ای دودویی روبرو هستیم پس از این تابع خطا استفاده می‌کنیم. این تابع سعی می‌کند تا میانگین خطای میان پیش‌بینی و مقدار

واقعی را کاهش دهد. توابع خطا cross entropy جریمه‌ی سنگینی برای مقادیر غلط با اطمینان بالا در نظر می‌گیرند. همچنین، این نوع توابع پیش‌بینی‌های صحیحی با اطمینان پایین را نیز جریمه می‌کنند. به همین دلیل و دودویی بودن مسئله این معیار مورد انتخاب قرار گرفته‌است تا بتوان مدلی با اطمینان بالا ساخت.

جایگزین موجود برای این معیار، Hinge Embedding Loss است. برای استفاده از این تابع مقادیر متغیر هدف باید ۱ و ۱- باشند که در این اینجا باید متغیر هدف را دوباره تغییر داد. از طرفی این تابع سعی بر درست بودن علامت پیش‌بینی می‌کند و این تفاوت را در نظر می‌گیرد. به همین دلیل، برای مسائلی که مشابهت یا عدم مشابهت دو شی در میان است می‌تواند مفید باشد که در این جا مد نظر نیست و نمی‌تواند کمک‌کننده باشد.

ه) accuracy لزوما معیار درستی نمی‌تواند باشد. برای مثال، اگر احتمال سرطان ۰.۱ درصد باشد، یعنی از هر ۱۰۰۰ نفر ۱ نفر سرطان داشته‌باشد، با پیش‌بینی‌کردن سالم بودن همه به دقت ۹۹.۹ درصد می‌رسیم در صورتی که فرد سرطانی را تشخیص نداده‌ایم و این می‌تواند مشکل‌ساز شود. به همین دلیل، از معیارهای دیگر مانند precision، recall و confusion matrix استفاده شده‌است تا بتوانیم خطاها را به همراه منابع آن‌ها شناسایی کنیم و دقیق‌تر باشیم. این مقادیر در زیر قابل مشاهده می‌باشند.

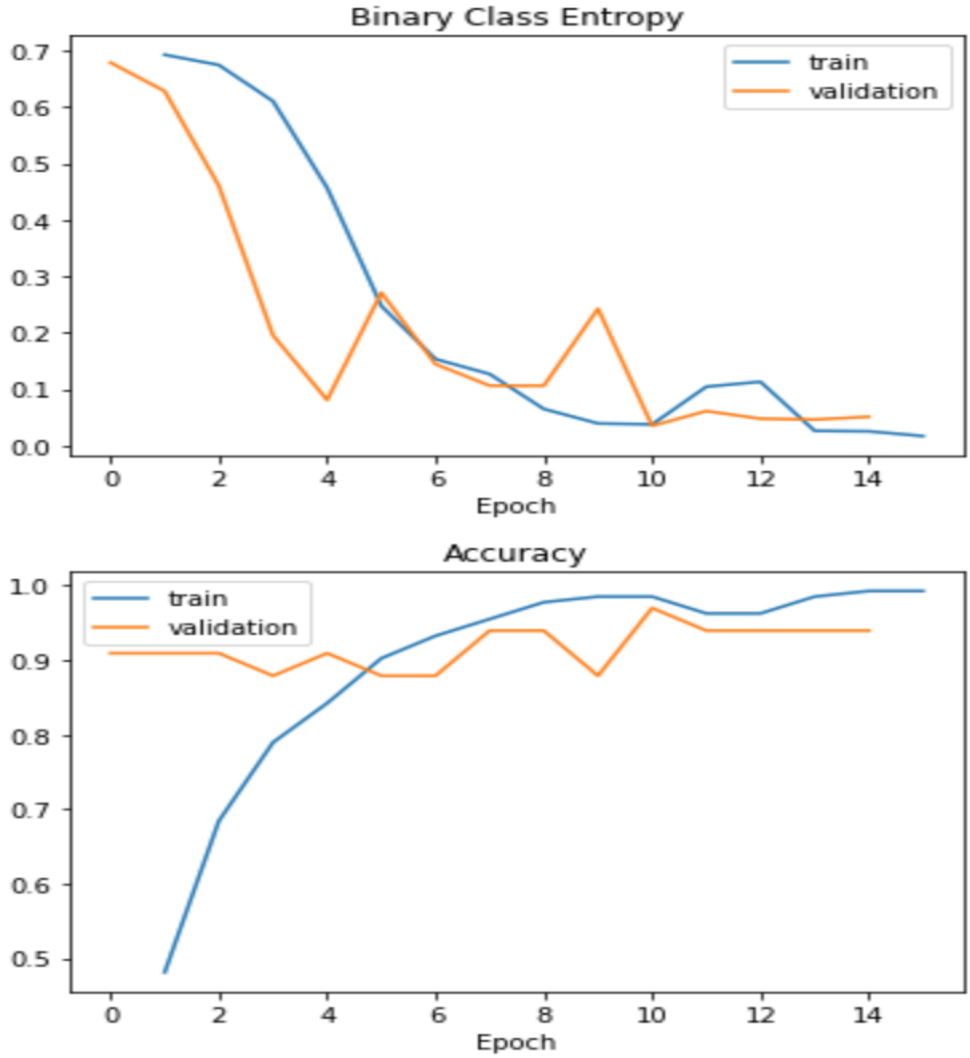
```
Classificaiton report:
              precision    recall  f1-score   support

           0       1.00      0.86      0.92        21
           1       0.88      1.00      0.93        21

    accuracy                           0.93        42
   macro avg       0.94      0.93      0.93        42
weighted avg       0.94      0.93      0.93        42

[[18  3]
 [ 0 21]]
```

شکل ۱۹ – مقدار معیارهای دیگر

همانطور که می‌بینید، confusion matrix نشان می‌دهد که ۳ داده‌ای که در واقعی صفر بوده‌اند، یک پیش‌بینی شده‌اند. یعنی دقیقا می‌دانیم خطا در کجا رخ داده‌است.

و) به دلیل استفاده از dataloader به صورت mini batch و تقسیم تصادفی روش مورد استفاده از ابتدا stochastic mini batch بوده‌است. نتیجه‌ی اجرا به ازای batch با سایز ۳۲، ۶۴ و ۱۲۸ در زیر آمده‌است.

- Batch with size = ۳۲

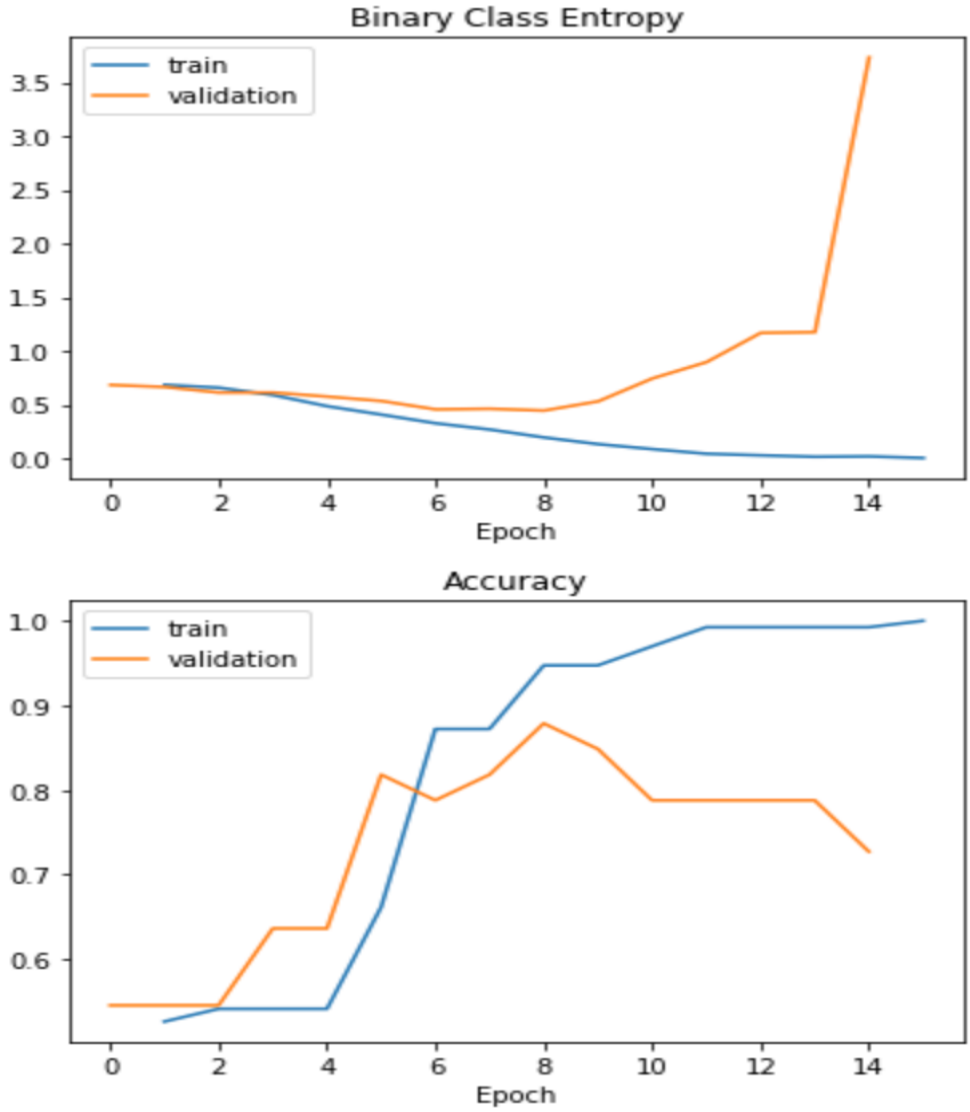


شکل ۲۰ – نمودار دقت برای سایز **batch** ۳۲ در هر **epoch**

```
Accuracy: 0.9285714285714286
Classificaiton report:
              precision    recall  f1-score   support

           0       0.89      1.00      0.94        25
           1       1.00      0.82      0.90        17

    accuracy                           0.93        42
   macro avg       0.95      0.91      0.92        42
weighted avg       0.94      0.93      0.93        42

Loss: 0.19308958668261766
Confusion Matrix:
[[25  0]
 [ 3 14]]
```

شکل ۲۱ – معیارهای دقت برای سایز **batch** ۳۲ در هر **epoch**

- Batch with size = ۶۴

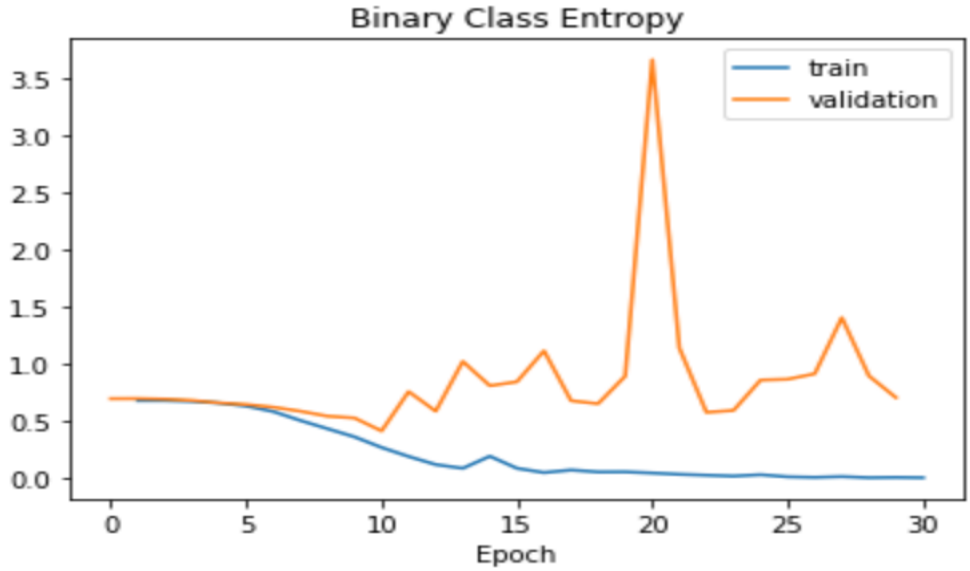


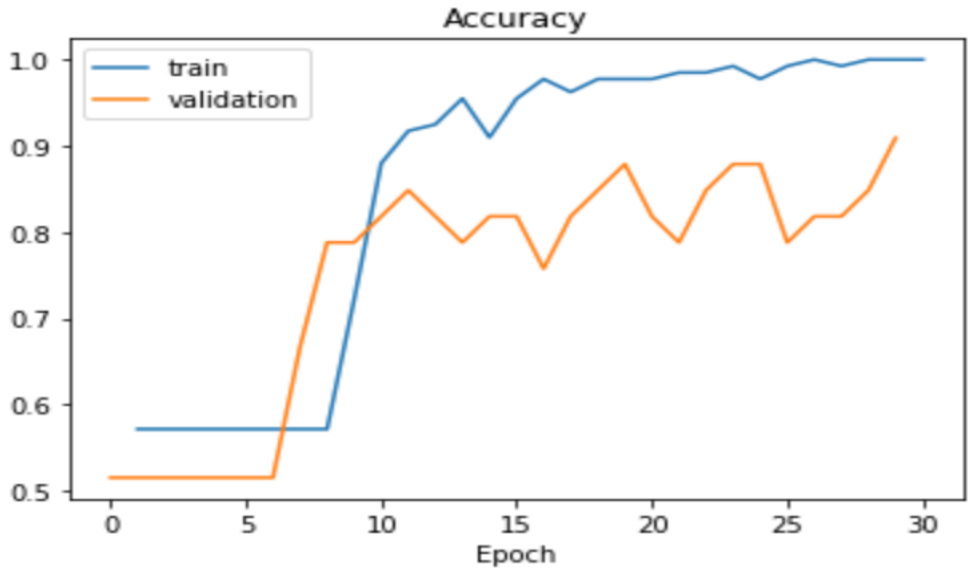
شکل ۲۲ – نمودار دقت برای سایز **batch** ۶۴ در هر **epoch**

```
Accuracy: 0.9285714285714286
Classificaiton report:
              precision    recall  f1-score   support

           0       1.00      0.86      0.92        21
           1       0.88      1.00      0.93        21

    accuracy                           0.93        42
   macro avg       0.94      0.93      0.93        42
weighted avg       0.94      0.93      0.93        42

Loss: 0.21166428923606873
Confusion Matrix:
[[18  3]
 [ 0 21]]
```

شکل ۲۳ – معیارهای دقت برای سایز **batch** ۶۴ در هر **epoch**

- Batch with size = ۱۲۸





شکل ۲۲ – نمودار دقت برای سایز **batch** ۶۴ در هر **epoch**

```
Accuracy: 0.9047619047619048
Classificaiton report:
              precision    recall  f1-score   support

           0       0.88      0.96      0.92        24
           1       0.94      0.83      0.88        18

    accuracy                           0.90        42
   macro avg       0.91      0.90      0.90        42
weighted avg       0.91      0.90      0.90        42

Loss: 0.6666709780693054
Confusion Matrix:
[[23  1]
 [ 3 15]]
```

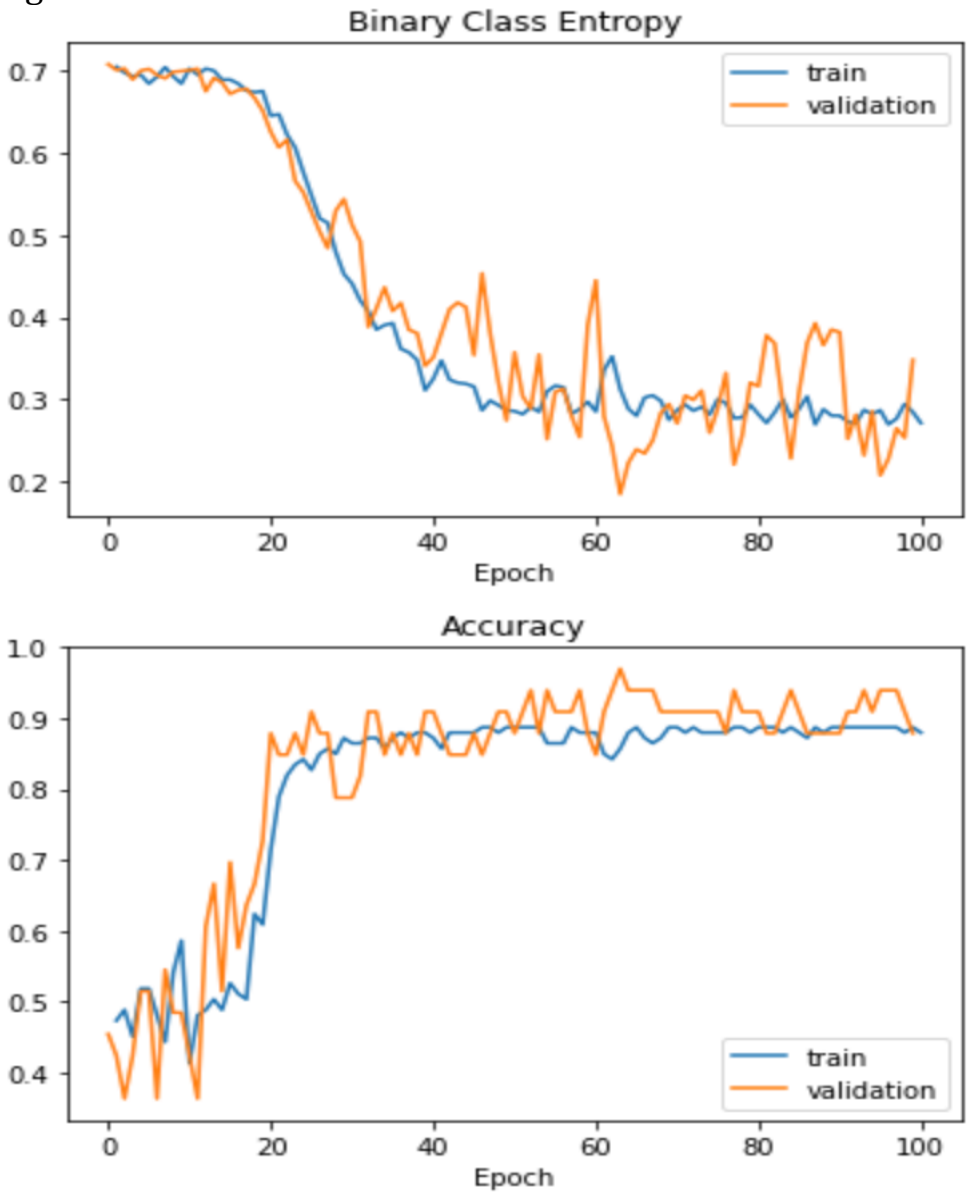
شکل ۲۳ – معیارهای دقت برای سایز **batch** ۶۴ در هر **epoch**

در حالت ۳۲، داده‌های به بخش‌های ۳۲ تایی تقسیم می‌شوند. به همین صورت، در حالت ۶۴ تایی و ۱۲۸ تایی... به علت متفاوت بودن تعداد batchها و تعداد داده‌های در هر $batch$ نمودارها و خروجی متفاوت‌است. نوع حرکت کردن و همگرایی در این تعدادها متفاوت است. هر چه تعداد داده‌های یک $batch$ بیشتر باشد، هر $iteration$ طولانی‌تر است اما تعداد $batch$ کمتراست. از طرفی $batch$ با سایز زیاد امکان همگرایی سریع‌تر و بهتر را فراهم می‌کند و در عوض، اگر در اکسترمم محلی بیافتد، نمی‌تواند از آن خارج شود چراکه با دیدن داده‌ی زیاد پارامترهای خود را به روز می‌کند و امکان پرش ندارد. در مقابل $batch$ با سایز کم، همگرایی را کمی سخت‌تر می‌کند اما امکان خروجی از اکسترمم محلی را به دلیل پرش دارد.
سایز مناسب در این سوال همانطور که می‌بینید، سایز ۳۲ بوده است. دلیل آن این است که این با این سایز در اکسترمم محلی گیر نکرده‌ایم و مرتبا پارامترها را به‌روز کرده‌ایم و همانطور که می‌بینید، به نسبت دو سایز دیگر دقت بالاتری روی داده‌های اعتبارسنجی و آزمون داریم و این حاکی از قدرت تعمیم‌دهی بالای این مدل است.

ح) epoch به یک بار دیدن کل داده‌ها گفته می‌شود در صورتی که iteration به بررسی هر batch گفته می‌شود. تعداد epochها در سوال قبل ۳۰ بوده‌است. در صورتی که تعداد iterationها به تعداد batchهای آموزش ضربدر ۳۰ بوده‌است یعنی با batch size ۱۲۸، ۶۰ iteration داشته‌ایم.
بهترین تعداد epoch جایی است که دقت validationکاهش نمی‌یابد یا حتی افزایش می‌یابد. برای این کار یک تابع best_epoch نوشته‌شده‌است که با بررسی ثابت‌ماندن یا افزایش خطاها در epochهای مختلف تعداد بهینه epoch را می‌یابد. در این جا ۸ به عنوان تعداد بهینه epoch به دست می‌آید. همانطور که در نمودار می‌بینید، این نقطه یک کمینه است که پس از آن تابع خطا افزایش می‌یابد و مقدار دقت کاهش می‌یابد. اگر تعداد epochها افزایش یابد، فرآیند ovefitting روی داده‌ی آموزش رخ می‌دهد. به این معنی که دقت داده‌ی آموزش به ۱۰۰ درصد نزدیک می‌شود ولی دقت validation و تست کاهش می‌یابد. در حقیقت، قدرت تعمیم‌دهی مدل کاهش می‌یابد و مدل فقط داده‌ی آموزش را حتی با noiseهایش یاد می‌گیرد.
ط) توابع فعال‌ساز لایه‌های ماقبل آخر تغییر داده‌شده‌اند و به ازای هر کدام نتایج به صورت زیر به دست آمد.

از توابع $ReLU$، $Sigmoid$ و $tanh$ استفاده شده‌است.
توابع $Sigmoid$ و $tanh$ مشکل $vanishing\ gradient$ دارند ولی $ReLU$ این مشکل را ندارد. ذات $ReLU$ خطی است که با مسئله ما سازگارتر است اما ذات دو تابع دیگر غیرخطی است. در بین این توابع $tanh$ متقارن و با مرکز صفر است که برای ما کاربرد ندارد. در تابع $ReLU$ تمام مقادیر منفی صفر می‌شوند. ما نیز در مسئله مقادیر منفی نداریم و این کمک‌کننده است. دقت و بازدهی $ReLU$ بیشینه است چراکه از طرف مثبت حد ندارد و به همین دلیل، به مشکل $vanishing\ gradient$ برنمی‌خورد. از طرفی سرعت اجرا و همگرایی $ReLU$ همانطور که در نمودارها می‌بینید، از دو رقیب خود بیشتر است.

- Sigmoid Activation Function

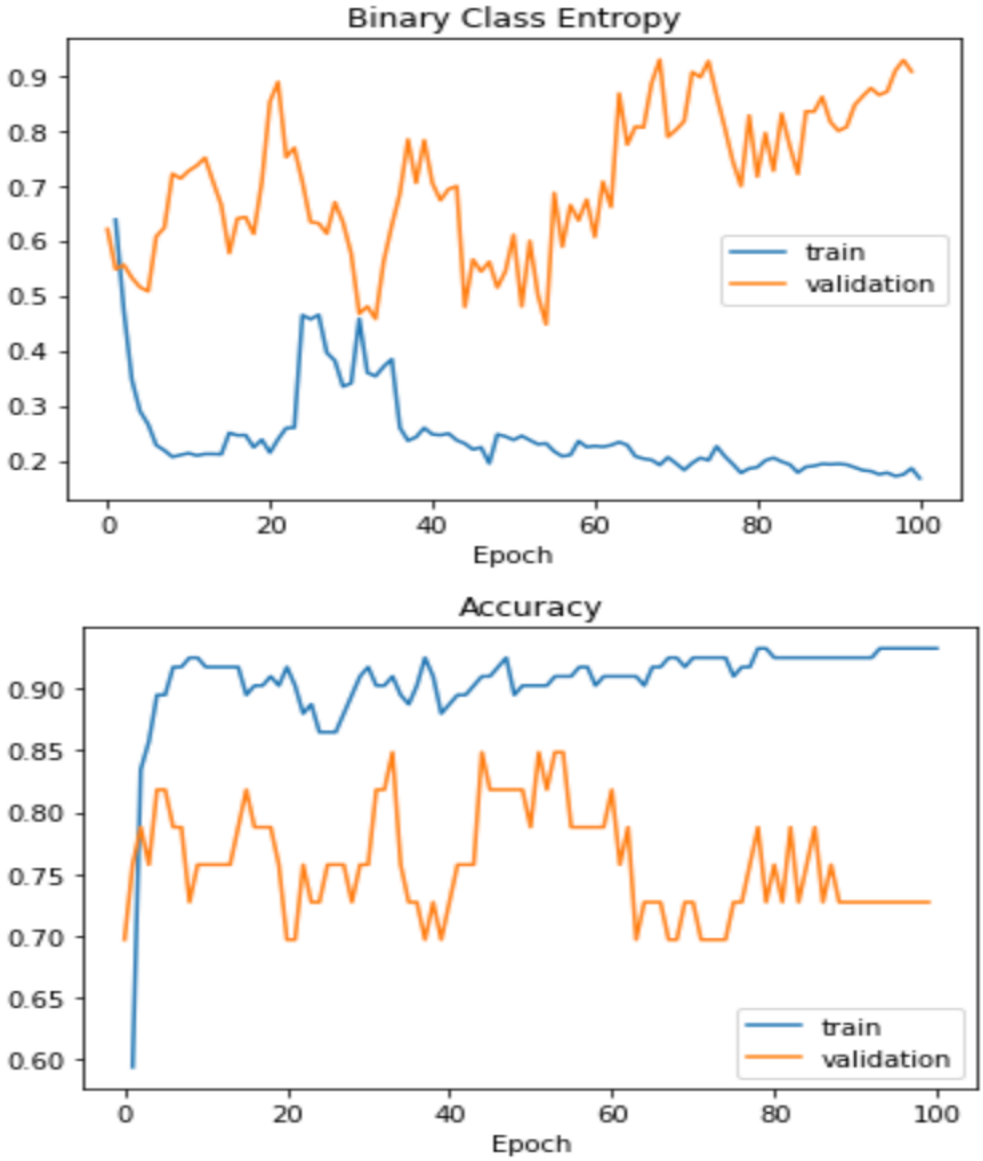


شکل ۲۴ – نمودار دقت و خطا برای تابع فعالسازی **Sigmoid**

```
Accuracy: 0.7380952380952381
Classificaiton report:
              precision    recall  f1-score   support

           0       0.72      0.68      0.70        19
           1       0.75      0.78      0.77        23

    accuracy                           0.74        42
   macro avg       0.74      0.73      0.73        42
weighted avg       0.74      0.74      0.74        42

Loss: 0.764928936958313
Confusion Matrix:
[[13  6]
 [ 5 18]]
```

شکل ۲۵ – مقادیر معیارهای دقت و خطا برای تابع فعالسازی **Sigmoid**

- Tanh Activation Function

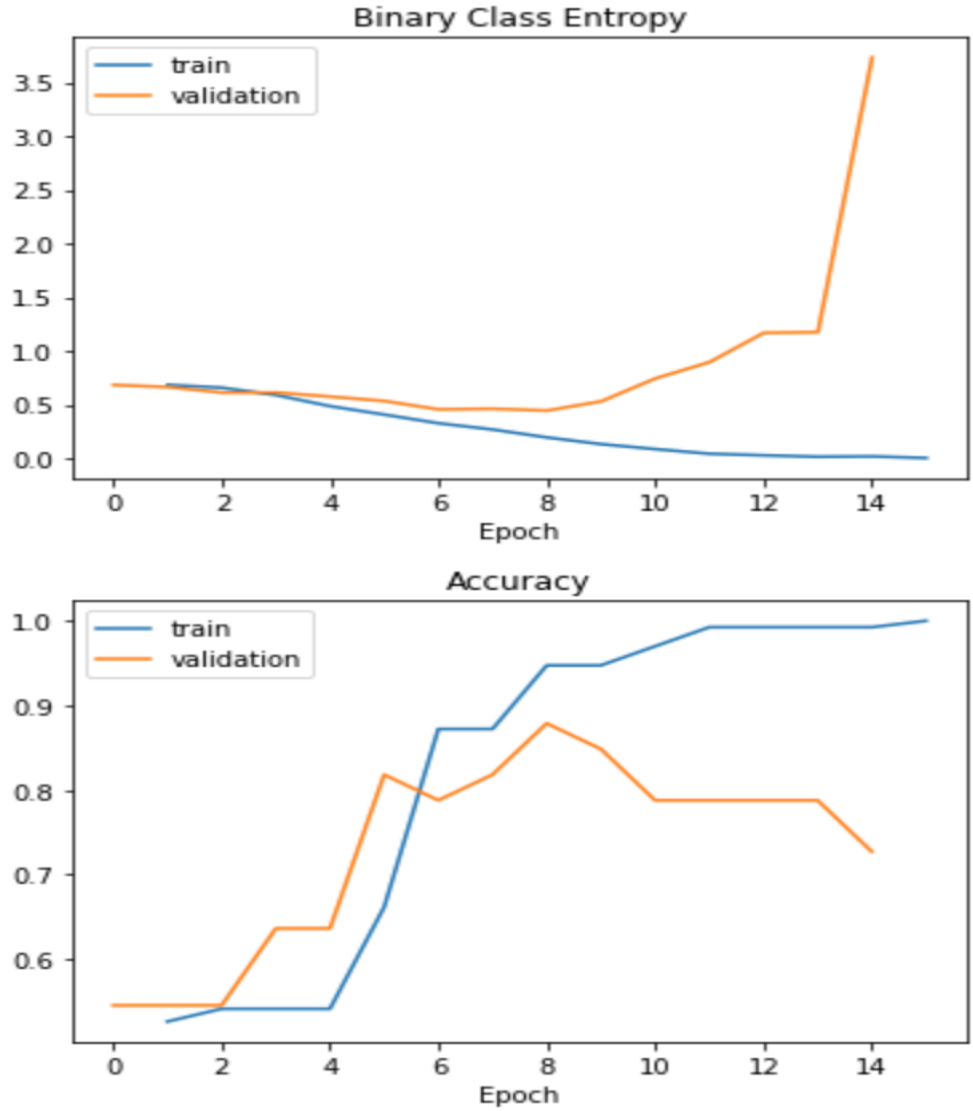


شکل ۲۶ – نمودار دقت و خطا برای تابع فعالسازی **Tanh**

```
Accuracy: 0.7142857142857143
Classificaiton report:
              precision    recall  f1-score   support

           0       0.65      0.85      0.74        20
           1       0.81      0.59      0.68        22

    accuracy                           0.71        42
   macro avg       0.73      0.72      0.71        42
weighted avg       0.74      0.71      0.71        42

Loss: 0.8308720588684082
Confusion Matrix:
[[17  3]
 [ 9 13]]
```

شکل ۲۷ – مقادیر معیارهای دقت و خطا برای تابع فعالسازی **Tanh**

- ReLU Activation Function

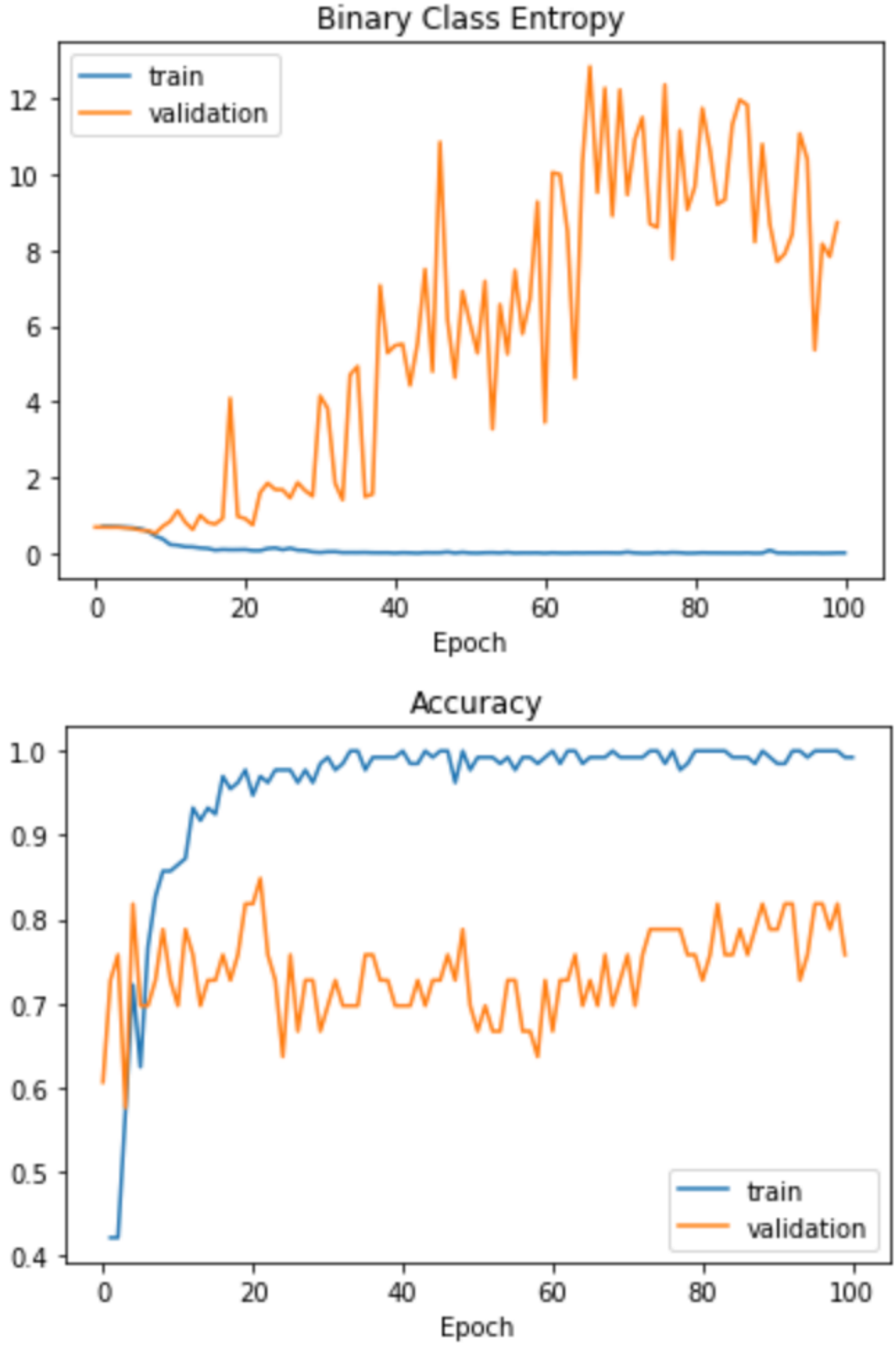


شکل ۲۸ – نمودار دقت و خطا برای تابع فعالسازی **ReLU**

```
Accuracy: 0.9285714285714286
Classificaiton report:
              precision    recall  f1-score   support

           0       1.00      0.86      0.92        21
           1       0.88      1.00      0.93        21

    accuracy                           0.93        42
   macro avg       0.94      0.93      0.93        42
weighted avg       0.94      0.93      0.93        42

Loss: 0.21166428923606873
Confusion Matrix:
[[18  3]
 [ 0 21]]
```

شکل ۲۹ – مقادیر معیارهای دقت و خطا برای تابع فعالسازی **ReLU**

ی) خیر. به نظر شبکه پیچیدگی به حد کافی دارد و از اینجا به بعد، موجب overfitting و کاهش دقت روی مجموعه دادگان تست می‌شویم. در حقیقت، قدرت تعمیم شبکه با این کار کاهش پیدا می‌کند و پیچیدگی با مورد اضافه می‌شود. نتایج زیر پس از اضافه کردن دو لایه اتفاق افتاده‌است که تایید کننده فرضیه است.

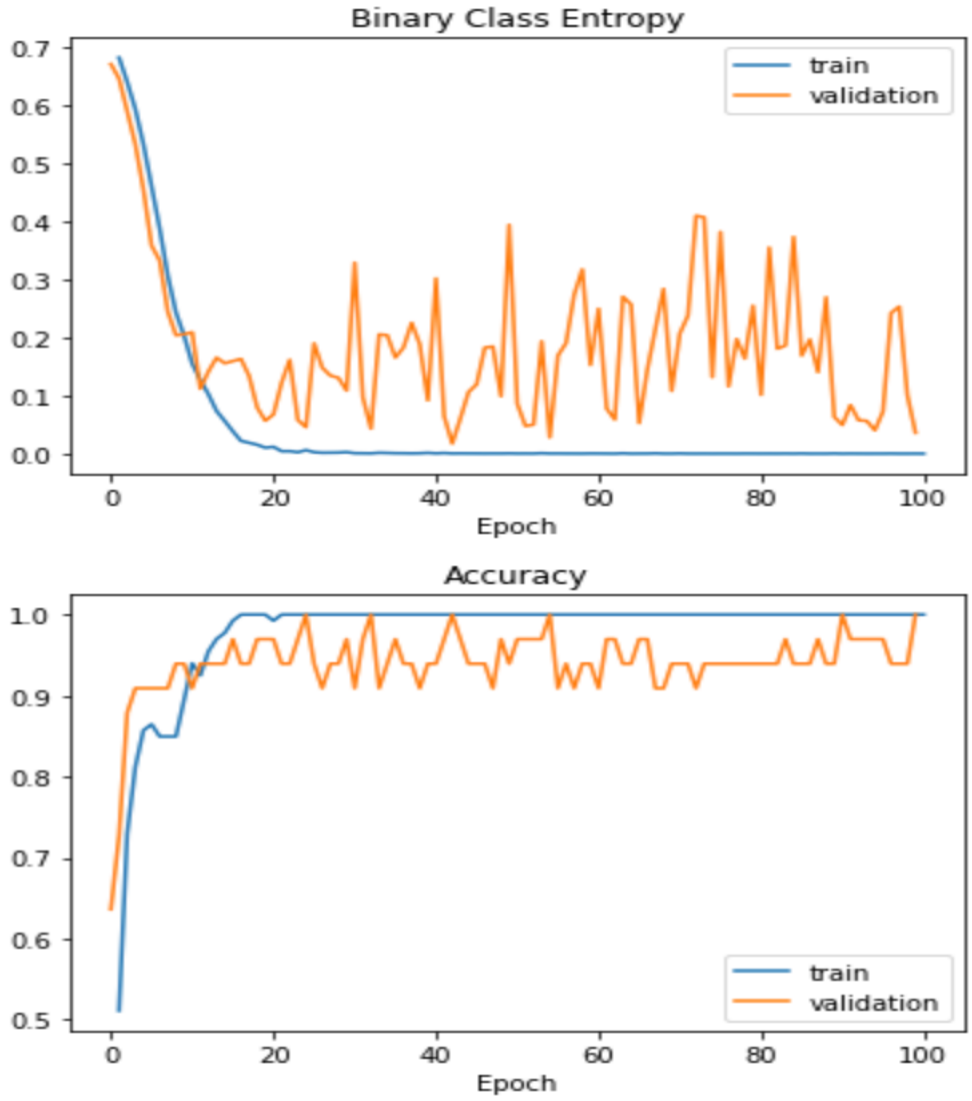


شکل ۳۰ – نمودار دقت و خطا پس از اضافه کردن دو لایه

```
Accuracy: 0.8809523809523809
Classificaiton report:
              precision    recall  f1-score   support

           0       0.67      1.00      0.80        10
           1       1.00      0.84      0.92        32

    accuracy                           0.88        42
   macro avg       0.83      0.92      0.86        42
weighted avg       0.92      0.88      0.89        42

Loss: 2.6612601280212402
Confusion Matrix:
[[10  0]
 [ 5 27]]
```

شکل ۳۱ – مقادیر معیارهای دقت و خطا پس از اضافه کردن دو لایه

ک) معماری بهترین شبکه به‌دست‌آمده به صورت زیر است.
۸ لایه با تعداد نورون‌های زیر: (به ترتیب)
Input size, ۵۱۲, ۲۵۶, ۱۲۸, ۶۴, ۳۲, ۱۶, ۱
تابع فعالساز لایه‌های ماقبل آخر: ReLU
بهینه ساز: Adam با نرخ یادگیری ۰.۰۰۲
تابع loss: BCELoss یا همان Binary Class Entropy
تابع فعالساز لایه‌ی آخر: Sigmoid
با dropout ۱۰ درصدی برای جلوگیری از overfitting
سایز batch: ۳۲
این شبکه به دقت قابل قبولی رسیده‌است. شاید بتوان آن را بهتر کرد. این کار با تغییر پارامترهایی مانند تعداد نورون‌ها هر لایه و ... یا استفاده از کاهش ابعاد ویژگی و حذف noise و distortion از ویژگی‌ها امکان‌پذیر است. این کار اتکاپذیری و قدرت تعمیم شبکه را بالا می‌برد.
ل) خیر. این اتفاق نمی‌افتد. چرا که اکنون پیچیدگی شبکه بیش از حد است. با کمتر کردن تعداد لایه‌ها و نورون‌های هر لایه نیاز به آموزش بیشتری است و به همین دلیل، overfitting دیرتر رخ می‌دهد. از طرفی پیچیدگی شبکه کمتر است و این احتمال overfit شدن را کاهش می‌دهد چراکه پارامترهای مسئله به اندازه کافی است.
این کار امتحان شده‌است و نتایج آن در زیر قابل مشاهده است. دلیل این موضوع آن است که از پارامترها و پیچیدگی مسئله کاسته شده‌است و همچنین، تعداد epochهای بیشتری برای آموزش نیاز است. از طرفی دقت validation نیز بالا مانده‌است و این نشان از قدرت تعمیم بالاتر این شبکه است.



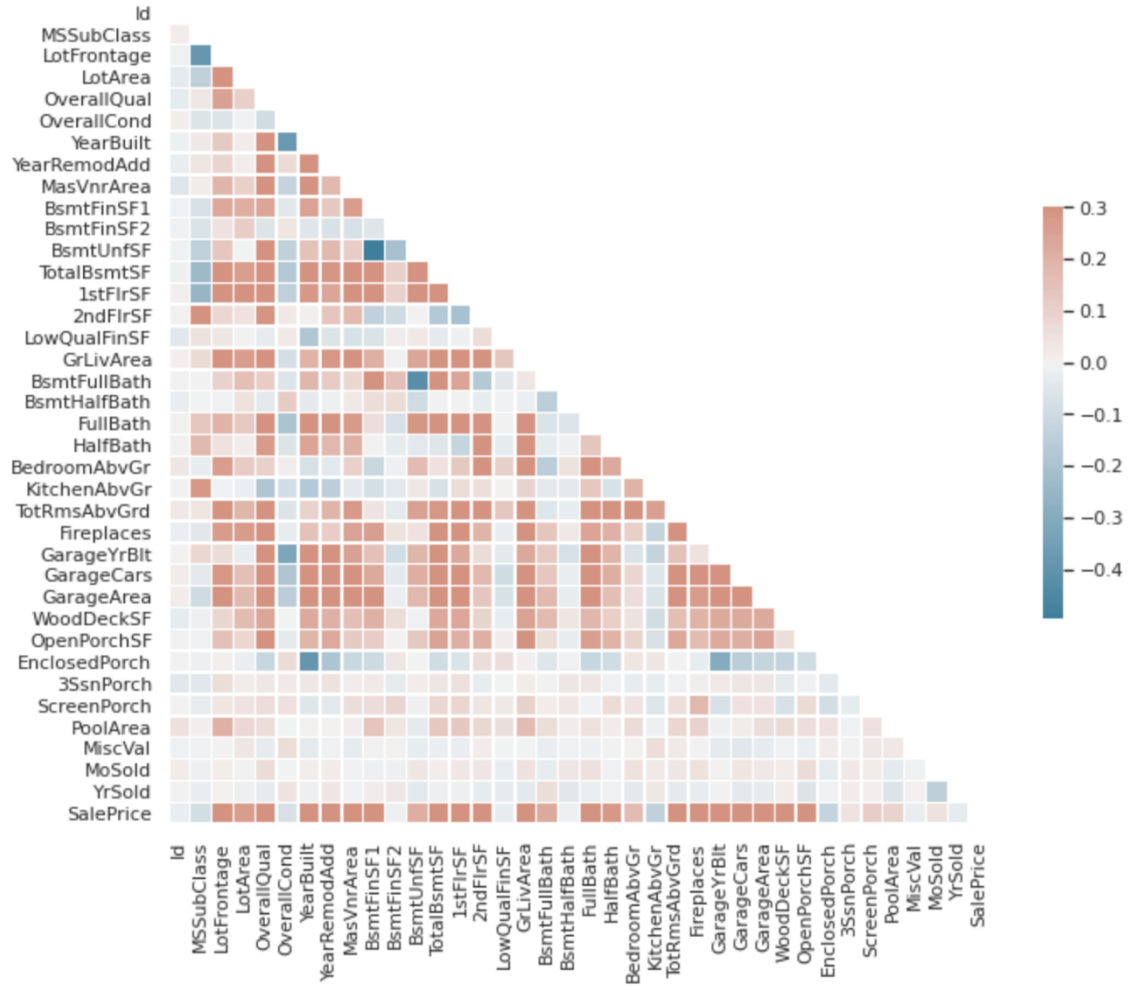
شکل ۳۲ – نمودار دقت و خطا پس از کاستن از لایه‌ها و تعداد نورون‌ها

```
Accuracy: 0.8333333333333334
Classificaiton report:
              precision    recall  f1-score   support

           0       0.88      0.74      0.80        19
           1       0.81      0.91      0.86        23

    accuracy                           0.83        42
   macro avg       0.84      0.82      0.83        42
weighted avg       0.84      0.83      0.83        42

Loss: 1.4825496673583984
Confusion Matrix:
[[14  5]
 [ 2 21]]
```

شکل ۳۳ – مقادیر معیارهای دقت و خطا پس از کاستن از لایه‌ها و تعداد نورون‌ها

سوال ۳ – Dimension Reduction

الف) ماتریس همبستگی داده سوال یک به صورت زیر است.

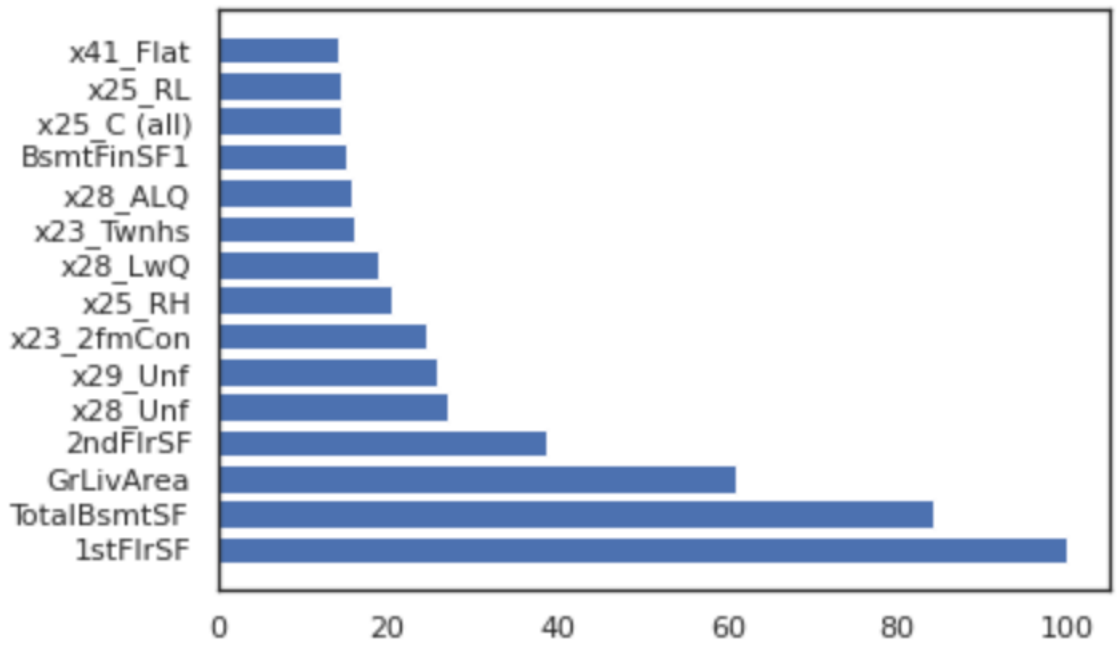


همانطور که می‌بینید، خانه‌های پررنگ نشان‌دهنده میزان همبستگی بالای ویژگی‌ها می‌باشند. طیف قرمز نشان‌دهنده رابطه مثبت خطی و طیف آبی نشان‌دهنده رابطه منفی خطی می‌باشد.
ویژگی‌هایی که با ویژگی هدف رابطه‌ی خطی بهتری دارند می‌توانند در تخمین متغیر هدف نقش موثری داشته‌باشند. به عنوان مثال، OverallQuall و LotArea همبستگی بالایی دارند و می‌توانند ویژگی خوبی برای تخمین باشند.
از طرفی برای ویژگی‌های انتخابی باید حواسمان باشد تا دو متغیر با میزان همبستگی بالا انتخاب نشوند چراکه داده جدیدی وارد مساله نمی‌کنند و صرفا باعث افزونگی می‌شوند. به عنوان مثال، GarageCars و GarageArea همبستگی بالایی با یکدیگر دارند و بودن هر دوی آن‌ها داده‌ی جدیدی اضافه نمی‌کند و کمکی به تخمین نمی‌کند.

ب) اهمیت ویژگی‌ها با استفاده از Linear Regression و ضرایب ویژگی‌ها محاسبه شده و در نمودار زیر قابل مشاهده‌است. این مقدار اهمیت‌ها هم مقیاس شده‌اند و همچنین، نسبت به بیشینه نرمالیزه شده‌اند و به صورت درصد از آن نشان‌داده شده‌اند.

	coefficient	stdev	importance	importance_normalized
1stFlrSF	2.216567e+10	0.460478	1.020681e+10	1.000000e+02
TotalBsmtSF	1.714789e+10	0.500164	8.576756e+09	8.402976e+01
GrLivArea	3.012932e+10	0.206319	6.216259e+09	6.090307e+01
2ndFlrSF	2.502911e+10	0.157217	3.934999e+09	3.855270e+01
x28_Unf	6.983804e+09	0.397021	2.772714e+09	2.716534e+01
...	...	...	...	...
MiscVal	6.589277e-03	1.000000	6.589277e-03	6.455767e-11
OpenPorchSF	5.115821e-03	1.000000	5.115821e-03	5.012166e-11
MasVnrArea	4.208650e-03	1.000000	4.208650e-03	4.123376e-11
MoSold	3.407954e-03	1.000000	3.407954e-03	3.338903e-11
BsmtHalfBath	3.165913e-03	1.000000	3.165913e-03	3.101766e-11

شکل ۳۴ – جدول اهمیت متغیرها



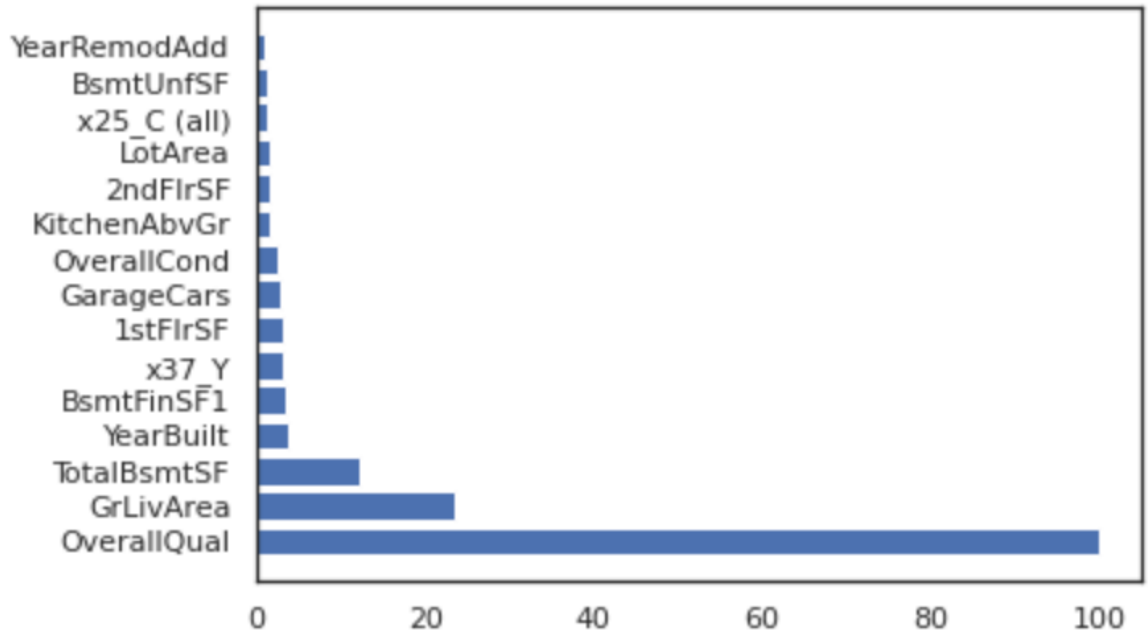
شکل ۳۵ – اهمیت ۱۵ ویژگی اول به صورت بار پلات

قابل ذکر است به دلیل زیاد بودن ویژگی‌ها همه‌ی آن‌ها روی یک بار پلات قابل نمایش نیستند. به همین دلیل، ۱۵ ویژگی پر اهمیت انتخاب شده و نمایش داده‌شده‌اند.

اهمیت ویژگی‌ها با استفاده از Decision Tree(CART) محاسبه شده و در نمودار زیر قابل مشاهده‌است.

	importance	importance_normalized
OverallQual	0.564884	100.000000
GrLivArea	0.132882	23.523773
TotalBsmtSF	0.069964	12.385582
YearBuilt	0.021023	3.721584
BsmtFinSF1	0.019541	3.459264
...	...	...
x26_Basment	0.000000	0.000000
x26_CarPort	0.000000	0.000000
x27_Ex	0.000000	0.000000
x28_No	0.000000	0.000000
x30_Slab	0.000000	0.000000

شکل ۳۶ – جدول اهمیت متغیرها با استفاده از **Decision Tree**



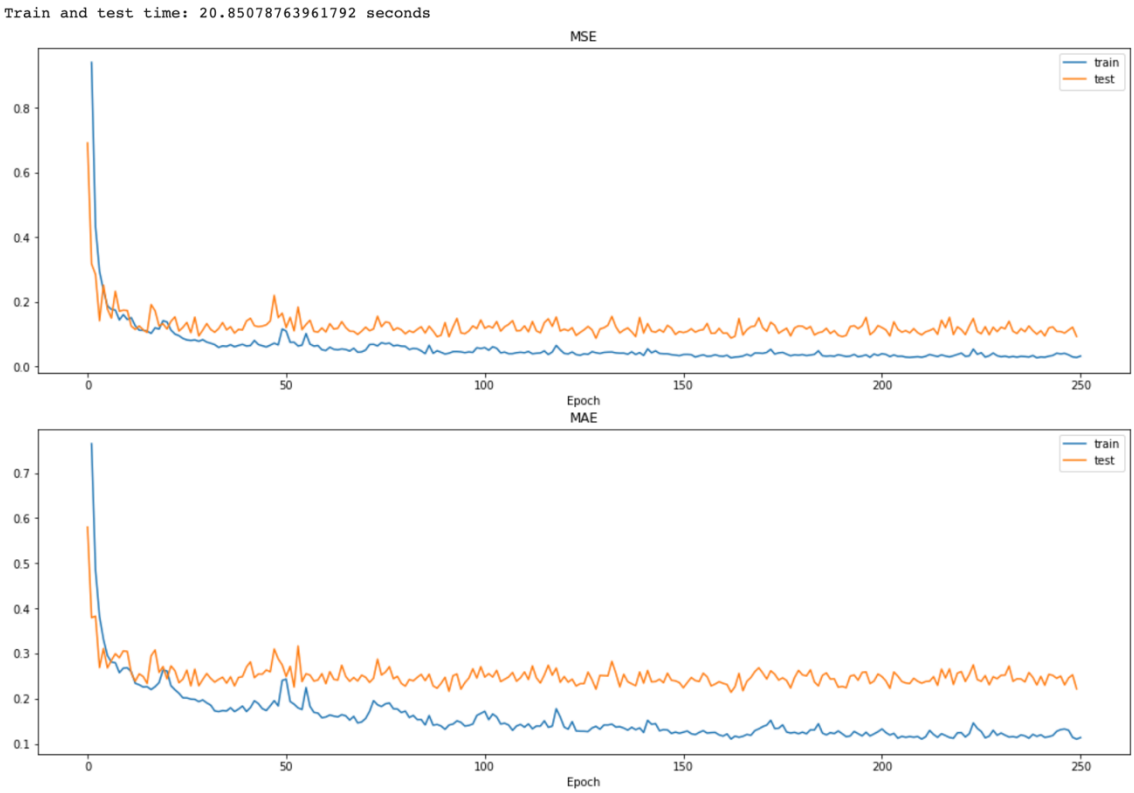
شکل ۳۷ – اهمیت ۱۵ ویژگی اول به صورت بار پلات

ج) ویژگی‌های حذف شده به ترتیب عبارتند از:

['x۱۳_AsphShn', 'x۱۳_WdShing', 'x۳۱_Unf', 'x۲۸_SWISU', 'x۱۱_GasA',
'x۳۰_CWD', 'x۳۰_Oth', 'x۳۲_Ex', 'x۲۴_Basment', 'x۲۸_Timber', 'x۲۲_۱.۵Unf',
'x۲۴_Attchd', 'x۸_Brk Cmn', 'x۸_ImStucc', 'x۲۷_Po', 'x۱۳_Plywood',
'x۱۳_VinylSd', 'x۱۱_OthW', 'x۳۰_New', 'x۲۸_BrkSide', 'x۲۸_Blmngtn',
'x۲۸_NPkVill', 'x۱۵_Gtl', 'x۲۲_SLvl', 'x۲۲_۱.۵Fin', 'MiscVal', 'x۳۸_Po', 'x۳۸_TA',
'x۲۵_Gar۲', 'x۸_AsphShn', 'x۸_VinylSd', 'x۱۳_ImStucc', 'x۳۳_RRAn', 'x۴۱_Ex',
'x۳۳_PosA', 'x۱۳_Stone', 'x۲_Fa', 'x۳_PConc', 'x۲۵_TenC', 'x۲۵_No',
'MasVnrArea', 'x۳۰_Con', 'x۱۳_HdBoard', 'x۳۵_No', 'x۲۱_No', 'x۳۵_GLQ',
'x۱۲_No', 'x۱۲_Ex', 'x۴۲_RRAn', 'x۲۲_SFoyer', 'x۲۸_SawyerW',
'BsmtHalfBath', 'x۲۹_AllPub', 'x۲۴_Detchd', 'x۲۵_Shed', 'x۱۸_Corner',
'x۲۶_Normal', 'x۱۷_Stone', 'x۱۳_AsbShng', 'x۸_Stucco', 'x۱۳_Stucco',
'x۵_SBrkr', 'x۵_FuseF', 'x۲۴_CarPort', 'x۱۶_۲fmCon', 'x۲۸_Blueste', 'MoSold',
'x۳۳_RRNe', 'x۳۳_Feedr', 'x۱۷_BrkFace', 'x۲۷_Gd', 'x۲۷_TA', 'x۱۹_Gd',
'OpenPorchSF', 'x۳۶_RM', 'x۳۱_Rec', 'x۳۱_LwQ', 'x۲۶_Alloca', 'x۱۴_Pave',
'x۲۵_Othr', 'x۱_HLS', 'x۱_Lvl', 'x۴۲_Artery', 'x۴۲_RRNn', 'x۲۰_No',
'x۴۲_Norm', 'x۳۴_Fa', 'x۳۴_TA', 'x۳۴_Ex', 'x۳۴_Gd', 'x۲۰_MnWw',
'x۲۸_Gilbert', 'x۲۱_Gd', 'x۲۱_TA', 'BsmtUnfSF', 'x۲_Gd', 'x۸_Wd Shng',
'x۳۲_Po', 'x۲۰_GdPrv', 'x۲۷_Ex', 'x۵_FuseP', 'x۲۶_AdjLand', 'LowQualFinSF',
'۲ndFlrSF', '۱stFlrSF', 'x۳۸_Ex', 'x۳۷_Mod', 'x۳۰_ConLw', 'x۲۲_۲.۵Unf',
'x۲۰_MnPrv', 'x۸_CBlock', 'x۱۳_CBlock', 'x۳۲_Fa', 'x۲۳_Gd', 'x۲۳_Ex', 'x۲_TA',
'x۲۶_Partial', 'x۳۳_RRNn', 'x۱۱_Wall', 'x۸_Stone', 'x۳۱_GLQ', 'x۳۱_ALQ',
'YrSold', 'x۳۵_BLQ', 'x۳۵_ALQ', 'x۳۵_Rec', 'BedroomAbvGr', 'x۸_MetalSd',
'x۲۷_Fa', 'x۲۸_CollgCr', 'x۱۶_Duplex', 'x۱۶_۱Fam', 'MSSubClass',
'x۲۲_۱Story', 'x۳_Stone', 'x۸_Other', 'Fireplaces', 'x۸_Plywood',
'x۲۴_BuiltIn', 'x۱۱_Floor', 'x۲۱_Fa', 'x۳_Slab', 'x۳۶_RH', 'x۱۷_None', 'x۳۶_FV',
'x۱۷_BrkCmn', 'x۱۰_WdShake', 'x۱۰_Roll', 'x۱۰_Tar&Grv', 'x۱۰_CompShg',
'x۱۰_WdShngl', 'x۱۷_No', 'x۳۷_Maj۱', 'x۰_IR۲', 'x۰_IR۳', 'x۰_Reg', 'x۹_Y',
'x۲۸_Sawyer', 'x۱۴_Grvl', 'x۸_AsbShng', 'x۴_Pave', 'x۴_Grvl', 'x۴_No',
'x۱۲_Fa', 'x۳۶_RL', 'x۵_FuseA', 'x۲۹_NoSeWa', 'x۴۲_Feedr', 'x۰_IR۱',
'x۱۸_CulDSac', 'x۳۹_Shed', 'x۷_RFn', 'x۷_Fin', 'x۶_Y', 'x۴۰_Ex', 'x۶_N', 'x۶_P',
'x۷_Unf', 'x۳۵_LwQ', 'BsmtFinSF۲', 'x۲۸_NWAmes', 'x۲۸_NAmes',
'x۲۲_۲Story', 'x۱_Bnk', 'x۱۵_Mod', 'x۱_Low', 'x۳۳_PosN', 'x۳۰_ConLD',
'x۸_HdBoard', 'x۱۳_CemntBd', 'x۸_CmentBd', 'x۵_Mix', 'x۲۸_ClearCr',
'x۳۸_Fa', 'x۲۱_Po', 'GarageCars', '۳SsnPorch', 'x۸_Wd Sdng', 'x۱۳_Wd Sdng',
'x۲۳_Fa', 'x۲۸_Mitchel', 'x۳۱_BLQ', 'x۱۸_FR۳', 'TotRmsAbvGrd', 'x۳۰_ConLI',
'x۲۶_Family', 'x۳۲_Gd']

در خروجی کد در هر مرحله ویژگی حذف‌شده به همراه p-value آن گزارش شده‌است.
زمان انجام این کار ۱۷.۶۵ ثانیه شده‌است.

مدل با داده‌ی کاهش بعد یافته آموزش یافته و خطا و مدت زمان آموزش به صورت زیر است. مدت زمان آموزش و آزمون برابر ۲۰.۸۵ ثانیه شده‌است.



شکل ۳۸ – نمودار خطاها در هر **epoch** به ازای داده‌ی آموزش و تست به همراه زمان این کار


```
Minimum MSE: 0.08712012444933255
```

شکل ۳۹ – نمودار پیش‌بینی بر حسب مقادیر واقعی و عدد کمینه خطا

د) در این قسمت ابتدا با استفاده از روش PCA ابعاد به ۵۰ بعد کاهش یافته‌است و سپس، به بهترین مدل سوال ۲ ورودی داده‌شده‌است و شبکه آموزش داده‌شده‌است. زمان اجرای PCA و آموزش شبکه به ترتیب برابر ۰.۰۰۵۹ ثانیه و ۱.۳۹ ثانیه می‌باشد. دقت و خطای مدل بر روی داده‌ی تست به صورت زیر به دست آمد.

```
Test Accuracy: 0.9285714285714286
Classificaiton report:
              precision    recall  f1-score   support

           0       0.95      0.90      0.93        21
           1       0.91      0.95      0.93        21

    accuracy                           0.93        42
   macro avg       0.93      0.93      0.93        42
weighted avg       0.93      0.93      0.93        42

Loss: 0.4035155475139618
Confusion Matrix:
[[19  2]
 [ 1 20]]
```

شکل ۴۰ – دقت و خطای مدل با **PCA** بر روی داده‌ی تست

ه) در این قسمت ابتدا با استفاده از شبکه Autoencoder ابعاد به ۵۰ بعد کاهش یافته‌است و سپس، به بهترین مدل سوال ۲ ورودی داده‌شده‌است و شبکه آموزش داده‌شده‌است. زمان اجرای Autoencoding و آموزش شبکه به ترتیب برابر ۵.۳۷ ثانیه و ۱.۳۴ ثانیه می‌باشد. دقت و خطای مدل بر روی داده‌ی تست به صورت زیر به دست آمد.

```
Test Accuracy: 0.8333333333333334
Classificaiton report:
              precision    recall  f1-score   support

           0       0.76      0.81      0.79        16
           1       0.88      0.85      0.86        26

    accuracy                           0.83        42
   macro avg       0.82      0.83      0.83        42
weighted avg       0.84      0.83      0.83        42

Loss: 0.5030772686004639
Confusion Matrix:
[[13  3]
 [ 4 22]]
```

شکل ۴۱ – دقت و خطای مدل مدل با **AutoEncoding** بر روی داده‌ی تست

و) جدول زیر برای مقایسه سه حالت شبکه‌ی سوال دو شکل گرفته‌است.

جدول ۱ – مقایسه دقت شبکه‌های مختلف

	دقت داده تست	خطای داده تست	زمان (ثانیه)
بهترین شبکه سوال ۲	٪ ۹۲.۸۵	۰.۱۹۳۰	۱.۷۹۷۰
AutoEncoder	٪ ۸۳.۳۳	۰.۵۰۳۱	۶.۷۰۷۲
PCA	٪ ۹۲.۸۵	۰.۴۰۳۵	۱.۳۹۰۶

همانطور که مشاهده می‌کنید، با کاهش ابعاد زمان اجرای آموزش شبکه کاهش یافته‌است. حتی در روش PCA همراه با کاهش ابعاد سریع از بهترین شبکه سوال ۲ عمل کرده‌است. در روش AutoEncoder به دلیل نیاز آموزش AutoEncoder که خود ۵۰ epoch است این فرآیند به طول انجامیده‌است وگرنه خود آموزش شبکه کوتاه‌تر بوده‌است. بهترین شبکه سوال ۲ دقت قابل قبولی ارائه داده‌است. با کاهش ابعاد معمولا قابلیت تعمیم شبکه بالاتر می‌رود اما در اینجا این اتفاق رخ نداده‌است و در روش PCA همان مانده‌است و روش AutoEncoder کاهش یافته‌است. اما این ممکن است به دلیل پارامتر دیگر مسئله یا حالت خاص تقسیم داده‌ی تست باشد.
با کاهش ابعاد به مهم‌ترین ابعاد علاوه بر کاهش طول آموزش در مسائل طولانی، معمولا قدرت تعمیم شبکه نیز بالاتر می‌رود چرا که به ویژگی‌های بی‌ارزش حساس نمی‌شود.