به نام خدا

دانشگاه تهران

دانشکده مهندسی برق و کامپیوتر

درس شبکه‌های عصبی و یادگیری عمیق

تمرین سری 2

نام و نام خانوادگی	حمید رضا کاشانی
شماره دانشجویی	810100441
تاریخ ارسال گزارش	1401/1/19

فهرست گزارش سوالات (لطفاً پس از تکمیل گزارش، این فهرست را به‌روز کنید.)

سوال ۱ –(MLP)Classification

الف)روش های متفاوت تقسیم داده ها در ماشین لرنینگ:(منبع سایت howsam.org)

Resubstitution

ساده‌ترین و نه بهترین روش برای تعیین خطای مدل در الگوریتم‌های یادگیری ماشین، روش Resubstitution است. در این روش تمامی داده‌ها برای آموزش مدل استفاده می‌شوند. تفاوت خروجی مدل و خروجی واقعی به عنوان خطا در نظر گرفته شده و سپس میانگین خطای همه داده‌ها به عنوان خطای کل در نظر گرفته می‌شود. به این خطا، خطای Resubstitution گفته می‌شود.

Holdout

در این روش، داده‌ها به دو دسته train و test تقسیم می‌شوند. این تقسیم می‌تواند به صورت 60/40 ، 70/30 یا 80/20 باشد. بنابراین مدل مورد نظر روی داده‌های train آموزش دیده و روی داده‌های test مورد ارزیابی قرار می‌گیرد. به این روش، اعتبارسنجی Holdout گفته می‌شود. در روش Holdout، اگر کلاس‌های مختلف در هر گروه test یا train توزیع یکسانی نداشته باشند، مدل، درست آموزش نخواهد دید. از این جهت کلاس‌ها باید توزیع یکسانی در هر دو گروه train و test داشته باشند. به این پروسه، stratification گفته می‌شود.

K-Fold Cross-Validation

در این روش، داده‌ها به k بخش تبدیل می‌شوند. هرکدام از داده‌ها به صورت تصادفی در یکی از این k بخش قرار می‌گیرند. در این روش، آموزش و تست k بار تکرار می‌شوند. در هر تکرار k-1 بخش به عنوان داده train و یکی به عنوان test در نظر گرفته می‌شود. خطای مدل برابر با میانگین مدل در k تکرار است.

Leave-One-Out Cross-Validation (LOOCV)

در این تکنیک، از تمامی داده‌ها به جز یک داده برای آموزش و از داده باقی مانده برای تست مدل استفاده می‌شود. این فرآیند N بار تکرار می‌شود که N تعداد داده‌ها را نشان می‌دهد. مزیت این روش این است که از تمامی داده‌ها برای آموزش و تست مدل استفاده خواهد شد. نرخ خطای مدل در این روش برابر با میانگین نرخ خطا در هر تکرار است. شکل زیر تکنیک LOOCV را نشان می‌دهد.

Random Sub-sampling

در این تکنیک تعدادی از داده‌ها به صورت تصادفی انتخاب شده و داده‌های تست را تشکیل می‌دهند. باقیمانده داده‌ها نیز برای آموزش مورد استفاده قرار می‌گیرند. نرخ خطای مدل در این روش نیز برابر با میانگین نرخ خطا در هر تکرار است. شکل زیر تکنیک Random Subsampling را نشان می‌دهد.

Bootstrapping

در این تکنیک، داده‌های آموزش به صورت تصادفی و با استفاده از جایگذاری انتخاب می‌شوند. نمونه‌هایی که انتخاب نشده‌اند نیز برای تست مورد استفاده قرار می‌گیرد. در این روش، بر خلاف روش k-fold، تعداد نمونه‌های انتخاب شده در هر تکرار متفاوت است. نرخ خطای مدل در این روش نیز برابر با میانگین نرخ خطا در هر تکرار است. شکل زیر تکنیک Bootstrapping را نشان می‌دهد.

ما ابتدا داده ها را به دو دسته test و train تقسیم می کنیم و سپس داده های train را به به صورت رندوم تقسیم به دو قسمت train و validation تبدیل میکنیم و روی داده های train مدل را آموزش می دهیم

داده های ورودی ما خودش به صورت دو دسته trainوtest به داده می شود و در هنگام تمرین دادن مدل ها را با 0.2 validation فیت می کنیم.

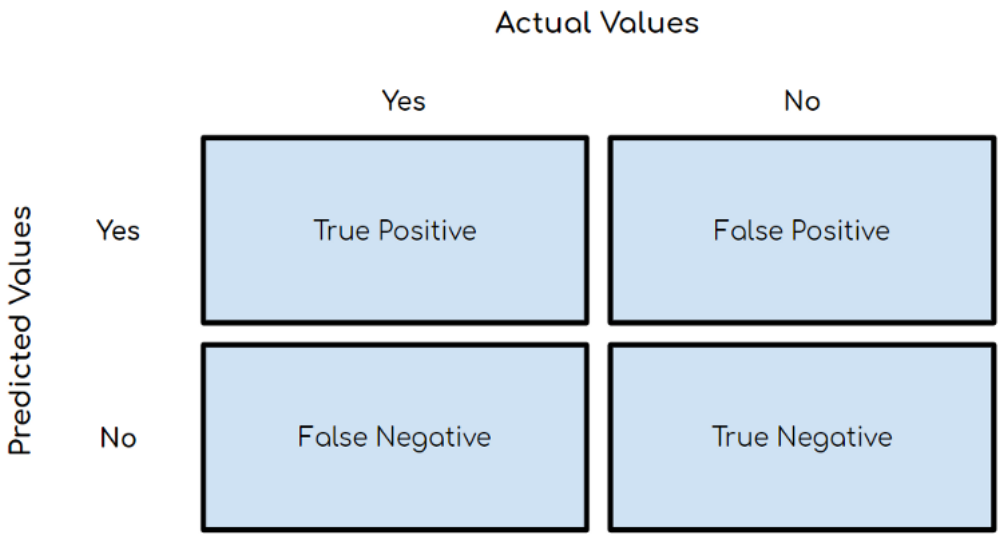
50000داده برای تمرین و 10000داده برای تست داریم

ب)

ماتریس آشفتگی :

یک ماتریس در هم ریختگی که به عنوان ماتریس خطا نیز شناخته می‌شود، یک جدول خلاصه‌شده برای ارزیابی عملکرد یک مدل طبقه‌بندی است. تعداد پیش‌بینی‌های صحیح و نادرست با مقادیر شمارش خلاصه می‌شوند و بر اساس هر کلاس شکسته می‌شوند.

در زیر تصویری از ساختار یک ماتریس در هم ریختگی ۲×۲ آورده شده‌است. برای مثال، اجازه دهید فرض کنیم که ده نمونه وجود داشتند که در آن‌ها یک مدل طبقه‌بندی «بله» را پیش‌بینی می‌کرد که در آن ارزش واقعی نیز «بله» بود. سپس شماره ده در گوشه سمت چپ بالا در ربع مثبت حقیقی قرار می‌گرفت. این منجر به برخی اصطلاحات کلیدی می‌شود:



ساختار ماتریس در هم ریختگی ۲×۲

- مثبت (P): مشاهده مثبت است. (مثلا: سگ هست)
- منفی (N): مشاهده مثبت نیست. (مثلا: سگ نیست)
- مثبت واقعی (TP) : نتایج زمانی حاصل می‌شوند که مدل به درستی کلاس مثبت را پیش‌بینی می‌کند.
- منفی واقعی (TN) : در جایی که مدل به درستی کلاس منفی را پیش‌بینی می‌کند، نتایج به دست می‌آیند.
- مثبت کاذب (FP) : که همچنین خطای نوع ۱ نیز نامیده می‌شود، نتیجه‌ای که در آن مدل به اشتباه کلاس مثبت را هنگامی که در واقع منفی است، پیش‌بینی می‌کند.
- منفی کاذب (FN) : که همچنین یک خطای نوع ۲ نیز نامیده می‌شود، نتیجه‌ای که در آن مدل به طور نادرست کلاس منفی را وقتی که واقعا مثبت است پیش‌بینی می‌کند.

معیار Recall یا یادآوری:

حداکثر مقدار این معیار یک ویا ۱۰۰ درصد و حداقل مقدار آن صفر است و هرچه مواردی که ما انتظار داشتیم پیش بینی شوند ولی برنامه پیش بینی نکرده‌است که به آن False Negative می‌گوییم نسبت به پیش بینی‌های درست یا True Positive بیشتر باشد مقدار Recall کمتر خواهد شد.

$$Recall = \frac{TP}{TP + FN}$$

معیار Precision یا دقت:

حداکثر مقدار این معیار یک ویا ۱۰۰ درصد و حداقل مقدار آن صفر است و هرچه مواردی که برنامه به غلط پیش بینی کرده است که به آن False Positive می‌گوییم نسبت به پیش بینی‌های درست یا True Positive بیشتر باشد مقدار Precision کمتر خواهد شد.

$$Precision = \frac{TP}{TP + FP}$$

معیار f1-score:

زمانی که می‌خواهید معیار ارزیابی شما میانگینی از دو مورد قبلی باشد یعنی همان Recall یا Precision می‌توانید از میانگین هارمونیک این دو معیار استفاده کنید که به آن معیار f1-score می‌گویند.

$$F1 = \frac{2}{\frac{1}{Recall} \cdot \frac{1}{Precision}} = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

تابع برای سیاه سفید کردن عکس:

```
def rgb_to_gray(img):
  R, G, B = img[:,:,:,0], img[:,:,:,1], img[:,:,:,2]
  imgGray = 0.2989 * R + 0.5870 * G + 0.1140 * B
  return imgGray
```

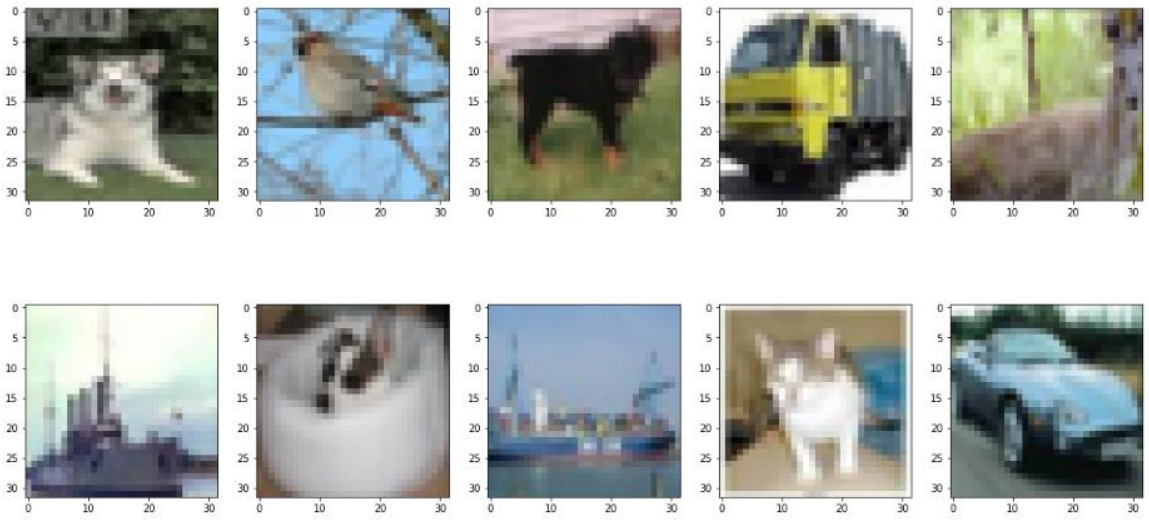
پیش پردازش داده ها:

```
from keras.datasets import cifar10
(x_train_rgb,y_train),(x_test_rgb,y_test)=cifar10.load_data()
x_train_gray =rgb_to_gray(x_train_rgb)
x_test_gray = rgb_to_gray(x_test_rgb)
x_test_gray = x_test_gray/255
x_train_gray = x_train_gray/255
x_train = x_train_gray.reshape(x_train_gray.shape[0],32*32)
x_test = x_test_gray.reshape(x_test_gray.shape[0],32*32)
y_train = tf.keras.utils.to_categorical(y_train,10)
```

```
y_test = tf.keras.utils.to_categorical(y_test,10)
```

ترسیم 10 عکس از داده ها:

```
figure = plt.figure(figsize=(20, 10))
for i in range(10):
  photo= np.random.choice(x_train_rgb.shape[0], size=15, replace=Fals
e)
  ax = figure.add_subplot(2, 5, i+1)
  ax.imshow(np.squeeze(x_train_rgb[photo[i]]))
```

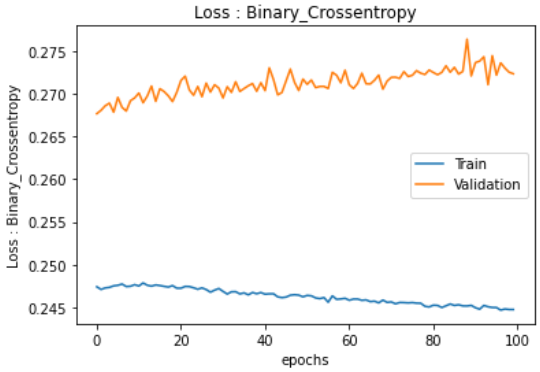


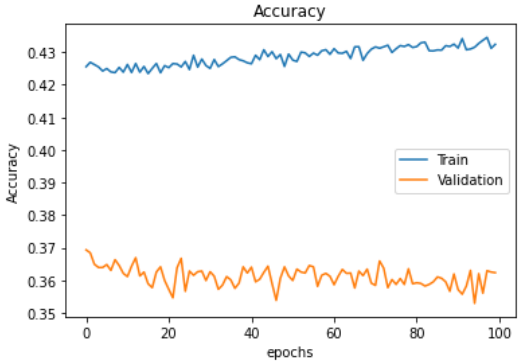
ت)3 شبکه با لایه های متفاوت

شبکه اول: دو لایه 32 نورون ، زمان ترین 2:33

```
model1 = Sequential([
    layers.Input(shape=(32*32)),
    layers.Dense(32,activation='relu'),
    layers.Dense(32,activation='relu'),
    #layers.Dense(10,activation='relu'),
    #layers.Dense(3,activation='relu'),
    layers.Dense(10,activation='softmax')
])
model1.compile(
    optimizer='adam',
    loss='BinaryCrossentropy',
    metrics=['accuracy']
    #metrics=['mae','accuracy']
)
train_model1 = model1.fit(x_train,y_train,epochs=100, validation_spli
t=0.2,batch_size=128)
```

نمودار ها:





و خطا و دقت به ترتیب:

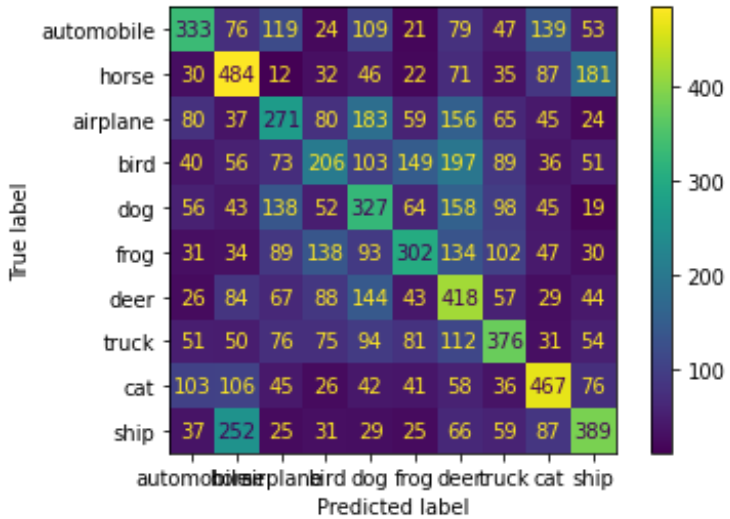
0.2759501338005066

0.3573000133037567

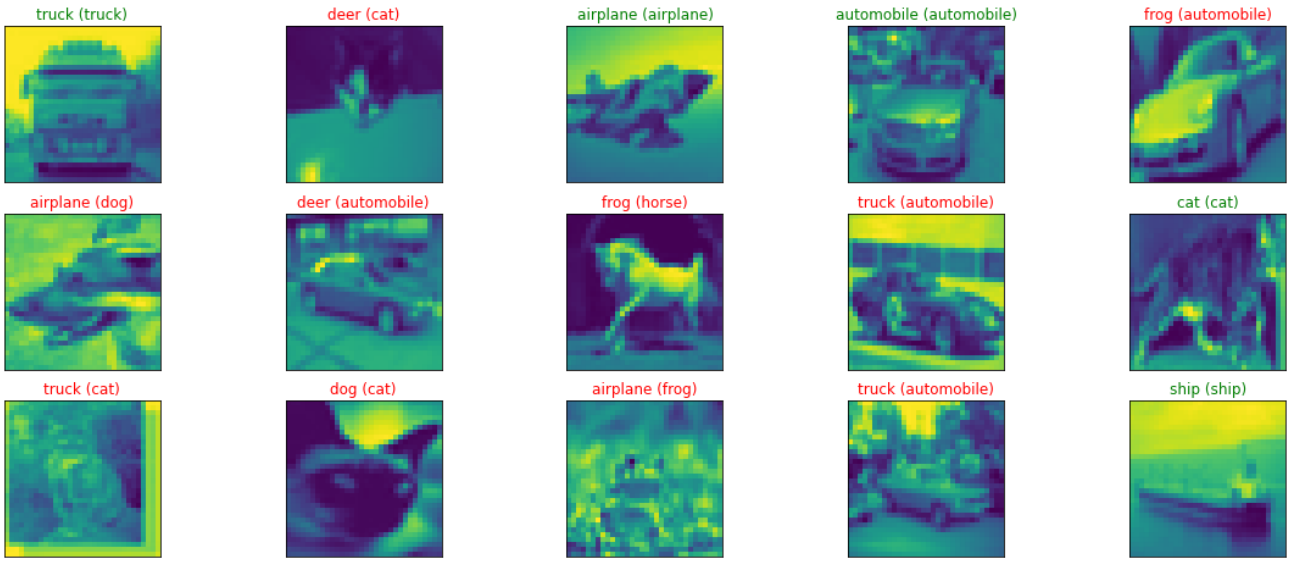
و همینطور این 4 پارامتر precision_recall_fscore_support به ترتیب:

```
[0.42312579 0.39607201 0.29617486 0.27393617 0.27948718 0.37422553
 0.28847481 0.39004149 0.46100691 0.42236699]
[0.333 0.484 0.271 0.206 0.327 0.302 0.418 0.376 0.467 0.389]
[0.37269166 0.43564356 0.28302872 0.23515982 0.30138249 0.33425567
 0.34136382 0.38289206 0.4639841  0.4049974 ]
[1000 1000 1000 1000 1000 1000 1000 1000 1000 1000]
```

ماتریس آشفتگی:

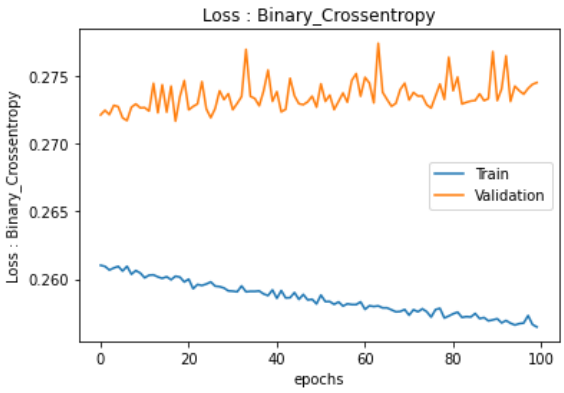


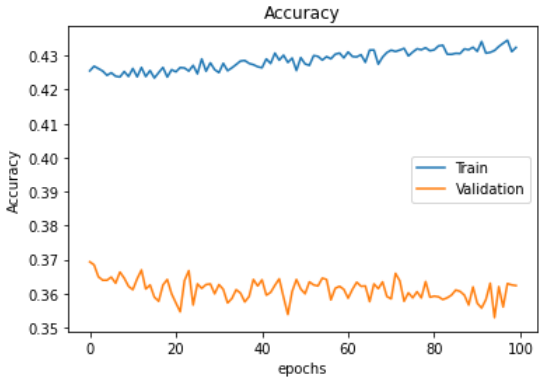
و همینطور نتیجه رندوم از این شبکه:



شبکه دوم: یه لایه 32 و یه لایه 10 با زمان ترین 02:22

```
model2 = Sequential([
    layers.Input(shape=(32*32)),
    layers.Dense(32,activation='relu'),
    layers.Dense(10,activation='relu'),
    #layers.Dense(10,activation='relu'),
    #layers.Dense(3,activation='relu'),
    layers.Dense(10,activation='softmax')
])
model2.compile(
    optimizer='adam',
    loss='BinaryCrossentropy',
    metrics=['accuracy']
    #metrics=['mae','accuracy']
)
train_model2 = model2.fit(x_train,y_train,epochs=100, validation_spli
t=0.2,batch_size=128)
```





خطا و دقت به ترتیب:

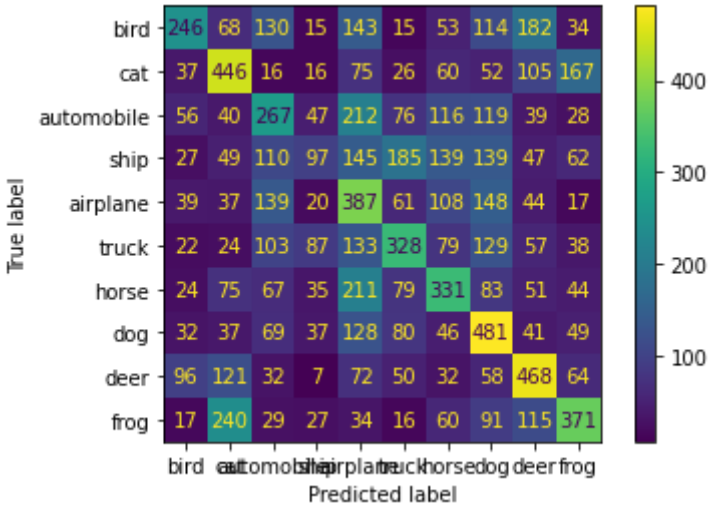
0.2759501338005066

0.3573000133037567

همینطور این 4 پارامتر precision_recall_fscore_support به ترتیب:

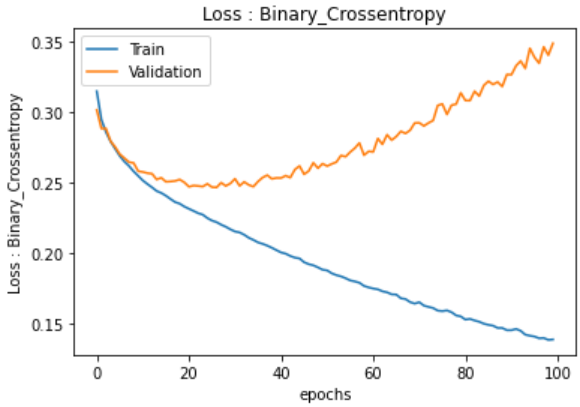
```
[0.41275168 0.39226033 0.27754678 0.25       0.2512987  0.3580786
 0.32324219 0.34016973 0.4073107  0.42448513]
[0.246 0.446 0.267 0.097 0.387 0.328 0.331 0.481 0.468 0.371]
[0.30827068 0.41740758 0.27217125 0.13976945 0.30472441 0.34237996
 0.3270751  0.3985087  0.43555142 0.3959445 ]
[1000 1000 1000 1000 1000 1000 1000 1000 1000 1000]
```

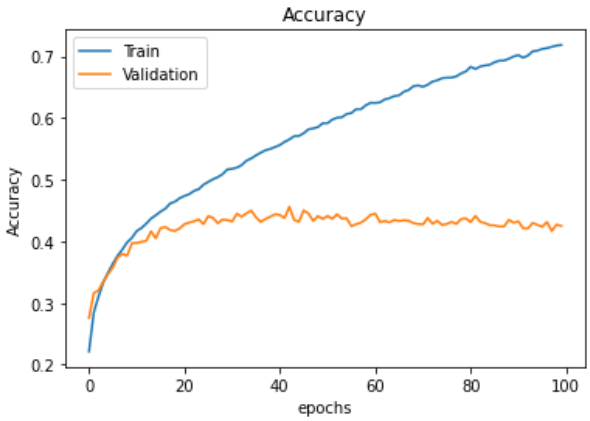
ماتریس آشفتگی:



شبکه سوم: لایه 32*32 و لایه 32 و تایم ترین 11:39

```
model3 = Sequential([
    layers.Input(shape=(32*32)),
    layers.Dense(32*32,activation='relu'),
    layers.Dense(32,activation='relu'),
    #layers.Dense(10,activation='relu'),
    #layers.Dense(3,activation='relu'),
    layers.Dense(10,activation='softmax')
])
model3.compile(
    optimizer='adam',
    loss='BinaryCrossentropy',
    metrics=['accuracy']
    #metrics=['mae','accuracy']
)
train_model3 = model3.fit(x_train,y_train,epochs=100, validation_spli
t=0.2,batch_size=128)
```





خطا و دقت به ترتیب:

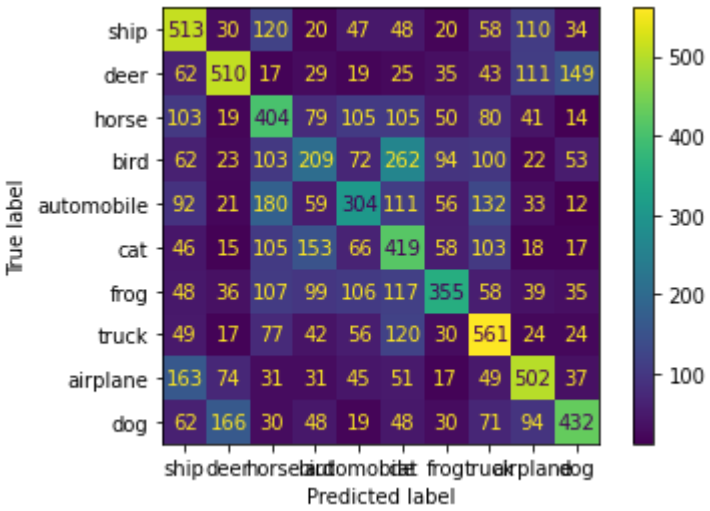
0.35073500871658325

0.41819998621940613

همینطور این 4 پارامتر precision_recall_fscore_support به ترتیب:

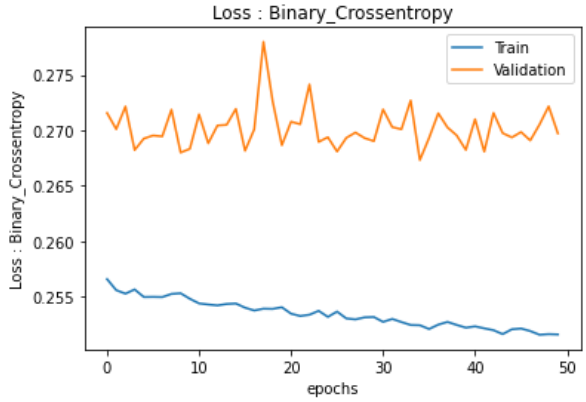
```
[0.4275     0.55982437 0.34412266 0.27178153 0.36233611 0.32082695
 0.47651007 0.44701195 0.50503018 0.53531599]
[0.513 0.51  0.404 0.209 0.304 0.419 0.355 0.561 0.502 0.432]
[0.46636364 0.53375196 0.37166513 0.23629169 0.33061446 0.36339983
 0.40687679 0.49756098 0.50351053 0.47814056]
[1000 1000 1000 1000 1000 1000 1000 1000 1000 1000]
```

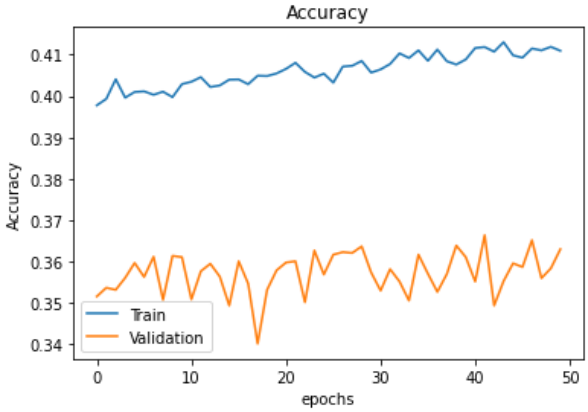
ماتریس آشفتگی:



د)مدل اول شبکه قبل مدل خوبی بود

32: زمان آموزش 03:37

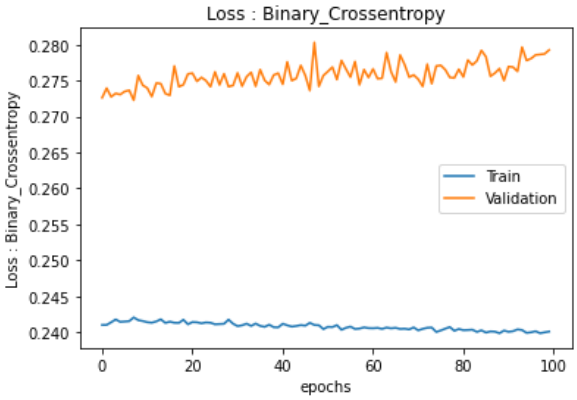


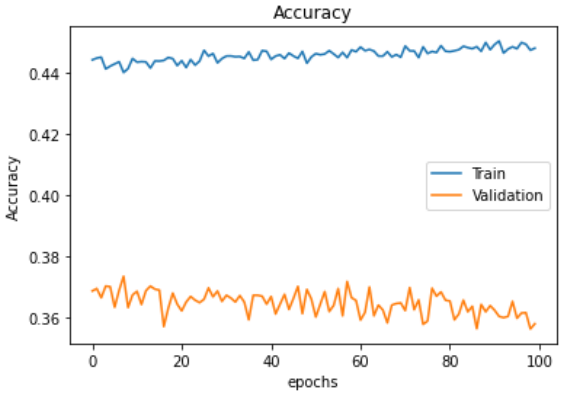


0.2702089548110962

0.3594000041484833

64: زمان آموزش 02:22

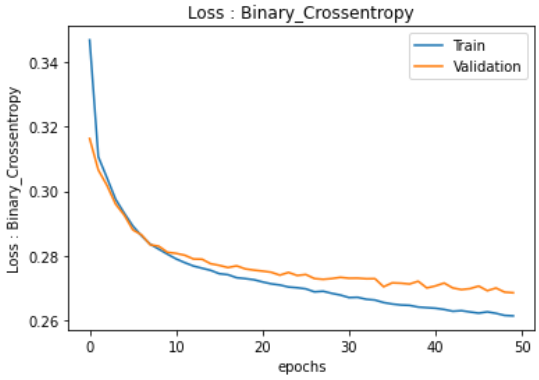


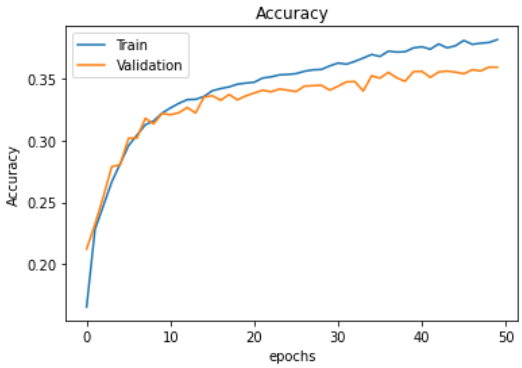


0.2691643238067627

0.35989999771118164

128: زمان آموزش 01:09

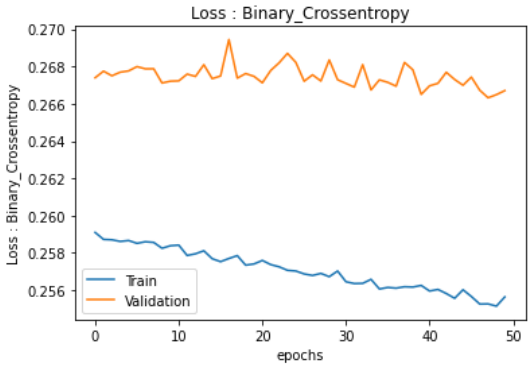


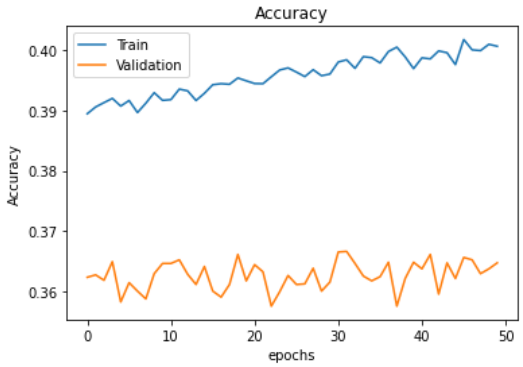


0.2681870460510254

0.36309999227523804

512: زمان آموزش 00:41



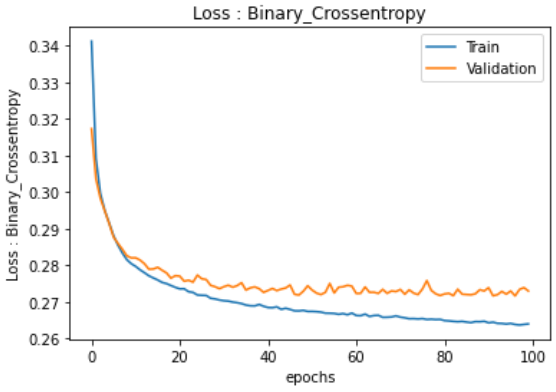


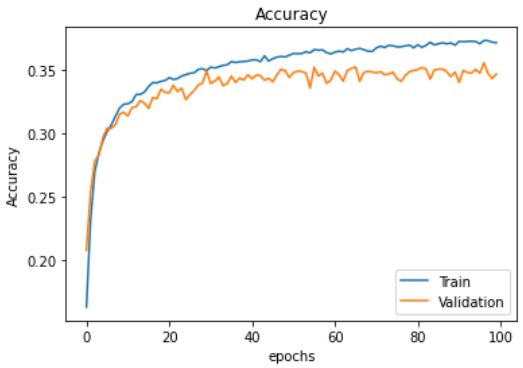
0.2666318714618683

0.37070000171661377

ه) کل توابع نورون ها رو عوض می کنیم و شبکه یک استفاده می شود

Relu: زمان اجرا 01:58

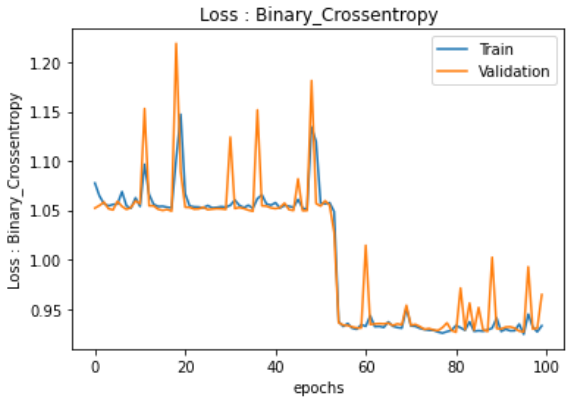


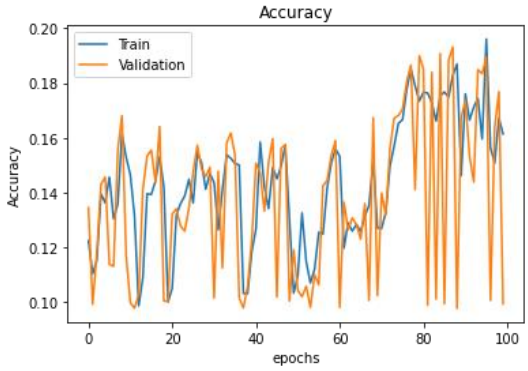


0.2720027267932892

0.3517000079154968

TanH: زمان اجرا 02:00

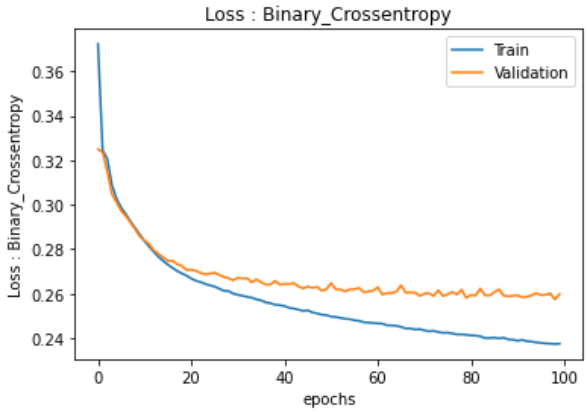


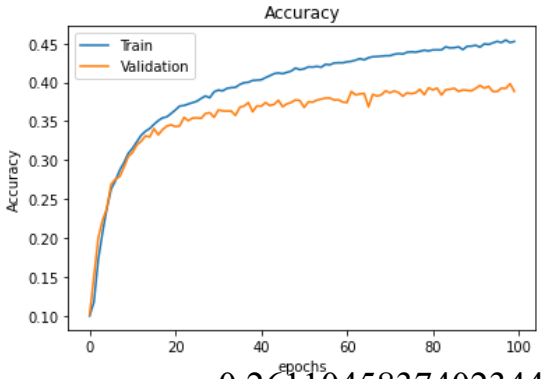


0.9621915817260742

0.09960000216960907

Sigmoid : زمان اجرا 02:22

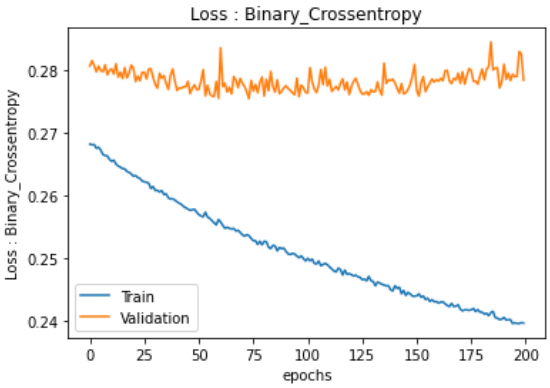


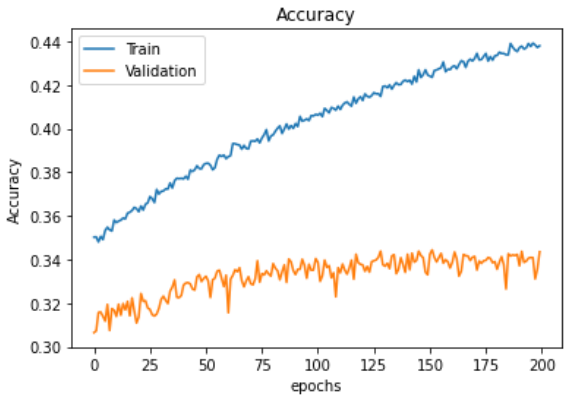


0.2611045837402344

0.38449999690055847

Softmax : رمان اجرا 03:22



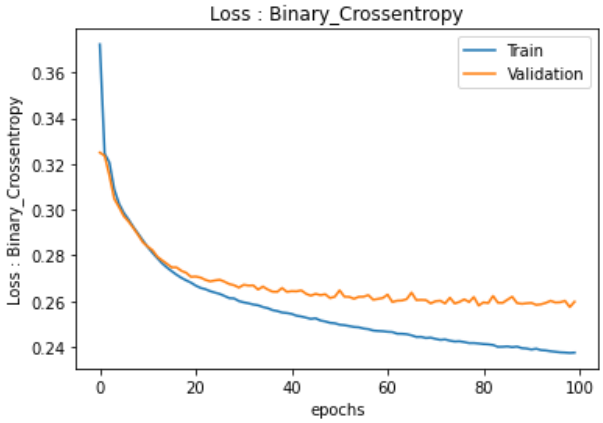


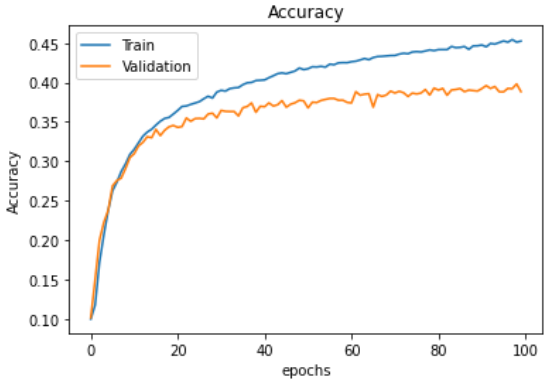
0.27904704213142395

0.3411000072956085

و) همان شبکه یک

: BinaryCrossentropy

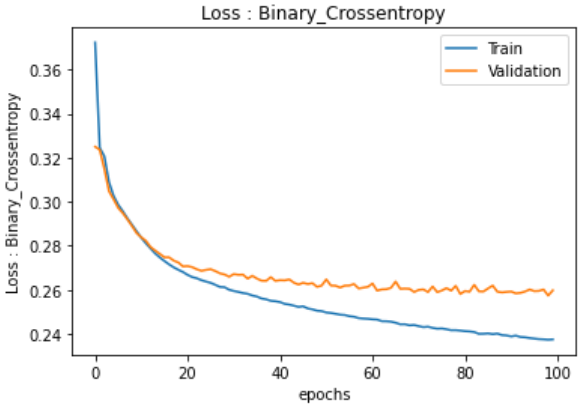


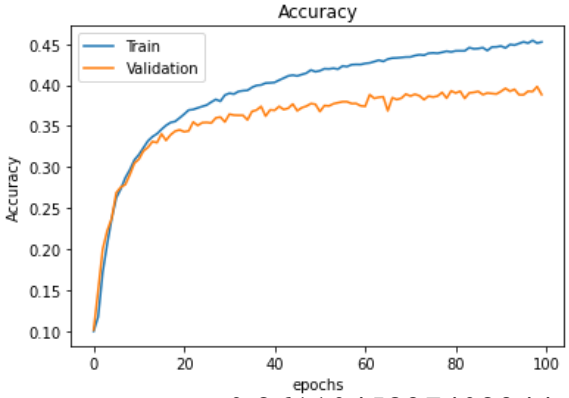


0.2611045837402344

0.38449999690055847

: CategoricalCrossentropy

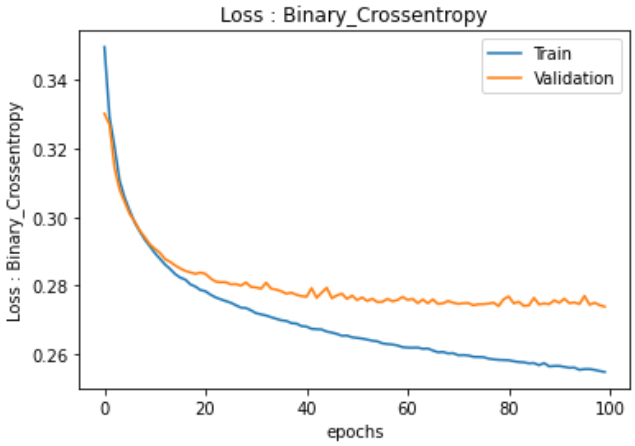


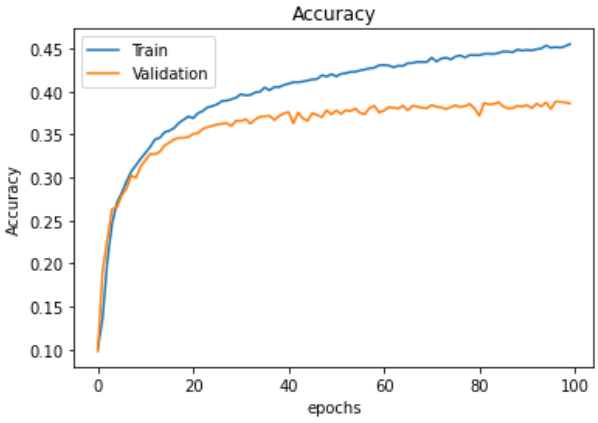


0.2611045837402344

0.38449999690055847

: Poisson



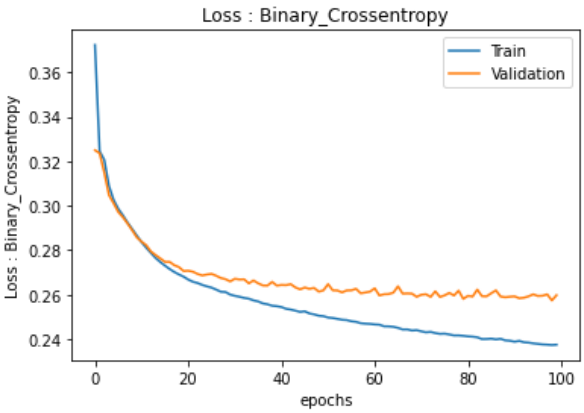


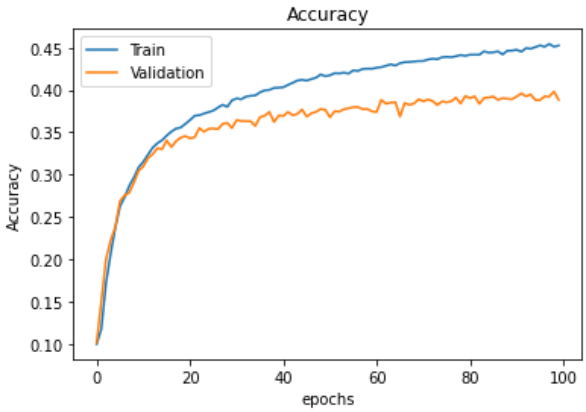
0.2740287184715271

0.3871000111103058

ج)همان شبکه شماره یک

: Adam

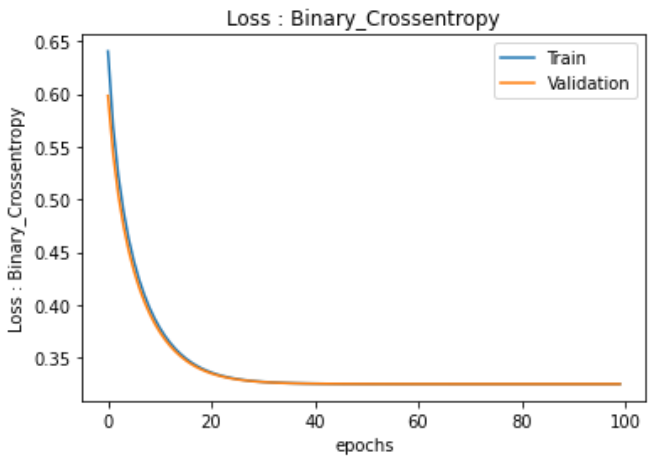


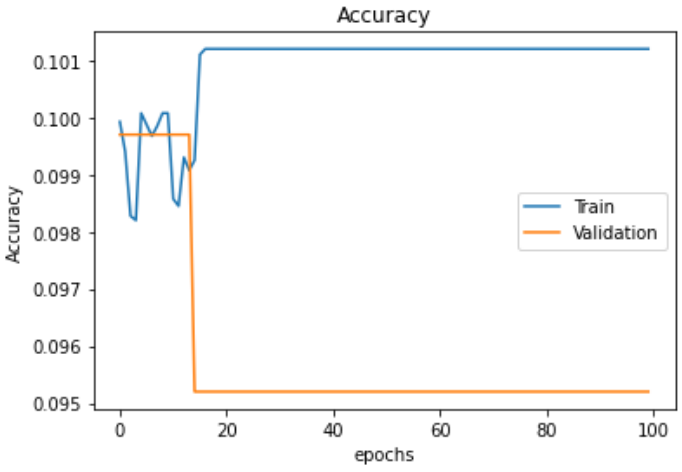


0.2611045837402344

0.38449999690055847

: Ftrl

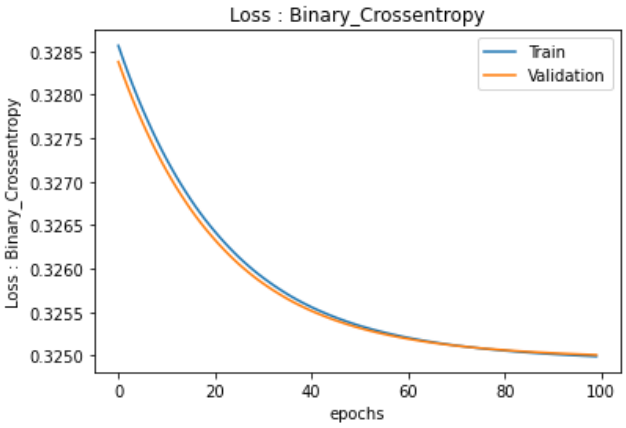


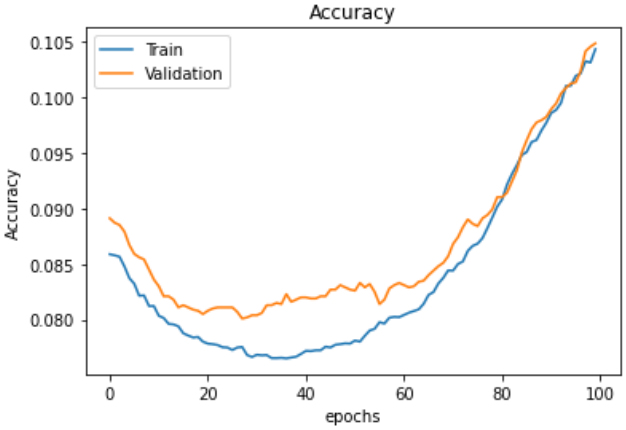


0.3251250088214874

0.10000000149011612

: Adadelta





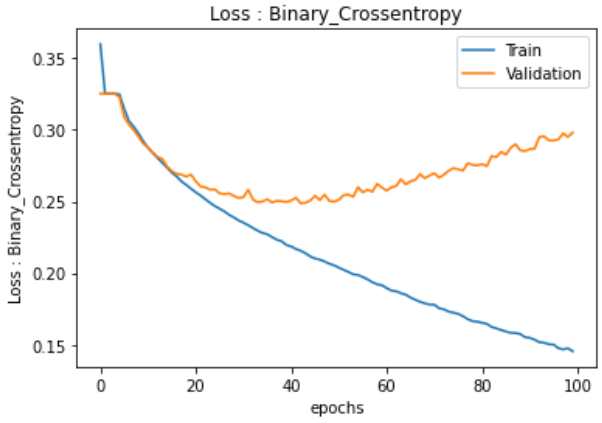
0.3249925971031189

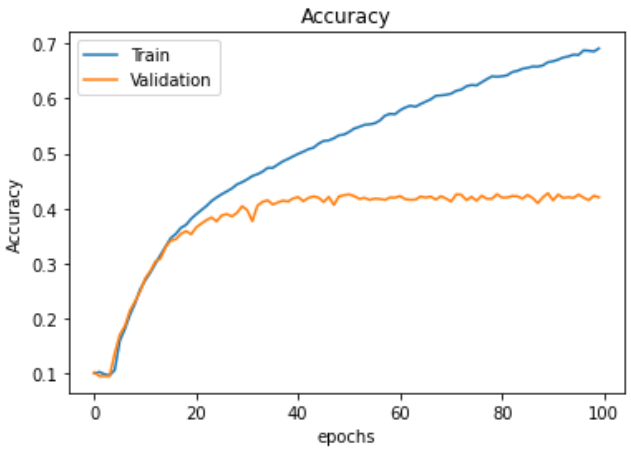
0.10350000113248825

ح) افزودن لایه به شبکه

4[1 لایه 32*10و32و32و32: زمان 06:22

```
model1_sigmoid_1 = Sequential([
    layers.Input(shape=(32*32)),
    layers.Dense(32*10,activation='sigmoid'),
    layers.Dense(32,activation='sigmoid'),
    layers.Dense(32,activation='sigmoid'),
    layers.Dense(32,activation='sigmoid'),
    layers.Dense(10,activation='sigmoid')
])
model1_sigmoid_1.compile(
    optimizer='adam',
    loss='BinaryCrossentropy',
    metrics=['accuracy']
    #metrics=['mae','accuracy']
)
```



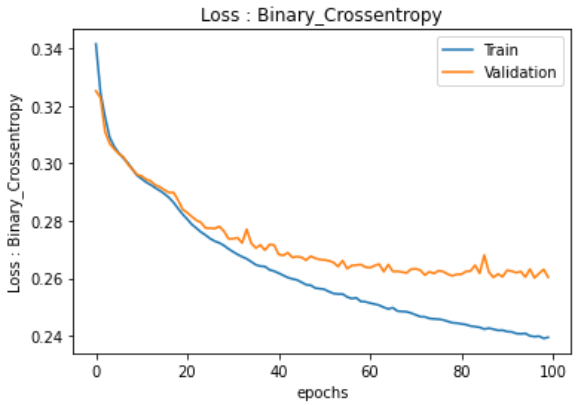


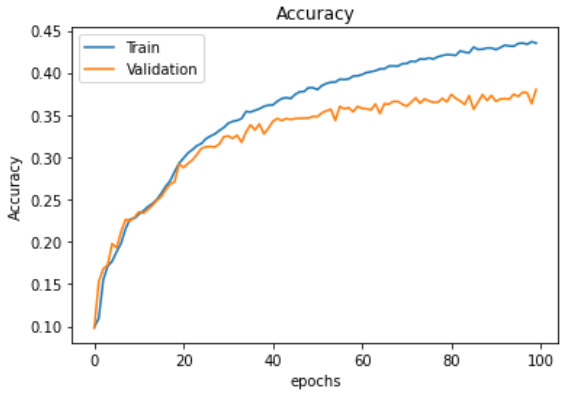
0.30024227499961853

0.41929998993873596

4)لایه 32و32و32و32 : زمان 03:22

```
model1_sigmoid_2 = Sequential([
    layers.Input(shape=(32*32)),
    layers.Dense(32,activation='sigmoid'),
    layers.Dense(32,activation='sigmoid'),
    layers.Dense(32,activation='sigmoid'),
    layers.Dense(32,activation='sigmoid'),
    layers.Dense(10,activation='sigmoid')
])
model1_sigmoid_2.compile(
    optimizer='adam',
    loss='BinaryCrossentropy',
    metrics=['accuracy']
    #metrics=['mae','accuracy']
)
```





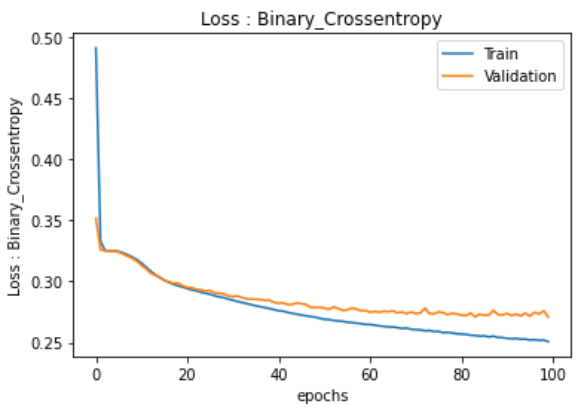
0.2622953951358795

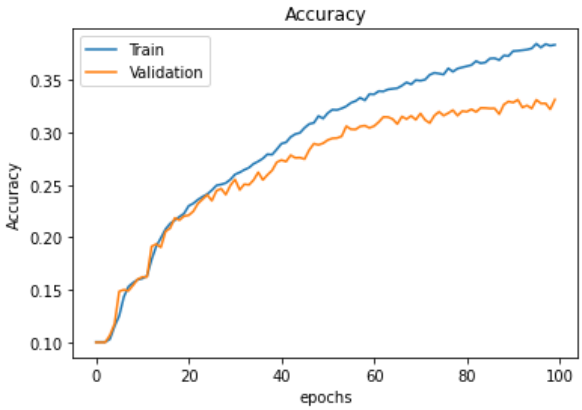
0.37700000405311584

3) 4 لایه 32و32و10و10 : زمان 03:22

```
model1_sigmoid_3 = Sequential([
    layers.Input(shape=(32*32)),
    layers.Dense(32,activation='sigmoid'),
    layers.Dense(32,activation='sigmoid'),
    layers.Dense(10,activation='sigmoid'),
    layers.Dense(10,activation='sigmoid'),
    layers.Dense(10,activation='sigmoid')
])
model1_sigmoid_3.compile(
```

```
    optimizer='adam',
    loss='BinaryCrossentropy',
    metrics=['accuracy']
    #metrics=['mae','accuracy']
)
```





0.27165040373802185

0.33079999685287476

ط) انتخاب تعداد لایه بیشترین تاثیر را دارد سپس انتخاب تعداد نورون هر لایه سپس تعداد bach های هر دوره سپس تابع فعال ساز

ن) بهترین مدل برای شبکه 4 لایه قسمت ح شماره یک است که دقت 41 درصد داده است و زمان اجرای آن و بقیه اطلاعات در بخش خود آمده است

به علت نمایش تمام خروجی ها در قسمت قبل دیگر اینجا تکرار نمی شود

سوال ۲ – MLP (Regression)

الف)ابتدا فایل csv را با استفاده از تابع پانداس read_csv() وارد کرده و سپس نمایش می دهیم تا دیتا ها را مشاهده کنیم تا تعداد ویژگی ها و همینطوری کمی و کیفی بودن اطلاعات را بررسی می کنیم دید کلی از دیتا را مشاهده کنید:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4600 entries, 0 to 4599
Data columns (total 18 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   date           4600 non-null   object
 1   price          4600 non-null   float64
 2   bedrooms       4600 non-null   float64
 3   bathrooms      4600 non-null   float64
 4   sqft_living    4600 non-null   int64
 5   sqft_lot       4600 non-null   int64
 6   floors         4600 non-null   float64
 7   waterfront     4600 non-null   int64
 8   view           4600 non-null   int64
 9   condition      4600 non-null   int64
 10  sqft_above     4600 non-null   int64
 11  sqft_basement  4600 non-null   int64
 12  yr_built       4600 non-null   int64
 13  yr_renovated   4600 non-null   int64
 14  street         4600 non-null   object
 15  city           4600 non-null   object
 16  statezip       4600 non-null   object
 17  country        4600 non-null   object
dtypes: float64(4), int64(9), object(5)
memory usage: 647.0+ KB
None
```

	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	sqft_above	sqft_basement	yr_built	yr_renovated	street	city	statezip	country
0	2014-05-02 00:00:00	3.130000e+05	3.0	1.50	1340	7912	1.5	0	0	3	1340	0	1955	2005	18810 Densmore Ave N	Shoreline	WA 98133	USA
1	2014-05-02 00:00:00	2.384000e+06	5.0	2.50	3650	9050	2.0	0	4	5	3370	280	1921	0	709 W Blaine St	Seattle	WA 98119	USA
2	2014-05-02 00:00:00	3.420000e+05	3.0	2.00	1930	11947	1.0	0	0	4	1930	0	1966	0	26206-26214 143rd Ave SE	Kent	WA 98042	USA
3	2014-05-02 00:00:00	4.200000e+05	3.0	2.25	2000	8030	1.0	0	0	4	1000	1000	1963	0	857 170th Pl NE	Bellevue	WA 98008	USA
4	2014-05-02 00:00:00	5.500000e+05	4.0	2.50	1940	10500	1.0	0	0	4	1140	800	1976	1992	9105 170th Ave NE	Redmond	WA 98052	USA
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4595	2014-07-09 00:00:00	3.081667e+05	3.0	1.75	1510	6360	1.0	0	0	4	1510	0	1954	1979	501 N 143rd St	Seattle	WA 98133	USA
4596	2014-07-09 00:00:00	5.343333e+05	3.0	2.50	1460	7573	2.0	0	0	3	1460	0	1983	2009	14855 SE 10th Pl	Bellevue	WA 98007	USA
4597	2014-07-09 00:00:00	4.169042e+05	3.0	2.50	3010	7014	2.0	0	0	3	3010	0	2009	0	759 Ilwaco Pl NE	Renton	WA 98059	USA
4598	2014-07-10 00:00:00	2.034000e+05	4.0	2.00	2090	6630	1.0	0	0	3	1070	1020	1974	0	5148 S Creston St	Seattle	WA 98178	USA

سپس تابعی را تعریف می کنیم تا پیش پردازش های لازم را روی این دیتا انجام دهد و همینطور دیتا های کیفی را تبدیل به کمی کرده و همچنین مقادیر دیتا ها را نرمالیزه کند یعنی اعداد را به اعداد بین صفر و یک اسکیل می کنید.

اسم این تابع را ready_to_process(data) را گذاشته ایم که یه عنوان ورودی فقط دیتای خام پانداس را می گیرد و پیش پرداز های توضیح شده را روی آن انجام می دهد و دیتای آماده شده برای پردازش را به عنوان خروجی باز می گرداند.

کد تابع ساخته شده به صورت زیر است:

```
def ready_to_process(data) :
  data_object = data.select_dtypes(include='object')
  data_number = data.drop(columns= data_object.keys())
  data_object = data_object.fillna('NA')
  scaler = MinMaxScaler()
  for n in data_object.keys():
    x=data_object[n]
    x= x.astype("category").cat.codes
    x = scaler.fit_transform(x.values.reshape((-1, 1)))
    data_object[n]=x
  for n in data_number.keys():
    data_number[n]= data_number[n].fillna(data_number.mean())
    data_number[n] = scaler.fit_transform(data_number[n].values.resha
pe((-1, 1)))
  return pd.concat([data_object , data_number],axis=1)
```

این تابع ایتدا داده های کمی و کیفی را جدا می کند داده های کیفی را را کمی کرده سپس نرمال می کند و داده های ناموجودش را na می گذارد همچنین برای داده های کمی داده های ناموجود را میانگین داده ها جایگزین می کند و سپسنرمالیزه می کند سر آخر هر دو داده را کنار هم گذاشته و به عنوان خروجی بر می گرداند.

دیتا به صورت زیر در میاید:

	date	street	city	statezip	country	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	sqft_above	sqft_basement	yr_built	yr_renovated
0	0.000000	0.336428	0.837209	0.815789	0.0	0.011771	0.333333	0.18750	0.073652	0.006775	0.2	0.0	0.0	0.50	0.107301	0.000000	0.482456	0.995531
1	0.000000	0.861848	0.813953	0.763158	0.0	0.089658	0.555556	0.31250	0.249051	0.007835	0.4	0.0	1.0	1.00	0.331858	0.058091	0.184211	0.000000
2	0.000000	0.506410	0.418605	0.342105	0.0	0.012862	0.333333	0.25000	0.118451	0.010534	0.0	0.0	0.0	0.75	0.172566	0.000000	0.578947	0.000000
3	0.000000	0.942308	0.069767	0.092105	0.0	0.015795	0.333333	0.28125	0.123766	0.006885	0.0	0.0	0.0	0.75	0.069690	0.207469	0.552632	0.000000
4	0.000000	0.961981	0.720930	0.407895	0.0	0.020684	0.444444	0.31250	0.119210	0.009186	0.0	0.0	0.0	0.75	0.085177	0.165975	0.666667	0.989076
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4595	0.985507	0.761936	0.813953	0.815789	0.0	0.011590	0.333333	0.21875	0.086560	0.005330	0.0	0.0	0.0	0.75	0.126106	0.000000	0.473684	0.982622
4596	0.985507	0.212202	0.069767	0.078947	0.0	0.020095	0.333333	0.31250	0.082764	0.006460	0.4	0.0	0.0	0.50	0.120575	0.000000	0.728070	0.997517
4597	0.985507	0.891468	0.744186	0.486842	0.0	0.015679	0.333333	0.31250	0.200456	0.005939	0.4	0.0	0.0	0.50	0.292035	0.000000	0.956140	0.000000
4598	1.000000	0.773210	0.813953	0.934211	0.0	0.007649	0.444444	0.25000	0.130600	0.005581	0.0	0.0	0.0	0.50	0.077434	0.211618	0.649123	0.000000
4599	1.000000	0.334660	0.209302	0.342105	0.0	0.008296	0.333333	0.31250	0.085042	0.006952	0.4	0.0	0.0	0.75	0.123894	0.000000	0.789474	0.000000

4600 rows × 18 columns

این دیتا آماده پردازش است.

ب)حال ستون قیمت را از دیتا جدا کرده و آن را به عنوان برچسب در نظر می گیریم و سپس 20 درصد دیتا را برای تست جدا می کنیم :

```
Y = ready_data['price'].to_numpy().reshape((-1, 1))
X = ready_data.drop(columns='price').to_numpy()
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.2)
```

حال چهار شبکه عصبی برای آموزش این دیتای لیبل شده آماده می کنیم در دوتا تعداد لایه ها را تغییر می دهیم و در دو تای دیگر تایع فعال ساز نورون ها را:

شبکه اول:دو لایه 1024 تایی و تابع relu

```
activation1 = 'relu'
layer1 = 2
model1 = Sequential([
    layers.Input(shape=(X.shape[1],)),
    layers.Dense(1024, activation=activation),
    layers.Dense(1024, activation=activation),
    layers.Dense(1, activation='linear')
])
```

شبکه دو : چهار لایه 1024 تایی و تابع relu

```
activation2 = 'relu'
layer2 = 4
model2 = Sequential([
    layers.Input(shape=(X.shape[1],)),
    layers.Dense(1024, activation=activation),
    layers.Dense(1024, activation=activation),
    layers.Dense(1024, activation=activation),
    layers.Dense(1024, activation=activation),
    layers.Dense(1, activation='linear')
])
```

شبکه سه :د دو لایه 1024 تایی و تابع tanh

```
activation3 = 'tanh'
layer3 = 2
model3 = Sequential([
    layers.Input(shape=(X.shape[1],)),
    layers.Dense(1024, activation=activation),
    layers.Dense(1024, activation=activation),
    layers.Dense(1, activation='linear')
])
```

شبکه چهار : چهار لایه 1024 تایی و تابع tanh

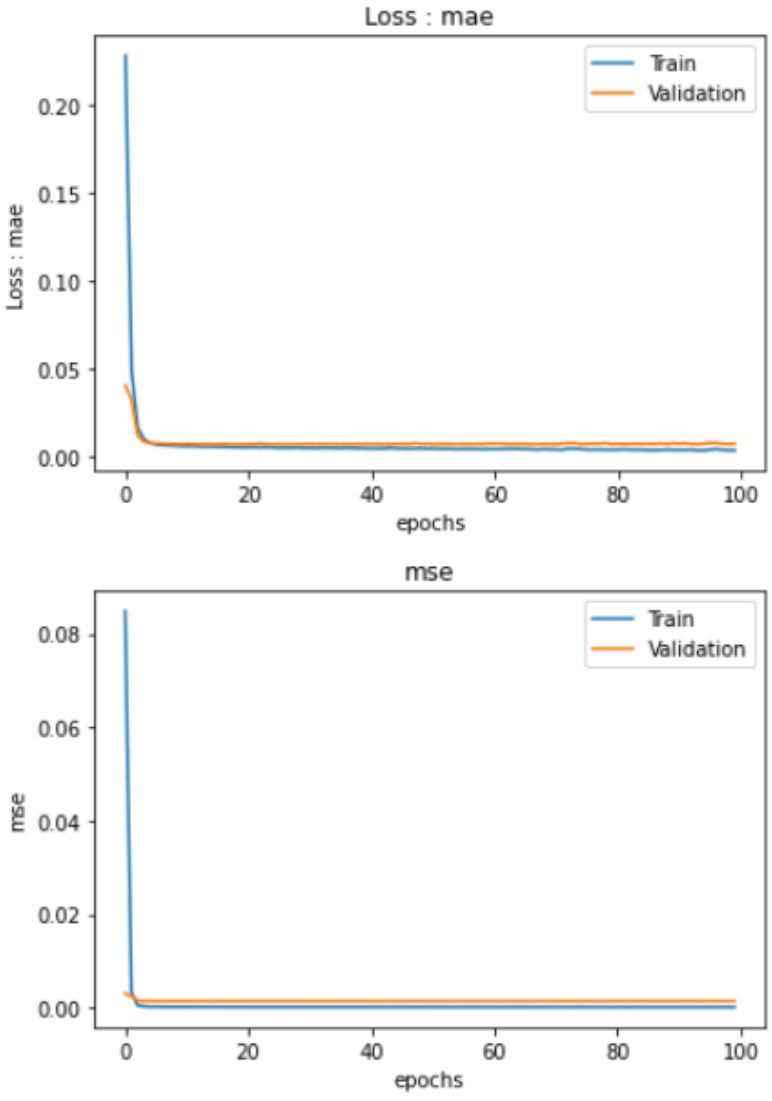
```
activation4 = 'tanh'
layer4 = 4
model4 = Sequential([
    layers.Input(shape=(X.shape[1],)),
    layers.Dense(1024, activation=activation),
    layers.Dense(1024, activation=activation),
    layers.Dense(1024, activation=activation),
    layers.Dense(1024, activation=activation),
```

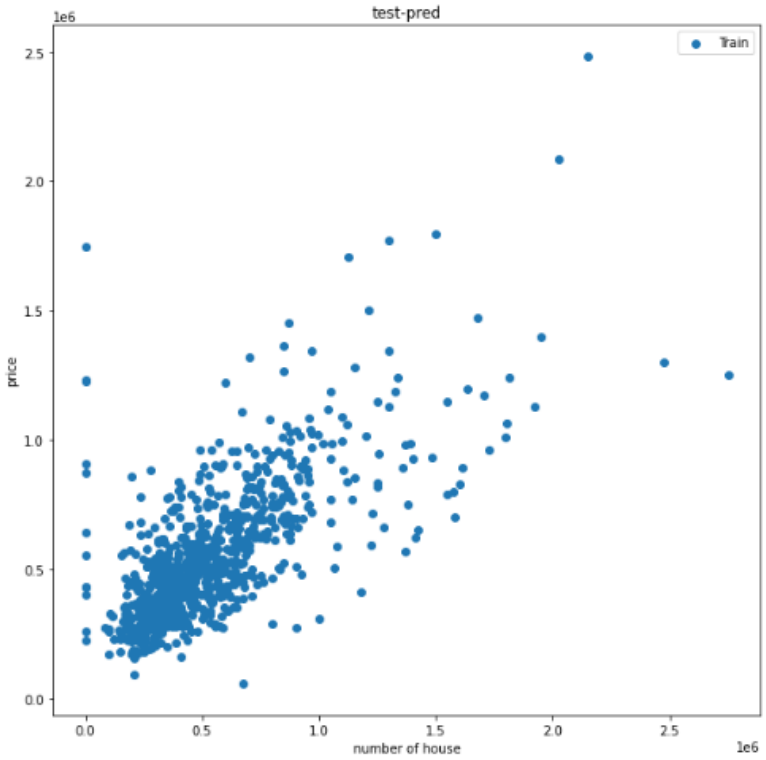
```
    layers.Dense(1, activation='linear')
])
```

در شبکه اول یک رگرسیون دو لایه که ساده ترین نوع رگرسیون است با تعداد کافی نورن قرار داده شده است در شبکه دوم قبل از رگرسیون یه پیش پرداز و حذف اعوجاج و نویز دو لایه هم قرار دادیم ویک شبکه 4 لایه را تشکیل داده ایم و در دو شبکه دیگر همین دو شبکه را با تابع فعال ساز نرم تر جایگیزن کرده ایم تا ضرایب نرمتر به ضرایب بهینه سر بخورند.

ج) این شبکه را با تابع loss ، mse تمرین می دهیم و نمودار های خروجی را برای هر ایپاک نمایش می دهیم و برای داده های تست مقدار واقعی و مقدار پیشبینی شده را در یک نمودار رسم می کنیم که هر چه روی خط نیم ساز قرار بگیره پیشبینی دقیق تر است و سپس 10 خانه ی اول تست را قیمتشان به همراه پیشبینی آن پیرینت شده است و سر آخر هم مقدار خطای mae , mse آنها در پایین آن آمده است:

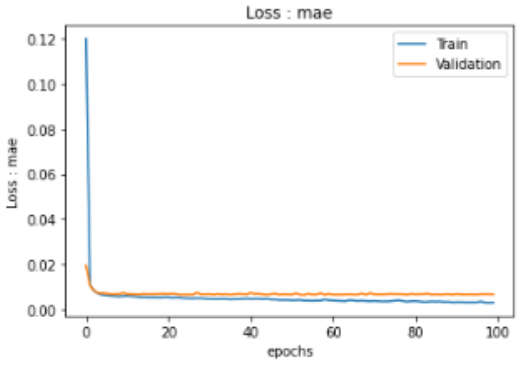
شبکه یک:

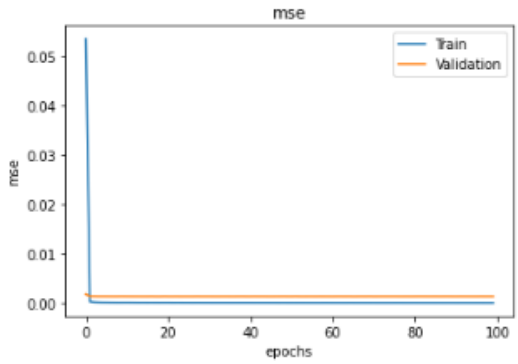


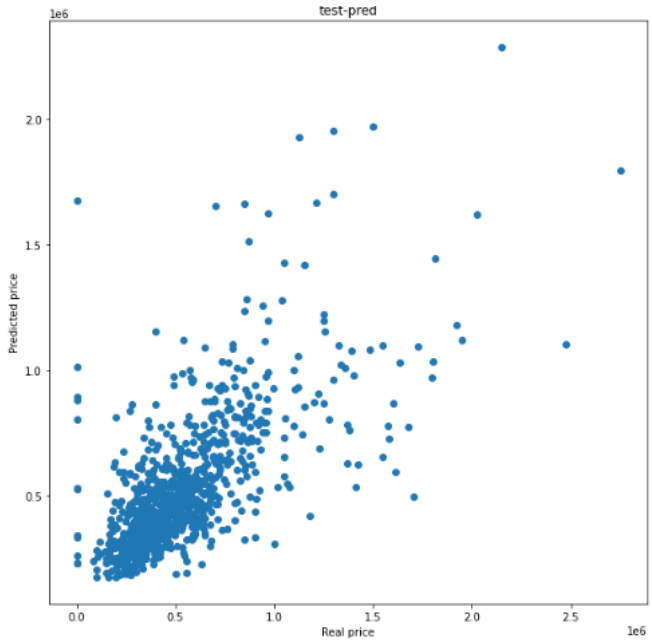



```
0 the price is 335000.000000 and te perd is 379227.312500 and delat(y_test-y_pred) -44227.312500
1 the price is 309780.000000 and te perd is 340870.312500 and delat(y_test-y_pred) -31090.312500
2 the price is 309487.500000 and te perd is 602972.062500 and delat(y_test-y_pred) -293484.562500
3 the price is 315275.000000 and te perd is 382357.343750 and delat(y_test-y_pred) -67082.343750
4 the price is 575000.000000 and te perd is 614761.375000 and delat(y_test-y_pred) -39761.375000
5 the price is 265000.000000 and te perd is 580298.000000 and delat(y_test-y_pred) -315298.000000
6 the price is 195000.000000 and te perd is 254216.578125 and delat(y_test-y_pred) -59216.578125
7 the price is 690000.000000 and te perd is 852611.125000 and delat(y_test-y_pred) -162611.125000
8 the price is 445000.000000 and te perd is 571959.000000 and delat(y_test-y_pred) -126959.000000
9 the price is 723000.000000 and te perd is 642832.312500 and delat(y_test-y_pred) 80167.687500
29/29 [==============================] - 0s 4ms/step - loss: 0.0056 - mse: 7.5929e-05
0.005647563841193914
7.592944166390225e-05
```

شبکه دو:

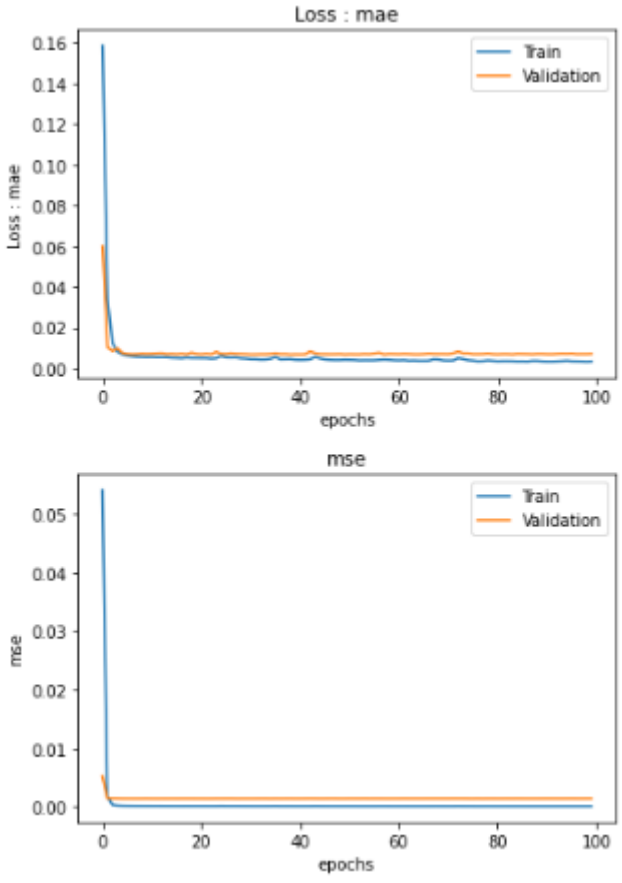


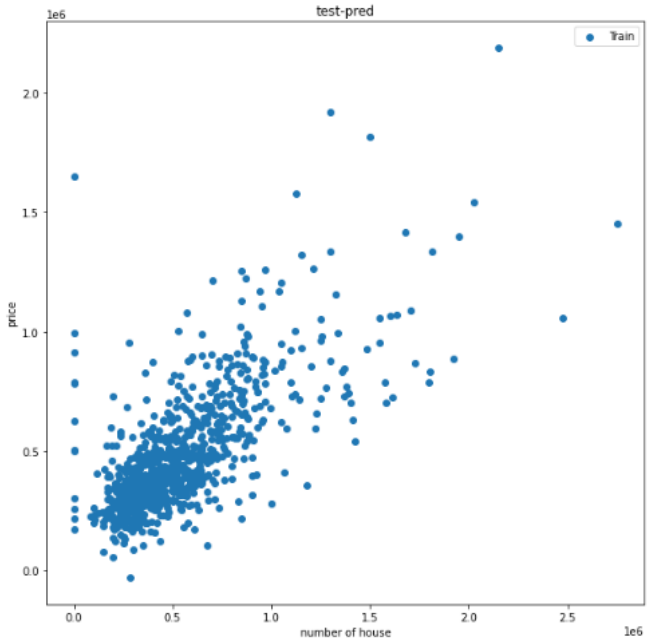





```
0 the price is 335000.000000 and te perd is 384248.593750 and delat(y_test-y_pred) -49248.593750
1 the price is 309780.000000 and te perd is 354513.031250 and delat(y_test-y_pred) -44733.031250
2 the price is 309487.500000 and te perd is 610336.937500 and delat(y_test-y_pred) -300849.437500
3 the price is 315275.000000 and te perd is 414220.062500 and delat(y_test-y_pred) -98945.062500
4 the price is 575000.000000 and te perd is 531259.500000 and delat(y_test-y_pred) 43740.500000
5 the price is 265000.000000 and te perd is 434312.031250 and delat(y_test-y_pred) -169312.031250
6 the price is 195000.000000 and te perd is 211465.453125 and delat(y_test-y_pred) -16465.453125
7 the price is 690000.000000 and te perd is 798668.687500 and delat(y_test-y_pred) -108668.687500
8 the price is 445000.000000 and te perd is 479137.343750 and delat(y_test-y_pred) -34137.343750
9 the price is 723000.000000 and te perd is 716426.375000 and delat(y_test-y_pred) 6573.625000
29/29 [==============================] - 0s 7ms/step - loss: 0.0057 - mse: 7.9994e-05
0.0057263742201030254
7.999449735507369e-05
```

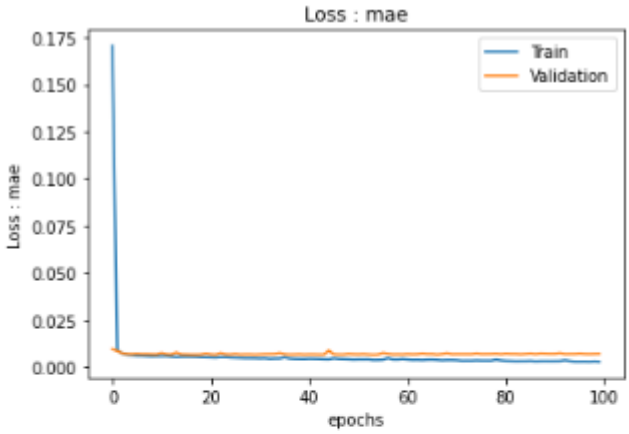
شبکه سه:

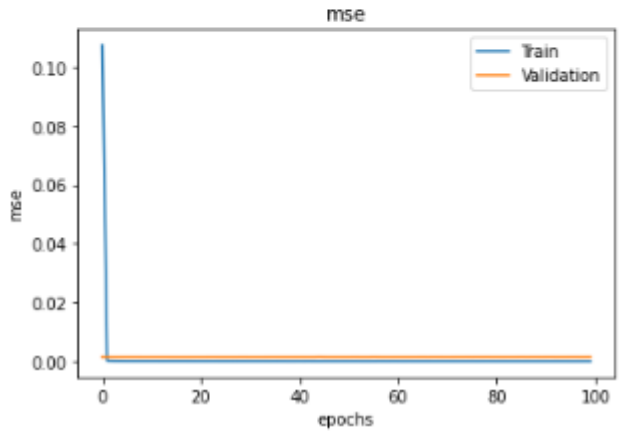


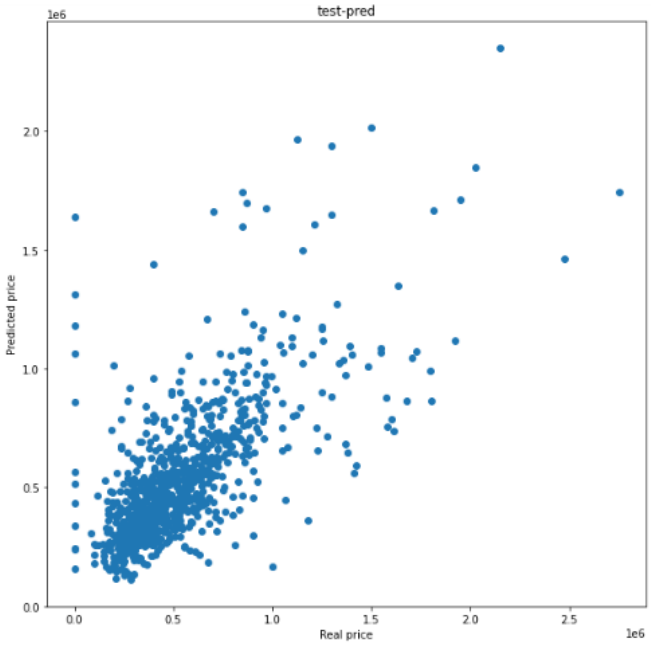



```
0 the price is 335000.000000 and te perd is 318494.000000 and delat(y_test-y_pred) 16506.000000
1 the price is 309780.000000 and te perd is 383886.000000 and delat(y_test-y_pred) -74106.000000
2 the price is 309487.500000 and te perd is 478311.281250 and delat(y_test-y_pred) -168823.781250
3 the price is 315275.000000 and te perd is 368754.156250 and delat(y_test-y_pred) -53479.156250
4 the price is 575000.000000 and te perd is 491460.250000 and delat(y_test-y_pred) 83539.750000
5 the price is 265000.000000 and te perd is 414243.843750 and delat(y_test-y_pred) -149243.843750
6 the price is 195000.000000 and te perd is 225406.765625 and delat(y_test-y_pred) -30406.765625
7 the price is 690000.000000 and te perd is 744542.437500 and delat(y_test-y_pred) -54542.437500
8 the price is 445000.000000 and te perd is 388310.500000 and delat(y_test-y_pred) 56689.500000
9 the price is 723000.000000 and te perd is 583572.625000 and delat(y_test-y_pred) 139427.375000
29/29 [==============================] - 0s 4ms/step - loss: 0.0059 - mse: 7.7972e-05
0.005874520633369684
7.79717811383307e-05
```

شبکه چهار:





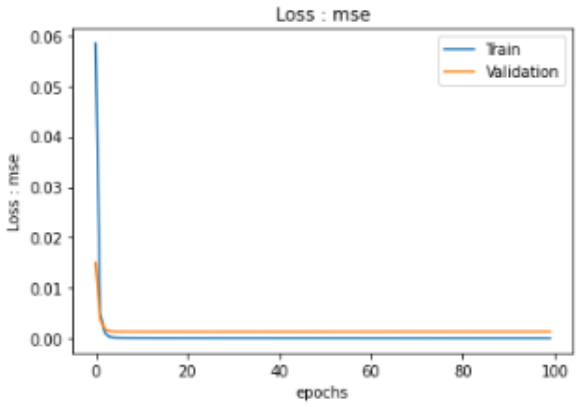


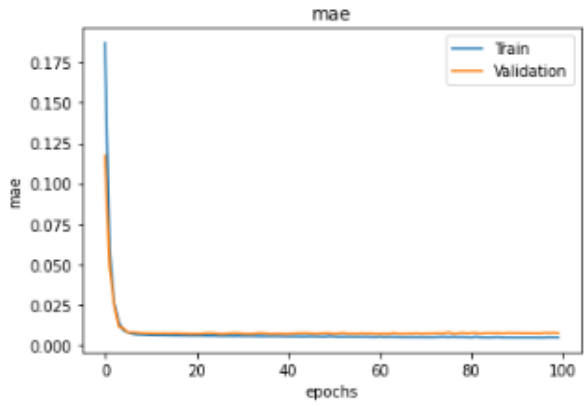

```
0 the price is 335000.000000 and te perd is 330089.312500 and delat(y_test-y_pred) 4910.687500
1 the price is 309780.000000 and te perd is 324559.343750 and delat(y_test-y_pred) -14779.343750
2 the price is 309487.500000 and te perd is 613424.125000 and delat(y_test-y_pred) -303936.625000
3 the price is 315275.000000 and te perd is 445355.875000 and delat(y_test-y_pred) -130080.875000
4 the price is 575000.000000 and te perd is 581754.375000 and delat(y_test-y_pred) -6754.375000
5 the price is 265000.000000 and te perd is 404339.656250 and delat(y_test-y_pred) -139339.656250
6 the price is 195000.000000 and te perd is 169996.328125 and delat(y_test-y_pred) 25003.671875
7 the price is 690000.000000 and te perd is 805470.812500 and delat(y_test-y_pred) -115470.812500
8 the price is 445000.000000 and te perd is 520319.156250 and delat(y_test-y_pred) -75319.156250
9 the price is 723000.000000 and te perd is 659319.687500 and delat(y_test-y_pred) 63680.312500
29/29 [==============================] - 1s 24ms/step - loss: 0.0057 - mse: 8.0344e-05
0.00570531003177166
8.03443617769517e-05
```

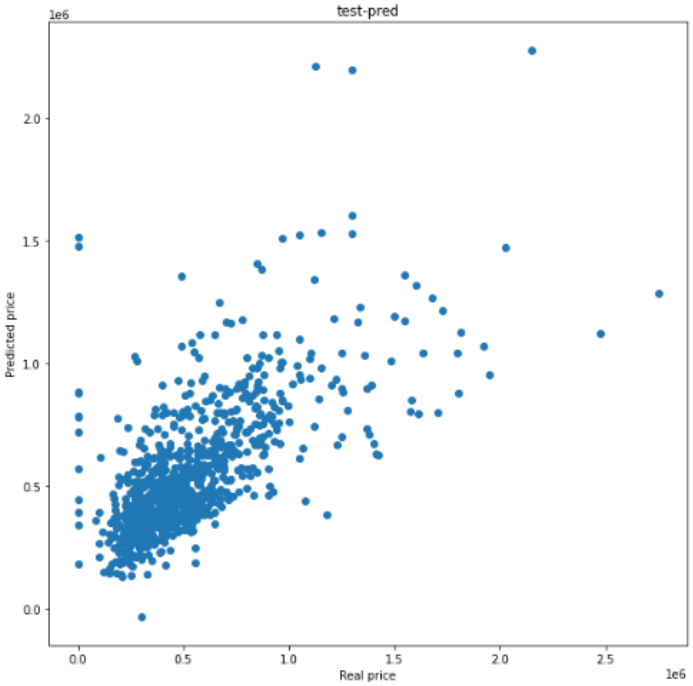
همان طور که ملاحضه می کنید شد شبکه تا 20 ایپاک حدودا ثابت می شود ولی می شود تعداد ایپاک ها رو بین 5 تا 10 هم در نظر گرفت اما با 20 ایپاک کاملا عملکرد شبکه مشخص می شود.

د)این قسمت تابع loss را mae می گیرم و دوباره شبکه ها را تمرین می دهیم:

شبکه اول:

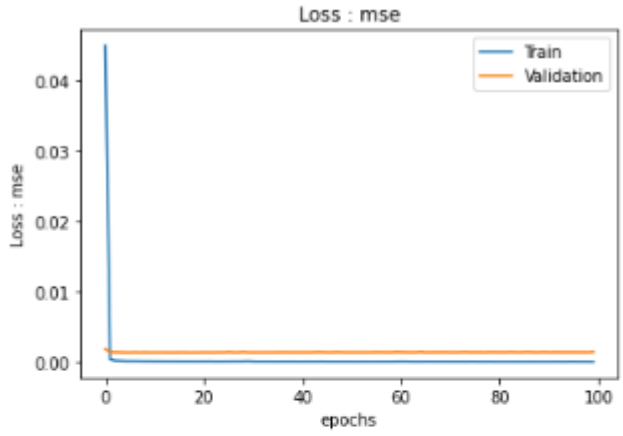


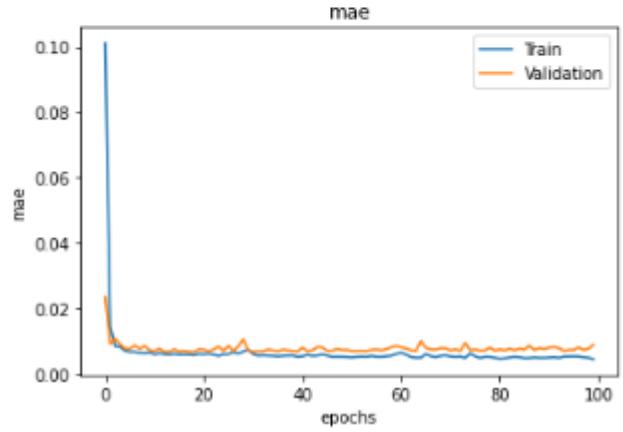


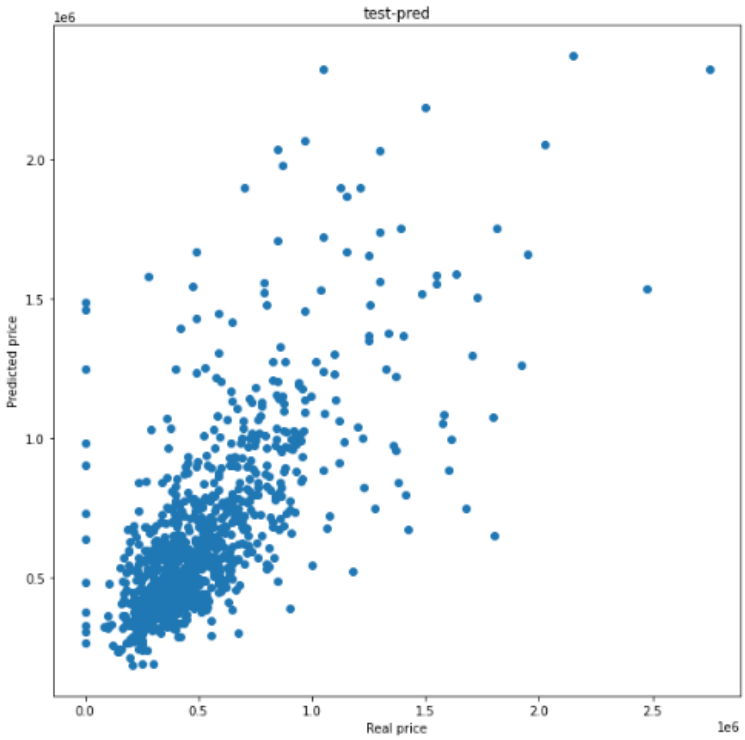



```
0 the price is 335000.000000 and te perd is 325068.906250 and delat(y_test-y_pred) 9931.093750
1 the price is 309780.000000 and te perd is 339134.000000 and delat(y_test-y_pred) -29354.000000
2 the price is 309487.500000 and te perd is 591471.875000 and delat(y_test-y_pred) -281984.375000
3 the price is 315275.000000 and te perd is 404418.531250 and delat(y_test-y_pred) -89143.531250
4 the price is 575000.000000 and te perd is 650428.812500 and delat(y_test-y_pred) -75428.812500
5 the price is 265000.000000 and te perd is 562616.687500 and delat(y_test-y_pred) -297616.687500
6 the price is 195000.000000 and te perd is 265174.375000 and delat(y_test-y_pred) -70174.375000
7 the price is 690000.000000 and te perd is 791763.625000 and delat(y_test-y_pred) -101763.625000
8 the price is 445000.000000 and te perd is 516327.468750 and delat(y_test-y_pred) -71327.468750
9 the price is 723000.000000 and te perd is 529817.187500 and delat(y_test-y_pred) 193182.812500
29/29 [==============================] - 0s 5ms/step - loss: 8.2599e-05 - mae: 0.0059
8.259920286945999e-05
0.005920475348830223
```

شبکه دوم:

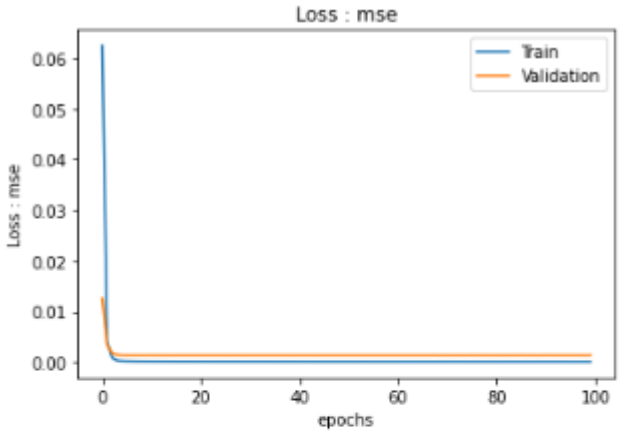


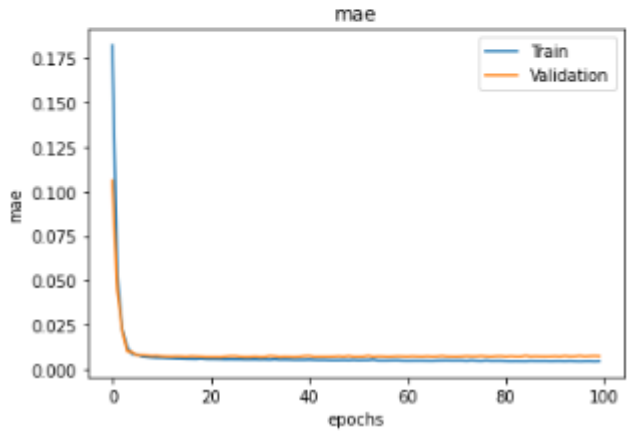


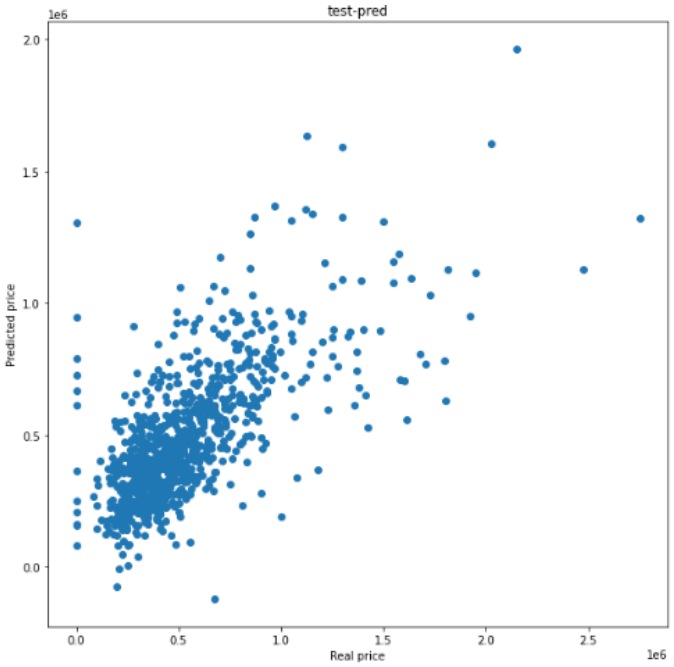



```
0 the price is 335000.000000 and te perd is 301056.562500 and delat(y_test-y_pred) 33943.437500
1 the price is 309780.000000 and te perd is 436019.937500 and delat(y_test-y_pred) -126239.937500
2 the price is 309487.500000 and te perd is 688023.437500 and delat(y_test-y_pred) -378535.937500
3 the price is 315275.000000 and te perd is 534006.937500 and delat(y_test-y_pred) -218731.937500
4 the price is 575000.000000 and te perd is 752909.187500 and delat(y_test-y_pred) -177909.187500
5 the price is 265000.000000 and te perd is 682480.562500 and delat(y_test-y_pred) -417480.562500
6 the price is 195000.000000 and te perd is 277874.687500 and delat(y_test-y_pred) -82874.687500
7 the price is 690000.000000 and te perd is 929294.937500 and delat(y_test-y_pred) -239294.937500
8 the price is 445000.000000 and te perd is 606521.125000 and delat(y_test-y_pred) -161521.125000
9 the price is 723000.000000 and te perd is 852775.875000 and delat(y_test-y_pred) -129775.875000
29/29 [==============================] - 0s 11ms/step - loss: 1.1496e-04 - mae: 0.0074
0.000114964677777607
0.007394431624561548
```

شبکه سوم:

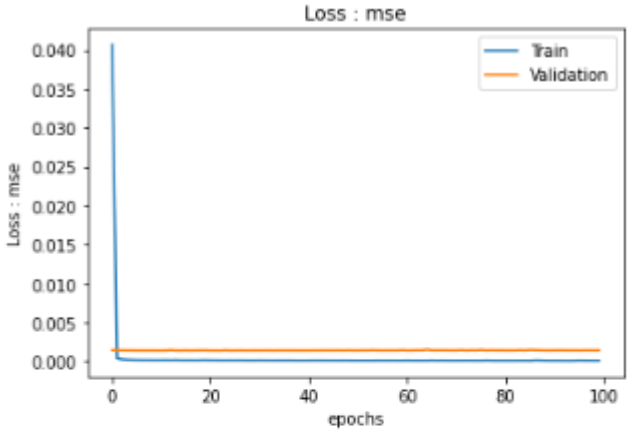


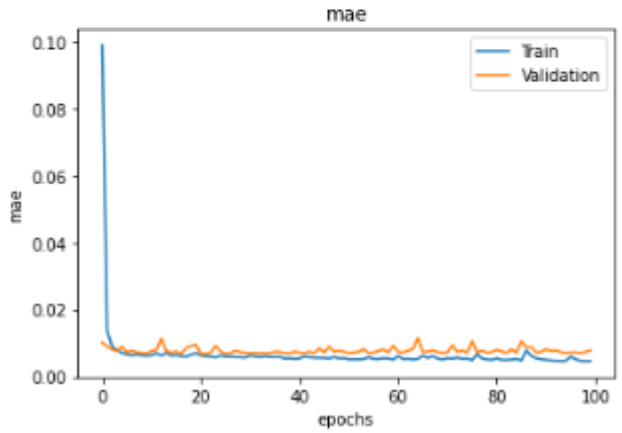


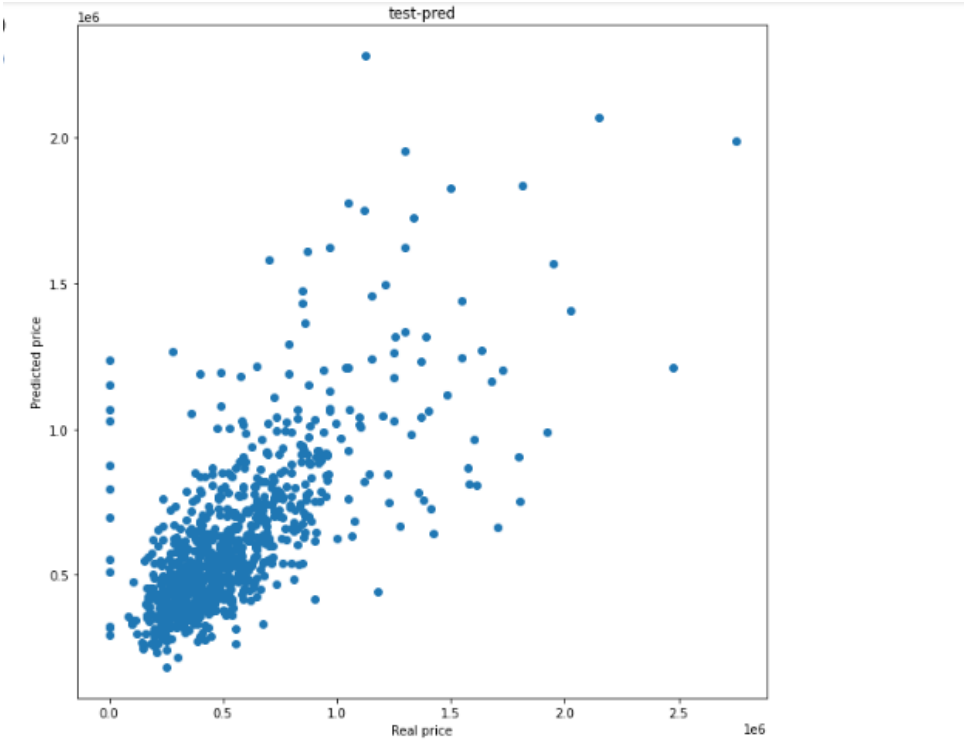



```
0 the price is 335000.000000 and te perd is 119652.406250 and delat(y_test-y_pred) 215347.593750
1 the price is 309780.000000 and te perd is 333536.593750 and delat(y_test-y_pred) -23756.593750
2 the price is 309487.500000 and te perd is 525586.187500 and delat(y_test-y_pred) -216098.687500
3 the price is 315275.000000 and te perd is 431903.281250 and delat(y_test-y_pred) -116628.281250
4 the price is 575000.000000 and te perd is 600884.125000 and delat(y_test-y_pred) -25884.125000
5 the price is 265000.000000 and te perd is 475726.656250 and delat(y_test-y_pred) -210726.656250
6 the price is 195000.000000 and te perd is 346073.093750 and delat(y_test-y_pred) -151073.093750
7 the price is 690000.000000 and te perd is 765077.875000 and delat(y_test-y_pred) -75077.875000
8 the price is 445000.000000 and te perd is 427312.062500 and delat(y_test-y_pred) 17687.937500
9 the price is 723000.000000 and te perd is 625520.500000 and delat(y_test-y_pred) 97479.500000
29/29 [==============================] - 0s 7ms/step - loss: 7.9722e-05 - mae: 0.0060
7.972186722327024e-05
0.00599106913805008
```

شبکه چهار:








```
0 the price is 335000.000000 and te perd is 368032.187500 and delat(y_test-y_pred) -33032.187500
1 the price is 309780.000000 and te perd is 456156.375000 and delat(y_test-y_pred) -146376.375000
2 the price is 309487.500000 and te perd is 659216.000000 and delat(y_test-y_pred) -349728.500000
3 the price is 315275.000000 and te perd is 480674.937500 and delat(y_test-y_pred) -165399.937500
4 the price is 575000.000000 and te perd is 691939.187500 and delat(y_test-y_pred) -116939.187500
5 the price is 265000.000000 and te perd is 550151.125000 and delat(y_test-y_pred) -285151.125000
6 the price is 195000.000000 and te perd is 312429.812500 and delat(y_test-y_pred) -117429.812500
7 the price is 690000.000000 and te perd is 922607.812500 and delat(y_test-y_pred) -232607.812500
8 the price is 445000.000000 and te perd is 576318.062500 and delat(y_test-y_pred) -131318.062500
9 the price is 723000.000000 and te perd is 766578.812500 and delat(y_test-y_pred) -43578.812500
29/29 [==============================] - 0s 11ms/step - loss: 8.3885e-05 - mae: 0.0063
8.388507558265701e-05
0.006292890757322311
```

ه) تابع MAE به صورت زیر است :

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

مجموع قدر مطلق اختلاف هر دیتا با میانگین خودش تقسیم بر تعداد کل داده ها میانگین تمام اختلاف های داده ها از میانگین داده ها

تابع MSE به صورت زیر است:

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

مجموع مجذور اختلاف هر دیتا با میانگین خودش تقسیم بر تعداد کل داده ها یا همان میانگین تمام مجذور اختلاف های داده ها از میانگین داده ها

MSE دیتا هایی که از میانگین دور ترهستن رو ضریب بیشتری می دهد و اهمیتش روی اختلاف های زیاد بیشتر است در واقع مینیم سازی این تابع داده هار رو بیشتر کنار هم نگه میداره نسبت به MAE

مقایسه ج و د نشان می دهند که شبکه های محاسبه شده با loss ، mae دارای مقدار خطای کمتری هستند اما نسب به شبکه های محاسبه شده با loss ، mse دارای پراکندگی بیشتری هستند در واقع اگر بخواهیم همه داده ها تا مقداری که می شود درست تشخیص داده بشوند mse را استفاده می کنیم ولی اگر بخواهیم اکثر داده ها با دقت بیشتری تشخیص داده شوند از mse استفاده می کنیم.

سوال ۳ – کاهش ابعاد

الف) ما در روش pca میخواهیم با استفاده از جبر خطی و مقادیر ویژه ابعاد اضافی و با اطلاعات کم دیتای خودمان را از بین ببیرم این روش به این صورت است که :

ابتدا ماتریس کواریانس داده خود را به دست می آوریم ($x^T=[1,x_1,x_2,..,x_n]$)

$$R = \sum_{i=1}^{m} x^i x^{iT} = V\Lambda V^T$$

این ماتریس مقدار وابستگی تمام ابعاد ورودی را به هم به ما می دهد حال ما با به دست آوردن مقادیر و بردار های ویژه این ماتریس می توانیم ابعاد با اطلاعات بیشتر را از ابعاد بدون اطلاعات جدا کنیم معیار وجود اطلاعات هم داشتن واریانس بیشتر است . پس بردار ویژه و مقدار ویژه ماتریس ها را به دست می آوریم و سپس مقادیر ویژه سفر را جدا کرده در ابتدا بردار قرار می دهیم و بردار های آنها را نیز در ماتریس جا به جا می کنیم تا بتوانیم ماتریس را به دو قسمت بردار ویژه صفر و گیر صفر تقسیم کنیم حال بردار های غیر صفر را جدا کرده و ماتریس جدید به وجود آمده در واقع تبدیلی برای کاهش بعد دیتا ی ورودی است و با ضرب کردن ترانهاده ی آن تابع در ابتدای دیتای تبدیل شده که به جای ورودی های حذف شده صفر گذاشتیم دیتا اولیه درست می شود که یعنی اطلاعاتی را از دست ندادیم.

$$V = \begin{bmatrix} 0 & V_3 \\ V_1 & 0 \end{bmatrix} \quad V_1 = \text{span}([y^1 \cdots y^k]) = \mathbf{0} \quad V_3 = \text{span}([y^{k+1} \cdots y^N])$$

$$V = \left[\underbrace{v_1 .. v_r}_{V_1} \underbrace{v_{r+1} ... v_n}_{V_2}\right] = [V_1\ V_2]$$

$$V^T x = \begin{bmatrix} V_1^T \\ V_2^T \end{bmatrix} x = \begin{bmatrix} V_1^T x \\ V_2^T x \end{bmatrix} = \begin{bmatrix} 0_{r\times 1} \\ z_{(n-r)\times 1} \end{bmatrix}$$

Nullity part which will be removed.

Informative part which forms the new space

$$x \epsilon R^n \xrightarrow{Dimensionality\ Reduction} z(x) = Tr(x) = V_2^T x \quad z \epsilon R^{n-r}$$

$$z \epsilon R^{n-r} \xrightarrow{Data\ retrieving} x = Tr^{-1}(z) = V \begin{bmatrix} 0_{r\times 1} \\ z \end{bmatrix} = VV^T x = x \epsilon R^{n-r}$$

حال چون گفته شده از کتابخانه pca استفاده نشود ما خودمان باید آن را پیاده کنیم تا بتوانیم ابتدا تشخیص دهیم چند بعد باید کاهش داد که یعنی بعد های با اندازه مقدار ویژه کم و نزدیک به صفر را حذف کنیم و بعد با کنار هم چیندن بردار های ویژه ای که حذف نکردیم در ماتریس ترانهاده شده و ضربش در داده ها ابعاد داده ها را کم کنبم.

ابتدا تابع منحنی تجمعی مقدار واریانس مقدار ویژه ها را تایپ میکنیم:

```
def plot_comp_var(X):
  cov_mat = np.cov(X.T)
  eig_vals = np.linalg.eig(cov_mat)[0]
  eig_vals_sort = [(np.abs(eig_vals[i])) for i in range(len(eig_val)]
  cum_exp_val = np.cumsum(eig_vals_sort/sum(eig_vals))
  return cum_exp_val
```

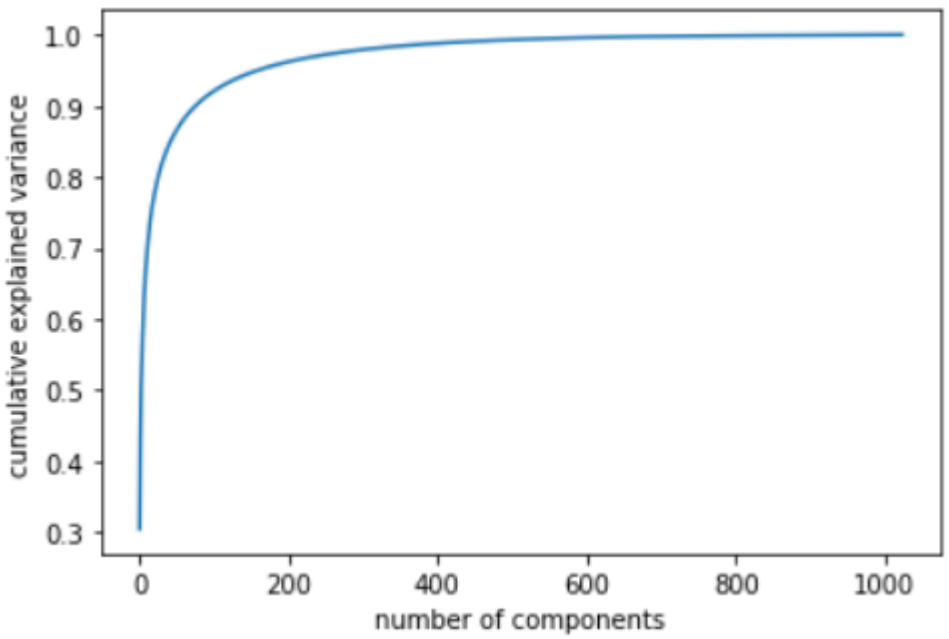
این تابع ابتدا ماتریس کواریانس ورودی را محاسبه می کند

سپس مقادیر ویژه آن را محاسبه کرده و در متغییری ذخیره میکند

بعد مقادیر ویژه را سورت کرده و نرمالیزه می کند تا جمع تمامش یک شود

سرآخر به صورت تجمعی در آورده و خروجی می دهد برای رسم کردن.

ما داده های سوال یک را همانند کاری که انجام دادیم پیش پردازش می کنیم و وارد این تابع کرده ایم تا خروجی را رسم کنیم .حاصل:



همان طور که ملاحضه می شود بیش از 90 درصد اطلاعات در 150 بعد ابتدایی موجود است پس می توان با دقت خوبی داده ها را از 32*32 به 150 کاهش داد.

حال تابع کاهش بعد را می نویسیم:

```
def pca_n_dec(X,X1,n):
  cov_mat = np.cov(X.T)
  eig_vals, eig_vecs = np.linalg.eig(cov_mat)
  eig_pairs = [(np.abs(eig_vals[i]), eig_vecs[:,i]) for i in range(len(eig_vals))]
  v = eig_pairs[:n]
  y=[]
  for i in v:
    y.append(i[1])
  v_tr = np.array(y)
  x=[]
  for i in range(len(X1)):
    x += [np.dot(v_tr,X1[i])]
  x_out_n=np.array(x)
  return x_out_n
```

مقادیر ویژه و بردار های ویژه را سورت می کنیم

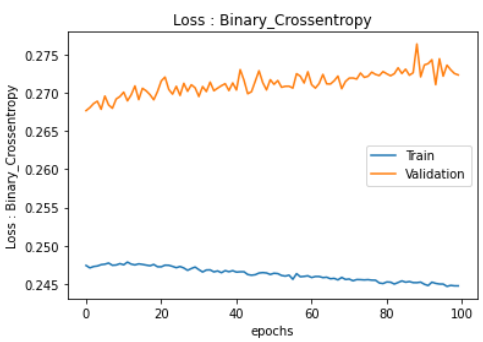
150 تای اول را نگه می داریم و تبدیل به آرایه میکنیم

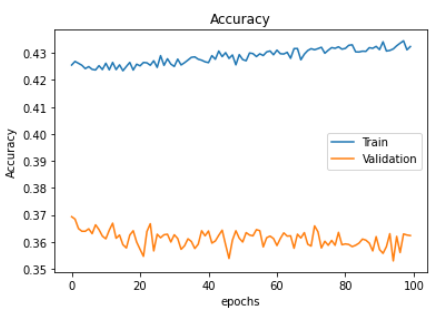
سپس تمام داده ها رو درونش ضرب داخلی میکنیم تا داده های جدید به دست بیایند

سر آخر هم دادهها را به عنوان خروجی پاس میدهیم.

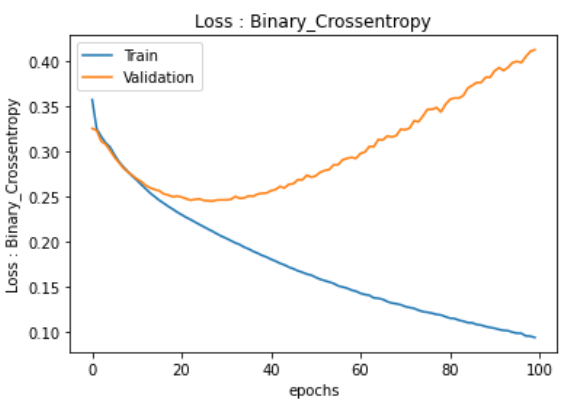
ب)حال داده های train و test را به تابع می دهیم تا کاهش بعد انجام شود و سپس به شبکه ی ایجاد شده در سوال یک میدهیم تا بهبود را مشاهده کنیم

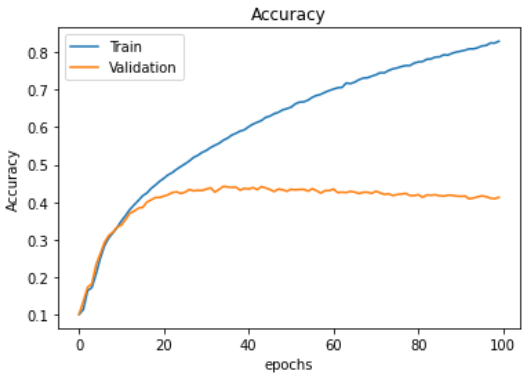
که شرایط از دقت 35 درصد به دقت 40 درصد افزایش یافته است که نشان دهنده بهبود عملکرد شبکه است.





این دو نمودار عملکرد قدیم شبکه است.





و این دو نمودار برای بعد از انجام pca است که به وضوع بهبود یافته است.

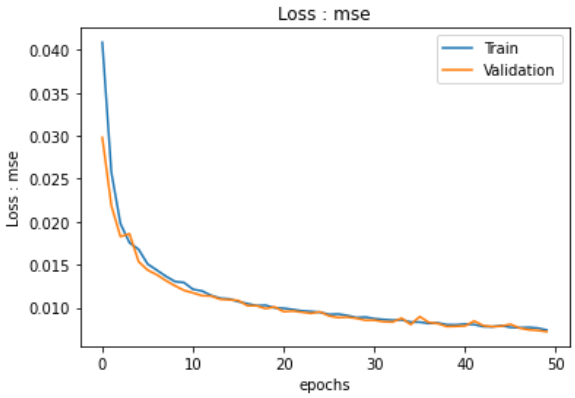
ج) حال می خواهیم با روش شبکه auto-encoder ابعاد را کاهش دهیم و سپس دوباره با همان شبکه قبل مدل را آموزش دهیم تا نتیجه را مشاهده کنیم.

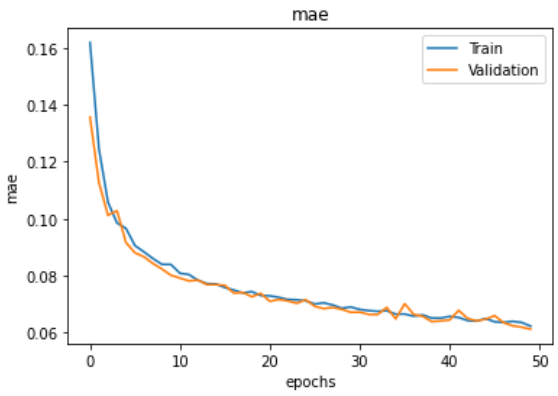
برای ساخت auto encoder از کتابخانه keras استفاده می شود و بر خلاف کد های قبلی این بار باید برای هر لایه اسم گذاری کرد تا بتوان لایه وسطی را که لایه encode شده است را از خروجیش استفاده کرد و ما شبکه auto encoder خود را اینگونه تعریف می کنیم:

```
import keras
input_img = keras.Input(shape=(32*32,))
encoded1 = layers.Dense(512, activation='relu')(input_img)
encoded2 = layers.Dense(256, activation='relu')(encoded1)
encoded3 = layers.Dense(150, activation='relu')(encoded2)

decoded1 = layers.Dense(256, activation='relu')(encoded3)
decoded2 = layers.Dense(512, activation='relu')(decoded1)
decoded3 = layers.Dense(32*32, activation='sigmoid')(decoded2)
autoencoder = keras.Model(input_img, decoded3)
encoder = keras.Model(input_img, encoded3)

encoded_input = keras.Input(shape=(150,))
decoder_layer = autoencoder.layers[-1]
#decoder = keras.Model(encoded_input, decoded3)

autoencoder.compile(
    optimizer='adam',
    loss='mse',
    metrics=['mae']
    #metrics=['mae','accuracy']
)
```

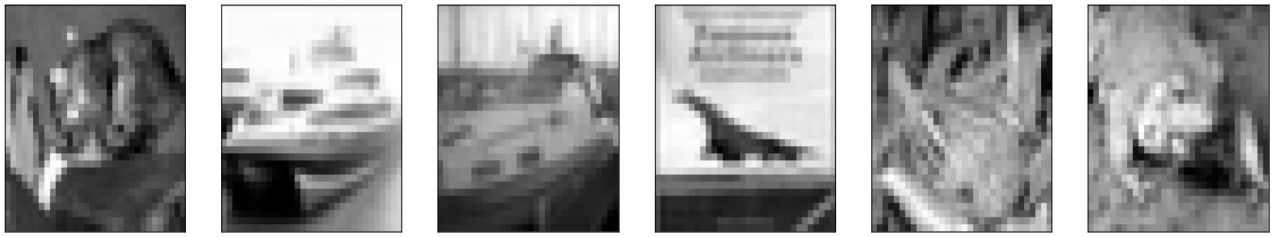
ما دو لایه دیکودر قرار دادیم تا ورودی را از 32*32 به 150 برسانیم همان عددی که در قسمت قبل محاسبه کردیم این دولایه را 512 و256 نورونه قرار دادیم.

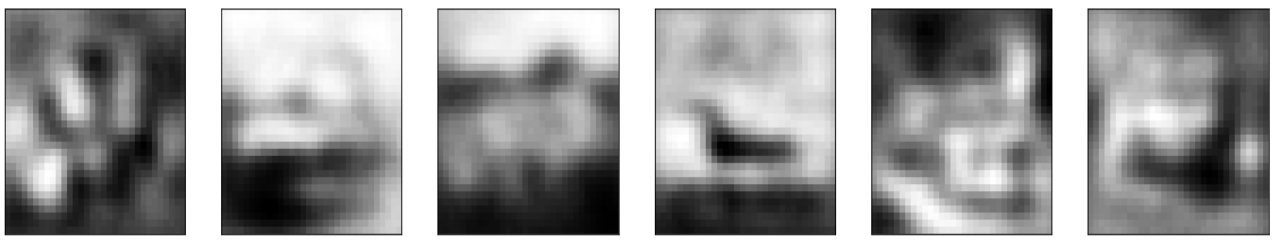
ما شبکه را به علت کمبود وقت به جای 100 ایپاک در 50 ایپاک ترین دادیم که همان هم نتیجه قایل قیولی را ارائه داد نمودار کاهش خطای آن را مشاهده بفرمایید:



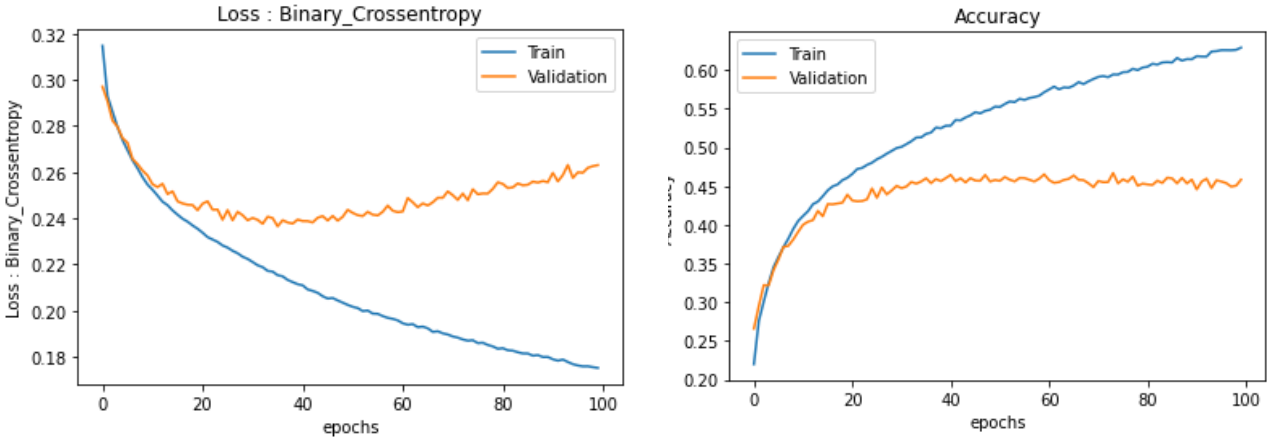


حال خروجی شبکه را هم به نمایش گذاشته می شود و به علت کیفت پایین عکس ورودی و ورود عکس هی جدید test به شبکه عکس خارج شده از شبکه ما هم کیفت خوبی ندارد و زیاد قایل تشخیص نیست اما یه علت این که خطا کلی کمتر از 0.01 است قایل قبول است





حال داده های کاهش بعد یافته با استفاده از این شبکه را که از لایه ی میانی شبکه می گیرم را وارد شبکه قدیمی خودمان می کنیم تا بعد از تمرین دادن شبکه متوجه نتیجه بشویم که خروجی های آن به صورت زیر است:

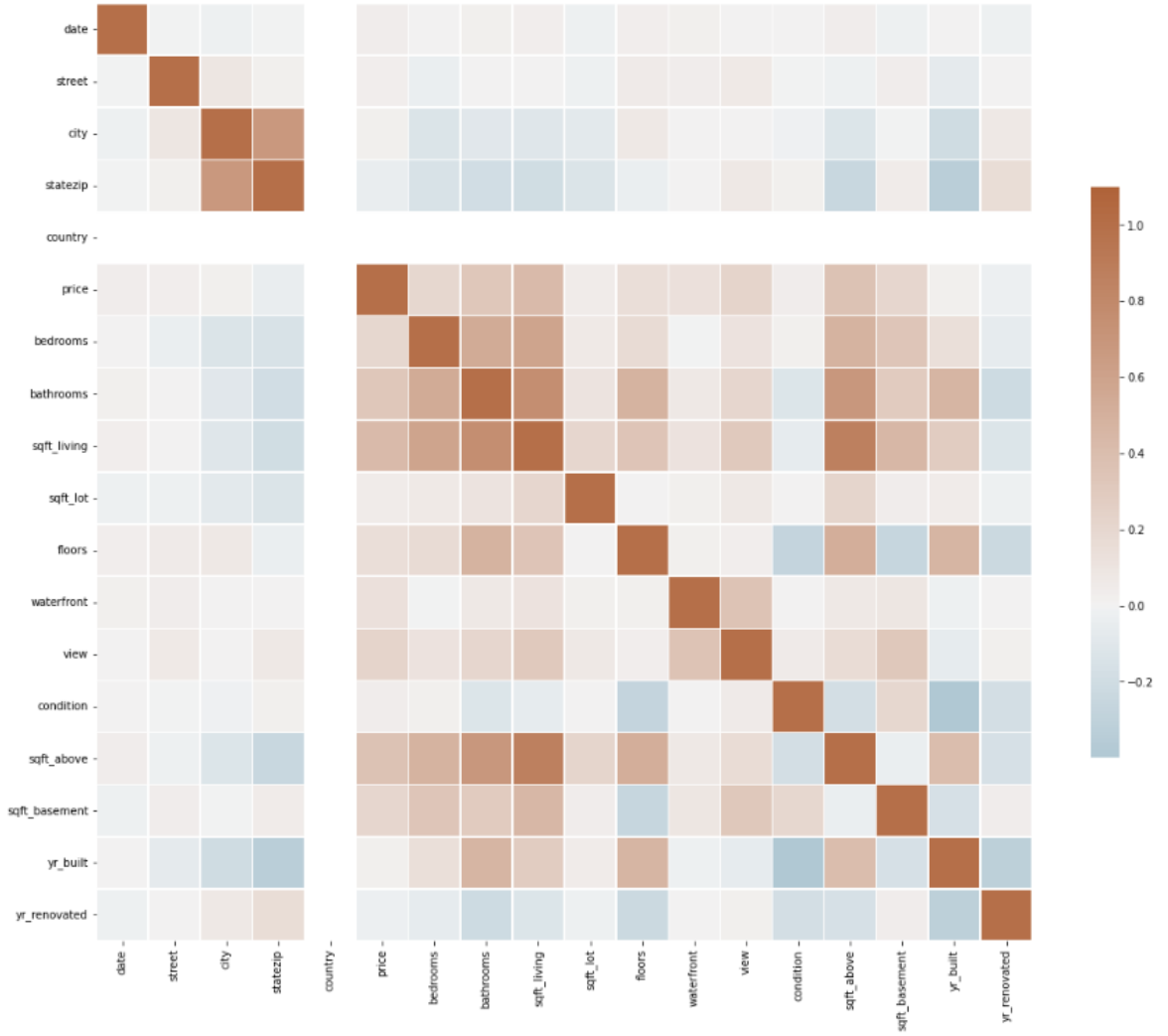


و همینطور برای داده های تست 45 درصد دقت و 0.26 خطا داریم که از دو شبکه قبل بهتر از و این شبکه از pca هم نتیجه بهتری گرفت.

	دقت (درصد)	خطا	زمان(دقیقه)
شبکه بدون کاهش بعد	35.7	0.27	2:32
pca	40.2	0.41	4:22
Auto encoder	44.5	0.26	4:02

همانطور که مشاهده می کنید به وضوح شبکه با کاهش بعد auto encoder برای آن دو لایه شبکه که ما در سوال یک آموزش دادیم بهتر کار کرده است و دقت بالاتری برای داده های جدید دارد همینطور خطای کمتری هم در آن هست.

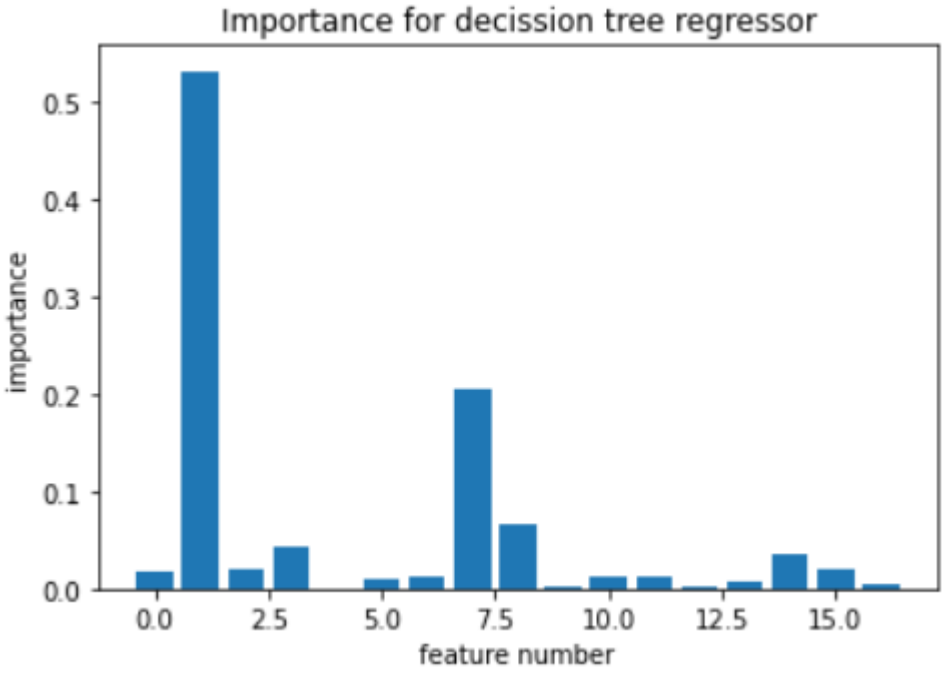
د) ابتدا ماتریس همبستگی را مطابق صورت سوال رسم میکنیم. فقط دقت داریم برای محاسبه ماتریس همبستگی باید همه ویژگیها به صورت عددی باشند، پس از پیشپردازشی که در سوال 1 استفاده کردیم برای تبدیل ویژگیهای categorical به numerical استفاده میکنیم.

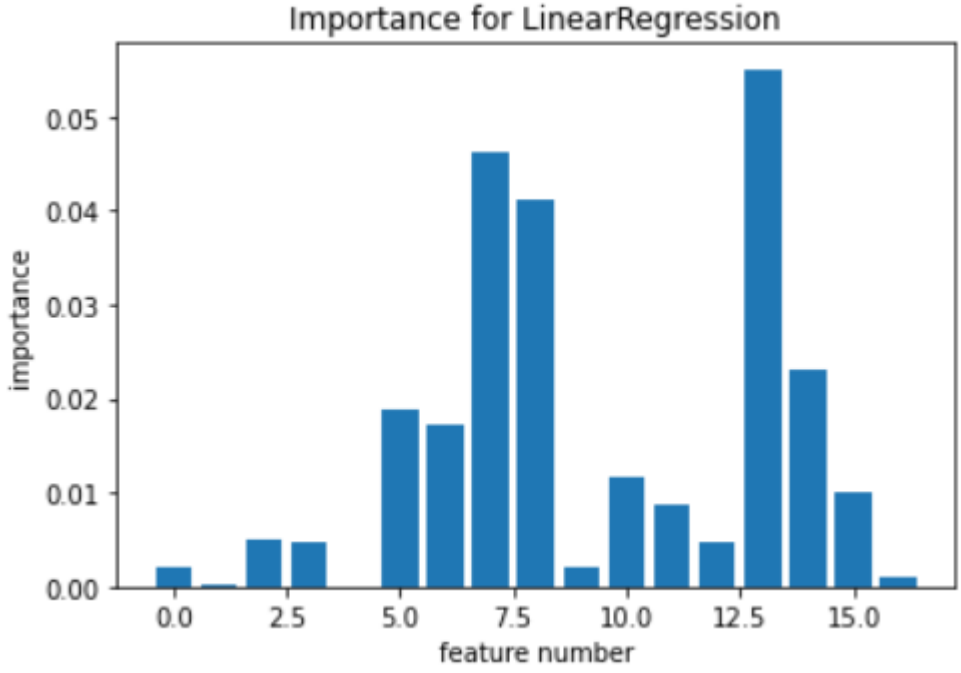


همانگونه که مشاهده میشود، در این ماتریس میزان همبستگی خطی ویژگیها به یکدیگر مشخص میشود، یعنی هرچی درایه یک سطر و ستون از لحاظ اندازه بزرگتر باشند میزان همبستگی خطی میان ویژگی سطر و ستون متناظر بیشتر خواهد بود و با داشتن یکی میتوانیم دیگری را توصیف کنیم. با این تعریف انتظار داریم که درایههای قطر اصلی این ماتریس برابر 1 باشند که همینطور هم هست. برای انتخاب ویژگیهای مناسب جهت تخمین قیمت باید ویژگیهایی را انتخاب کنیم که بیشترین همبستگی را با قیمت خانه داشته باشند

و) با استفاده از مدلهای Linear Regression و Decision Tree ، اهمیت هر ویژگی را در سوال 2 بدست آورید و در یک بارپلات نمایش دهید. برای این کار میتوانید از کتابخانه scikit-learn و متدهای DecisionTreeRegressorو LinearRegression استفاده کنید .

ما داده ها را لیبل دار کرده و با استفاده از کتابخانه های معرفای شده بار پلات های هر یک را رسم کرده ایم


Importance for decission tree regressor
importance
0.0
0.1
0.2
0.3
0.4
0.5
0.0
2.5
5.0
7.5
10.0
12.5
15.0
feature number


Importance for LinearRegression
importance
0.00
0.01
0.02
0.03
0.04
0.05
0.0
2.5
5.0
7.5
10.0
12.5
15.0
feature number