به نام خدا

دانشگاه تهران

دانشکده مهندسی برق و کامپیوتر

شبکه های عصبی و یادگیری عمیق

تمرین سری دوم

نام و نام خانوادگی	علی عدالت
شماره دانشجویی	۸۱۰۱۹۹۳۴۸
تاریخ ارسال گزارش	

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

سوال 1 – MLP (Regression)

الف

در این قسمت به پیش پردازش داده‌ها می‌پردازیم. تعدادی از ستون‌ها دارای مقدار null هستند. نیاز است این مقادیر null را پر کنیم. در Figure 1 لیست ویژگی‌ها با مقدار null و تعداد null به ازای هر ویژگی آمده است.

LotFrontage	259
Alley	1369
MasVnrType	8
MasVnrArea	8
BsmtQual	37
BsmtCond	37
BsmtExposure	38
BsmtFinType1	37
BsmtFinType2	38
Electrical	1
FireplaceQu	690
GarageType	81
GarageYrBlt	81
GarageFinish	81
GarageQual	81
GarageCond	81
PoolQC	1453
Fence	1179
MiscFeature	1406

Figure 1 لیست ویژگی‌ها با مقدار **null**

تعداد کل داده‌ها ۱۴۶۰ تا است. با توجه به تعداد کل داده‌ها، ویژگی‌هایی که در بالا تر از ۷۰ درصد داده‌ها مقدار null دارند را حذف می‌کنیم. این ویژگی ها باید از ویژگی‌هایی باشند که در آنالیز مورد نظر تاثیر کمی داشته باشند. با توجه به این شرایط ویژگی‌های ['Alley', 'PoolQC', 'Fence', 'MiscFeature'] را حذف می‌کنیم. بعد از حذف این ویژگی‌ها باید به پر کردن مقادیر null دیگر ویژگی‌ها بپردازیم. برای تعیین نحوه پر کردن این مقادیر باید عددی یا غیر عددی بودن این ویژگی‌ها را تعیین کنیم. از بین ویزگی‌ها با مقدار null ویژگی‌های LotFrontage، MasVnrArea و GarageYrBlt عددی هستند. برای پر کردن ویزگی عددی بدون outlier از میانگین و در غیر این صورت از میانه استفاده می‌کنیم. ویژگی‌های بدون outlier را با میانگین که آماره تمرکز و مرکز مناسبی است پر می‌کنیم. در مواردی که outlier داریم، میانه

آماره‌ی مناسبی برای مرکز است و به همین دلیل از آن استفاده می‌کنیم. برای بررسی وجود outlier به ازای هر ویژگی عددی از رسم boxplot استفاده می‌کنیم. نمودار boxplot در Figure 2 آمده است.

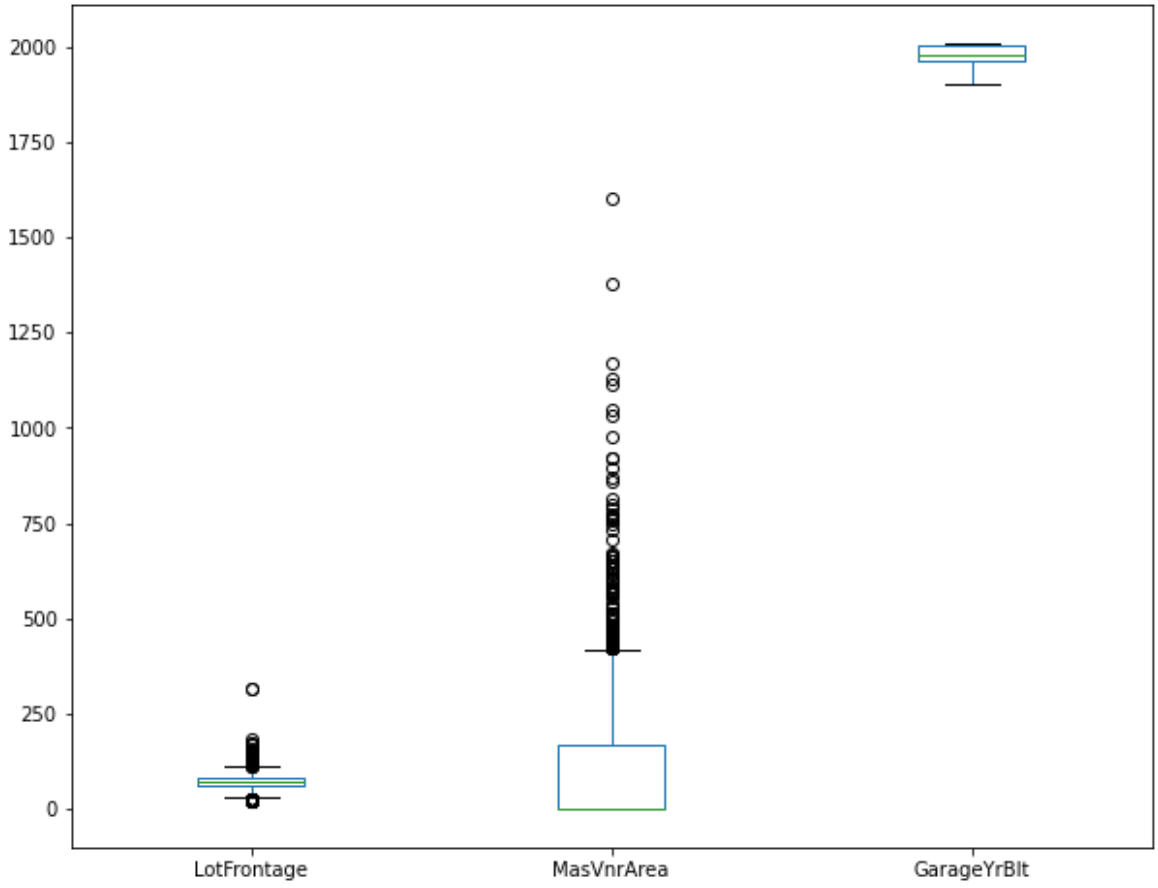

Figure 2 نمودار **boxplot** برای ویژگی‌های عددی با **null**

همانطور که دیده می‌شود GarageYrBlt مقدار outlier ندارد پی برای پر کردن مقادیر null ان از میانگین استفاده می‌کنیم. برای پر کردن دو ویژگی دیگر در Figure 2 از میانه استفاده می‌کنیم. ویژگی‌های دیگر که در زیر آمده است همگی categorical هستند و برای پر کردن آن‌ها از mode استفاده می‌کنیم.

['MasVnrType', 'BsmtQual', 'BsmtCond', 'BsmtExposure', 'BsmtFinType1', 'BsmtFinType2', 'Electrical', 'FireplaceQu', 'GarageType', 'GarageFinish', 'GarageQual', 'GarageCond']

ویژگی id به تحلیل ما ربطی ندارد و به همین دلیل این ویژگی را حذف می‌کنیم. بعد از این مراحل تمام ویژگی‌های مربوط به تحلیل ما باقی مانده اند و هیچ ویژگی ای مقدار null ندارد.

در این قسمت به بررسی outlier های ویژگی‌ها می‌پردازیم. در Figure 3 نمودار boxplot تمام ویژگی‌های عددی آمده است. همان طور که دیده می‌شود فقط دو ویژگی قیمت خانه و LotArea دارای outlier زیاد با فاصله زیاد از box هستند. بقیه ویژگی‌ها نیاز به تغییر ندارند. یک راه برای کم کردن تاثیر

outlier ها تبدیل مقدار آنها به مقادیر نزدیک box است. اما همان طور که در ایجا دیده می‌شود outlier ها در ویژگی مساحت خانه معنا دار هستند و تاثیر مستقیم در قیمت خانه داشته اند.

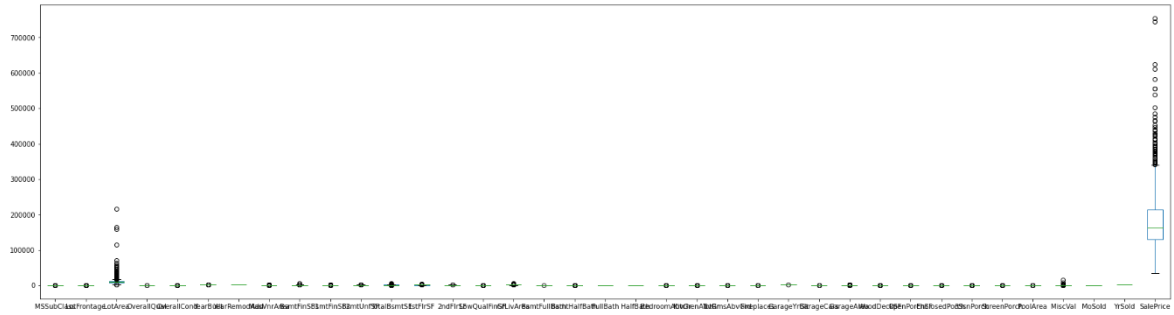

Figure 3 نمودار **boxplot** تمام ویژگی‌های عددی

همانطور که دیده می شود، مقادیر outlier در ویزگی مساحت باعث outlier شدن قیمت خانه می‌شود. یعنی با تغییر مقادیر outlier در ویزگی‌ها مقدار قیمت تغییر خواهد کرد. پس نمی توان مقدار outlier ها را تغییر داد تا اثر آنها در یادگیری کمتر شود. این تغییر outlier ها در مواردی که باعث تغییر ویژگی مورد بررسی نمی‌شود، می‌تواند به یادگیری بهتر و جنرالیزیشن بالاتر کمک کند. عکس Figure 3 در فایل boxplot_all_numerical.png کنار گزارش ارسال می‌شود تا بتوان جزئیات بیشتری در آن دید.

بعد از این مراحل، باید ویژگی‌های غیر عددی را به عدد تبدیل کنیم تا بتوانیم از رگرسیون استفاده کنیم. برای عددی کردن یا encode داده‌های categorical باید ابتدا ترتیب داشتن یا نداشتن آن‌ها را تعیین کنیم. در داده‌های ترتیب دار هر مقدار را با رتبه آن مقدار میان کل مقادیر عوض می‌کنیم. برای فهم بیشتر، encode یک ویژگی ترتیب دار را بررسی می‌کنیم. ExterQual یک ویژگی غیر عددی ترتیب دار است. مقادیر ممکن برای این ویزگی ترتیبی به شکل Figure 4 دارند.

```
'mapping':{'Po':0,'Fa':1,'TA':2,'Gd':3,'Ex':4}
```

Figure 4 ترتیب مقادیر ویژگی **ExterQual**

در اینجا Ex به معنای عالی و Po به معنای ضعیف است. پس رتبه اول یا بیشترین امتیاز مربوط به Ex و کمترین امتیاز باید مربوط به Po باشد. به همین دلیل ما به Ex امتیاز ۴ و به Po امتیاز صفر دادیم. در تمام داده‌ها بر اساس map در Figure 4 مقدار ویژگی exterQual را به امتیاز متناظر آن تبدیل می‌کنیم. برای تمام ویزگی‌های ترتیب دار دیگر مانند همین عمل می‌کنیم. ویژگی‌های بدون ترتیب در زیر آمده اند.

['MSZoning', 'Street', 'Utilities', 'LotConfig', 'Neighborhood', 'Condition1', 'Condition2', 'BldgType', 'HouseStyle', 'RoofStyle', 'RoofMatl', 'Exterior1st', 'Exterior2nd', 'MasVnrType', 'Foundation', 'Heating', 'CentralAir', 'Electrical', 'GarageType', 'SaleType', 'SaleCondition']

برای تبدیل این ویژگی‌ها از روش one-hot استفاده می‌کنیم. در این نوع ویژگی‌ها هیچ ترتیبی وجود ندارد و این که کدام مقدار برای داده موجود بوده مهم است. یعنی اگر یک ویژگی از این نوع دو مقدار a و b می‌گیرد، این که داده برای این ویژگی مقدار a دارد و مقدار b ندارد مهم است. به همین دلیل از روش one-hot استفاده می‌کنیم. در این روش به ازای تمام مقادیر ممکن هر ویژگی ستون اضافه می‌کنیم و این ستون‌ها را با یک و صفر پر می‌کنیم. ستون مربوط به ویژگی اولیه را حذف می‌کنیم. در Figure 5 ستون‌های اضافه شده برای ویژگی GarageType آمده است.

GarageType_Attchd	GarageType_Basment	GarageType_BuiltIn	GarageType_CarPort	GarageType_Detchd
1	0	0	0	0
1	0	0	0	0
1	0	0	0	0
0	0	0	0	1
1	0	0	0	0

Figure 5 ستون‌های اضافه شده برای نمایش **one-hot** ویژگی **GarageType**

سطر اول در Figure 5 به این معنی است که مقدار ویژگی GarageType برابر Attch است. هر سطر فقط در یک ستون می‌تواند یک داشته باشد. بعد از این مراحل تمام مقادیر null ویژگی‌ها پر شده است. تمام ویژگی‌های غیر مرتبط پاک شده است و تمام ویژگی‌ها به عدد تبدیل شده اند. داده‌ها بعد از این پیش پردازش و پاکسازی در فایل cleanQ1.csv ذخیره شده اند که در ادامه از آنها برای رگرسیون و پیش بینی قیمت خانه استفاده می‌کنیم.

ب

در این بخش ابتدا به طور تصادفی ۲۰ در صد داده‌ها را برای تست و بقیه را برای آموزش انتخاب می‌کنیم. برای ایجاد شبکه عصبی و آموزش آن از keras استفاده می‌کنیم. در این جا می‌خواهیم یک شبکه MLP را برای رگرسیون آموزش دهیم. دو حالت برای تعداد لایه‌ها و دو حالت برای تابع فعالساز لایه‌های پنهان برای شبکه MLP در نظر می‌گیریم و آنها را باهم مقایسه می‌کنیم.

ویژگی‌های مختلف در داده‌ها دارای بازه‌ها و range های متفاوتی هستند. این تفاوت بازه می‌تواند باعث افزایش تاثیر ویژگی با بازه‌ی بزرگتر در پیش بینی شود که مطلوب نیست. در این صورت وزن ویژگی با بازه بزرگتر بیشتر می‌شود و اندک تفاوتی در آن مقدار پیش بینی را کلی تغییر می‌دهد. این موضوع ما را

به نویز حساس می‌کند و باعث overfit و کاهش جنرالیزیشن می‌شود. همچنین یکسان نبودن بازه ویژگی‌ها و توزیع آنها مدل سازی را سخت تر می‌کند. برای رفع مشکل بازه متفاوت ویژگی‌ها و توزیع متفاوت آنها از StandardScaler بر روی ویژگی‌های تست و آموزش به طور جدا استفاده می‌کنیم. در این استاندارد سازی قیمت خانه یا y ما برای پیش بینی حضور ندارد. چون می‌خواهیم قیمت مانند آنچه تا کنون گزارش شده برای داده‌های جدید گزارش کنیم و نمی‌خواهیم بازه‌ی تغییر قیمت تغییری کند.

برای انجام مقایسه نیاز است که یک ساختار پایه برای شبکه در نظر بگیریم. برای این کار از MLP با لایه ورودی ۱۹۱ نرونی(به تعداد ویژگی‌های هر داده) ، لایه پنهان ۶۴ نرونی و لایه خروجی یک نرونی با تابع فعالساز خطی استفاده می‌کنیم. برای مقایسه تعداد لایه پنهان، به تعداد لایه‌ی پنهان این ساختار اضافه می‌کنیم که این لایه‌های پنهان همگی ۶۴ نرونی هستند. برای بررسی تابع‌های فعالساز مختلف، تابع فعالساز لایه‌های پنهان را تغییر می‌دهیم. در Figure 6 نحوه ایجاد شبکه برای حالات مختلف آمده است.

```
def create_model(loss_t, nl, active, lr):
  model = Sequential()
  model.add(Dense(64, activation=active, input_shape=(191,)))   #Hidden Layer 1
  for i in range(nl):
    model.add(Dense(64, activation=active)) #Hidden Layer 2, 3, ....
  model.add(Dense(1)) #Last layer with one output positive price
  model.summary()

  # Configure the Network
  opt = Adam(learning_rate=lr)
  model.compile(loss=loss_t, optimizer=opt)
  return model
```

Figure 6 نحوه ایجاد **MLP** برای حالات مختلف برای رگرسیون

برای این که بتوان ساختارهای مختلف را مقایسه کرد، باید در تمام حالات از یک optimizer، یک تابع هزینه و یک learning rate استفاده کنیم. برای همین موضوع ما در تمام حالات از Adam برای بهینه‌سازی و از MSE برای تابع هزینه استفاده کردیم. در تمام موارد نرخ یادگیری 0.03 است. برای حالات مختلف شبکه از یک و دو لایه‌ی پنهان و تابع‌های فعالساز relu و tanh استفاده کردیم. در Figure 7 و Figure 8 نمودار loss برای داده‌های تست و آموزش به ازای هر epoch برای تابع فعالساز relu و لایه‌های پنهان به ترتیب یک و دو آمده است. در Figure 9 و Figure 10 نمودار مقادیر پیش بینی شده به ازای مقدار واقعی برای داده‌های تست به ازای تابع فعالساز relu و لایه‌های پنهان به ترتیب یک و دو آمده است.

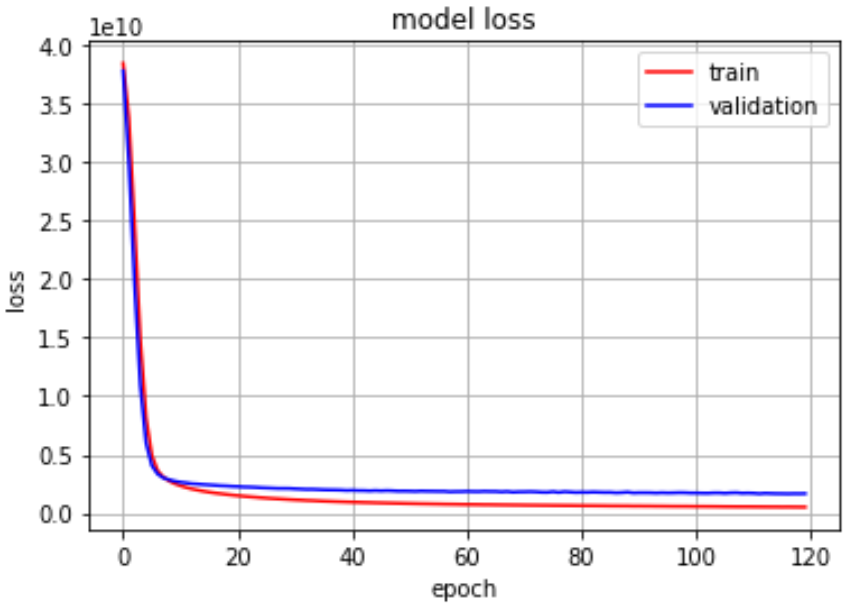

Figure 7 میزان **loss** به ازای هر **epoch** داده آموزش و تست به عنوان **validation** با فعالساز **relu** و یک لایه مخفی

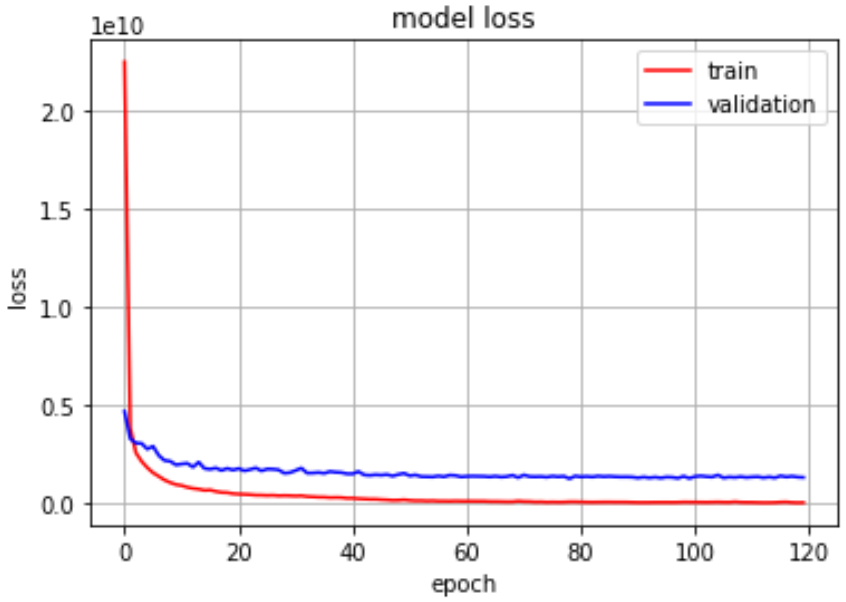

Figure 8 میزان **loss** به ازای هر **epoch** داده آموزش و تست به عنوان **validation** با فعالساز **relu** و دو لایه مخفی

همانطور که در Figure 7 و Figure 8 دیده می‌شود، در هر دو حالت در تعداد epoch کمتر از ۱۰۰ میزان loss روی تست کمینه شده است و یادگیری کامل شده است. این مورد نشان می‌دهد با تابع فعالساز relu یادگیری به سرعت و کمتر از ۱۰۰ تا epoch انجام می‌شود. در حالت دو لایه دیده می‌شود که loss روی تست و آموزش از مقدار کمتری شروع شده است. این یعنی با دیدن یک epoch اول میزان یادگیری در حالت دو لایه مخفی بیشتر از یک لایه مخفی بوده است و مدل توانسته به جواب بهینه بیشتر نزدیک شود.

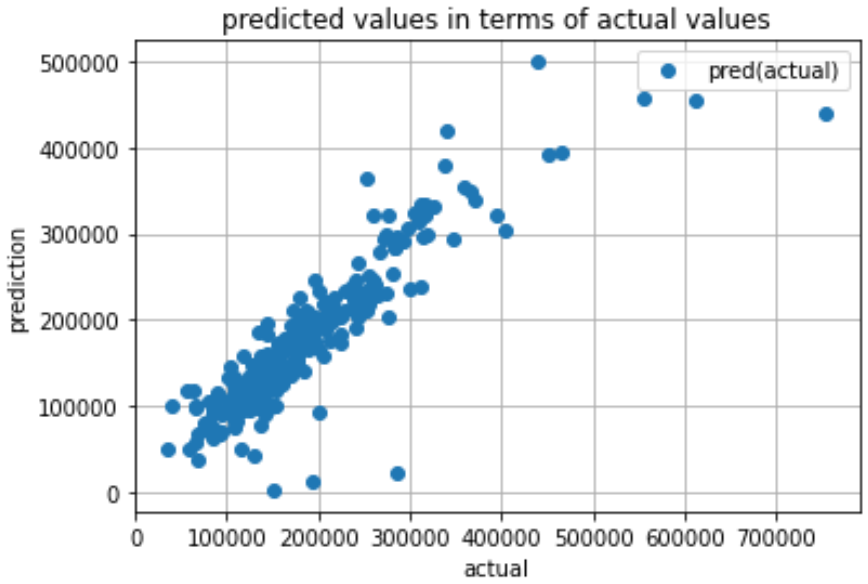

Figure 9 نمودار پیش بینی بر اساس مقدار واقعی تست برای حالت تابع فعالساز **relu** و یک لایه مخفی

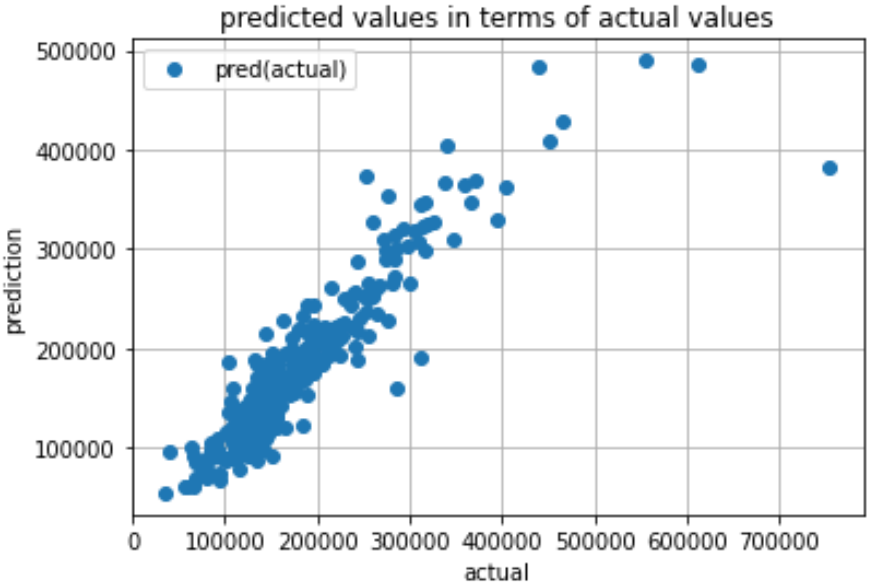

Figure 10نمودار پیش بینی بر اساس مقدار واقعی تست برای حالت تابع فعالساز **relu** و دو لایه مخفی

همانطور که در Figure 9 و Figure 10 دیده می‌شود، در حالت دو لایه نمودار پیش بینی بر اساس مقدار واقعی بیشتر به معادله y=x نزدیک است. در هر دو حالت تمام پیش بینی‌ها مثبت بوده و نقاط نزدیک معادله y=x است اما در حالت یک لایه مجموع فاصله داده‌ها از این خط از حالت دو لایه بیشتر است. در حالت دو لایه نقاط متمرکز و نزدیک خط y=x هستند و فاصله کمی از آن دارند. پراکندگی نقاط اطراف خط کم است. این موضوع برای بیشتر نقاط وجود دارد و فقط تعدادی داده با مقدار واقعی خیلی

بالا دارای پیش بینی کمتر و دور از واقعیت هستند و پراکنده و دور از خط اند. در حالت یک لایه بعد از مقدار واقعی ۳۰۰۰۰۰ نقاط دارای تمرکز پایینی هستند و پراکنده اطراف y=x قرار دارند. بعد از این مقدار فاطله نقاط از خط بالا هست. در قبل از این مقدار نیز تعدادی نقطه با فاصله از خط وجود دارد که از تمرکز داده‌ها دور هستند. این نقاط پراکنده که از حالت دو لایه تعداد بیشتری دارند باعث افزایش مجموع فاصله نقاط از خط می‌شوند. می دانیم بهترین و بهینه ترین پیش بینی زمانی است که معادله پیش بینی بر اساس واقعیت y=x باشد و تمام نقاز متمرکز روی این خط باشند. مجموع فاصله نقاط از این خط کم باشد و پراکندگی نقاط همانطور که گفته شد پایین باشد. پس بر این اساس حالت دو لایه با تابع فعالساز relu عملکرد بهتری بر روی داده تست دارد و ساختار بهتری است.

در Figure 11 و Figure 13 نمودار loss برای داده‌های تست و آموزش به ازای هر epoch برای تابع فعالساز tanh و لایه‌های پنهان به ترتیب یک و دو آمده است. در Figure 12 و Figure 14 نمودار مقادیر پیش بینی شده به ازای مقدار واقعی برای داده‌های تست به ازای تابع فعالساز tanh و لایه‌های پنهان به ترتیب یک و دو آمده است.

Figure 11 مقدار **loss** در هر **epoch** برای داده آموزش و تست که **validation** نامیده شده است برای یک لایه پنهان و تابع فعالساز **tanh**

همانطور که در Figure 11 دیده می‌شود، در بیش از ۱۰۰۰ تا epoch مقدار کمینه loss برای تست بدست آمده است. اینجا تست، validation نامیده شده است. پس در حالت تابع فعالساز tanh بسبت به حالت با همین ساختار و تابع فعالساز relu، آموزش کندتر انجام می‌شود. تعداد epoch لازم برای کمینه

شدن loss داده تست در حالت tanh بسیار بیشتر از relu است. در Figure 12 نمودار پیش بینی بر اساس مقدار واقعی برای یک لایه پنهان و تابع فعالساز tanh آمده است.

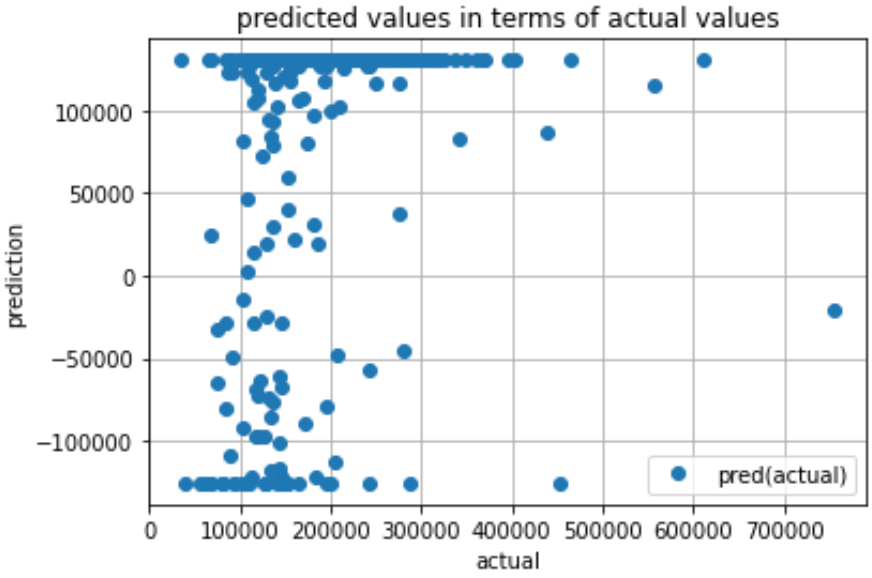

Figure 12 نمودار پیش بینی بر اساس واقعیت داده تست با یک لایه مخفی و تابع فعالساز **tanh**

همانطور که در Figure 12 دیده می‌شود، پیش بینی‌ها کاملا با واقعیت متفاوت است و این یعنی در مینیمم محلی گیر کرده ایم. پس این ساختار با تعداد epoch خیلی بیشتر از حالت relu نتوانسته یادگیری خوبی را انجام دهد. این دلیل باعث می‌شود این ساختار را برای رگرسیون انتخاب نکنیم. در Figure 13 مقدار loss در هر epoch برای دو لایه پنهان و تابع فعال ساز tanh آمده است. همانطور که دیده می‌شود تا ۳۰۰۰ تا epoch نیز همچنان loss بر روی داده تست در حال کم شدن است.

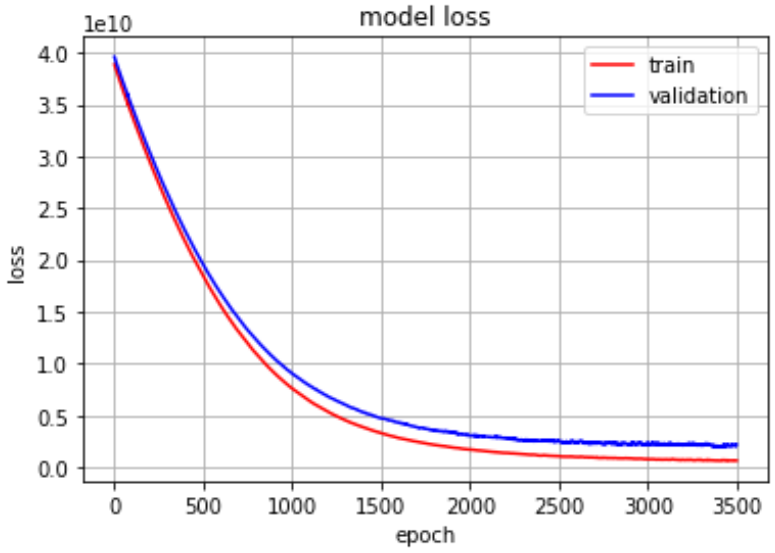

Figure 13 مقدار **loss** در هر **epoch** برای دو لایه پنهان و تابع فعال ساز **tanh**

در Figure 14 نمودار پیش بینی بر اساس مقدار واقعی قیمت خانه برای داده تست آمده است. همانطور که دیده می‌شود، برای قیمت‌های واقعی بالا پراکندگی بالا داریم. بقیه نقاط اطراف یک خط متمرکز هستند که شیب آن از y=x بیشتر است. این یعنی ما پیش بینی بیش از واقعیت داریم. این نمودار نشان می‌دهد که نتوانسته‌ایم قیمت خانه را برای داده دیده نشده خوب پیش بینی کنیم و با ۳۰۰۰ تا epoch یادگیری خوب نداشته ایم. پیش بینی از حالت دو لایه و یک لایه با relu بد تر است. به همین دلیل ساختار با دو لایه و تابع فعالساز tanh را به عنوان ساختار مناسب انتخاب نمی‌کنیم. ساختار با دو لایه پنهان و تابع فعالساز tanh توانسته یادگیری بهتری از حالت یک لایه داشته باشد چون پیش بینی ها به واقعیت نزدیک تر هستند. این به دلیل توان یادگیری و مدل کردن بیشتری است که MLP با دو لایه پنهان ایجاد می‌کند. تمام ساختارهای با تابع فعالساز tanh به تعداد epoch خیلی بیشتر از ساختارهای با relu نیاز دارند و کند تر از آن هستند. این می‌تواند به دلیل gradient vanishing و اندازه‌ی خیلی کم گرادیان باشد.

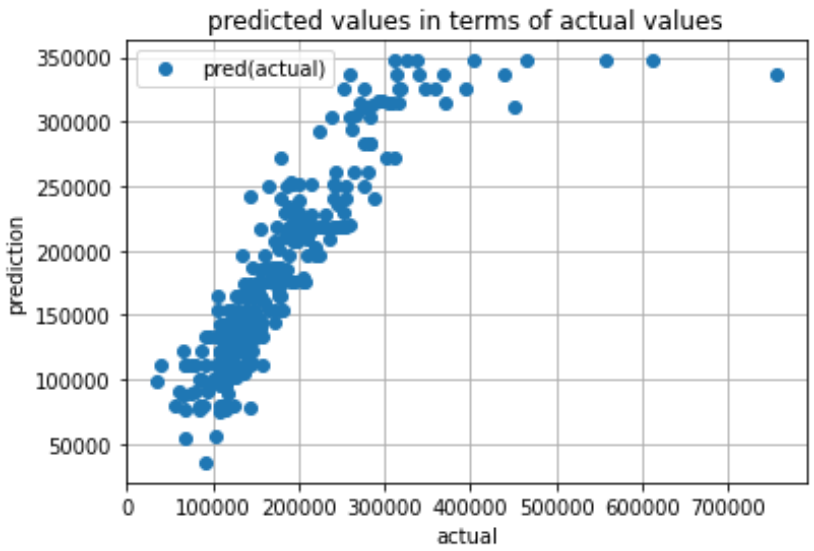

Figure 14 پیش بینی بر اساس مقدار واقعی قیمت خانه در داده تست با دو لایه پنهان و تابع فعالساز **tanh**

همانطور که در چهار حالت مورد بررسی دیدیم، دو لایه پنهان قدرت یادگیری و مدل کردن بالاتری را ایجاد می‌کند. همچنین دیدیم که تابع فعالساز tanh یادگیری را به دلیل اندازه کم گرادیان و gradient vanishing کند می‌کند. چون این تابع خروجی را بین منفی یک و یک نگه می‌دارد. به همین دلایل و با بررسی پیش بینی ساختارهای مختلف در برابر مقادیر واقعی، دیدیم که ساختار MLP با دو لایه پنهان و تابع فعالساز relu بهترین عملکرد را دارد.

ج

در اینجا از ساختار MLP با دو لایه مخفی و تابع فعالساز relu که بهترین ساختار قسمت قبل بود استفاده می‌کنیم. در این قسمت تابع هزینه را MSE تعریف می‌کنیم و متریک‌های MSE و MAE را در هر epoch در یادگیری بدست می‌آوریم. در یادگیری از ModelCheckpoint استفاده می‌کنیم که هر بار loss برای داده تست کمتر شود، وزن‌های یادگرفته شده مدل را نگه می‌دارد. این باعث می‌شود که بهترین مدل بدون overfit برای تعداد epoch بیشتر در یادگیری را داشته باشیم. همچنین ModelCheckpoint در هر epoch اعلام می‌کند که آیا به loss بهتری رسیده ایم یا نه. از روی این اعلام ها می‌توان به تعداد epoch که بعد آن overfit رخ می‌دهد رسید. این تعداد جایی است که بعد آن loss دیگر کمتر نخواهد شد. این تعداد epoch تعداد بهینه epoch برای یادگیری است. در Figure 15 نمودار MSE به ازای داده‌های آموزش و تست در کنار هم آمده است. آخرین بهبود loss مربوط به epoch شماره ۶۰ (تعداد بهینه epoch)است.

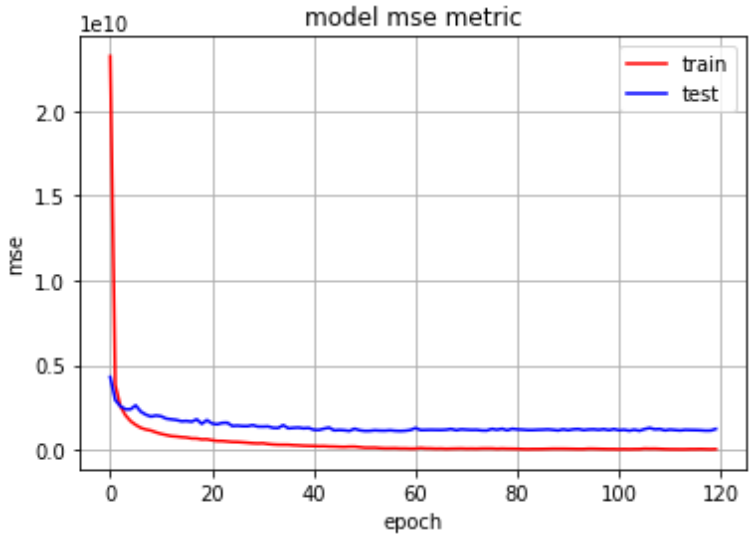

Figure 15 مقدار **MSE** در هر **epoch** برای تست و **train** وقتی تابع هزینه **MSE** است

در Figure 16 نمودار MAE به ازای داده‌های آموزش و تست در کنار هم آمده است. همانطور که در Figure 15 و Figure 16 دیده می‌شود، دو متریک در طول یادگیری روی داده آموزش کم می‌شوند. بر روی داده تست نیز دو متریک عملکرد مشابهی دارند، ابتدا کم می‌شوند و از جایی به بعد ثابت می‌مانند. در اینجا ما از MSE به عنوان تابع هزینه استفاده کردیم و سعی کردیم وزن‌ها را طوری یاد بگیریم که این هزینه را کمینه کنیم. همان طور که دیده می‌شود، عملکرد MAE در epoch ها مانند MSE است. این یعنی در این روند یادگیری ما هزینه مبتنی بر MAE را نیز کم کرده ایم و توانسته ایم به وزن‌هایی برسیم که مقدار MSE و MAE را کمینه کنیم. یعنی توانسته ایم فاصله پیش بینی از واقعیت را کمینه کنیم.

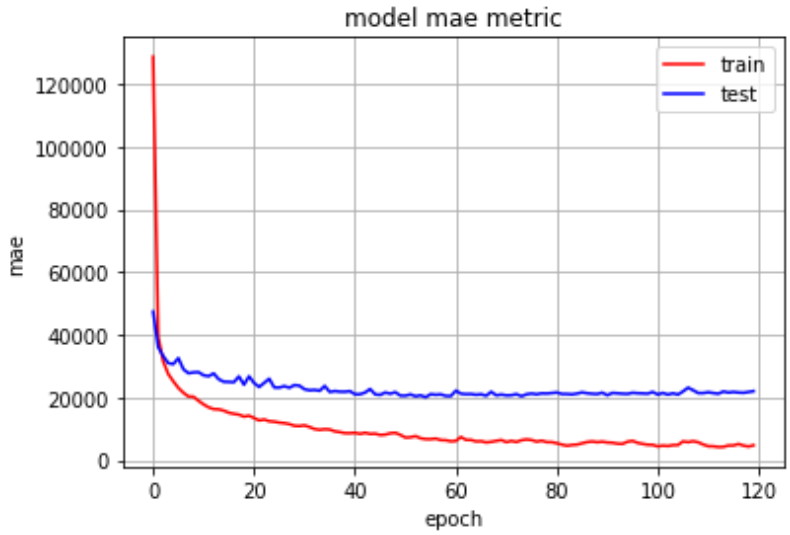

Figure 16 مقدار **MAE** در هر **epoch** برای تست و **train** وقتی تابع هزینه **MSE** است

در Figure 17 نمودار قیمت پیش بینی شده به ازای قیمت واقعی داده‌های تست آمده است. همانطور که دیده می‌شود نقاط متمرکز اطراف خط y=x قرار دارند. تمرکز در همه‌ی قیمت‌ها یکی است. برای مقادیر واقعی بزرگ نیز پیش بینی‌ها به واقعیت نزدیک است و این نقاط نیز در فاصله نزدیکی از y=x قرار دارند. نکته قابل توجه اینجا است تعداد نقاط با اختلاف زیاد از خط y=x بسیار کم است و مجموع فاصله نقاط دور از این خط نیز کم است. این نزدیکی به خط y=x و تمرکز نقاط در فاصله نزدیک آن نشان می‌دهد که پیش بینی‌ها نزدیک واقعیت بوده و map به خوبی انجام شده و یادگیری مناسب بوده است. توانسته ایم روی داده دیده نشده عملکرد خوبی داشته باشیم و تعمیم خوبی ایجاد کنیم. نمودار برحسب بهترین مدل بدست آمده از ModelCheckpoint است.

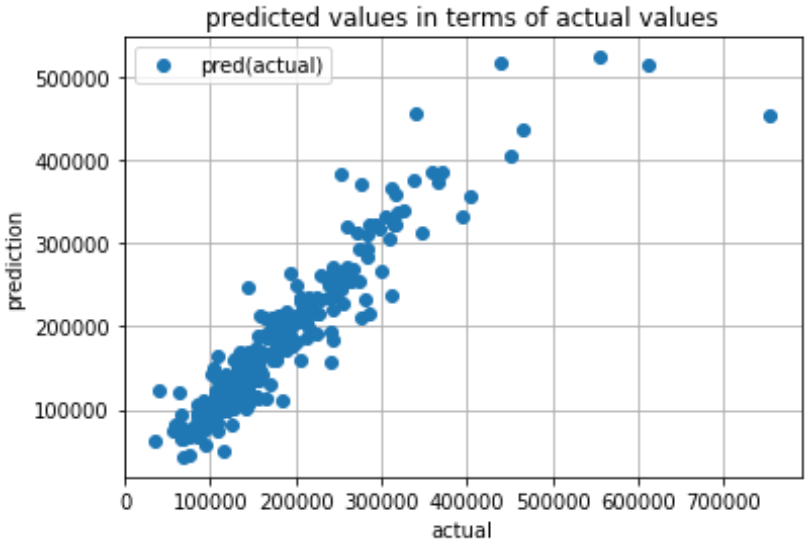

Figure 17 نمودار قیمت پیش بینی شده به ازای قیمت واقعی داده‌های تست با تابع هزینه **MSE**

دو متریک MSE و MAE برای ارزیابی بعد از هر epoch استفاده می‌شود. این دو معیار بررسی می‌کنند که پیش بینی ها چقدر به مقادیر واقعی نزدیک هستند. با انجام epoch های بیشتر پیش بینی‌های ما به مقادیر واقعی داده‌های آموزش نزدیک تر می‌شود. چون برای این هدف وزن‌ها را تغییر می‌دهیم. به همین دلیل با افزایش epoch دو نمودار MSE و MAE روی داده‌های آموزش کاهش می‌یابد. میزان نزدیکی پیش بینی به واقعیت بر روی داده‌های تست، معیار ما برای تشخیص مدل بهینه است. زمانی که تفاوت پیش بینی از واقعیت داده تست کمینه است دو معیار کمینه می‌شوند (۶۰ امین epoch) و در این زمان ما بهترین مدل را داریم. البته نگاه دو متریک به فاصله پیش بینی از واقعیت یکسان نیست.

د

در اینجا از تابع MAE به عنوان تابع هزینه استفاده می‌کنیم و مقدار معیارهای MSE و MAE را بعد از هر epoch بدست می‌آوریم و رسم می‌کنیم. در Figure 18 نمودار MSE به ازای داده‌های آموزش و تست در کنار هم آمده است. در Figure 19 نمودار MAE به ازای داده‌های آموزش و تست در کنار هم آمده است.

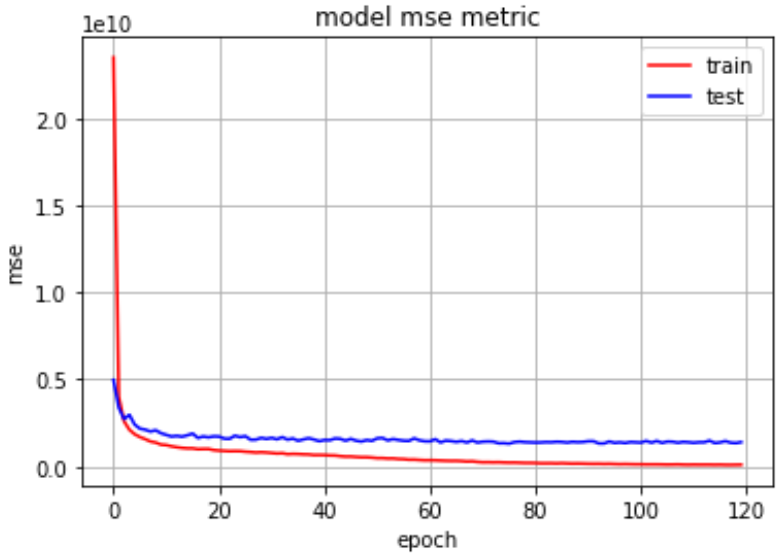

Figure 18 نمودار مقدار **MSE** بعد از هر **epoch** با تابع هزینه **MAE**

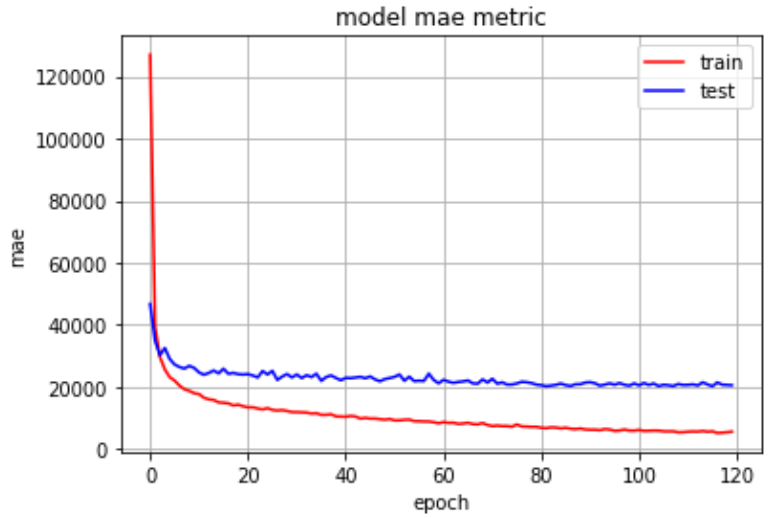

Figure 19نمودار مقدار **MAE** بعد از هر **epoch** با تابع هزینه **MAE**

همانطور که در Figure 18 و Figure 19 دیده می‌شود، با افزایش تعداد epoch ها مقدار متریک‌ها برای داده‌های آموزش کم می‌شود و پیش بینی‌ها برای داده آموزش به واقعیت نزدیک تر می‌شود. متریک‌ها برای داده‌های تست ابتدا کاهش می‌یابد و سپس ثابت می‌شود. کمترین loss در ۱۱۸ امین epoch رخ می‌دهد که تعداد epoch بهینه است. کمترین مقدار متریک‌ها و loss در epoch بهینه رخ می‌دهد که جایی است که تفاوت پیش بینی برای داده تست کمترین فاصله از واقعیت را دارد. نمودار MSE و MAE برای داده تست روند مشابهی دارند. نکته قابل توجه افزایش تعداد epoch بهینه نسبت به قسمت قبل است. در Figure 20 پیش بینی بر اساس واقعیت آمده است.

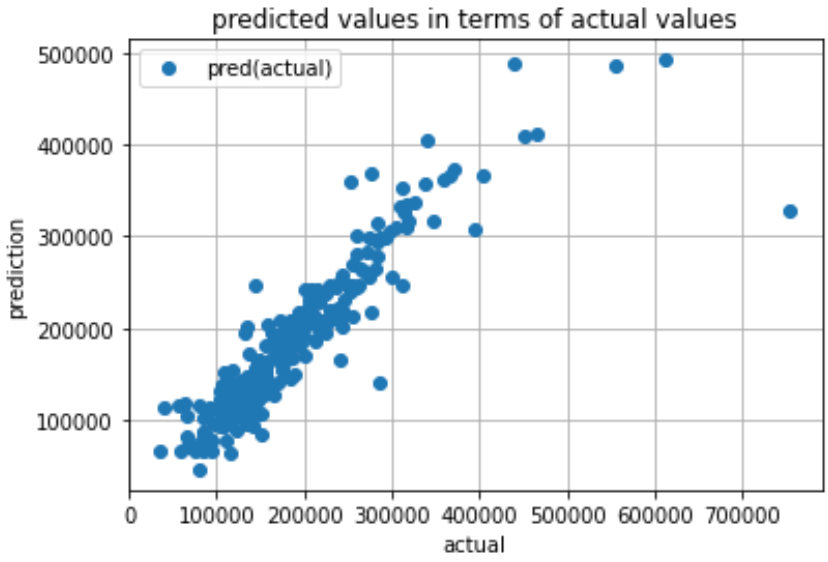

Figure 20 پیش بینی بر اساس واقعیت برای داده‌های تست با تابع هزینه **MAE**

همانطور که در Figure 20 دیده می‌شود، تعداد نقطه‌ها با فاصله زیاد از خط y=x نسبت به بخش قبل بیشتر است. نقاط قبل از ۲۰۰۰۰۰ به صورت متمرکز و با پراکندگی کم اطراف خط قرار دارند اما بعد از آن تمرکز نقاط اطراف خط کمتر می‌شود و پراکندگی داده‌ها اطراف خط زیاد تر می‌شود. تعداد نقاط با فاصله زیاد در نقاط پراکنده اطراف خط زیاد است و از بخش قبل بیشتر است. در بخش قبل پراکندگی اطراف خط داشتیم و نقاطی از خط فاصله داشتند ولی این نقاط در فاصله کمی از خط قرار داشتند.

۵

تابع هزینه MSE و MAE در Figure 21 و Figure 22 آمده است. هر دو از متوسط فاصله پیش بینی از واقعیت استفاده می‌کنند.

$$\mathrm{MAE} = \frac{1}{n}\sum_{j=1}^{n}|y_j - \hat{y}_j|$$

Figure 21 تابع **MAE**

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

Figure 22 تابع **MSE**

در MSE از مربع فاصله هر پیش بینی نسبت به واقعیت استفاده می‌شود این در حالی است که در MAE از قدر مطلق فاصله هر پیش بینی نسبت به واقعیت استفاده می‌شود. این استفاده از توان دوم فاصله در MSE باعث می‌شود که هر چه فاصله پیش بینی از واقیت بیشتر باشد، هزینه به صورت توان دو بیشتر شود. نسبت به MAE هزینه این موارد بیشتر است. این یعنی MSE به فاصله زیاد اهمییت زیادتری می‌دهد و سعی می‌کند که تمام پیش بینی‌ها فاصله کمی از واقعیت داشته باشند. این موضوع را در مقایسه Figure 20 و Figure 17 می‌توان دید. همانطور که در Figure 20 دیده می‌شود، تعداد نقطه‌ها با فاصله زیاد از خط y=x نسبت به بخش ج و Figure 17 بیشتر است. نقاط قبل از ۲۰۰۰۰۰ به صورت متمرکز و با پراکندگی کم اطراف خط قرار دارند اما بعد از آن تمرکز نقاط اطراف خط کمتر می‌شود و پراکندگی داده‌ها اطراف خط زیاد تر می‌شود. تعداد نقاط با فاصله زیاد در نقاط پراکنده اطراف خط زیاد است و از بخش ج و Figure 17 بیشتر است. در Figure 17 پراکندگی اطراف خط داشتیم و نقاطی از خط فاصله داشتند

ولی این نقاط در فاصله کمی از خط قرار داشتند. این به دلیل اهمیت بیشتر MSE به فاصله زیاد (نگرانی بیشتر نسبت به outlier)و یکسان دیدن این موارد با دیگر موارد در MAE است. همچنین دیدیم که تعداد بهینه epoch در بخش ج کمتر از د است. این موضوع نیز می‌تواند به دلیل اهمیت بیشتر MSE به فاصله زیاد باشد. اگر فاصله نقاط از واقعیت زیاد شود در MSE میزان Loss بسیار بیشتر از MAE می‌شود و باعث می‌شود که میزان گرادیان بیشتر شود و وزن‌ها تغییرات بیشتری داشته باشند. این به این معنی است که گام ما در MSE بزرگتر از MAE در جهت بهینه محلی است و ما سریع تر به نقطه بهینه می‌رسیم. این موضوع باعث می‌شود با MSE تعداد epoch بهینه کمتر شود. اگر پیش بینی تمام داده‌ها فاصله کمی از واقعیت نداشته باشد(فاصله کمتر از یک) مقدار MSE بزرگتر از MAE می‌شود(در این تمرین قیمت‌ها فاصله‌های بزرگی دارند چون بازه قیمت‌ها بزرگ است). این موضوع باعث بیشتر شدن اندازه گرادیان در روش با MSE می‌شود و باعث می‌شود گام بزرگتری در جهت کمینه محلی برداریم. این موضوع باعث کمتر شدن تعداد epoch زمان استفاده از MSE می‌شود.

سوال ۲ – (Classification) MLP

الف

مجموعه داده سوال شامل 208 داده است. دو کلاس R یا rock و کلاس M یا metal cylinder کلاس‌های ما هستند. هر داده یک الگو است که از طریق جهش سیگنال های سونار از یک سیلندر در زوایای مختلف و تحت شرایط مختلف بدست آمده است. لیبل هر داده R یا M است که نشان دهنده‌ی جنس سیلندر است. در کل ۱۱۱ داده مربوط به فلز و ۹۷ داده مربوط به سنگ داریم. در Figure 23 تعداد داده در هر کلاس آمده است.

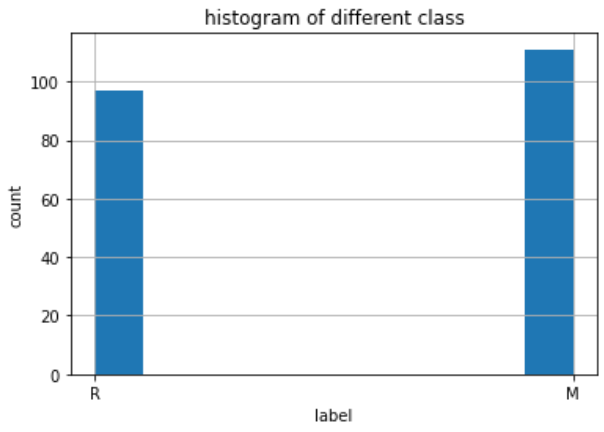

Figure 23 تعداد داده در هر کلاس مجموعه داده **sonar**

هر داده شما ۶۰ عدد بین صفر و یک است که هر کدام مقدار انرژی در یک باند فرکانسی خاص را مشخص می‌کند. همانطور که گفته شد، جهش های سیگنال دریافتی مربوط به زوایای مختلف هستند. نکته مهم این است که زاویه دریافتی در تعیین لیبل ماثر است. به همین دلیل نمی‌توان داده‌ها را به مجموعه آموزش، ارزیابی و تست تقسیم کرد. چون در این روش ممکن است در داده‌های تست مواردی مربوط به یک زاویه خاص وجود داشته باشد که در داده آموزش هیچ نمونه ای از آن نداریم. برای زاویه های مختلفی این اتفاق ممکن است رخ دهد. این مورد باعث می‌شود که یادگیری خوبی نداشته باشیم و نتوانیم برای داده‌های مربوط به زاویه‌هایی در داده تست پیش بینی خوبی داشته باشیم. این موضوع باعث کاهش عملکرد بر روی داده دیده نشده تست می‌شود.

دلیل این ضعف فرایند یادگیری نیست. دلیل آن این است که مجموعه آموزش نماینده مناسبی برای جامعه نیست. برای حل این مشکل باید در هر سه دسته آموزش، ارزیابی و تست به اندازه کافی نمونه از هر زاویه داشته باشیم. این موضوع باعث می شود داده آموزش نماینده مناسبی برای کل جامعه باشد و ما بتوانیم الگوی موجود بین داده‌های هر دسته در جامعه اصلی را یاد بگیریم. برای این کار از نمونه برداری stratified بر اساس زاویه داده‌ها استفاده می‌کنیم.

نکته دیگر این است که زاویه مربوط به هر داده در مجموعه داده مشخص نیست. نیاز داریم زاویه داده‌ها را مشخص کنیم تا بعد بر اساس این زاویه‌ها مجموعه‌های آموزش، ارزیابی و تست را تشکیل دهیم. میدانیم که داده‌های مربوط به یک زاویه شباهت زیادی به هم دارند و شباهت داده‌های دو زاویه مختلف به هم کم است. پس با توجه به این موضوع برای مشخص کردن زاویه می‌توان از خوشه بندی داده‌ها استفاده کرد. هر خوشه مشخص کننده‌ی یک زاویه است. برای خوشه بندی از KMeans استفاده می‌کنیم. برای تعیین k در kmeans از silhouette_score استفاده می‌کنیم. در Figure 24 مقدار silhouette_score به ازای هر k آمده است. k با بیشترین silhouette_score بهترین و بهینه ترین انتخاب است.

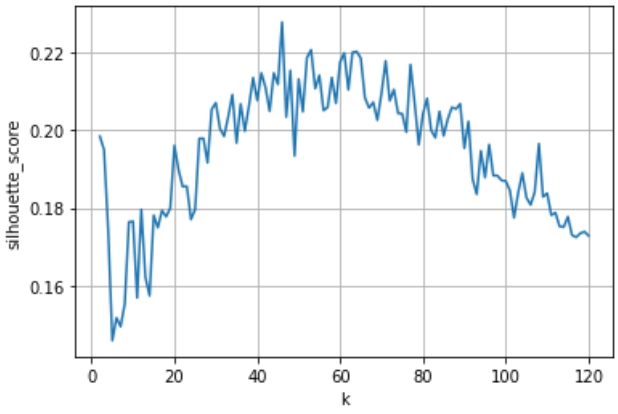

Figure 24 مقدار **silhouette_score** به ازای هر **k**

بر اساس Figure 24 بیشترین مقدار در k=46 اتفاق می‌افتد. البته نمودار بسیار نویزی است و باید بر اساس روند کلی آن تصمیم گرفت. ۴۰ مقدار مناسب است. نکته دیگری که باید در نظر بگیریم این است که k باید به گونه‌ای باشد که بتوان از هر زاویه حداقل یک نمونه در سه دسته آموزش، ارزیابی و تست داشته باشیم. یعنی هر خوشه باید حداقل سه نمونه داشته باشد. برای این اتفاق باید k های کوچک تر از ۴۰ را بررسی کنیم و بزرگترین k را که شرط حداقل تعداد هر خوشه را برقرار می‌کند انتخاب کنیم. با توجه به این موارد k=25 مقدار بهینه برای تعداد خوشه است. بعد از خوشه بندی یک ستون به داده‌ها اضافه می‌کنیم که در آن به ازای هر داده شماره خوشه آن را نگه می‌داریم. خوشه بندی داده‌ها برای بدست آوردن داده‌های مربوط به یک زاویه بر اساس کار گورمن و سجینوفسکی[1] است.

برای تعیین داده‌های آموزش یک نمونه به اندازه‌ی ۵۰ درصد کل داده‌ها به روش stratified از کل انتخاب می‌کنیم. برای تعیین داده‌های ارزیابی ۵۰ درصد داده‌های غیر آموزش را مانند قبل نمونه برداری می‌کنیم. داده‌های خارج از دو دسته قبل داده‌های تست است. برای نمونه برداری stratified، داده‌ها را بر اساس شماره خوشه گروه بندی می‌کنیم. برای تعیین داده آموزش از هر گروه نصف داده‌ها را انتخاب می‌کنیم. برای تعیین داده ارزیابی هم به همین شکل عمل می‌کنیم. تعداد داده هر دسته در داده‌های ارزیابی و آموزش در Figure 25 و Figure 26 آمده است.

	cluster_n	size	samp_size
0	0	14	7
1	1	16	8
2	2	7	4
3	3	7	4
4	4	13	6
5	5	6	3
6	6	8	4
7	7	6	3
8	8	6	3
9	9	7	4
10	10	10	5
11	11	9	4
12	12	5	2
13	13	5	2
14	14	4	2
15	15	10	5
16	16	6	3
17	17	7	4
18	18	7	4
19	19	21	10
20	20	5	2
21	21	4	2
22	22	9	4
23	23	12	6
24	24	4	2

Figure 25 تعداد داده هر خوشه در داده آموزش

[1] Gorman, R. P., and Sejnowski, T. J. (1988). “Analysis of Hidden Units in a Layered Network Trained to Classify Sonar Targets” in Neural Networks, Vol. 1, pp. 75-89.

	cluster_n	size	samp_size
0	0	7	3
1	1	8	4
2	2	3	1
3	3	3	1
4	4	7	3
5	5	3	1
6	6	4	2
7	7	3	1
8	8	3	1
9	9	3	1
10	10	5	2
11	11	5	2
12	12	3	1
13	13	3	1
14	14	2	1
15	15	5	2
16	16	3	1
17	17	3	1
18	18	3	1
19	19	11	5
20	20	3	1
21	21	2	1
22	22	5	2
23	23	6	3
24	24	2	1

Figure 26 تعداد داده هر خوشه در داده ارزیابی

این روش تعیین داده‌های آموزش، ارزیابی و تست باعث می‌شود که در هر سه دسته از هر زاویه مربوط به دریافت سیگنال سونار، نمونه داشته باشیم. این موضوع باعث می‌شود که داده آموزش نماینده مناسبی از جامعه باشد و بتوان بر اساس آن الگوهای مربوط به دو کلاس در جامعه اصلی را یاد گرفت. این کار باعث افزایش قدرت تعمیم مدل‌هایی می‌شود که بر اساس این داده‌ها آموزش می‌بینند. این مورد باعث می‌شود برای داده‌های دیده نشده از جامعه عملکرد خوبی داشته باشیم.

اعداد برای هر نمونه بین صفر و یک هستند و همگی یک بازه دارند. ولی ممکن است توزیع‌های متفاوتی داشته باشند. برای هم بازه و هم توزیع کردن ویژگی‌های هر داده از StandardScaler استفاده می‌کنیم. این پیش پردازش را قبل از انجام دسته‌بندی به صورت جدا بر روی ویزگی‌های داده‌های آموزش، ارزیابی و تست انجام می‌دهیم. لیبل‌ها را نیز به صورت one-hot در می‌آوریم چون می‌خواهیم در مدل‌ها احتمال تغلق به هر کلاس را در بیاوریم.

برای طبقه بندی داده‌ها از MLP با دو لایه مخفی استفاده می‌کنیم. لایه ورودی دارای ۶۰ نرون به اندازه‌ی ویژگی‌ها است. لایه مخفی اول شامل ۴۸ نرون با تابع فعالساز relu است. بایه مخفی دوم شامل ۲۴ نرون و تابع فعالساز relu است. لایه خروجی نیز شامل ۲ نرون به اندازه‌ی تعداد کلاس‌ها با تابع فعالساز softmax است. شبکه در Figure 27 آمده است.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 48)                2928
_________________________________________________________________
dense_1 (Dense)              (None, 24)                1176
_________________________________________________________________
dense_2 (Dense)              (None, 2)                 50
=================================================================
Total params: 4,154
Trainable params: 4,154
Non-trainable params: 0
_________________________________________________________________
```

Figure 27 ساختار شبکه برای **MLP** با دو لایه مخفی

ب

در این بخش مدل توصیف شده در قسمت قبل را آموزش می‌دهیم. برای آموزش از بهینه ساز adam و تابع هزینه categorical_crossentropy استفاده می‌کنیم. در طول آموزش متریک accuracy را رسد می‌کنیم. در فرآیند یادگیری از ModelCheckpoint استفاده می‌کنیم تا مدل با کمترین loss روی داده ارزیابی را نگه داریم. در این جا از اندازه‌ی batch=32 برای یادگیری استفاده می‌کنیم. نمودار تغییرات خطا در هر epoch بر روی داده‌های آموزش و ارزیابی در Figure 28 آمده است.

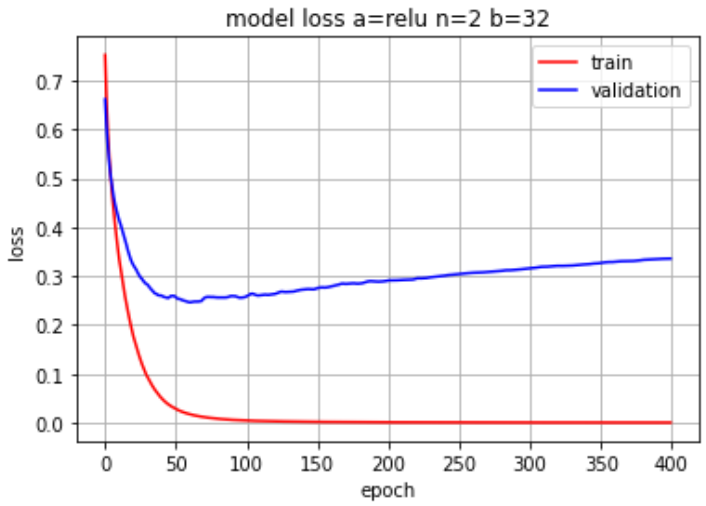

Figure 28 تغییرات خطا در **epoch** های مختلف بر روی داده آموزش و ارزیابی

نمودار تغییرات دقت در هر epoch بر روی داده‌های آموزش و ارزیابی در Figure 29 آمده است.

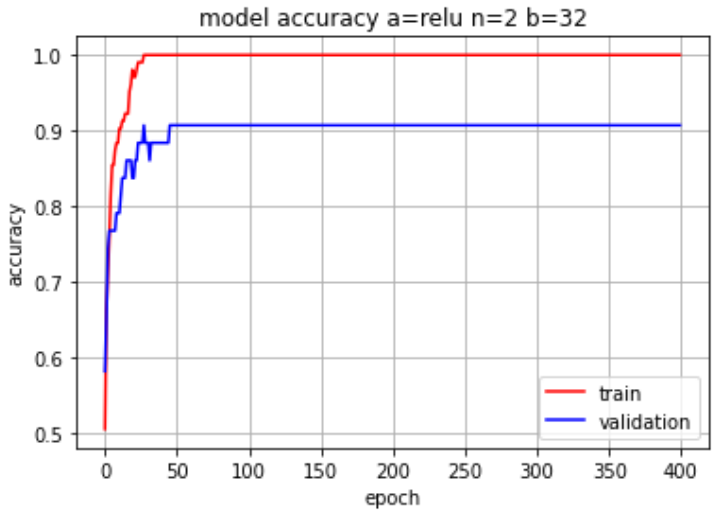

Figure 29 تغییرات دقت در **epoch** های مختلف روی داده آموزش و ارزیابی

ج

مقدار خطا و دقت مدل آموزش دیده شده بر روی داده تست در Figure 30 آمده است. ماتریس confusion برای داده‌های تست در آمده است.

```
2/2 [==============================] - 0s 6ms/step - loss: 0.5469 - accuracy: 0.7880
Test Loss 0.4573431611061096
Test Accuracy 0.8225806355476379
```

Figure 30 مقدار دقت و خطا بر روی داده‌های تست با مدل پایه با دو لایه مخفی

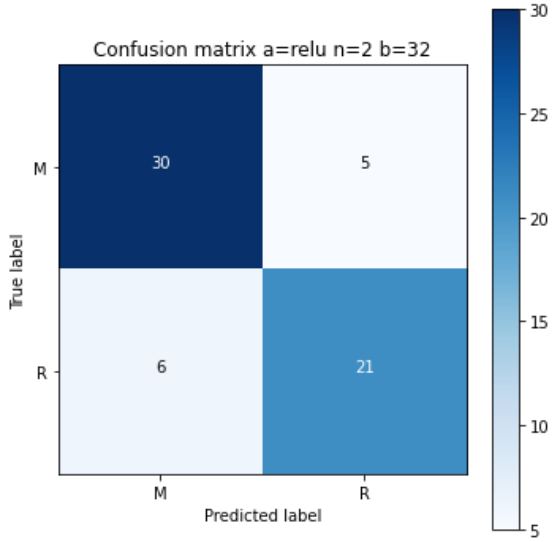

Figure 31 ماتریس **confusion** روی داده‌های تست

د

معیار خطای مورد استفاده ما cross entropy برای دو کلاس است. از ای معیار به صورت گسترده برای بهینه سازی در مسائل طبقه بندی استفاده می‌شود. می توانیم بفهمیم که ایده آنتروپی برای بهینه سازی یک مدل طبقه بندی مفید است. هر داده دارای یک برچسب کلاس شناخته شده با احتمال 1.0 و احتمال 0.0 برای همه برچسب های دیگر است. یک مدل می تواند احتمال تعلق داده به هر برچسب را تخمین بزند. پس از آنتروپی می توان برای محاسبه تفاوت بین این دو توزیع احتمال استفاده کرد. معیار cross entropy این تفاوت را حساب می‌کند. ما علاقه مندیم که cross entropy را برای کل مجموعه داده های آموزشی به حداقل برسانیم. میانگین cross entropy در تمام نمونه های آموزشی را می‌خواهیم کمینه کنیم.

جایگزین این معیار استفاده شده می‌تواند KL divergence باشد. در زیر رابطه cross entropy و KL آمده است.

$$D_{KL}(p,q) = H(q,p) - H(p)$$

همانطور که دیده می‌شود تنها تفاوت در $H(p)$ است که p توزیع لیبل داده‌ها و q توزیع پیش بینی شده است. عبارت $H(p)$ ثابت است چون p مشخص است. پس کمینه کردن دو معیار یکسان است. در حالت minibatch با توجه به داده‌های batch ممکن است توزیع لیبل‌های batch یا p' با توزیع p یکسان نباشد. این مورد باعث ایجاد مشکل برای KL می‌شود. این درحالی است که cross entropy به این موضوع مقاوم تر است به همین دلیل از آن استفاده کردیم.

جایگزین دیگر hinge است. در این معیار لیبل‌ها مثبت و منفی در نظر گرفته می‌شود و می‌خواهیم تفاوت علامت داده‌ها کمینه باشد. در اینجا اطمینان دسته بندی علاوه بر دسته بندی درست اهمیت دارد. مانند آنچه در svm است این معیار به دنبال جدا سازی دو دسته با بیشترین margin است. در آزمایشات معمولا cross entropy عملکرد بهتری از آن دارد به همین دلیل از آن استفاده نکردیم.

ه

خیر. در قسمت الف در Figure 23 دیدیم که تعداد داده‌های دو دسته برابر نیست. در اصطلاح مجموعه داده بالانس نمی‌باشد. متریک accuracy این پیش فرض را دارد که داده ها بالانس است به همین دلیل برای نشان دادن عملکرد مناسب نیست. اگر یک مدل داشته باشیم که همه داده ها را به کلاس با تعداد بیشتر نسبت دهد، مقدار accuracy بالا می‌رود ولی مدل عملا هیچ چیزی یاد نگرفته است. مقدار false

negative و false positive در تعیین عملکرد طبقه بند ماثر هستند. یکه طبقه بند خوب مدلی است که FN و FP در پیش بینی آن پایین باشد. در مثال ما اهمیت این دو مقدار یکسان است. باید مجموع این دو مقدار کمینه باشد تا بهترین عملکرد را داشته باشیم. معیار Precision به FP و معیار recall به FN اهمیت می‌دهد. مقدار این دو معیار برای تعیین عملکرد مهم است. از آنجا که اهمیت FN و FP یکسان است، پس Precision و recall هم اهمیت هستند. با توجه به این موضوع معیاری مانند F1-score که به این دو متریک اهمیت یکسان می‌دهد و نوعی میانگین هم وزن این دو معیار است، یک معیار مناسب برای بررسی عملکرد طبقه بند بر روی مجموعه داده غیر بالانس است. در Figure 32 مقدار معیارهای دیگر برای طبقه بندی آمده است. معیار دیگری که می‌توان برای بررسی عملکرد در این شرایط استفاده کرد، balanced accuracy است که در آن وزن هر کلاس در محاسبه accuracy در نظر گرفته می‌شود.

```
              precision    recall  f1-score   support

           0       0.83      0.86      0.85        35
           1       0.81      0.78      0.79        27

    accuracy                           0.82        62
   macro avg       0.82      0.82      0.82        62
weighted avg       0.82      0.82      0.82        62

f1 0.8187616263619453
precision 0.8205128205128205
recall 0.8174603174603174
accuracy 0.8225806451612904
```

Figure 32 مقدار معیارهای دیگر برای طبقه بندی

۹

نتایج بر اساس اندازه batch به مقدار ۳۲ در Figure 28 و Figure 29 و Figure 32 آمده است. در این قسمت از batch size های ۶۴ و ۱۲۸ برای یادگیری استفاده می‌کنیم و دیگر شرایط و پارامترهای یادگیری را ثابت نگه می‌داریم. زمانی که از batch با اندازه ۳۲ استفاده می‌کنیم یعنی در هر epoche، هر بار بعد از دیدن ۳۲ داده آموزش، به تغییر پارامترها و وزن‌ها می‌پردازیم. در هر epoch ابتدا داده‌ها را به صورت راندم به بسته‌ها یا batch های ۳۲ تایی تقسیم می‌کنیم. در هر epoch قرار است تمام دیتای آموزش یک بار دیده شود. بعد از دیدن تمام داده‌های یک batch، ما به بروز کردن پارامتر ها در الگوریتم بهینه سازی می‌پردازیم. در اینجا یکبار برای اندازه‌ی batch به مقدار ۳۲، یک بار ۶۴ و یک بار ۱۲۸ به یادگیری

بر روی داده آموزش می‌پردازیم و عملکرد را مقایسه می‌کنیم. هرچه اندازه batch کوچکتر باشد، میزان محاسبات برای محاسبه loss در هر بار به روز کردن وزن‌ها کمتر می‌شود و به روز رسانی سریعتر می‌شود. ولی با اندازه کوچک ما پراکنده تر به سمت مینیمم محلی حرکت می‌کنیم چون داده‌های دیده شده راندم تر و با شباهت کمتری نسبت به batch بعدی می‌توانند باشند. این در حالی است که هر چه اندازه batch بیشتر باشد، ما مستقیم تر به سمت مینیمم محلی حرکت می‌کنیم و پراکندگی حرکت کمتر است. با توجه به این که اندازه کمتر پراکندگی و پرش در اطراف مسیر به سمت مینیم محلی ایجاد می‌کند، در این شرایط احتمال فرار از یک مینیمم غیر مطلوب بیشتر است. به همین دلیل است که می‌گویند با افزایش اندازه batch باید learning rate را هم به طور متناسب بیشتر کنیم تا افزایش اندازه‌ی گام توانایی فرار را بیشتر کنیم. نحوه یادگیری با توجه به اندازه batch در Figure 33 آمده است.

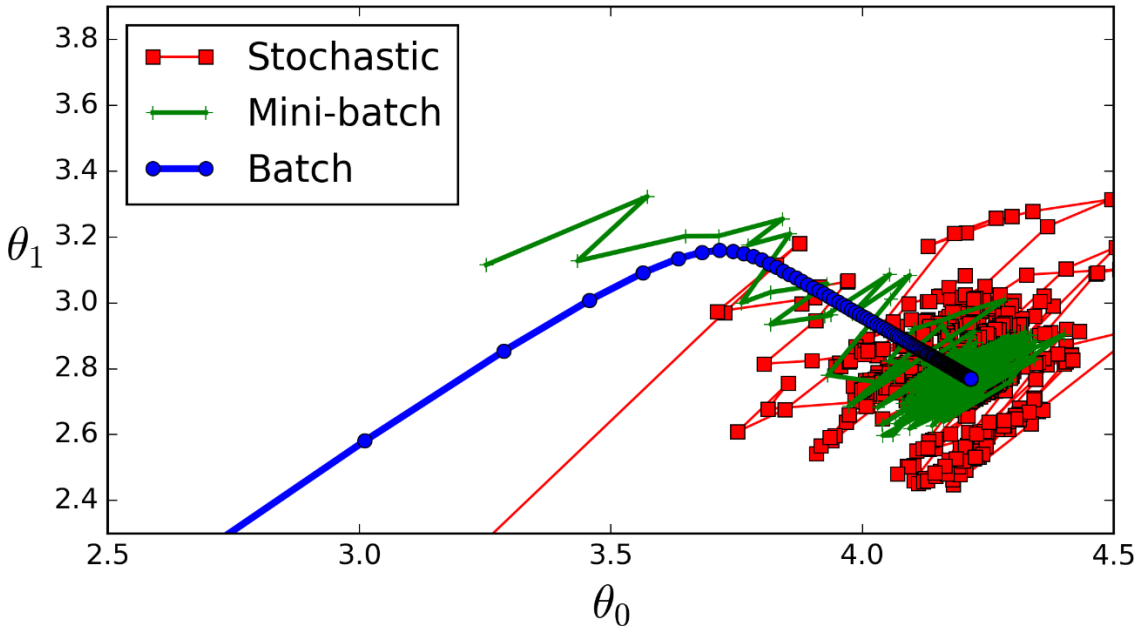

Figure 33 نحوه یادگیری با اندازه متفاوت **batch**

نمودار تغیرات خطا و دفت و مقدار پارامترهای عملکرد طبقه بندی برای اندازه batch برابر ۶۴ در Figure 34 و Figure 35 و Figure 36 آمده است. نمودار تغیرات خطا و دفت و مقدار پارامترهای عملکرد طبقه بندی برای اندازه batch برابر ۱۲۸ در Figure 37 و Figure 38 و Figure 39 آمده است.

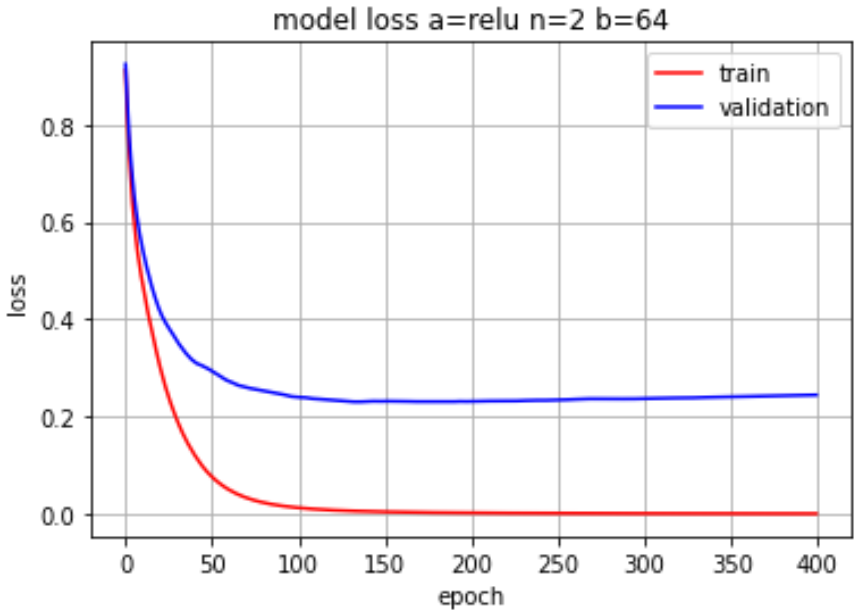

Figure 34 تعییرات خطا با اندازه **batch** برابر ۶۴

```
Test Loss 0.5176635980606079
Test Accuracy 0.8387096524238586
              precision    recall  f1-score   support

           0       0.84      0.89      0.86        35
           1       0.84      0.78      0.81        27

    accuracy                           0.84        62
   macro avg       0.84      0.83      0.83        62
weighted avg       0.84      0.84      0.84        62

f1 0.8344017094017094
precision 0.8389189189189189
recall 0.8317460317460317
accuracy 0.8387096774193549
```

Figure 35 مقادیر متریک های بررسی عملکرد طبقه بندی **b=64**

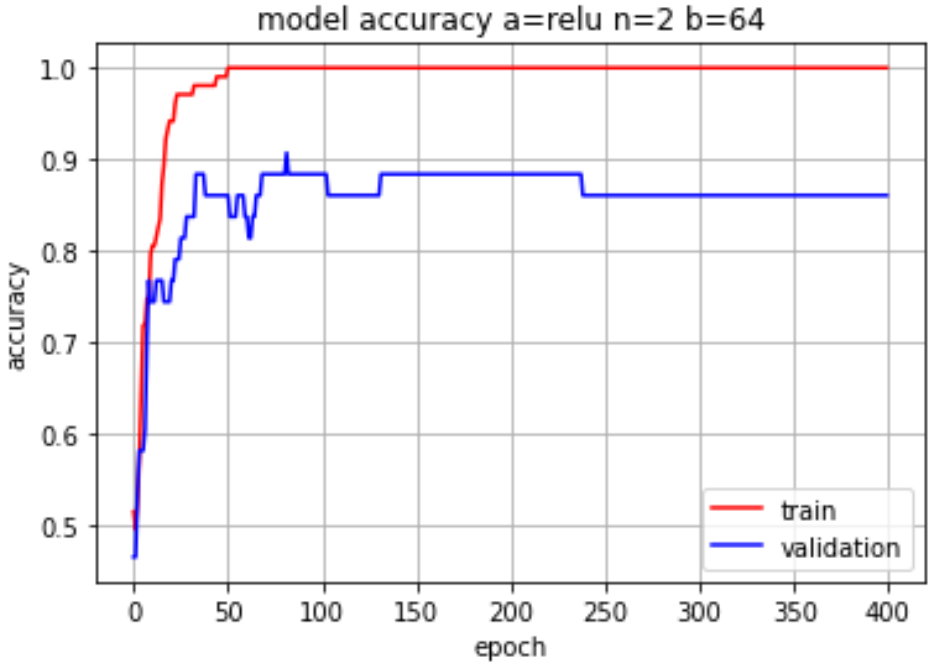

Figure 36 تعییرات دقت با اندازه **batch** برابر ۶۴

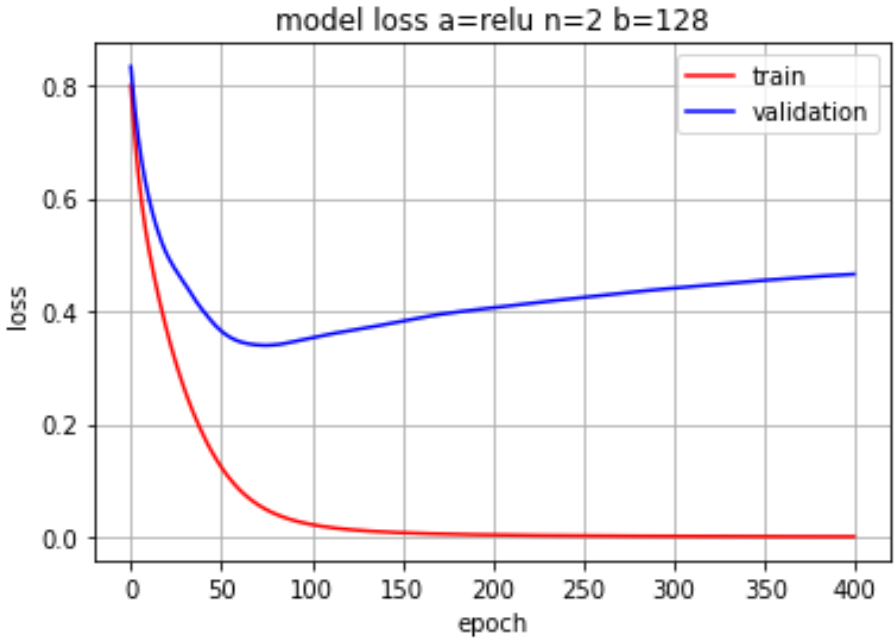

Figure 37 تعییرات خطا با اندازه **batch** برابر ۱۲۸

```
Test Loss 0.39912593364715576
Test Accuracy 0.8225806355476379
              precision    recall  f1-score   support

           0       0.83      0.86      0.85        35
           1       0.81      0.78      0.79        27

    accuracy                           0.82        62
   macro avg       0.82      0.82      0.82        62
weighted avg       0.82      0.82      0.82        62

f1 0.8187616263619453
precision 0.8205128205128205
recall 0.8174603174603174
accuracy 0.8225806451612904
```

Figure 38 مقادیر متریک های بررسی عملکرد طبقه بندی **b=128**

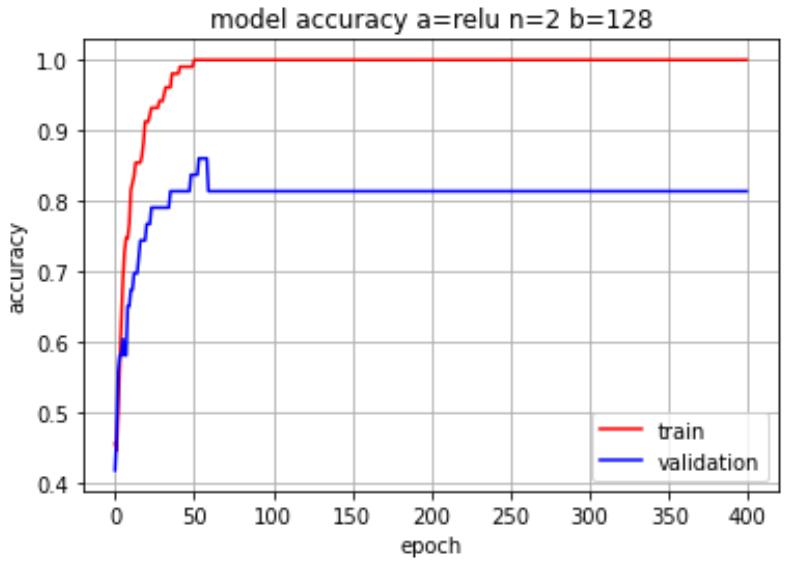

Figure 39 تغییرات دقت با اندازه **batch** برابر ۱۲۸

در حالت ۱۲۸ می‌بینیم که در نمودار Figure 39 میزان بهترین دقت بر روی داده ارزیابی از حالت‌های دیگر اندازه batch کمتر است. همچنین کمترین loss بر روی داده ارزیابی در این حالت از دیگر حالات بیشتر است. این به معنی تعمیم پایین تر در این حالت از دیگر حالات است. این در حالی است که یادگیری به خوبی بر روی داده آموزش انجام شده است. دلیل این عدم جنرالیزیشن گیر کردن در مینیم محلی نامطلوب است که به خاطر وجود نداشتن پرش در طی یادگیری و به خاظر اندازه‌ی بالای batch size است. در حالت ۳۲ میزان کمینه loss بر روی داده ارزیابی ازحالت ۶۴ بیشتر است. میزان متریک‌ها برای طبقه

بندی داده تست در حالت ۳۲ و ۱۲۸ از حالت ۶۴ کمتر است. این یعنی جنرالیزیشن در حالت batch size=32 کمتر از اندازه‌ی ۶۴ است. به همین دلیل جنرالیزشن بالا در حالت ۶۴ و بهترین مقادیر متریک ها در حالت ۶۴، اندازه batch ۶۴ را به عنوان مقدار بهینه انتخاب می‌کنیم. جهش داشتن همیشه ما را از مینیمم محلی مطلوب نجات نمی‌دهد و ممکن است به دلیل جهش‌های بزرگ با اندازهی نرخ یادگیری نسبتا بزرگ به سزعت در مینیمم محلی نامطلوب بیافتیم. این باعث توقف یادگیری می‌شود و باعث عملکرد بد روی داده دیده نشده می‌شود. در حالت ۳۲ این موضوع را مشاهده می‌کنیم که باعث عملکرد ضعیف نسبت به ۶۴ شده است.

ح

به هر بار دیده شدن تمام داده‌ها در فرآیند بهینه سازی برای یادگیری پارامترها epoch گفته می‌شود. هر بار انجام به روز رسانی پارامترها در فرآیند یادگیری به ازای دیدن تعدادی از داده‌ها(batch)، iteration گفته می‌شود. اندازه batch از یک تا تعداد کل داده‌های آموزش می‌تواند باشد. در حالت mini-bach به اندازه تعداد batch ها، iteration در هر epoch انجام می‌دهیم. در بخش‌های قبل ۴۰۰ تا epoch انجام می‌دادیم. این مقدار انتخابی است و باید به اندازه‌ای انتخاب شود که یادگیری به بهترین شکل انجام شود. در حالت اندازه batch به ترتیب برابر 32، 64 و 128 با توجه به تعداد کل داده‌های آموزش که ۱۰۴ تا است، به ترتیب در هر epoch باید 4 و 2 و 1 بار iteration انجام دهیم. اگر تعداد epoch از مقدار بهینه بیشتر باشد، overfit رخ می‌دهد و قدرت تعمیم پایین می‌آید. چون در این شرایط داریم داده‌های آموزش را به خاطر می‌سپاریم. برای تعیین تعداد بهینه epoch از مقدار loss بر روی داده ارزیابی استفاده می‌کنیم. تعدادی epoch مانند ۴۰۰ تا آموزش انجام می‌دهیم و تغییرات loss در epoch ها بر روی داده ارزیابی را بررسی می‌کنیم. تعداد epoch بهینه شماره epoch ای است که بعد از آن دیگر loss روی ارزیابی کمتر نمی‌شود و بعد آن loss بیشتر است. شماره epoch با کمترین loss روی داده ارزیابی بهینه است.

ط

برای مقایسه توابع فعالساز باید دیگر شرایط و پارامترهای یادگیری ثابت باشد. از شبکه توصیف شده در بخش الف با بهترین اندازه batch یعنی ۶۴ استفاده می‌کنیم. نتایج برای تابع فعالساز relu در Figure 34 و Figure35 Figure 29 و Figure 36 آمده است. نتایج برای تابع فعالساز tanh در Figure 44 و Figure

45 و Figure 47 آمده است. نتایج برای تابع فعالساز sigmoid درFigure 40 و Figure 41 و Figure 43 آمده است.

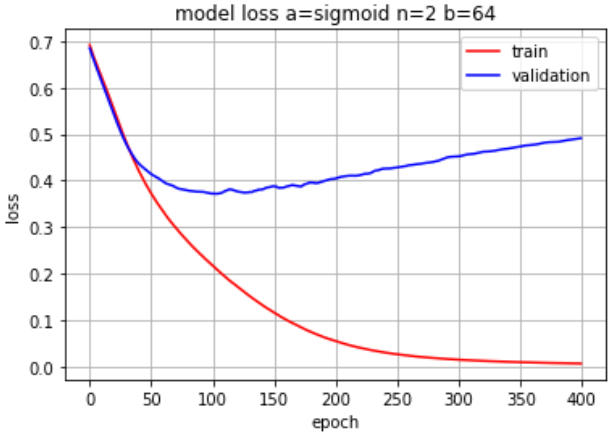

Figure 40 تغییرات خطا برای تابع فعالساز **sigmoid**

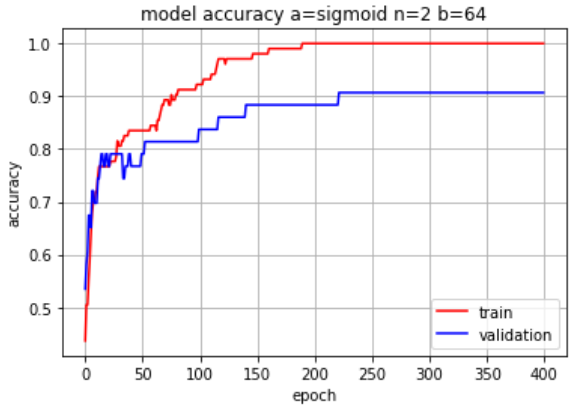

Figure 41 تغییرات دقت برای تابع فعالساز **sigmoid**

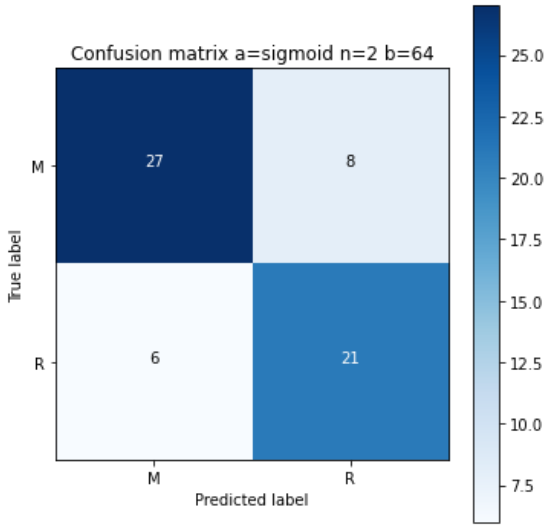

Figure 42 ماتریس درهم ریختگی با تابع فعالساز **sigmoid**

```
Test Loss 0.5872059464454651
Test Accuracy 0.774193525314331
              precision    recall  f1-score   support

           0       0.82      0.77      0.79        35
           1       0.72      0.78      0.75        27

    accuracy                           0.77        62
   macro avg       0.77      0.77      0.77        62
weighted avg       0.78      0.77      0.77        62

f1 0.7720588235294118
precision 0.7711598746081505
recall 0.7746031746031746
accuracy 0.7741935483870968
```

Figure 43 مقدار متریک های طبقه بندی تابع فعالساز **sigmoid**

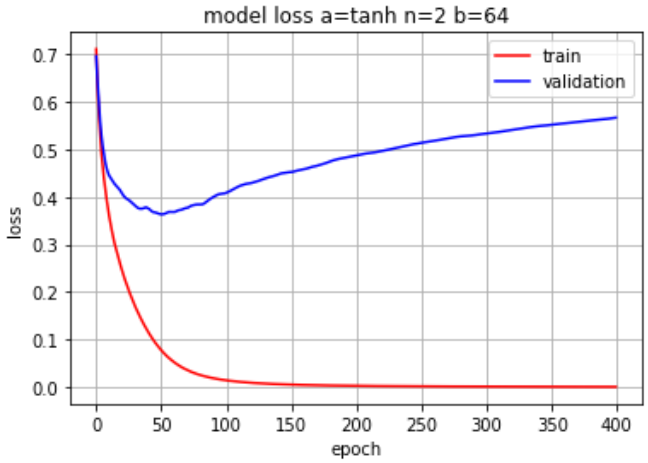

Figure 44 تغییرات خطا برای تابع فعالساز **tanh**

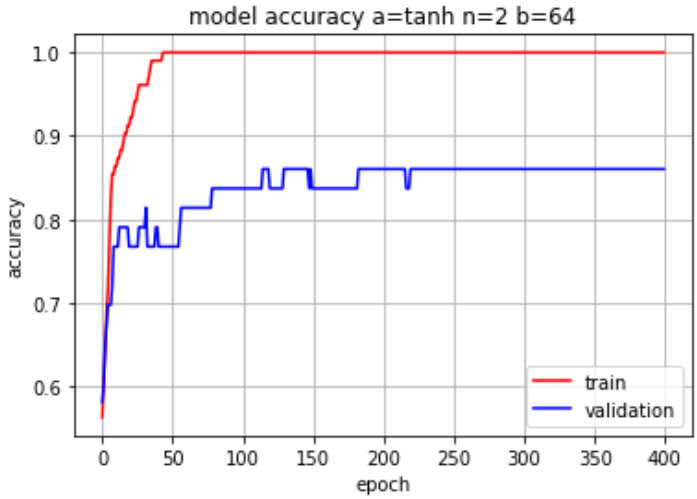

Figure 45 تغییرات دقت برای تابع فعالساز **tanh**

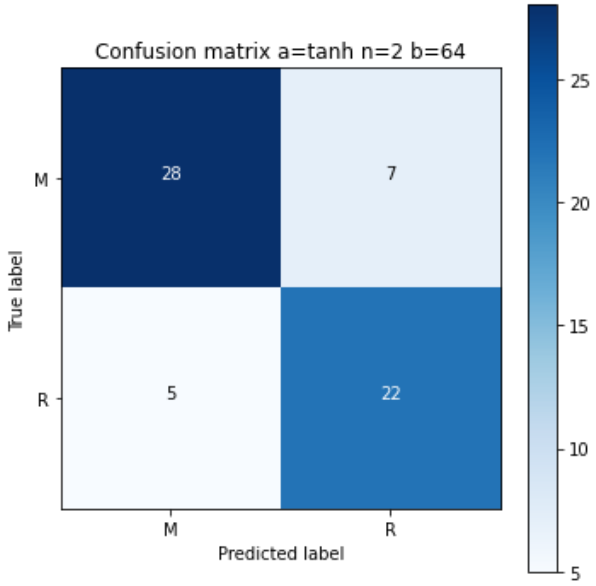

Figure 46 ماتریس درهم ریختگی تابع فعالساز **tanh**

```
Test Loss 0.5215604305267334
Test Accuracy 0.8064516186714172
              precision    recall  f1-score   support

           0       0.85      0.80      0.82        35
           1       0.76      0.81      0.79        27

    accuracy                           0.81        62
   macro avg       0.80      0.81      0.80        62
weighted avg       0.81      0.81      0.81        62

f1 0.8046218487394958
precision 0.8035527690700104
recall 0.8074074074074074
accuracy 0.8064516129032258
```

Figure 47 مقدار متریک‌های طبقه بندی تابع فعالساز **tanh**

همانطور که در دیده می‌شود در حالت relu متریک‌های طبقه بندی داده تست از تمام دیگر حالات مقدار بیشتری دارند. به همین دلیل relu بهترین تابع فعالساز است. توابع sigmoid و tanh هر دو مقدار های کوچکی را خروجی می‌دهند. خروجی sigmoid همواره بین صفر و یک است. این باعث می‌شود انفجار رخ ندهد و overfit کمتر رخ دهد. گرادیان برای این تابع کوچک است و همیشه بین صفر و یک است. این باعث می‌شود ضرب گرادیان‌ها کوچکتر شود. اگر عمق شبگه بالا باشد vanishing گرادیان رخ می‌دهد و وزن‌ها تغییر خیلی کم دارند یا اصلا ندارند. این موضوع سرعت یادگیری با کم می‌کند. (به دلیل مقدار بسیار کم نمی تواند تغییر قابل توجهی ایجاد کند). شبکه از یادگیری بیشتر امتناع می کند یا به شدت

کند است (بستگی به مورد استفاده و تا زمانی که شیب یا محاسبات توسط محدودیت های مقدار float ضربه بخورد). مزیت این تابع این است که غیر خطی است و می توان شبکه عمیق ساخت. همچنین مشتق پذیر است و تغییرات مشتق آن نرم است.

تابع tanh مانند sigmoid است چون می‌توان آن را انتقال یافته آن دانست. تنها تفاوت آن با sigmoid بازه خروجی آن است که بین منفی یک و یک است.

در نگاه اول به نظر می رسد که relu همان مشکلات عملکرد خطی را داشته باشد ، زیرا در محور مثبت خطی است. اول از همه ، ReLu ماهیتی غیرخطی دارد. و ترکیبات ReLu نیز غیر خطی هستند. (هر تابعی را می توان با ترکیب ReLu تقریبی داد). عالی است ، بنابراین این بدان معنی است که می توانیم لایه ها را روی هم قرار دهیم. دامنه آن از صفر تا بی نهایت است. این بی نهایت شدن خروجی باعث انفجار می‌شود و این می‌تواند باعث شود overfit بیشتر رخ دهد. نکته دیگری که می خواهم در اینجا بحث کنم، پراکنده بودن فعال سازی است. یک شبکه عصبی بزرگ را با تعداد زیادی نورون تصور کنید. استفاده از sigmoid یا tanh باعث می شود که تقریباً همه سلولهای عصبی به روش آنالوگ فعال شود. این بدان معناست که تقریباً همه فعال سازی ها برای توصیف خروجی یک شبکه پردازش می شوند. به عبارت دیگر فعال سازی متراکم است. این هزینه بر است. در حالت ایده آل می خواهیم چند نورون در شبکه فعال نشود و در نتیجه فعالیت ها را کم و کارآمد کند. ReLu این مزیت را به ما می دهد. شبکه ای را با وزن اولیه تصادفی تصور کنید (یا نرمال شده است) و تقریباً 50٪ شبکه به دلیل مشخصه 0 ReLu فعالیت می‌کند (خروجی 0 برای مقادیر منفی x). این بدان معناست که تعدادی نورون در حال فعالیت و شبکه سبک تر است. این فعالیت نکردن تعدادی نرون به افزایش قدرت تعمیم کمک می‌کند. این در مینیم loss بر روی داده ارزیابی دیده می‌شود. در حالت relu کمینه loss بر روی داده ارزیابی از بقیه حالات کمتر است. این موضوع در مقایسه Figure 34 و Figure 40 و Figure 44 مشخص می‌شود. این موضوع را در مقدار متریکهای طبقه بندی روی داده تست هم می‌بینیم. در ReLu. به دلیل وجود خط افقی در X منفی، گرادیان می تواند به سمت 0 حرکت کند. برای فعال سازی در آن منطقه از ReLu ، گرادیان 0 خواهد بود که به همین دلیل وزن ها تنظیم نمی شود. این بدان معناست که آن دسته از نورون ها که به آن حالت می روند دیگر به تغییرات خطا و ورودی پاسخ نمی دهند (به این دلیل که گرادیان 0 است ، هیچ چیز تغییر نمی کند). به این مسئله مرگ ReLu گفته می شود. این مشکل می تواند باعث از بین رفتن چندین نورون و انفعال کل شبکه شود، زیرا بخش مهمی از شبکه منفعل است.

ی

بله. برای بررسی این موضوع تعداد لایه‌های مخفی از ۲ تا ۴ را بررسی می‌کنیم. برای دیگر پارامترهای یادگیری نیز می‌توان تمام حالاتی که تا کنون بررسی کردیم را در نظر بگیریم. تمام حالات را بررسی کردیم و برای هر حالت نمودار تغییرات خطا و دقت و معیارهای ارزیابی بر روی داده تست بدست آمده است. برای برای اندازه batch ۶۴ که بهترین مقدار بدست آمده است و تابع فعالساز relu حالات مختلف تعداد لایه را بررسی می‌کنیم. با توجه به قسمت‌های قبل برای بررسی اثر هر تعداد لایه مخفی کافی است مقادیر متریک‌های ارزیابی را برای batch برابر ۶۴ و تابع فعالساز relu بررسی کنیم. برای حالت سه و چهار لایه مخفی لایه سوم ۱۲ نرون و در حالت چهار لایه، لایه چهارم ۶ نرون دارد.

```
Test Loss 0.393806517124176
Test Accuracy 0.8548387289047241
              precision    recall  f1-score   support

           0       0.84      0.91      0.88        35
           1       0.88      0.78      0.82        27

    accuracy                           0.85        62
   macro avg       0.86      0.85      0.85        62
weighted avg       0.86      0.85      0.85        62

f1 0.8501208702659147
precision 0.8585526315789473
recall 0.846031746031746
accuracy 0.8548387096774194
```

Figure 48 دو لایه مخفی

```
Test Loss 0.44037824869155884
Test Accuracy 0.8709677457809448
              precision    recall  f1-score   support

           0       0.89      0.89      0.89        35
           1       0.85      0.85      0.85        27

    accuracy                           0.87        62
   macro avg       0.87      0.87      0.87        62
weighted avg       0.87      0.87      0.87        62

f1 0.8687830687830688
precision 0.8687830687830688
recall 0.8687830687830688
accuracy 0.8709677419354839
```

Figure 49 سه لایه مخفی

```
Test Loss 0.5142185091972351
Test Accuracy 0.774193525314331
              precision    recall  f1-score   support

           0       0.77      0.86      0.81        35
           1       0.78      0.67      0.72        27

    accuracy                           0.77        62
   macro avg       0.78      0.76      0.77        62
weighted avg       0.78      0.77      0.77        62

f1 0.7654054054054054
precision 0.7759197324414716
recall 0.7619047619047619
accuracy 0.7741935483870968
```

Figure 50 ۴ لایه مخفی

در تمام حالات مورد نظر پارامترهای مورد بررسی، حالت batch size=64 و تابع فعالساز relu و سه لایه مخفی بهترین عملکرد را بر روی داده تست دارد. عملکرد دیگر حالات در فایل NNDL_HW2_Q2.ipynb قابل مشاهده است که به دلیل زیاد بودن از آوردن آن خودداری کردم. در Figure 48 و Figure 49 و Figure 50 متریک‌های ارزیابی برای سه حالت مختلف مورد نظر برای تعداد لایه آمده است. در این سه حالت دیگر پارامترهای شبکه یکسان است. همانطور که دیده می‌شود تمام متریکها در حالت سه لایه از دو لایه لالاتر است. این به معنی عملکرد بهتر و تعمیم بهتر روی داده تست است. متریک f1 به اندازه 1.6 درصد بهتر شده است. در حالت ۴ لایه مخفی تمام متزیک ها به مقدار زیاد کمتر از حالت دو لایه است. F1 ۹ درصد کمتر است. با افزودن لایه تعداد پارامترهای مدل را بالا می‌بریم که این موضوع کمک می‌کند مدل بتواند الگوهای پیچیده تری را مدل کند. در حالت ۳ لایه افزایش پارامتر باعث شده مدل به پیچدگی مسئله نزدیک شود و به همین دلیل یلدگیری بهتر شده است و توانستیم روی تست بهتر عمل کنیم. چون توانستیم الگوی واقعی بین دو دسته را بهتر یاد بگیریم. در حالت ۴ لایه پارامترها بیش از حد زیاد شده و ما دچار overfit شدیم. دلیل آن این است که توانایی مدل برای مدل سازی از الگوی اصلی میان داده‌ها بیشتر شده و مدل به حافظه سپردن داده‌ها پرداخته است. افزایش لایه و پارامتر و پیچدگی اگر به اندازه‌ای باشد که از پیچدگی مسئله بالاتر نباشد، کمک کننده است و می‌تواند قدرت تعمیم را افزایش دهد. اگر پارامترها بیش از حد زیاد باشد به دلیل ثابت بودن تعداد داده آموزش overfit رخ می‌دهد. در هر لایه مخفی ما نمایش جدید از فضای ورودی داریم. این نمایش در هر لایه بر اساس نمایش لایه قبل ساخته می‌شود. در هر لایه یک سطح از خلاشه سازی انجام می‌شود. لفزایش لایه‌ها می‌تواند ما را به فضای ویژگی

خلاصه شده‌ای برساند که دو کلاس در واقعیت به راحتی جدایی پذیر باشند. این مورد باعث می‌شود که مدل قدرت تعمیم بالایی داشته باشد.

ک

بهترین شبکه شامل سه لایه مخفی به ترتیب با نرون‌های 48 و 24 و 12 است. اندازه batch برابر 64 و تابع فعالساز relu است. ساختار بهترین شبکه در Figure 51 آمده است.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_42 (Dense)             (None, 48)                2928
_________________________________________________________________
dense_43 (Dense)             (None, 24)                1176
_________________________________________________________________
dense_44 (Dense)             (None, 12)                300
_________________________________________________________________
dense_45 (Dense)             (None, 2)                 26
=================================================================
Total params: 4,430
Trainable params: 4,430
Non-trainable params: 0
```

Figure 51 ساختار بهترین شبکه

بله می‌توان شبکه را بهبود داد. یکی از راه‌های ممکن بررسی تعداد نرون مناسب برای هر لایه در کنار تعداد لایه‌ها است. یک راه دیگر که آنرا انجام دادیم و نتایج را گزارش می‌کنیم، استفاده از dropout در شبکه است. در این روش، در حین تمرین درصدی از سلولهای عصبی روی یک لایه خاص غیرفعال می شود. این باعث می‌شود که لایه خود را مجبور به یادگیری یک مفهوم با نورون های مختلف می کند. این شکل یادگیری قدرت تعمیم را بهبود می‌بخشید. برای بهبود شبکه، بعد از ورودی از dropout استفاده می‌کنیم. این باعث می‌شود که هر بار با بخشی از ویژگی‌ها به یادگیر جنس سیلندر بپردازیم. این موضوع باعث می‌شود که اگر نویزی در یک ویژگی باشد، از یادگیری آن خود داری کنیم و این نویز تاثیری در مدل سازی ما نداشته باشد. نکته اینجا این است که ویژگی‌ها طوری است که اگر تعداد کمی زا حذف کنیم باز می‌توان ویژگی‌های اصلی تعیین کننده کلاس را از بقیه بدست آورد. این موضوع باعث می‌شود ما الگوی اصلی موجود بین دو دسته را یاد بگیریم و تعمیم بهتری داشته باشیم. در Figure 52 ساختار مورد نظر آمده است.

```
model = Sequential()
model.add(Dropout(0.2, input_shape=(60, )))
model.add(Dense(48, activation='relu'))
model.add(Dense(24, activation='relu'))
model.add(Dense(12, activation='relu'))
model.add(Dense(2, activation='sigmoid'))
```

Figure 52 ساختار بهبود یافته شبکه با **dropout**

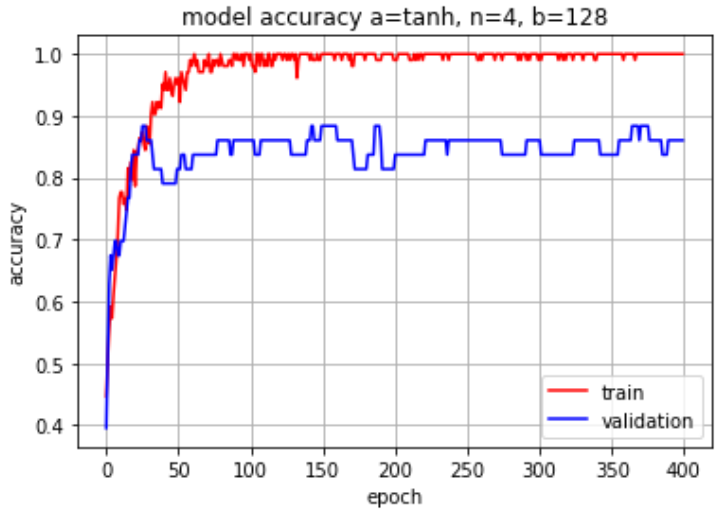

Figure 53 تغیرات دقت برای شبکه بهبود یافته

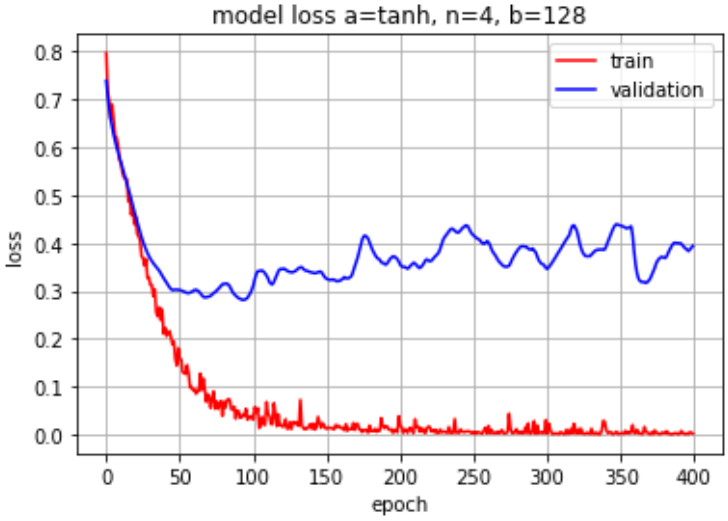

Figure 54 تغییرات خطا برای شبکه بهبود یافته

```
Test Loss 0.4165648818016052
Test Accuracy 0.8870967626571655
              precision    recall  f1-score   support

           0       0.89      0.91      0.90        35
           1       0.88      0.85      0.87        27

    accuracy                           0.89        62
   macro avg       0.89      0.88      0.88        62
weighted avg       0.89      0.89      0.89        62

f1 0.884666489503056
precision 0.8867521367521367
recall 0.8830687830687831
accuracy 0.8870967741935484
```

Figure 55 مقدار متریک‌های ارزیابی روی تست برای شبکه بهبود یافته

همانطور که در Figure 55 دیده می‌شود تمام متریک‌های ارزیابی طبقه بند از بهترین شبکه گزارش شده بهتر است. یعنی توانسته ایم شبکه را بهبود دهیم. F1 به اندازه 1.6 درصد بهبود یافته است.

ل

خیر. در epoch بیشتری این اتفاق می‌افتد. کافی است نمودار تغییرات خطا را بررسی کنیم. در تمام حالات دیگر پارامترها ثابت است و فقط تعداد لایه و نرون لایه‌ها فرق دارد. یک بار از یک لایه مخفی با ۳۰ نرون استفاده می‌کنیم. در Figure 56 نمودار تغییرات دقت برای یک لایه مخفی با ۳۰ نرون آمده است. در نزدیک ۸۰ تا epoch دقت آموزش یک شده است در حالی که دقت روی دیتای ارزیابی ۱۰ درصد کمتر است.

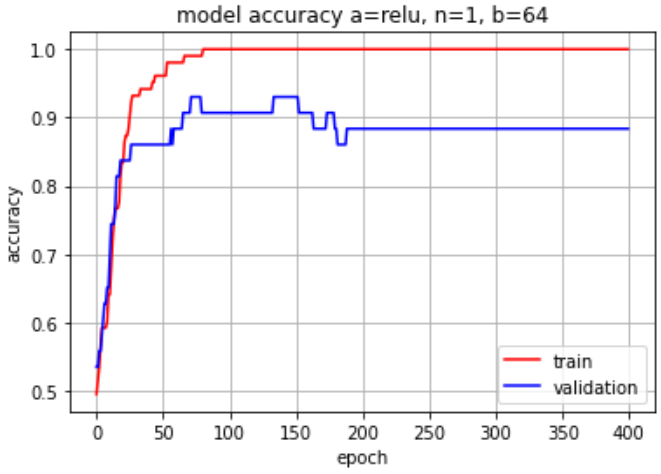

Figure 56 این نمودار تغییرات دقت برای یک لایه مخفی با ۳۰ نرون

از یک لایه مخفی با ۱۰ نرون استفاده می‌کنیم تا ببینیم در چه تعداد epoch با overfit مواجه می‌شویم. در اینجا اختلاف بزرگتر مساوی ۱۰ درصد دقت در آموزش و ارزیابی را overfit در نظر می‌گیریم. در Figure 57 نمودار تغییرات دقت برای یک لایه مخفی با ۱۰ نرون آمده است. در ۱۳۰ تا epoch ما overfit داریم. همانطور که دیدیم با کاهش تعداد نرون، تعداد epoch برای overfit افزایش یافت.

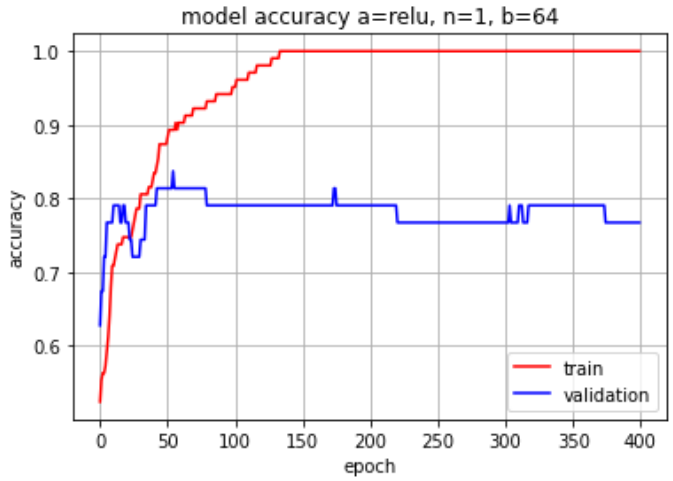

Figure 57 نمودار تغییرات دقت برای یک لایه مخفی با ۱۰ نرون

برای بررسی کاهش تعداد لایه، از شبکه بدون لایه مخفی استفاده می‌کنیم. نمودار تغییرات دقت برای شبکه بدون لایه مخفی در Figure 58 آمده است. در ۴۰۰ تا epoch ما overfit داریم. پس کاهش تعداد لایه، overfit در شماره epoch بیشتری اتفاق می‌افتد. پس کاهش نرون و لایه باعث افزایش تعداد epoch برای overfit می‌شود.

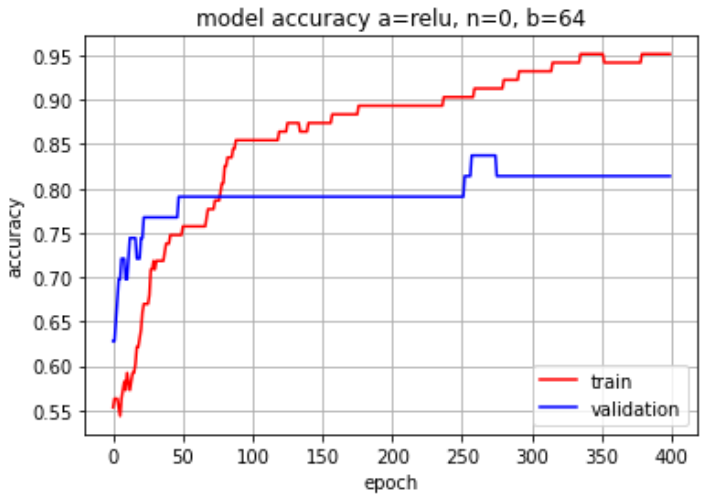

Figure 58 نمودار تغییرات دقت برای شبکه بدون لایه مخفی

دلیل این موضوع کاهش پارامترهای مدل و پیچیدگی قابل مدل سازی توسط آن است. زمانی که مدل توانایی بالایی برای مدل کردن ساختارهای پیچیده دارد، با تعدادی epoch می‌تواند مرز جداکننده دو کلاس را طوری پیداکند که تمام داده‌های کلاس صفر یک طرف آن و بقیه در طرف دیگر آن باشند. نیاز نیست که وزن‌ها با دقت بالایی محاسبه شود. ساختار پر انعطاف کمک می‌کند با تعدادی epoch و مقداری دقیق کردن وزن‌ها مرز مورد نظر را بیاید. این مرز را اگر چند جمله ای در نظر بگیریم، درجه بالایی با پارمتر زیاد دارد. به طوری که می‌تواند یک داده صفر را در وسط داده‌های یک به درستی مجزا کند. زمانی که تعداد پارامترها را از طریق کاهش نرون و لایه کم می‌کنیم، مدل نمی تواند مرزی با پیچدگی بالا مانند آنچه در بالا گفته شد بسازد. در اینجا باید مرز ساده تری را که توانایی دارد، طوری جابه جا کند تا بتواند تمام داده‌ها را بخاطر بسپارد. در این زمان overfit رخ می‌دهد. برای اینکه این اتفاق بافتد و وزن‌ها به مقدار و دقت لازم برسند باید تعداد بار زیادی تمام داده‌ها را ببینیم. این یعنی به تعداد epoch بیشتری نیاز داریم.

سوال 3 – Dimension Reduction

الف

ماتریس همبستگی داده سوال یک شامل ویژگی‌ها بعد از تمیز کردن و پیش پردازش داده‌ها و عددی سازی ویژگی‌های غیر عددی به همراه قیمت را رسم کردیم. در Figure 59 ماتریس همبستگی آمده است. ماتریس شامل مثلث پایین است که دلیل ان تقارن ماتریس است. می‌دانیم $corr(X,Y) = corr(Y,X)$. اعداد خانه‌ها از 0.25 تا 0.9- است که به ترتیب رنگها از قرمز کم رنگ به سمت آبی پر رنگ تغییر می‌کند. این موضوع در کنار تصویر ماتریس هم آمده است. به دلیل تعداد زیاد ویزگی‌ها، ماتریس در Figure 59 ریز به نمایش آمده است. برای دیدن تصویر با کیفیت ماتریس می توان به فایل corr.png مراجعه کرد. همانطور که در Figure 59 دیده می‌شود تعداد زیادی از خانه‌ها تقریبا سفید و به معنای کورولیشن صفر است. این یعنی دو ویژگی مربوط به خانه تقریبا سفید، ارتباط خطی با هم ندارند. در بخش اول ما به دنبال پیش بینی قیمت خانه بودیم. پس در این ماتریس به دنبال ستون و ردیف مربوط به قیمت می‌گردیم. همانطور که در ماتریس و Figure 59 دیده می‌شود، در ردیف مربوط به قیمت تعدادی خانه رنگ سفید دارند، یعنی تعدادی ویژگی هیچ ارتباط خطی با قیمت ندارند. دیگر خانه‌ها قرمز هستند که نشان دهنده‌ی ارتباط خطی ضعیف است. چون رنگ قرمز کورولیشن 0.25 است که از یک فاصله زیادی دارد. در ستون مربوط به قیمت بیشتر خانه‌ها سفید است. این یعنی ویژگی‌های متناظر آنها ارتباط خطی با قیمت ندارند.

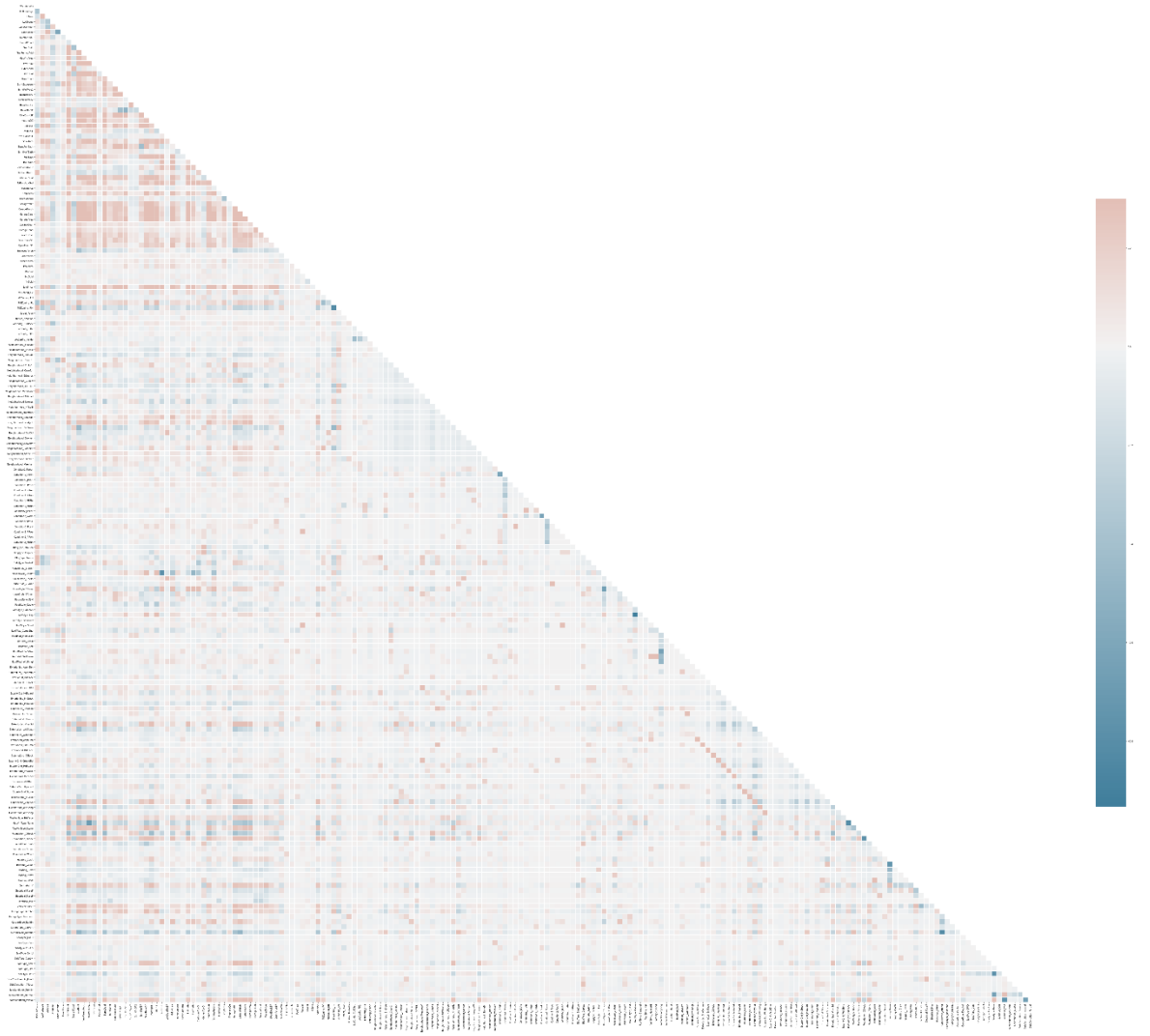

Figure 59 ماتریس همبستگی

نکته مهم این است که اندازه‌ی ستون قیمت از سطر آن بیشتر است. این یعنی تعداد زیادی از ویژگی‌ها بیش از نصف هیچ ارتباط خطی با قیمت ندارند. این یعنی اگر قیمت به صورت خطی از ویژگی‌ها تعیین شود، تعداد زیادی از ویژگی‌ها را می‌توان حذف کرد. این در صورتی است که این کم کردن این ویژگی‌ها با کورولیشن صفر هیچ اثر محسوسی بر روی کیفیت پیش بینی قیمت نمی‌گذارد. و باز می‌توان به خوبی قیمت را پیش بینی کرد.

ب

در اینجا با استفاده از linear regression می‌خواهیم اهمیت ویژگی ها را بدست آوریم. برای این کار ویژگی‌ها را standard scale می‌کنیم. مدل linear regression را بر روی تمام داده‌ها آموزش می‌دهیم.

قیمت داده‌ها ماتریس y و ویژگی‌های تمام داده‌های مجموعه داده، ماتریس x است. بعد از آموزش ضرایب ویژگی‌ها را استخراج می‌کنیم. ضریب هر ویژگی اهمیت آن را نشان می‌دهد. اهمیت ویژگی‌ها به صورت barplot در Figure 60 آمده است. عکس با کیفیت و قابل zoom در feature_importance_barplot_linear_regression.png آمده است.

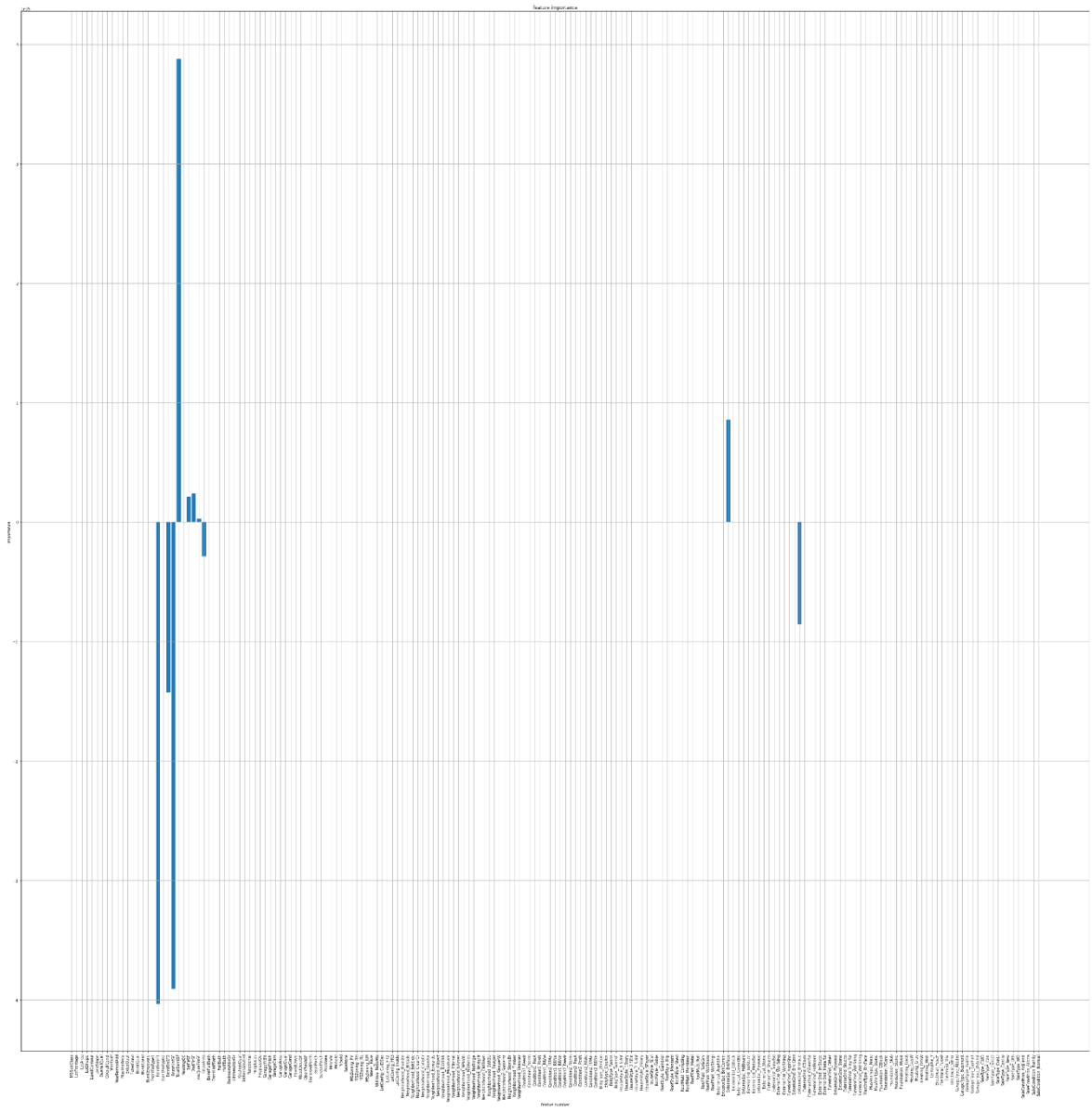

Figure 60 اهمیت ویژگی‌ها با **linear regression**

همانطور که در Figure 60 دیده می‌شود، در این نمودار بیشتر ستون‌ها مقدار بسیار نزدیک به صفر دارند. این یعنی ویژگی‌های متناظر اهمیت بسیار پایینی در تعیین قیمت خانه دارند. یعنی می‌توان این ویژگی‌ها را حذف کرد و باز پیش بینی خوبی داشت. تعداد این ویژگی‌ها که می‌توان با حذف آنها کاهش بعد داد، زیاد است.

در ادامه به نمایش اهمیت ویژگی‌ها به کمک DecisionTreeRegressor می‌پردازیم. مانند قبل این regressor را آموزش می‌دهیم و اهمیت ویژگی‌ها را از آن می‌گیریم. نمودار barplot اهمیت ویژگی‌ها با این روش در Figure 61 آمده است.

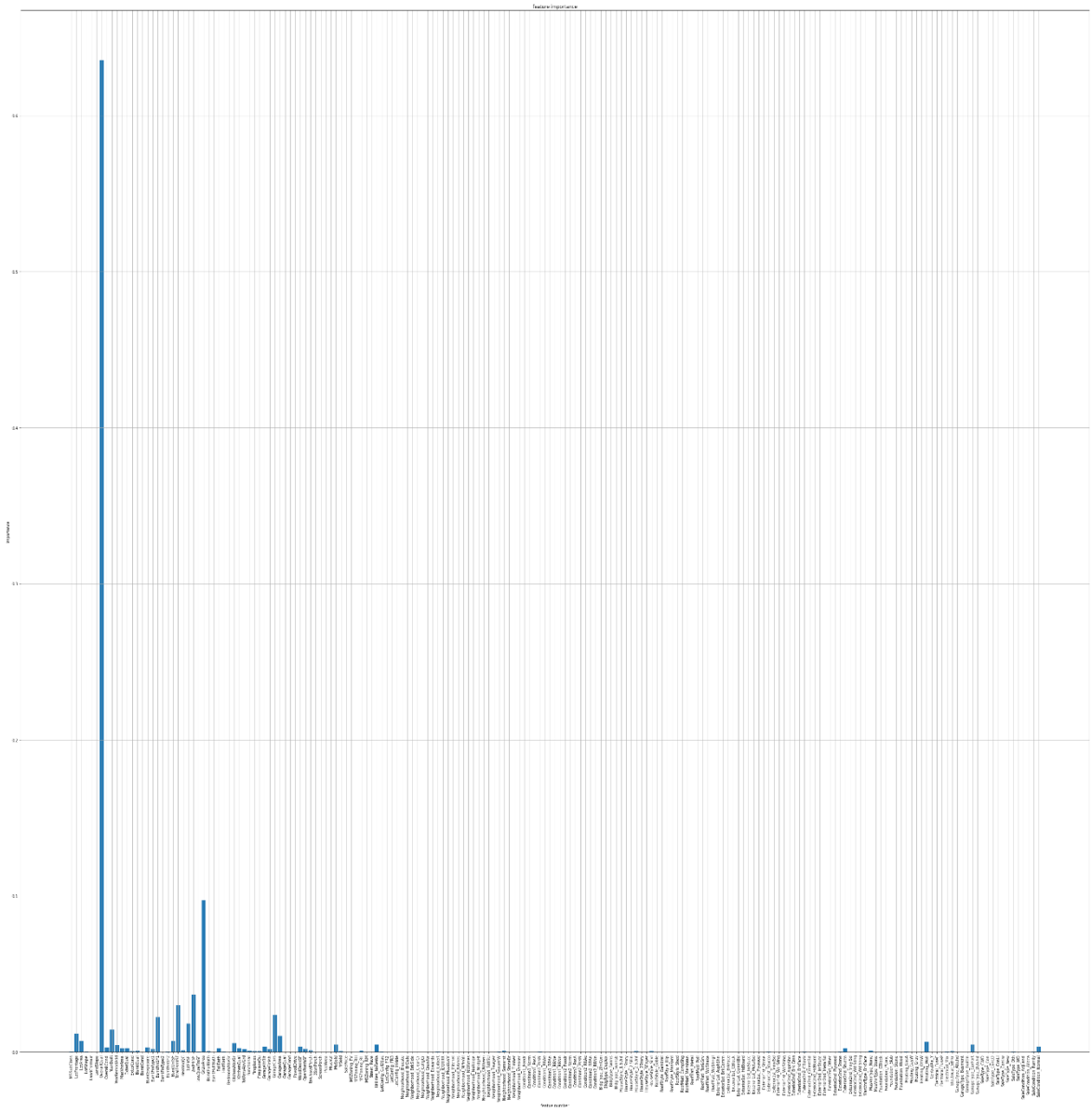

Figure 61 اهمیت ویژگی‌ها **decision_tree_regressor**

همانطور که در Figure 61 دیده می‌شود تعداد زیادی از ستون‌ها صفر است و تعدادی نیز مقدار نزدیک به صفر دارند. این یعنی تعداد زیادی از ویژگی‌ها اثری در پیش بینی قیمت خانه ندارند و می‌توان آنها را حذف کرد. با این کاهش بعد پیش بینی باز به خوبی قبل انجام می‌شود و ما اطلاعات مفیدی را از دست نمی‌دهیم.

ج

در قسمت‌های قبل دیدیم که تعداد زیادی از ویژگی‌ها اهمیت بسیارپایینی در پیش بینی قیمت دارند و می‌توان آنها را حذف کرد. در اینجا به روش backward elimination این ویژگی‌های کم اهمیت را حذف می‌کنیم. برای اینکار بار باید مانند قبل مثلا با linear regression ابتدا قیمت را بر اساس ویژگی‌ها پیش بینی کنیم. بعد از میان ویژگی‌ها کم اهمیت ترین را بیابیم و حذف کنیم. بعد کاهش یک بعد دوباره با ویژگی‌های باقی مانده همین کار را ادامه می‌دهیم. هر بار با ویژگی‌های باقی مانده رگرسیون انجام می‌دهیم و کم اهمیت ترین ویژگی را حذف می‌کنیم. برای انجام رگرسیون از Ordinary Least Squares استفاده می‌کنیم. ویژگی‌های حذف شده به ترتیب در زیر آمده است.

```
['RoofStyle_Gambrel', 'Heating_GasW', 'Electrical_FuseP', 'BsmtFinType1',
'BldgType_Duplex', 'LotConfig_Inside', 'Exterior2nd_Stucco', 'BsmtFinSF2',
'LotShape', 'EnclosedPorch', 'Exterior1st_BrkFace', 'Exterior2nd_Stone',
'SaleType_ConLw', 'Condition2_RRAn', 'BsmtUnfSF', 'GarageFinish', 'MiscVal',
'Neighborhood_MeadowV', 'Neighborhood_Somerst', 'SaleType_WD',
'Neighborhood_Sawyer', 'Neighborhood_IDOTRR', 'Heating_GasA', 'LowQualFinSF',
'Condition2_RRNn', 'RoofStyle_Gable', 'SaleCondition_Family',
'Condition1_RRNn', 'BsmtHalfBath', 'Exterior1st_CemntBd', 'Electrical_FuseF',
'LandSlope', 'Foundation_Stone', 'Heating_Grav', 'Exterior2nd_CBlock',
'Exterior1st_CBlock', 'PavedDrive', 'BsmtCond', 'Condition1_Feedr',
'Condition1_PosA', 'YrSold', 'Heating_Wall', 'SaleCondition_Alloca',
'Exterior1st_Stucco', 'Exterior2nd_Other', 'Neighborhood_SWISU',
'Exterior1st_AsphShn', 'Exterior2nd_AsphShn', 'Exterior1st_MetalSd',
'Exterior2nd_MetalSd', 'BsmtFullBath', 'Neighborhood_Blueste',
'Condition1_PosN', 'Condition1_RRAn', 'Condition2_Feedr',
'Neighborhood_Veenker', 'Condition2_Norm', 'CentralAir_Y', 'Exterior1st_Stone',
'SaleType_ConLI', 'HouseStyle_SFoyer', 'YearRemodAdd', 'Electrical_Mix',
'Exterior2nd_Wd Shng', 'GarageCond', 'HouseStyle_2Story', 'HouseStyle_2.5Unf',
'Foundation_CBlock', 'Foundation_PConc', 'LandContour', 'Condition1_RRNe',
'SaleType_Oth', 'HalfBath', '1stFlrSF', 'FireplaceQu', 'SaleCondition_Partial',
'FullBath', 'BldgType_2fmCon', 'LotConfig_FR2', 'Exterior1st_BrkComm',
'Exterior2nd_Brk Cmn', 'LotConfig_FR3', 'Neighborhood_OldTown',
'GarageType_CarPort', 'GarageType_Attchd', 'GarageType_BuiltIn', 'GarageArea',
'SaleType_CWD', 'Electrical_SBrkr', 'HeatingQC', 'Neighborhood_SawyerW',
'Neighborhood_Timber', 'Exterior1st_WdShing', 'RoofStyle_Mansard',
'SaleCondition_AdjLand', 'OpenPorchSF', 'GarageQual', 'Exterior2nd_BrkFace',
'GarageType_Detchd', 'GarageType_Basment', 'SaleType_ConLD', 'MSSubClass',
'ExterCond', 'Exterior2nd_HdBoard', 'Exterior2nd_Plywood', 'RoofStyle_Shed',
'Condition2_RRAe', 'Neighborhood_ClearCr', 'WoodDeckSF', 'Utilities_NoSeWa',
'MasVnrType_BrkFace', '3SsnPorch', 'Street_Pave', 'BsmtFinType2',
'HouseStyle_1.5Unf', 'Exterior2nd_VinylSd', 'HouseStyle_SLvl',
'Neighborhood_NAmes', 'Neighborhood_Edwards', 'Neighborhood_Gilbert',
'Neighborhood_CollgCr', 'Neighborhood_Mitchel', 'Condition1_RRAe',
'Neighborhood_NWAmes', 'Exterior2nd_Wd Sdng', 'Condition2_PosA',
'Exterior1st_ImStucc']
```

برای تشخیص ویژگی کم اهمیت در هر مرحله از مقادیر pvalue حاصل از OLS استفاده می‌کنیم. OLS به ازای هر ویژگی یک مقدار pvalue متناظر با آزمون فرض می‌دهد. فرض صفر آزمون فرض برای

هر ویژگی این است که ویژگی در تعیین خروجی یا قیمت تاثیری ندارد و ضریب آن صفر است. فرض مقابل این است که ویژگی در تعیین خروجی ماثر است. اگر مقدار pvalue متناظر ویژگی بالای 0.05 باشد یعنی فرض صفر را نمی‌توان رد کرد. هر چه مقدار pvalue بزرگتر از 0.05 بیشتر باشد، اطمینان ما بالا تر است که ویژگی بی تاثیر است. پس در هر مرحله بعد از انجام رگرسیون با ویژگی‌ها باقی مانده، ویژگی با بیشترین pvalue را اگر pvalue بیش از 0.05 داشت حذف می‌کنیم. چون ای ویژگی را با اطمینان بسیار بالا بی تاثیر در پیش بینی می‌دانیم. زمان انجام کاهش بعد و حذف ۱۲۷ ویژگی بالا به ثانیه در Figure 62 آمده است.

```
time of BE:  8.363815069198608
```

Figure 62 زمان انجام **backward elimination**

ویژگی‌های انتخاب شده بعد از این مرحله برای انچام رگرسیون با شبکه بخش یک در زیر آمده است.

```
['LotFrontage', 'LotArea', 'OverallQual', 'OverallCond', 'YearBuilt',
'MasVnrArea', 'ExterQual', 'BsmtQual', 'BsmtExposure', 'BsmtFinSF1',
'TotalBsmtSF', '2ndFlrSF', 'GrLivArea', 'BedroomAbvGr', 'KitchenAbvGr',
'KitchenQual', 'TotRmsAbvGrd', 'Functional', 'Fireplaces', 'GarageYrBlt',
'GarageCars', 'ScreenPorch', 'PoolArea', 'MoSold', 'MSZoning_FV',
'MSZoning_RH', 'MSZoning_RL', 'MSZoning_RM', 'LotConfig_CulDSac',
'Neighborhood_BrDale', 'Neighborhood_BrkSide', 'Neighborhood_Crawfor',
'Neighborhood_NPkVill', 'Neighborhood_NoRidge', 'Neighborhood_NridgHt',
'Neighborhood_StoneBr', 'Condition1_Norm', 'Condition2_PosN', 'BldgType_Twnhs',
'BldgType_TwnhsE', 'HouseStyle_1Story', 'HouseStyle_2.5Fin', 'RoofStyle_Hip',
'RoofMatl_CompShg', 'RoofMatl_Membran', 'RoofMatl_Metal', 'RoofMatl_Roll',
'RoofMatl_Tar&Grv', 'RoofMatl_WdShake', 'RoofMatl_WdShngl',
'Exterior1st_HdBoard', 'Exterior1st_Plywood', 'Exterior1st_VinylSd',
'Exterior1st_Wd Sdng', 'Exterior2nd_CmentBd', 'Exterior2nd_ImStucc',
'MasVnrType_None', 'MasVnrType_Stone', 'Foundation_Slab', 'Foundation_Wood',
'Heating_OthW', 'SaleType_Con', 'SaleType_New', 'SaleCondition_Normal']
```

بعد از این مرحله به آموزش بهترین شبکه بخش اول با تابع خطا MSE می‌پردازیم. از داده با کاهش بعد بالا استفاده می‌کنیم. زمان آموزش در Figure 63 آمده است.

```
fit time: 20.948923587799072
```

Figure 63 زمان آموزش مدل بخش یک با داده کاهش بعد یافته

نمودار تعییرات خطا در Figure 64 و پیش بینی بر اساس مقدار واقعی در Figure 65 و مقدار loss مدل بر روی داده تست در Figure 66 آمده است. همانطور که دیده می‌شود، پیش بینی‌ها مانند قبل نزدیک واقعیت است و نمودار پیش بینی بر اساس واقیت به y=x نزدیک است. نقاط به صورت متمرکز در فاصله نزدیکی از این خط قرار دارند. مقدار خطا بر روی داده تست نیز مقدار کمی بیش تر از قبل است. در

گذشته 1121563520 بود و اکنون 59861120 بیشتر است. زمان آموزش نسبت به قبل که 33.19 ثانیه بود، 12.24 ثانیه کاهش یافته است. کل زمان لازم برای آموزش و کاهش بعد 29.31 ثانیه است که از زمان یادگیری بدون کاهش بعد ۳.۸۸ ثانیه کمتر است. نمودار تغییرات خطا روندی مانند قبل دارد و میزان خطای ارزیابی و آموزش به قبل بسیار نزدیک است.

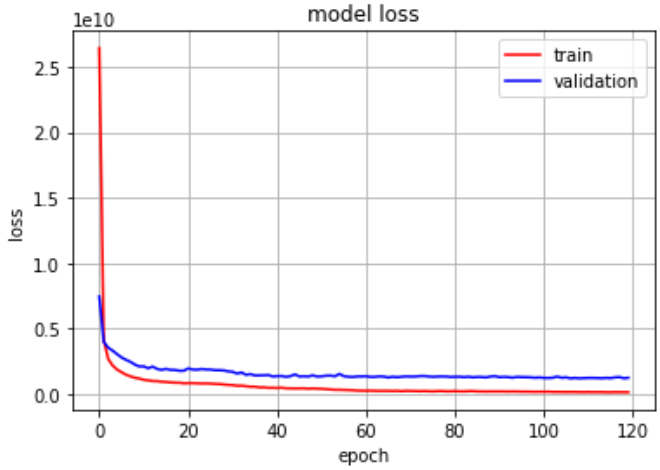

Figure 64 نمودار تغییرات خطا مدل بخش یک با کاهش بعد

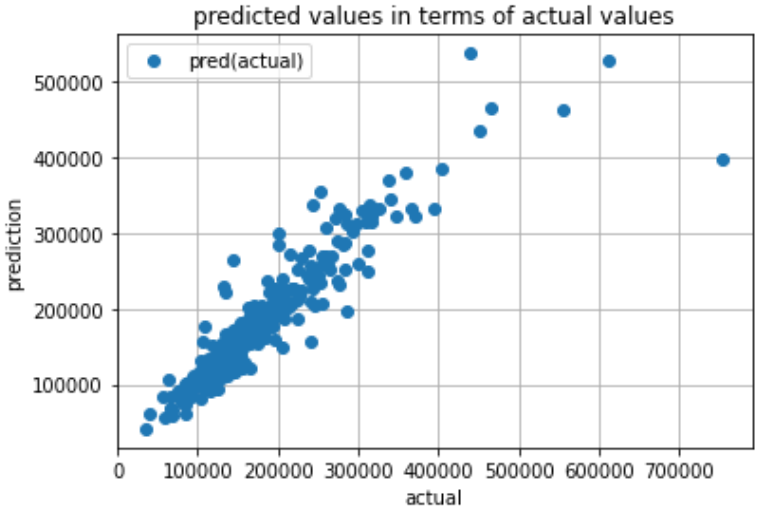

Figure 65 پیش بینی بر اساس مقدار واقعی برای مدل بخش یک با کاهش بعد

```
10/10 [==============================] - 0s 2ms/step - loss: 1032122539.6364
loss: 1181424640.0
```

Figure 66 میزان خطا بر روی داده تست با کاهش بعد

بررسی نمودار تغییرات خطا و پیش بینی بر اساس واقعیت و زمان آموزش و کاهش بعد نشان می‌دهد که با کاهش بعد انجام شده توانستیم کل زمان آموزش را شامل کاهش بعد کاهش دهیم و کیفیت پیش بینی نزدیک به قبل داشته باشیم.

د

برای یافتن اندازه‌ی ابعاد مناسب برای PCA از رسم مقدارهای ویژه متناظر هر کامپوننت استفاده می‌کنیم. از PCA با تعداد کامپوننت ۶۰ استفاده می‌کنیم و مقادیر ویژه را که نشان دهنده‌ی میزان پراکندگی پوشش داده شده است را نمایش می‌دهیم. مقادیر ویژه کامپوننت‌ها برای PCA با ۶۰ کامپوننت در Figure 67 آمده است. همانطور که دیده می‌شود بعد از ۴۰ کامپوننت میزان واریانس پوشش نزدیک صفر است. پس تعداد کامپوننت بهینه برابر ۴۰ است. در قسمت بعد نیز ابعاد را به ۴۰ بعد کاهش می‌دهیم.

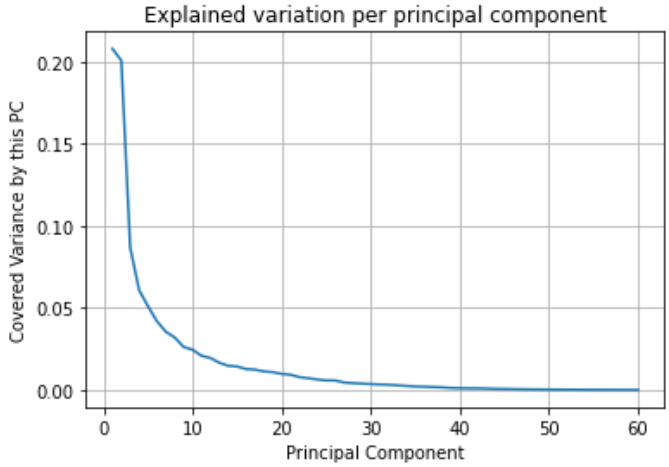

Figure 67 مقادیر ویژه کامپوننت‌ها برای **PCA** با ۶۰ کامپوننت

از PCA با ۴۰ کامپوننت برای کاهش بعد استفاده می‌کنیم. زمان لازم برای یادگیری PCA و انجام کاهش بعد به ثانیه در Figure 68 آمده است.

```
PCA time: 0.014031410217285156
```

Figure 68 زمان یادگیری **PCA** و انجام کاهش بعد

از بهترین شبکه بر اساس بهترین پارامترهای بدست آمده در بخش دو تمرین قسمت ک استفاده می‌کنیم. این شبکه را با داده‌های بخش دو آموزش می‌دهیم. زمان آموزش به ثانیه و خطا و دقت روی داده تست در Figure 69 آمده است.

```
fit time: 17.63992166519165
WARNING:tensorflow:5 out of the last 15 calls to <function
2/2 [==============================] - 0s 7ms/step - loss:
Test Loss 0.43234601616859436
Test Accuracy 0.8709677457809448
              precision    recall  f1-score   support

           0       0.86      0.91      0.89        35
           1       0.88      0.81      0.85        27

    accuracy                           0.87        62
   macro avg       0.87      0.86      0.87        62
weighted avg       0.87      0.87      0.87        62

f1 0.8675213675213675
precision 0.8724324324324324
recall 0.8645502645502645
accuracy 0.8709677419354839
```

Figure 69 دقت، خطا و زمان آموزش مدل با داده حاصل از **PCA**

۵

در این بخش برای کاهش بعد از autoencoder استفاده می‌کنیم. می‌دانیم که با ساختار feedforward و دو لایه مخفی توانایی داریم complex nonlinear correlations های موجود بین ویژگی‌ها را حذف کرد. پس ساختار شبکه autoencoder را مانند Figure 71 تعریف می‌کنیم. شبکه شامل ۵ لایه مخفی است. دو لایه مخفی برای encode و دو لایه مخفی برای decode داریم. یک لایه مخفی در وسط نمایش در ابعاد جدید را نگه می‌دارد که گلوگاه می‌نامیم. لایه ورودی ۶۰ نرونه است. لایه گلوگاه ۴۰ نرونه است. لایه مخفی اول encode ۵۴ نرون و لایه مخفی دوم encode ۴۹ نرون دارد. نرون‌های بخش decode بر اساس بخش encode تعیین می‌شوند. هر لایه مخفی از تابع فعالساز relu استفاده می‌کند. برای آموزش از بهینه ساز adam و تابع هزینه mse استفاده می‌کنیم. زمان اموزش این شبکه به ثانیه در Figure 70 آمده است. همانطور که دیده می‌شود زمان آموزش آن از PCA بیشتر است.

```
fit autoencoder time: 12.288222074508667
```

Figure 70 زمان آموزش **autoencoder**

```
n_inputs = 60
# define encoder
visible = Input(shape=(n_inputs,))
# encoder level 1
e = Dense(n_inputs*0.9)(visible)
# e = BatchNormalization()(e)
e = LeakyReLU()(e)
# encoder level 2
e = Dense(round(n_inputs*0.9*0.9)+1)(e)
# e = BatchNormalization()(e)
e = LeakyReLU()(e)
# bottleneck
n_bottleneck = 40
bottleneck = Dense(n_bottleneck)(e)
# define decoder, level 1
d = Dense(round(n_inputs*0.9*0.9)+1)(bottleneck)
# d = BatchNormalization()(d)
d = LeakyReLU()(d)
# decoder level 2
d = Dense(n_inputs*0.9)(d)
# d = BatchNormalization()(d)
d = LeakyReLU()(d)
# output layer
output = Dense(n_inputs, activation='linear')(d)
```

Figure 71 ساختار **autoencoder**

با استفاده از بخش encode شبکه بالا، داده‌ها را به فضای ۴۰ بعدی می‌بریم و کاهش بعد انجام می‌دهیم. از بهترین شبکه در بخش دوم تمرین که در قسمت قبل نیز استفاده کردیم، برای آموزش در این بخش استفاده می‌کنیم. زمان آموزش شبکه، خطا و دقت بر روی داده تست در Figure 72 آمده است.

```
fit time: 17.449257850646973
2/2 [==============================] - 0s 4ms/step - loss: 0.4792 - accuracy: 0.8515
Test Loss 0.40352731943130493
Test Accuracy 0.8709677457809448
              precision    recall  f1-score   support

           0       0.86      0.91      0.89        35
           1       0.88      0.81      0.85        27

    accuracy                           0.87        62
   macro avg       0.87      0.86      0.87        62
weighted avg       0.87      0.87      0.87        62

f1 0.8675213675213675
precision 0.8724324324324324
recall 0.8645502645502645
accuracy 0.8709677419354839
```

Figure 72 زمان آموزش و دقت و خطا بر روی داده تست برای داده‌های کاهش بعد یافته با **autoencoder**

و

زمان آموزش و دقت و خطا بر روی داده تست برای بهترین شبکه سوال دو در Figure 73 آمده است.

جدول مقایسه تاثیر روش‌های کاهش بعد در طبقه بندی در Table 1 آمده است.

```
fit time: 17.890084505081177
WARNING:tensorflow:6 out of the last 11 calls to <function M
2/2 [==============================] - 0s 5ms/step - loss: 0
Test Loss 0.30786240100860596
Test Accuracy 0.8709677457809448
              precision    recall  f1-score   support

           0       0.86      0.91      0.89        35
           1       0.88      0.81      0.85        27

    accuracy                           0.87        62
   macro avg       0.87      0.86      0.87        62
weighted avg       0.87      0.87      0.87        62

f1 0.8675213675213675
precision 0.8724324324324324
recall 0.8645502645502645
accuracy 0.8709677419354839
```

Figure 73 زمان آموزش دقت و خطا روی تست بهترین شبکه سوال دو بدون کاهش بعد

Table 1 مقایسه دقت شبکه‌های مختلف

	دقت داده تست	خطای داده تست	زمان
بهترین شبکه سوال دو	0.871	0.3079	17.89
AutoEncoder	0.871	0.4035	29.74
PCA	0.871	0.4323	17.65

همانطور که در Table 1 دیده می شود، دقت بر روی داده تست در تمام روش‌ها یکسان است. میزان خطا روی تست در حالت بدون کاهش بعد به میزان 0.1 کمتر است و این درحالی است که میزان خطا برای دو روش کاهش بعد اختلاف 0.03 و ناچیزی دارد. این موضوع نشان می‌دهد که با کاهش بعد کمی خطا افزایش می‌یابد ولی باز دقت ثابت است. یعنی با کاهش بعد در پیش بینی مانند گذشته عمل می‌کنیم. خطا در روش autoencoder کمتر است یعنی بهتر و بیشتر کورولیشن بین ویژگی‌ها را حذف کرده است. زمان روش autoencoder بسیار بیشتر از روش‌های دیگر است. این یعنی با انجام کاهش بعد در این روش

کل زمان آموزش از حالت بدون کاهش بعد بیشتر شده است ولی دقت بر روی داده تست ثابت بوده است. ما انتظار داریم وقتی کاهش بعد می‌دهیم، زمان آموزش کمتر شود و دقت و عملکرد ما در پیش بینی تغییری نکند. به همین دلیل زمان زیاد آموزش، روش autoencoder بهینه و قابل قبول نیست. در روش PCA زمان آموزش کمتر از حالت بدون کاهش بعد شده است و ما توانسته‌ایم عملکرد مشابهی در پیش بینی داشته باشیم. این عملکرد PCA مطلوب ما است. در کل PCA سریعتر از autoencoder است و هزینه محاسباتی کمتری دارد. به همین دلیل در Table 1، زمان PCA بسیار کمتر از autoencoder است. PCA یک تبدیل خطی است و فقط توانایی حذف کورولیشن‌های خطی را دارد ولی autoencoder می‌تواند کورولیشن‌های غیر خطی را هم حذف کند. چون autoencoder استفاده شده، توانایی مدل کردن تابع پیچیده غیر خطی را دارد. در اینجا می‌بینیم که در هر دو روش PCA و autoencoder میزان خطا نزدیک است. با توجه به این موضوع به نظر می‌رسد کورولیشن غیر خطی قوی بین ویژگی‌ها نداریم به همین دلیل بهبود چشم گیری صورت نگرفته است. اگر کورولیشن غیر خطی نداشته باشیم، اختلاف loss دو روش کاهش بعد می‌تواند نشان دهنده حذف بهتر کورولیشن خطی در autoencoder باشد. در روش‌های کاهش بعد، میزان خطا از حالت بدون کاهش بعد بیشتر است که این به دلیل حذف تعدادی از ویژگی‌ها در روش های کاهش بعد است. این کاهش باعث شده که مقداری از اطلاعات کم اهمیت را کنار بگذاریم.