



به نام خدا
دانشگاه تهران
دانشکده مهندسی برق و کامپیوتر



درس شبکه‌های عصبی و یادگیری عمیق

تمرین اول

نام و نام خانوادگی	فرید سیاهکلی – سعید شکوفا
شماره دانشجویی	810198418 – 810198510
تاریخ ارسال گزارش	1401.08.01

فهرست

1	پاسخ 1. شبکه عصبی Mcculloch-Pitts
1	1-1. ضرب کننده باینری دو بیتی
4	پاسخ 2 – AdaLine and MadaLine
4	2-1. AdaLine
6	2-2. MadaLine
9	پاسخ 3 – Restricted Boltzmann Machine
9	3-1. سیستم توصیه گر
10	پاسخ 4 – MLP
10	4-1. Multi-Layer Perceptron

شکل‌ها

- 1 شکل 1 شمای کلی شبکه.....1
- 2 شکل 2 جبر کلی ضرب کننده.....2
- 2 شکل 3 کد پیاده سازی2
- 3 شکل 4 تصویر خروجی به ازای جایگشت های ورودی3
- 4 شکل 5 دو دسته داده4
- 4 شکل 6 پیاده سازی بخش بهینه سازی نوروں آدالاین4
- 5 شکل 7 نمودار آموزش و خط جدا کننده دیتا برای تست ست شماره یک5
- 5 شکل 8 نمودار آموزش و خط جدا کننده دیتا برای تست ست شماره دو5
- 7 شکل 9 جداسازی دیتا با سه نوروں (خط)7
- 7 شکل 10 جداسازی دیتا با چهار نوروں (خط)7
- 7 شکل 11 جداسازی دیتا با هشت نوروں (خط)7
- 10 شکل 12 اطلاعات دیتافریم10
- 10 شکل 13 تعداد Nan برحسب ستون ها10
- 11 شکل 14 ماتریس همبستگی بر روی دیتاست11
- 11 شکل 15 نمودارهای توزیع11
- 12 شکل 16 نمودار sqft_living برحسب price12
- 12 شکل 17 تولید ستون ماه و سال از روی تاریخ12
- 12 شکل 18 جداسازی داده آموزش و تست12
- 13 شکل 19 اسکیل دیتای آموزش و تست13
- 13 شکل 20 تعریف شبکه عصبی و تابع دیتا13
- 14 شکل 21 نمودار خطا برای خطای test و validation برای 50 ایپاک14
- 15 شکل 22 نمودار پیش بینی و هدف قیمت15
- 15 شکل 23 نمودار پیش بینی و هدف قیمت بدون اسکیل15

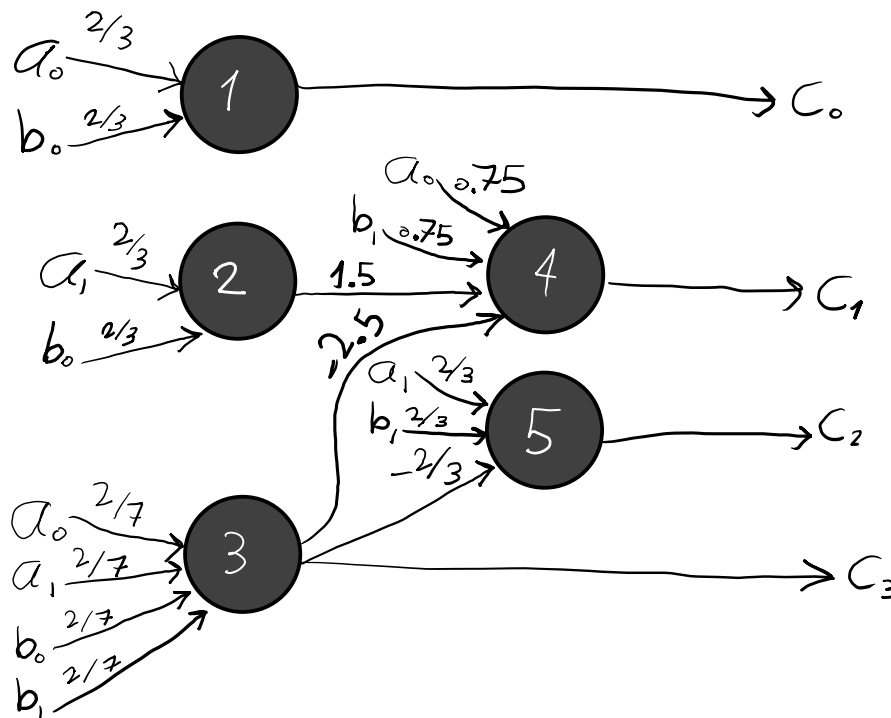
جدول‌ها

- جدول 1 دقت شبکه ها به ازای تعداد نوروں متفاوت 8
- جدول 2 آموزش شبکه به ازای دو لاس و بهینه سازی متفاوت 14
- جدول 3 پیش بینی و قیمت هدف برای 5 دیتای رندوم 16

پاسخ 1. شبکه عصبی Mcculloch-Pitts

۱-۱. ضرب کننده باینری دو بیتی

(الف)



شکل 1 شمای کلی شبکه

شبکه عصبی بالا مربوط به ضرب کننده می باشد که تمامی بایاس ها برابر صفر و تمامی threshold ها برای تمامی نورون ها برابر یک می باشد (در صورتی که خروجی نورون بیشتر از یک باشد خروجی آن یک و در صورتی که کمتر باشد خروجی آن صفر می شود).

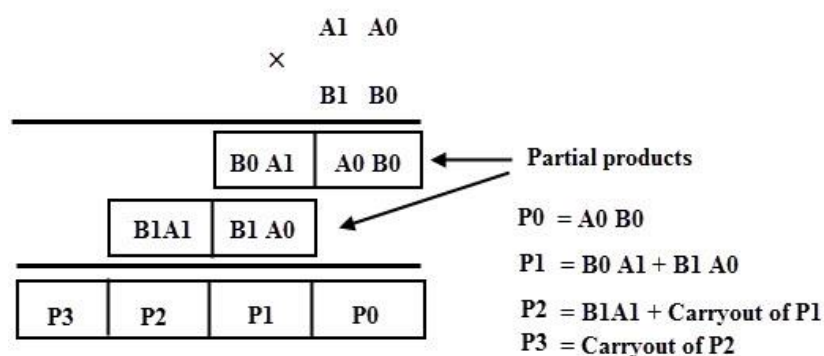
- نورون شماره یک مقدار a_0, b_0 را آند میکند که مقدار c_0 را به ما میدهد.
- نورون شماره دو مقدار a_1, b_0 را OR کرده .
- نورون شماره 3 مقادیر a_0, a_1, b_0, b_1 را AND کرده و خروجی c_3 را به ما میدهد.

از آنجا که اگر خروجی نورون شماره 3 (یا همان c_3) برابر یک باشد باید c_1, c_2 صفر باشند خروجی این نورون با ضریب منفی وارد نورون شماره 4 و 5 می شود تا از یک شدن خروجی این نورون ها جلوگیری کند.

نورون شماره 4 در واقع از OR کردن {OR کردن a0,b1 و OR کردن a1,b0} در صورتی که خروجی نورون 3 یک نباشد به دست می آید ولی اگر خروجی نورون 3 یک باشد مقدار خروجی نورون 4 برابر صفر می شود.

نورون شماره 5 از OR کردن a1,b1 در صورتی که C3 یا همان خروجی نورون شماره 3 یک نباشد به دست می آید و اگر خروجی نورون شماره 3 یک باشد مقدار آن صفر خواهد بود.

خروجی تمام نورون ها بر اساس فرمول های زیر حساب شده است:



شکل 2 جبر کلی ضرب کننده

ب) کد نوشته شده به زبان پایتون:

```

1 import numpy as np
2
3 def activation(x):
4     teta = 1
5     x[x >= 1] = 1
6     x[x < 1] = 0
7     return x
8
9 def mult(a1, a0, b1, b0):
10    x = np.array([a0, a1, b0, b1])
11    x = x.reshape(4, 1)
12    w1 = np.array([[2/3, 0, 2/3, 0],
13                  [0, 2/3, 2/3, 0],
14                  [2/7, 2/7, 2/7, 2/7]])
15    x2 = np.dot(w1, x)
16    res = activation(x2)
17    res = np.zeros([4,1]);
18    res[3] = x2[0]
19    res[2] = 0.75*a0 + 0.75*b1 + 1.5*x2[1] - 2.5*x2[2]
20    res[1] = 2/3*a1 + 2/3*b1 - 2/3*x2[2]
21    res[0] = x2[2]
22    res = activation(res)
23    return res.reshape(1,4)
24
25 print(mult(0, 0, 0, 0))
26 print(mult(0, 0, 0, 1))
27 print(mult(0, 0, 1, 1))
28 print(mult(0, 1, 0, 1))
29 print(mult(1, 0, 1, 0))
30 print(mult(1, 0, 0, 1))
31 print(mult(1, 0, 1, 1))
32 print(mult(1, 1, 1, 1))

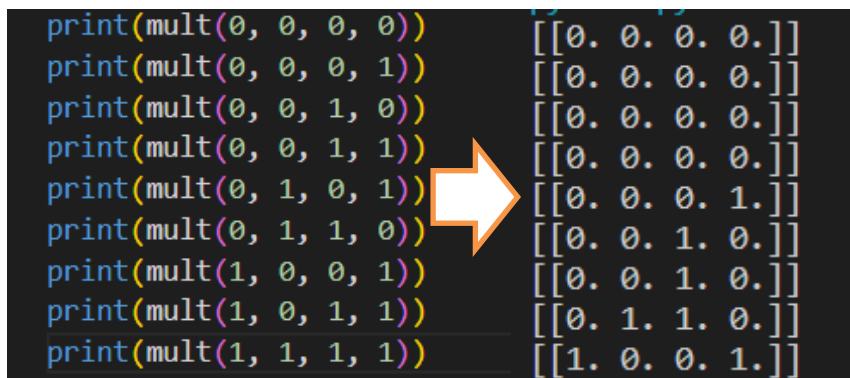
```

شکل 3 کد پیاده سازی

این کد ابتدا خروجی نورون ها 1 و 2 و 3 را با استفاده از ضرب ماتریسی به دست می آورد و در واقع ورودی ها را در ماتریس سطری (دارای 4 ستون) قرار می دهیم و سپس در ماتریس ضرایب که دارای 3 سطر و 4 ستون می باشد به صورت ماتریسی ضرب میکنیم که 3 خروجی به ما می دهد دو تا از خروجی ها حاصل C0, C3 را می دهند. و یکی از خروجی های دیگر به همراه C3 با ضرایبی که در قسمت قبل تعیین شد به همراه ورودی ها وارد دو نورون دیگر (نورون های 4 و 5) می شوند و خروجی C1, C2 را می دهند که این قسمت از کد به صورت ضرب ضرایب در مقدار های ورودی مورد نیاز برای هر خروجی نوشته شده است. لازم به ذکر است که خروجی هر نورون وارد تابع activation می شود که اگر بزرگتر از یک باشد برابر یک و اگر کوچک تر از یک باشد برابر صفر قرار گیرد.

در آخر مقدار تمامی حالت ها به شبکه عصبی داده شده است و خروجی ها به شرح زیر است:

Print(a1, a0, b1, b0) -> [res[3], res[2], res[1], res[0]]



```

print(mult(0, 0, 0, 0)) [[0. 0. 0. 0.]]
print(mult(0, 0, 0, 1)) [[0. 0. 0. 0.]]
print(mult(0, 0, 1, 0)) [[0. 0. 0. 0.]]
print(mult(0, 0, 1, 1)) [[0. 0. 0. 0.]]
print(mult(0, 1, 0, 1)) [[0. 0. 0. 1.]]
print(mult(0, 1, 1, 0)) [[0. 0. 1. 0.]]
print(mult(1, 0, 0, 1)) [[0. 0. 1. 0.]]
print(mult(1, 0, 1, 1)) [[0. 1. 1. 0.]]
print(mult(1, 1, 1, 1)) [[1. 0. 0. 1.]]

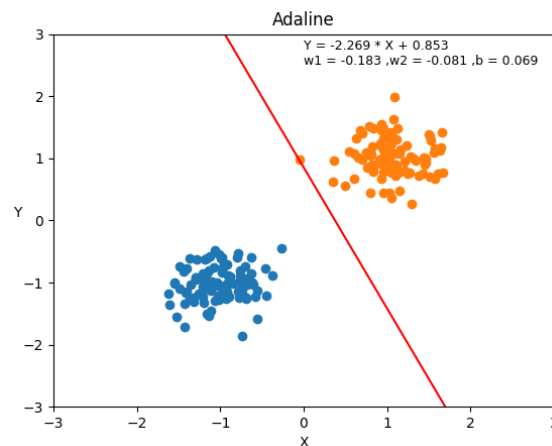
```

شکل 4 تصویر خروجی به ازای جایگشت های ورودی

همانطور که دیده می شود خروجی شبکه عصبی برای تمامی حالت ها درست است.

۲-۱. AdaLine

الف) دو دسته داده به صورت زیر می شود. (در ادامه نمودار آن کشیده شده)



شکل 5 دو دسته داده

ب) نورون AdaLine به صورت زیر است که در آن دو وزن به ورودی های x و y داده می شود. همچنین یک نورون بایاس نیز در شبکه وجود دارد. تابع هزینه به صورت زیر است.

$$error = 0.5 \times (t - net)^2$$

$$w^+ = w^- + \alpha \times error \times x_i$$

$$b^+ = b^- + \alpha \times error$$

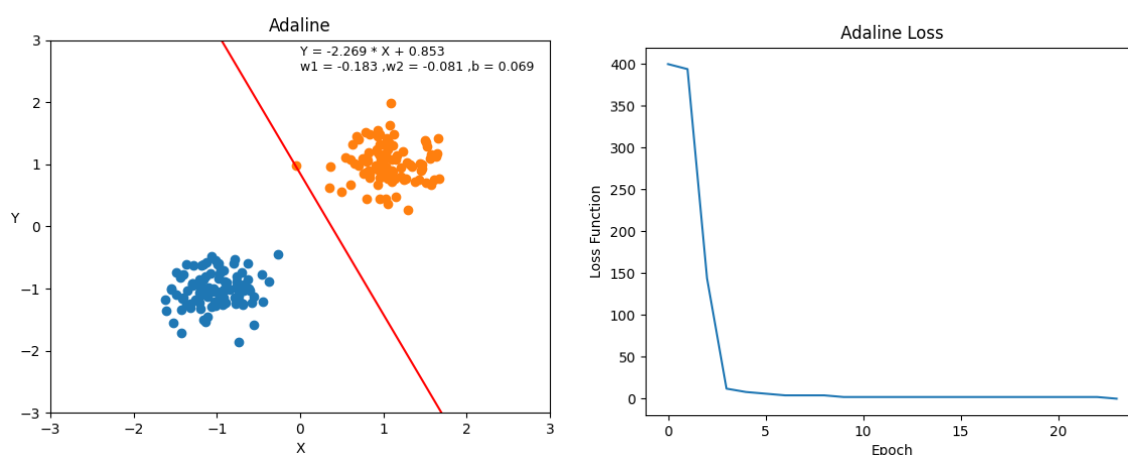
که این بخش به صورت زیر پیاده سازی نیز شده است.

```
for p in train_set:
    x1 , x2 , t = p["x1"] , p["x2"] , p["t"]
    net = w1*x1 + w2*x2 + b
    h = activation_function(net)
    w1 = w1 + lr*(t-h)* x1
    w2 = w2 + lr*(t-h)* x2
    b = b + lr*(t-h)
```

شکل 6 پیاده سازی بخش بهینه سازی نورون آدالاین

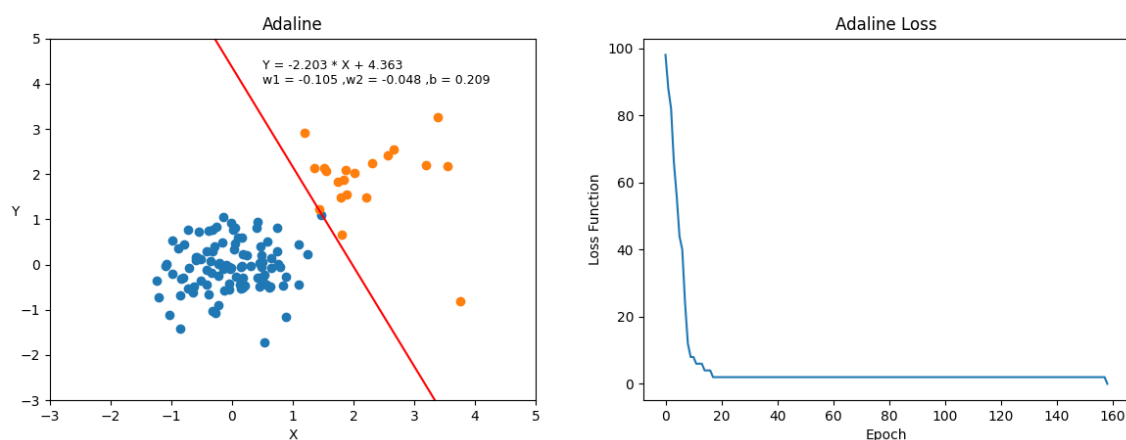
با توجه به تابع فعالسازی موجود برای این نورون، پس از آنکه خروجی به صورت منفی و یا مثبت یک شد، تابع هزینه صفر خواهد شد و بهینه سازی متوقف می شود. در نتیجه خط همواره در نزدیکی یکی از دیتاها خواهد بود. نمودار خطای شبکه نیز به صورت زیر می یابد.

دلیل خوب کار کردن این شبکه این است که شبکه Adaline برای داده هایی که تعداد آنها و واریانس آن ها یکسان است به خوبی عمل کرده و می تواند داده ها را از هم دیگر جدا کند در اینجا نیز داده به تعداد یکسان و با واریانس یکسان هستند پس شبکه به خوبی توانسته آن ها را از هم دیگر جدا کند.



شکل 7 نمودار آموزش و خط جدا کننده دیتا برای تست ست شماره یک

ج) دو دسته داده جدید به صورت زیر هستند. پس از بهینه سازی نتیجه به صورت زیر می شود. همچنین نمودار خطای شبکه به صورت زیر می یابد.



شکل 8 نمودار آموزش و خط جدا کننده دیتا برای تست ست شماره دو

مشاهده می‌شود که در این حالت تعداد ایپاک بیشتری طول کشیده تا شبکه به خط مورد نظر برسد و خطا را صفر کند. پس می‌توان نتیجه گرفت که برای داده‌هایی که پراکندگی بالایی دارد و به صورت رگرسیون خطی جداناپذیر هستند، استفاده از AdaLine مناسب نیست و باید متودهایی مانند MadaLine به کار گرفته شود.

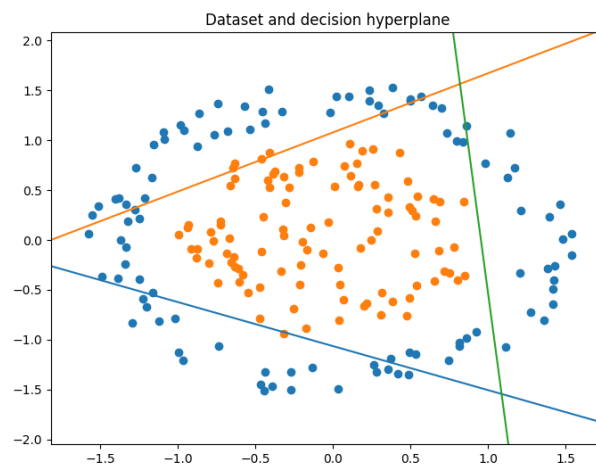
در واقع در صورتی که تعداد داده‌ها و واریانس داده‌ها یکسان نباشد شبکه AdaLine به خوبی نمیتواند داده‌ها را از هم دیگر جدا کند و مقداری خطا دارد همانطور که نتیجه جداسازی بر این موضوع صحت گذاشت.

۲-۲. MadaLine

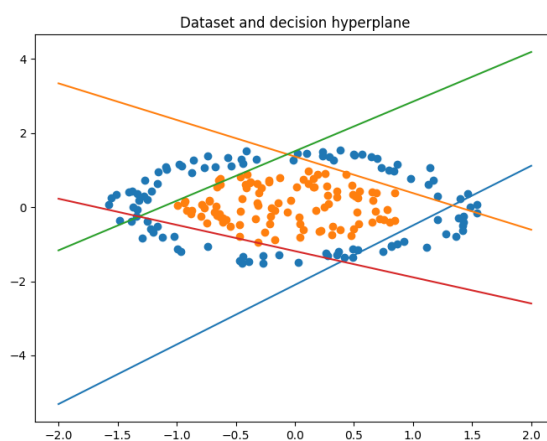
الف) مدالین I یک لایه پنهان تطبیقی و یک ابزار منطقی ثابت (AND، OR، اکثریت گیت) در لایه بیرونی دارد. هدف از روش آموزشی آن کاهش خطای شبکه (در هر نمایش ورودی) با ایجاد کمترین اختلال ممکن در وزن مقداری آدالین است.

توجه داشته باشید که خروجی Adaline باید در علامت تغییر کند تا بر خروجی شبکه تأثیر بگذارد. اگر ورودی خالص به یک آدالین زیاد باشد، برگشت خروجی آن مستلزم مقدار زیادی تغییر در وزن‌های ورودی آن است. بنابراین، الگوریتم تلاش می‌کند تا یک آدالین (گره پنهان) موجود را پیدا کند که ورودی خالص آن کوچک‌ترین مقدار باشد و معکوس شدن خروجی آن خطای شبکه را کاهش دهد. این معکوس خروجی با اعمال الگوریتم LMS به وزن‌های روی اتصالات منتهی به آن آدالین انجام می‌شود. در اینجا سه شبکه مدالین با تعداد متفاوت نورون‌های مخفی وجود دارد که نتایج آنها پس از آموزش در زیر آورده شده است.

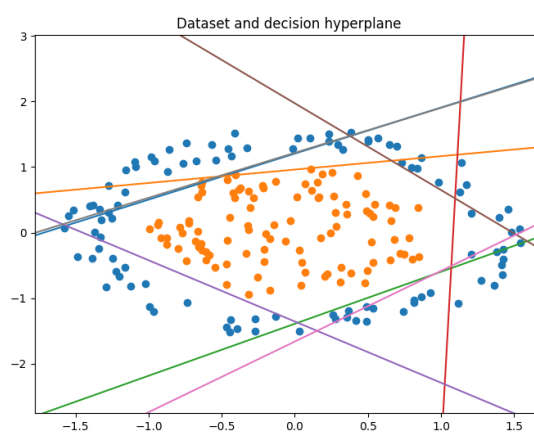
ب) در این بخش هر سه شبکه با تعداد نورون‌های متفاوت آموزش دیده و سپس در بخش بعد نتایج آنها مورد بررسی قرار می‌گیرد.



شکل 9 جداسازی دیتا با سه نورون (خط)



شکل 10 جداسازی دیتا با چهار نورون (خط)



شکل 11 جداسازی دیتا با هشت نورون (خط)

ج) نمودار دقت برای هر سه حالت که به ازای تعداد ایپاک یکسان آموزش دیده اند به صورت زیر است. بدیهی است که با افزایش تعداد نورون های پنهان، پروسه learning پیچیده تر شده و دقت نیز افزایش می یابد.

جدول 1 دقت شبکه ها به ازای تعداد نورون متفاوت

Num of Neurons	Accuracy
3	90%
4	95%
8	89%

پاسخ ۳ – Restricted Boltzmann Machine

۳-۱. سیستم توصیه گر

(A

(B

(C

(D

(E

(F

(G

(H

۴-۱. Multi-Layer Perceptron

(A) اطلاعات دیتافریم خوانده شده به شرح زیر است.

```
#   Column      Non-Null Count  Dtype
---  -
0   id          21613 non-null    int64
1   date         21613 non-null    object
2   price        21613 non-null    float64
3   bedrooms     21613 non-null    int64
4   bathrooms    21613 non-null    float64
5   sqft_living  21613 non-null    int64
6   sqft_lot     21613 non-null    int64
7   floors       21613 non-null    float64
8   waterfront   21613 non-null    int64
9   view         21613 non-null    int64
10  condition    21613 non-null    int64
11  grade        21613 non-null    int64
12  sqft_above   21613 non-null    int64
13  sqft_basement 21613 non-null    int64
14  yr_built     21613 non-null    int64
15  yr_renovated 21613 non-null    int64
16  zipcode      21613 non-null    int64
17  lat          21613 non-null    float64
18  long         21613 non-null    float64
19  sqft_living15 21613 non-null    int64
20  sqft_lot15   21613 non-null    int64
dtypes: float64(5), int64(15), object(1)
memory usage: 3.5+ MB
None
```

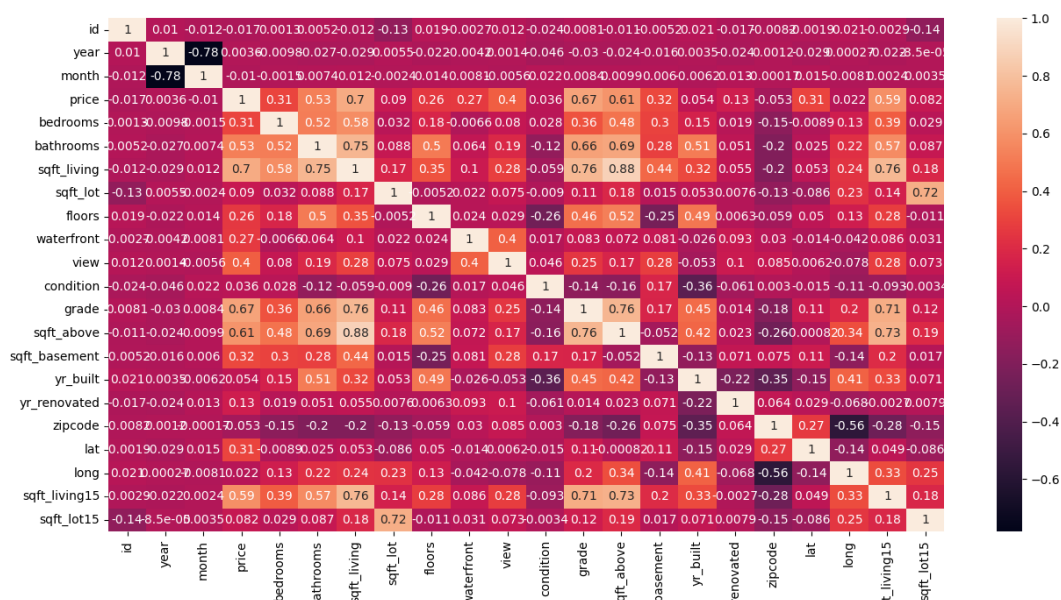
شکل 12 اطلاعات دیتافریم

(B) تعداد داده‌های NaN بر حسب ستون:

```
Num of NaN:
id          0
date        0
price       0
bedrooms    0
bathrooms   0
sqft_living 0
sqft_lot    0
floors      0
waterfront  0
view        0
condition   0
grade       0
sqft_above  0
sqft_basement 0
yr_built    0
yr_renovated 0
zipcode     0
lat         0
long        0
sqft_living15 0
sqft_lot15  0
dtype: int64
```

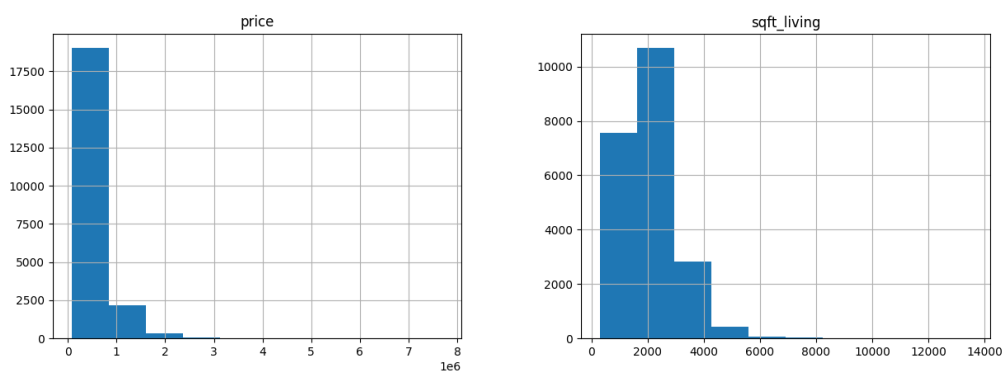
شکل 13 تعداد Nan بر حسب ستون ها

(C) ماتریس Correlation برای همه فیچرها رسم شده است. می‌توان دید که فیچر متراژ خانه با قیمت بیشترین Correlation را دارد.

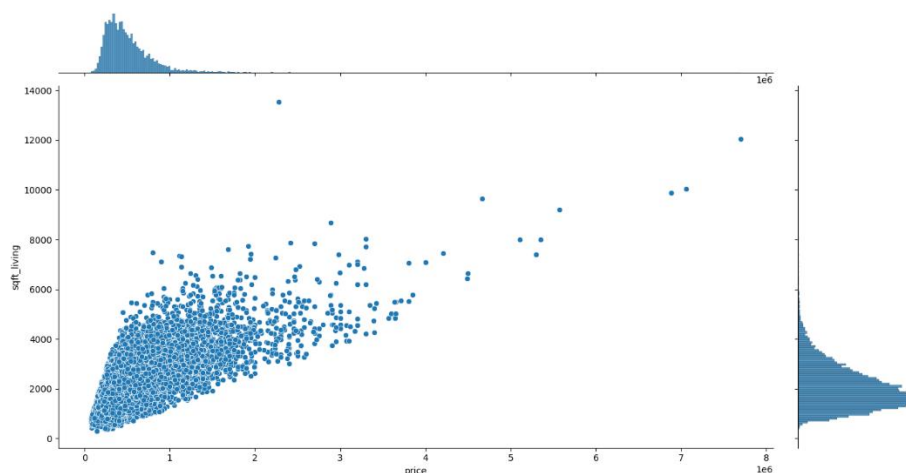


شکل 14 ماتریس همبستگی بر روی دیتاست

(D) نمودار توزیع قیمت و نمودار فیچر sqft_living که بیشترین Correlation را با قیمت دارد.



شکل 15 نمودارهای توزیع



شکل 16 نمودار `sqft_living` برحسب `price`

(E) جداسازی ماه و سال از ستون `Date` و ذخیره آن در ستون های جدید و سپس پاک کردن ستون `Date`.

0	10	0	2014
1	12	1	2014
2	2	2	2015
3	12	3	2014
4	2	4	2015

21608	5	21608	2014
21609	2	21609	2015
21610	6	21610	2014
21611	1	21611	2015
21612	10	21612	2014
Name: month, Length: 21613, dtype: int32		Name: year, Length: 21613, dtype: int32	

شکل 17 تولید ستون ماه و سال از روی تاریخ

(F) جداسازی داده های آموزشی و ارزیابی (80 به 20)

```
#F
msk = np.random.rand(len(df)) <= 0.8

train = df[msk]
test = df[~msk]
```

شکل 18 جداسازی داده آموزش و تست

(G) اسکیل جداگانه دیتاهای آموزش و ارزیابی با کلاس های متفاوت به جهت جلوگیری از رخداد معضل Data Leakage.

```
#G
scaler = MinMaxScaler()

train = scaler.fit_transform(train_df)
test = scaler.transform(test_df)
```

شکل 19 اسکیل دیتای آموزش و تست

(H) تعریف کلاس برای دیتاست و تعریف مدل شبکه عصبی MLP:

```
##H
class data_delivery():
    def __init__(self,data):
        self.dataset = data

    def __len__(self):
        return self.dataset.shape[0]

    def __getitem__(self,idx):
        temp = [*self.dataset[idx][0:3], *self.dataset[idx][4:22]]
        return torch.tensor(temp, dtype=torch.float32), self.dataset[idx][3]

train_loader = data_delivery(train)
test_loader = data_delivery(test)

model = nn.Sequential(
    nn.Linear(21,42),
    nn.ReLU(),
    nn.Linear(42, 42),
    nn.ReLU(),
    nn.Linear(42, 21),
    nn.ReLU(),
    nn.Linear(21, 10),
    nn.ReLU(),
    nn.Linear(10, 1)
)
```

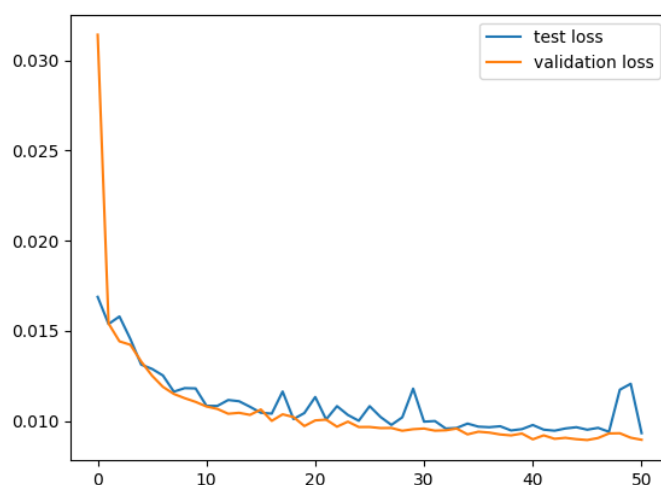
شکل 20 تعریف شبکه عصبی و تابع دیتا

(I) در اینجا Optimizer های Adam و SGD و همچنین Loss Function های L1 و L2 مورد بررسی قرار می گیرند.

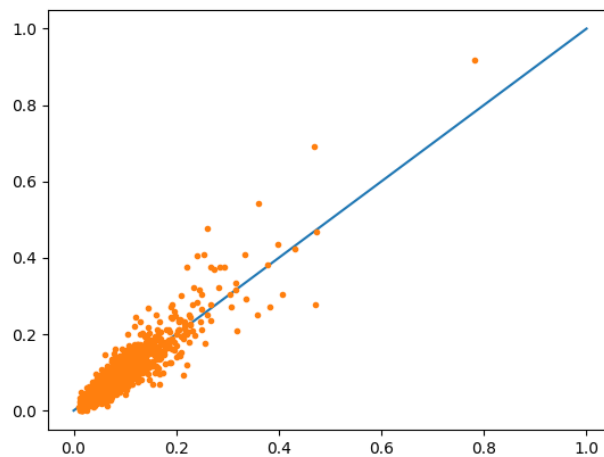
جدول 2 آموزش شبکه به ازای دو لاس و بهینه سازی متفاوت

	Adam	SGD
L1loss	<pre>[23] print(f'best loss:', los_min)</pre> <p>best loss: 0.021215556558829823</p>	<pre>print(f'best loss:', los_min)</pre> <p>best loss: 0.1011214179590459</p>
L2loss	<pre>print(f'best loss:', los_min)</pre> <p>best loss: 0.25044500188529367</p>	<pre>print(f'best loss:', los_min)</pre> <p>best loss: 0.4697685603017472</p>

(J) نمودار loss بر روی دیتای تست به ازای Loss Function های مختلف به صورت زیر می باشد.
نمودار تارگت ها برحسب پیش بینی شبکه: (در حالت ایده آل نقاط به خط همانی نزدیک می شوند)

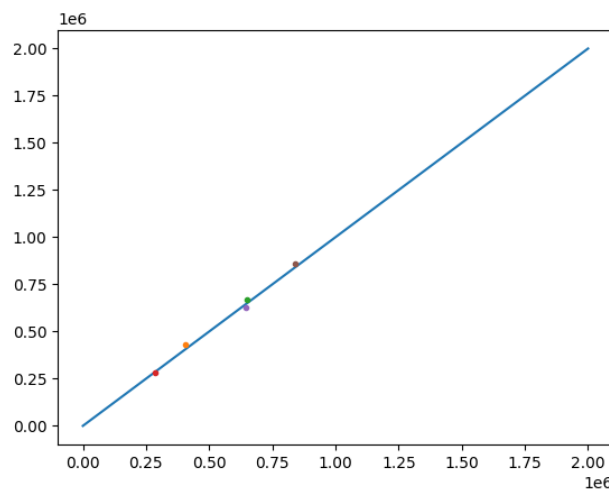


شکل 21 نمودار خطا برای خطای test و validation برای 50 اپیاک



شکل 22 نمودار پیش بینی و هدف قیمت

(K) برای پیش بینی قیمت نیاز است تا خروجی شبکه را با تبدیل معکوس به مقدار واقعی Map کنیم. برای اینکار از تابع inverse transform استفاده می‌کنیم. نمودار تارگت ها برحسب پیش بینی پس از inverse گیری و برگشت از اسکیل:



شکل 23 نمودار پیش بینی و هدف قیمت بدون اسکیل

جدول 3 پیش بینی و قیمت هدف برای 5 دیتای رندوم

Predictions	Targets
405585	430000
651059	671000
644218	625000
840418	862500
287370	282000