به نام خدا

دانشگاه صنعتی امیرکبیر (پلی تکنیک تهران)

دانشکده مهندسی کامپیوتر

شبکه های عصبی

تکلیف دوم

استاد درس: دکتر صفابخش

امیرحسین کاشانی

۴۰۰۱۳۱۰۷۱

نیم سال اول ۱۴۰۱-۱۴۰۲

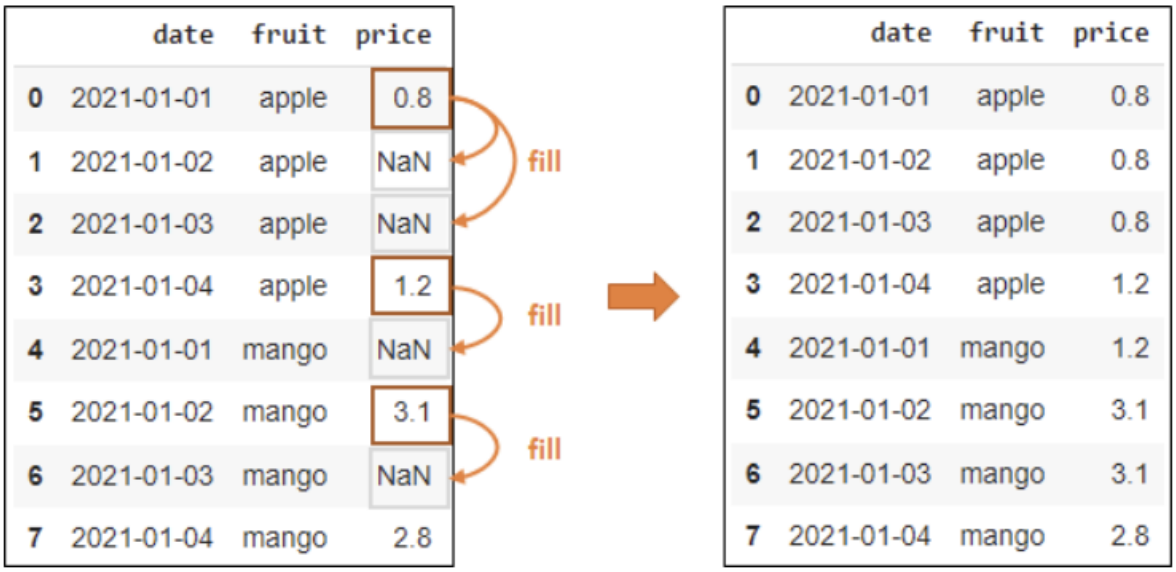
سوال ۱)

الف)

اولین نکته در تخمین مقادیر گم شده این است که ویژگی که تعداد زیادی مقادیر گم شده داشته باشد به طور کلی مناسب نیست و نباید از آن استفاده نمود به بیان دیگر درصورتی که درصد زیادی از مقادیر null بودن آن ستون را drop می‌کنیم. همچنین در صورتی که تعداد داده های ما زیاد بودن می‌توانیم سطر هایی که مقدار null دارند را در شرایطی که به آموزش لطمه نمی‌زنند حذف کنیم (مسلما این روش به عنوان تخمین مقادیر گم شده محسوب نمی‌شوند و در ادامه به روش های تخمین می‌پردازیم)

روش های تخمین

- استفاده از یک مقدار ثابت برای مقادیر null ، این عدد می‌تواند یک عدد دلخواه باشد یا میانگین یا میانه داده های موجود
- تخمین مقدار مورد نظر با استفاده از میانگین بر روی داده های دیگری که المان های مشابه با ما دارند برای مثال اگر فیلد humidity را می‌خواهیم تخمین بزنیم به جای میانگین کل میانگین سطر هایی را به عنوان تخمین مان فرض کنیم که در Region آن ها مانند سطر مورد نظر است.(برای حالات گسسته می‌توانید از پرتکرار ترین المان نیز استفاده نمود)
- روش forward fill که با استفاده از سطر قبلی، مقادیر خالی را پر می‌کند این روش ها بیشتر برای داده هایی که توالی زمانی در برداشت داده وجود داشته اند کاربرد دارد زیرا با توجه به نزدیکی زمانی می‌توانیم فرض کنیم که مقادیر نیز شبیه به هم بوده اند.

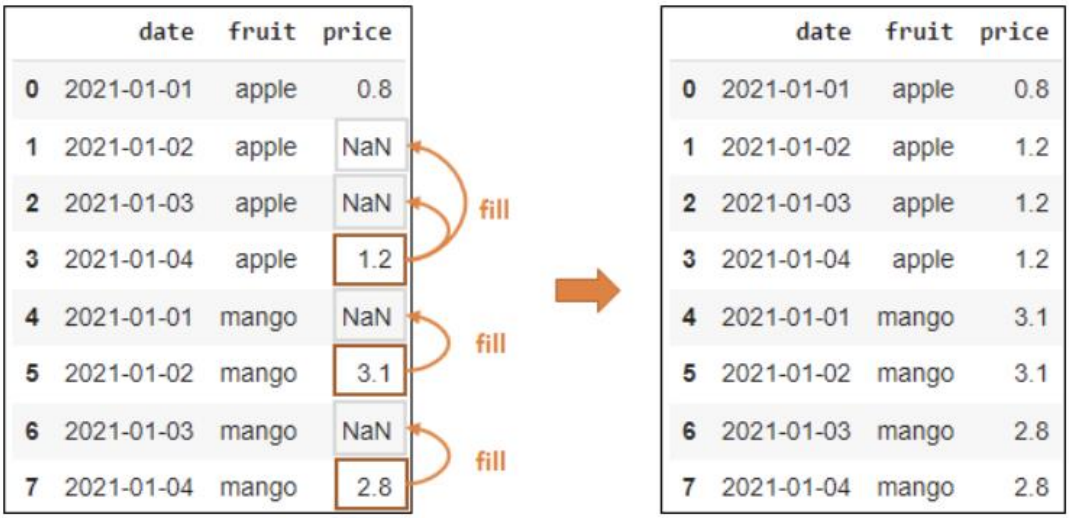


	date	fruit	price
0	2021-01-01	apple	0.8
1	2021-01-02	apple	NaN
2	2021-01-03	apple	NaN
3	2021-01-04	apple	1.2
4	2021-01-01	mango	NaN
5	2021-01-02	mango	3.1
6	2021-01-03	mango	NaN
7	2021-01-04	mango	2.8

	date	fruit	price
0	2021-01-01	apple	0.8
1	2021-01-02	apple	0.8
2	2021-01-03	apple	0.8
3	2021-01-04	apple	1.2
4	2021-01-01	mango	1.2
5	2021-01-02	mango	3.1
6	2021-01-03	mango	3.1
7	2021-01-04	mango	2.8

- Forward fill with group در این روش از سطر قبلی برداشته نمی‌شود بلکه از آخرین سطری که گروه مشترک با این سطر داشته مقادیر برداشته می‌شود.

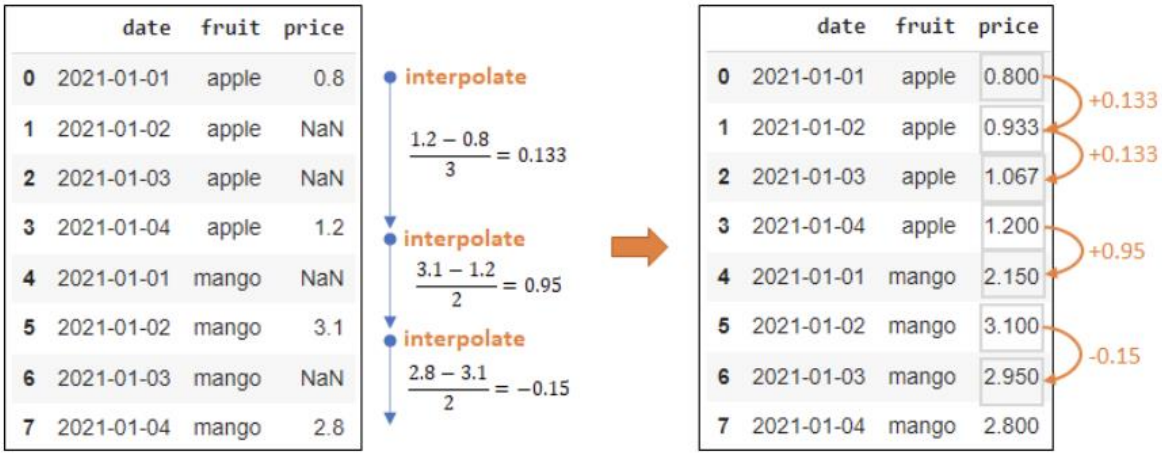
- Back fill مشابه forward می‌باشد با این تفاوت که از سطر بعد استفاده می‌شود.



	date	fruit	price
0	2021-01-01	apple	0.8
1	2021-01-02	apple	NaN
2	2021-01-03	apple	NaN
3	2021-01-04	apple	1.2
4	2021-01-01	mango	NaN
5	2021-01-02	mango	3.1
6	2021-01-03	mango	NaN
7	2021-01-04	mango	2.8

	date	fruit	price
0	2021-01-01	apple	0.8
1	2021-01-02	apple	1.2
2	2021-01-03	apple	1.2
3	2021-01-04	apple	1.2
4	2021-01-01	mango	3.1
5	2021-01-02	mango	3.1
6	2021-01-03	mango	2.8
7	2021-01-04	mango	2.8

- استفاده از روش های interpolate، این روش یک راه حل برای تخمین مقادیر داده ها می‌باشد که ترکیبی از سطرهای قبل و بعد را به کار می‌برد. برای مثال می‌توانید به شکل زیر توجه کنید که از حالت خطی برای تخمین مقادیر بهره برده است.(این روش نیز مانند می‌تواند بر روی گروه نیز اجرا شود)



	date	fruit	price
0	2021-01-01	apple	0.8
1	2021-01-02	apple	NaN
2	2021-01-03	apple	NaN
3	2021-01-04	apple	1.2
4	2021-01-01	mango	NaN
5	2021-01-02	mango	3.1
6	2021-01-03	mango	NaN
7	2021-01-04	mango	2.8

	date	fruit	price
0	2021-01-01	apple	0.800
1	2021-01-02	apple	0.933
2	2021-01-03	apple	1.067
3	2021-01-04	apple	1.200
4	2021-01-01	mango	2.150
5	2021-01-02	mango	3.100
6	2021-01-03	mango	2.950
7	2021-01-04	mango	2.800

- Fill value based on conditions: در این حالت می‌توانیم با استفاده از قوانین از تعریف شده از طرف خودمان مقادیر مورد نظر را بایک قانون خاص پر کنیم. مانند شکل زیر

	date	weekday	fruit	price
0	2021-01-01	True	apple	0.8
1	2021-01-02	False	apple	NaN
2	2021-01-03	False	apple	NaN
3	2021-01-04	True	apple	1.2
4	2021-01-01	True	mango	NaN
5	2021-01-02	False	mango	3.1
6	2021-01-03	False	mango	NaN
7	2021-01-04	True	mango	2.8

mean_price
1.00
1.00
1.00
1.00
2.95
2.95
2.95
2.95

	date	weekday	fruit	price
0	2021-01-01	True	apple	0.800
1	2021-01-02	False	apple	1.250
2	2021-01-03	False	apple	1.250
3	2021-01-04	True	apple	1.200
4	2021-01-01	True	mango	2.950
5	2021-01-02	False	mango	3.100
6	2021-01-03	False	mango	3.688
7	2021-01-04	True	mango	2.800

در این تمرین برای پیش بینی مقادیر عدد از میانگین آن ها استفاده کردیم و برای Region از ماکسیمم تکرار موجود در ستون بهره بردیم.

	data
HumidityMin	12.183320
HumidityMax	23.216552
TVOC	2.307843
eCO2	5.455893
N2ppm	7.606216
SteamSpeed	39.960274
H2-Sensor1	13.998814
H2-Sensor2	18.654023
H2-Sensor3	68.833983
H2-Sensor4	51.539613
Visibility-Left-Sensor	1017.662098
Visibility-Right-Sensor	1015.267384
Cloud9am	4.437415
Cloud3pm	4.504730
TempMin	16.988293
TempMax	21.678913

و تمامی مقادیر categorical نیز با استفاده از بیشترین المان موجود در آن ستون پرشده است.

```
In 18  1  data = data.fillna(data[['SteamDir' ,'RedLightDir1' ,'RedLightDir2','FireAlarm']].mode().iloc[0])
```

تعداد null های موجود در ستون های داده بعد از اعمال اصلاحات

```
In 21  1  data.isnull().sum()

Out 21
```

	data
Home-Loc	0
HumidityMin	0
HumidityMax	0
TVOC	0
eCO2	0
N2ppm	0
SteamDir	0
SteamSpeed	0
RedLightDir1	0
RedLightDir2	0
H2-Sensor1	0
H2-Sensor2	0
H2-Sensor3	0
H2-Sensor4	0
Visibility-Left-Sensor	0
Visibility-Right-Sensor	0
Cloud9am	0
Cloud3pm	0
TempMin	0
TempMax	0
FireAlarm	0

ب)

علت اصلی نرمالسازی داده ها بهبود فرآیند به روز رسانی وزن ها می‌باشد و به طور کلی تجربه آموزش مدل های مختلف نشان داده است که آموزش ویژگی هایی که در بازه های نزدیک به صفر هستند آسان تر و سریع تر بوده است که از دلایل آن می‌توان به توابع فعال سازی اشاره کرد حول مقادیر نزدیک به صفر تصمیم گیریشان عوض می‌شود، به بیان دیگر این توابع فعال سازی تغییرات در مقادیر بزرگ را آنچنان خوب رصد نمی‌کنند (بدلیل وجود بخش های flat برای مثال sigmoid دو بخش flat دارد که یک در بخش منفی و مثبت می‌باشد) و وقتی یک ضریب در یک عدد بسیار بزرگ شود با توجه به اینکه در ناحیه ای می‌افتد که flat می‌باشد مشتق چندانی بزرگی نخواهد داشت و بسیار کند به مرز تغییر تصمیم حرکت خواهد کرد از این رو بهتر است که سعی کنیم تا به نرمال سازی رنج ویژگی های داده هایمان را به صفر (بین صفر و یک یا بازه های در این حدود) نزدیک کنیم.

در بین ویژگی های موجود Visibility-Left-Sensor و Visibility-Right-Sensor با توجه به اینکه در حدود مقدار ۱۰۰۰ در بازه ای به اندازه ۶۰ واحد نوسان می‌کنند به احتمال زیاد باعث ایجاد دشواری بیشتری برای مدل می‌شوند. علت این انتخاب نیز به این دلیل است که این ویژگی های مقادیر بزرگی هستند و در وزن دهی اولیه بیشترین تاثیر را رو مقدار خروجی ما می‌گذارند و تا زمانی که وزن های این ستون اصلاح نشوند سایر وزن ها چندان به درستی آموزش نمی‌بینند.

```
In 37  1  cols_to_norm = ['HumidityMin','HumidityMax','TVOC','eCO2','N2ppm','SteamSpeed','H2-Sensor1','H2-Sensor2','H2-Sensor3','H2-Sensor4','Visibility-Left-Sensor',
          'Visibility-Right-Sensor','Cloud9am','Cloud3pm','TempMin','TempMax']
       2  data[cols_to_norm] = data[cols_to_norm].apply(lambda x: (x - x.min()) / (x.max() - x.min()))

In 38  1  data

Out 38
```

	Home-Loc	HumidityMin	HumidityMax	TVOC	eCO2	N2ppm	SteamDir	SteamSpeed	RedLightDir1	RedLightDir2	H2-Sensor1	H
6	AliceSprings	0.657568	0.714012	0.009680	0.060325	0.391608	WNW	0.234375	SE	ENE	0.084615	
7	MelbourneAirport	0.600496	0.644914	0.000000	0.109049	0.496503	N	0.523438	N	N	0.315385	
8	Watsonia	0.550868	0.479846	0.009680	0.076566	0.363636	W	0.257502	NW	W	0.100000	
9	MountGinini	0.275434	0.355086	0.000000	0.063293	0.531903	NE	0.187500	NE	ENE	0.030769	
10	Witchcliffe	0.394541	0.541267	0.001489	0.063293	0.531903	S	0.250000	E	SSE	0.130769	
11	Woomera	0.682382	0.748560	0.037975	0.120650	0.776224	SW	0.523438	N	NW	0.184615	
12	Albury	0.205955	0.366603	0.000000	0.063293	0.531903	SE	0.093750	E	S	0.053846	
13	MountGambier	0.466501	0.479846	0.000000	0.030162	0.076923	SSW	0.164062	WNW	WSW	0.046154	
14	AliceSprings	0.322581	0.428023	0.000000	0.048724	0.762238	SSE	0.304688	SE	SE	0.130769	

تصویر بعد از اعمال نرمال سازی

ج)

برای تحلیل داده های categorical بهترین حالت استفاده از one_hot_encoding می‌باشد اما تا زمانی این روش مناسب است که تعداد دسته های ویژگی ها کم باشد ولی با توجه به اینکه تعداد دسته های موجود برای هر ویژگی زیاد تر است نمی‌توانیم برای آن ها از one_hot_encoding بهره ببریم در نتیجه از label_encode استفاده می‌کنیم که مقادیر گسسته ما را به اعداد تبدیل می‌کند و تعداد دسته های مارا نیز حفظ می‌کند.

```
In 65  1  data[['Home-Loc' ,'SteamDir','RedLightDir1' , 'RedLightDir2']] = data[['Home-Loc' ,'SteamDir','RedLightDir1' , 'RedLightDir2']].apply(LabelEncoder().fit_transform)
       2  data
```

Out 65

	Home-Loc	HumidityMin	HumidityMax	TVOC	eCO2	N2ppm	SteamDir	SteamSpeed	RedLightDir1	RedLightDir2	H2-Sensor1	H2-Senso
0	18	0.513234	0.537746	0.008592	0.030162	0.293706	12	0.281250	12	8	0.184615	0.1
1	13	0.873449	0.706334	0.013403	0.092807	0.209790	13	0.250000	13	14	0.169231	0.2
2	42	0.583127	0.654511	0.008592	0.063293	0.531903	14	0.421875	5	13	0.069231	0.3
3	39	0.806452	0.667946	0.014892	0.092807	0.531903	1	0.421875	2	0	0.084615	0.1
4	47	0.744417	0.518234	0.000745	0.063293	0.531903	3	0.335938	9	12	0.146154	0.1
5	37	0.444169	0.431862	0.000000	0.027842	0.188811	13	0.132812	14	3	0.115385	0.1
6	3	0.657568	0.714012	0.009680	0.060325	0.391608	14	0.234375	9	1	0.084615	0.1
7	19	0.600496	0.644914	0.000000	0.109049	0.496503	3	0.523438	3	3	0.315385	0.4
8	44	0.550868	0.479846	0.009680	0.076566	0.363636	13	0.257502	7	13	0.100000	0.2

نمونه این از داده ها بعد از اعمال این بخش

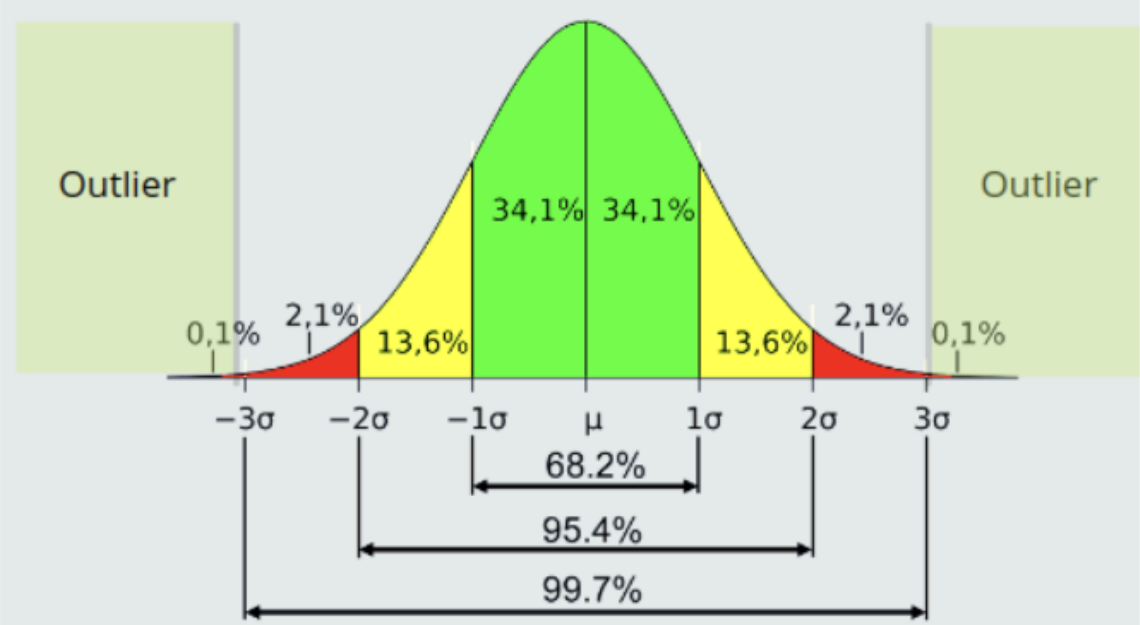
د)

Outlier ها مجموعه داده های هستند که به شکل قابل ملاحظه ای با بقیه داده ها تفاوت زیادی دارند و در امر آموزش و ارزیابی موجود بروز خطا در تحلیل های آماری می‌شوند. به طور برای مقابله با خطا دو سیاست مطرح می‌شود یک اینکه داده پرت را حذف شده در نظر بگیریم(مانند اینکه از سر و ته داده حدود ۱ تا ۵ درصد داده ها را خط بزنیم)، دو اینکه بعد از تشخیص outlier آن را با یک ماکسیمم یا مینیممی جایگزین کنیم (مانند جایگزین کردن داده های n درصد ابتدایی با کوچکترین مقدار باقی ماند در دیتا ها و n درصد انتهایی با بزرگترین داده باقی مانده).

روش های یافتن outlier

Z_score

این معیار ار لحاظ آماری بیان می‌کند که چه میزان داده مورد منظر از میانگین داده ها دور است و از تقسیم فاصله تا میانگین بر انحراف معیار بدست می‌آید.



با توجه به شکل بالا به عنوان یک قانون مرسوم بعد از ۳ انحراف معیار فاصله از میانگین را داده پرت منظور می‌کنند.

Local Outlier Factor(LOF)

این روش با مقایسه تراکم محلی در اطراف داده مورد نظر بررسی می‌کند که آیا این داده یک داده پرت است یا خیر مفهوم تراکم از فاصله k نزدیکترین همسایه می‌آید و هرچه فاصله بیشتر باشد تراکم کمتر است. مقدار تراکم یک داده با میانگین همسایه های آن سنجیده می‌شود.

$$local\ desnsity = \frac{k}{\sum_{i=1}^{k} d_i}$$

$$LOF = \frac{average\ local\ density\ of\ neighbors}{local\ density\ of\ focus\ point}$$

درصورتی که LOF مقداری بیش از یک بگیرید یعنی فاصله آن از حالت نرمال نسبت به بقیه داده ها بیشتر است اگر این مقدار از حد آستانه بیشتر شود می‌توانیم آن را به عنوان داده پرت فرض کنیم.

Outlier Detection using In-degree Number (ODIN)

این روش به نوعی بر عکس روش k-NN می‌باشد در این روش ابتدا K را مشخص می‌کنیم و سپس برای هر نود محاسبه میکنیم که چند بار به عنوان نزدیک ترین k عضو انتخاب شده است هر چه قدر این میزان بزرگتر باشد به این معنی است که این داده به سایر داده ها نزدیک تر است و هر چه قدر کمتر باشد به این معنا است که این داده به صورت ایزوله در دیتاست ما حضور دارد.

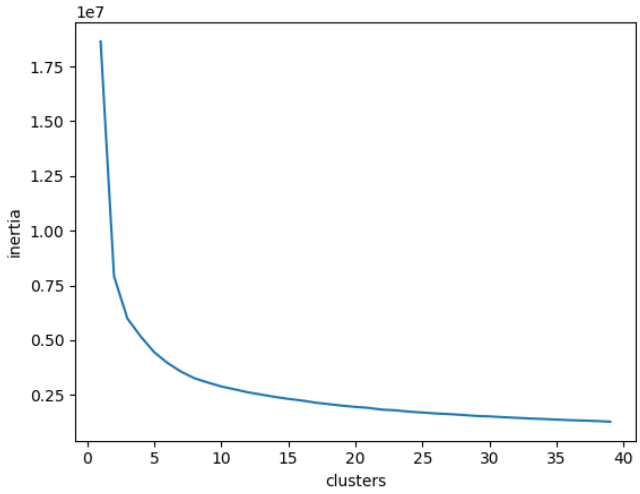
Auto Encoder

در این روش یک شبکه عصبی برای تشخیص اینکه آیا داده مورد نظر پرت است یا خیر تشکیل می‌شود.

توضیح استفاده از k_means

الگوریتم K_means یک الگوریتم بدون نظارت برای خوشـــه بندی داده ها اســـت که در ابتدا k مرکز که k یک ابرمتغیر (hyper parameter) می‌باشد را بصورت تصادفی انتخاب می‌کند و سپس هر داده بررسی می‌شود که به کدام دسته متعلق است حال درگام بعدی مرکز دسـته به روز رسـانی می‌شـود(دو حالت برای به روز رسـانی دسـته وجود دارد یک اینکه صـرفا میانگین داده ها را به عنوان مرکز دسته فرض کنیم دو اینکه مرکزی ترین داده را به عنوان مرکز دسته انتخاب کنیم، لازم به ذکر است که عموما روش اول مورد استفاده قرار می‌گیرد) این کار را تازمانی ادامه می‌دهیم تا به همگرایی برسیم و به روز رسانی در نقاط دسته ها نداشته باشیم یا جابه جایی مراکز از یک threshold کمتر شـــود.در گام آخر داده هایی که از مرکز دســـته خود دور هســـتند را به عنوان داده پرت اعلام می‌نماییم.

در این سوال ابتدا الگوریتم را برای K های متفاوت اجرا کردیم و سـپس براسـاس معیار انسـجام دسـته بندی عدد k= 25 برای این مساله فرض شده است.(شکل زیر نمودار انسجام به تعداد کلاستر ها می‌باشد)



سپس داده هایی که فاصله آن ها تا مرکز خودشان بیشتر از ۹ بوده است حذف گردیده اند. تعداد کل داده ها ۶۹۹۹۹ بوده است که از این تعداد ۱۰۶۴ داده پرت شناسایی شده اند که معاد ۱.۵۴ درصد از داده ها بوده اند.

```
initial size of data 69999
number of lines removed 1064
removed percentage 1.5200217145959227
```

بخش ه)

عکس زیر نشان دهند تعداد اعضای هر یک از بخش ها می‌باشد

```
data length : 69999
train data length : 48999  --- %69.9995714224489
test data length : 13999  --- %19.99885712653038
validation data length : 7001 --- %10.001571451020729
```

سوال ۲)

ساده ترین شبکه عصبی ، شبکه single-layer پرسپترون می‌باشد که در آن هر نرون ورودی به نرون خروجی متصل می‌گردد که هر لایه در این شبکه می‌تواند تابع فعال سازی مربوط به خود را داشته باشد. در سال ۱۹۶۹ در مقاله ای توسط آقای مینسکی اثبات گردید که پرسپترون تک لایه فقط امکان حل مسائل خطی را دارد و هرکز نمی‌تواند مساله XOR را حل کند از این رو در صورتی که مدل تک لایه ی ما بتواند داده مورد نظر را به درستی دسته بندی کند این مسال جدایی پذیر خطی خواهد بود.

```
1532/1532 [==============================] - 1s 438us/step
accuracy in train 0.783485377252597
```

```
219/219 [==============================] - 0s 480us/step
accuracy in validation 0.776746179117269
```

```
438/438 [==============================] - 0s 439us/step
accuracy in test 0.7790556468319165
```

با توجه به نتایج بدست آمده مدل مساله ما خطی نمی‌باشد و نیاز به استفاده از مدل های پیچیده تر در این مساله وجود دارد.

سوال ۳)

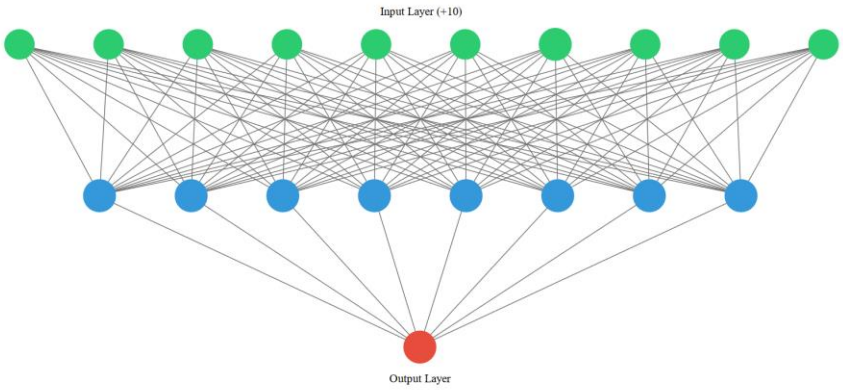
نتایج خام بدســت آمده از اجرا های متفاوت در شــکل های ترســیم شــده (تعداد epoch المان اول داخل پرانتز و تعداد نورون لایه پنهان المان دوم) و عدد سمت راست نشان دهنده میزان صحت داده آموزش داده شده بر روی validation می‌باشد.(معماری این بخش ۳ لایه فرض شــده که لایه ورودی ۲۰ نورون (به تعداد ویژگی ها) ، لایه نهایی یک نورون و لایه ی میانی برای حالات مختلف آزمایش گردیده است.)

```
Out 127    {(20, 3): 0.776746179117269,
            (20, 5): 0.776746179117269,
            (20, 8): 0.776746179117269,
            (20, 10): 0.776746179117269,
            (20, 15): 0.776746179117269,
            (20, 20): 0.776746179117269,
            (50, 3): 0.776746179117269,
            (50, 5): 0.8388801599771462,
            (50, 8): 0.8393086701899729,
            (50, 10): 0.8060277103270961,
            (50, 15): 0.223253820882731O5,
            (50, 20): 0.776746179117269}
```

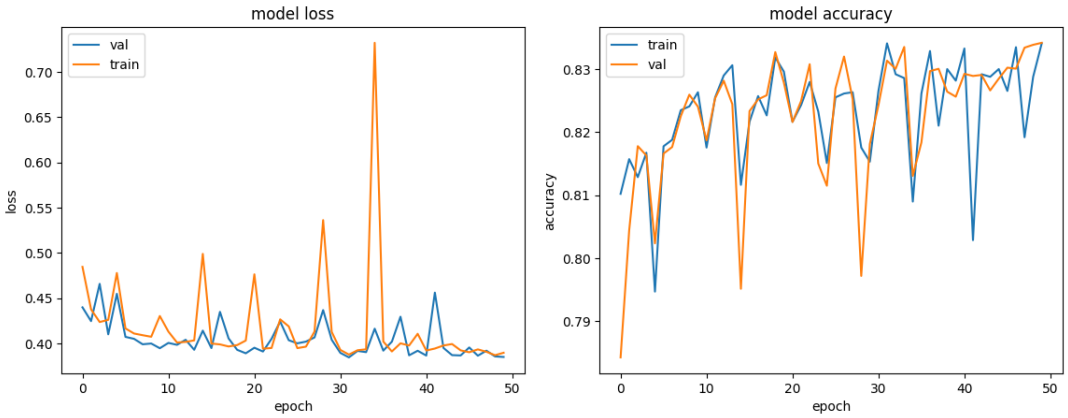
در جدول زیر با توجه به اینکه در صورت سوال گفته شده که حتما جدول ترسیم شود سطر های مهم تر بصورت خلاصه آورده شده اند.

Epoch	Neurons	Accuracy
۲۰	۳	۰/۷۷۶
۲۰	۲۰	۰/۷۷۶
۵۰	۸	۰/۸۳۹
۵۰	۳	۰/۸۳۸
۵۰	۱۰	۰/۸۰۶
۵۰	۱۵	۰/۲۲۳

با توجه به نتایج بدست آمده مدل پیشنهادی ما برای حل این مساله شکبه عصبی 1-8-20 می‌باشد که توابع فعال سازی آن نیز در شکل مشاهده می‌کنید (در این بخش هنوز بهینه سازی در بر روی تابع فعال سازی صورت نگرفته است)



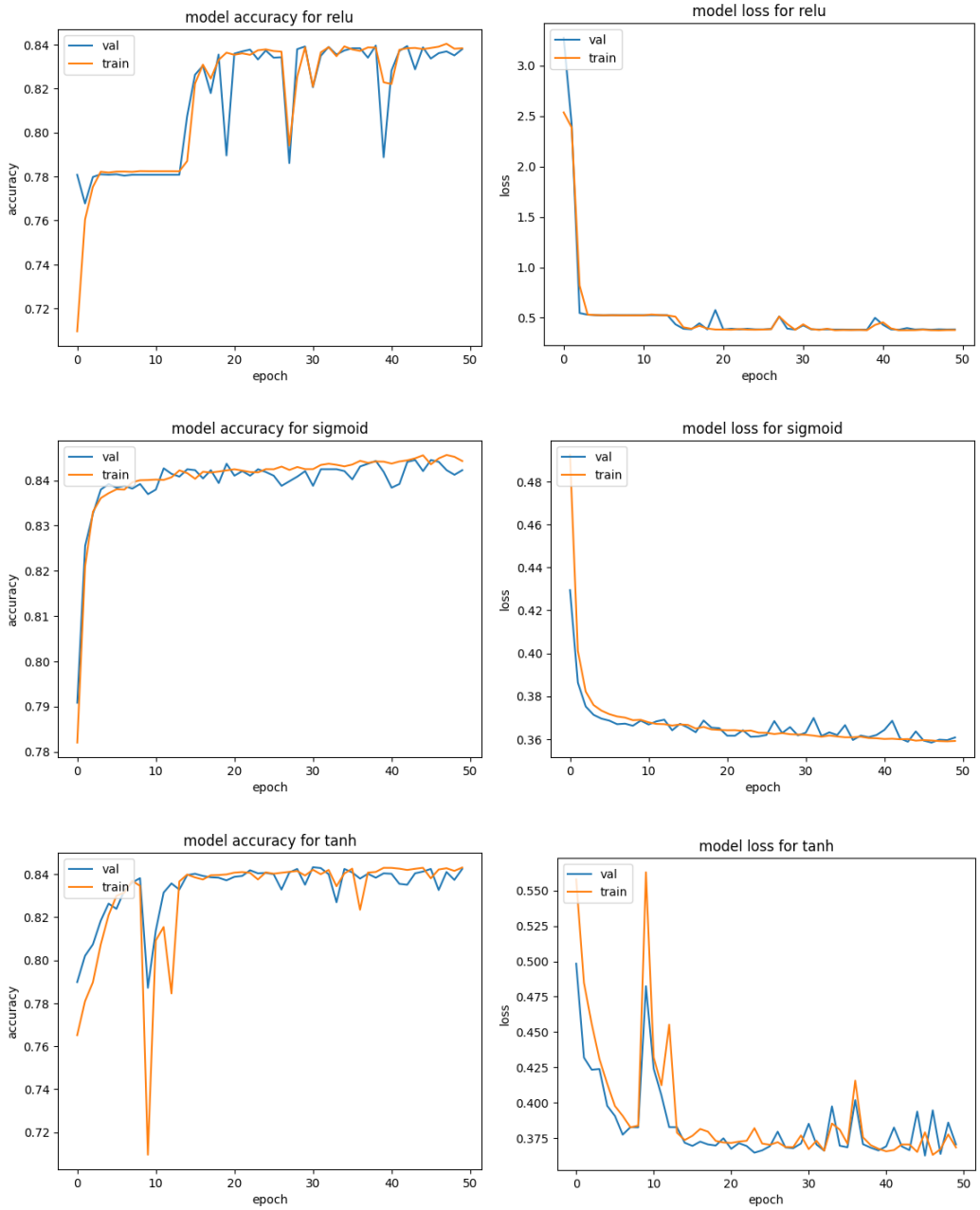
نمودار های مدل نهایی (1-8-20) در ۵۰ epoch (نمودار val برای validation set می‌باشد)

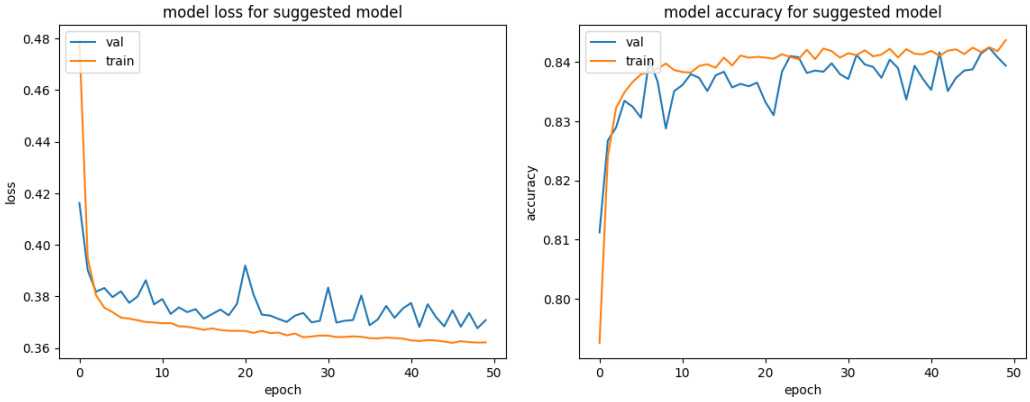


سوال ۴)

تابع فعال سازی یکی از مهمترین بخش های شبکه های عصبی می‌باشد که تصمیم می‌گیرد که آیا یک نورون فعال شود یا خیر و این نتیجه را به نرون های لایه ی بعد منتقل می‌کند. Activation function خروجی مد نظر را بین ۰ تا ۱ یا ۱- تا ۱ نرمال می‌کند و در فرآیند آموزش به روش backpropagation به اجرای صحیح gradient descent کمک می‌کنند.

نتایج بدست آمده توسط activation functions های بدست آمده متفاوت است زیرا پاسخ هر یک از این توابع به مقدار خطا متفاوت است و از هریک باید در جای مناسب خود با توجه به مساله استفاده نمود.





مدل پیشـــنهادی یک مدل ترکیبی اســت که در دولایه اول از relu اســتفاده شــود و در لایه ســوم از sigmoid بهره گرفته شــده است.(معماری مورد استفاده 1-8-20 می‌باشد)

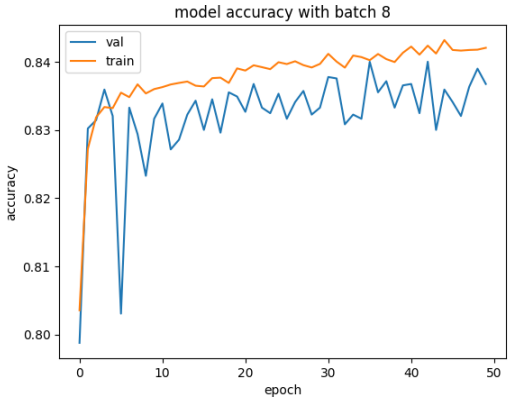
سوال ۵)

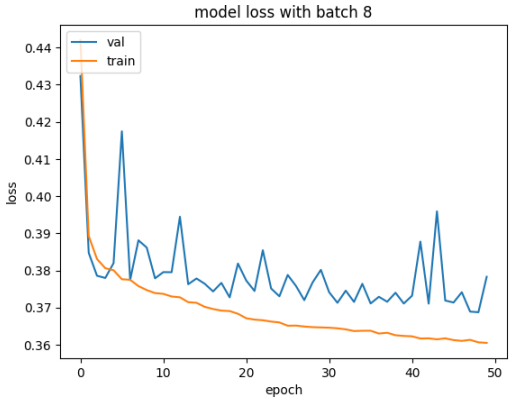
در برخـی مـوارد، اسـتفاده از میـانگین تغییـرات وزن هـا از روی مجموعـه ای از داده هـا (یـا حتـی یـک دوره کامـل، اگـر نمونـه های زیادی وجـود نداشـته باشـد) و یـک بـروز رسـانی وزن (برابـر میـانگین شـرایط اصـلاح وزن) می‌توانـد اثـر بهتـری از بـه روز رسانی مدل به ازای هـر داده داشـته باشـد. ایـن روش رونـد آمـوزش را نـرم تـر می‌کنـد و از نوسـانات ناگهـانی مـدل جلـوگیری می‌کنـد. در برخـی مـوارد، ایـن هموارسـازی ممکـن اسـت شـانس همگرایـی را بـه حـداقل محلـی را افـزایش دهـد. (از کتـاب Fausett) همچنـین در برخـی مـوارد امکـان مـوازی سـازی نیـز بـرای اجـرای فرآینـد آمـوزش نیـز فـراهم می‌گردد.(کتـاب Simon O. Haykin)

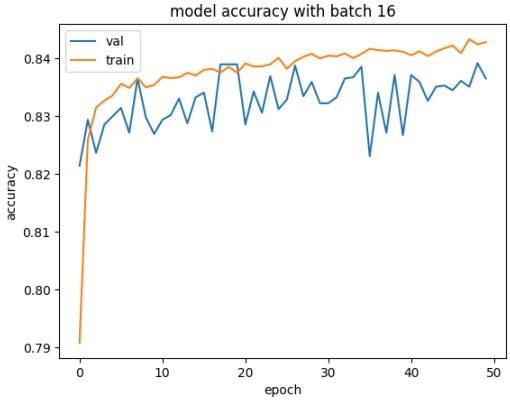
بـه طـور کلـی در صـورتی کـه batch_size را افـزایش دهـیم یـک epoch زود تـر بـه پایـان می‌رسـد بـه دلیـل اینکـه بـار محاسباتی یـک epoch کمتـر می‌شـود و تعـداد بـروز رسـانی هـا نیـز کـاهش می‌یابـد. امـا بـا توجـه بـه اینکـه میـزان دفعـات تغییـرات وزن را در یـک epoch کمتـر می‌کنـد نمـودار epoch_error کنـد تـر همگـرا می‌شـود امـا از لحـاظ زمـانی در مجموع با توجه بـه مقاومـت در مقابـل بـه روز رسـانی ناگهـانی در راسـتای داده هـای پـرت می‌تـوان گفـت همگرایـی سـریع تـر می‌شود.

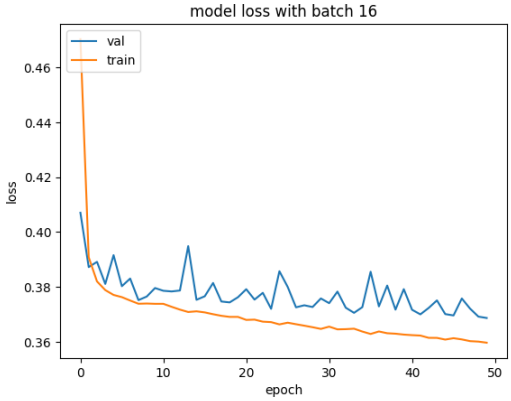
نمودار های بدست آمده برای accuracy

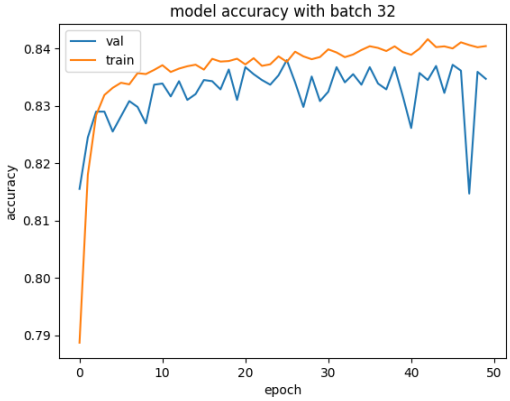
(لازم به ذکر است که نمـودار هـای قبلـی بـا batch_size = 32 ایجـاد شـده اسـت کـه مقـدار پـیش فـرض بـرای آمـوزش داده ها می‌باشد)

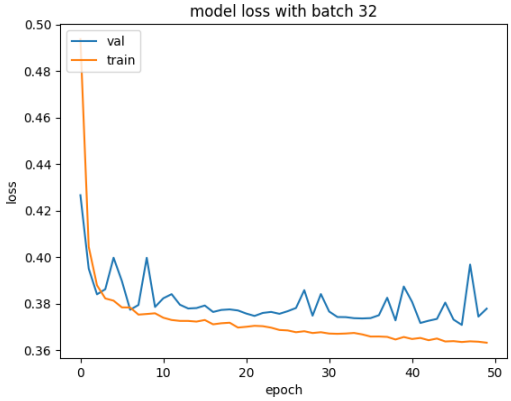

model accuracy with batch 8
val
train
accuracy
epoch

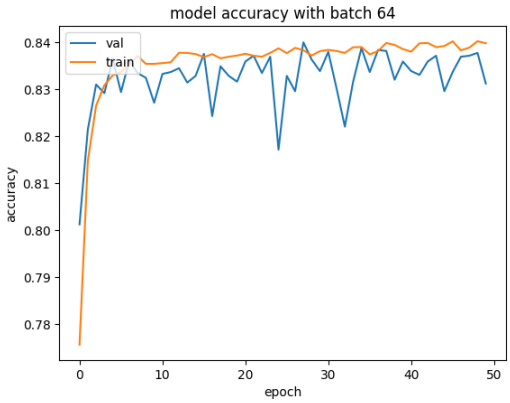

model loss with batch 8
val
train
loss
epoch

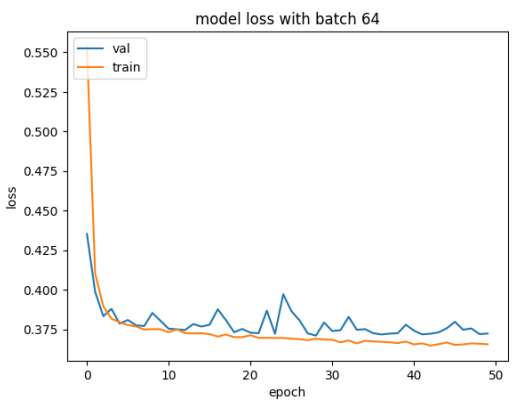

model accuracy with batch 16
val
train
accuracy
epoch

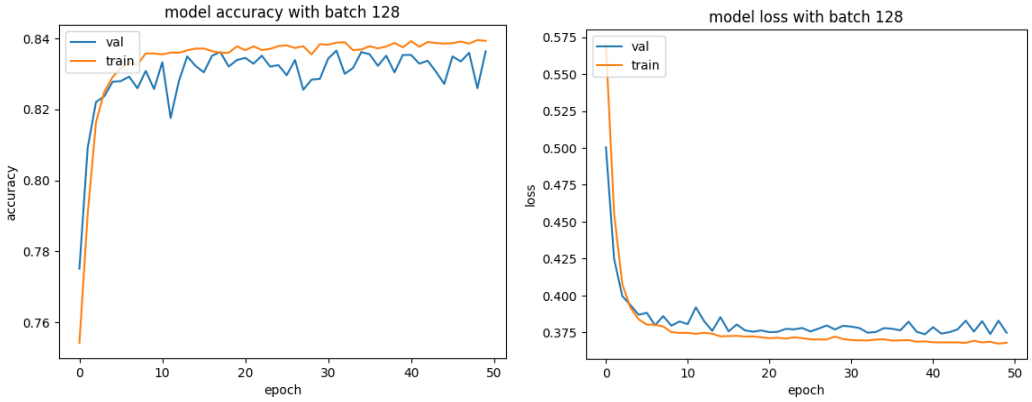

model loss with batch 16
val
train
loss
epoch


model accuracy with batch 32
val
train
accuracy
epoch


model loss with batch 32
val
train
loss
epoch


model accuracy with batch 64
val
train
accuracy
epoch


model loss with batch 64
val
train
loss
epoch



Confusion matrix for batch_size = 8

```
**************
accuracy  : 0.839345667547682
Confusion_matrix:  tf.Tensor(
[[10334   572]
 [ 1677  1416]], shape=(2, 2), dtype=int32)
***********
```

Confusion matrix for batch_size = 16

```
**************
accuracy  : 0.8397028359168512
Confusion_matrix:  tf.Tensor(
[[10289   617]
 [ 1627  1466]], shape=(2, 2), dtype=int32)
***********
```

Confusion matrix for batch_size = 32

```
**************
accuracy  : 0.8364168869204943
Confusion_matrix:  tf.Tensor(
[[10472   434]
 [ 1856  1237]], shape=(2, 2), dtype=int32)
***********
```

Confusion matrix for batch_size = 64

```
*************
accuracy  : 0.8376312593756697
Confusion_matrix:  tf.Tensor(
[[10489   417]
 [ 1856  1237]], shape=(2, 2), dtype=int32)
***********
```

Confusion matrix for batch_size = 128

```
*************
accuracy  : 0.835631116508322
Confusion_matrix:  tf.Tensor(
[[10166   740]
 [ 1561  1532]], shape=(2, 2), dtype=int32)
***********
```

نتایج بدست آمده حاکی از این است که در تعداد epoch ۵۰ ، تغییر اندازه batch کمکی به میزان دقت در انتها نکرد اما با افزایش آن، حرکت هموارتر و همگرایی از لحاظ زمانی بسیار سریع تر شد(مدت زمانی که batch= 8 نیاز دارد حدود ۴ برابر کند تر از batch = 128 می‌باشد).

سوال ۵ دومی)

Overfit زمانی رخ می‌دهد که جزییات بیش از حدی را از داده های آموزش فرابگیرد که این جزییات می‌تواند شامل نویز و داده های پرت داخل مدل نیز باشد که از نتایج آن عملکرد غیر واقع گرایانه و عالی در داده های آموزش و عملکرد ضعیف در داده هایی است که تاکنون ندیده است می‌باشد که در نتیجه مدل بدست آمده generalization خوبی برخوردار نیست.

برای شناسایی overfit می‌بایست عملکرد در داده های آموزش و تست را با یک دیگر مقایسه نمود اگر فاصله معنا داری بین این دو وجود داشته باشد می‌تواند یکی از نشانه های اصلی overfitting بر روی داده های آموزش باشد.

برای رفع مشکل

۱. از یک مدل ساده تر استفاده کنید، این ساده سازی می‌تواند در کم کردن تعداد لایه های شبکه عصبی باشد یا استفاده از توابع خطی به جای توابع غیر خطی به عنوان توابع فعال سازی باشد.

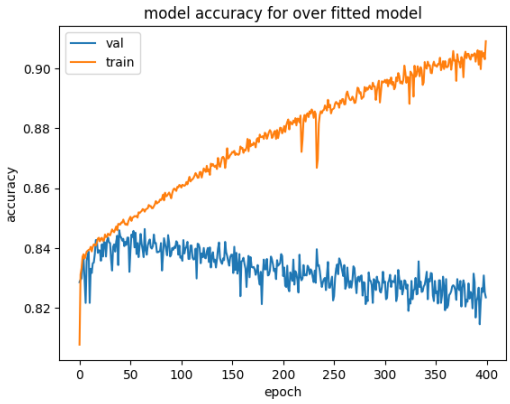
۲. افزودن داده های جدید به فرآیند آموزش

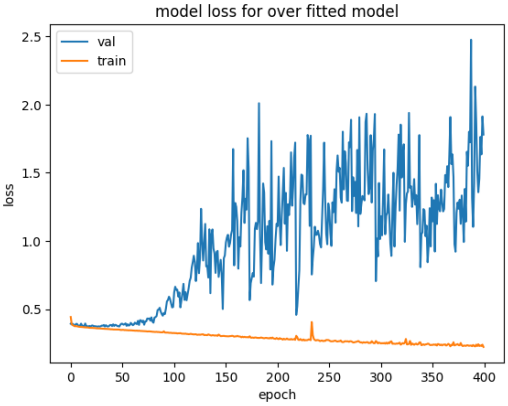
۳. کاهش تعداد epoch ، هر epoch به معنی یک بار بررسی همه داده ها است در صورتی که تعداد epoch را خیلی زیاد قرار دهیم مدل سمت حفظ داده های آموزش می‌رود و overfit می‌شود با افزایش تعداد داده ها می‌توانیم epoch های کمتری استفاده کنیم و به نتیجه مورد نظر خودمان برسیم

۴. افزودن داده به روش augmentation، مانند افزودن نویز به داده های قبلی ، در پردازش تصویر چرخاندن عکس ها یا mirror کردن آنها و ....

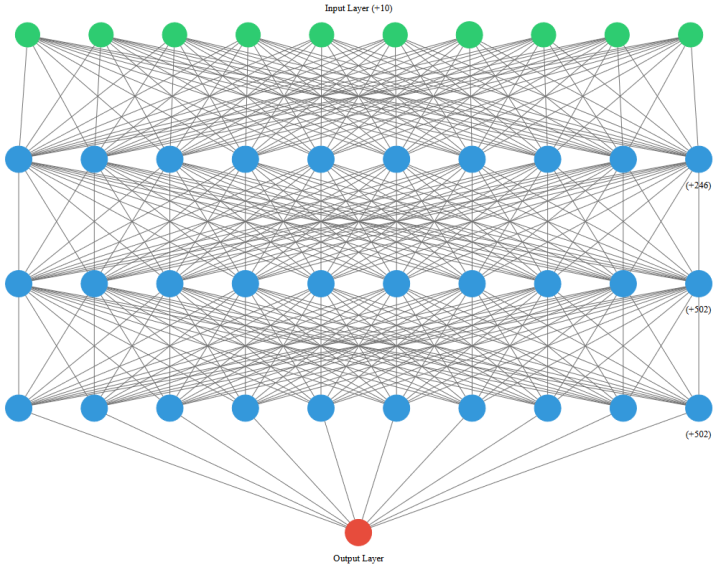
۵. استفاده از روش های ensemble نیز توصیه شده است البته این روش مشکل را حل نمی‌کند بلکه با ایجاد مدل های مختلف و رای گیری بین آن ها به نحوی دیگر مشکل را برطرف می‌کند.

برای ایجاد یک مدل با ویژگی بیش برازش در این مساله از افزودن تعداد لایه ها و تعداد نرون ها در کنار بالا بردن تعداد epoch ها بهر بردیم. معماری مورد استفاده (20-256-512-512-1) با activation_function= relu در ۴۰۰ epoch بدست آمده است



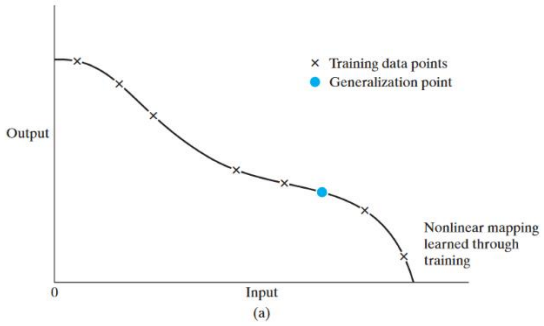


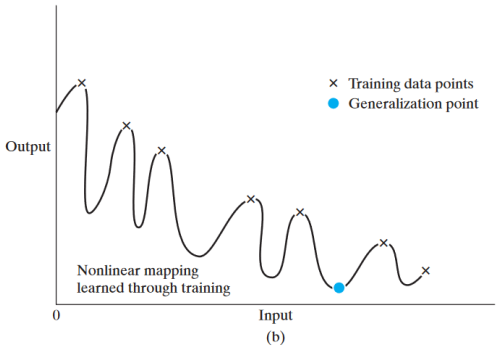

```
*************
accuracy   : 0.8255589684977498
Confusion_matrix:  tf.Tensor(
[[10068   848]
 [ 1594  1489]], shape=(2, 2), dtype=int32)
***********
```



سوال ۶)

یک شبکه عصبی generalization قابل قبولی دارد اگر داده های ورودی که که تاکنون ندیده است (داده های تست) را به درستی به خروجی مورد نظر نگاشت کند. در تعریف generalization فرض بر این است که داده های تستی که مورد بحث ما هستند از population مشابه جامعه آموزش بوجود آمده اند اما تاکنون دیده نشده اند. برای مثال شکل b خطای کمتری در بخش آموزش دارد اما در بخش تست عملکرد ضعیف تری از شکل a دارد.(شکل و پاراگراف از کتاب Simon O. Haykin)



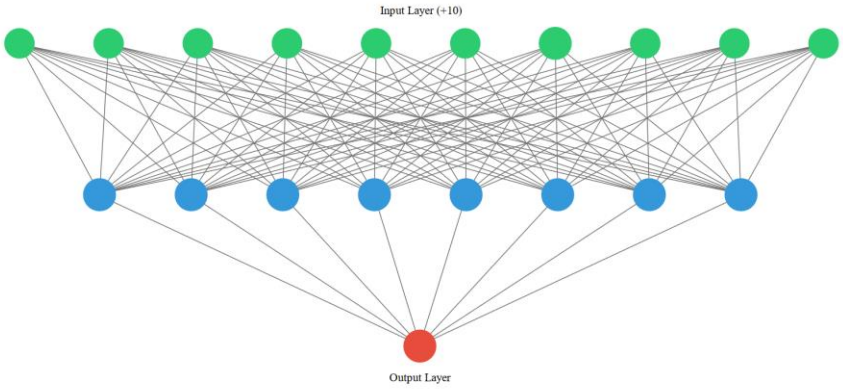


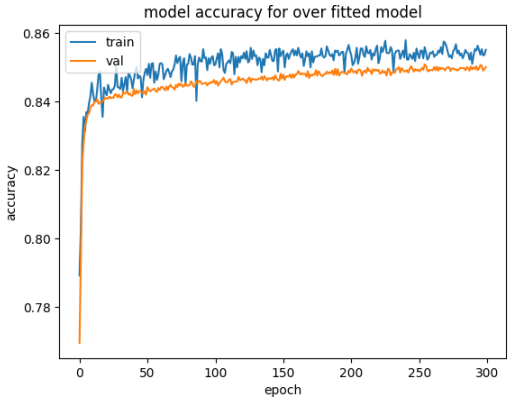
علت اهمیت تعمیم پذیری در این است که اگر این ویژگی را یک مدل نداشته باشد نمی‌تواند برای داده های جدید تصمیم گیری بکند و در هدف اصلی مدل که پیش بینی حالات جدید است باز می‌ماند.

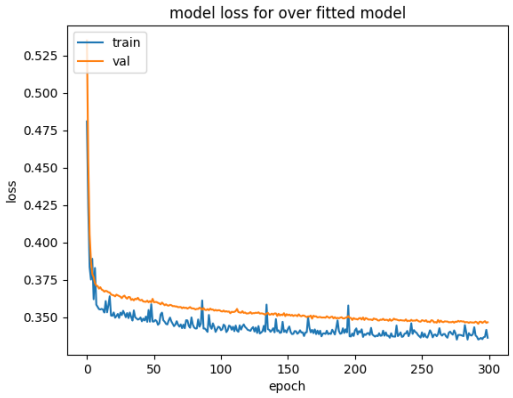
Generalization در مقابل overfitting تعریف می‌شود بدین معنی که اگر یک مدل تعمیم پذیری خوبی داشته باشد مدلی است که overfit یا over training بر روی آن صورت نگرفته است و می‌توان از راه حل هایی که برای رفع overfitting در بالا گفته شد برای بهبود تعمیم پذیری استفاده نمود مثل استفاده از پیچیدگی مدل و تعداد لایه های مناسب نه زیاد، ایجاد یا استفاده از تعداد داده های مناسب و عدم استفاده از تعداد epoch زیاد بر روی داده های آموزش (توقف آموزش در اولین نقطه مناسب)

برای ایجاد یک شبکه تعمیم پذیر باید از ساده ترین مدل ممکن که عملکرد قابل قبول برای ما دارد استفاده نمود و این عملکرد قابل قبول به معنی معیار دقت بالا و تفاوت نه چندان زیاد دقت در آموزش و تست می‌باشد.

با توجه به تعابیر بالا مدلی که قسمت انتهای بخش ۴ معرفی شد با batch_size = 128 در epoch =300 می‌تواند یک نمونه خوب برای مدلی با ویژگی تعمیم پذیری باشد. زیرا از ساده ترین مدل هایی است که عملکرد قابل قبول برای ما ارائه می کند.








```
*************
accuracy  : 0.8409172083720265
Confusion_matrix:  tf.Tensor(
[[10385   521]
 [ 1706  1387]], shape=(2, 2), dtype=int32)
***********
```