نام درس : شبکه های عصبی

نام استاد : دکتر صفابخش

نام و شماره دانشجویی : فاطمه توکلی ، ۴۰۰۱۳۱۰۱۶

تمرین شماره ۱

۱.

پنج نمونه اولیه از داده به صورت زیر است:

index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	b	30.83	0.0	u	g	w	v	1.25	t	t	1	f	g	00202	0	+
1	a	58.67	4.46	u	g	q	h	3.04	t	t	6	f	g	00043	560	+
2	a	24.50	0.5	u	g	q	h	1.5	t	f	0	f	g	00280	824	+
3	b	27.83	1.54	u	g	w	v	3.75	t	t	5	t	g	00100	3	+
4	b	20.17	5.625	u	g	w	v	1.71	t	f	0	f	s	00120	0	+

ابعاد مجموعه داده به صورت:

```
ds.shape
```

```
(690, 16)
```

مجموعه داده مقدار شامل مقدار null نمی باشد اما زمانیکه مقادیر متمایز هر ستون را میگیریم در برخی از ستون ها ? نیز وجود دارد که با داده با بیشترین تکرار در هر ستون جایگذاری می شود:

```
for i in range(0,16):
  mode = ds[ds[i]!="?"][i].mode()[0]
  ds[i] = ds[i].replace("?", mode)
```

برخی ستون ها دارای مقدار string هستند که باید تبدیل به int /float شوند تا بتوان به عنوان ورودی به شبکه عصبی داد برای محاسبات :

روش زیر مشابه روش Labelencoder() است و به ازای مقادیر یکتای هر ستون به آن ها مقدار عددی می دهد

```
#change string value to numeric value for each cloumn
lst_s_to_n = [0,3,4,5,6,8,9,11,12,15]
for i in lst_s_to_n:
  ds[i] = ds[i].astype('category').cat.rename_categories(range(0, ds[i].nunique()))
```

ستون هایی که مقدار [0,1] دارند نیاز به scale ندارند اما بقیه ستون ها scale میکنیم:

```
lst_scale = [1,2,3,4,5,6,7,10,12,13,14]
for i in lst_scale:
  ds[i]=ds[i]/(max(ds[i])-min(ds[i]))
```

پنج داده اول مجموعه بعد از اعمال تغییرات مورد نیاز به صورت زیر در می آیند:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1.0	0.463609	0.000000	0.5	0.0	0.923077	0.875	0.043860	1.0	1.0	0.014925	0.0	0.0	0.1010	0.00000	0.0
1	0.0	0.882256	0.159286	0.5	0.0	0.769231	0.375	0.106667	1.0	1.0	0.089552	0.0	0.0	0.0215	0.00560	0.0
2	0.0	0.368421	0.017857	0.5	0.0	0.769231	0.375	0.052632	1.0	0.0	0.000000	0.0	0.0	0.1400	0.00824	0.0
3	1.0	0.418496	0.055000	0.5	0.0	0.923077	0.875	0.131579	1.0	1.0	0.074627	1.0	0.0	0.0500	0.00003	0.0
4	1.0	0.303308	0.200893	0.5	0.0	0.923077	0.875	0.060000	1.0	0.0	0.000000	0.0	1.0	0.0600	0.00000	0.0

.۲

میدانیم ستون اخرlabel است و مابقی ستون ها ویژگی هستند

طی دو مرحله از متد train_test_split که از کتابخانه sklearn.model_selection است استفاده کردیم که بتوانیم به test,train, validation تقسیم کنیم :

اندازه هر کدام از مجموعه ها به صورت زیر است:

```
X = ds.iloc[:,0:15]
y = ds[15]
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=22)
X_test , X_val, y_test, y_val =train_test_split(X_test,y_test,test_size=0.33,random_state=22)
```

```
X_train.shape
```

```
(483, 15)
```

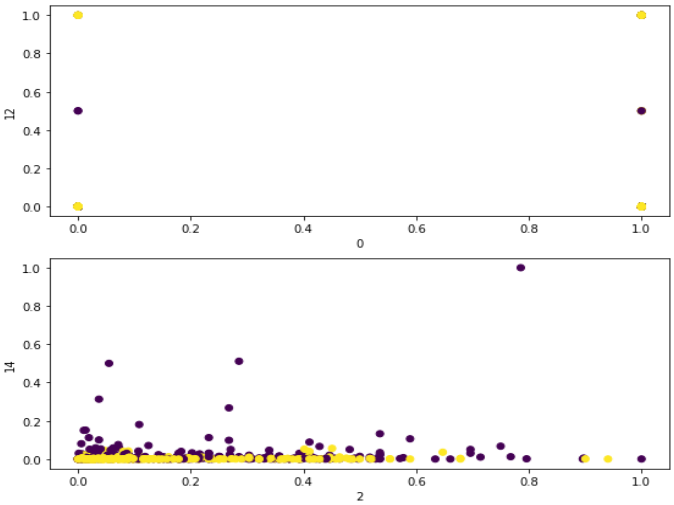
```
X_test.shape
```

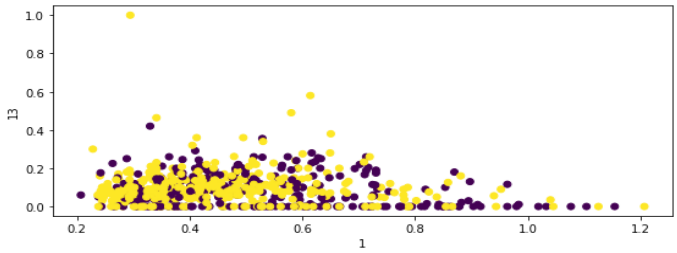
```
(138, 15)
```

```
X_val.shape
```

```
(69, 15)
```

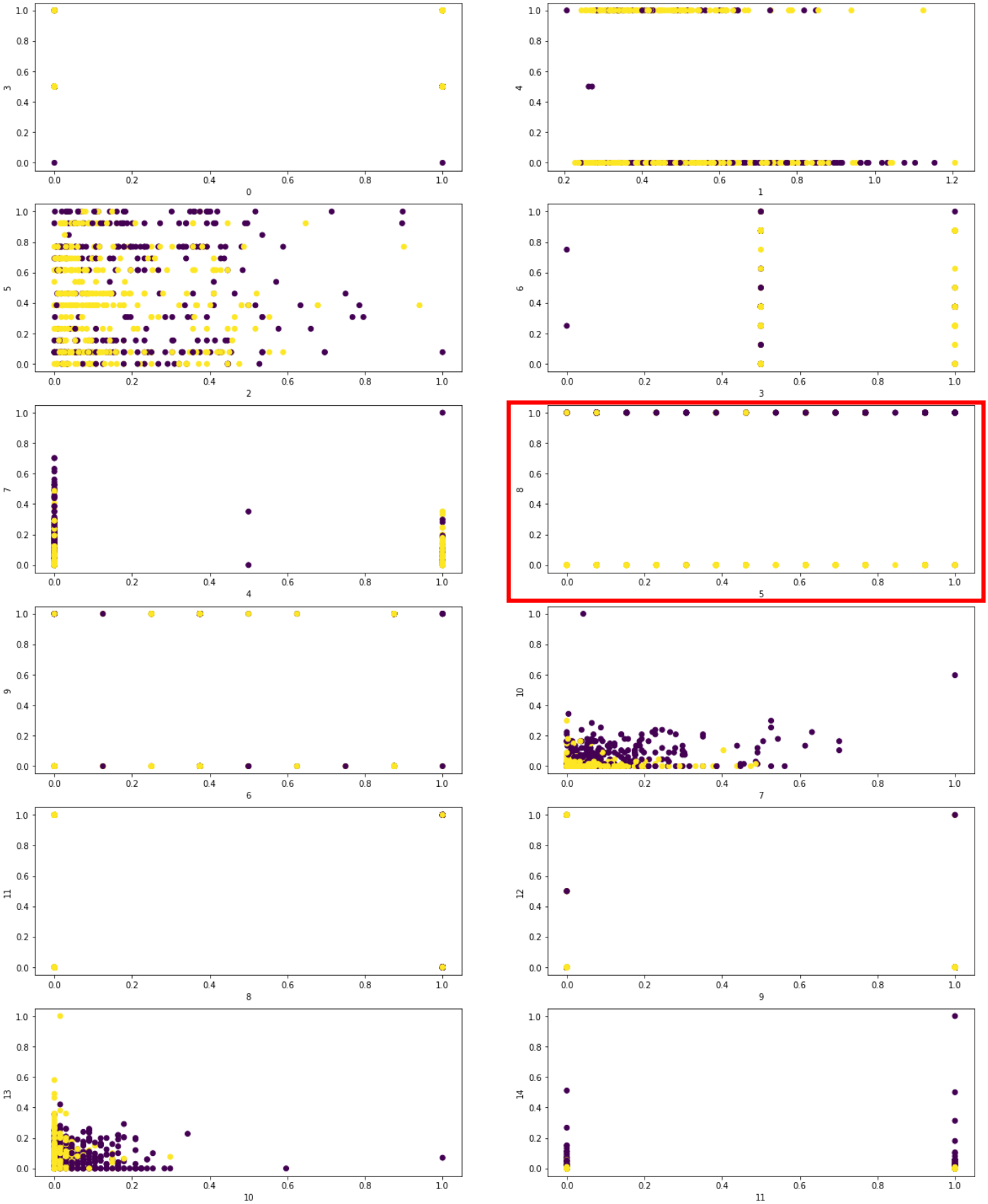
تمامی ستون ها را دو به دو بر اساس label آن ها رسم کرده و به نظر می رسد ستون ۸ شامل مقادیر متمایز کننده ایی است در ترکیب با ستون های دیگر دارد که در شکل ها مشخص شده است(در برخی اندکی خطا در جدا پذیری وجود دارد)

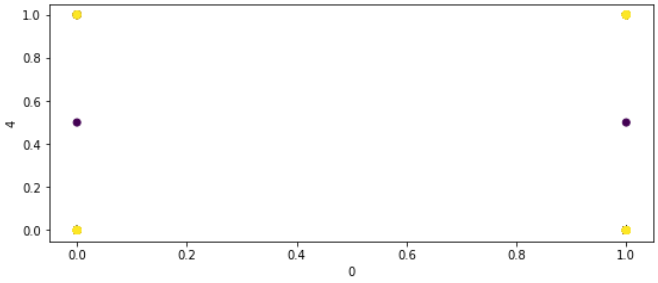


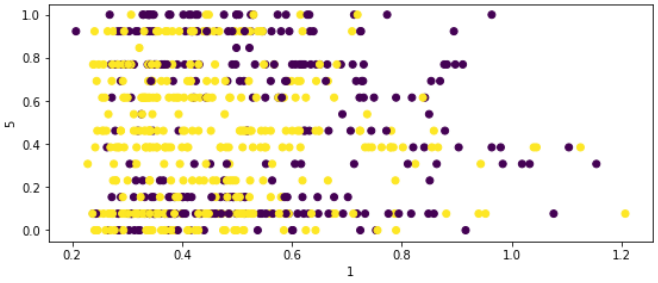


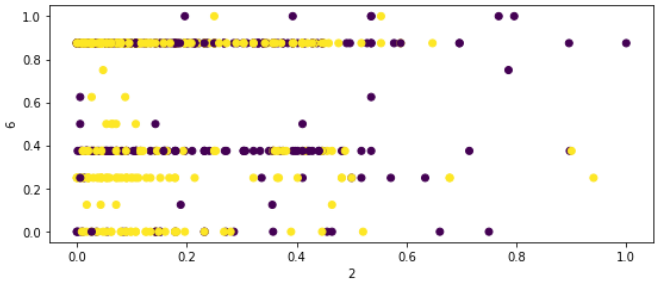

0
1
2
3
4
5
6
7
8
9
10
11
12
13
14

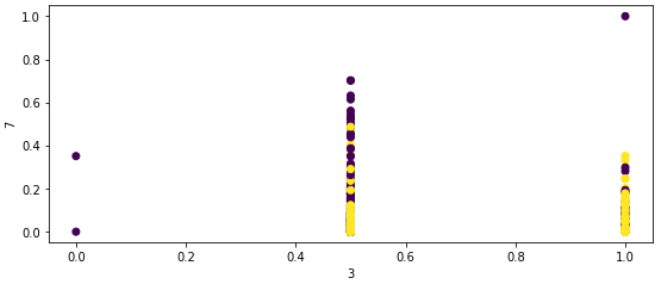

2
0
3
1
4
2
5
3
6
4
7
5
8
6
9
7
10
8
11
9
12
10
13
11
14
12

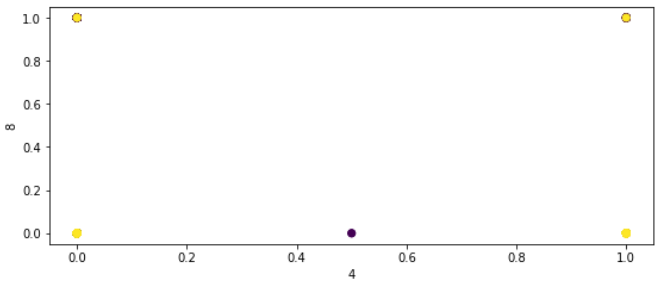

3
0
4
1
5
2
6
3
7
4
8
5
9
6
10
7
11
8
12
9
13
10
14
11

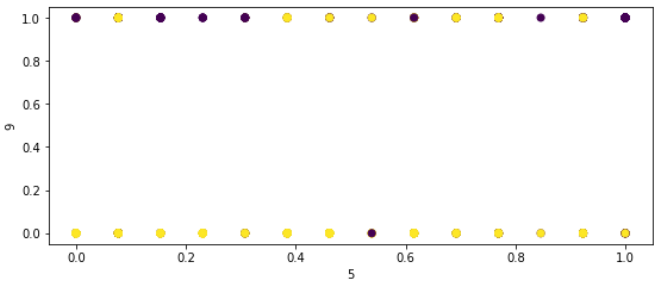

4
0

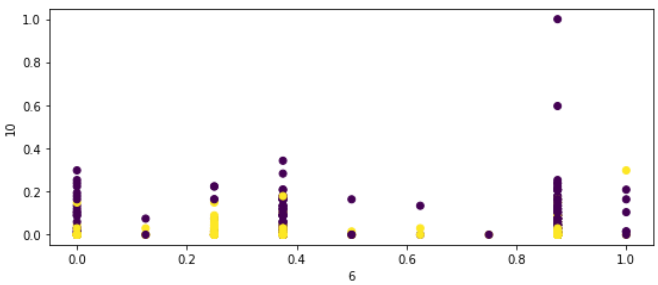

5
1

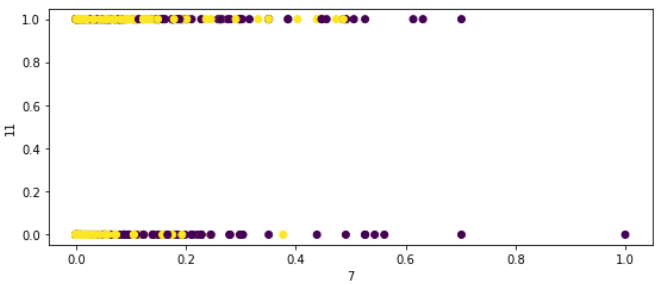

6
2

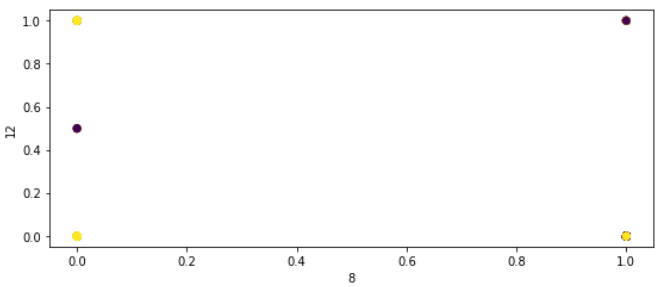

7
3

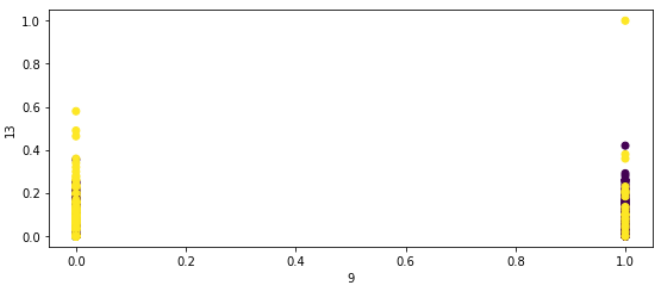

8
4

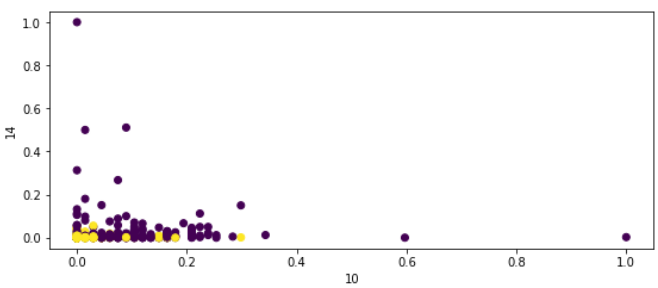

9
5

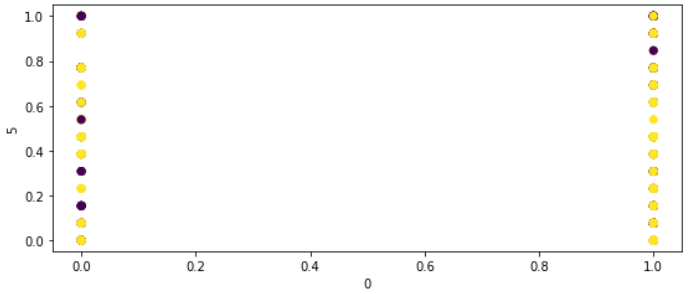

10
6

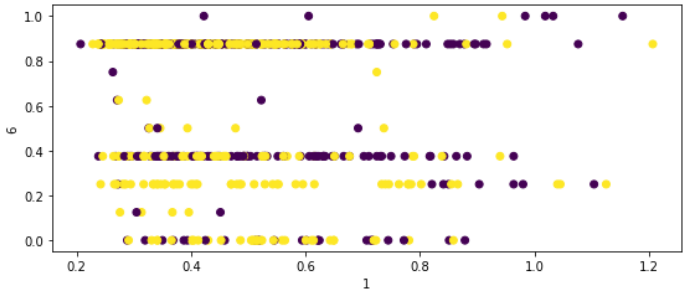

11
7

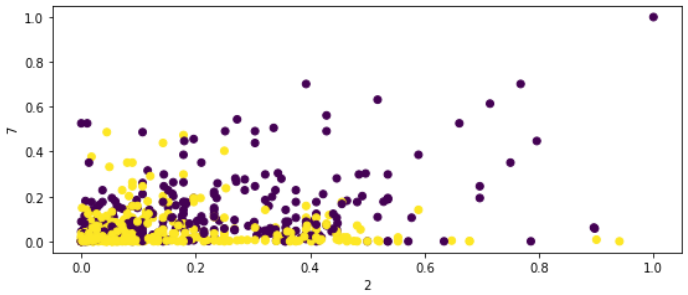

12
8

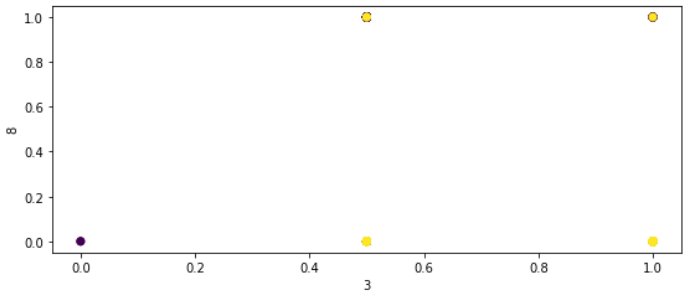

13
9

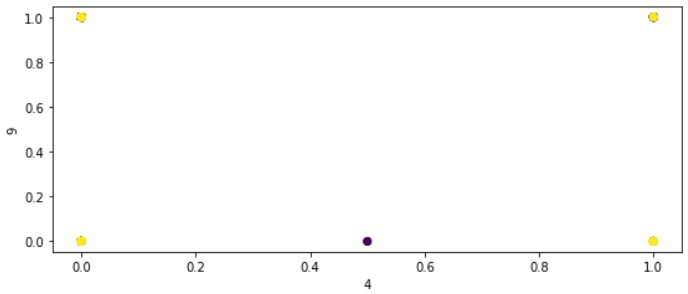

14
10

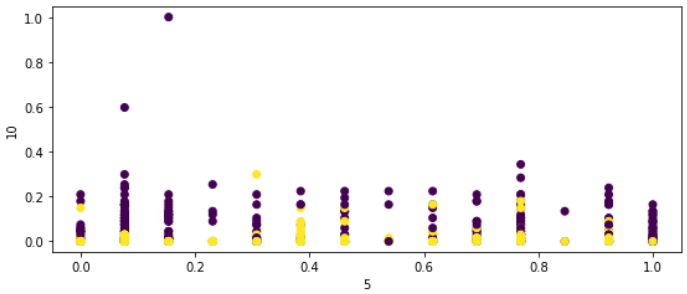

5
0

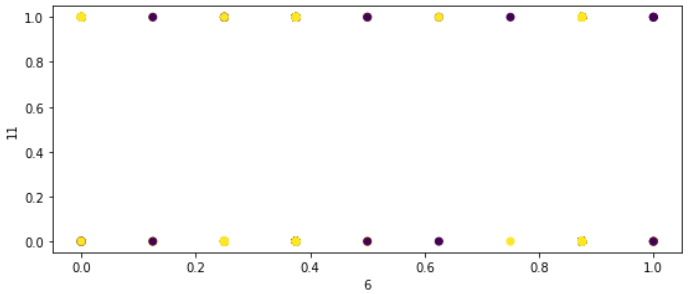

6
1

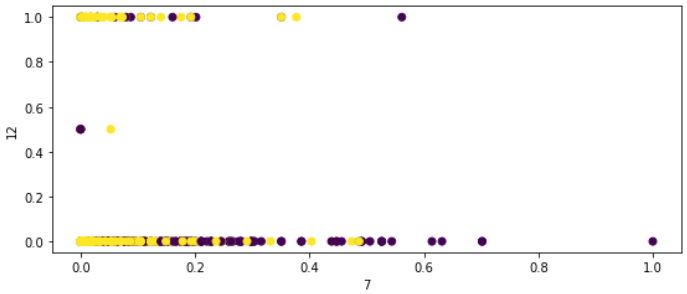

7
2

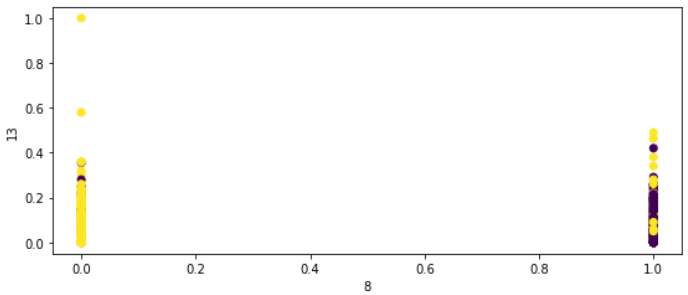

8
3

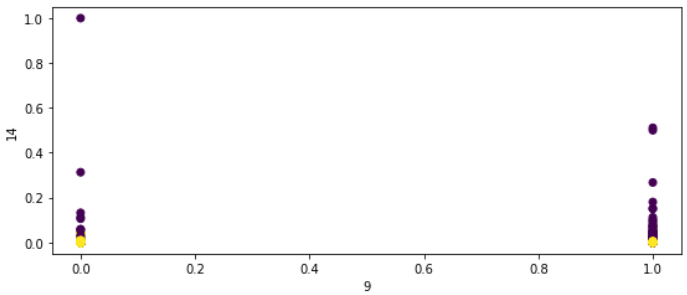

9
4

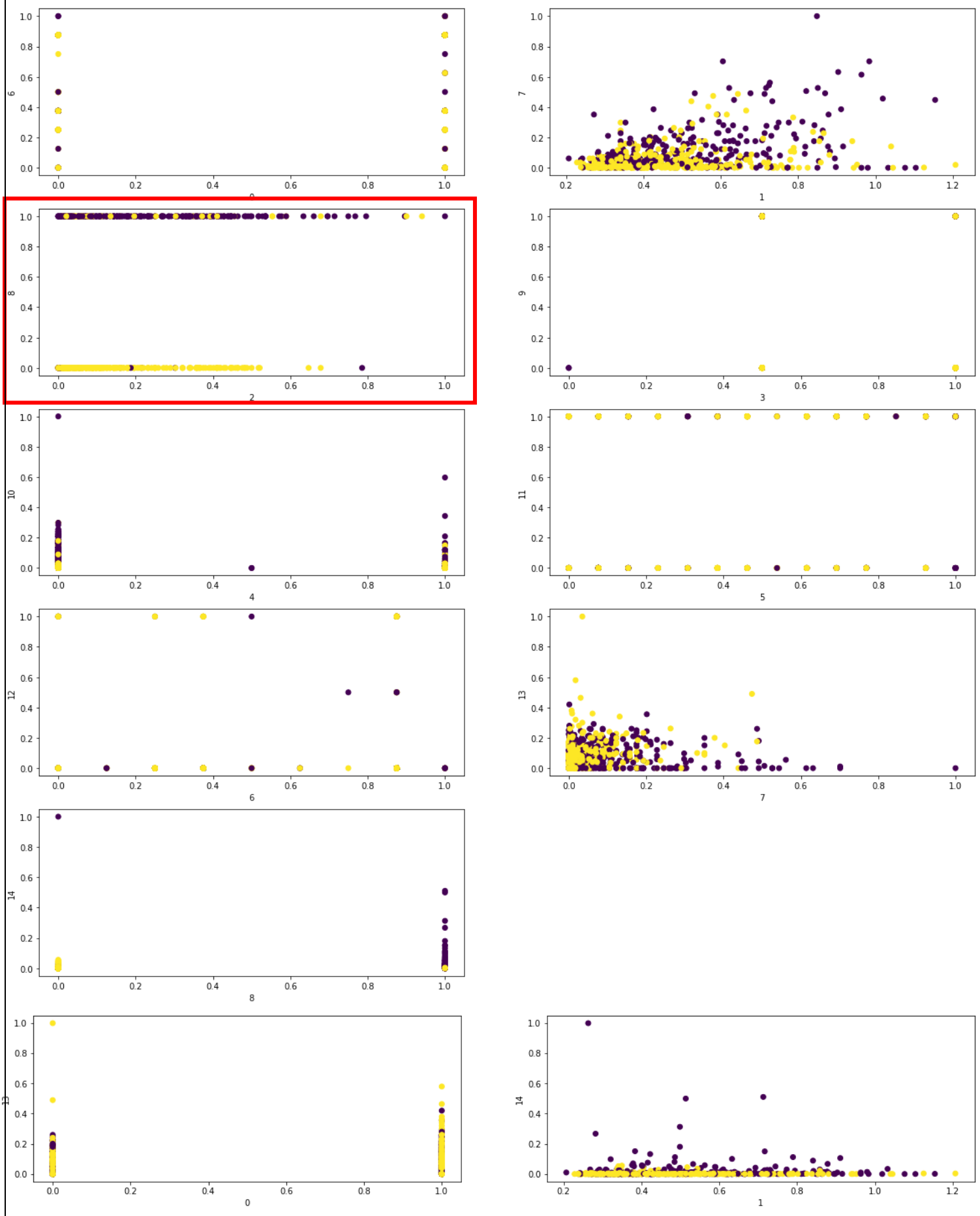

10
5


11
6

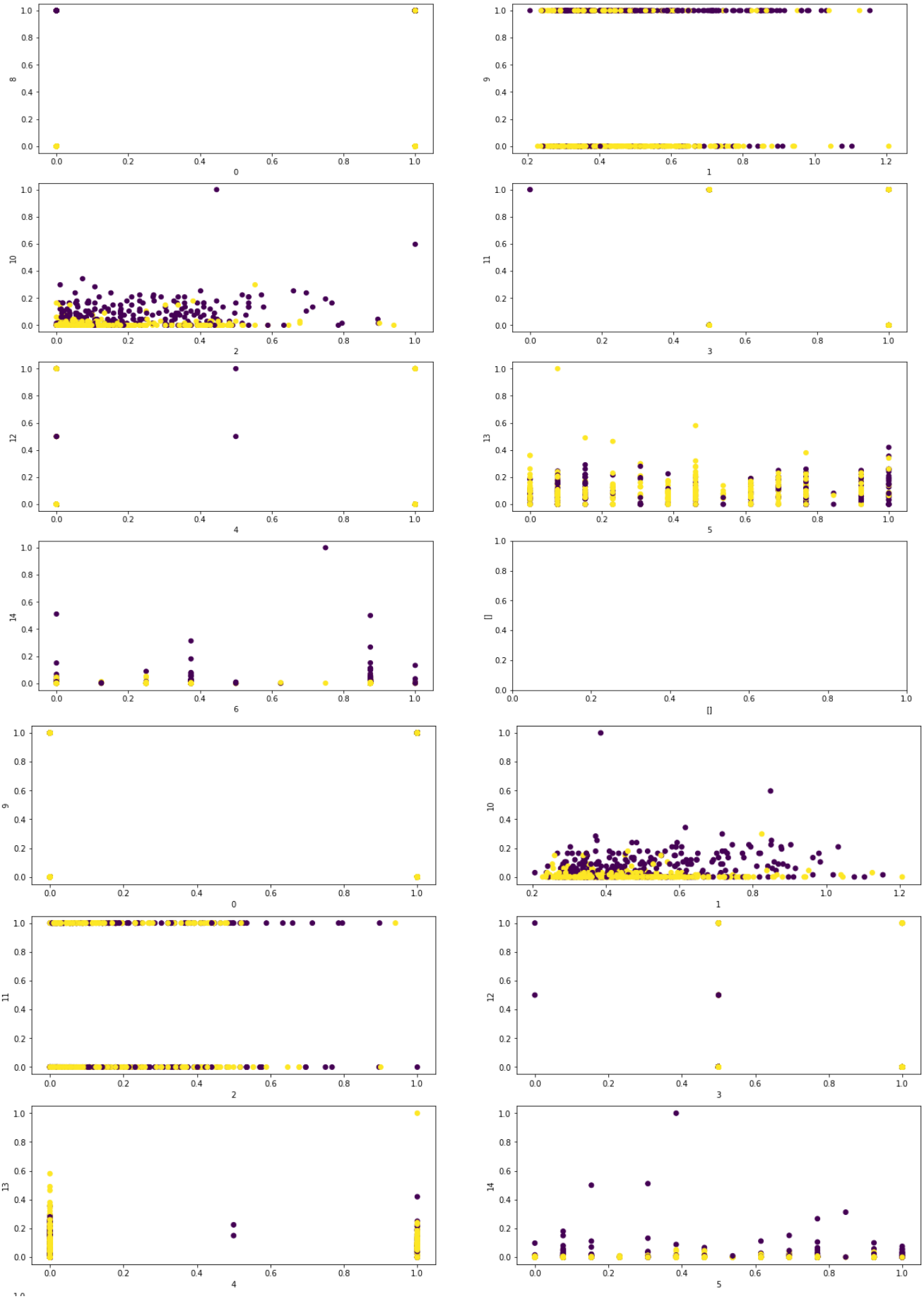

12
7

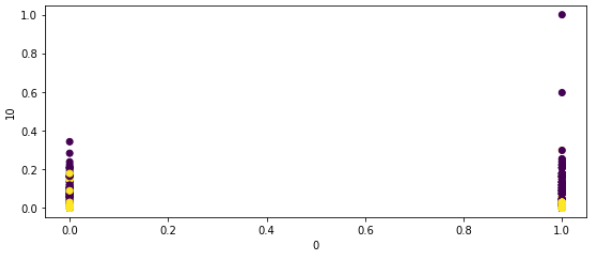

13
8

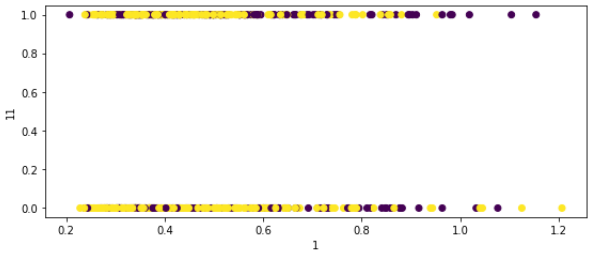

14
9

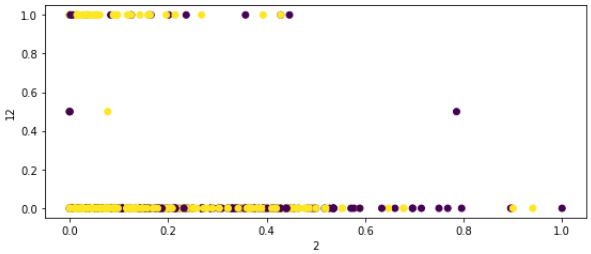

6
0
7
1
8
2
9
3
10
4
11
5
12
6
13
7
14
8
13
0
14
1

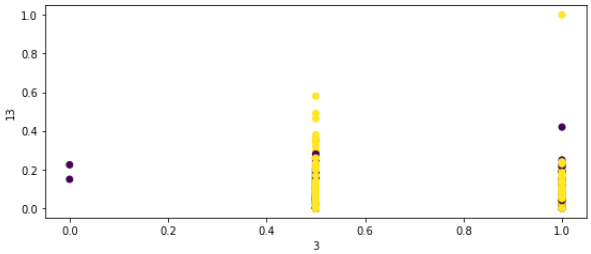

7
0
8
1
9
2
10
3
11
4
12
5
13
6
14
7
11
0
12
1
13
2
14
3

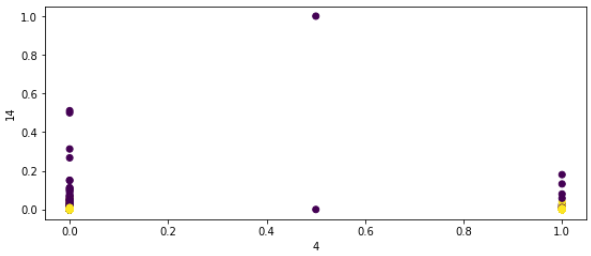

8
0
9
1
10
2
11
3
12
4
13
5
14
6
[]
[]
9
0
10
1
11
2
12
3
13
4
14
5


10
0
0.0
0.2
0.4
0.6
0.8
1.0

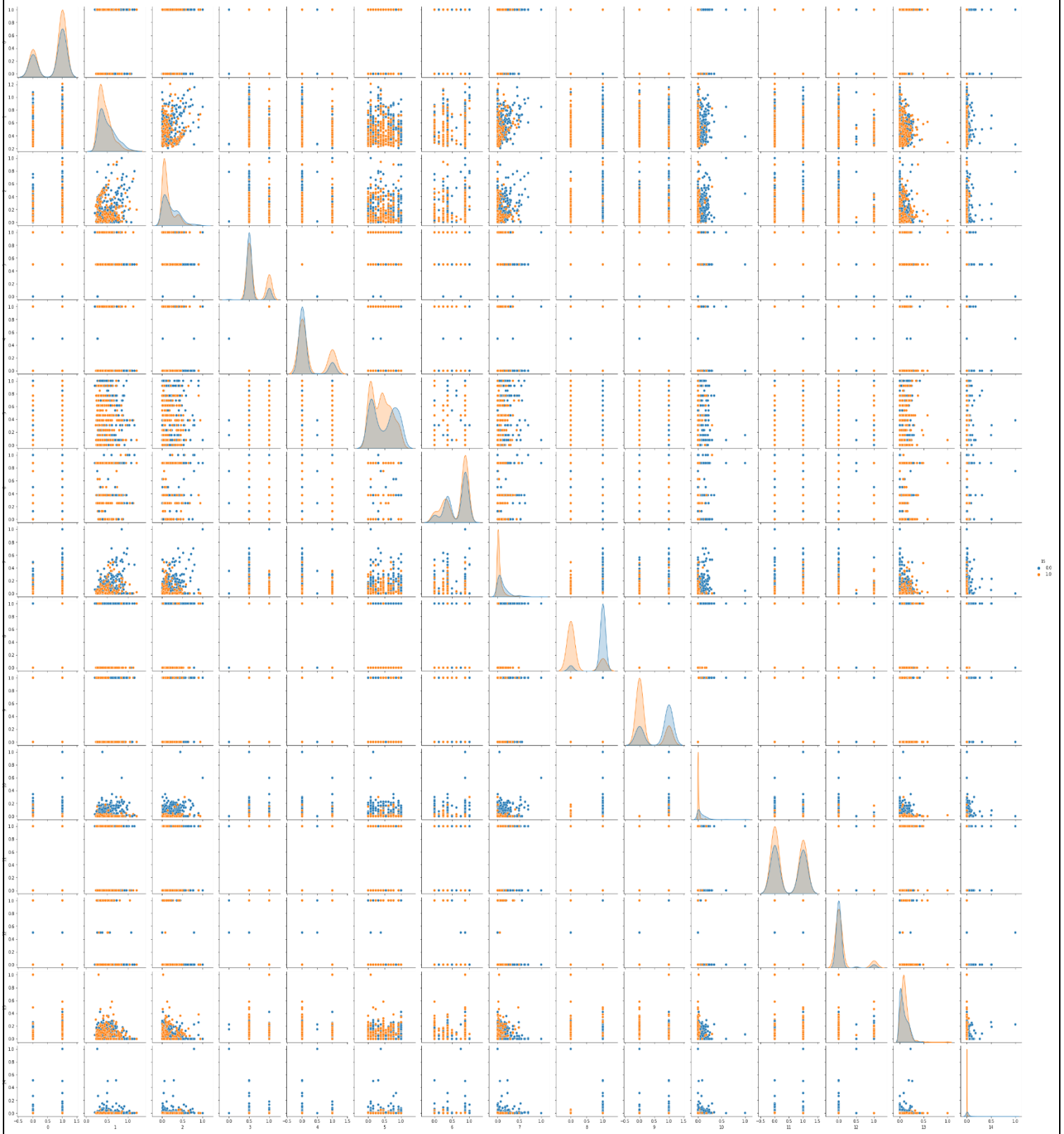

11
1
0.2
0.4
0.6
0.8
1.0
1.2


12
2
0.0
0.2
0.4
0.6
0.8
1.0


13
3
0.0
0.2
0.4
0.6
0.8
1.0


14
4
0.0
0.2
0.4
0.6
0.8
1.0

این شکل نیز توسط sns.pairplot برای کل داده ها کشیده شده است و همانطور که پیداست در بعضی قسمت ها داده ما به صورت خطی جدا پذیر است:



۳.

مدل:

مدل‌های Keras این مدل‌های شبکه عصبی واقعی را نشان می‌دهند. این مدل ها لایه ها را در اشیا گروه بندی می کنند. دو نوع مدل در Keras موجود است: مدل ترتیبی و مدل عملکردی.

مدل Keras Sequential : این یک پشته خطی از متدها است که یک پشته خطی از لایه ها را در یک tf.keras.Model گروه بندی می کند. طبق نامش، وظیفه اصلی آن چیدمان لایه‌های کراس به ترتیب متوالی است.در این مدل، داده ها از یک لایه به لایه دیگر منتقل می شوند. جریان داده ها تا رسیدن به لایه نهایی ادامه می یابد. اکثر ANN ها از مدل API متوالی استفاده می کنند.

Keras.layers.Dense یک لایه شبکه عصبی متصل منظم است. لایه متراکم عملیات زیر را روی ورودی انجام می دهد و خروجی را برمی گرداند.

output = activation(dot(input, kernel) + bias)

```
model = keras.models.Sequential([
                                 keras.layers.Dense(units=132,activation='relu'),
                                 keras.layers.Dense(units =10, activation = 'relu'),
                                 keras.layers.Dense(2, activation = 'softmax')

                                ])
```

Adamیک الگوریتم بهینه سازی جایگزین برای نزول گرادیان تصادفی برای آموزش مدل های یادگیری عمیق است Adam .بهترین ویژگی‌های الگوریتم‌های AdaGrad و RMSProp را برای ارائه یک الگوریتم بهینه‌سازی که می‌تواند شیب‌های پراکنده را در مسائل نویزدار مدیریت کند، ترکیب می‌کند.

Sparse_categorical_crossentropy آموزش یک شبکه عصبی pass data forward،از طریق مدل، و مقایسه پیش بینی ها با برچسب ها است. این مقایسه توسط یک تابع loss انجام می شود.

```
model.compile(optimizer='adam', loss = 'sparse_categorical_crossentropy',metrics=['accuracy'])
```

```
model.fit(X_train,y_train,epochs=10)
```

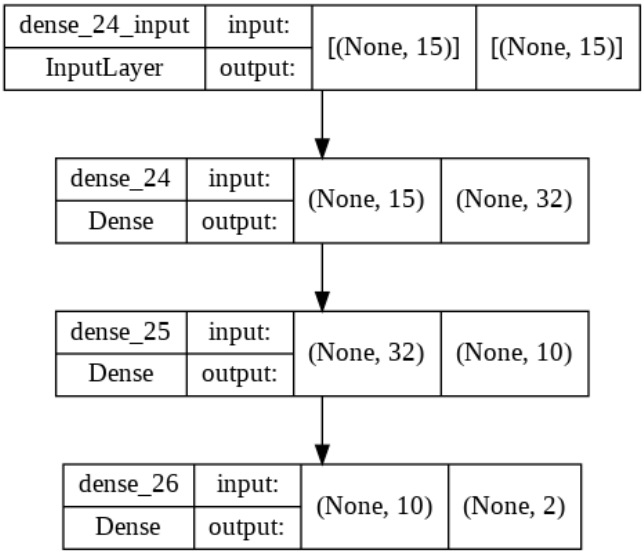
```
Epoch 1/10
16/16 [==============================] - 0s 2ms/step - loss: 0.6858 - accuracy: 0.5197
Epoch 2/10
16/16 [==============================] - 0s 2ms/step - loss: 0.5702 - accuracy: 0.7971
Epoch 3/10
16/16 [==============================] - 0s 2ms/step - loss: 0.4926 - accuracy: 0.8199
Epoch 4/10
16/16 [==============================] - 0s 2ms/step - loss: 0.4360 - accuracy: 0.8509
Epoch 5/10
16/16 [==============================] - 0s 2ms/step - loss: 0.4048 - accuracy: 0.8613
Epoch 6/10
16/16 [==============================] - 0s 2ms/step - loss: 0.3858 - accuracy: 0.8613
Epoch 7/10
16/16 [==============================] - 0s 2ms/step - loss: 0.3767 - accuracy: 0.8634
Epoch 8/10
16/16 [==============================] - 0s 2ms/step - loss: 0.3699 - accuracy: 0.8509
Epoch 9/10
16/16 [==============================] - 0s 2ms/step - loss: 0.3627 - accuracy: 0.8613
Epoch 10/10
16/16 [==============================] - 0s 2ms/step - loss: 0.3623 - accuracy: 0.8613
```

دقت مدل بر روی داده تست به صورت زیر است:

```
test_loss,test_accuracy = model.evaluate(X_test,y_test, verbose=2)
```

```
5/5 - 0s - loss: 0.3041 - accuracy: 0.8986 - 19ms/epoch - 4ms/step
```

گراف مدل:



۴. با ساختار سه لایه به صورت زیر شروع کردیم:

```
model = keras.models.Sequential([
                                 keras.layers.Dense(units=128,activation='relu'),
                                 keras.layers.Dense(units =20, activation = 'relu'),
                                 keras.layers.Dense(2, activation = 'softmax')

                                ])
```

```
model.compile(optimizer='adam', loss = 'sparse_categorical_crossentropy',metrics=['accuracy'])
```

```
model.fit(X_train,y_train,epochs=10)
```

```
Epoch 1/10
16/16 [==============================] - 0s 1ms/step - loss: 0.6169 - accuracy: 0.7578
Epoch 2/10
16/16 [==============================] - 0s 2ms/step - loss: 0.5093 - accuracy: 0.8406
Epoch 3/10
16/16 [==============================] - 0s 1ms/step - loss: 0.4404 - accuracy: 0.8489
Epoch 4/10
16/16 [==============================] - 0s 1ms/step - loss: 0.3928 - accuracy: 0.8592
Epoch 5/10
16/16 [==============================] - 0s 1ms/step - loss: 0.3708 - accuracy: 0.8634
Epoch 6/10
16/16 [==============================] - 0s 2ms/step - loss: 0.3685 - accuracy: 0.8509
Epoch 7/10
16/16 [==============================] - 0s 2ms/step - loss: 0.3561 - accuracy: 0.8592
Epoch 8/10
16/16 [==============================] - 0s 2ms/step - loss: 0.3486 - accuracy: 0.8634
Epoch 9/10
16/16 [==============================] - 0s 2ms/step - loss: 0.3557 - accuracy: 0.8551
Epoch 10/10
16/16 [==============================] - 0s 2ms/step - loss: 0.3441 - accuracy: 0.8634
<keras.callbacks.History at 0x7f72c6e416d0>
```

سپس دیدیم با تعداد نورون کمتر به نتایج نزدیک به قبل میرسیم:

```
model = keras.models.Sequential([
                                    keras.layers.Dense(units=20,activation='relu'),
                                    keras.layers.Dense(units =5, activation = 'relu'),
                                    keras.layers.Dense(2, activation = 'softmax')

                                   ])
```

```
model.compile(optimizer='adam', loss = 'sparse_categorical_crossentropy',metrics=['accuracy'])
```

```
model.fit(X_train,y_train,epochs=10)
```

```
Epoch 1/10
16/16 [==============================] - 0s 2ms/step - loss: 0.7294 - accuracy: 0.3913
Epoch 2/10
16/16 [==============================] - 0s 1ms/step - loss: 0.6878 - accuracy: 0.5135
Epoch 3/10
16/16 [==============================] - 0s 2ms/step - loss: 0.6685 - accuracy: 0.5569
Epoch 4/10
16/16 [==============================] - 0s 2ms/step - loss: 0.6504 - accuracy: 0.6025
Epoch 5/10
16/16 [==============================] - 0s 2ms/step - loss: 0.6340 - accuracy: 0.6273
Epoch 6/10
16/16 [==============================] - 0s 2ms/step - loss: 0.6181 - accuracy: 0.6480
Epoch 7/10
16/16 [==============================] - 0s 1ms/step - loss: 0.5998 - accuracy: 0.6998
Epoch 8/10
16/16 [==============================] - 0s 2ms/step - loss: 0.5779 - accuracy: 0.7412
Epoch 9/10
16/16 [==============================] - 0s 2ms/step - loss: 0.5542 - accuracy: 0.7785
Epoch 10/10
16/16 [==============================] - 0s 1ms/step - loss: 0.5244 - accuracy: 0.8240
<keras.callbacks.History at 0x7f72c6c3d0d0>
```

لایه اخر را تغییر داده و optimizer را از نوع object کلاس adam با ضریب یادگیری ۰/۰۰۰۱ و loss را از نوع objectکلاس sparseدر آوردیم و مقدار from_logits = True قرار دادیم تا متوجه شود ورودی ما احتمال نیست:

```
model = keras.models.Sequential([
                                  keras.layers.Dense(units=30,activation='relu'),
                                  keras.layers.Dense(units =15, activation = 'relu'),
                                  keras.layers.Dense(2)

                                 ])
```

```
model.compile(
              optimizer = keras.optimizers.Adam(learning_rate= 0.00001),
              loss = keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics = ['accuracy'])
```

```
model.fit(X_train,y_train,epochs=10)
```

```
Epoch 1/10
16/16 [==============================] - 0s 1ms/step - loss: 0.6942 - accuracy: 0.4948
Epoch 2/10
16/16 [==============================] - 0s 2ms/step - loss: 0.6935 - accuracy: 0.4990
Epoch 3/10
16/16 [==============================] - 0s 2ms/step - loss: 0.6927 - accuracy: 0.5010
Epoch 4/10
16/16 [==============================] - 0s 2ms/step - loss: 0.6919 - accuracy: 0.5010
Epoch 5/10
16/16 [==============================] - 0s 2ms/step - loss: 0.6912 - accuracy: 0.5052
Epoch 6/10
16/16 [==============================] - 0s 2ms/step - loss: 0.6905 - accuracy: 0.5072
Epoch 7/10
16/16 [==============================] - 0s 2ms/step - loss: 0.6898 - accuracy: 0.5093
Epoch 8/10
16/16 [==============================] - 0s 2ms/step - loss: 0.6891 - accuracy: 0.5093
Epoch 9/10
16/16 [==============================] - 0s 2ms/step - loss: 0.6884 - accuracy: 0.5114
Epoch 10/10
16/16 [==============================] - 0s 1ms/step - loss: 0.6877 - accuracy: 0.5155
<keras.callbacks.History at 0x7f72b66cdf50>
```

```
model = keras.models.Sequential([
                                    keras.layers.Dense(units=30,activation='relu'),
                                    keras.layers.Dense(units =15, activation = 'relu'),
                                    keras.layers.Dense(2, activation='softmax')

                                   ])
```

```
model.compile(
               optimizer = keras.optimizers.Adam(learning_rate= 0.00001),
               loss = keras.losses.SparseCategoricalCrossentropy(),
               metrics = ['accuracy'])
```

```
model.fit(X_train,y_train,epochs=10)
```

```
Epoch 1/10
16/16 [==============================] - 0s 1ms/step - loss: 0.6993 - accuracy: 0.4596
Epoch 2/10
16/16 [==============================] - 0s 1ms/step - loss: 0.6989 - accuracy: 0.4617
Epoch 3/10
16/16 [==============================] - 0s 1ms/step - loss: 0.6985 - accuracy: 0.4638
Epoch 4/10
16/16 [==============================] - 0s 2ms/step - loss: 0.6980 - accuracy: 0.4638
Epoch 5/10
16/16 [==============================] - 0s 2ms/step - loss: 0.6976 - accuracy: 0.4679
Epoch 6/10
16/16 [==============================] - 0s 2ms/step - loss: 0.6972 - accuracy: 0.4679
Epoch 7/10
16/16 [==============================] - 0s 2ms/step - loss: 0.6968 - accuracy: 0.4741
Epoch 8/10
16/16 [==============================] - 0s 2ms/step - loss: 0.6964 - accuracy: 0.4803
Epoch 9/10
16/16 [==============================] - 0s 2ms/step - loss: 0.6960 - accuracy: 0.4803
Epoch 10/10
16/16 [==============================] - 0s 1ms/step - loss: 0.6956 - accuracy: 0.4886
<keras.callbacks.History at 0x7f72b6735190>
```

اضافه کردن و کم کردن لایه های و مقدار نورون هر لایه با اعداد مختلف تست شد و بهترین حالت در صفحه بعد آورده شده است:

Best one:

```
model = keras.models.Sequential([
                                    keras.layers.Dense(units=30,activation='relu'),
                                    keras.layers.Dense(units =15, activation = 'relu'),
                                    keras.layers.Dense(2, activation='softmax')

                                   ])
```

```
model.compile(optimizer='adam', loss = 'sparse_categorical_crossentropy',metrics=['accuracy'])
```

```
model.fit(X_train,y_train,epochs=10)
```

```
Epoch 1/10
16/16 [==============================] - 0s 2ms/step - loss: 0.7198 - accuracy: 0.5694
Epoch 2/10
16/16 [==============================] - 0s 1ms/step - loss: 0.6471 - accuracy: 0.6667
Epoch 3/10
16/16 [==============================] - 0s 1ms/step - loss: 0.6058 - accuracy: 0.7950
Epoch 4/10
16/16 [==============================] - 0s 1ms/step - loss: 0.5685 - accuracy: 0.8178
Epoch 5/10
16/16 [==============================] - 0s 1ms/step - loss: 0.5271 - accuracy: 0.8302
Epoch 6/10
16/16 [==============================] - 0s 1ms/step - loss: 0.4836 - accuracy: 0.8344
Epoch 7/10
16/16 [==============================] - 0s 1ms/step - loss: 0.4441 - accuracy: 0.8406
Epoch 8/10
16/16 [==============================] - 0s 1ms/step - loss: 0.4127 - accuracy: 0.8634
Epoch 9/10
16/16 [==============================] - 0s 1ms/step - loss: 0.3913 - accuracy: 0.8696
Epoch 10/10
16/16 [==============================] - 0s 1ms/step - loss: 0.3784 - accuracy: 0.8737
<keras.callbacks.History at 0x7f72b37cc990>
```

If change epoches = 35 :

```
Epoch 35/35
16/16 [==============================] - 0s 2ms/step - loss: 0.2276 - accuracy: 0.9110
```

دقت ما بر روی مجموعه تست به صورت زیر است:

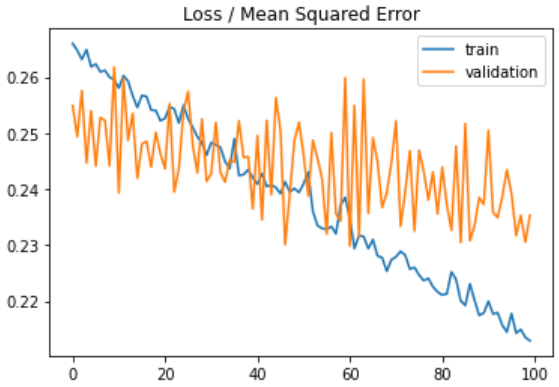
```
test_loss,test_accuracy = model.evaluate(X_test,y_test,verbose=2)
```

```
5/5 - 0s - loss: 0.3208 - accuracy: 0.9130 - 20ms/epoch - 4ms/step
```

5.

- یکی از راه این است که تعداد epochs را زیاد کنیم و تفاوت loss داده آموزش و داده تست را بیابیم:

من در این قسمت epochs =100 قرار داده و loss هردو را رسم کرده :



همانطور که مشاهده می شود بعد از ۶۰ بار تکرار آموزش خطای داده آموزش کم شده ولی خطای مجموعه اعتبارسنجی همچنان ثابت مانده(اگر بالا وپایین شدن های ناگهانی نادیده بگیریم)است

- همچنین با پیچیده تر کردن مدل نیز همین اتفاق می افتد ما افزدون لایه ایی جدید مدل را مجددا fit کرده:

```
model = keras.models.Sequential([
                                 keras.layers.Dense(units=32,activation='relu'),
                                 keras.layers.Dense(units =10, activation = 'relu'),
                                 keras.layers.Dense(units = 5, activation='relu'),
                                 keras.layers.Dense(2, activation = 'softmax')

                                ])
```

```
model.compile(optimizer='adam', loss = keras.losses.SparseCategoricalCrossentropy(),metrics=['accuracy'])
```

همانطور که میبینید تا تکرار ۸۰ ام به طور همزمان خطای آموزش و اعتبارسنجی کاهش می یابد ولی بعد از آن خطای آموزش کاهش می یابد اما خطای اعتبار سنجی رو به افزایش است.

۶.

در لایه آخر مدل قبل یک پارامتر kernel_regularizer =l2 را اضافه میکنیم که نحوه به روزرسانی وزن ها را تحت تاثیر قرار می دهد:

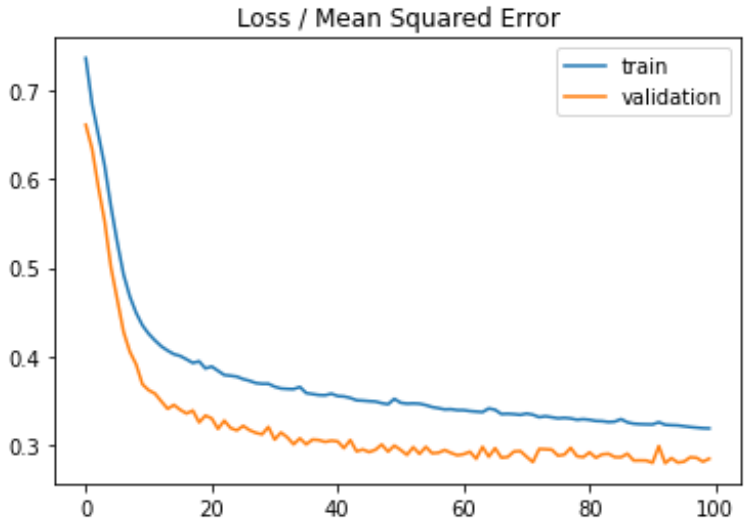
L2 regularization به صورت زیر حساب میشود که به هنگام به روز رسانی وزن ها به آن اضافه می شود:

loss = l2 * reduce_sum(square(x))

```
model = keras.models.Sequential([
                                 keras.layers.Dense(units=32,activation='relu'),
                                 keras.layers.Dense(units =10, activation = 'relu'),
                                 keras.layers.Dense(2, activation = 'softmax',kernel_regularizer='l2')
                                ])
```

```
model.compile(optimizer='adam', loss = keras.losses.SparseCategoricalCrossentropy(),metrics=['accuracy'])
```

```
m2 = model.fit(X_train_n,y_train_n,epochs = 100,validation_split=0.1 )
```



همانطور که مشاهده می شود خطای داده اعتبار سنجی با تغییر اعمال شده همگام با داده آموزش کم شده است