«به نام خدا»

تکلیف پنجم – سوال اول – مرضیه علیدادی – 9631983

(کد های مربوط، در دو فرمت py. و ipynb. ضمیمه شده اند.)

.1

b) در این دیتاســـت missing value ای وجود ندارد. مقادیر Null ای وجود ندارد. همه ی attribute های input از نوع float هستند و نمی توانند دارای مقادیری از نوع string , ... (مثلا 'missed') به مفهوم missing value باشند.

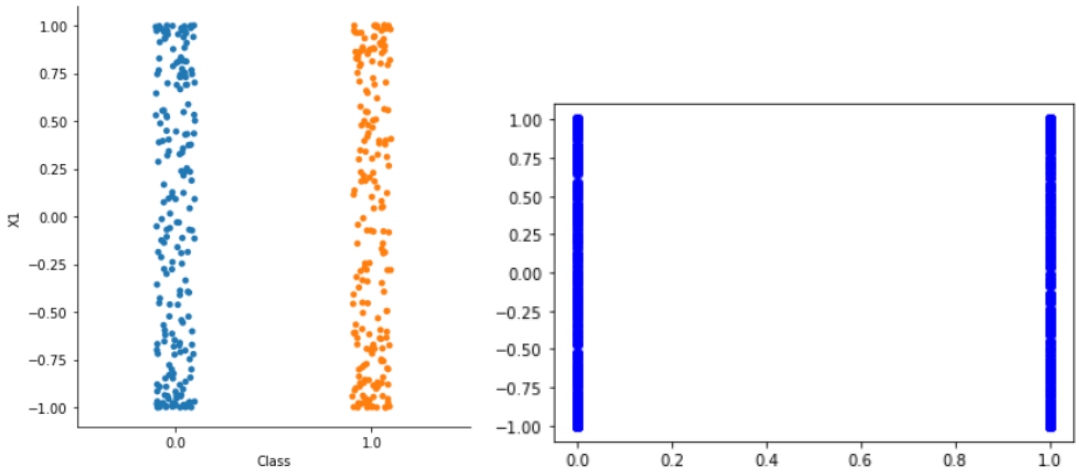
داده های عددی را به این صورت normalize کردم:

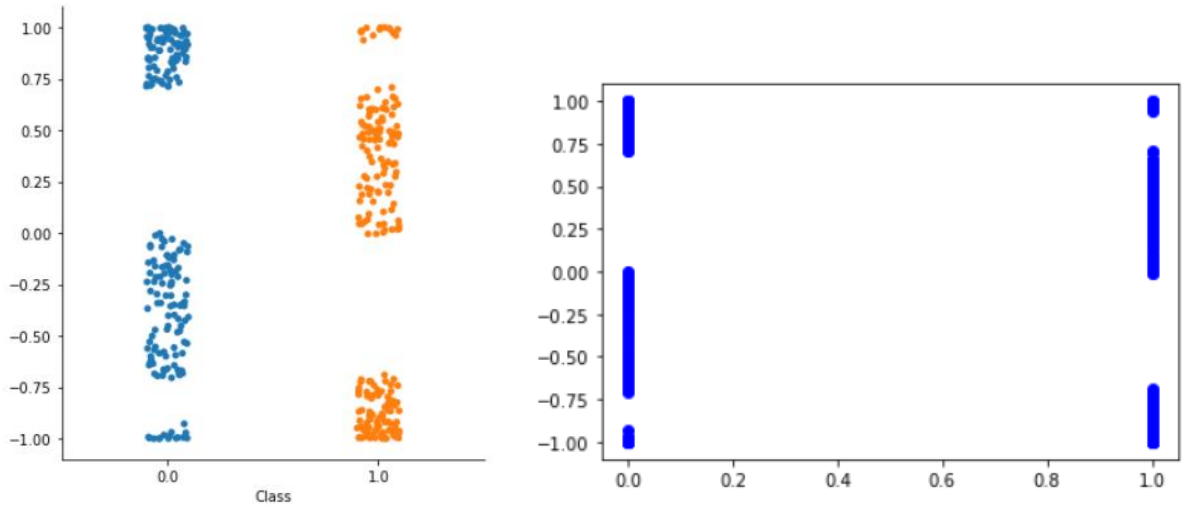
	X1	X2	Class
0	0.319185	0.947693	0
1	0.106471	-0.994316	0
2	0.074209	-0.997243	0
3	0.166776	-0.985995	0
4	0.233616	0.972329	0
...	...	...	...
395	0.183833	0.982957	1
396	0.042103	0.999113	1
397	0.122420	0.992478	1
398	0.151509	-0.988456	1
399	0.231688	0.972790	1

400 rows × 3 columns

برای encode کردن متغیر Class، با توجه به اینکه جز input ها نیست و با توجه به اینکه در تحلیل ها اثر ندارد، حساسیتی روی مقادیری که می گیرد، نداریم. پس از Ordinal encoding ســـاده می توانیم اســـتفاده کنیم. ولی اینجا خودش مقادیرش به صورت عددی است؛ و نیازی به انکد کردن نداریم کلا.

c) پراکندگی هر کدام از دو ستون input این دیتاست، از نظر پراکندگی دسته های Class، به این شکل است:





e) توزیع دسته های ستون Class، به ترتیب در دو مجموعه ی آموزش و تست:

```
y_train.value_counts()
```

```
1    162
0    158
Name: Class, dtype: int64
```

```
y_test.value_counts()
```

```
0    42
1    38
Name: Class, dtype: int64
```

g) با استفاده از قابلیت های sklearn با استفاده از درخت تصمیمی که در بخش قبل تولید کرده بودم، متغیر هدف را برای داده های تست پیشبینی کردم. نتیجه را با مقادیر واقعی متغیر هدف مقایسه کردم. بدین صورت شد:

```
[[33  9]
 [ 2 36]]
              precision    recall  f1-score   support

           0       0.94      0.79      0.86        42
           1       0.80      0.95      0.87        38

    accuracy                           0.86        80
   macro avg       0.87      0.87      0.86        80
weighted avg       0.88      0.86      0.86        80
```

- ماتریس تهیه شده بدین صورت تفسیر می شود:

هر سطر نشان دهنده ی یکی از دسته های واقعی (actual) است. و هر ستون نشاندهنده ی یکی از دسته های پیشبینی شده.

داده هایی که در دسته ی 0 قرار می گیرند، 42 تا هستند. که 33 تا از آن ها درست پیشبینی شده اند.

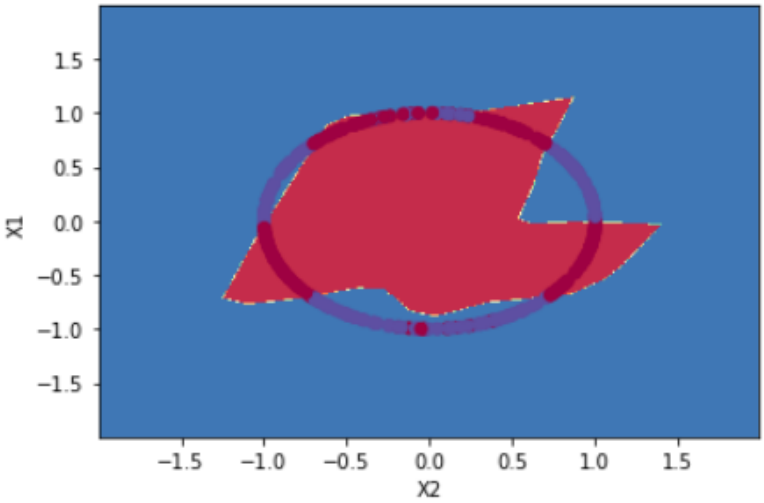
داده هایی که در دسته ی 1 قرار می گیرند، 38 تا هستند. که 36 تا از آن ها درست پیشبینی شده اند.

- گزارشی که در ادامه آمده، همین مطالب را به صورت درصدی بیان می کند:

دقت پیشبینی داده های دسته های 0 و 1، به ترتیب برابر 79% و 95% است.

و به طور کلی، دقت مدل برابر 86% است.

h) مختصات دیتا ها را بدست آورده و آن ها روی نمودار مشخص کرده است. و مقدار پیشبینی شده را هم روی آن رسم کرده است. همانطور که مشخص است، دقت خوبی دارد.



i) بهترین activation همان حالا دیفالت (Relu) اســت. با افزایش تعداد لایه ها و کاهش تعداد نرون های هر لایه، نتیجه ی خوبی حاصل می شود.

در کل، در دو حالت می توان دقت را بالا برد. یکی اینکه تعداد iteration ها را به تدریج زیاد کنیم. و وقتی به حالتی رســیدیم که دقت از آن به بعد کاهش یافت، همان تعداد را در نظر بگیریم. راه دوم که بهتر اســت و دقت بیشــتری حاصــل می شــود این اســت که، با آزمون و خطا به تدریج تعداد لایه ها را افزایش بدهیم و تعداد نرون های هر لایه را کاهش بدهیم، تا در یک آستانه ای به بهترین دقت برسیم.

تمام تســت هایی که گرفتم و ســرچ هایی که کردم، Relu به عنوان بهترین activation انتخاب شــد. پس activation موثر ترین پارامتر است. (اگر منظور از موثر ترین پارامتر، قطعیت در انتخاب مقدار آن باشد.)

ولی اگر موثر ترین را این در نظر بگیریم که بیشــترین تاثیر در افزایش دقت را ایجاد می کند، hidden_layer_sizes موثر ترین است.

```
mlp10 = MLPClassifier(hidden_layer_sizes=(8, 8, 8, 8), max_iter=1300)
mlp10.fit(X_train, y_train)
predictions10 = mlp10.predict(X_test)
print(confusion_matrix(y_test,predictions10))
print(classification_report(y_test,predictions10))
```

```
[[36  6]
 [ 3 35]]
              precision    recall  f1-score   support

           0       0.92      0.86      0.89        42
           1       0.85      0.92      0.89        38

    accuracy                           0.89        80
   macro avg       0.89      0.89      0.89        80
weighted avg       0.89      0.89      0.89        80
```