«به نام خدا»

تکلیف سوم – سوال پنجم – مرضیه علیدادی – 9631983

(کد های مربوط، در دو فرمت py. و ipynb. ضمیمه شده اند. – فایل dot_file.dot نیز ضمیمه شده است.)

.4

a) دیتاست با وجود مقادیر NULL به این صورت است:

	T3_resin	Serum_thyroxin	Serum_triiodothyronine	Basal_TSH	Abs_diff_TSH	Outcome
0	107.0	10.1	2.2	0.9	2.7	1.0
1	NaN	9.9	3.1	2.0	5.9	1.0
2	127.0	12.9	2.4	NaN	0.6	1.0
3	109.0	NaN	1.6	1.4	1.5	1.0
4	105.0	7.3	1.5	NaN	-0.1	1.0
5	105.0	6.1	2.1	1.4	7.0	1.0
6	110.0	NaN	1.6	1.6	2.7	1.0
7	114.0	NaN	2.4	1.5	5.7	1.0
8	106.0	?	2.2	1.5	NaN	1.0
9	107.0	13.0	1.1	0.9	3.1	1.0

ابتدا بررسی کردم و متوجه شدم که مقادیر NULL علاوه بر دو شکل NULL و NaN ، به صورت "?" هم ذخیره شده اند. پس آن ها را هم به NaN تبدیل کردم.

سپس تایپ همه ی متغیر ها را به float تبدیل کردم تا بتوان از آن ها میانگین گرفت.

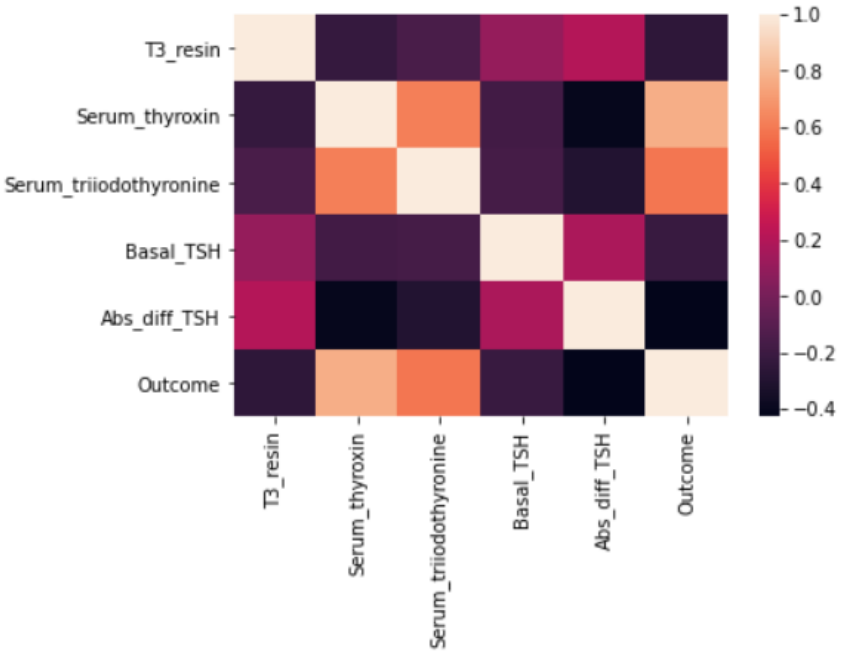
درنهایت، پس از جایگذاری مقادیر NULL با میانگین هر ستون، دیتاست به این شکل در آمد:

	T3_resin	Serum_thyroxin	Serum_triiodothyronine	Basal_TSH	Abs_diff_TSH	Outcome
0	107.000000	10.100000	2.2	0.900000	2.700000	1.0
1	107.761628	9.900000	3.1	2.000000	5.900000	1.0
2	127.000000	12.900000	2.4	1.274269	0.600000	1.0
3	109.000000	10.870115	1.6	1.400000	1.500000	1.0
4	105.000000	7.300000	1.5	1.274269	-0.100000	1.0
5	105.000000	6.100000	2.1	1.400000	7.000000	1.0
6	110.000000	10.870115	1.6	1.600000	2.700000	1.0
7	114.000000	10.870115	2.4	1.500000	5.700000	1.0
8	106.000000	10.870115	2.2	1.500000	2.047093	1.0
9	107.000000	13.000000	1.1	0.900000	3.100000	1.0

b) همبستگی ها:

	T3_resin	Serum_thyroxin	Serum_triiodothyronine	Basal_TSH	Abs_diff_TSH	Outcome
T3_resin	1.000000	-0.224923	-0.150091	0.103323	0.204132	-0.248920
Serum_thyroxin	-0.224923	1.000000	0.617121	-0.179007	-0.407017	0.776032
Serum_triiodothyronine	-0.150091	0.617121	1.000000	-0.166763	-0.298197	0.591060
Basal_TSH	0.103323	-0.179007	-0.166763	1.000000	0.176113	-0.210160
Abs_diff_TSH	0.204132	-0.407017	-0.298197	0.176113	1.000000	-0.427268
Outcome	-0.248920	0.776032	0.591060	-0.210160	-0.427268	1.000000

نمودار heatMap:



c) پراکندگی این ستون، در ابتدا به این شکل بود:

(به طور مشخص imbalanced است)

```
1.000000    144
2.000000     30
1.172414     11
Name: Outcome, dtype: int64
```

1.17 همان مقدار میانگین ستون بوده، که آن را جایگزین مقادیر null کرده بودم.

با توجه به نامعتبر بودن 1.17 از لحاظ مفهوم در این ستون، با توجه به اینکه به 1 نزدیک تر است و به طور کلی فراوانی 1 بیشتر است، احتمال اینکه 1 بوده باشند بیشتر است؛ پس آن ها را به 1 تبدیل کردم.

پس پراکندگی این ستون، به این شکل در آمد:

```
1.0    155
2.0     30
Name: Outcome, dtype: int64
```

فرضاً برای بالانس کردن آن می خواهیم از هریک از این دو مقدار، 50 درصد در این ستون وجود داشته باشد.

$$x = \frac{p(records) - rare}{1 - p}$$ = 125

پس باید 125 سطر با income = 2 به این دیتاست اضافه کنیم.

درنهایت، پراکندگی این ستون در دیتاست به این شکل شد:

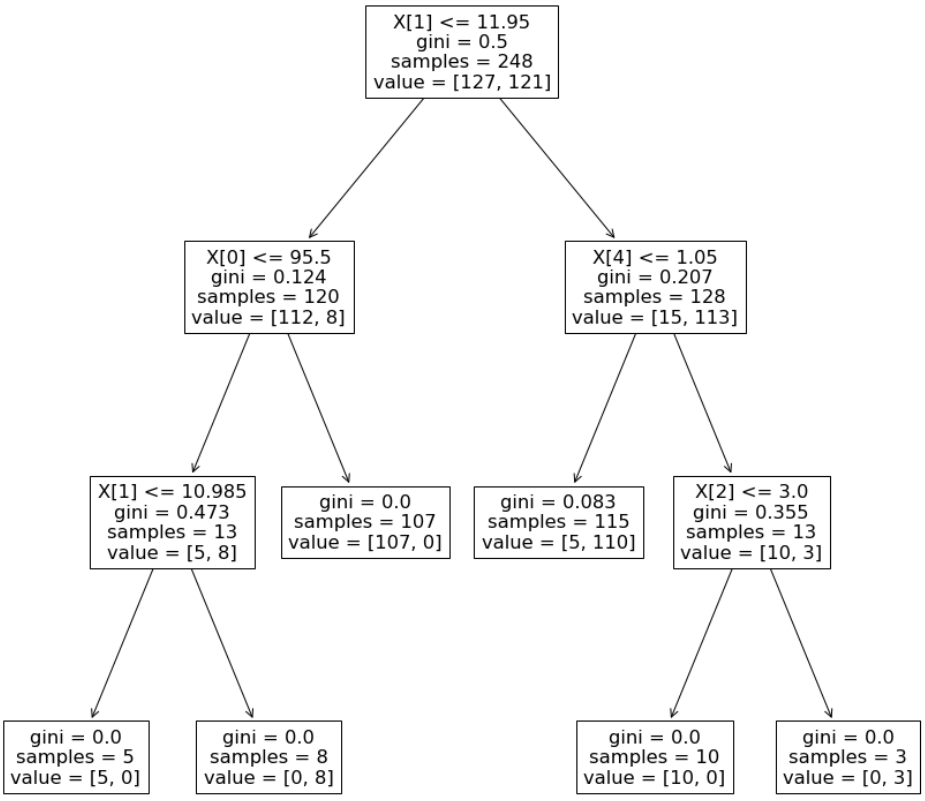
```
2.0    155
1.0    155
Name: Outcome, dtype: int64
```

(d

```
print(x_train.shape);
print(x_test.shape);
print(y_train.shape);
print(y_test.shape);
```

```
(248, 5)
(62, 5)
(248,)
(62,)
```

(e



f) 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:

```
print(confusion_matrix(y_test, y_pred))
```

```
[[26  2]
 [ 0 34]]
```

```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

         1.0       1.00      0.93      0.96        28
         2.0       0.94      1.00      0.97        34

    accuracy                           0.97        62
   macro avg       0.97      0.96      0.97        62
weighted avg       0.97      0.97      0.97        62
```

همانطور که مشخص است، از بین 62 پیشبینی، 60 تا را درست حدس زد و 2 تا را اشتباه حدس زد. یعنی دقت درخت تصمیم گیری ما، 96.77% است.

g)

- در حالت max_depth=1 : 3 اشتباه
- در حالت max_depth=2 : 6 اشتباه
- در حالت max_depth=3 : 2 اشتباه
- در حالت max_depth=4 : 2 اشتباه
- در حالت max_depth=5 : 2 اشتباه
- در حالت max_depth=6 : 1 اشتباه
- در حالت max_depth=7 : 1 اشتباه
- در حالت max_depth=8 : 2 اشتباه
- در حالت max_depth=9 : 0 اشتباه
- در حالت اتوماتیک: در این حالت، عمق را بررسی کردم و برابر 7 بود. دقت در این حالت برابر 100% بود.

به طور واضح، میتوان گفت هر چه max_depth بیشتر باشد، دقت بیشتر می شود. پس 9 را به عنوان بهترین در نظر می گیرم.

h) با استفاده از متد feature_importances بر روی مدل خود، می توانیم اهمیت هر کدام از ویژگی های دیتاست مورد نظر را، بدست آوریم.

این متد، یک امتیاز به هرکدام از ویژگی های دیتاست می دهد. هرچه امتیاز بیشتر باشد، آن ویژگی، نسبت به متغیر خروجی(هدف)، مهم تر یا مرتبط تر است.

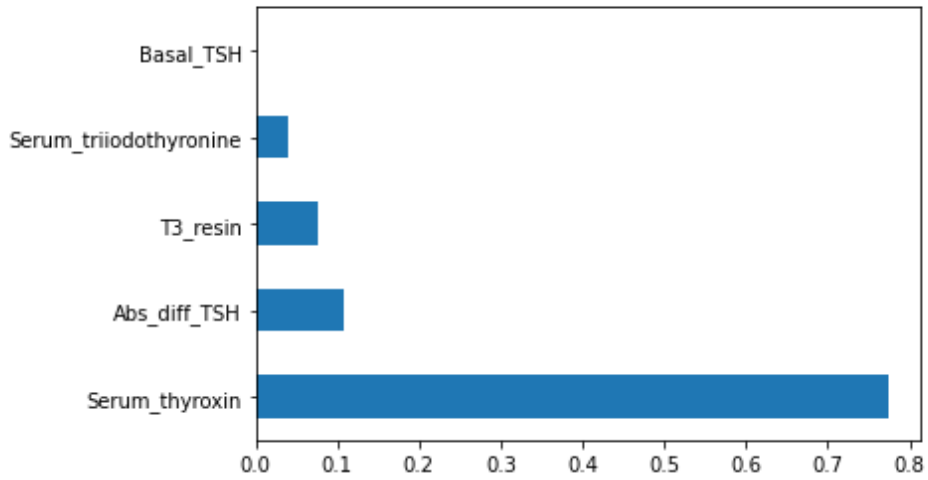
مثلا می توان با استفاده از آن، 10 تا مهم ترین متغیرهای دیتاست را مشاهده کرد.

مقداری که این متد برای متغیر های درخت تصمیم دیتاست ما نشان می دهد، به این صورت است:

```
print(cart.feature_importances_)
```

```
[0.07676913 0.77528725 0.04035761 0.         0.10758601]
```

```
feat_importances = pd.Series(cart.feature_importances_, index=x_train.columns)
feat_importances.nlargest(10).plot(kind='barh')
plt.show()
```



نتیجه ی این متد نشان می دهد، که متغیر Serum_thyroxin مهم ترین و مرتبط ترین متغیر نسبت به استخراج متغیر هدف است. و متغیر Basal_TSH کم اهمیت ترین و نامربوط ترین است.

i) Classifier که در بخش g به عنوان بهترین انتخاب شده بود، آنی بود که از عمق ماکسیمم 9 در آن استفاده شده بود؛ که دقت آن 100% برآورد شده بود. از همان Classifier در این بخش استفاده می کنم.

یک فایل با فرمت dot. ساختم و نتیجه ی این متد را در آن ذخیره کردم:

```
<_io.TextIOWrapper name='Desktop/dot_file.dot' mode='w' encoding='cp1256'>
```

فایل ضمیمه شده است.

j) گراف حاصل از dot_file.dot :

```
Source.from_file("Desktop/dot_file.dot")
```