به نام خدا

دانشگاه تهران

پردیس دانشکده‌های فنی

دانشکده برق و کامپیوتر

# درس سیستم‌های هوشمند

## تمرین شماره 2

**نام و نام خانوادگی : شایان واصف**

**شماره دانشجویی : 810197603**

مهر 1400

# سر فصل مطالب

# درخت تصمیم ( تحلیلی )

## طراحی طبقه بند

### بخش شبیه سازی

ابتدا دیتاست را تشکیل می‌دهیم :

| | blood pressure | CL_level | Cigarette | Weight | cond |
|---|---|---|---|---|---|
| **0** | T | N | F | OW | T |
| **1** | F | N | T | N | F |
| **2** | F | C | F | OW | T |
| **3** | F | H | T | OW | T |
| **4** | T | C | T | F | T |
| **5** | T | H | T | N | T |
| **6** | F | H | F | F | F |
| **7** | T | N | T | N | T |
| **8** | T | C | F | F | T |
| **9** | F | N | F | OW | F |
| **10** | F | C | T | N | T |
| **11** | T | H | F | OW | F |
| **12** | T | N | T | OW | T |
| **13** | T | H | F | F | F |

شکل 1-1 : دیتاست کلی

خروجی های حاصل از آموزش مدل توسط الگوریتم ID3 به صورت زیر است :

```
   blood pressure CL_level Cigarette Weight cond
0               T        N         F     OW    T
1               F        N         T      N    F
2               F        C         F     OW    T
3               F        H         T     OW    T
4               T        C         T      F    T
5               T        H         T      N    T
6               F        H         F      F    F
7               T        N         T      N    T
8               T        C         F      F    T
9               F        N         F     OW    F
10              F        C         T      N    T
11              T        H         F     OW    F
12              T        N         T     OW    T
13              T        H         F      F    F
```

```
P-node : T
This is best-feature : CL_level
this is tree : {'CL_level': {'C': 'T'}}
   blood pressure CL_level Cigarette Weight cond
3               F        H         T     OW    T
5               T        H         T      N    T
6               F        H         F      F    F
11              T        H         F     OW    F
13              T        H         F      F    F
P-node : F
This is best-feature : Cigarette
this is tree : {'Cigarette': {'F': 'F'}}
this is tree : {'Cigarette': {'F': 'F', 'T': 'T'}}
this is tree : {'CL_level': {'C': 'T', 'H': {'Cigarette': {'F': 'F', 'T':
'T'}}}}
   blood pressure CL_level Cigarette Weight cond
0               T        N         F     OW    T
1               F        N         T      N    F
7               T        N         T      N    T
9               F        N         F     OW    F
12              T        N         T     OW    T
P-node : T
This is best-feature : blood pressure
this is tree : {'blood pressure': {'F': 'F'}}
this is tree : {'blood pressure': {'F': 'F', 'T': 'T'}}
this is tree : {'CL_level': {'C': 'T', 'H': {'Cigarette': {'F': 'F', 'T':
'T'}}, 'N': {'blood pressure': {'F': 'F', 'T': 'T'}}}}
Final tree :{'CL_level': {
'C': 'T',
'H': {'Cigarette': {'F': 'F', 'T': 'T'}},
'N': {'blood pressure': {'F': 'F', 'T': 'T'}}
}
}
```

تحلیلی

برای اولین گره مادر ، بهره اطلاعات مربوط به هر 4 تا ویژگی دیتاست را محاسبه می‌کنیم :

```
print(InfoGain(df,'blood pressure','cond'))
print(InfoGain(df,'CL_level','cond'))
print(InfoGain(df,'Cigarette','cond'))
print(InfoGain(df,'Weight','cond'))
```

```
0.04812703040826927
0.2467498197744391
0.15183550136234136
0.029222565658954647
```

شکل 2-1 : بهره اطلاعات بدست آمده از هر ستون ویژگی

1. طبق نتایج بالا ، شروع درخت را با ویژگی "سطح کلسترول" بسط می‌دهیم .بنابراین این ستون را از ادامه محاسبات حذف می‌کنیم و 3 ستون باقی می‌ماند .

*در داخل ستون "سطح کلسترول" سه مقدار Unique وجود دارد :

$$\begin{cases} N : \text{نرمال} \\ C : \text{بحرانی} \\ H : \text{بالا} \end{cases}$$

بر حسب هر کدام از آنها ، دیتاست جدید را شکل می‌دهیم :

C

| | blood pressure | Cigarette | Weight | cond |
|---|---|---|---|---|
| 2 | F | F | OW | T |
| 4 | T | T | F | T |
| 8 | T | F | F | T |
| 10 | F | T | N | T |

شکل 3-1 : دیتاست بدست آمده از گره C

همانطور که مشخص است ، با انتخاب گره C ، تمام لیبل ها موجود مقدار T دارند ، بنابراین کار این گره پایان یافته است و return می‌کنیم و به سراغ گره بعدی می‌رویم :

H

| | blood pressure | Cigarette | Weight | cond |
|---|---|---|---|---|
| 3 | F | T | OW | T |
| 5 | T | T | N | T |
| 6 | F | F | F | F |
| 11 | T | F | OW | F |
| 13 | T | F | F | F |

شکل 4-1 : دیتاست بدست آمده از گره H

با انتخاب H ، مشخص است که هنوز می‌توانیم عمق درخت را افزایش دهیم . بنابراین برای ستون های باقی مانده بهره اطلاعات را بدست می‌آوریم :

```
print(InfoGain(df_new,'blood pressure','cond'))
print(InfoGain(df_new,'Cigarette','cond'))
print(InfoGain(df_new,'Weight','cond'))
```

```
0.01997309402197489
0.9709505944546686
0.5709505944546686
```

شکل 5-1 : بهره اطلاعات بدست آمده با انتخاب H به عنوان گره مادر لایه 1 (Parent node )

2. طبق بالا ، ستون "مصرف سیگار" بیشترین بهره اطلاعات را دارد . پس به عنوان گره بعدی انتخاب می‌شود و از ویژگی های باقی مانده حذف می‌شود . در ستون "مصرف سیگار" دو مقدار Unique وجود دارد :

$$\begin{cases} T : \text{بله} \\ F : \text{خیر} \end{cases}$$

T

```
df_new[df_new['Cigarette']=='T'].drop('Cigarette',axis=1)
```

| | blood pressure | Weight | cond |
|---|---|---|---|
| 3 | F | OW | T |
| 5 | T | N | T |

شکل 5-1 : دیتاست بدست آمده از زیرگره T

همانطور که مشخص است ، با انتخاب زیر گره T ، تمام لیبل ها موجود مقدار T دارند ، بنابراین کار این گره پایان یافته است و return می‌کنیم و به سراغ زیر گره بعدی می‌رویم .

F

```
df_new[df_new['Cigarette']=='F'].drop('Cigarette',axis=1)
```

| | blood pressure | Weight | cond |
|---|---|---|---|
| 6 | F | F | F |
| 11 | T | OW | F |
| 13 | T | F | F |

شکل 6-1 : دیتاست بدست آمده از زیرگره F

همانطور که مشخص است ، با انتخاب زیر گره T ، تمام لیبل ها موجود مقدار F دارند ، بنابراین کار این گره پایان یافته است و return می‌کنیم و به سراغ گره بعدی می‌رویم .

N

| | blood pressure | Weight | cond |
|---|---|---|---|
| **0** | T | OW | T |
| **1** | F | N | F |
| **7** | T | N | T |
| **9** | F | OW | F |
| **12** | T | OW | T |

شکل 7-1 : دیتاست بدست آمده از گره N

با انتخاب N ، مشخص است که هنوز می‌توانیم عمق درخت را افزایش دهیم . بنابراین برای ستون های باقی مانده بهره اطلاعات را بدست می‌آوریم :

```
print(InfoGain(df_new1,'blood pressure','cond'))
print(InfoGain(df_new1,'Weight','cond'))
```

```
0.9709505944546686
0.01997309402197489
```

شکل 8-1 : بهره اطلاعات بدست آمده با انتخاب N به عنوان گره مادر لایه 1 (Parent node )

3. طبق اطلاعات بالا ، ستون "فشار خون" بهره اطلاعات بیشتری دارد ، پس به عنوان زیر گره بعدی اتخاب می‌شود و این ویژگی را حذف می‌کنیم . در این ستون دو مقدار Unique وجود دارد :

T

| | Weight | cond |
|---|---|---|
| **0** | OW | T |
| **7** | N | T |
| **12** | OW | T |

شکل 8-1 : دیتاست بدست آمده از زیر گره T

همانطور که مشخص است ، با انتخاب زیر گره T ، تمام لیبل ها موجود مقدار T دارند ، بنابراین کار این گره پایان یافته است و return می‌کنیم و به سراغ زیر گره بعدی می‌رویم .

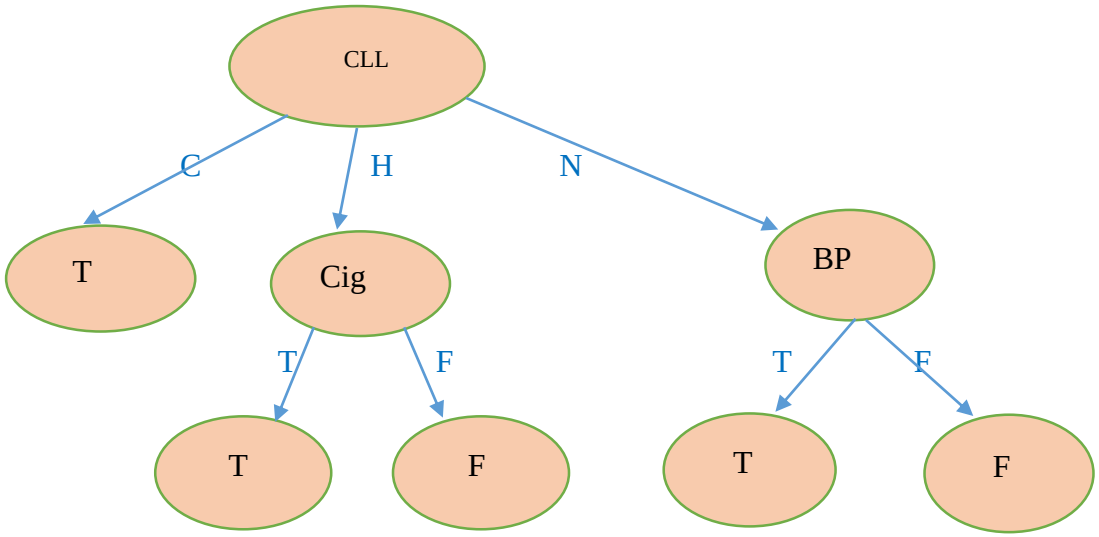
F

| | Weight | cond |
|---|---|---|
| 1 | N | F |
| 9 | OW | F |

شکل 9-1 : دیتاست بدست آمده از زیر گره F

همانطور که مشخص است ، با انتخاب زیر گره T ، تمام لیبل ها موجود مقدار F دارند ، بنابراین کار این گره پایان یافته است و return می‌کنیم .

در این مرحله چون هیچ گره ای باقی نمانده است ، کار ما تمام شده است . شکل درخت نهایی به صورت زیر خواهد بود :




شکل 10-1 : شکل نهایی درخت تصمیم

## آزمون طبقه بند

### شبیه سازی :

دیتاست تست را تشکیل می‌دهیم :

| | blood pressure | CL_level | Cigarette | Weight | cond |
|---|---|---|---|---|---|
| 0 | T | N | T | F | T |
| 1 | T | H | T | F | T |
| 2 | T | H | F | N | F |
| 3 | T | N | F | N | F |
| 4 | F | N | T | OW | T |

شکل 2-1 : داده تست

تابعی می‌نویسیم که بصورت بازگشتی برای هر سطر دیتاست ( هر Sample ) لیبل مورد نظر را جستجو کند و با لیبل اصلی مقایسه کند. طبق عکس زیر ، دارای دقت 60 درصد بر روی داده تست هستیم :

```
test(df_test,tree)
```

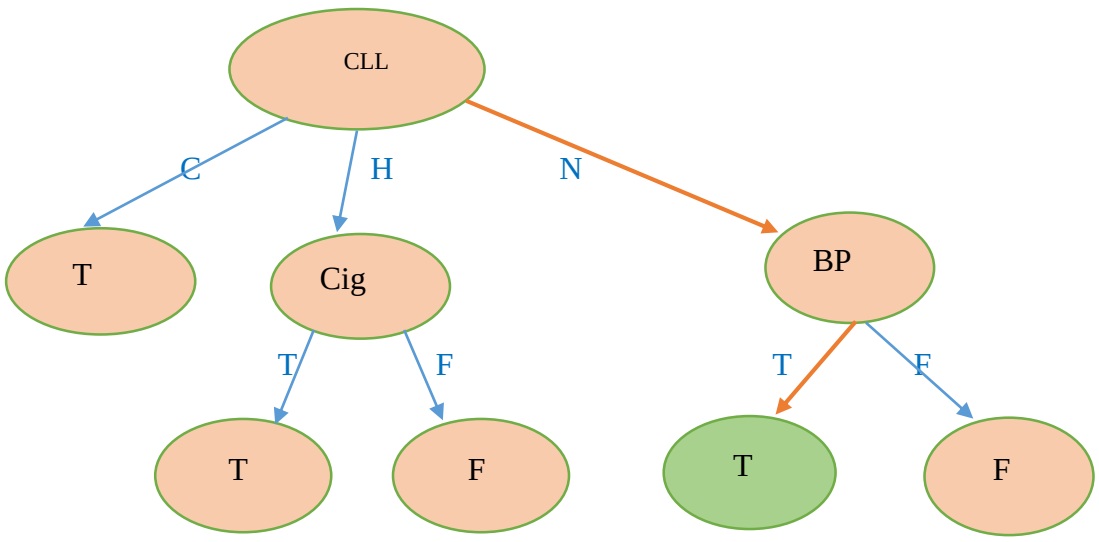
```
  predicted
0         T
1         T
2         F
3         T
4         F
The prediction accuracy is:  60.0 %
```

شکل 2-2 : دقت بدست آمده بر روی داده تست

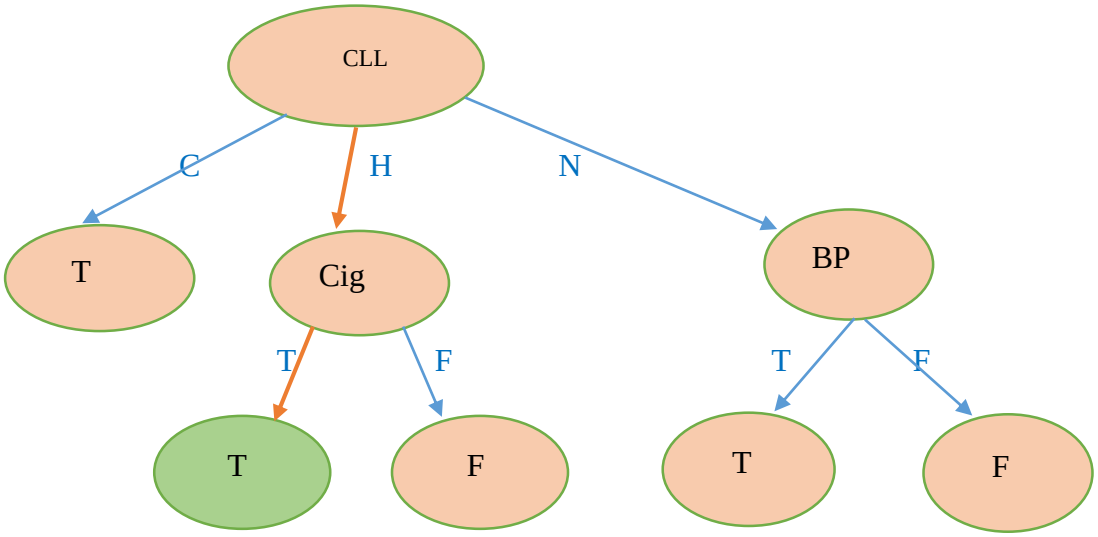
## تحلیلی

در روش تحلیلی ، هر سطر از داده ها که یک Sample می‌باشد را نگاه می‌کنیم و به ترتیب گره ها بدست آمده در درخت پیش می‌رویم تا به لیبل مورد نظر برسیم ، و در نهایت لیبل بدست آمده را با لیبل داده تست مقایسه می‌کنیم .

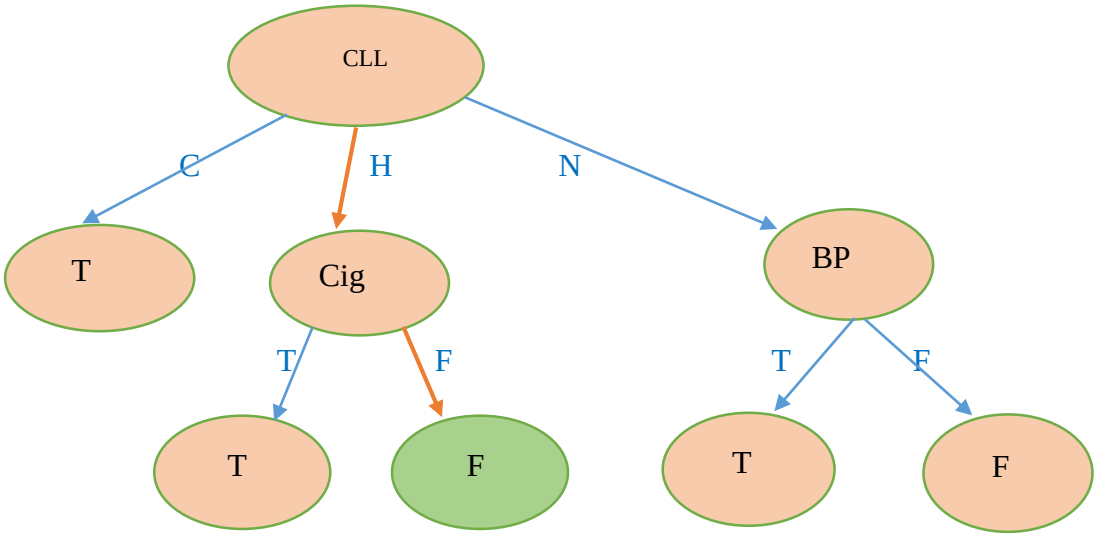
- داده اول :



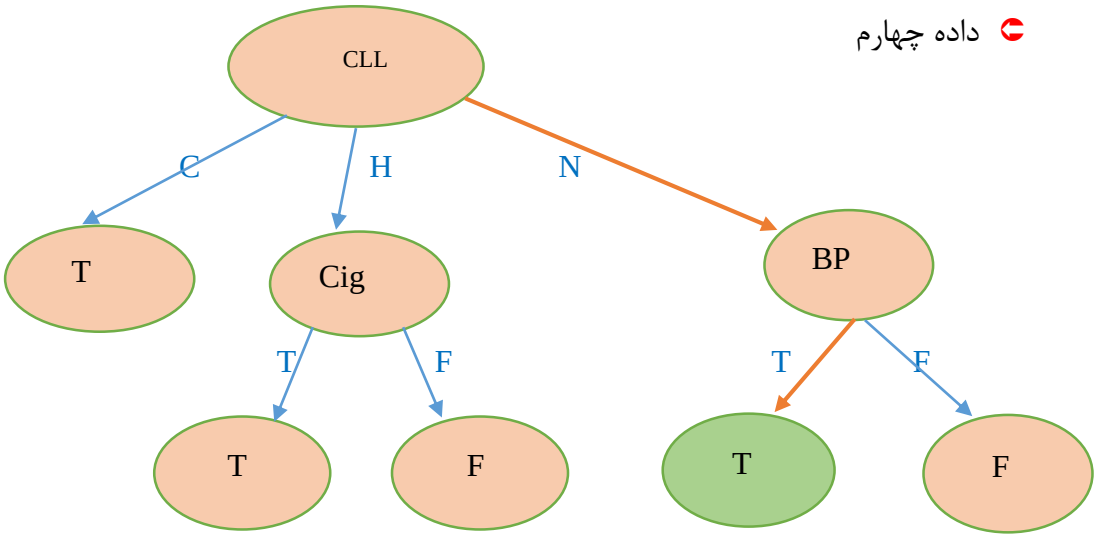
داده دوم :



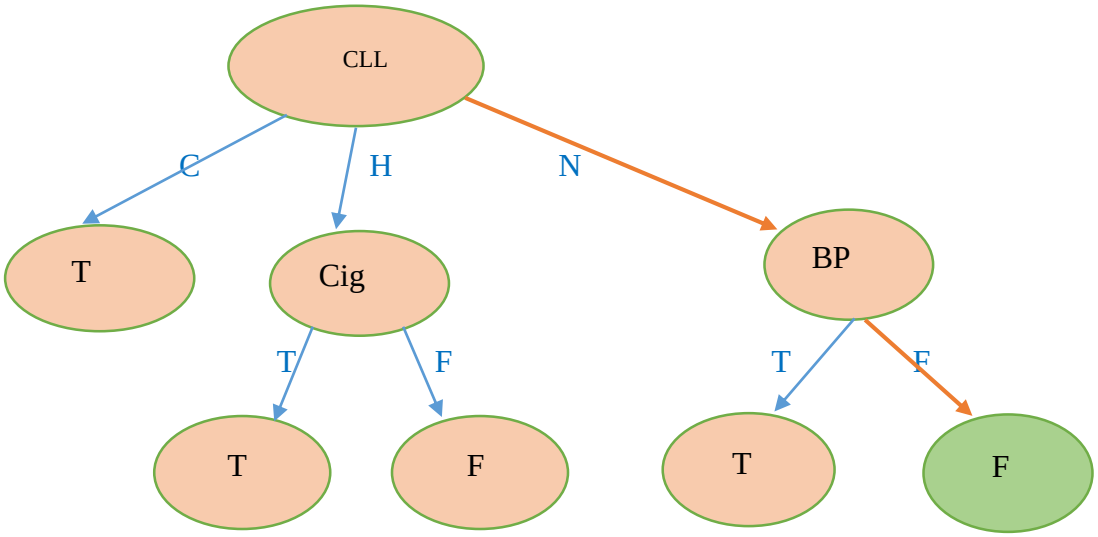

داده سوم



داده چهارم




داده پنجم




بنابراین ماتریس آشفتگی به صورت زیر خواهد بود :

| | T | F |
|---|---|---|
| T | 2 | 1 |
| F | 1 | 1 |

شکل 3-2 : ماتریس آشفتگی

## افزایش قوام طبقه بند

طبیعتا چون درخت های تصمیم هیچ شرطی برای توقف ندارد ، تا جایی که تمامی داده ها را طبقه بندی کند درخت پیش می‌رود . طبیعتا اگر یک سری نویز در داده ها داشته باشیم ، با این کار نویز های دیتاست را نیز مدل می‌کنیم و طبیعتا به مدل بسیار Over fit می‌شویم . دو رویکرد در مواجه با این مشکل وجود دارد :

- (Early Stopping) Pre-Pruning

در این روش در هر مرحله ، خطای مربوط به Cross Validation را چک می‌کنیم . اگر این خطا به طور خوبی کم نشد ، درخت را متوقف می‌کنیم .

- Post-Pruning

در این روش ابتدا درخت کامل تشکیل می‌شود .که در دو روش انجام می‌شود :

- کمترین خطا : در این روش درخت از نقطه‌ای که کمترین خطای Cross validation را دارد هرز می‌گردد.
- کوچکترین درخت : در این روش درخت کمی بیشتر از حداقل خطا هرس می‌شود. در واقع درخت کوچکتر به قیمت افزایش اندک خطا قابل درکتر است.

## درخت تصمیم ( شبیه سازی )

### الف ) طراحی طبقه بند :

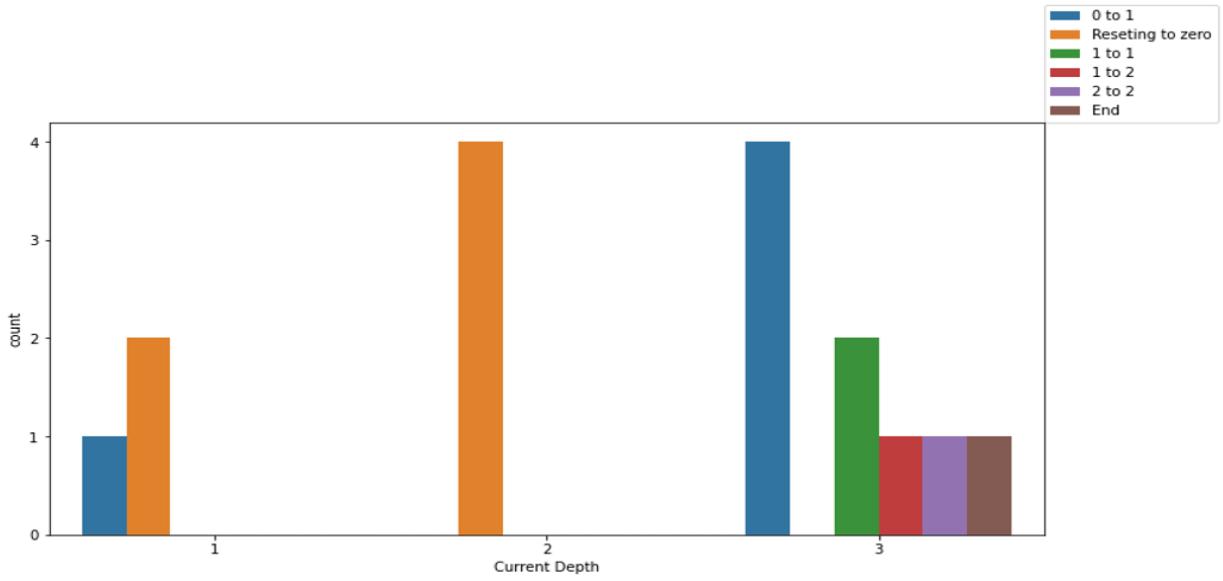
**Max Depth =3**

برای آموزش طبقه بند از یک روش بازگشتی استفاده می‌کنیم به طوریکه گره مادر (Parent node) هر مرحله نقطه شروع الگوریتم در روش بازگشتی می‌باشد . به بیان دیگر ما در هر Parent node تابع ID3 نوشته شده را فراخوانی می‌کنیم .

*نکته مهم در استفاده از ساختار بازگشتی در این است که متغیر ها محلی هستند و در طول فرآیند ما آنها را با خود حمل نمی‌کنیم . طبیعتا برای اینکه ما در هر مرحله تابع را صدا میزنیم ، باید شرطی گذاشته شود تا هنگامی که کار ما با آن تابع تمام شد ، آن تابع از حافظه پاک شود . این کار توسط دستور return انجام می‌شود که وقتی که کار ما با تمام برگ های گره مادر تمام شد ، یک مقداری return سود و به سراغ گره بعدی برویم .

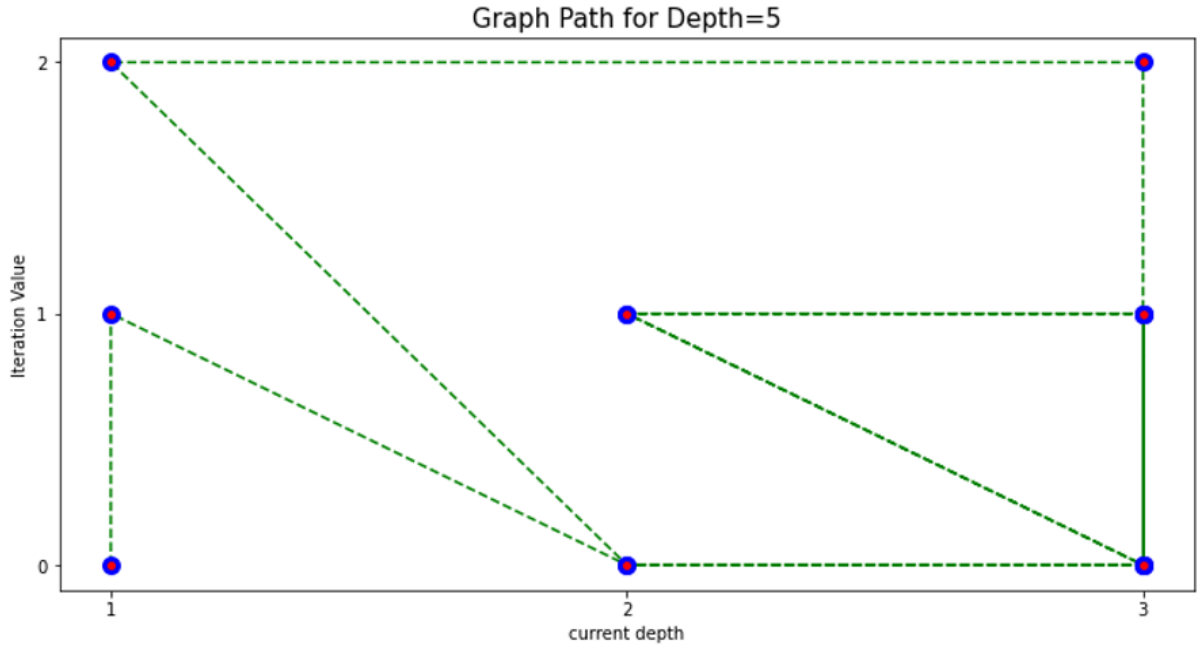
در هر بار مشاهده گره مادر ، در یک حلقه Iteration میکنیم تا تمام برگ ها را پوشش دهیم . متغیری که با تعویض برگ به صورت محلی عوض می‌شود را j می‌نامیم . هر بار که j صفر می‌شود به این معنی است که از تابع بازگشتی return شده ایم و یه حلقه جدید را شروع کرده ایم ، بنابراین عمق ما باید یکی اضافه شود . این عمق را current depth نام گذاری کرده‌ایم و چون جزو متغیر های تابع ID3 می‌باشد به صورت محلی تعیین می‌شود و بروز رسانی می‌شود .

در زیر برای عمق درخت برابر 3 ، count plot مربوط به تغییرات j را در هر عمق رسم کرده‌ایم :



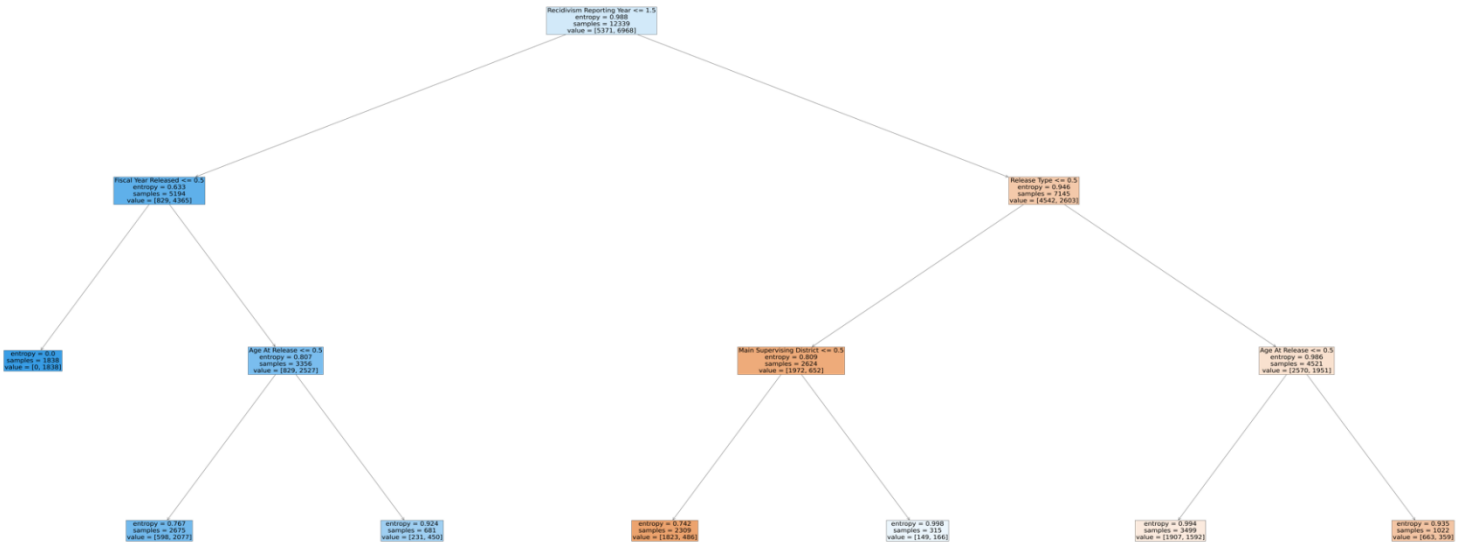

شکل 2-1-1-1 : count plot مربوط به تغییرات j در هر عمق

همچنین در نمایی دیگر ، می‌توانیم به صورت گراف رابطه بین تغییرات j و current depth را بکشیم :




شکل 2-1-1-2 : رابطه بین تغییرات j و current depth

در نهایت درخت را به عمق 3 رسم می‌کنیم :

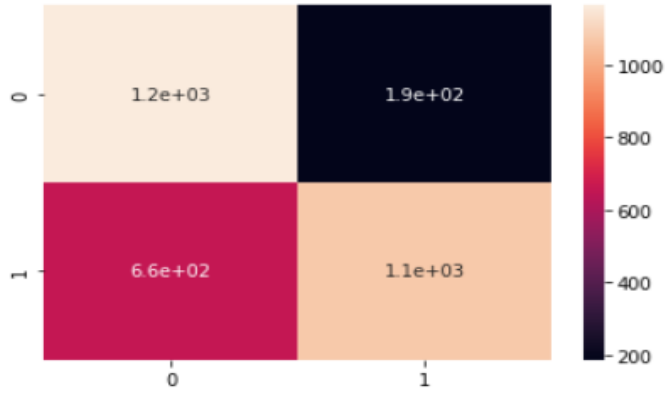


شکل 3-1-1-2 : درخت حاصل از آموزش با عمق 3

در نهایت ماتریس آشفتگی و گزارش طبقه بندی به صورت زیر خواهد بود :

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.64 | 0.86 | 0.73 | 1350 |
| 1 | 0.85 | 0.62 | 0.72 | 1735 |
| accuracy | | | 0.73 | 3085 |
| macro avg | 0.75 | 0.74 | 0.73 | 3085 |
| weighted avg | 0.76 | 0.73 | 0.72 | 3085 |

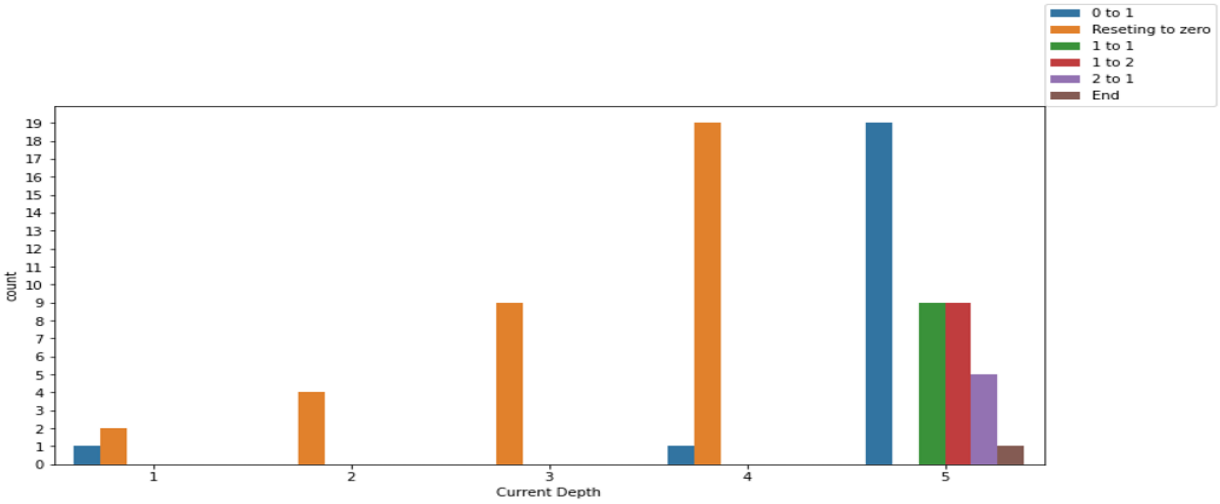
The prediction accuracy is: 72.6094003241491 %




شکل 4-1-1-2 : ماتریس آشفتگی و گزارش طبقه بند

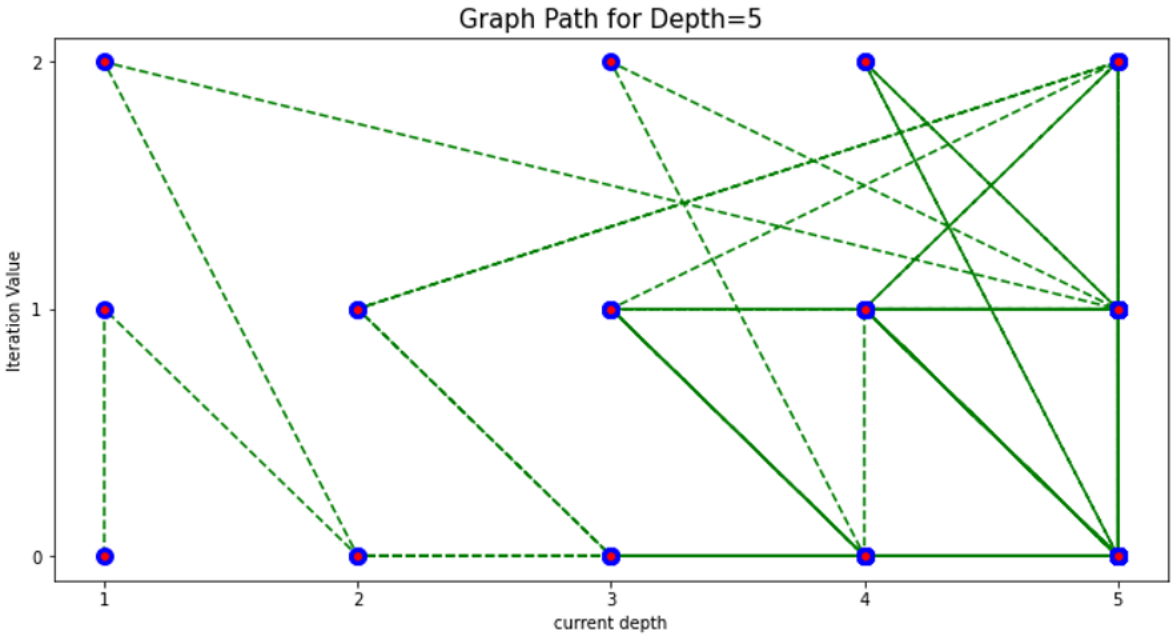
## Max Depth=5

تمامی مراحل گفته شده در قسمت قبل را برای عمق جدید انجام می‌دهیم .



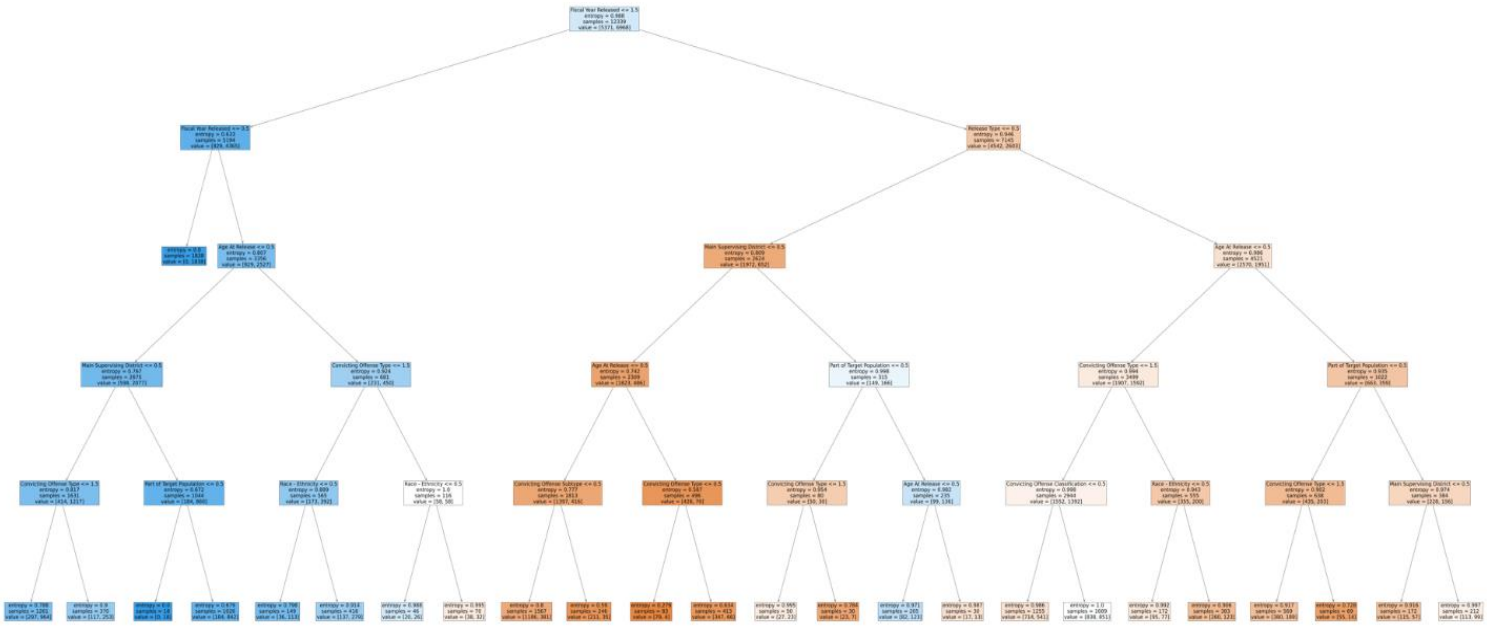

شکل 2-1-2-1 : count plot مربوط به تغییرات j در هر عمق

همچنین در نمایی دیگر ، می‌توانیم به صورت گراف رابطه بین تغییرات j و current depth را بکشیم




شکل 2-1-2-2 : رابطه بین تغییرات j و current depth

در نهایت درخت را به عمق 5 رسم می‌کنیم :

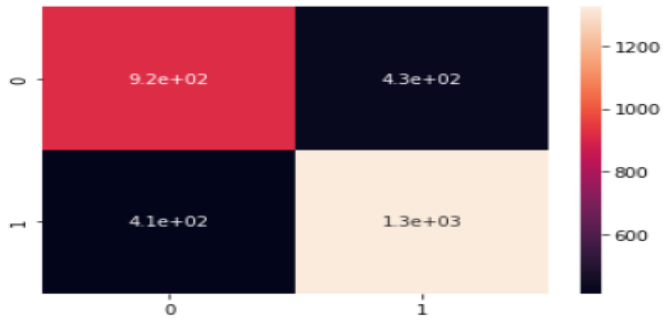


شکل 3-1-2-2 : درخت حاصل از آموزش با عمق 5

در نهایت ماتریس آشفتگی و گزارش طبقه بندی به صورت زیر خواهد بود :

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.69 | 0.68 | 0.69 | 1350 |
| 1 | 0.75 | 0.76 | 0.76 | 1735 |
| accuracy | | | 0.73 | 3085 |
| macro avg | 0.72 | 0.72 | 0.72 | 3085 |
| weighted avg | 0.73 | 0.73 | 0.73 | 3085 |

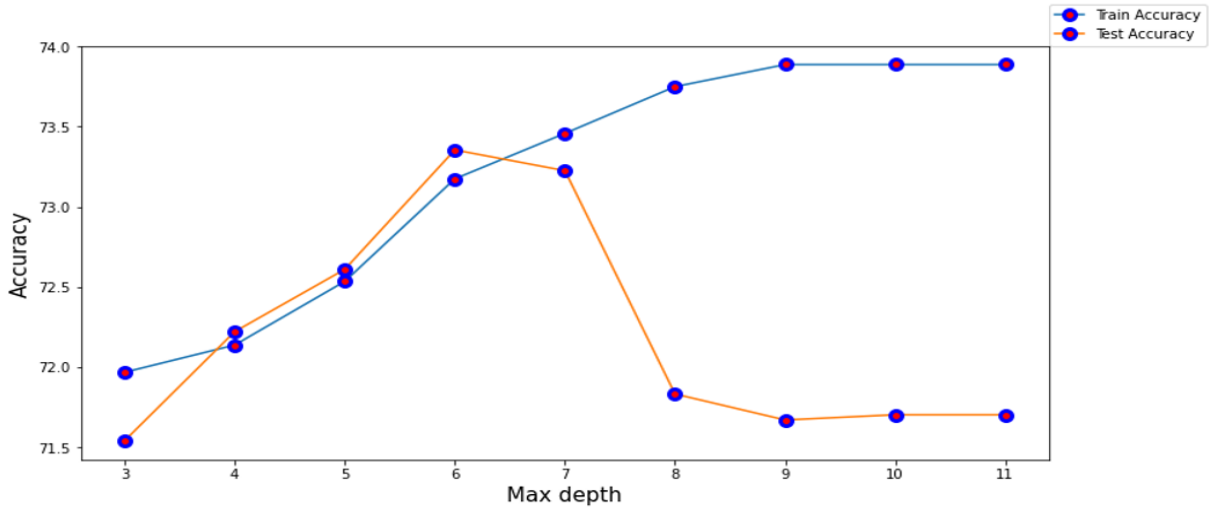
The prediction accuracy is: 72.70664505672609 %




شکل 4-2-1-2 : ماتریس آشفتگی و گزارش طبقه بند

**تحلیل :**

همانطور که از مقایسه دو بخش اول و دوم مشخص است ، با افزایش عمق از 3 به 5 تنها 0.1 به دقت ما اضافه شده است. بنابراین با افزایش عمق درخت بهبودی محسوسی در طبقه بند نخواهیم داشت و صرفا حجم محاسبات را افزایش داده‌ایم .ابتدا نموداری از دقت طبقه بند بر حسب عمق درخت را برای هر دو داده Train و Test در یک نمودار رسم می‌کنیم تا مقدار بهینه عمق را پیدا کنیم :



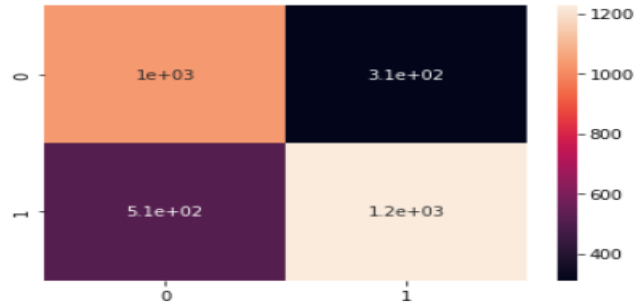

شکل 1-3-1-2 : دقت طبقه بند برای داده آموزش و تست بر حسب عمق درخت

طبق شکل بالا ، برای عمق 6 ، بهترین دقت را در مجموعه تست داریم . همانطور که مشخص است با افزایش عمق ، دقت در مجموعه آموزش افزایش پیدا کرده و تا حدی اشباع می‌شود ، چون تا بیشتر از 74 درصد نمی‌رسد .

## استفاده از جنگل تصادفی

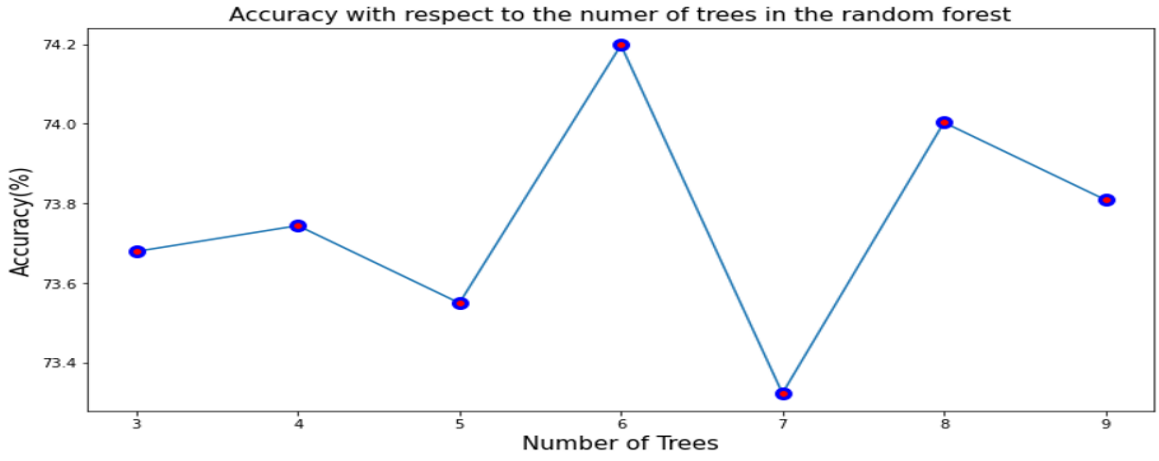
ابتدا برای تعداد 3 درخت ، جنگل تصادفی را آموزش می‌دهیم و سپس تحلیلی را انجام می‌دهیم :

```
The prediction accuracy is:  73.322528363047
              precision    recall  f1-score   support

           0       0.67      0.77      0.72      1347
           1       0.80      0.71      0.75      1738

    accuracy                           0.73      3085
   macro avg       0.73      0.74      0.73      3085
weighted avg       0.74      0.73      0.73      3085

The prediction accuracy is:  73.322528363047 %
AxesSubplot(0.125,0.125;0.62x0.755)
73.322528363047
```




شکل 1-1-2-2 : ماتریس آشفتگی و گزارش طبقه بند

همانطور که مشاهده شد ، دقت نسبت به حالت قبل کمی افزایش پیدا کرد ولی محسوس نیست . ابتدا دقت مجموعه تست را بر حسب تعداد درخت انتخابی رسم می‌کنیم :



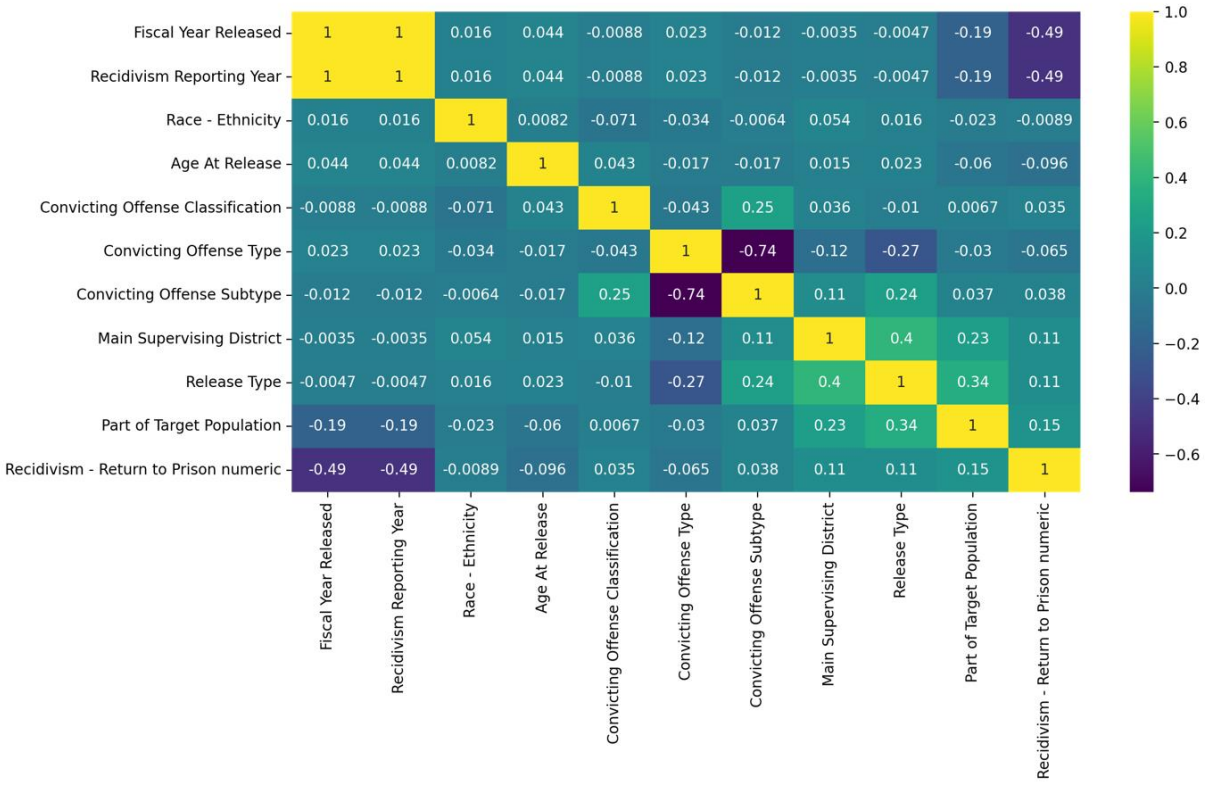

شکل 2-1-2-2 : دقت داده تست به ازای تعداد درخت

همانطور که مشخص است ، در جنگل تصادفی با افزایش تعدا درخت ها over fit نمی‌شویم و لزوما با افزایش درخت ها دقت افزایش پیدا نخواهد کرد.

برای آنکه متوجه شویم که چرا از تغییر درخت تصمیم به جنگل تصادفی ، دقت تغییر محسوسی نکرد ، از Correlation بین ستون های ویژگی استفاده می‌کنیم تا یک تحلیل آماری ارائه دهیم .

Correlation در واقع معیاری از شباهت بین دو آرایه ارائه می‌دهد که عددی بین 1 و 1- می‌باشد. مقدار مثبت آن بیانگر رابطه مستقیم درایه های دو آرایه می‌باشد و مقدار منفی آن نشانه رابطه عکس می‌باشد .

در ابدت به کمک دستور Label encode ، برای ستون های Categorical ، لیبل های عددی تعریف می‌کنیم و سپس توسط دستور df.corr() ، با کمک Heat map به جدول زیر می‌رسیم :

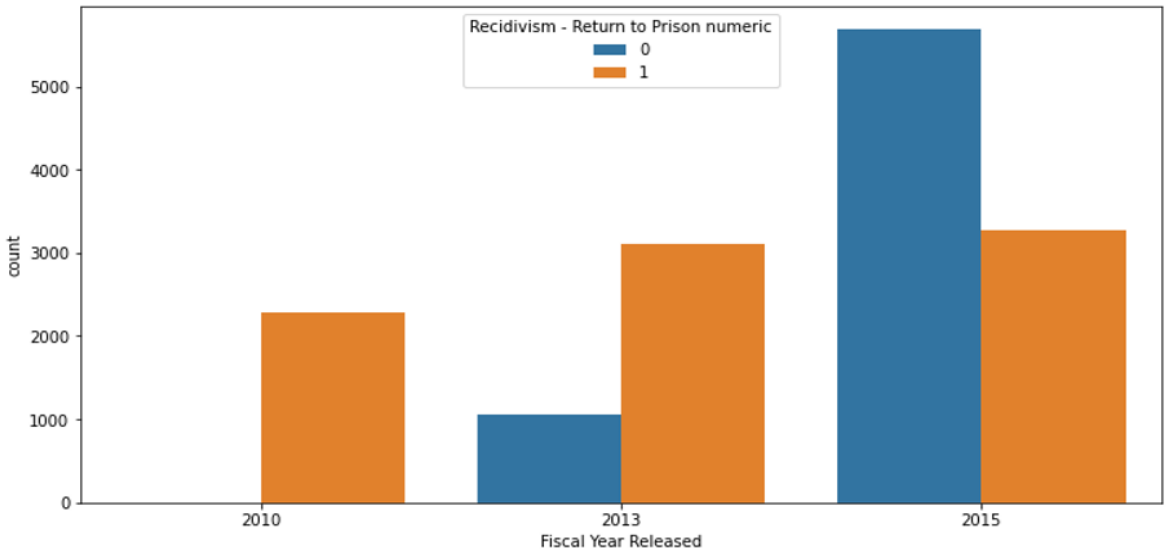



شکل 3-1-2-2 : Heatmap مربوط به Correlation ستون های ویژگی

طبق رنگ بندی مشاهده شده ، رنگ زرد بیشترین مقدار Correlation را نشان می‌دعد و رنگ بنفش کمترین آن را نشان می‌دهد . مشاهدات خود را از جدول در زیر به ترتیب می‌نویسم :

1. دو ستون "Fiscal Year Released" و "Recidivism Reporting Year" با یکدیگر Correlation 1 دارند ، به این معنی که تغییرات این دو ستون در دیتاست ، عینا مشابه هم می‌باشد .

2. دو ویژگی "Convincing office type" و "Convincing office subtype" با هم Correlation بالای 0.74- را دارند که نشان می‌دهد رابطه این ستون تقریبا عکس می‌باشد .
3. دو ستون "Fiscal Year Released" و "Recidivism Reporting Year" با ستون Target یعنی "Recidivism-Return to prison numeric" ، 0.49- Correlation دارند ، که نشان میدهد آزاد شدن یا نشدن زندانی رابطه بسیار قوی با این دو ستون دارد ولی در جهت عکس آن تغییر می‌کند . برای آنکه درک بهتری از این موضوع داشته باشیم ، Count plot مربوط به ستون "Fiscal year Released" را رسم می‌کنیم :




شکل 4-1-2-2 : Count plot مربوط به ستون "Fiscal year Released"

همانطور که از شکل بالا مشاهده می‌کنیم ، با افزایش سال از 2010 تا 2015 تعداد لیبل 0 که در واقع تکرار جنایت بوده ، افزایش پیدا کرده است .

*بنابراین به دلیل اینکه میزان اهمیت ستون "Fiscal year Released" نسبت به ستون های ویژگی دیگر بسیار بیشتر می‌باشد ، با همین ویژگی نیز میتوان طبقه بندی را انجام داد و بنابراین استفاده از روش جنگل تصادفی نسبت به درخت تصمیم ، بهبود قابل توجهی نمی‌دهد.

*همچنین Data Frame زیر میزان اهمیت ویژگی ها را در آموزش مدل بیان میکند :

| Feature | Importance |
|---|---|
| Fiscal Year Released | 0.478419 |
| Recidivism Reporting Year | 0.395111 |
| Part of Target Population | 0.052799 |
| Release Type | 0.027319 |
| Age At Release | 0.021852 |
| Main Supervising District | 0.015571 |
| Convicting Offense Type | 0.007629 |
| Convicting Offense Classification | 0.000661 |
| Convicting Offense Subtype | 0.000639 |
| Race - Ethnicity | 0.000000 |

شکل 5-1-2-2 : درصد اهمیت ویژگی های مدل در آموزش جنگل تصادفی

طبق نتایج بالا هم ، می‌توان نتیجه گیری کرد که دو ستون ذکر شده 40+48=98 درصد در طبقه بند مذکور موثر هستند.

*نکته مهم در انتخاب ویژگی های یک مدل در این است که ویژگی ها با هم Correlation زیادی نداشته باشند . در واقع بین دو ویژگی با Correlation 1 ، یکی از ویژگی ها نباید اطلاعات جدید اضافه کند . در حالیکه در جدول بالا ، دو ویژگی ذکر شده هر دو به یک مقدار از اهمیت برخوردار هستند.

دلیل این امر استفاده از روش جنگل تصادفی است ، که چون در هر حالت روی یک درخت تصمیم گیری می‌کند ، امکان دارد در هر حالت یکی از دو ویژگی بالا به عنوان مهم ترین ویژگی با درصد بالایی شناخته شود ولی چون در نهایت ما به صورت میانگین درصد اهمیت هر کدام از ویژگی ها را اعلام می‌کنیم ، به این بیان است که این دو ویژگی بیشترین تاثیر را روی دقت کلی طبقه بند گذاشته اند.

## استفاده از کتابخانه :

- درخت تصمیم ( Max Depth =3 )

طبق بخش اول قسمت الف ، این بار به کمک کتابخانه آماده درخت تصمیم را پیاده سازی می‌کنیم .

بعد از fit کردن مدل ، می‌توانیم می‌توانیم مثل قسمت قبل درصد اهمیت هر کدام از ویژگی ها را اعلام کنیم:

| Feature | Importance |
|---|---|
| Recidivism Reporting Year | 0.702114 |
| Fiscal Year Released | 0.191365 |
| Release Type | 0.058594 |
| Main Supervising District | 0.030817 |
| Age At Release | 0.017110 |
| Race - Ethnicity | 0.000000 |
| Convicting Offense Classification | 0.000000 |
| Convicting Offense Type | 0.000000 |
| Convicting Offense Subtype | 0.000000 |
| Part of Target Population | 0.000000 |

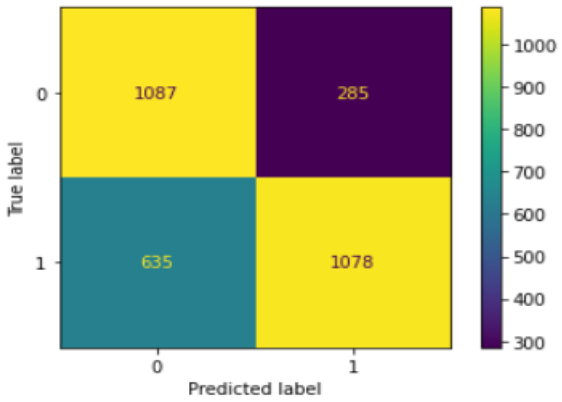
شکل 2-3-1-1 : درصد اهمیت ویژگی های مدل در آموزش درخت تصمیم

طبق نتیجه بدست آمده در بالا ، حال میتوانیم ببینیم که تنها یک ویژگی "Recidivism Reporting Year" با اختلاف بیشترین درصد اهمیت را میان ویژگی های دیگر دارد .

*نکته مهم در اینجا این است که هر چقدر من عمق را بیشتر کنم و در واقع از تعداد بیشتری از ظرفیت دیتاست خود استفاده کنم ، میزان اهمیت یکی از دو ویژگی ذکر شده در دیتاست من افزایش پیدا میکند و باعث می‌شود که ویژگی دوم در عمق های بیشتر هیچ اطلاعاتی به من ندهد . در واقع این دلیلی بر این است که چرا با افزایش عمق ، دقت طبقه بند افزایش محسوس پیدا نمی‌کند.

*در جدول بالا ، ویژگی دوم که با ویژگی اول Correlated است ، در رتبه دوم از نظر اهمیت قرار دارد .

در نهایت ماتریس آشفتگی و گزارش طبقه بند را بدست می‌آوریم :




| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.63 | 0.79 | 0.70 | 1372 |
| 1 | 0.79 | 0.63 | 0.70 | 1713 |
| accuracy | | | 0.70 | 3085 |
| macro avg | 0.71 | 0.71 | 0.70 | 3085 |
| weighted avg | 0.72 | 0.70 | 0.70 | 3085 |

شکل 2-3-1-2 : ماتریس آشفتگی و گزارش طبقه بند

### درخت تصمیم ( Max Depth =5 )

تمامی مراحل ذکر شده برای قسمت قبل را این بار برای عمق 5 تکرار می‌کنیم .

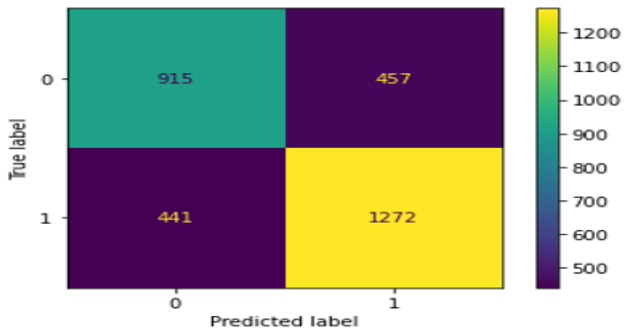
بعد از fit کردن مدل ، می‌توانیم می‌توانیم مثل قسمت قبل درصد اهمیت هر کدام از ویژگی ها را اعلام کنیم:

| | Feature Importance |
|---|---|
| **Recidivism Reporting Year** | 0.855577 |
| **Release Type** | 0.056109 |
| **Main Supervising District** | 0.036283 |
| **Age At Release** | 0.021499 |
| **Convicting Offense Type** | 0.015254 |
| **Part of Target Population** | 0.005698 |
| **Convicting Offense Classification** | 0.003474 |
| **Race - Ethnicity** | 0.003065 |
| **Convicting Offense Subtype** | 0.003041 |
| **Fiscal Year Released** | 0.000000 |

شکل 1-2-3-2 : درصد اهمیت ویژگی های مدل در آموزش درخت تصمیم

طبق نتیجه گیری که در بخش قبل کردیم ، با افزایش عمق از 3 به 5 ، درصد اهمیت یکی از دو ویژگی از 70 به 85 می‌رسد و ویژگی دوم که در اینجا "Fiscal Year Released" می‌باشد دارای اهمیت صفر می‌باشد .

در نهایت ماتریس آشفتگی و گزارش طبقه بند را بدست می‌آوریم:

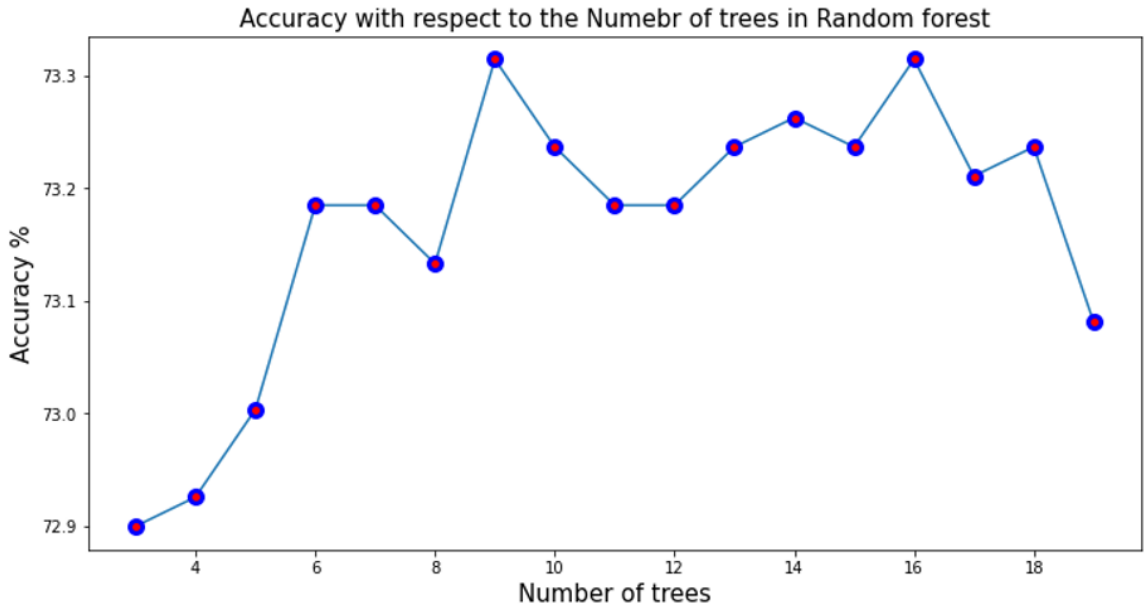



| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.67 | 0.67 | 0.67 | 1372 |
| 1 | 0.74 | 0.74 | 0.74 | 1713 |
| accuracy | | | 0.71 | 3085 |
| macro avg | 0.71 | 0.70 | 0.70 | 3085 |
| weighted avg | 0.71 | 0.71 | 0.71 | 3085 |

شکل 2-2-3-2 : ماتریس آشفتگی و گزارش طبقه بند

### جنگل تصادفی

بعد از fit کردن مدل ، نوداری از دقت طبقه بند بر حسب تعداد درخت انتخابی رسم می‌کنیم :



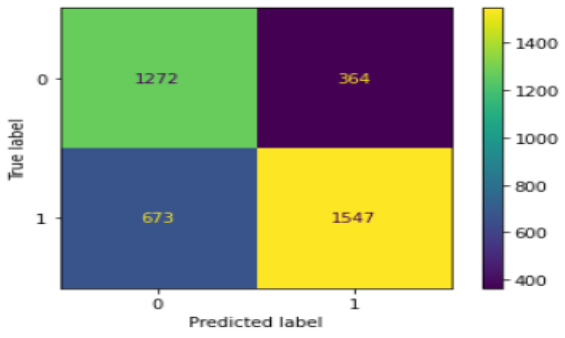

شکل 1-3-3-2 : دقت داده تست به ازای تعداد درخت

همانطور که در قسمت ب هم نتیجه رفتیم ، با افزایش تعداد درخت ها لزوما دقت طبقه بند افزایش پیدا نمی‌کند.

در نهایت ماتریس آشفتگی و گزارش طبقه بند برای عمق 3 و تعداد درخت 3 بصورت زیر می‌باشد :

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.78 | 0.65 | 0.71 | 1945 |
| 1 | 0.70 | 0.81 | 0.75 | 1911 |
| accuracy | | | 0.73 | 3856 |
| macro avg | 0.74 | 0.73 | 0.73 | 3856 |
| weighted avg | 0.74 | 0.73 | 0.73 | 3856 |

0.7310684647302904




شکل 2-3-3-2 : ماتریس آشفتگی و گزارش طبقه بند

## نزدیکترین همسایه

### کا-همسایه نزدیک

ابتدا تابعی تعریف می‌کنیم که فاصله اقلیدسی بین دو Sample را پیدا کند .

سپس تابعی می‌نویسیم که با گرفتن داده آموزش و همچنین یک Sample از داده تست ، k نزدیکترین Sample های داده آموزش به داده تست دلخواه را پیدا کند . برای مثال در زیر 3 نزدیکترین داده در مجموعه آموزش به Sample دوم در مجموعه تست آورده شده است :

```
print("The sample is :{}\n".format(X_test.values[1]))
get_neighbors(X_train.values, y_train.values, X_test.values[1], 3, distance=distance)
```

```
The sample is :[1.279e+01 2.670e+00 2.480e+00 2.200e+01 1.120e+02 1.480e+00 1.360e+00
 2.400e-01 1.260e+00 1.080e+01 4.800e-01 1.470e+00 4.800e+02]

[(array([1.184e+01, 2.890e+00, 2.230e+00, 1.800e+01, 1.120e+02, 1.720e+00,
         1.320e+00, 4.300e-01, 9.500e-01, 2.650e+00, 9.600e-01, 2.520e+00,
         5.000e+02]), 22.02181191455417, 2),
 (array([1.27e+01, 3.87e+00, 2.40e+00, 2.30e+01, 1.01e+02, 2.83e+00,
         2.55e+00, 4.30e-01, 1.95e+00, 2.57e+00, 1.19e+00, 3.13e+00,
         4.63e+02]), 22.072559887788277, 2),
 (array([1.35e+01, 3.12e+00, 2.62e+00, 2.40e+01, 1.23e+02, 1.40e+00,
         1.57e+00, 2.20e-01, 1.25e+00, 8.60e+00, 5.90e-01, 1.30e+00,
         5.00e+02]), 23.036019621453704, 3)]
```

شکل 1-1-3 : نمایی از خروجی تابع get_neighbors

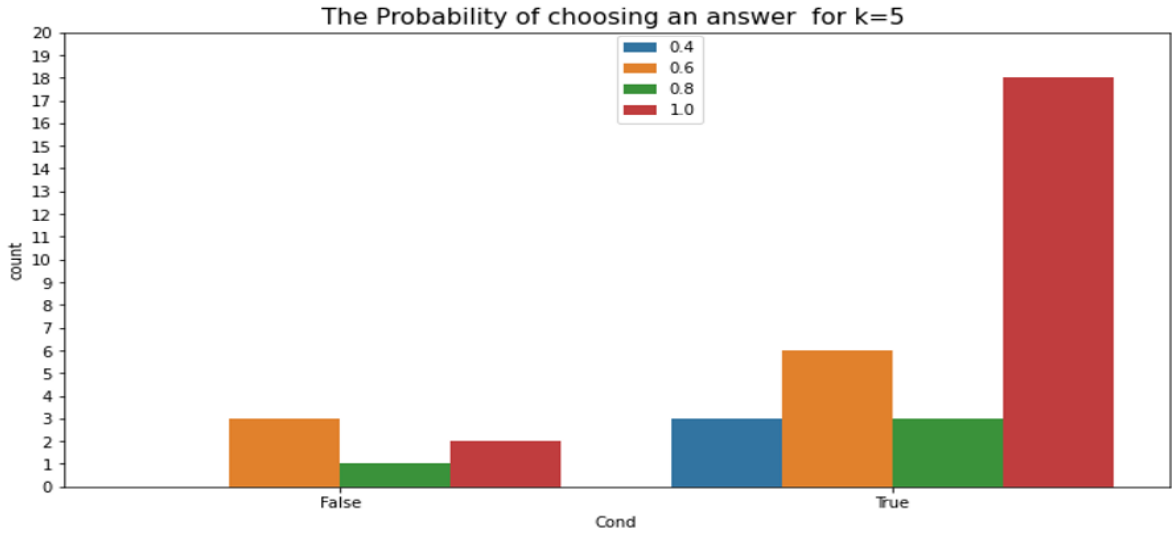
در ادامه تابعی می‌نویسیم که طبق لیبل های همسایه های بدست آمده در اطراف داده تست ، لیبل داده تست را تخمین بزند . همچنین احتمال انتخاب لیبل مورد نظر بر حسب تعداد همسایه های هر کلاس در اطراف داده تست بدست می‌آوریم .

در نهایت برای تمامی داده های تست ، لیبل متناظر و احتمال انتخاب آن لیبل را برای مجموعه تست بدست می‌آوریم و در یک دیتاست ذخیره می‌کنیم . 5 سطر اول این دیتاست به صورت زیر است :

| | Label | Predict | Probability |
|---|---|---|---|
| **0** | 1.0 | 1.0 | 1.0 |
| **1** | 3.0 | 2.0 | 0.6 |
| **2** | 2.0 | 2.0 | 1.0 |
| **3** | 1.0 | 1.0 | 1.0 |
| **4** | 2.0 | 2.0 | 1.0 |
| **5** | 2.0 | 2.0 | 0.8 |

شکل 3-1-2 : دیتاست حاصل از آموزش بر روی مجموعه تست

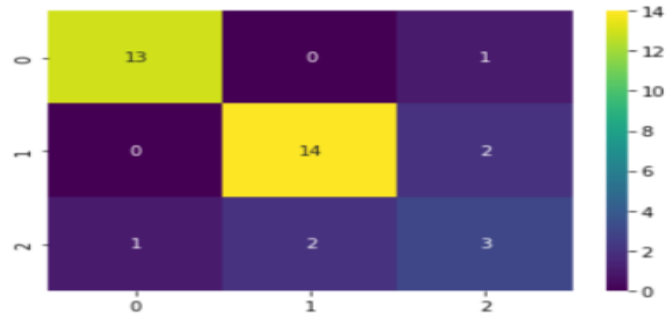
دلیل تشکیل دیتاست بالا ، برای این است که متوجه شویم برای هر داده تست با چه درصد اطمینانی لیبل را به داده assign می‌کنیم . بسته به این که درست حدس زده باشیم یا غلط ، نمودار زیر را با قرار دادن پارامتر "hue" بر روی ستون احتمال ، رسم می‌کنیم :




شکل 3-1-3 : میزان اطمینان از انتخاب لیبل برای مجموعه تست

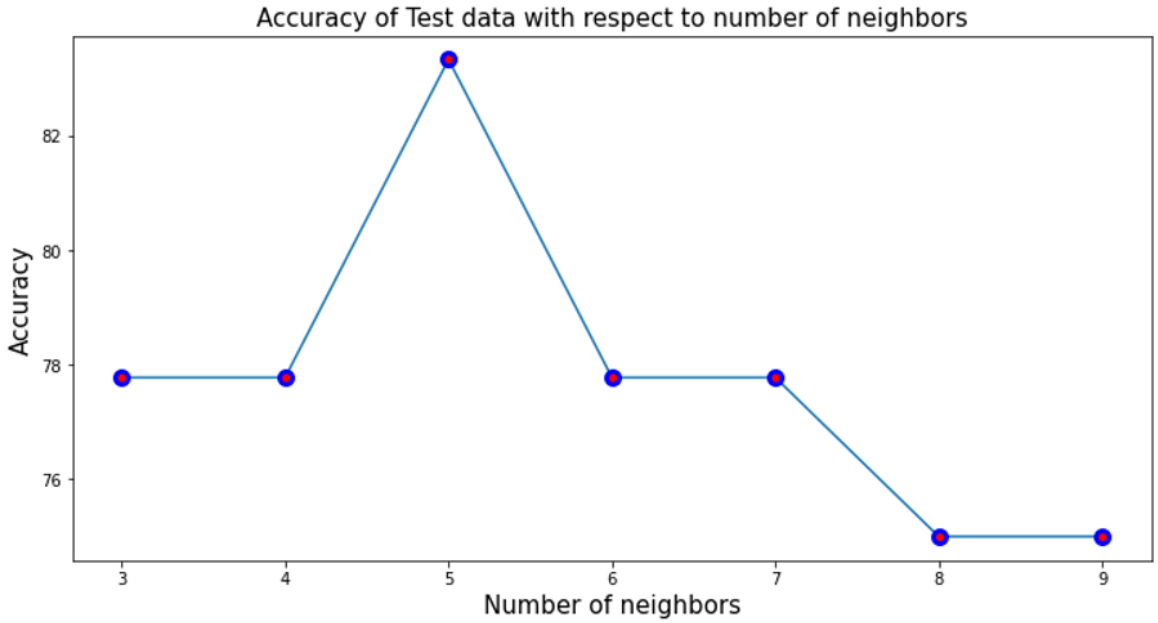
همانطور که از نتایج بالا مشخص است ، برای هنگامی که لیبل درست را حدس زدیم ، تنها در 3 حالت با احتمال 0.4 انتخاب کردیم . همچنین هنگامی که لیبل غلط را حدس زدیم ، در دو حالت با احتمال 1 ( قطع ) تصمیم گیری کردیم.در نهایت ماتریس آشفتگی حاصل از آموزش و گزارش طبقه بند به صورت زیر است :

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 1 | 0.93 | 0.93 | 0.93 | 14 |
| 2 | 0.88 | 0.88 | 0.88 | 16 |
| 3 | 0.50 | 0.50 | 0.50 | 6 |
| accuracy | | | 0.83 | 36 |
| macro avg | 0.77 | 0.77 | 0.77 | 36 |
| weighted avg | 0.83 | 0.83 | 0.83 | 36 |



شکل 3-1-4 : ماتریس آشفتگی و گزارش طبقه بند

در نهایت نمودار دقت طبقه بند را بر اساس تعداد نزدیکترین همسایه رسم می‌کنیم :




شکل 3-1-5 : دقت طبقه بند بر حسب نزدیکترین همسایه

طبق نتایج بالا ، برای k=5 ، بهترین دقت را ( 83.33 ) بدست می‌آوریم .

## یادگیری بر اساس معیار

در ادامه دو روش یادگیری را برسی می‌کنیم :

### 1. LMNN (Largest Margin Nearest Neighbor)

LMNN یک روش آماری در ML هست که فاصله Mahanalobis که بصورت $\|(x_j - x_i)^T C^{-1}(x_j - x_i)\|$ که در آن C ماتریس کواریانس هست را به صورت Supervised یاد می‌گیرد تا دقت طبقه بند نزدیکترین همسایه را افزایش دهد . برای هر Sample دلخواه ، دو نوع داده در نظر می‌گیریم :

- Target neighbor: این نقاط قبل اینکه داده را به فضای جدید انتقال دهیم ، تعیین شده اند. در واقع k همسایه مربوط به $x_i$ که در فاصله D از آن گرفته اند و با $x_i$ لیبل مشترکی دارند .
  *هدف این است که این k همسایه ، نزدیکترین همسایه به $x_i$ در فضای جدید باشند .
- Imposters: این نقاط در همسایگی $x_i$ با لیبل متفاوت قرار دارند. هدف این است که نقاط در دورترین همسایگی با $x_i$ قرار بگیرند.

حال هدف ما پیدا کردن ماتریس M است که فاصله بین دو داده در فضای جدید به صورت زیر بدست بیاید :

$$d(x_i, x_j) = (x_i - x_j)^T M (x_i - x_j)$$

*دقت کنید که $x_i, x_j$ بردار هایی در فضای n بعدی هستند. در واقع فاصله $d(x_i, x_j)$ بر این اساس باید تعیین شود که فاصله بین $x_i$ و Target neighbor کمینه شود و فاصله بین $x_i$ و Imposter بیشینه شود .

با در نظر گرفتن Target neighbor بصورت $x_l$ و Imposter به صورت $x_j$ ، شرط زیر باید برقرار باشد :

$$d(x_i, x_l) - d(x_i, x_j) \geq 1$$

به این منظور دو ترم برای تابع هزینه تعریف می‌کنیم :

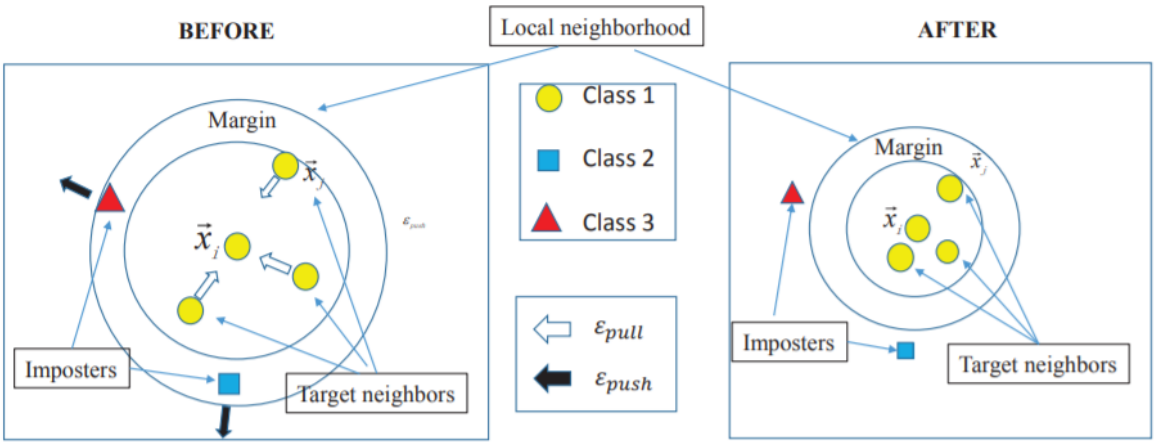
- ترم اول فاصله زیاد بین $x_i, x_j$ جریمه ( penalize ) می‌کند . ( $\varepsilon_{Pull}$ )
- ترم دوم که فاصله کم بین $x_i, x_l$ جریمه می‌کند . ( $\varepsilon_{Push}$ )

$$\varepsilon_M = (1-\lambda) \sum_i \sum_{j \in S_i} d(x_i, x_j) + \lambda \sum_i \sum_{j \in S_i, l \in P_i} (1 + d(x_i, x_j) - d(x_i, x_l))$$

$S_i \in$ Target neighbor , $P_i \in$ Imposter

که $\lambda$ بزرگتر از 0 می‌باشد و تعادلی بین دو ترم اول و دوم ایجاد می‌کند.

نمایی از تاثیر دو ترم اول و دوم را در شکل زیر می‌بینیم :




شکل 1-1-2-3 : نقش ترم اول ( $\varepsilon_{Pull}$ ) و دوم ( $\varepsilon_{Push}$ ) تابع هزینه

در زیر ابتدا به داده های آموزش آنرا fit Transform می‌کنیم و سپس بر روی داده تست Transform می‌کنیم . ( *نکته مهم اینجا این است که ما بر داده تست تنها Transform می‌کنیم ، زیرا اگر fit هم کرده باشیم با پدیده Data Leakage مواجه می‌شویم که نتایج خوبی ندارد )

در زیر ماتریس M را نیز بدست آوردیم که به ابعاد (13,13) می‌باشد . همچنین برای اینکه رابطه اولیه را صحت سنجی کنیم ، فاصله دو داده در فضای انتقال یافته را با فاصله بدست آمده از تبدیل اولیه که در فضای اصلی بود ، مقایسه می‌کنیم و می‌بینیم که نتایج یکی می‌باشد .

```
X_tr_new=lmnn.fit_transform(X_train,y_train)
X_test_new=lmnn.transform(X_test)
#X_test_metric,X_test_metric=lmnn.transform(X_test,y_test)
```

```
lmnn.get_mahalanobis_matrix()
```

```
(13, 13)
```

```
(X_train.iloc[0]-X_train.iloc[1]).T @ lmnn.get_mahalanobis_matrix() @ (X_train.iloc[0]-X_train.iloc[1])
```

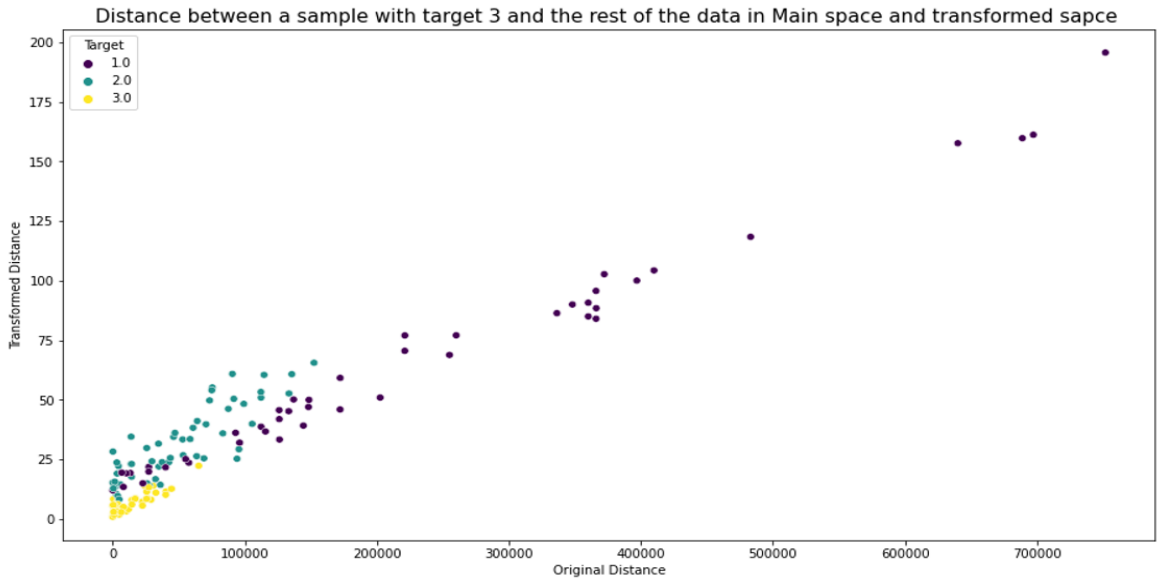
```
21.955137907636434
```

```
(X_tr_new[0]-X_tr_new[1]).T @ (X_tr_new[0]-X_tr_new[1])
```

```
21.95513790763644
```

شکل 2-1-2-3 : ماتریس M و مقایسه فاصله بدست آمده در فضای جدید با تبدیل در فضای اولیه

همچنین برای اینکه دیدی از نحوه Scale فواصل در فضای جدید نسبت به فضای قبلی داشته باشیم ، برای یک Sample دلخواه با لیبل 3 ، فواصل دو به دوی آنرا با بقیه Sample ها در فضای اصلی و تبدیل یافته حساب می‌کنیم و رسم می‌کنیم :



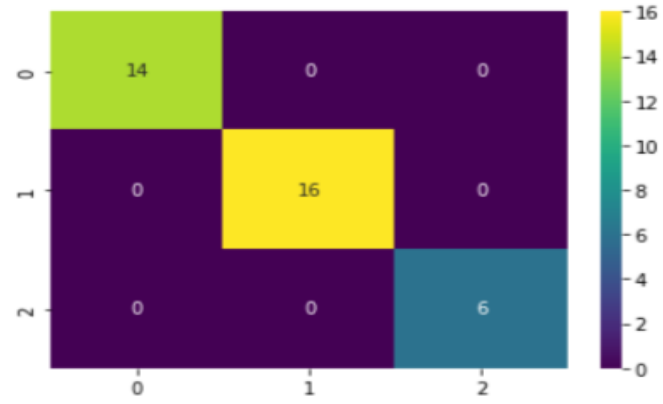

شکل 2-1-2-3 : نحوه Scale فاصله بین یک Sample و سایر Sample ها در فضای جدید و تبدیل یافته

همانطور که مشاهده می‌کنید ، y-axis که در واقع فواصل Scale شده را نشان می‌دهد باعث شده که داده های زرد رنگ  با لیبل 3 ) در نزدیکترین فاصله با داده تست ما قرار گیرند و دو لیبل دیگر که Imposter های ما هستند در فاصله دورتری نسبت به لیبل 1 قرار گیرند .

*در نهایت برای k=5 ، ماتریش آشفتگی و گزارش طبقه بند به صورت زیر می‌باشد . همانطور که مشخص است ، در فضای جدید داده های تست را برای همسایه به خوبی لیبل بندی کردیم و دقت 100 درصد گرفتیم :

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 1 | 1.00 | 1.00 | 1.00 | 14 |
| 2 | 1.00 | 1.00 | 1.00 | 16 |
| 3 | 1.00 | 1.00 | 1.00 | 6 |
| accuracy | | | 1.00 | 36 |
| macro avg | 1.00 | 1.00 | 1.00 | 36 |
| weighted avg | 1.00 | 1.00 | 1.00 | 36 |

The test accuracy is : 100.0




شکل 4-1-2-3 : ماتریس آشفتگی و گزارش طبقه بند برای LMNN و k=5

## 2. LFDA (Local Fisher Discriminant Analysis)

میدانیم Scatter Matrix به صورت زیر محاسبه می‌شود که کارکردی مشابه ماتریس کواریانس دارد:

$$S = \sum_{k=1}^{n} (x_k - m)(x_k - m)^T$$

LFDA از سودمندی هر دو روش FDA و LPP استفاده می‌کند که به مختصر هر کدام را در زیر شرح می‌دهیم:

### ➢ FDA:

طبق این تعریف ، دو ماتریس درون کلاس $S^{\omega}$ و برون کلاسی $S^{b}$ را بصورت زیر تعریف می‌کنیم :

$$\begin{cases} S^{\omega} = \sum_{i=1}^{l} \sum_{j=1, j\#i}^{l} (x_j - \mu_i)(x_j - \mu_i)^T \\ S^{b} = \sum_{i=1}^{l} n_i (\mu_i - \mu)(\mu_i - \mu)^T \end{cases}$$

که $\mu_i$ میانگین درون کلاس i و $\mu$ میانگین تمام Sample ها می‌باشد .

ماتریس $T_{FDA}$ که از تبدیل FDA بدست می‌آید به صورت زیر بدست می‌آید :

$$T_{FDA} = argmax_{\ T \in R^{d*m}} (T^t\ S^{\omega} T)^{-1} T^t S^b T\ , t : Transpose$$

در واقع ماتریس T به گونه ای پیدا می‌شود که فاصله بین کلاسی بیشینه و فاصله درون کلاسی کمینه شود . ستون های ماتریس $T_{FDA}$ در واقع بردارهای ویژه متناظر با مقدار ویژه بدست آمده از رابطه زیر می‌باشد :

$$T_{FDA} = (\varphi_1 | \varphi_2 | \dots | \varphi_2 |)\ \ s.t \rightarrow S^b \varphi = \lambda S^{\omega} \varphi$$

### ➢ LPP:

فرض کنید A ماتریس Affinity ( ماتریس شباهت ) باشد که درایه (i,j) آن شباهت بین $x_i, x_j$ را بیان می‌کند به گونه ای که اگر $x_j$ در k نزدیکترین همسایه $x_i$ قرار داشته باشد ، $A_{ij} = 1$ باشد و در غیر این صورت صفر می‌باشد .

ماتریس تبدیل LPP به صورت زیر بدست می‌آید :

$$T_{LLP} = argmin_{\ T \in R^{d*m}} \sum_{i,j=1}^{n} A_{ij} \left\| T^t x_i - T^t x_i \right\|^2 , s.t\ T^t X D X^t\ T = I$$

که D ماتریس قطری است که درایه i ام آن به صورت زیر بدست می‌آید :

$$D_{i,i} = \sum_{j=1}^{n} A_{ij}$$

در واقع ماتریس $T_{LLP}$ به گونه ای بدست می‌آید که جفت داده های نزدیک در فضای اصلی در فضای تبدیل یافته نیز نزدیک همدیگر باشند.

در واقع کاربر LPP در زمان مواجه با Multimodality می‌باشد . برای مثال فرض کنید که برای Disease Diagnose ، احتمال وجود چند دلیل برای بیماری وجود دارد . LPP کمک می‌کند که ساختار کلی دیتا در فضای جدید حفظ شود.

*مانند قسمت قبل ستونهای $T_{LLP}$ دارای بردار ویژه هایی است ، که با حل معادله مقادیر ویژه مربوطه بدست می‌آید .

*در نهایت برای آنکه هر دو Concept بالا را با هم ترکیب کنیم ، ماتریس درون کلاس و برون کلاسی را بصورت زیر تعریف می‌کنیم :

$$S^{\omega} = \sum_{i,j=1}^{l} W_{i,j}^{\omega}(x_i - x_j)(x_i - x_j)^T$$

$$S^{b} = \sum_{i=1}^{l} W_{i,j}^{b}(x_i - x)(x_i - x_j)^T$$

$$,s.t \begin{cases} W_{i,j}^{\omega} = \begin{cases} \frac{A_{ij}}{n_l} & if\ y_i = y_j = l \\ 0 & O.w \end{cases} \\ W_{i,j}^{b} = \begin{cases} \frac{A_{ij}}{(\frac{1}{n} - \frac{1}{n_l})} & if\ y_i = y_j = l \\ 1/n & O.w \end{cases} \end{cases}$$

که n تعداد کل داده ها و $n_l$ تعداد داده های درون دسته $l$ می‌باشد .

در نهایت ماتریس $T_{LFDA}$ به صورت زیر بدست می‌آید :

$$T_{LFDA} = tr(argmax_{T \in R^{d*m}}(T^t S^{\omega} T)^{-1} T^t S^b T\ ) , t : Transpose$$

*نکته مهم در استفاده از LFDA ، قابلیت استفاده از Kernel Trick هست ، که محاسبات در بعد بالا را بسیار ساده می‌کند.

مانند قسمت قبل و اینبار T را نیز بدست آوردیم که به ابعاد (13,13) می‌باشد . همچنین برای اینکه رابطه اولیه را صحت سنجی کنیم ، فاصله دو داده در فضای انتقال یافته را با فاصله بدست آمده از تبدیل اولیه که در فضای اصلی بود ، مقایسه می‌کنیم و می‌بینیم که نتایج یکی می‌باشد.

```
(X_train.iloc[0]-X_train.iloc[1]).T @ lfda.get_mahalanobis_matrix() @ (X_train.iloc[0]-X_train.iloc[1])
```

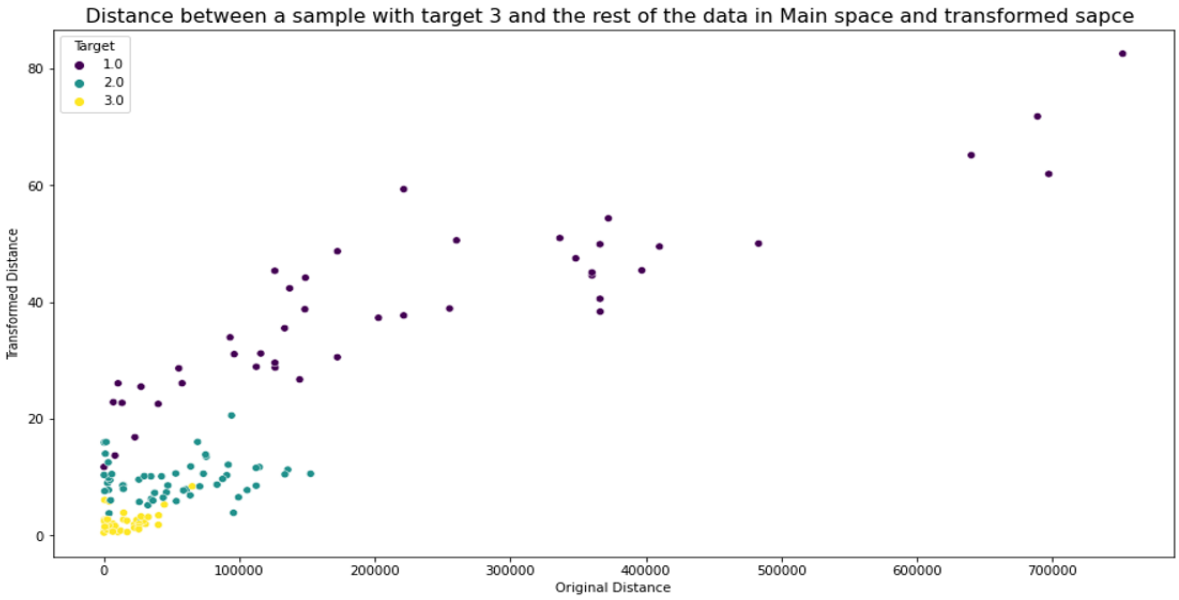
```
6.261669350977045
```

```
(X_tr_new1[0]-X_tr_new1[1]).T @ (X_tr_new1[0]-X_tr_new1[1])
```

```
6.26166935097704
```

شکل 3-2-2-1 : ماتریس T و مقایسه فاصله بدست آمده در فضای جدید با تبدیل در فضای اولیه

همچنین برای اینکه دیدی از نحوه Scale فواصل در فضای جدید نسبت به فضای قبلی داشته باشیم ، برای یک Sample دلخواه با لیبل 3 ، فواصل دو به دوی آنرا با بقیه Sample ها در فضای اصلی و تبدیل یافته حساب می‌کنیم و رسم می‌کنیم :



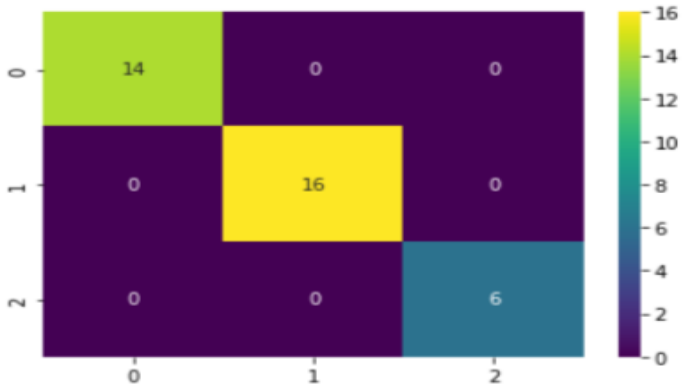

شکل 3-2-2-2 : نحوه Scale فاصله بین یک Sample و سایر Sample ها در فضای جدید و تبدیل یافته

همانطور که از شکل بالا مشخص است ، برای دادگان زرد( لیبل 3 ) فاصله درون کلاسی کمینه شده و فاصله بین کلاسی دادگان زرد با سبز و بنفش(لیبل های 1 و 2 ) بیشینه شده است .

*در نهایت برای k=5 ، ماتریش آشفتگی و گزارش طبقه بند به صورت زیر می‌باشد. همانطور که مشخص است ، در فضای جدید داده های تست را برای همسایه به خوبی لیبل بندی کردیم و دقت 100 درصد گرفتیم :

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 1 | 1.00 | 1.00 | 1.00 | 14 |
| 2 | 1.00 | 1.00 | 1.00 | 16 |
| 3 | 1.00 | 1.00 | 1.00 | 6 |
| accuracy | | | 1.00 | 36 |
| macro avg | 1.00 | 1.00 | 1.00 | 36 |
| weighted avg | 1.00 | 1.00 | 1.00 | 36 |

The test accuracy is : 100.0



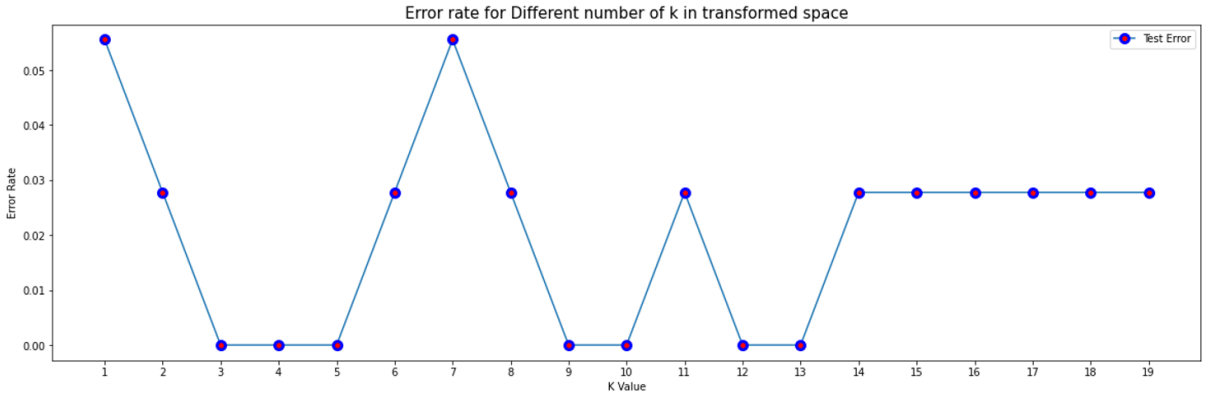

شکل 3-2-2-3 : ماتریس آشفتگی و گزارش طبقه بند برای LFDA و k=5

## تعداد مطلوب همسایه برای هر متریک

در نهایت برای اینکه تعداد همسایه مناسب را در فضای تبدیل یافته هر حالت پیدا کنیم ، نمودار خطا بر حسب تعدا همسایه ها را می‌کشیم :

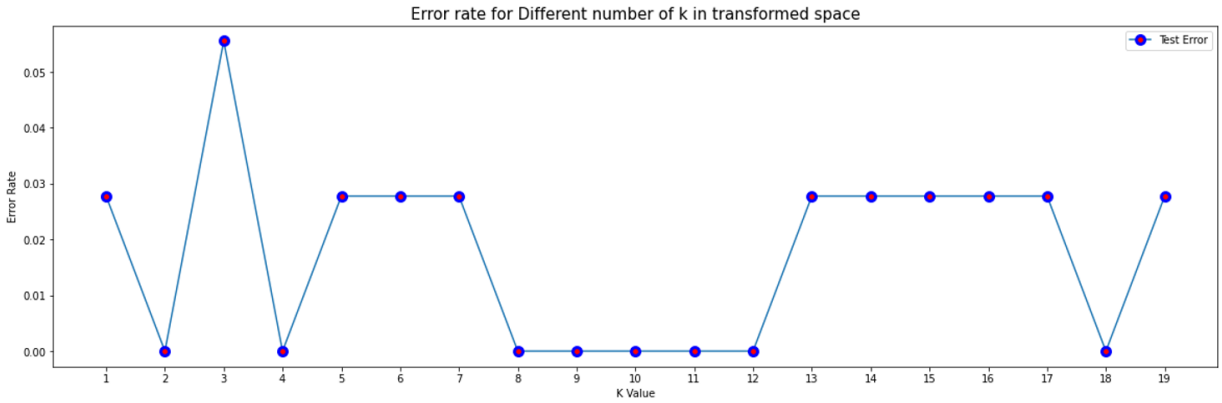
*نکته مهم اینکه ، مقدار همسایه بصورت پارامتری هم در تابع LFDA/LMNN و هم در تابع KNN داده میشود

### LMNN




شکل 3-2-2-4 : نمودار خطا بر حسب تعداد همسایه برای متریک LMNN

## LFDA




شکل 5-2-2-3 : نمودار خطا بر حسب تعداد همسایه برای متریک LFDA

همانطور که از نمودار ها مشخص است ، برای LFDA می‌توانیم برای تعداد همسایه های بزرگتری نسبت به LMNN ، خطای صفر داشته باشیم ولی هر دوی آنها تقریبا بعد از k=13 دچار خطا می‌شوند.

این به این معنا است که مدل LDFA قابلیت عمومیت بخشی بهتری نسبت به LMNN دارد ولی هر دو روش عملی و کاربردی هستند.