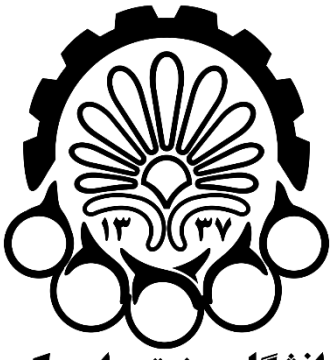
به نام خدا



دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

تمرین درس یادگیری ماشین-سری دوم

فردین آیار

شماره دانشجویی: ۹۹۱۳۱۰۴۰

استاد: دکتر ناظرفرد

دانشکده کامپیوتر- پاییز ۹۹

سوالات تشریحی

۱) برای انتخاب مقدار بهینه k روش های مختلفی ارائه شده است؛ اگرچه مقدار مناسبی به صورت عمومی نمی توان برای آن تعیین کرد. یک روش مناسب استفاده از یک مجموعه مجزا از train و test، به عنوان validation است. این مجموعه که با نسب معینی از داده‌ها جدا می‌شود، برای انتخاب k مناسب استفاده می شود. عیب این روش این است که داده ها را به سه دسته تقسیم می‌کند، بنابراین تعداد عناصر هر دسته کم می‌شود. راه‌حل این مشکل استفاده از از روش k-fold cross validation است. به این صورت که مجموعه داده ها را به K دسته تقسیم می کنند و هر بار از یک دسته به عنوان مجموعه آزمون و از سایر دسته‌ها به عنوان مجموعه آموزش استفاده می‌شود. به ازای مقادیر مختلف k عمل فوق را انجام می‌دهیم و بدین‌وسیله مقدار بهینه k را تعیین می کنیم.

۲)

الف) به ازای k های مختلف، تعداد داده‌هایی که درست یا نادرست برچسب خورده اند را مشخص می کنیم. نتایج در جدول زیر ارائه می‌شود: (از مقادیر زوج k، به علت تصادفی شدن انتخاب در بعضی موارد، صرف‌نظر شده است)

K	درست	نادرست
۱	۴	۱۰
۳	۸	۶
۵	۱۰	۴
۷	۱۰	۴
۹	۰	۱۴

بهترین خطا برای k ۵ و ۷ است و مقدار دقت متناظر با آنها ۰.۹۰۹ می باشد.

ب) مقادیر کوچک k باعث بیش برازش می‌شود و مرزهای تصمیم را پیچیده می‌کند. همچنین در صورت وجود داده های نویزی، دقت الگوریتم را کاهش می‌دهد. در مقابل، مقادیر بزرگ k باعث کم برازش می‌شود که این مورد نیز باعث افزایش خطا می شود.

ج) برای k=5 خروجی منفی و برای k=7 خروجی مثبت می باشد. با توجه به شکل داده ها، به نظر می‌رسد برچسب مثبت صحیح است.

۳) الگوریتم های پارامتریک برای یادگیری تابع خروجی ، تعداد ثابتی پارامتر در نظر می‌گیرند و با افزایش داده‌های آموزش، تعداد این پارامترها افزایش نمی‌یابد. حال آنکه الگوریتم های غیرپارامتریک، فرضیات ثابتی در مورد تعداد پارامترهای تابع خروجی ندارند و با افزایش داده های آموزشی، تعداد پارامترهای آن ممکن است افزایش یابد. بنابراین الگوریتم های knn و

درخت تصمیم هر دو غیر پارامتریک هستند. در واقع این الگوریتم ها دارای یک سری هایپرپارامتر هستند.(به عنوان مثال تعداد همسایه ها یا عمق درخت تصمیم)

۴) مدل های Generative بر اساس توزیع هر یک از کلاس‌ها، برچسب داده ها را تخمین می زنند. به بیان دیگر، این مدل ها به توزیع داده ها اهمیت می دهد و بر اساس آن، برچسب داده ها را با احتمال شرطی تعیین می کنند. از آنجایی که توزیع کلاس ها تعیین می شود، این مدل ها قادر به تولید نمونه های جدید هستند.

در مقابل، مدل های Discriminative به نحوه تولید داده ها و توزیع کلاسها اهمیتی نمی‌دهد و سعی می کند با ایجاد مرزهای تصمیم، برچسب ها را تعیین کند. بدیهی است که در چنین حالتی مدل توانایی ایجاد داده های نمونه جدید را ندارد.

بر اساس این توضیحات روش است که هر دو الگوریتم knn و درخت تصمیم Discriminative هستند. زیرا با تعیین توزیع داده ها سروکار ندارند و تنها آنها را بر اساس مرزهایی، برچسب می‌زنند.

۵) به الگوریتم هایی که در زمان ورود داده های آموزش، عمل خاصی برای یادگیری انجام نمی‌دهند و تنها در زمان انجام آزمون، محاسبات خود را انجام می دهند، الگورتیم های تنبل گفته می شود. الگوریتم knn طبق این تعریف یک الگوریتم تنبل است؛ زیرا تا زمانی که برای برچسب زدن به یک داده فراخوانی نشود، هیچ یادگیری روی داده های آموزش انجام نمی شود.ویژگی اصلی این الگوریتم ها سرعت پایین آن ها در زمان آزمون است. البته در مورد الگوریتم knn می توان با انجام تغییراتی مانند تعیین مرز تصمیم یا تعیین محدوده نقاط، این موضوع را بهبود داد.

۶) هرس درخت تصمیم، با کاهش عمق درخت، باعث کاهش بیش برازش بر داده های آموزش می‌شود. در این روش، شاخه هایی که با توسعه بیش از حد گره های بالاتر ساخته و باعث بیش برازش شده اند، هرس می گردد. این هرس با توجه به زمان انجام آن، به دو دسته تقسیم می شود:

الف) پیش هرس: اجازه رشد به شاخه هایی که باعث بیش برازش می شوند را نمی‌دهد.

ب) پس هرس: بعد از دسته بندی و تشکیل درخت نهایی، شاخه های نامطلوب را حذف و باعث کاهش عمق آن می شود.

۷)

$$\begin{cases} S = (9^+, 5^-) \\ S_{youth} = (2^+, 3^-) \\ S_{middle} = (4^+, 0^-) \\ S_{senior} = (3^+, 2^-) \end{cases}$$

$$Gain(S, age) = Entropy(S) - \sum_{v \in Values(age)} \frac{|S_v|}{S} Entropy(S_v)$$

$$= 0.940 - \left(\frac{5}{14} \times 0.970 + \frac{4}{14} \times 0 + \frac{5}{14} \times 0.970\right) = 0.247$$

$$\begin{cases} S = (9^+, 5^-) \\ S_{high} = (2^+, 2^-) \\ S_{medium} = (4^+, 2^-) \\ S_{low} = (3^+, 1^-) \end{cases}$$

$Gain(S, income) = Entropy(S) - \sum_{v \in Values(income)} \frac{|S_v|}{S} Entropy(S_v) =$

$0.940 - \left(\frac{4}{14} \times 1 + \frac{6}{14} \times 0.918 + \frac{4}{14} \times 0.811\right) = 0.029$

$$\begin{cases} S = (9^+, 5^-) \\ S_{yes} = (3^+, 1^-) \\ S_{no} = (6^+, 4^-) \end{cases}$$

$$Gain(S, student) = Entropy(S) - \sum_{v \in Values(student)} \frac{|S_v|}{S} Entropy(S_v)$$

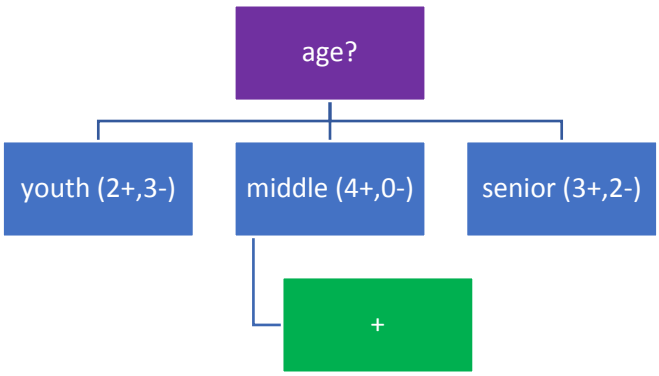
$$= 0.940 - \left(\frac{4}{14} \times 0.811 + \frac{10}{14} \times 0.97\right) = 0.015$$

$$\begin{cases} S = (9^+, 5^-) \\ S_{excellent} = (3^+, 3^-) \\ S_{fair} = (6^+, 2^-) \end{cases}$$

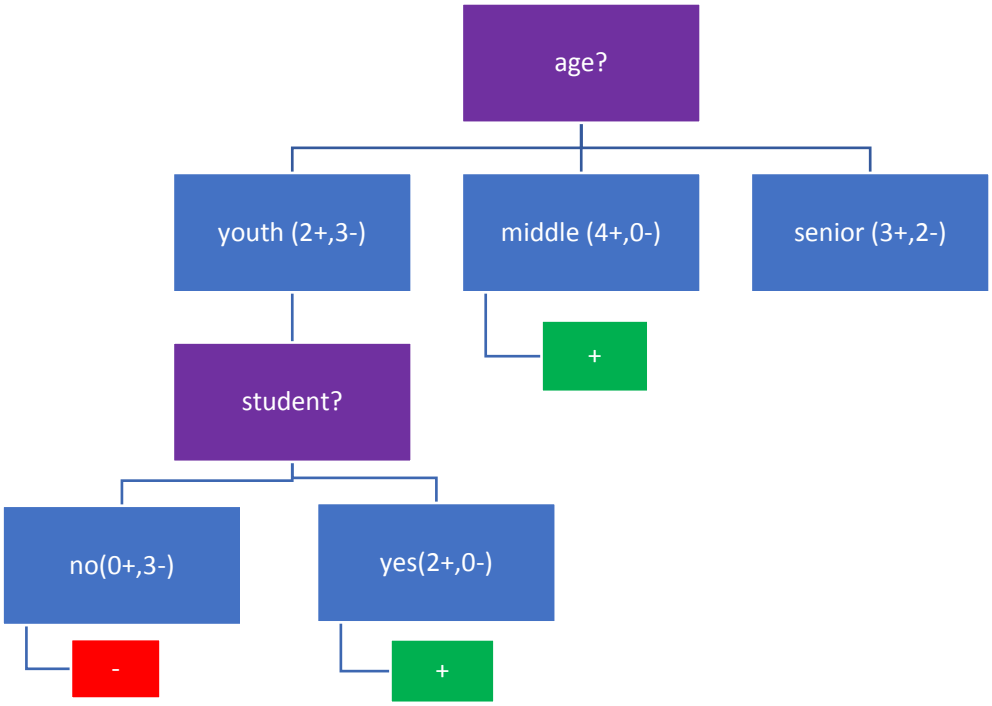
$Gain(S, credit) = Entropy(S) - \sum_{v \in Values(credit)} \frac{|S_v|}{S} Entropy(S_v) = 0.940 -$

$\left(\frac{6}{14} \times 1 + \frac{8}{14} \times 0.811\right) = 0.048$

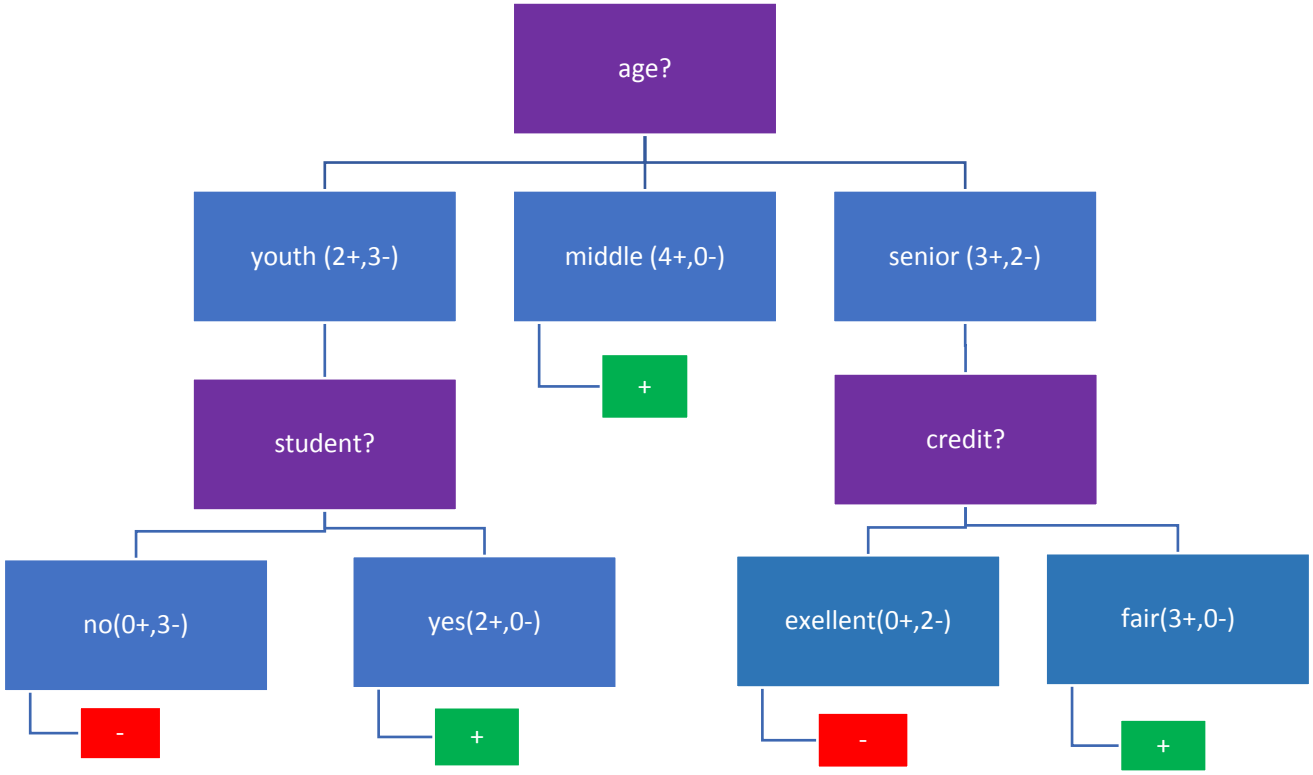
بنابراین age به عنوان ریشه درخت انتخاب می شود.



به زیرشاخه *youth* می پردازیم. بدون نیاز به محاسبات از روی جدول مشخص است ویژگی *student* برای آن حداکثر *gain* را دارد؛ زیرا هرجا مقدار آن *yes* بوده خروجی مثبت، و هرجا *no* بوده خروجی منفی بوده است.(برای *age = youth*) شکل درخت به روز شده به این صورت می باشد:



به سراغ زیر شاخه senior می‌رویم. برای این زیرشاخه نیز بدون نیاز به محاسبات، از روی جدول مشخص است که ویژگی credit بیشترین gain را دارد. بنابراین شکل نهایی درخت را می‌توان مستقیماً رسم کرد.

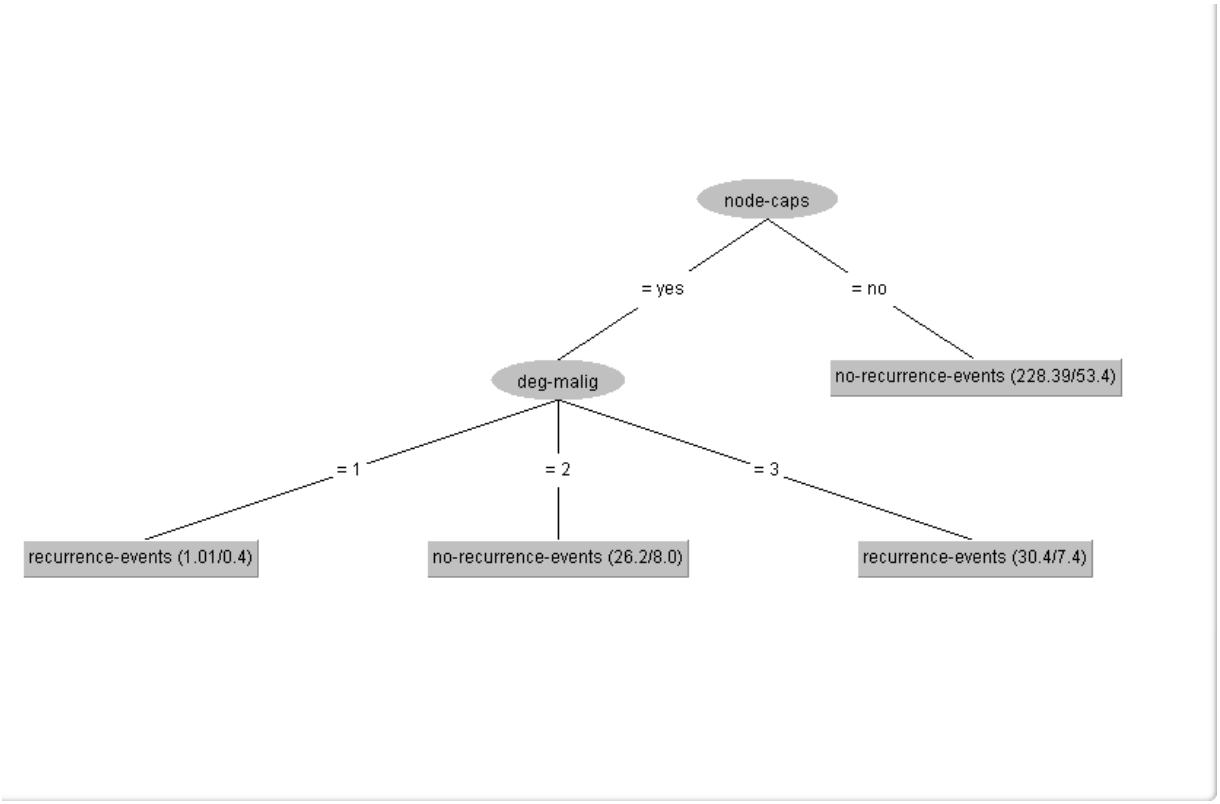


۸) با توجه به اینکه مرز تصمیم در درخت تصمیم، متشکل از خطوط عمودی و افقی (موازی محورهای مختصات) است؛ این مرز به صورت پلکانی در راستای خط جداکننده قرار می گیرد؛ و با افزایش مقدار K این شکل پلکانی همچنان حفظ می‌شود و کاملا بر خط موردنظر منطبق نمی‌شود. بنابراین با روش عادی درخت تصمیم و با عمق محدود، نمی‌توان مسائل جداپذیر خطی را بدون خطا دسته بندی کرد.

۹) این الگوریتم مجموعه ای از درخت هایی که بهم وابسته نیستند تشکیل می‌دهد و براساس سیستمی شبیه رای گیری، برچسب داده ها را با توجه به خروجی هر درخت تعیین می‌کند. در این روش، همانطور که گفته شد، درخت‌ها به هم وابسته نیستند بنابراین خطای همدیگر را پوشش می دهند؛ در نتیجه نیاز به افزایش عمق درخت برای کاهش خطا نیست. مستقل بودن درخت ها به این معنی است که تعداد و عمق ویژگی در درخت ها می تواند متفاوت باشد.

WEKA

۱) در تصویر زیر، شکل درخت تصمیم نشان داده شده است.



در ادامه ماتریس درهم‌ریختگی و سایر پارامترهای خواسته شده، ارائه شده است. برای ارائه اطلاعات خواسته شده، فرض می شود کلاس no-recurrence-event کلاس positive می باشد.

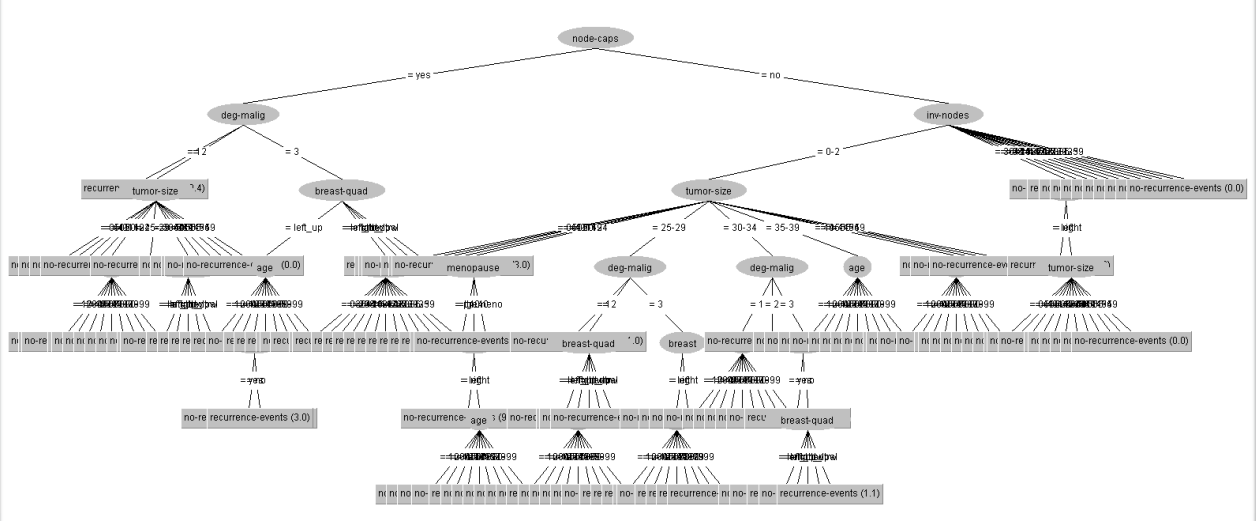
```
 a  b   <-- classified as
46  8 |  a = no-recurrence-events
23  9 |  b = recurrence-events
```

TP	TN	FP	FN	Precision	Recall	F-Measure
46	9	23	8	0.667	0.852	0.748

همانطور که مشاهده می‌شود، تعداد ویژگی های تاثیرگذار در درخت‌تصمیم، به نسبت تعداد کل ویژگی های موجود، بسیار کمتر است. این مورد، که به دلیل هرس شدن درخت اتفاق می‌افتد، نشان می‌دهد که تنها دو پارامتری که در درخت مشاهده می شوند، می توانند تا حد زیادی داده ها را دسته بندی کنند؛ و در واقع این دو پارامتر بیشترین تاثیر را در خروجی نهایی دارند. نکته دیگر اینکه مقدار Recall این درخت نسبت به Precision آن بالا است؛ یعنی داده های کلاس P را به خوبی

برچسب می‌زند، اما تعداد زیادی از داده ها کلاس N را نیز اشتباها P تشخیص می دهد. این مورد از ماتریس درهم‌ریختگی نیز کاملا مشخص است.

۲) این پارامتر هرس شدن درخت تصمیم را نشان می دهد. در این قسمت مقدار آن را true قرار می‌دهیم و درخت هرس نشده را نشان می دهیم.



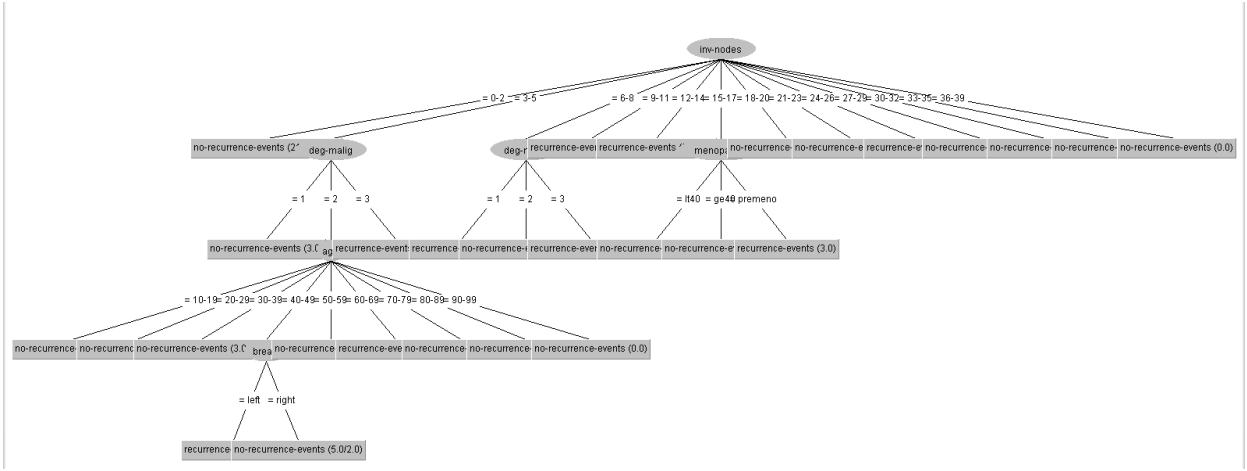
از آنجا که تعداد ویژگی ها در درخت هرس نشده بسیار زیاد است، شکل درخت تصمیم بسیار پیچیده شده و به درستی قابل نمایش نیست. در ادامه ماتریس درهم‌ریختگی و سایر پارامترها ارائه شده است.

```
 a  b   <-- classified as
41 13 |  a = no-recurrence-events
20 12 |  b = recurrence-events
```

TP	TN	FP	FN	Precision	Recall	F-Measure
41	12	20	13	0.672	0.759	0.713

به نظر می رسد هرس نکردن درخت تصمیم، به جز در مورد Precision، TN و FP که باعث اندکی بهبودی آن ها شده، در سایر پارامترها باعث بدتر شدن نتیجه شده است. این مورد نشان می‌دهد که هرس نکردن درخت باعث بیش برازش روی داده‌های آموزش شده است. بنابراین همان درخت قسمت قبل برای دسته‌بندی داده ها عملکرد مناسب‌تری داشته است.

۳) ابتدا با اعمال ۱۵ درصد نویز، درخت هرس شده را رسم می‌کنیم.

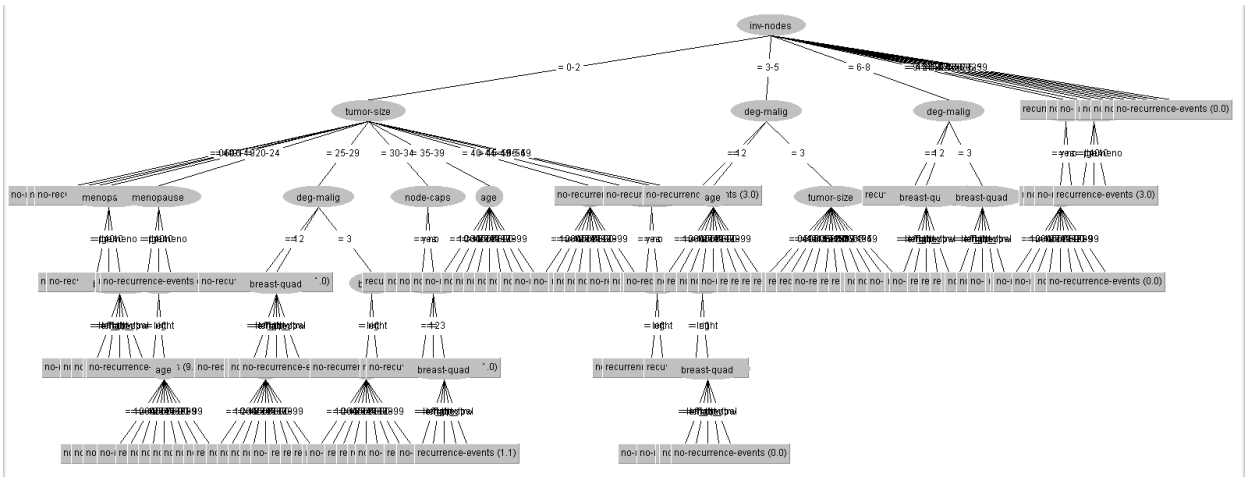



```
 a  b   <-- classified as
47  7 |  a = no-recurrence-events
23  9 |  b = recurrence-events
```

TP	TN	FP	FN	Precision	Recall	F-Measure
47	9	23	7	0.671	0.870	0.758

اعمال نویز باعث شده درخت تصمیم هرس شده نسبت به درخت تصمیم هرس شده بدون نویز، عمق بیشتری داشته باشد.

حال درخت هرس نشده را رسم می‌کنیم.




```
 a  b   <-- classified as
39 15 |  a = no-recurrence-events
19 13 |  b = recurrence-events
```

TP	TN	FP	FN	Precision	Recall	F-Measure
39	13	19	15	0.672	0.722	0.696

سوالات پیاده‌سازی

۱) کد مربوط به این قسمت در فایل knn_np.py قرار دارد. نام ستون ها، به صورت دستی در دیتاست وارد شده اند و فایل دیتاست تحت عنوان mammographic_masses.csv در پوشه dataset موجود است.

ویژگی هایی که دارای مقدار نامعلوم هستند با مقدار مد(mode) آن ویژگی در سایر داده ها جایگزین شده‌اند. این انتخاب با توجه به nominal بودن بیشتر ویژگی ها صورت گرفته است. همچنین ستون مربوط به سن، باتوجه به متفاوت بودن مقدار آن‌ها نسبت به سایر ویژگی‌ها، با استفاده از میانگین و انحراف معیار نرمال‌سازی شده است. لازم به ذکر است در هر مرحله، داده‌های آزمون با توجه به پارامترهای داده‌های آموزش نرمال شده اند.

ماتریس درهم‌ریختگی و روش k-fold cross validation به صورت تابعی پیاده شده اند و برای آن‌ها از کتابخانه آماده استفاده نشده است.

الف)

- K=1

accuracy = 0.76174

	predicted		
	0	1	
actual	403	113	0
	116	329	1

- K=3

accuracy = 0.80848

	predicted		
	0	1	
actual	424	92	0
	92	353	1

- K=5

accuracy = 0.81165

predicted

	0	1	
0	425	91	actual
1	90	355	

- K=7

accuracy = 0.80955

predicted

	0	1	
0	427	89	actual
1	94	351	

- K=15

accuracy = 0.81578

predicted

	0	1	
0	437	79	actual
1	98	347	

- K=30

accuracy = 0.81479

predicted

	0	1	
0	445	71	Actual
1	107	338	

همانطور که مشاهده می شود، با افزایش مقدار k، مقدار دقت الگوریتم، به جز در k=7 ، افزایشی بوده است. این افزایش تا k=15 ادامه داشته و پس از آن به نظر می‌رسد دوباره کاهشی شده است.این مورد می تواند به این دلیل باشد که با افزایش بیشتر k، مدل به سمت کم برازش حرکت می‌کند. همچنین با فرض کلاس صفر به عنوان کلاس positive، مقدار TP با افزایش مقدار k، افزایشی بوده است. در مجموع به نظر می‌رسد حتی مقدار k=5 برای این مسئله کافی می باشد.

ب) بهترین نتیجه مرحله قبل، k=15 می باشد.

- فاصله اقلیدسی

accuracy = 0.81578

predicted

	0	1	
0	437	79	actual
1	98	347	

- فاصله منهتن

accuracy = 0.82204

predicted

	0	1	
0	440	76	actual
1	95	350	

- فاصله کسینوسی

accuracy = 0.78770

predicted

	0	1	
actual	402	114	0
	90	355	1

ج) کد مربوط به این قسمت در فایل scikit_learn.py قرار دارد. نتایج زیر مربوط به کتابخانه آماده می باشد.

- K=1

accuracy = 0.74194

predicted

	0	1	
actual	385	131	0
	117	328	1

- K=3

accuracy = 0.80431

predicted

	0	1	
actual	423	93	0
	95	350	1

- K=5

accuracy = 0.80956

predicted

	0	1

15

428	88	0	actual
95	350	1	

- K=7

accuracy = 0.81684

predicted			
0	1		
432	84	0	actual
92	353	1	

- K=15

accuracy = 0.82099

predicted			
0	1		
441	75	0	actual
97	348	1	

- K=30

accuracy = 0.81999

predicted			
0	1		
448	68	0	Actual
105	340	1	

در مقایسه با نتایج قسمت الف،خروجی کتابخانه آماده تفاوت بسیار کمی دارد. این مورد می‌تواند به تفاوت های پیاده سازی مربوط باشد.به عنوان مثال مشخص نیست از بین نقاطی که فاصله یکسانی دارد، کدام نقطه انتخاب می شود. همچنین

متوسط زمان اجرا برای قسمت الف، حدود ۱۳.۵ ثانیه و برای کتابخانه آماده حدود ۰.۳ ثانیه می باشد. این نتایج نشان می‌دهد که کتابخانه آماده از نظر سرعت، برتری زیادی دارد.

۲) کد مربوط به این قسمت در فایل reg.py قرار دارد. داده های شافل شده در فایل regression.csv قرار دارند. طبق بررسی انجام شده، نرمال سازی داده ها در این مسئله، تاثیر خاصی بر خروجی ندارد؛ بنابراین از آن صرف نظر شده است.(کد مربوط به آن کامنت شده است)
مقدار k بهینه برای مینیمم شدن خطای آزمون: ۱۹
خطای آزمون : ۸۹.۴۰۹۸
خطای آموزش: ۶۱.۷۹۵۸

۳)کد مربوط به این قسمت در فایل dt.py قرار دارد. نام ستون ها به صورت دستی وارد شده اند و دیتاست های مربوطه در پوشه Dataset قرار دارند. ویژگی هایی که دارای مقدار نامعلوم هستند با مقدار مد(mode) آن ویژگی در سایر داده ها جایگزین شده‌اند. این انتخاب با توجه به nominal بودن بیشتر ویژگی ها صورت گرفته است.در برنامه پیاده سازی شده، ستون اول داده ها، به دلیل بی ارتباط بودن آن با خروجی، حذف شده است.
خروجی های مدنظر با استفاده از کتابخانه scikit learn به صورت زیر است.(برای محاسبه ماتریس درهم‌ریختگی و دقت الگوریتم، از توابع ساخته شده در سوال ۱ استفاده شده است)

accuracy = 0.915

	predicted		
	0	1	
actual	134	6	0
	11	49	1