داده کاوی

تمرین سوم

آرش حریرپوش
9731505

آرش حریرپوش
9731505

❖ بخش اول – مباحث تئوری و مسائل تشریحی

1. مزایا و معایب روش های دسته بندی مشتاق (مثل درخت تصمیم، بیزین، شبکه عصبی) و روش های دسته بندی تنبل (مثل KNN) را مقایسه کنید.

- دسته بندی تنبل زمان کمتری بر روی داده های آموزشی و زمان بیشتری بر روی داده های تست نسبت به دسته بندی مشتاق صرف می کند.
- با توجه به اینکه دسته بندی مشتاق شامل یک فرضیه یکتا برای کل دسته بندی می باشد، باعث می شود تا انجام این دسته بندی زمان زیادی به طول انجامد و سطح دسته بندی کاهش یابد. و نیاز است تا قبل از استفاده از دسته بندی های مشتاق مدل بر روی داده های آموزشی آموزش داده شود. (زمان آموزش در روش های دسته بندی مشتاق بیشتر است).
- با توجه به اینکه دسته بندی تنبل شامل سطح فرضیه بهتری می باشد، دقت این نوع دسته بندی ها بهتر می شود.
- در دسته بندی های تنبل با توجه به اینکه داده های آموزشی را باید ذخیره کنیم، هزینه ذخیره سازی افزایش می یابد و شامل ساختار های high indexing می شود.

(منبع: https://brainly.in/question/9289566#:~:text=K%20people%20helped-,Eager%20classification%20is%20seen%20to%20be%20much%20faster%20than%20the,any%20new%20tuples%20to%20classify.&text=accuracy%20of%20classification%20.-,in%20this%20classification%20trading%20tuples%20have%20to%20be%20stored%20increasing,involved%20high%20end%20indexing%20structures. و
https://cs.nyu.edu/~jcf/classes/g22.3033-002/slides/session7/ClassificationAndPrediction.pdf)

آرش حریرپوش
9731505

.2

الف) تراکنش های زیر را در نظر بگیرید. با فرض min_conf = 80 % , min_sup = 60% تمام frequent itemset ها را با استفاده از الگوریتم های Apriori و FP-growth پیدا کنید.

TID	Items_bought
T1	{M, O, N, K, E, Y}
T2	{D, O, N, K, E, Y}
T3	{M, A, K, E}
T4	{M, U, C, K, Y}
T5	{C, O, O, K, I, E}

با توجه به min_sup داریم:

$$\min\_sup = 60\% \Rightarrow \frac{\sigma(X \cup Y)}{N} = \frac{\sigma(X \cup Y)}{5} = 60\% \Rightarrow \text{count}_{min} = 3$$

با استفاده از الگوریتم Apriori، frequent itemset ها برابر است با:

1-itemset

Item	Count
{A}	1
{C}	2
{D}	1
{E}	4
{I}	1
{K}	5
{M}	3
{N}	2
{O}	3
{U}	1
{Y}	3

Frequent 1-itemset

Item	Count
{E}	4
{K}	5
{M}	3
{O}	3
{Y}	3

2-itemset

Item	Count
{E, K}	4
{E, M}	2
{E, O}	3
{E, Y}	2
{K, M}	3
{K, O}	3
{K, Y}	3
{M, O}	1
{M, Y}	2
{O, Y}	2

Frequent 2-itemset

Item	Count
{E, K}	4
{E, O}	3
{K, M}	3
{K, O}	3
{K, Y}	3

3-itemset

Item	Count
{E, K, O}	3
{K, M, O}	1
{K, M, Y}	2
{K, O, Y}	2

Frequent 3-itemset

Item	Count
{E, K, O}	3

آرش حریرپوش
9731505

با استفاده از الگوریتم FP-growth، frequent itemset ها برابر است با:

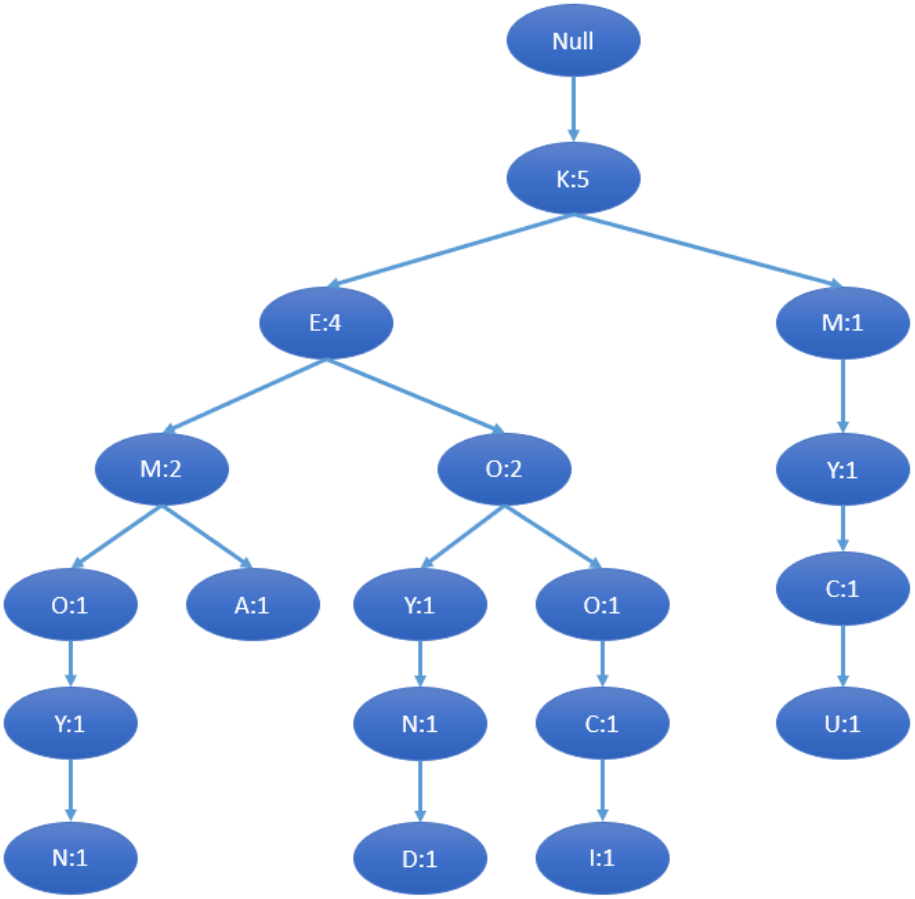
Item	Count
{A}	1
{D}	1
{I}	1
{U}	1
{C}	2
{N}	2
{M}	3
{O}	3
{Y}	3
{E}	4
{K}	5

با توجه به جدول بالا در هر itemset، هر یک از item ها را به ترتیب count مرتب می کنیم:

TID	Item_bought
T1	{K, E, M, O, Y, N}
T2	{K, E, O, Y, N, D}
T3	{K, E, M, A}
T4	{K, M, Y, C, U}
T5	{K, E, O, O, C, I}

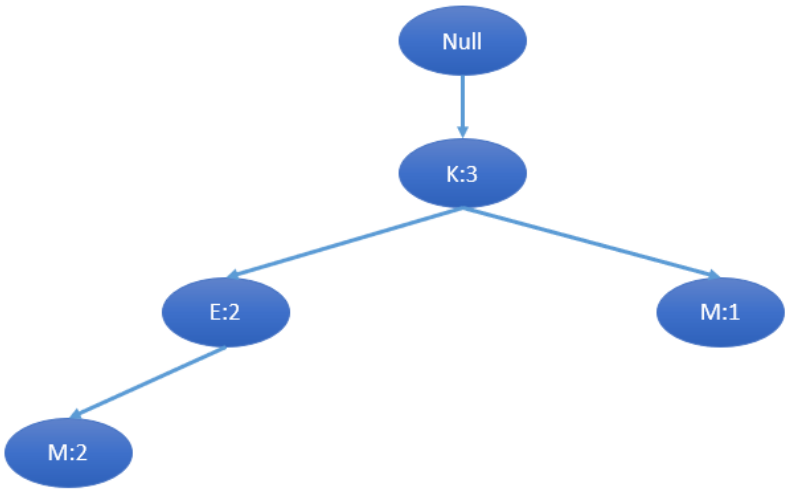
ابتدا با توجه به جدول بالا FP-tree را رسم می کنیم:

آرش حریرپوش
9731505



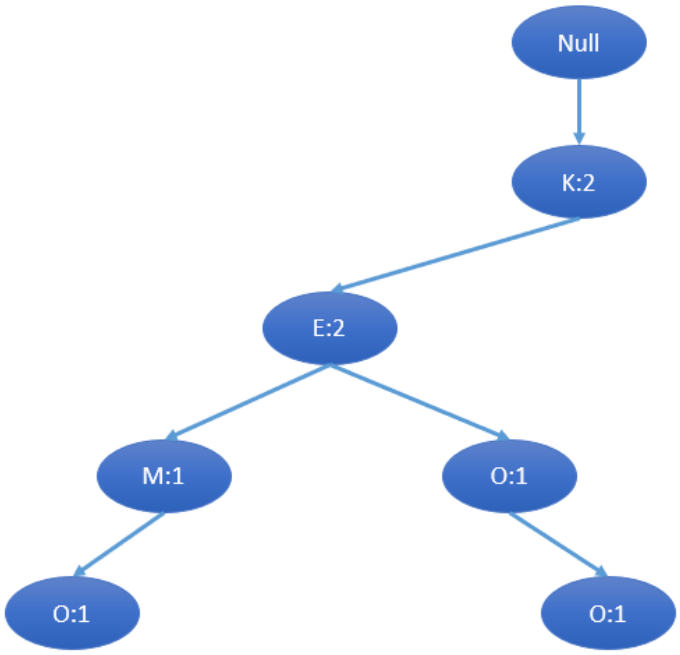
با توجه به اینکه مقدار count در item های A، D، I، U، C و N کمتر از مقدار $count_{min}$ می باشد، بنابراین این item ها جزو frequent itemset نمی باشند، بنابراین برای بقیه نود مسیر هایی که شامل آن نود ها می شود را به صورت زیر رسم می کنیم:

مسیر هایی که شامل M نود باشد:

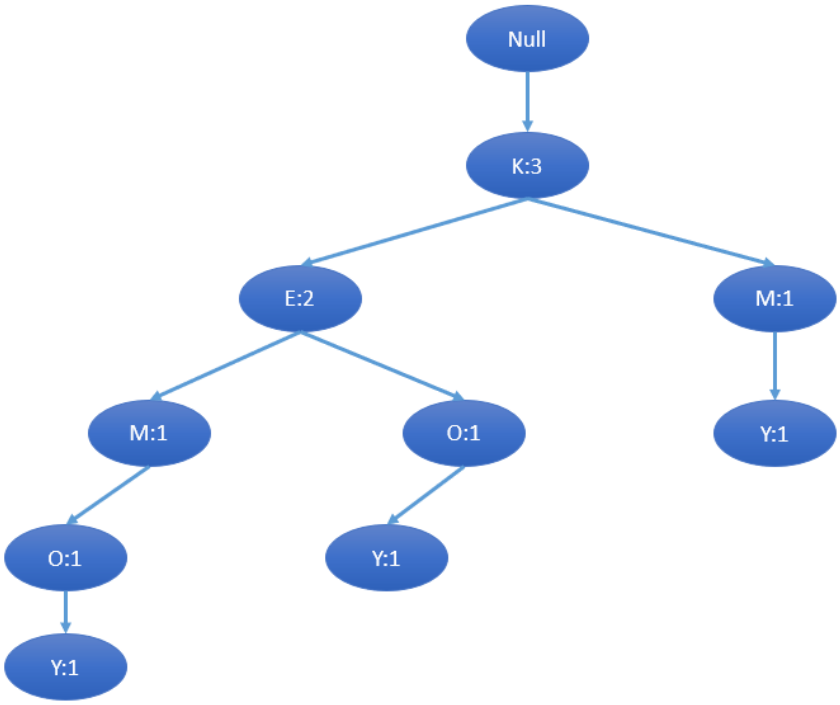


آرش حریرپوش
9731505

مسیر هایی که شامل O نود باشد:

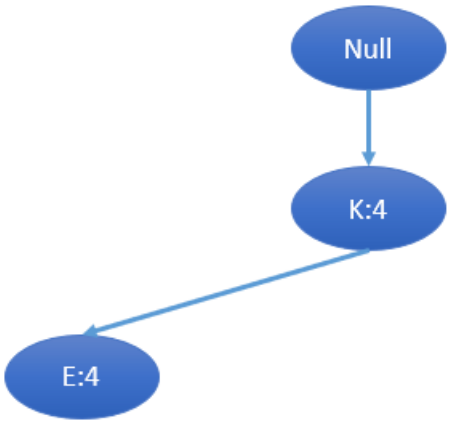


مسیر هایی که شامل Y نود باشد:

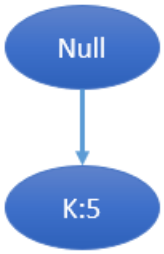


آرش حریرپوش
9731505

مسیر هایی که شامل E نود باشد:



مسیر هایی که شامل K نود باشد:



بنابراین frequent itemset برابر است با:

Suffix	Frequent Itemsets
A	{}
D	{}
I	{}
U	{}
C	{}
N	{}
M	{M, KM}
O	{O, EO, KO, KEO}
Y	{Y, KY}
E	{E, KE}
K	{K}

ب) الگوریتم های Apriori و FP-growth را از لحاظ بهینگی عملکرد مقایسه کنید.

الگوریتم FP-growth با ایجاد یک ساختار فشرده FP-tree و تولید نکردن مجموعه های کاندیدا به فضای حافظه کمتری نسبت به Apriori نیاز دارد. همچنین الگوریتم FP-growth تنها دو بار پایگاه داده را اسکن می کند در حالی که الگوریتم Apriori پایگاه داده را چندین بار برای تولید مجموعه های کاندیدا در هر مرحله اسکن می کند. و از نظر زمان نیز الگوریتم FP-growth به دلیل عدم استفاده از مجموعه های کاندیدا در هر مرحله سریع تر از الگوریتم Apriori می باشد.

(منبع: https://www.ques10.com/p/354/how-fp-tree-is-better-than-apriori-algorithm-1/)

3. در بسیاری از موارد در دنیای واقعی، برای انجام عمل دسته بندی، بخش زیادی از داده های موجود در پایگاه های داده برچسب مشخصی ندارند. از جمله روش های پیشنهادی برای حل این مشکل، روش های مبتنی بر یادگیری نیمه نظارتی، یادگیری فعالانه و یادگیری انتقالی می باشند. درباره چگونگی این روش ها، موارد استفاده هر کدام و همچنین چالش های آن ها به اختصار توضیح دهید.

یادگیری نیمه نظارتی:

این یادگیری زمانی استفاده می شود که در داده های آموزشی مقدار کمی از داده ها label دارند و مقدار بسیار زیادی از داده ها label ندارند. هدف یادگیری نیمه نظارتی این است که علاوه بر استفاده از داده های دارای label که در یادگیری نظارت شده استفاده می شود از تمام داده های موجود به صورت بهینه استفاده کند. در این نوع از یادگیری تعداد بسیار کمی از داده ها دارای label می باشند و باید سعی کنیم تا از طریق داده های زیادی که label ندارند یادگیری را انجام دهیم؛ همچنین ممکن است حتی خود label ها نیز در داده های دارای label درست نباشد.

برای استفاده موثر از داده های بدون label ممکن است از روش های بدون نظارت مثل خوشه بندی ایده گرفته شود. و زمانی که الگو ها پیدا شد، از ایده ها یا روش های یادگیری نظارت شده استفاده می شود تا داده های بدون label را label گذاری کنیم یا برای پیش بینی های بعدی label ها را به داده های بدون label اختصاص دهیم.

یادگیری فعالانه:

در این روش یادگیری مدل می تواند در طول فرآیند از اپراتور انسانی پرس و جو کند تا بتواند ابهامات را در طی فرآیند یادگیری برطرف کند. یادگیری فعال نوعی یادگیری نظارت شده است و به دنبال این است که عملکردی همانند یادگیری نظارت شده passive یا بهتر از آن را داشته باشد، اگر چه که در مورد داده های جمع آوری شده یا استفاده شده توسط مدل موثر تر است. ایده پشت یادگیری فعال این است که الگوریتم یادگیری ماشین می تواند با داده های آموزشی بدون label کمتری به دقت های بالاتری دست یابد، در

صورتی که بتواند انتخاب کند از کدام داده ها یادگیری را انجام دهد. یک مدل یادگیری فعال ممکن است پرس و جو هایی را مطرح کند، که معمولا به صورت داده های بدون label می باشد تا فرد متخصص آن ها را label گذاری کند.

یادگیری فعال و یادگیری نیمه نظارتی بر روی یک مشکل اما در جهت های متفاوتی کار می کنند. در حالی که مدل های نیمه نظارتی از آنچه learner در مورد داده های بدون label می داند استفاده می کند؛ مدل های فعال سعی می کند تا جنبه های ناشناخته را کشف کند.

از روش یادگیری فعال زمانی استفاده می شود که داده های زیادی در دسترس نمی باشد و جمع آوری یا label گذاری داده های جدید پر هزینه می باشد. فرآیند یادگیری فعال اجازه می دهد تا نمونه برداری از دامنه به طور مستقیم باعث کاهش تعداد نمونه ها و بیشینه شدن کارآیی مدل شود.

از روش یادگیری فعال معمولا در برنامه هایی استفاده می شود که بدست آوردن label داده ها پر هزینه باشد، برای مثال محاسبات زیستی برنامه ها.

یادگیری انتقالی:

یادگیری انتقالی نوعی از یادگیری می باشد که در آن یک مدل ابتدا برای یک کار آموزش داده می شود، سپس برخی یا کل مدل به عنوان نقطه شروع برای یک کار مرتبط با کار مدل عنوان شده استفاده می شود. این روش در مشکلاتی مفید است که یک کار در ارتباط با کار اصلی مورد نظر وجود داشته باشد و برای آن کار مرتبط داده زیادی وجود داشته باشد.

یادگیری انتقالی برای مدل هایی مناسب است که به صورت افزایشی آموزش می یابند و مدل فعلی می تواند به عنوان نقطه شروع برای ادامه آموزش استفاده کرد. مثل شبکه های یادگیری عمیق.

(منبع: https://machinelearningmastery.com/types-of-learning-in-machine-learning/)

4. دو دسته اصلی روش های یادگیری گروهی را نام برده و تفاوت آن ها را توضیح دهید. مشخص نمایید هر یک از روش های پیشنهادی در بهبود چه مشکلی حین آموزش مدل یادگیری موثر هستند؟

دو دسته اصلی روش های یادگیری گروهی برابر Bagging و Boosting می باشد، که تفاوت های آن ها عبارتند از:

Boosting	Bagging
راهی برای ترکیب پیشبینی هایی می باشد که مربوط به type های متفاوت می باشند.	ساده ترین راه برای ترکیب پیشبینی هایی می باشد که مربوط به یک type می باشند.
هدف آن کاهش bias است نه variance.	هدف آن کاهش variance است نه bias.
در این روش وزن هر مدل بر اساس عملکردش مشخص می شود.	در این روش هر مدل وزن یکسانی را دریافت می کند.
مدل های جدید تحت تاثیر عملکرد مدل های قبلی ساخته می شوند.	هر مدل مستقل از بقیه مدل ها ساخته می شود.
هر زیر مجموعه جدید شامل عناصری است که توسط مدل های قبلی به درستی دسته بندی نشده اند.	زیر مجموعه داده های متفاوت به صورت تصادفی با replacement از کل مجموعه داده های آموزشی ایجاد می شود.
سعی می کند تا bias را کاهش دهد.	سعی می کند تا مشکل over fitting را برطرف کند.
در صورتی که classifier پایدار و ساده (bias مدل زیاد باشد) باشد از این روش استفاده می کنیم.	در صورتی که classifier ناپایدار (variance مدل زیاد باشد) باشد از این روش استفاده می کنیم.

بنابراین Bagging در بهبود مشکل over fitting حین آموزش مدل و Boosting در بهبود مشکل under fitting حین آموزش مدل موثر است.

5. در ارتباط با ماشین های بردار پشتیبان به سوالات زیر پاسخ دهید.

الف) با توجه به SVM دو کلاسه شرح داده شده در کلاس، یک سناریو برای SVM چند کلاسه (M کلاس) با ذکر توضیحات ارائه دهید.

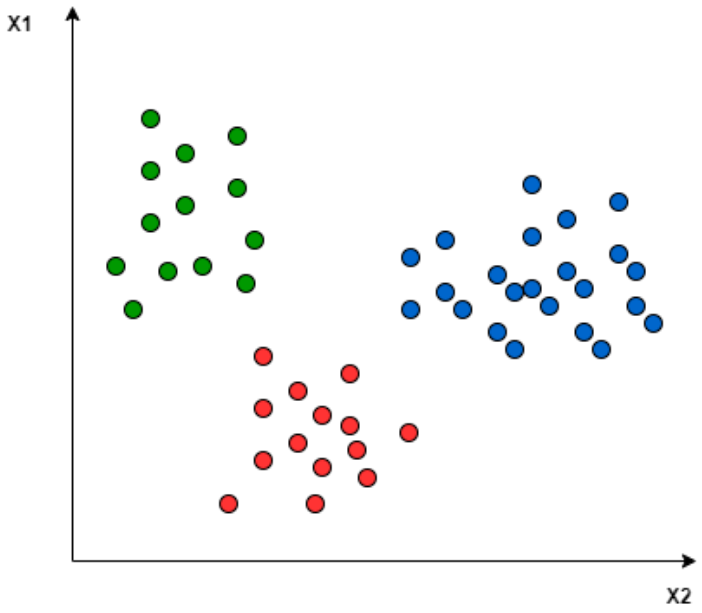
برای انجام svm چند کلاسه نیاز است تا داده ها را به فضای با ابعاد بیشتری منتقل کنیم تا بتوانیم هر دو کلاس را به صورت خطی از یک دیگر جدا کنیم؛ پس از انجام این کار به دو روش می توانیم m کلاس را دسته بندی کنیم:

- **روش One-to-One**: در این روش مسئله دسته بندی چند کلاسه به چندین مسئله دسته بندی binary تبدیل می شود و برای هر جفت از کلاس ها یک دسته بندی binary انجام می شود؛ در

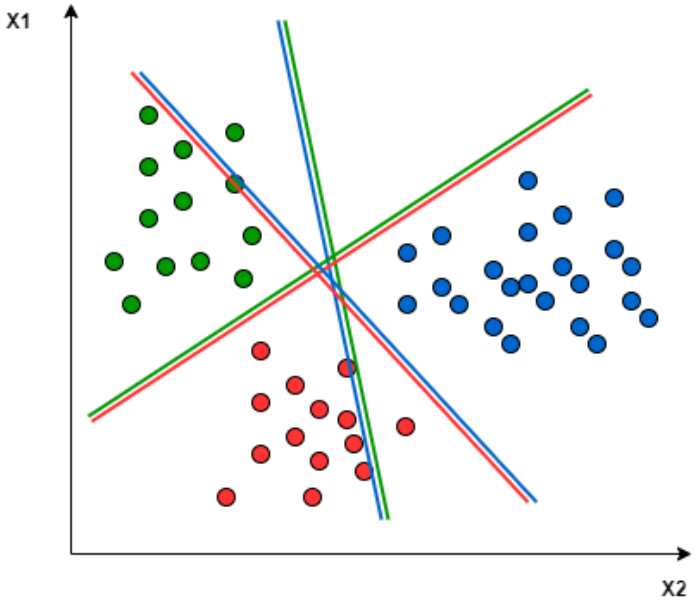
این حالت برای دسته بندی داده ها از $\frac{m(m-1)}{2}$ تا SVM استفاده می شود که هر یک از آن ها یک خط بین دو تا از کلاس های در نظر گرفته شده رسم می کند.

- روش **One-to-Rest**: در این روش برای هر کلاس یک دسته بندی binary انجام می شود. در این حالت داده ها به کمک m تا SVM دسته بندی می شوند، که هر یک از آن ها یک کلاس را دسته بندی می کند. (در این روش بین هر کلاس از داده ها و بقیه کلاس ها به کمک SVM یک hyperplane رسم می شود)

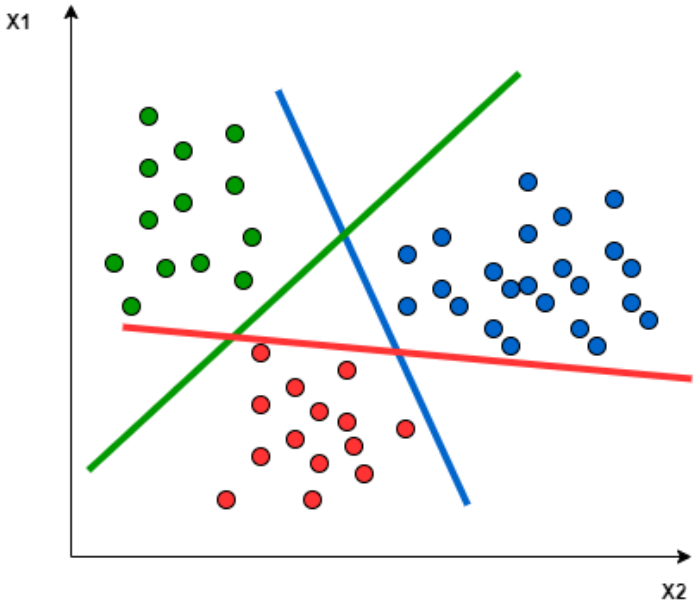
برای مثال داده های زیر را در نظر بگیرید که مطابق شکل زیر به سه دسته سبز، آبی، قرمز نمایش داده شده اند:



در صورتی که این داده ها را به روش One-to-One دسته بندی کنیم، به یک hyperplane نیاز داریم تا هر دو کلاس را بدون در نظر گرفتن کلاس سوم از یک دیگر جدا کند. این بدین معنی است که این جداسازی تنها داده های دو کلاس در نظر گرفته شده در یک تقسیم را بررسی می کند. برای مثال در شکل زیر، خط قرمز-آبی سعی می کند تا تنها margin بین داده های آبی و قرمز را بیشینه کند. و با داده های سبز کاری ندارد. شکل زیر جداسازی انجام شده به کمک این روش را نشان می دهد:



در روش One-to-Rest، نیاز داریم تا به کمک یک hyperplane یک کلاس را از بقیه کلاس ها جدا کنیم. این بدین معنی است که در این جداسازی تمام داده ها را در نظر می گیریم، بدین صورت که تمام داده ها را به دو دسته تقسیم می کنیم: یک دسته شامل داده هایی می باشد که در کلاسی است که می خواهیم آن را از بقیه جدا کنیم و دسته دیگر نیز بقیه داده ها را شامل می شود. برای مثال در شکل زیر، خط سبز داده های کلاس سبز را از بقیه داده ها جدا می کند. شکل زیر جداسازی انجام شده به کمک این روش را نشان می دهد:



(منبع: https://www.baeldung.com/cs/svm-multiclass-classification)

ب) منظور از مدل با حاشیه سخت چیست؟

در صورتی که شرط دسته بندی صحیح هر یک از داده ها را به صورت زیر در نظر بگیریم:

اگر $Y_i(W * X_i + b) \geq 1$ دسته بندی داده صحیح است و در غیر این صورت دسته بندی داده نادرست است.

در این حالت اگر داده ها به صورت خطی مستقل از یک دیگر باشند تنها به کمک یک hyperplane می توانیم آن ها را از یک دیگر جدا کنیم، اما در صورتی که داده ها دارای outlier باشند دیگر نمی توانیم آن ها را به صورت خطی از یک دیگر تفکیک کنیم. بنابراین به این مدل از SVM ها مدل با حاشیه سخت گفته می شود، به این دلیل که محدودیت های سخت گیرانه ای برای دسته بندی صحیح داده ها در نظر گرفته ایم. بنابراین مدل با حاشیه سخت در SVM به مدل هایی گفته می شود که به کمک یک hyperplane دو کلاس را به طور کامل از یک دیگر جدا می کند، یعنی هیچ خطایی را برای تعیین مرز بین نقاط در نظر نمی گیرد.

(زمانی که در فرمول $f(w) = \frac{||w||^2}{2} + C(\sum_{i=1}^{n} \xi_i)^k$ مقدار C را برابر بینهایت قرار دهیم تا مقدار margin به شدت محدود شود)

(منبع: https://towardsdatascience.com/support-vector-machines-svm-c9ef22815589 و https://www.robots.ox.ac.uk/~az/lectures/ml/lect2.pdf)

ج) فرض کنید یک ماشین بردار پشتیبان با مرزبندی خطی آموزش داده اید و متوجه می شوید دچار کم برازش شده است. برای حل این مشکل پارامترهای مدل خود را چگونه تغییر می دهید؟

برای برطرف کردن مشکل underfitting در این حالت در صورتی که از فرمول $f(w) = \frac{||w||^2}{2} + C(\sum_{i=1}^{n} \xi_i)^k$ استفاده کرده باشیم مقدار C و K را افزایش می دهیم تا در هنگام آموزش مدل میزان خطای کمتری را قبول کند و مدل آموزش داده شده را به سمت تبدیل شدن به یک مدل سخت پیش می بریم، همچنین می توانیم مقدار x_i ها را به یک تابع $\Phi(x)$ می دهیم، و برای برطرف کردن این مشکل نیاز است تا مدل را پیچیده تر کنیم، که این کار را با استفاده از توابع kernel انجام می دهیم و یک مدل غیر خطی برای داده ها آموزش می دهیم.

6. جدول زیر نوع کتاب های خریده شده یک فروشگاه را نشان می دهد:

نوع کتاب خریداری شده	شماره خرید
مذهبی، رمان، شعر	۱
مذهبی، رمان، تاریخی	۲
روانشناسی، رمان، تاریخی، شعر	۳
روانشناسی، تاریخی، شعر	۴
روانشناسی، مذهبی، رمان	۵
روانشناسی، مذهبی، رمان، تاریخی	۶
روانشناسی	۷
روانشناسی، مذهبی، رمان	۸
روانشناسی، مذهبی، تاریخی	۹
روانشناسی، مذهبی	۱۰

الف) با استفاده از روش FP-Growth تمامی frequent itemset هایی که به کتاب نوع رمان ختم می شود را بیابید. (support = 20%)

با توجه به support داریم:

$$support = 20\% \Rightarrow \frac{\sigma(X \cup Y)}{N} = \frac{\sigma(X \cup Y)}{10} = 20\% \Rightarrow \text{count}_{min} = 2$$

Item	Count
{روانشناسی}	8
{مذهبی}	7
{رمان}	6
{تاریخی}	5
{شعر}	3

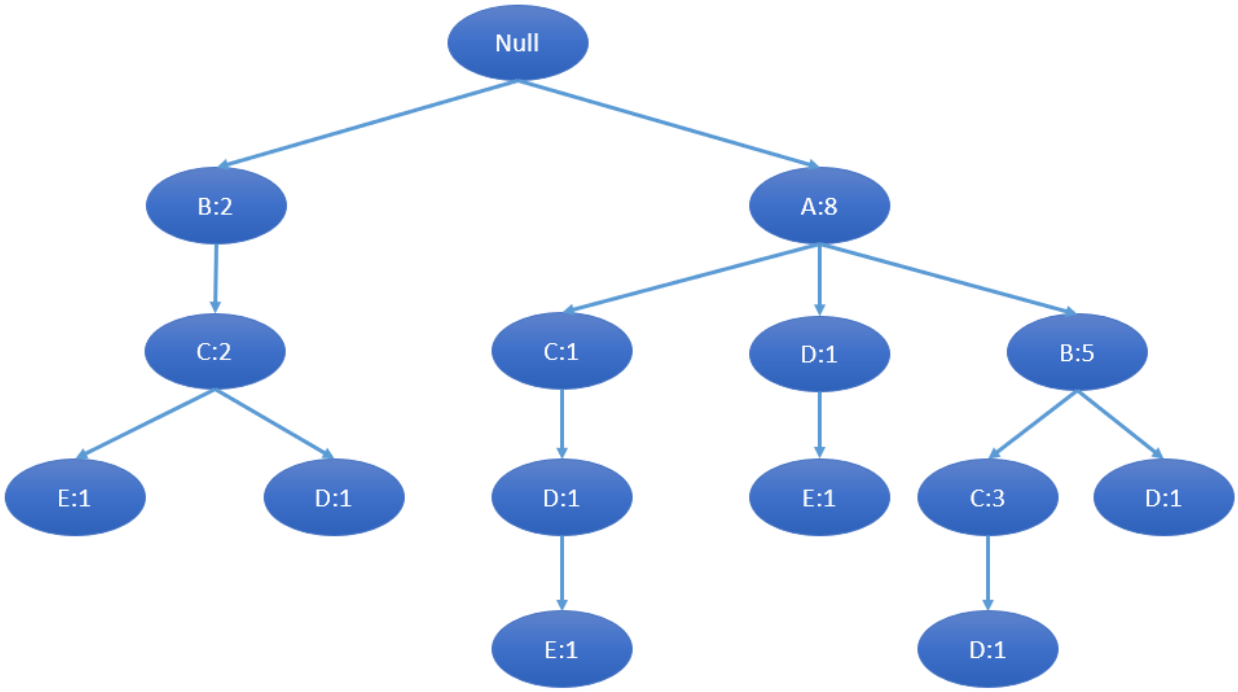
با توجه به جدول بالا به هر یک از عنوان کتاب ها یک حرف نسبت می دهیم:

A = روانشناسی، B = مذهبی، C = رمان، D = تاریخی، E= شعر

با توجه به جدول بالا در هر itemset، در جدول داده شده هر یک از item ها به ترتیب count فرار دارند بنابراین نیازی نیست تا آن ها را بر این اساس مرتب کنیم تنها بر اساس حروف نسبت داده شده بر هر یک از عناوین جدول داده شده را به صورت زیر باز سازی می کنیم:

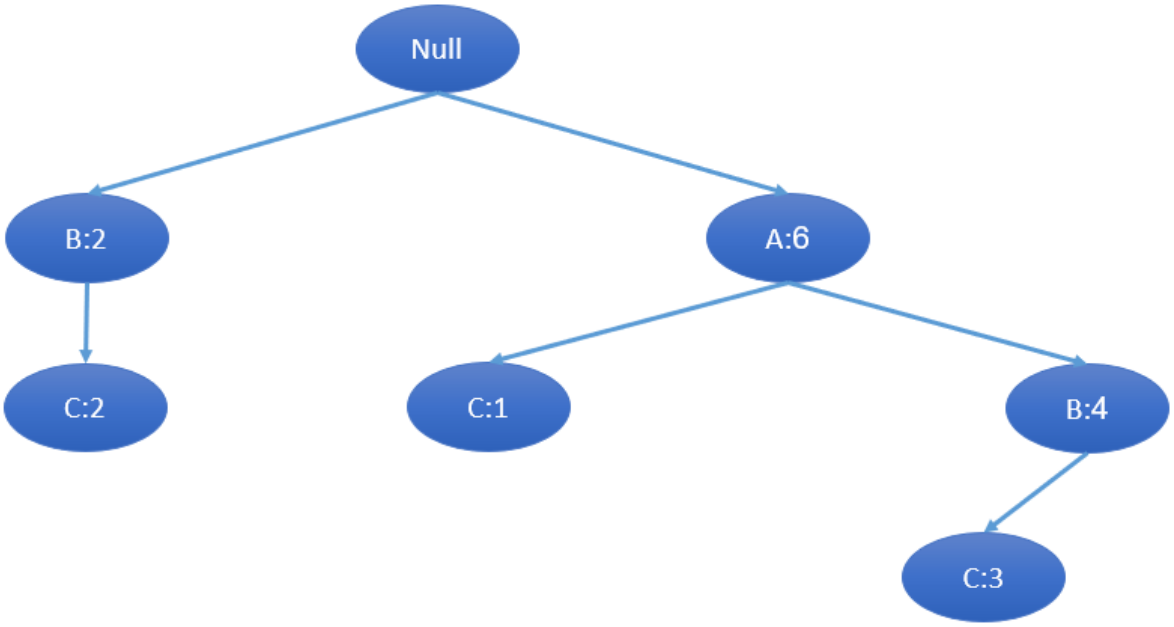
شماره خرید	نوع کتاب خریداری شده
1	{B, C, E}
2	{B, C, D}
3	{A, C, D, E}
4	{A, D, E}
5	{A, B, C}
6	{A, B, C, D}
7	{A}
8	{A, B, C}
9	{A, B, D}
10	{A, B}

سپس با توجه به جدول بالا FP-tree را رسم می کنیم:



سپس برای بدست آوردن تمامی frequent itemset هایی که به کتاب نوع رمان ختم می شود، مسیر هایی که شامل نود مرتبط با عنوان رمان یعنی C را به صورت زیر رسم می کنیم:

آرش حریرپوش
9731505



بنابراین تمامی frequent itemset هایی که به کتاب نوع رمان ختم می شود برابر است با:

$$frequent\ itemsets(C) = \{\{C\},\{B,C\},\{A,C\}\}$$

ب) با در نظر گرفتن confidence = %50 قواعد معتبر قابل استخراج از frequent itemset های به دست آمده از قسمت الف را بیابید.

برای بدست آوردن قواعد معتبری که می توان به کمک frequent itemset هایی که در مرحله قبل بدست آمده است، مقدار confidence را برای تمام قواعدی که می توان به کمک frequent itemset های بدست آمده ایجاد کرد را به صورت زیر محاسبه می کنیم:

Association Rule	Count	Confidence
$B \rightarrow C$	5	$\frac{5}{7} \simeq 0.71$
$C \rightarrow B$	5	$\frac{5}{6} \simeq 0.83$
$A \rightarrow C$	4	$\frac{4}{8} = 0.5$
$C \rightarrow A$	4	$\frac{4}{6} \simeq 0.66$

بنابراین با توجه به اینکه مقدار Confidence در همه قواعد ایجاد شده کمتر از 50% نمی باشد، بنابراین تمامی قواعد جدول بالا که از طریق frequent itemset های مرحله قبل بدست آمده اند قابل قبول می باشند.

آرش حریرپوش
9731505

❖ بخش دوم – پیاده سازی

.8

1) برای بدست آوردن پارامتر های مناسب برای آموزش مدل مورد نظر از طریق ترکیب تابع تقسیم های gini و entropy و همچنین عمق درخت ها از GridSearchCV استفاده می کنیم، که در اینجا برای درخت تصمیم و جنگل تصادفی تابع تقسیم می تواند gini یا entropy و بیشترین مقدار عمق درخت می تواند 5 یا 10 باشد، مناسب ترین دقت بدست آمده در درخت تصمیم و جنگل تصادفی برای داده های آموزش و تست به صورت زیر می باشد (مقدار تابع تقسیم با توجه به اینکه توسط GridSearchCV مشخص می شود در اجرا های متفاوت می تواند تغییر کند اما میزان دقت محاسبه شده برای داده های آموزش و تست در این دو مدل تغییر چندانی نمی کند) :

Model Name	Train Accuracy	Test Accuracy	criterion	Max depth
Decision Tree	84.6%	83.7%	entropy	5
Random Forest	93.2%	83.2%	entropy	10

همانطور که مشاهده می شود با تغییر مدل به جنگل تصادفی دقت مدل برای داده های آموزشی افزایش پیدا می کند و مقدار دقت برای داده های تست اندکی کاهش پیدا می کند، می توان مشاهده کرد که در مدل جنگل تصادفی با توجه به فاصله بین دقت مدل در داده های آموزش و تست overfitting رخ داده است، این در حالی است که مدل آموزش داده شده از طریق درخت تصمیم این مشکل را ندارد. (یکی از عواملی که باعث ایجاد این تغییر بین این دو مدل شده است متفاوت بودن مقدار عمق بین در این دو مدل می باشد، به صورت کلی نتایج بدست آمده از طریق این دو مدل خیلی با یک دیگر تفاوتی ندارد، تنها ممکن است حدود چند درصد دقت برای مدل جنگل تصادفی بهتر باشد اما به طور کلی فرق چندانی نتایج بدست آمده با یک دیگر ندارد)

2) مدت زمان آموزش و تست دو مدل عنوان شده به صورت زیر می باشد:

Model Name	Training Time (ms)	Test Accuracy Time (ms)	Test Accuracy Time (ms)
Decision Tree	7.98	9.97	8.98
Random Forest	275	67.8	46.9

همانطور که در جدول بالا مشاهده می شود مدل جنگل تصادفی نسبت به مدل درخت تصمیم به زمان بسیار زیاد تری برای آموزش مدل و محاسبه دقت برای داده های آموزشی و تست نیاز دارد.
(کد های این بخش در فایل Titanic_RF.ipynb قرار دارد)

.9

1) پیش پردازش داده ها در این سوال همانند سوال های قبلی می باشد.

2) دقت بدست آمده به کمک مدل SVM با هسته خطی برابر است با:

Model Name	Train Accuracy	Test Accuracy	Kernel
SVM	81.1%	80.4%	Linear

3) دقت این مدل تفاوت چندانی با دو مدل قبلی ندارد تنها حدود چند درصد دقت این مدل بر روی داده های آموزشی و تست از دو مدل قبلی کمتر است که آن هم می تواند به این دلیل باشد که داده ها به صورت خطی جدا پذیر نیستند و مدل SVM برای اینکه بتواند خطی پیدا کند که بیشترین margin را داشته باشد مقداری خطا را قبول می کند تا در عوض بتواند مدلی تعمیم یافته تر آموزش دهد. (مقدار های متفاوت C می تواند باعث شود که دقت مدل بر روی داده های آموزش و تست به شدت کاهش یا افزایش پیدا کند، در صورتی که random_state مقدار دهی نمی شد دقت مدل به ازای اجرای های مختلف بسیار متفاوت می شد اما با مقدار دهی این متغیر هنگام آموزش این مدل نتیجه مطلوب تری بدست می آید و دقت مدل با ازای اجرا های مختلف تغییر چندانی نمی یابد)

4) دقت بدست آمده به کمک مدل SVM با هسته غیر خطی برابر است با:

Model Name	Train Accuracy	Test Accuracy	Kernel
SVM	82.3%	79.8%	rbf

5) با تغییر کرنل SVM به یک کرنل غیر خطی دقت مدل بر روی داده های آموزش اندکی افزایش و بر روی داده های تست اندکی کاهش یافته است، که دلیل آن می تواند این باشد که در مدل غیر خطی می توان داده ها را بهتر از یک دیگر تفکیک کرد چرا که آن ها را به یک فضای تفکیک پذیر انتقال می دهیم به همین دلیل می توانیم خط بهتری را نسبت به حالت خطی برای مدل در نظر بگیریم اما این کار باعث شده تا تغییرات دقت مدل بر روی داده های آموزش و تست نیز اندکی افزایش یابد.
(کد های این بخش در فایل Titanic_SVM.ipynb قرار دارد)