



**Tehran Polytechnic University**  
**Computer Engineering Department**

---

**Data Mining**  
**Assignment Two**

---

**Name: Mohammad Hossein Badiei**

**Student ID: 9531701**

**Majors: Artificial Intelligence and Robotics (Amirkabir) | Electrical  
Engineering (Tehran)**

**Instructor: Dr. Ehsan Nazerafard**

**Spring 2021**

## پاسخ سوال 1

در این سوال خواسته شده که هر یک از مفاهیم موجود در سوال را تعریف کنیم.

### 1. Unsupervised Learning

یادگیری بدون نظارت یکی از روش‌های یادگیری است که بر روی داده‌های بدون برچسب اعمال می‌شود. در واقع در این نوع یادگیری آموزش بر روی دیتاهای بدون برچسب اعمال شده و سعی در استخراج الگو و ساخت مدل می‌نماید. در واقع در این نوع یادگیری، هدف دسته‌بندی داده‌هاست و کشف ساختاری خاص در داده‌ها. به همین دلیل که کاربر نظارتی بر مدل ندارد و مدل خود به تنهایی به کشف الگو بر روی داده‌های بدون برچسب می‌پردازد، به آن یادگیری بدون نظارت می‌گویند.

### 2. Supervised Learning

این روش یادگیری در واقع مربوط به یادگیری ماشینی است که یک سری داده‌های برچسب‌گذاری شده‌ی ورودی-خروجی را که به آنها نمونه‌های آموزشی می‌گویند را دریافت کرده و با استفاده از آنها تابعی را که وظیفه‌ی آن نگاشت ورودی به خروجی است، فرا می‌گیرد. به این دلیل به این نوع از یادگیری، نظارت شده می‌گویند که یک ناظر باید باشد که اطلاعات را در اختیار یادگیرنده قرار دهد و هدف از این نوع یادگیری، اکتشاف ارتباط ورودی و خروجی است.

### 3. Semi-Supervised Learning

این روش یادگیری در واقع هم از داده‌های برچسب‌گذاری شده و هم از داده‌های بدون برچسب به جهت بالا بردن دقت در یادگیری استفاده می‌کند. در این روش تعداد از داده‌ها برچسب دارند و تعدادی بدون برچسب هستند و سعی می‌کنیم همه‌ی داده‌ها را به صورت برچسب‌گذاری شده دسته‌بندی کرده و خود را به supervised learning نزدیک‌تر نماییم.

### 4. Outlier

این نوع داده در واقع یک (یا یک سری) داده‌ی آماری است که از سایر داده‌های آماری متعلق به یک نمونه‌ی تصادفی، فاصله‌ی غیر عادی داشته باشد. در واقع این داده نشان‌دهنده‌ی غیرعادی بودن یکی از مشاهدات یا به عبارتی یکی از sample هاست (دقت شود که این داده خطا نیست و با نویز فرق دارد، این داده یک داده‌ی غیر عادی در مجموعه است).

### 5. Dimension

دیمانسیون یا بعد در داده‌کاوی (به عبارت بهتر در فیلد آمار) به معنای مجموعه اطلاعاتی است که از یک رویداد قابل اندازه‌گیری، ثبت و البته ذخیره می‌شود. در واقع با استفاده از دیمانسیون مدلی را طرح می‌کنند که هر دیمانسیون یک مجموعه اطلاعات را در اختیار قرار داده است و به جهت سریع‌تر شدن بازیابی اطلاعات از یک رویداد کاربرد دارد.

### 6. Training, Validating and Testing Data

در این عبارت دو نوع داده عنوان شده است. داده‌های آموزش یا training data به جهت ساخت مدل مورد استفاده قرار می‌گیرند و در واقع فرایند یادگیری برای ساخت مدل با اطلاعات موجود در این داده‌ها انجام می‌پذیرد. اما داده‌های ارزیابی و تست یا validating and testing data به جهت بررسی کیفیت ساختار و کارایی مدلی است که ساخته شده و در واقع داده‌های تست به جهت بررسی performance مدل و ارزیابی و اعتبارسنجی از مدل مورد استفاده قرار می‌گیرند.

## 7. Data Warehousing

Data warehousing در واقع یک فرایند جمع‌آوری و مدیریت از داده‌ها است که از منابع مختلف تهیه می‌شوند. این فرایند از این جهت کاربردی است که مجموعه‌ای از داده‌ها از منابع ناهمگون در کنار یکدیگر قرار گرفته و می‌تواند برای تحلیل داده‌ها بکار گرفته شود. در واقع DW یک انبار از داده‌ها از منابع گوناگون است که برای تحلیل و گزارش‌دهی از داده‌ها کاربرد دارد.

## 8. Missing Values

مقادیر از دست رفته یا داده‌های از دست رفته در واقع زمانی مطرح می‌شوند که هیچ مقدار داده برای متغیر در یک مشاهده ثبت نشده باشد. این missing value می‌تواند بدلیل اشکال سیستماتیک در جمع‌آوری درست داده‌ها صورت گیرد.

## 9. Independent Variable

متغیرهایی که در یک مشاهده بکار گرفته می‌شوند بدین منظور که اثر سایر متغیرها را ارزیابی کنند، متغیر مستقل گویند و به متغیرهایی که به ارزیابی آنها می‌پردازیم متغیرهای وابسته می‌گویند. در واقع در مباحث آماری داده‌کاوی، یک سری متغیرها را بکار می‌گیرند و به آنها مقادیر مختلف را اعمال می‌کنند تا اثر بقیه‌ی متغیرها را به آنها متغیر وابسته می‌گویند ارزیابی کنند.

## پاسخ سوال 2

Dimensionality Reduction در واقع با حذف ویژگی‌هایی که اهمیت بالایی ندارند گویند.

تکنیک‌های Dimensionality Reduction :

**Principal Components Analysis (PCA) ✓**

**Forward Feature Construction ✓**

**Backward Feature Elimination ✓**

**Random Forests/Ensemble Trees ✓**

**Missing Values Ratio ✓**

**Low Variance Filter ✓**

**High Correlation Filter ✓**

سوال خواسته که یکی از متدهای فوق را توضیح دهیم که ما دو تا را تصمیم داریم به طور مختصر بیان کنیم.

**Low Variance Filter ✓**

در این متد داده‌هایی که واریانس کمی دارند از مجموعه حذف شده و از آن طریق کاهش ابعاد صورت می‌گیرد. در واقع همانطور که می‌دانیم واریانس معیاری برای نشان دادن پراکندگی داده‌هاست و وقتی واریانس من باشد نتیجه می‌گیریم که داده‌ها در این ویژگی بخصوص مشابه یکدیگر هستند و نتیجتاً این ستون در دیتاست یا هر کالکشن دیگر در پایگاه داده، اطلاعات چندانی را در خود ندارد

و لذا می‌توان این ستون ویژگی را حذف کرد. به عنوان مثال داده‌های تصویری را در نظر می‌گیریم که در آنها zero padding به تعداد مشخصی به کار برده شده است. ( zero padding معمولا در شبکه‌های عصبی عمیق برای اضافه کردن سطر و ستون‌های صفر به ماتریس تصویر برای مرزبندی انجام می‌شود.) حال فرض کنیم ستون‌های داده‌ی ما میانگین ستونی از رنگ‌های یک عکس است (البته ماتریس عکس رنگی سه بعدی است و اینجا باید یک تصویر سیاه و سفید را در نظر بگیریم) حال مثلا در این ستون در ردیف اول میانگین مقادیر صفر و یک از ستون اول تصویر اول است ولی چون در این تصاویر zero padding بکاربرده شده است لذا مقادیر تمامی سطرهای این ستون صفر بوده و مطمئنا چون برای همه‌ی تصاویر این مقدار برابر و مساوی با صفر است و واریانس را صفر خواهد کرد و نتیجتا این اطلاعات نیز برای تحلیل داده اغلب با اهمیت نخواهند بود و می‌توان این ستون را به کلی حذف نمود.

### Missing Values Ratio ✓

یا به عنوان یک متد دیگر می‌توان به این متد اشاره کرد که ستون داده‌ای که در آن missing value به وفور رخ داده است مسلما دارای اطلاعات مفیدی نخواهد بود چون در خیلی از مشاهدات به هر دلیل داده‌ای دریافت نشده و ذخیره‌سازی انجام نشده است و لذا بدون داده هم که دیگر تحلیلی وجود ندارد. نتیجتا می‌توان با استفاده از این متد، این گونه ستون از داده‌ها را هم حذف نمود.

### Feature Selection VS Feature Extraction

در واقع این دو مفهوم به کلی با یکدیگر تفاوت دارند. در استخراج ویژگی، یک سری ویژگی‌ها از داده تهیه شده و منجر به ایجاد یک محصول جدید که همان فیچر آن مجموعه از داده‌ها است، می‌شود که نمونه‌ی بارز آن عمل استخراج داده در شبکه‌های CNN است. اما در انتخاب داده یا feature selection، داده‌های اضافی (داده‌هایی با ویژگی‌های کم اهمیت) از مجموعه‌ی داده‌ها، فیلتر شده و داده‌های با اهمیت بیشتر حفظ می‌شوند که می‌بینیم کاملا با استخراج ویژگی تفاوت زیادی دارد.

## پاسخ سوال 3

این سوال از ما خواسته معیارهای ارزیابی دقت، فراخوانی و امتیاز اف را بر اساس ماتریس درهم ریختگی معرفی کنیم.

ماتریس درهم ریختگی به صورت زیر است:

	Predicted Positives	Predicted Negatives
Positives	True Positives	False Negatives
Negatives	False Positives	True Negatives

ابتدا به تعریف هر یک از عناصر ماتریس در هم ریختگی می پردازیم:

- ✓ True Positives : داده هایی که به مجموعه تعلق داشته و تشخیص مدل در قبال این داده ها صحیح بوده است.
- ✓ False Positives : داده هایی که به مجموعه تعلق نداشته ولی تشخیص مدل در قبال این داده ها اشتباه بوده است و آنها را خارج از مجموعه تشخیص داده.
- ✓ True Negatives : داده هایی که به مجموعه تعلق نداشته و تشخیص مدل در قبال این داده ها صحیح بوده و آن را خارج از مجموعه تشخیص داده است.
- ✓ False Negatives : داده هایی که به مجموعه تعلق نداشته ولی تشخیص مدل در قبال این داده ها اشتباه بوده است و آنها را متعلق به مجموعه تشخیص داده است.

حال سوال خواسته که معیارهای گفته شده را معرفی کنیم:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

$$\text{Fscore} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

همانطور که مشاهده می کنید، recall نسبتی است بین تعداد اسناد مرتبطی که درست تشخیص داده شده اند بر تعداد کل اسناد مرتبط ولی precision در واقع نشان می دهد چه تعداد اسنادی که درست تشخیص داده شده اند مرتبط هستند و این نسبت را برآورد می کند. همچنین امتیاز اف نیز با دخیل کردن recall و دقت، معیاری برای ارزیابی صحت را فراهم می کند.

## پاسخ سوال 4

این که همبستگی دو متغیر صفر باشد به این معناست که هیچ ارتباط خطی بین دو متغیر وجود ندارد. در واقع همبستگی ارتباط خطی بین دو متغیر را اندازه گیری می کند.

اما باید توجه رد که این مفهوم با مفهوم استقلال فرق می کند. در واقع دو متغیر که مستقل هستند هیچ شکلی از ارتباط بین آنها برقرار نیست. در واقع فرض کنید که یک متغیر با توان دوم دیگری به ازای نقاطی در ارتباط باشد، آنگاه همبستگی بین این دو متغیر صفر خواهد شد چون هیچ رابطه ی خطی بین این دو وجود ندارد ولی می دانیم که به واسطه ی رابطه ی غیر خطی بین آنها، این دو متغیر نمی توانند مستقل باشند. لذا نمی توانیم بگوییم که اگر دو متغیر همبستگی صفر دارند، لذا مستقل هستند. طبق توضیحاتی که دادیم، این جمله نادرست است.

## پاسخ سوال 5

data cleaning که البته به آن data cleansing یا پاکسازی داده‌ها نیز می‌گویند، در واقع فرایند تشخیص و تصحیح یا حذف داده‌های خراب شده و ناصحیح را از جدول داده‌ها بر عهده دارد و در واقع به فرایند شناسایی داده‌های ناقص، نادرست، ناصحیح یا قسمت‌های نادرست داده و سپس تصحیح یا حذف آنها اشاره می‌کند.

تجمیع داده یا data integration در واقع فرایند ترکیب داده از منابع مختلف و اراده‌ی آن به صورت یک نمای واحد به کاربر است. در واقع می‌توان گفت که برای داده‌های استخراج شده از منابع مختلف یک دیدگاه واحد وجود ندارد، لذا این داده‌ها پس از استخراج با الگوریتم‌های ترکیب داده که می‌توان از جمله‌ی آنها الگوریتم‌های owa و بیزین و کالمن فیلتر را معرفی کرد، داده‌ها را در کنار یکدیگر ترکیب کرده و یک دیدگاه که برآیند تجمیع داده هاست، به کاربر ارائه دهیم. به این تکنیک data integration می‌گویند.

در data transaction در واقع داده را از یک ساختار به یک ساختار دیگر تبدیل می‌کنیم. لذا یک تعریف کلی برای این تکنیک بدین صورت است که data transaction فرایندی است برای تغییر فرمت، ساختار یا مقدار داده به هر نحوی که می‌تواند یک نداشت ساده باشد تا هر تبدیل دیگری. این تکنیک در مباحث تلفیق داده از اهمیت بالایی برخوردار است.

## پاسخ سوال 6

ID	Items
T1	نان، الویه، پنیر
T2	نان، الویه
T3	نان، کره، مربا
T4	مربا، کره
T5	مربا، پنیر
T6	نان، کره، مربا

مرحله اول

Item	Count
نان	4
الویه	2
پنیر	2
کره	3
مربا	4

همگی موارد فوق دارای support بیشتر از minsupport دارند. لذا هیچ یک در این مرحله حذف نمی‌شوند.

مرحله دوم

Item	Count
(نان و الویه)	2
<del>(نان و پنیر)</del>	<del>1</del>
(نان و کره)	2
(نان و مربا)	2
<del>(الویه و پنیر)</del>	<del>1</del>
<del>(الویه و کره)</del>	<del>0</del>
<del>(الویه و مربا)</del>	<del>0</del>
<del>(پنیر و کره)</del>	<del>0</del>
<del>(پنیر و مربا)</del>	<del>1</del>
(کره و مربا)	3

مرحله سوم

Item	Count
<del>(نان و الویه و کره)</del>	<del>0</del>
<del>(نان و الویه و مربا)</del>	<del>0</del>
(نان و کره و مربا)	2

در هر مرحله آیتم‌های پر تکرار را نگه داشته و بر روی غیر، خط قرمز کشیدیم. لذا به صورت جمع بندی آیتم‌های پرتکرار به صورت زیر است.

$$\text{total frequent itemsets} = \cup_k L_k$$

$$L_1 = \{\text{نان}, \text{کره}, \text{الویه}, \text{مربا}, \text{پنیر}\}$$

$$L_2 = \{(\text{الویه و نان}), (\text{کره و نان}), (\text{مربا و نان}), (\text{مربا و کره})\}$$

$$L_3 = \{(\text{مربا و کره و نان})\}$$

حال تمامی قواعد انجمنی را می‌نویسیم. (دقت شود که آنهایی که حد آستانه را پاس نکرده بودند دیگر ذکر نمی‌شوند)

توجه شود که شرط minsupport را قبلا بررسی نموده بودیم.

$$\text{مربا} \Rightarrow \text{کره} \quad \text{Confidence} = \frac{3}{3} = 100\% \quad (\text{Confident})$$

$$\text{کره} \Rightarrow \text{مربا} \quad \text{Confidence} = \frac{3}{4} = 75\% \quad (\text{Confident})$$

$$\text{مربا} \Rightarrow \text{نان} \quad \text{Confidence} = \frac{2}{4} = 50\% \quad (\text{Not Confident})$$

$$\text{مربا} \Rightarrow \text{نان} \quad \text{Confidence} = \frac{2}{4} = 50\% \quad (\text{Not Confident})$$

$$\text{مربا} \Rightarrow \text{کره} \quad \text{Confidence} = \frac{2}{3} = 66.6\% \quad (\text{Confident})$$

$$\text{مربا} \Rightarrow \text{کره} \quad \text{Confidence} = \frac{2}{4} = 50\% \quad (\text{Not Confident})$$

$$\text{مربا} \Rightarrow \text{الویه} \quad \text{Confidence} = \frac{2}{4} = 50\% \quad (\text{Not Confident})$$

$$\text{مربا} \Rightarrow \text{الویه} \quad \text{Confidence} = \frac{2}{2} = 100\% \quad (\text{Confident})$$

$$\text{مربا} \Rightarrow \text{کره و نان} \quad \text{Confidence} = \frac{2}{2} = 100\% \quad (\text{Confident})$$

$$\text{مربا} \Rightarrow \text{کره و نان} \quad \text{Confidence} = \frac{2}{4} = 50\% \quad (\text{Not Confident})$$

$$\text{مربا} \Rightarrow \text{کره} \quad \text{Confidence} = \frac{2}{3} = 66.6\% \quad (\text{Confident})$$

$$\text{مربا} \Rightarrow \text{کره} \quad \text{Confidence} = \frac{2}{3} = 66.6\% \quad (\text{Confident})$$

$$\text{مربا} \Rightarrow \text{کره و نان} \quad \text{Confidence} = \frac{2}{2} = 100\% \quad (\text{Confident})$$

حال طبق خواسته سوال، تمامی موارد را به ترتیب اطمینان آنها می‌چینیم.

$$[\text{مربا} \Rightarrow \text{کره}] = [\text{نان} \Rightarrow \text{الویه}] = [\text{مربا} \Rightarrow \text{کره و نان}] = [\text{کره} \Rightarrow \text{مربا و نان}] < [\text{کره} \Rightarrow \text{مربا}] < [\text{نان} \Rightarrow \text{کره}] = [\text{نان} \Rightarrow \text{مربا و کره}] = [\text{مربا و نان} \Rightarrow \text{کره}]$$

## پاسخ سوال 7

برای ساخت درخت fp-tree باید ابتدا آیتم‌های پر تکرار را بیابیم و تکرار هر یک را محاسبه نماییم که در قسمت 6، این کار را انجام دادیم.

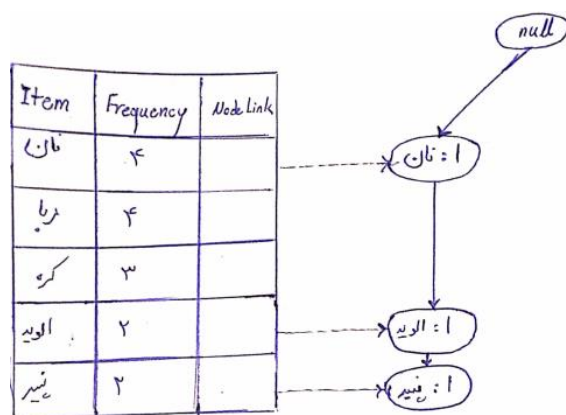
بخش الف



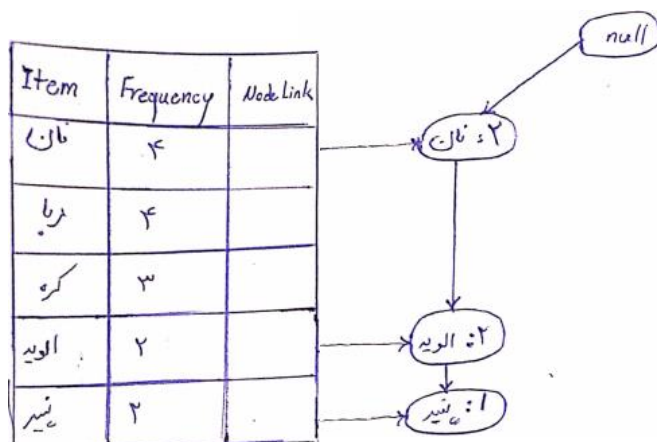
حال باید آیتم ها را به ترتیب تعداد تکرار از بزرگ به کوچک مرتب نماییم. لذا بدین صورت در می آید.

Item	Frequency	Node Link
نان	۴	
ربا	۴	
کره	۳	
الویه	۲	
نسر	۲	

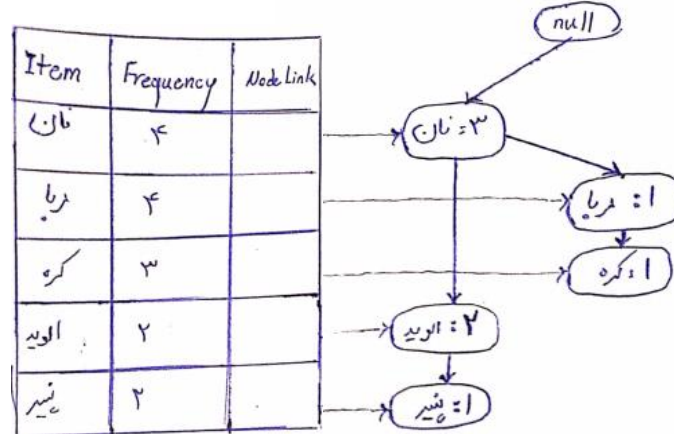
حال درخت را با first transaction آغاز به ساخت می کنیم.



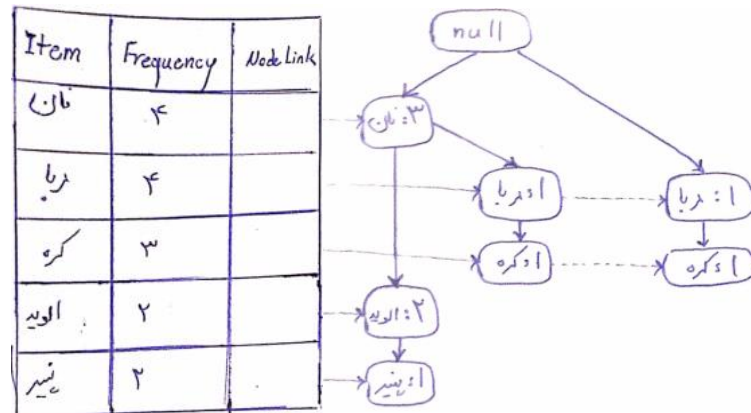
اکنون second transaction را اعمال می نماییم. (تفاوت در اندیس های مقابل هر item نسبت به درخت قبلی ایجاد شده است)



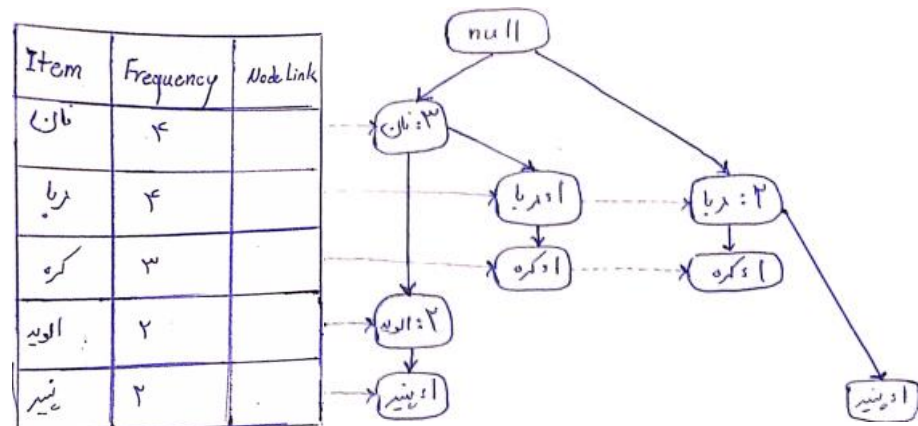
اکنون third transaction را اعمال می‌نماییم.



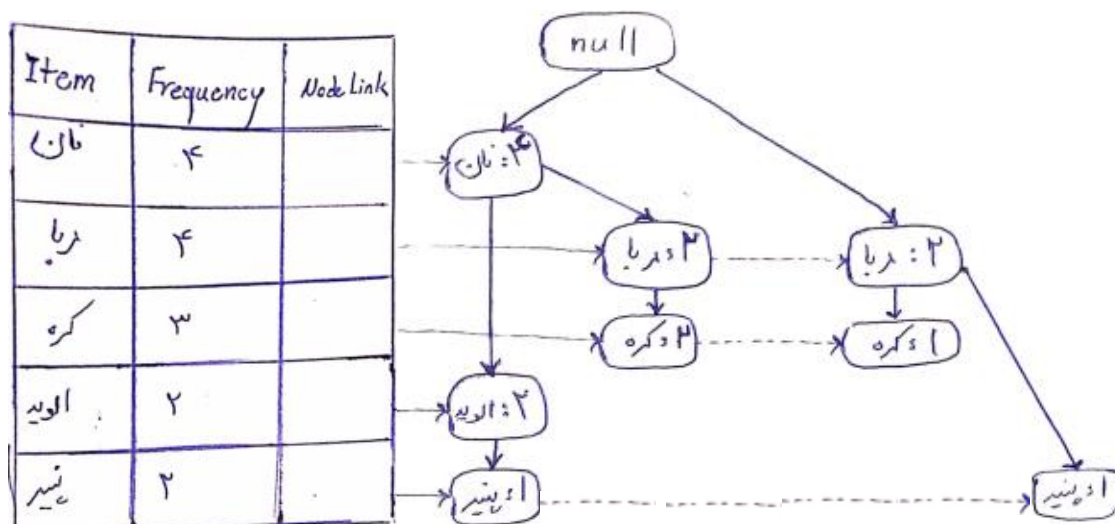
اکنون 4<sup>th</sup> transaction را اعمال می‌کنیم.



کنون 5<sup>th</sup> transaction را اعمال می‌کنیم.



و آخرین transaction اعمال شده به درخت به صورت زیر خواهد بود و درخت در این مرحله تکمیل می‌شود.



#### بخش ب

اکنون که fp-tree را ساختیم می‌توانیم به مانینگ پترن‌های پرتکرار بپردازیم.

پس ما تا این لحظه قسمت اول این الگوریتم را که ساخت fp-tree بود انجام دادیم. حال به مانینگ می‌پردازیم.

مرحله اول:

Items	Conditional pattern base
پنیر	{ {1 : مربا} , {1 : الویه , نان} }
الیویه	{ 2 : نان }
کره	{ {2 : مربا , نان} , {1 : مربا} }
مربا	{ 2 : نان }
نان	-

مرحله دوم: ( آنهایی که مین ساپورت را پاس نمی‌کنند یادداشت نمی‌شوند)

Items	Conditional pattern base	Conditional frequent pattern tree
پنیر	{ {1 : مربا} , {1 : الویه , نان} }	-
الیویه	{ 2 : نان }	{ 2 : نان }
کره	{ {2 : مربا , نان} , {1 : مربا} }	{ {2 : نان} , {3 : مربا} }
مربا	{ 2 : نان }	{ 2 : نان }
نان	-	-

Items	Frequent pattern generated
پنیر	-
الویه	{«2 : الویه , نان»}
کره	{«2 : کره , مربا , نان» , «3 : کره , مربا»}
مربا	{«2 : مربا , نان»}
نان	-

تمامی مجموعه‌های پرتکرار بیش از یکی توسط الگوریتم فوق پیدا شد و از طرفی مجموعه‌های تک آیتمی پرتکرار را نیز در ابتدای الگوریتم در ساخت fp-tree بدست آوردیم لذا مجموعه آیتم‌های پرتکرار ما به صورت زیر است.

Item set	Count
نان	4
مربا	4
کره	3
الویه	2
پنیر	2
مربا, کره	3
نان, کره	2
نان, مربا	2
نان, الویه	2
نان, مربا, کره	2

## پاسخ بخش پیاده‌سازی

برای حل سوالات این بخش یک کلاس با نام Data ایجاد کردیم و در این کلاس ده تابع تعریف نمودیم که هر تابع، پاسخ یک بخش از سوال است. لذا مدلی از کلاس درست کرده و تابع‌های کلاس را فراخوانی نمودیم.

```
if __name__ == "__main__":
    model=Data('players.csv');
    model.printFirstLastRow();
    model.printMissingValue();
    model.printWeightCharacteristics();
    model.printNationalityReport();
    model.promisingPlayer();
    model.plotPromisingPlayer();
    model.teamOfMaxPromisingPlayer();
    model.valueOfChelseaPromissingPlayers();
    model.contractUntilAndNotInNationalTeam();
    model.taremiReport();
```

حال به پاسخ سوالات می‌پردازیم:

## بخش 1

همانطور که در کد قبل دیدیم، فایل اکسل به ورودی کلاس داده می‌شود.

```
def __init__(self, file):  
    self.player = pd.read_csv(file);
```

حال توسط تابع `printFirstLastRow` موارد خواسته شده یعنی سطر اول و آخر را نمایش می‌دهیم.

```
def printFirstLastRow(self):  
    print ('First and last rows\n')  
    flplayer = self.player.iloc[[0, -1]]  
    print (flplayer, '\n')
```

پس از دستور `iloc` استفاده کرده و خروجی به صورت زیر در ترمینال ویندوز، پرینت می‌شود.

```
D:\dars\0000 0000\HW1>C:\Users\mohammad\AppData\Local\Programs\Python\Python36\python.exe main.py  
First and last rows  


|       | ID     | Name          | FullName          | Age | Height | ... | RWBRating | LBRating | CBRating | RBRating | GKRating |
|-------|--------|---------------|-------------------|-----|--------|-----|-----------|----------|----------|----------|----------|
| 0     | 158023 | L. Messi      | Lionel Messi      | 33  | 170    | ... | 69        | 65       | 55       | 65       | 22       |
| 19019 | 241493 | S. Cartwright | Samuel Cartwright | 19  | 185    | ... | 46        | 47       | 51       | 47       | 15       |

  
[2 rows x 90 columns]
```

## بخش 2

برای مشخص شدن `missing value` ها از دستور `isnull()` استفاده کردیم. با این دستور هر جایی که مقداری `miss` شده باشد، به کاربر `true` نمایش داده و هر جایی از جدول که دیتا موجود باشد، کاربر `false` دریافت می‌نماید.

```
def printMissingValue(self):  
    missingvalues = self.player.isnull()  
    print ('missing values\n')  
    print (missingvalues)  
    print ('number of missing values : ', missingvalues.sum().sum(), '\n')
```

نتیجتاً خروجی به صورت زیر خواهد شد. تعداد `missing values` را نیز به کاربر نمایش دادیم.

```
missing values  


|       | ID    | Name  | FullName | Age   | Height | Weight | ... | CDMRating | RWBRating | LBRating | CBRating | RBRating | GKRating |
|-------|-------|-------|----------|-------|--------|--------|-----|-----------|-----------|----------|----------|----------|----------|
| 0     | False | False | False    | False | False  | False  | ... | False     | False     | False    | False    | False    | False    |
| 1     | False | False | False    | False | False  | False  | ... | False     | False     | False    | False    | False    | False    |
| 2     | False | False | False    | False | False  | False  | ... | False     | False     | False    | False    | False    | False    |
| 3     | False | False | False    | False | False  | False  | ... | False     | False     | False    | False    | False    | False    |
| 4     | False | False | False    | False | False  | False  | ... | False     | False     | False    | False    | False    | False    |
| ...   | ...   | ...   | ...      | ...   | ...    | ...    | ... | ...       | ...       | ...      | ...      | ...      | ...      |
| 19015 | False | False | False    | False | False  | False  | ... | False     | False     | False    | False    | False    | False    |
| 19016 | False | False | False    | False | False  | False  | ... | False     | False     | False    | False    | False    | False    |
| 19017 | False | False | False    | False | False  | False  | ... | False     | False     | False    | False    | False    | False    |
| 19018 | False | False | False    | False | False  | False  | ... | False     | False     | False    | False    | False    | False    |
| 19019 | False | False | False    | False | False  | False  | ... | False     | False     | False    | False    | False    | False    |

  
[19020 rows x 90 columns]  
number of missing values : 36492
```

پس تعداد missing value ها برابر با 36492 داده است.

### بخش 3

برای حل این بخش همانطور که از ما خواسته شده بود از امکانات لایبری پانداس در دیتا مانیپینگ استفاده کنیم. لذا ما نیز از دستور describe استفاده کرده و تمامی مشخصات آماری مربوط به وزن بازیگران را استخراج نمودیم.

```
def printWeightCharacteristics(self):  
    print('\n',self.player[self.player.columns[5]].describe(),'\n')  
    #print(self.player.iloc[0,[0,1]])
```

نهایتاً خروجی به صورت زیر خواهد شد.

```
count    19020.000000  
mean      75.052419  
std        7.058038  
min        50.000000  
25%       70.000000  
50%       75.000000  
75%       80.000000  
max       110.000000  
Name: Weight, dtype: float64
```

آنها را در جدول زیر بازنویسی می کنیم.

Mean Weight	75.052419
Min Weight	50
Max Weight	110

### بخش 4

در این بخش، مسأله از ما خواسته است که کشوری را که دارای بیشتری بازیکنان و کمترین بازیکنان است را بیابیم. لذا به سادگی یک دیکشنری با نام ملیت تعریف نمودیم و کلید آن را ملیت و مقدار آن را از صفر شروع کرده و به ازای هر بار خواندن آن، یکی به آن ملیت اضافه نمودیم. ( البته بهتر بود اسمش را کشور قرار می دادم 😊 )

سپس دیکشنری را بر حسب مقادیر value ها سورت کرده و ابتدا و انتهای آن را پرینت کردیم.

```
def printNationalityReport(self):
    print('\n')
    nationality = {}
    for n in self.player.iloc[:,7]:
        if n not in nationality:
            nationality[n] = 1;
        else:
            nationality[n] = int(nationality[n]) + 1;
    sorted_nationality = {k: v for k, v in sorted(nationality.items(), key=lambda item: item[1])}

    print('nationality of manimum players')
    print(list(sorted_nationality)[1], sorted_nationality[list(sorted_nationality)[1]])
    print('nationality of maximum players')
    print(list(sorted_nationality)[-1], sorted_nationality[list(sorted_nationality)[-1]], '\n')
```

نهایتاً خروجی به صورت زیر قابل روئت است.

```
nationality of manimum players
Chad 1
nationality of maximum players
England 1706
```

آن را مجدداً در جدول زیر باز نویسی می کنیم.

nation of minimum players	Chad : 1
nationality of maximum players	England : 1706

## بخش 5

در این بخش مسأله از ما خواسته است که آینده دارترین بازیکنان را بر اساس معیار potential بالای 84 و growth بالای 4 را از داده ها استخراج کنیم. و آن را گزارش کنیم.

```
def promisingPlayer(self):
    x = list(self.player.iloc[:,9])
    y = list(self.player.iloc[:,10])
    #print(len(x))
    for n in range(len(self.player)):
        if int(x[n])>84 and int(y[n])>4:
            print('\n', n, self.player.iloc[n,1],x[n],y[n])
    print('\n')
```

در اینجا چون نیاز به مقایسه تک تک داده های ستون بود، آنها را به صورت list قرار داده و مقایسه را انجام دادیم.

نتایج این بازیکنان به صورت زیر است (دو عدد سمت راست بازیگر، عدد اول potential و عدد دوم growth است).

عدد شمت چپ ترین نیز شماره در جدول است.

12 K. Mbappé 95 5	583 Dani Olmo 87 8	1873 M. Boadu 86 11
45 T. Alexander-Arnold 92 5	589 E. Tapsoba 88 9	1970 Arthur Cabral 85 10

46 J. Sancho 93 6	590 Antony 88 9	2082 Nuno Mendes 86 12
68 L. Sané 90 5	622 C. Pavón 85 6	2100 E. Barco 85 11
70 F. de Jong 90 5	623 S. Berge 85 6	2128 Gabriel Martinelli 85 11
71 G. Donnarumma 92 7	634 R. Bentancur 85 6	2218 N. Bustos 85 11
72 M. Rashford 91 6	639 D. Čaleta-Car 85 6	2226 R. Sessegnon 85 11
75 M. de Ligt 92 7	735 Trincão 88 10	2327 C. Hudson-Odoi 86 12
76 K. Havertz 93 8	739 Emerson 88 10	2354 Gustavo Assunção 86 12
77 E. Haaland 92 7	740 D. Kulusevski 88 10	2398 G. Plata 86 12
115 Oyarzabal 89 5	753 L. Díaz 87 9	2406 W. Saliba 87 13
118 L. Martínez 91 7	755 M. Senesi 86 8	2416 M. Solomon 86 12
141 N. Süle 88 5	756 D. Malen 86 8	2439 G. Diangana 85 11
149 M. Ødegaard 89 6	760 I. Sarr 87 9	2451 G. Reyna 88 14
150 F. Valverde 90 7	765 C. Stengs 86 8	2467 A. Hložek 87 13
154 F. Mendy 88 5	766 Pedro Gonçalves 87 9	2538 Brahim 86 12
156 A. Wan-Bissaka 88 5	769 I. Konaté 88 10	2609 R. Aït Nouri 86 13
159 O. Dembélé 89 6	770 R. James 86 8	2612 J. Bellingham 88 15
160 A. Hakimi 88 5	773 N. De la Cruz 85 7	2634 Romário Baró 85 12
161 J. Gomez 88 5	778 G. Montiel 85 7	2639 A. Urzi 88 15
162 T. Strakosha 88 5	799 E. Camavinga 89 11	2659 Pedri 89 16
188 Gayà 88 5	824 T. Abraham 85 7	2705 Eric García 85 12
257 Fabián 88 6	825 D. McNeil 85 7	2769 Barrenetxea 85 12
259 Reguilón 87 5	839 A. Meret 86 8	2823 C. De Ketelaere 86 13
260 João Félix 93 11	843 K. Tierney 86 8	2903 Marcos Antonio 86 13
261 H. Aouar 89 7	895 Bruno Guimarães 86 8	2910 W. Fofana 85 12
264 G. Lo Celso 87 5	905 K. Diatta 85 8	2911 F. Wirtz 88 15
265 S. Bergwijn 87 5	909 Jovane Cabral 86 9	2923 Rafael Camacho 86 13
266 P. Kimpembe 87 5	927 D. Szoboszlai 87 10	3010 Evanilson 86 13
268 Rúben Neves 87 5	932 M. Greenwood 89 12	3115 Tomás Tavares 85 12
274 Richarlison 88 7	943 M. Guendouzi 86 9	3240 L. Pellegrini 86 14
276 A. Davies 89 8	968 S. Tonali 89 12	3341 G. Tsitaishvili 86 14
278 T. Souček 86 5	992 E. Palacios 85 8	3412 Nuno Tavares 85 13
279 Cucurella 89 8	994 Jovane Cabral 86 9	3434 M. Longstaff 85 13
280 Rúben Dias 87 6	995 O. Wijndal 85 8	3614 Y. Verschaeren 85 13
281 Renan Lodi 87 6	997 S. Tonali 89 12	3690 Daniel Bragança 86 14
282 D. Livaković 86 5	998 M. Olivera 85 8	3691 Eduardo Quaresma 85 13
283 M. Diaby 88 7	999 Galeno 85 8	3709 N. Madueke 86 14
323 O. Vlachodimos 86 5	1000 K. Diatta 85 8	4001 F. Pellistri 87 16
353 D. Calvert-Lewin 87 6	1002 D. Szoboszlai 87 10	4058 Reinier 86 15
354 Everton 86 5	1003 M. Guendouzi 86 9	4080 V. Kornienko 85 14
357 N. Barella 88 7	1005 Luiz Felipe 85 8	4266 B. Gilmour 86 15
362 C. Pulisic 87 6	1011 J. Bijlow 86 9	4336 T. Nianzou 85 14
364 Gonçalo Guedes 87 6	1012 J. David 88 11	4548 P. De la Vega 85 14
379 V. Tsygankov 86 6	1013 U. Račić 85 8	4583 J. Willock 85 14
380 M. Gómez 85 5	1036 M. Kudus 87 10	4617 H. Traorè 86 15
381 Éder Militão 87 7	1053 M. Greenwood 89 12	4618 J. Doku 88 17
382 N. Vlašić 86 6	1073 M. Ihattaren 86 9	5379 J. Gvardiol 86 16
383 Ferran Torres 88 8	1074 S. Chukwueze 86 9	5522 K. Adeyemi 86 17
384 Pau Torres 86 6	1084 Galeno 85 8	5702 A. Velasco 85 16
385 J. Koundé 86 6	1096 M. Olivera 85 8	5876 R. Cherki 88 19
421 David Neres 85 5	1110 E. Palacios 85 8	5908 C. Tzolis 87 18
432 Vinícius Jr. 93 13	1116 J. David 88 11	6061 E. Smith Rowe 85 16
433 M. Arambarri 85 5	1124 J. Boga 86 9	6064 Abel Ruiz 86 17
434 T. Ndombele 87 7	1127 M. Kudus 87 10	6069 Sergio Gómez 85 16
435 Palhinha 85 5	1138 S. Chukwueze 86 9	6200 Joelson Fernandes 87 18
437 L. Bailey 85 5	1145 M. Ihattaren 86 9	6425 Fábio Silva 85 16
451 D. Upamecano 90 10	1151 J. Veerman 85 8	6609 C. Jones 86 18



453 F. Neuhaus 86 6	1153 J. Veerman 85 8	6803 J. Musiala 85 17
454 M. Thuram 86 6	1161 M. Caqueret 86 10	6834 N. Williams 85 17
458 A. Saint-Maximin 86 6	1194 M. Demiral 85 9	6904 M. Vandevordt 86 18
462 L. Klostermann 85 5	1229 L. Martínez Quarta 85 9	7918 H. Elliott 85 18
465 Diogo Jota 86 6	1235 Tete 86 10	8568 L. Stergiou 86 19
466 N. Mukiele 87 7	1279 B. Soumaré 85 9	8750 T. Parrott 85 19
479 Unai Simón 86 6	1315 B. Saka 88 12	8970 S. Esposito 86 20
480 C. Söyüncü 85 5	1357 W. McKennie 85 9	10060 X. Simons 85 20
481 C. Nkunku 86 6	1385 Diogo Dalot 85 9	10650 Y. Demir 86 21
482 L. Jović 85 5	1386 Ansu Fati 90 14	12472 J. Branthwaite 85 22
483 T. Hernández 85 5	1422 F. Tomori 85 9	12798 L. Netz 85 22
484 M. Mount 87 7	1435 B. White 87 11	14022 T. Nakai 86 24
486 D. Henderson 87 7	1438 N. Zaniolo 86 10	14290 B. Arrey-Mbi 85 24
488 H. Barnes 85 5	1459 Pedro Neto 85 9	
516 Renato Sanches 85 6	1464 O. Kabak 86 10	
518 V. Osimhen 87 8	1520 I. Sangaré 85 10	
519 Gabriel 85 6	1541 A. Mac Allister 85 10	
520 A. Isak 86 7	1548 J. Todibo 86 11	
523 Carlos Soler 85 6	1566 Óscar 86 11	
524 D. Rice 86 7	1583 J. Reine-Adélaïde 85 10	
525 J. Ikoné 86 7	1584 S. Dest 86 11	
526 B. Kamara 85 6	1613 T. Almada 89 14	
529 P. Foden 88 9	1628 T. Kubo 88 13	
532 D. Zagadou 86 7	1666 Riqui Puig 88 13	
551 M. Locatelli 85 6	1726 Pedrinho 86 11	
553 P. Estupiñán 86 7	1728 A. Lunin 87 12	
558 Adama Traoré 85 6	1732 N. Pérez 85 10	
578 Matheus Cunha 87 8	1866 R. Gravenberch 89 14	
580 Junior Firpo 85 6		
582 Rodrygo 90 11		

## بخش 6

در این قسمت خواسته شده که نمودار بازیکنان آینده دار را بر اساس موقعیتشان گزارش کنیم. لذا بدلیل حجم بالا از ابزار boxplot استفاده کردیم و روند حل هم بدین صورت است که یک دیکشنری خالی را تعریف کردیم و موقعیت به صورت key و potential ها هم به صورت value و در قالب یک لیست به آن اضافه نمودیم و تا انتهای پیمایش ستون مربوط به position ها، این روند را ادامه دادیم.

کد به صورت زیر است. (بعد از آن شکل رسم شده قرار گرفته است.)

```

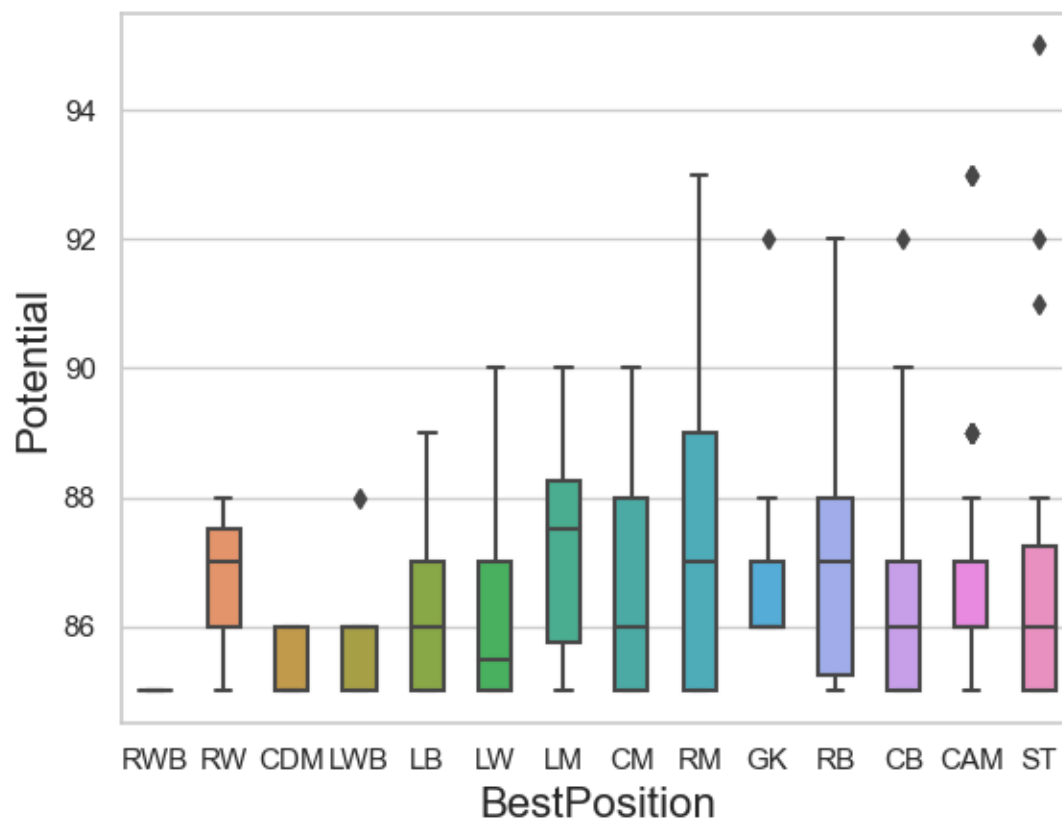
def plotPromisingPlayer(self):
    promising = {};
    x = list(self.player.iloc[:,9])
    y = list(self.player.iloc[:,10])
    for n in range(len(self.player)):
        if int(x[n])>84 and int(y[n])>4:
            for p in self.player.iloc[n,14].split(','):
                if p not in promising:
                    promising[p] = [self.player.iloc[n,9]]
                else:
                    promising[p].append(self.player.iloc[n,9])

    #print(promising)
    sorted_keys, sorted_vals = zip(*sorted(promising.items(), key=op.itemgetter(1)))

    # almost verbatim from question
    sns.set(context='notebook', style='whitegrid')
    sns.utils.axlabel(xlabel="BestPosition", ylabel="Potential", fontsize=16)
    sns.boxplot(data=sorted_vals, width=.48)
    #sns.swarmplot(data=sorted_vals, size=6, edgecolor="black", linewidth=.9)
    plt.xticks(plt.xticks()[0], sorted_keys)
    plt.show()

```

نهایتاً خروجی به صورت زیر plot نمودیم.



## بخش 7

در این بخش از ما خواسته شده است که بررسی کنیم کدام باشگاه دارای بیشترین تعداد بازیکنان آینده دار است. لذا در اینجا نیز یک دیکشنری تعریف نموده و key آن را برابر با باشگاه ها یا همان club قرار داده و value آن را به ازای هر بار پیمایش مربوط به آن باشگاه در پیمایش بازیکنان آینده دار، یکی اضافه نموده و در انتها که پیمایش به صورت کامل انجام شود، نهایتاً تعداد بازیکنان آینده دار متعلق به هر باشگاه بدست آمد.

```
def teamOfMaxPromisingPlayer(self):
    promising = {}
    x = list(self.player.iloc[:,9])
    y = list(self.player.iloc[:,10])
    for n in range(len(self.player)):
        if int(x[n])>84 and int(y[n])>4:
            p = self.player.iloc[n,15]
            if p not in promising:
                promising[p] = 1
            else:
                promising[p] = int(promising[p]) + 1

    sorted_promising = {k: v for k, v in sorted(promising.items(), key=lambda item: item[1])}
    print(sorted_promising, '\n')
```

همچنین در آخر نیز عملیات sort را روی value های این دیکشنری انجام دادیم.

نهایتاً خروجی به صورت زیر است:

```
{'PFC CSKA Moscow': 1, 'Athletic Club de Bilbao': 1, 'LA Galaxy': 1, 'Sheffield United': 1, 'Real Betis': 1, 'Watford': 1, 'Burnley': 1, 'Fiorentina': 1, 'Roma': 1, 'FC Schalke 04': 1, 'OGC Nice': 1, 'Vélez Sarsfield': 1, 'Granada CF': 1, 'FC Basel 1893': 1, 'Atlanta United': 1, 'Girona FC': 1, 'TSG 1899 Hoffenheim': 1, 'Famalicão': 1, 'West Bromwich Albion': 1, 'Sparta Praha': 1, 'Club Atlético Banfield': 1, 'Deportivo Alavés': 1, 'Genoa': 1, 'RSC Anderlecht': 1, 'Club Atlético Lanús': 1, 'Independiente': 1, 'PAOK': 1, 'SD Huesca': 1, 'KRC Genk': 1, 'Blackburn Rovers': 1, 'FC St. Gallen': 1, 'Millwall': 1, 'SPAL': 1, 'SK Rapid Wien': 1, 'Bayern München II': 1, 'Lazio': 2, 'Atlético Madrid': 2, 'West Ham United': 2, 'Dinamo Zagreb': 2, 'Dynamo Kyiv': 2, 'Sevilla FC': 2, 'Borussia Mönchengladbach': 2, 'Newcastle United': 2, 'Olympique de Marseille': 2, 'Feyenoord': 2, 'River Plate': 2, 'Stade Rennais FC': 2, 'SC Heerenveen': 2, 'Brighton & Hove Albion': 2, 'Paris Saint-Germain': 3, 'Real Sociedad': 3, 'Inter': 3, 'Napoli': 3, 'Everton': 3, 'Leicester City': 3, 'Sassuolo': 3, 'FC Porto': 3, 'AZ Alkmaar': 3, 'Club Brugge KV': 3, 'FC Red Bull Salzburg': 3, 'SC Braga': 3, 'Tottenham Hotspur': 4, 'Olympique Lyonnais': 4, 'Getafe CF': 4, 'Manchester City': 4, 'Hertha BSC': 4, 'Shakhtar Donetsk': 4, 'Liverpool': 5, 'FC Bayern München': 5, 'Juventus': 5, 'Valencia CF': 5, 'Wolverhampton Wanderers': 5, 'SL Benfica': 5, 'Villarreal CF': 5, 'Ajax': 5, 'LOSC Lille': 5, 'PSV': 5, 'Borussia Dortmund': 6, 'Milan': 6, 'Manchester United': 6, 'Bayer 04 Leverkusen': 6, 'RB Leipzig': 6, 'Arsenal': 7, 'FC Barcelona': 8, 'Chelsea': 8, 'Real Madrid': 9, 'Sporting CP': 10}
```

دیکشنری فوق به صورت سورت شده پرینت شده است و لذا همانطور که مشاهده می فرمایید، بیشترین تعداد بازیکنان آینده دار با تعداد 10 بازیکن، متعلق به Sporting CP است.

## بخش 8

در این بخش مجموع ارزش بازیکنان آینده دار چلسی را از ما خواسته است. لذا بحث استخراج بازیکنان آینده دار در این تابع نیز مثل روال گذشته انجام می شود ولی در هر بار یک مقایسه صورت می گیرد که آیا باشگاه مربوط به این بازیکن چلسی است یا خیر. در صورتی که چلسی باشد مقادیر با یکدیگر جمع کرده و در نهایت به کاربر نمایش داده می شود. این متغیر که مجموع ارزش ها را در آن می ریزیم را با value نامیده ایم.

```
def valueOfChelseaPromissingPlayers(self):
    value = 0;
    x = list(self.player.iloc[:,9])
    y = list(self.player.iloc[:,10])
    for n in range(len(self.player)):
        if int(x[n])>84 and int(y[n])>4:
            p = self.player.iloc[n,15]
            if p == 'Chelsea':
                value = value + int(self.player.iloc[n,16])
            #print(p)
    print(value)
```

همانطور که مشاهده می‌نمایید، داده‌ها از ستون 16 یعنی ValueEUR خوانده شده‌اند.

نهایتاً مقدار value به صورت زیر در می‌آید.

```
293900000
```

پس مجموع ارزش بازیکنان چلسی برابر با 293900000 خواهد بود.

## بخش 9

این قسمت دقیقاً مانند قسمت 5 است با این تفاوت که ستون‌ها NationalTeam و contractUtil می‌باشد و البته مقایسه در اینجا مقایسه‌ی بین string هاست مگر مراحل مانند قسمت 5 است و در اینجا نتایج را نمایش می‌دهیم.

```
def contractUntilAndNotInNationalTeam(self):
    num = 0;
    i=0;
    x = list(self.player.iloc[:,20])
    y = list(self.player.iloc[:,24])
    #print(self.player.iloc[:,20])
    #print(self.player.iloc[:,24])
    for n in range(len(self.player)):
        #i+=1
        if x[n]== 2021.0 and y[n]=='Not in team':
            #print(i)
            num +=1
            #print(self.player.iloc[n,1])
    print('\n', num, '\n')
```

در نهایت تعداد این بازیکنان را در متغیر num ریخته و آن را پرینت می‌کنیم.

```
6727
```

لذا تعداد بازیکنانی که در سال 2021 قراردادشان با باشگاهشان تمام میشود و همچنین در تیم ملی کشورشان حضور ندارند برابر با 6727 بازیگر است.

## بخش 10

این بخش که بخشِ آخرِ مسأله بود از ما خواسته که موقعیت، درآمد و باشگاه فعلی مهدی طارمی را پرینت کنیم. البته ما از بهترین موقعیتِ آن نیز پرینت گرفتیم. در این قسمت ابتدا سطر مربوطه به مهدی طارمی را در قسمت نام‌ها Name جستجو کردیم (M. Taremi) و نهایتاً ستون‌های خواسته شده در صورت سوال را استخراج کرده و به کاربر نمایش دادیم.

```
def taremiReport(self):
    report = self.player.loc[self.player['Name'] == 'M. Taremi']
    print('\n Mehdi Taremi reports\n')
    print('Positions')
    print(report.iloc[:,13], '\n')
    print('Best Position')
    print(report.iloc[:,14], '\n')
    print('WageEUR')
    print(report.iloc[:,17], '\n')
    print('Club')
    print(report.iloc[:,15])
```

نهایتاً خروجی به صورت زیر بدست آمد.

```
Mehdi Taremi reports

Positions
1017    ST,CF
1113    ST,CF
Name: Positions, dtype: object

Best Position
1017    ST
1113    ST
Name: BestPosition, dtype: object

WageEUR
1017    16000
1113    16000
Name: WageEUR, dtype: int64

Club
1017    FC Porto
1113    FC Porto
Name: Club, dtype: object
```

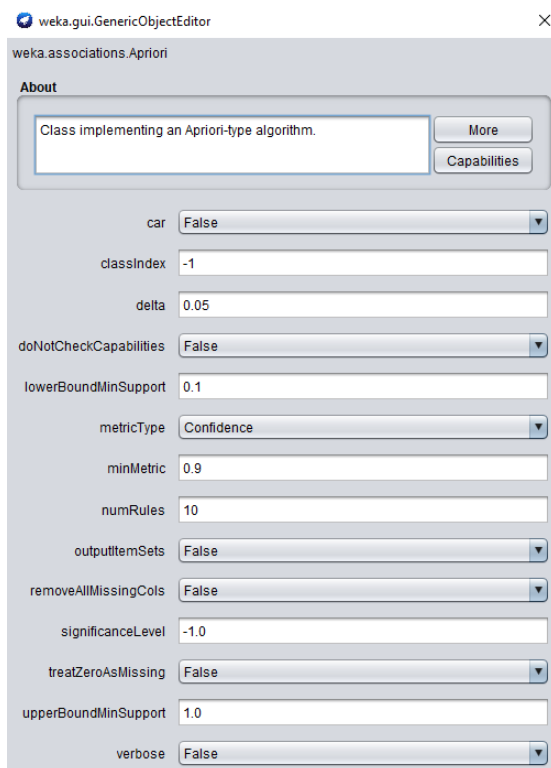
چون دو سطر جدول مربوط به مهدی طارمی بود لذا هر کدام دوتایی چاپ شده است.

نتایج را مجدداً در زیر بازنویسی می‌کنیم.

Position	ST,CF
Best Position	ST
WageEUR	16000
Club	FC Porto

## پاسخ قسمت قوانین انجمنی

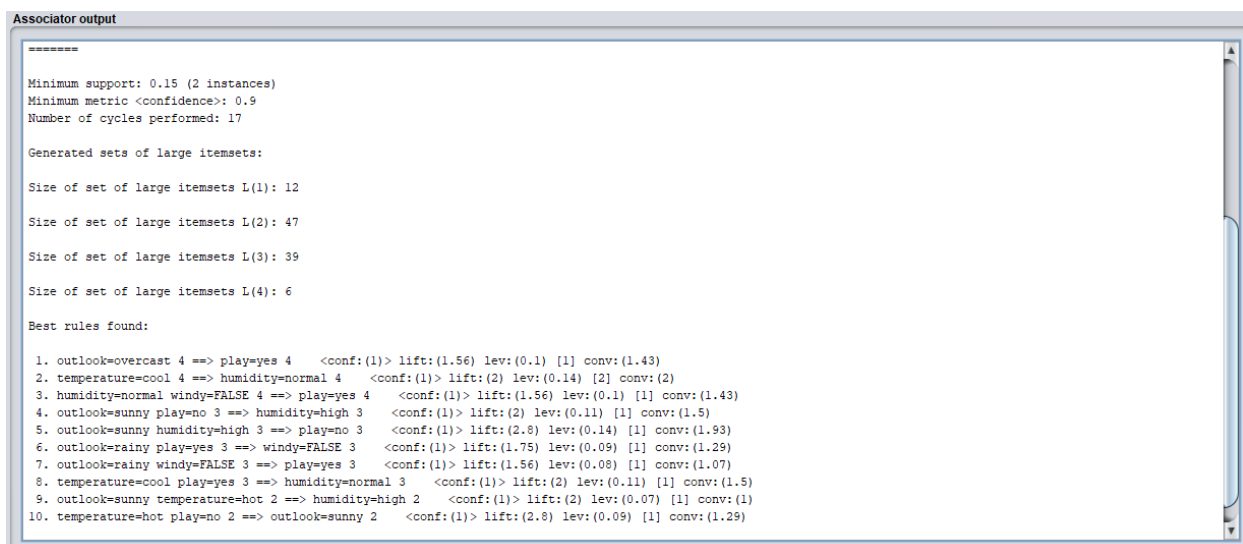
ابتدا نرم افزار وکا را نصب کرده و حالت دیفالت آن را در نظر می گیریم.



The screenshot shows the 'weka.gui.GenericObjectEditor' window with the 'Apriori' algorithm selected. The 'About' tab is active, displaying the description: 'Class implementing an Apriori-type algorithm.' Below this, there are several configuration options:

- car: False
- classIndex: -1
- delta: 0.05
- doNotCheckCapabilities: False
- lowerBoundMinSupport: 0.1
- metricType: Confidence
- minMetric: 0.9
- numRules: 10
- outputItemSets: False
- removeAllMissingCols: False
- significanceLevel: -1.0
- treatZeroAsMissing: False
- upperBoundMinSupport: 1.0
- verbose: False

در حالت دیفالت نرم افزار نتایج به صورت زیر است.



The screenshot shows the 'Associator output' window with the following text:

```
=====
Minimum support: 0.15 (2 instances)
Minimum metric <confidence>: 0.9
Number of cycles performed: 17

Generated sets of large itemsets:

Size of set of large itemsets L(1): 12
Size of set of large itemsets L(2): 47
Size of set of large itemsets L(3): 39
Size of set of large itemsets L(4): 6

Best rules found:

1. outlook=overcast 4 ==> play=yes 4 <conf:(1)> lift:(1.56) lev:(0.1) [1] conv:(1.43)
2. temperature=cool 4 ==> humidity=normal 4 <conf:(1)> lift:(2) lev:(0.14) [2] conv:(2)
3. humidity=normal windy=FALSE 4 ==> play=yes 4 <conf:(1)> lift:(1.56) lev:(0.1) [1] conv:(1.43)
4. outlook=sunny play=no 3 ==> humidity=high 3 <conf:(1)> lift:(2) lev:(0.11) [1] conv:(1.5)
5. outlook=sunny humidity=high 3 ==> play=no 3 <conf:(1)> lift:(2.8) lev:(0.14) [1] conv:(1.93)
6. outlook=rainy play=yes 3 ==> windy=FALSE 3 <conf:(1)> lift:(1.75) lev:(0.09) [1] conv:(1.29)
7. outlook=rainy windy=FALSE 3 ==> play=yes 3 <conf:(1)> lift:(1.56) lev:(0.08) [1] conv:(1.07)
8. temperature=cool play=yes 3 ==> humidity=normal 3 <conf:(1)> lift:(2) lev:(0.11) [1] conv:(1.5)
9. outlook=sunny temperature=hot 2 ==> humidity=high 2 <conf:(1)> lift:(2) lev:(0.07) [1] conv:(1)
10. temperature=hot play=no 2 ==> outlook=sunny 2 <conf:(1)> lift:(2.8) lev:(0.09) [1] conv:(1.29)
```

حال 5 پارامترها تغییر داده و تأثیر آنها در خروجی را بررسی می‌کنیم.

## حالت اول (تغییر minconf)

پارامتر minconf را از 0.9 به 0.3 کاهش داده و نتیجه را بررسی می‌کنیم.

The screenshot shows the 'About' dialog box for the Apriori algorithm. The 'minMetric' field is highlighted and set to 0.3. Other fields include 'car' (False), 'classIndex' (-1), 'delta' (0.05), 'doNotCheckCapabilities' (False), 'lowerBoundMinSupport' (0.1), 'metricType' (Confidence), 'numRules' (10), 'outputItemSets' (False), 'removeAllMissingCols' (False), 'significanceLevel' (-1.0), 'treatZeroAsMissing' (False), 'upperBoundMinSupport' (1.0), and 'verbose' (False).

خروجی به صورت زیر است.

```
Apriori
*****

Minimum support: 0.3 (4 instances)
Minimum metric <confidence>: 0.3
Number of cycles performed: 14

Generated sets of large itemsets:

Size of set of large itemsets L(1): 12
Size of set of large itemsets L(2): 9
Size of set of large itemsets L(3): 1

Best rules found:

1. outlook=overcast 4 ==> play=yes 4 <conf:(1)> lift:(1.56) lev:(0.1) [1] conv:(1.43)
2. temperature=cool 4 ==> humidity=normal 4 <conf:(1)> lift:(2) lev:(0.14) [2] conv:(2)
3. humidity=normal windy=FALSE 4 ==> play=yes 4 <conf:(1)> lift:(1.56) lev:(0.1) [1] conv:(1.43)
4. humidity=normal 7 ==> play=yes 6 <conf:(0.86)> lift:(1.33) lev:(0.11) [1] conv:(1.25)
5. play=no 5 ==> humidity=high 4 <conf:(0.8)> lift:(1.6) lev:(0.11) [1] conv:(1.25)
6. windy=FALSE 8 ==> play=yes 6 <conf:(0.75)> lift:(1.17) lev:(0.06) [0] conv:(0.95)
7. play=yes 9 ==> humidity=normal 6 <conf:(0.67)> lift:(1.33) lev:(0.11) [1] conv:(1.13)
8. play=yes 9 ==> windy=FALSE 6 <conf:(0.67)> lift:(1.17) lev:(0.06) [0] conv:(0.96)
9. temperature=mild 6 ==> humidity=high 4 <conf:(0.67)> lift:(1.33) lev:(0.07) [1] conv:(1)
10. temperature=mild 6 ==> play=yes 4 <conf:(0.67)> lift:(1.04) lev:(0.01) [0] conv:(0.71)
```

ابتدا مشاهده می‌کنیم که با کاهش minconf، تعداد سیکل‌های پردازش شده از 17 تا به 14 عدد کاهش یافته است. همچنین اگرچه در ساینز آیت‌مست L(1) تغییری حاصل نشده است ولی ساینز آیت‌مست‌های بعدی، کاهش چشمگیری داشته است.

به گونه‌ای که  $L(2)$  و  $L(3)$  و  $L(4)$ ، هر یک به ترتیب از 47 و 39 و 6 به 9، 1 و صفر کاهش سایز داشته است که البته تأثیر آن را می‌توان به اثری که بر روی افزایش minsup گذاشته است اشاره کرد و این مورد کاملاً قابل پیش‌بینی بود.

یعنی یکی از اثراتی که در این حالت نسبت به دیفالت مشاهده می‌شود، همین افزایش minsup از 0.15 به 0.3 است.

## حالت دوم (تغییر numRules)

در این حالت تعداد قوانین انجمنی را از 10 عدد به 3 عدد کاهش می‌دهیم و نتیجه را می‌بینیم.

```
Associator output
temperature
humidity
windy
play
=== Associator model (full training set) ===

Apriori
=====
Minimum support: 0.3 (4 instances)
Minimum metric <confidence>: 0.9
Number of cycles performed: 14

Generated sets of large itemsets:

Size of set of large itemsets L(1): 12
Size of set of large itemsets L(2): 9
Size of set of large itemsets L(3): 1

Best rules found:
1. outlook=overcast 4 ==> play=yes 4    <conf:(1)> lift:(1.56) lev:(0.1) [1] conv:(1.43)
2. temperature=cool 4 ==> humidity=normal 4    <conf:(1)> lift:(2) lev:(0.14) [2] conv:(2)
3. humidity=normal windy=FALSE 4 ==> play=yes 4    <conf:(1)> lift:(1.56) lev:(0.1) [1] conv:(1.43)
```

دقت بفرمایید که این حالت فقط در numRules با حالت دیفالت فرق دارد. لذا همانطور که مشاهده می‌کنیم مثل حالت اول تعداد سیکل‌های پردازش شده از 17 تا به 14 کاهش یافته و همچنین  $L(2)$  و  $L(3)$  و  $L(4)$ ، هر یک به ترتیب از 47 و 39 و 6 به 9، 1 و صفر کاهش سایز داشته است که البته تأثیر آن را می‌توان باز به minsup ارجاع داد که از 0.15 به 0.3 افزایش یافته است.

## حالت سوم (تغییر lowerBoundMinSupport)

در این قسمت نیز تنها تفاوتی که با حالت دیفالت ایجاد نمودیم، افزایش حد پایین minsupport است که انتظار داریم در صورتی که باعث محدود کردن support سیستم شود که می‌شود (چون در حالت دیفالت minsupport را سیستم 0.15 می‌گذاشت)، لذا سایز مجموعه آیت‌مست‌ها کاهش پیدا کند. حال نتیجه را می‌بینیم که آیا اینطور است یا خیر.



```
Associator output

Apriori
=====

Minimum support: 0.2 (3 instances)
Minimum metric <confidence>: 0.9
Number of cycles performed: 16

Generated sets of large itemsets:

Size of set of large itemsets L(1): 12
Size of set of large itemsets L(2): 26
Size of set of large itemsets L(3): 4

Best rules found:

1. outlook=overcast 4 ==> play=yes 4 <conf:(1)> lift:(1.56) lev:(0.1) [1] conv:(1.43)
2. temperature=cool 4 ==> humidity=normal 4 <conf:(1)> lift:(2) lev:(0.14) [2] conv:(2)
3. humidity=normal windy=FALSE 4 ==> play=yes 4 <conf:(1)> lift:(1.56) lev:(0.1) [1] conv:(1.43)
4. outlook=sunny play=no 3 ==> humidity=high 3 <conf:(1)> lift:(2) lev:(0.11) [1] conv:(1.5)
5. outlook=sunny humidity=high 3 ==> play=no 3 <conf:(1)> lift:(2.8) lev:(0.14) [1] conv:(1.93)
6. outlook=rainy play=yes 3 ==> windy=FALSE 3 <conf:(1)> lift:(1.75) lev:(0.09) [1] conv:(1.29)
7. outlook=rainy windy=FALSE 3 ==> play=yes 3 <conf:(1)> lift:(1.56) lev:(0.08) [1] conv:(1.07)
8. temperature=cool play=yes 3 ==> humidity=normal 3 <conf:(1)> lift:(2) lev:(0.11) [1] conv:(1.5)
```

همانطور که مشاهده می‌فرمایید دقیقاً پیش‌بینی ما درست بود. دقت کنید که بیشتر نباید این حد پایین را افزایش داد چون سائز تمامی مجموعه آیت‌ها را کاهش داده تا به صفر برسند. حال به اثرات دیگر توجه کنید که سیکل‌های پردازش‌شده نیز در این حالت یکی کاهش یافته. همچنین تفاوت این حالت نسبت به دیفالت این است که اگرچه ما تعداد قواعد را 10 قرار داده بودیم ولی قواعدی که سیستم استخراج کرده برابر با 8 عدد است. این یعنی حد پایین minsupport در صورت افزایش احتمالاً در کاهش پیدا کردن قواعد اثر می‌گذارد.

## حالت چهارم (تغییر removeAllMissingCols)

```
Associator output

Minimum support: 0.15 (2 instances)
Minimum metric <confidence>: 0.9
Number of cycles performed: 17

Generated sets of large itemsets:

Size of set of large itemsets L(1): 12
Size of set of large itemsets L(2): 47
Size of set of large itemsets L(3): 39
Size of set of large itemsets L(4): 6

Best rules found:

1. outlook=overcast 4 ==> play=yes 4 <conf:(1)> lift:(1.56) lev:(0.1) [1] conv:(1.43)
2. temperature=cool 4 ==> humidity=normal 4 <conf:(1)> lift:(2) lev:(0.14) [2] conv:(2)
3. humidity=normal windy=FALSE 4 ==> play=yes 4 <conf:(1)> lift:(1.56) lev:(0.1) [1] conv:(1.43)
4. outlook=sunny play=no 3 ==> humidity=high 3 <conf:(1)> lift:(2) lev:(0.11) [1] conv:(1.5)
5. outlook=sunny humidity=high 3 ==> play=no 3 <conf:(1)> lift:(2.8) lev:(0.14) [1] conv:(1.93)
6. outlook=rainy play=yes 3 ==> windy=FALSE 3 <conf:(1)> lift:(1.75) lev:(0.09) [1] conv:(1.29)
7. outlook=rainy windy=FALSE 3 ==> play=yes 3 <conf:(1)> lift:(1.56) lev:(0.08) [1] conv:(1.07)
8. temperature=cool play=yes 3 ==> humidity=normal 3 <conf:(1)> lift:(2) lev:(0.11) [1] conv:(1.5)
9. outlook=sunny temperature=hot 2 ==> humidity=high 2 <conf:(1)> lift:(2) lev:(0.07) [1] conv:(1)
10. temperature=hot play=no 2 ==> outlook=sunny 2 <conf:(1)> lift:(2.8) lev:(0.09) [1] conv:(1.29)
```

همانطور که مشاهده می‌فرمایید این پارامتر تغییری در سیستم ایجاد نکرده است. نه سائز آیت‌ها نه قوانین و نه در تعیین پارامتر minsup. لذا دو مورد در این سیستم از این نتیجه قابل برداشت است. اول اینکه سیستم missing value نداشته باشد. حالت دوم این می‌تواند باشد که این پارامتر تاثیری نمی‌گذارد. به نظر می‌رسد که منظور کلی ستون با بقیه‌ی داده‌های موجود باشد

لذا حالت یک محتمل تر و منطقی تر است. ولی به هر حال چیزی که در این پایگاه داده مطرح است این است که حذف missingValuesCols تاثیری بر نتایج ما در مقایسه با حالت دیفالت نگذاشته است.

## حالت پنجم (تغییر upperBoundMinSupport)

این تغییر حالت در پارامتر فوق و نسبت به حالت یک (نه حالت دیفالت) انجام گرفته است. این پارامتر را بر روی 0.2 قرار داده‌ایم و نتیجه به صورت زیر است.

```
Associator output
=====
Minimum support: 0.25 (3 instances)
Minimum metric <confidence>: 0.3
Number of cycles performed: 15

Generated sets of large itemsets:

Size of set of large itemsets L(1): 12

Size of set of large itemsets L(2): 26

Size of set of large itemsets L(3): 4

Best rules found:

1. outlook=sunny play=no 3 ==> humidity=high 3 <conf:(1)> lift:(2) lev:(0.11) [1] conv:(1.5)
2. outlook=sunny humidity=high 3 ==> play=no 3 <conf:(1)> lift:(2.8) lev:(0.14) [1] conv:(1.93)
3. outlook=rainy play=yes 3 ==> windy=FALSE 3 <conf:(1)> lift:(1.75) lev:(0.09) [1] conv:(1.29)
4. outlook=rainy windy=FALSE 3 ==> play=yes 3 <conf:(1)> lift:(1.56) lev:(0.08) [1] conv:(1.07)
5. temperature=cool play=yes 3 ==> humidity=normal 3 <conf:(1)> lift:(2) lev:(0.11) [1] conv:(1.5)
6. temperature=hot 4 ==> humidity=high 3 <conf:(0.75)> lift:(1.5) lev:(0.07) [1] conv:(1)
7. temperature=hot 4 ==> windy=FALSE 3 <conf:(0.75)> lift:(1.31) lev:(0.05) [0] conv:(0.86)
8. temperature=cool 4 ==> play=yes 3 <conf:(0.75)> lift:(1.17) lev:(0.03) [0] conv:(0.71)
9. humidity=high play=no 4 ==> outlook=sunny 3 <conf:(0.75)> lift:(2.1) lev:(0.11) [1] conv:(1.29)
10. temperature=cool humidity=normal 4 ==> play=yes 3 <conf:(0.75)> lift:(1.17) lev:(0.03) [0] conv:(0.71)
```

همانطور که مشاهده می‌کنید تعداد سیکل‌های پردازش یافته از 14 به 15 افزایش یافته است و همچنین minsuppoer نیز به مقدار 0.05 کاهش داشته است. این طبیعتاً تاثیر خود را در سایز آیت‌ست‌ها می‌گذارد که همینطور هم هست. یعنی سایز L(2) و L(3) نسبت به حالت اول افزایش داشته است.

برای قسمت بعدی که آخرین قسمت سوالات تمرین است و مربوط به تست گرفتن با کد جاوا بود، اینجانب تمامی فایل‌ها به درستی ران گرفته و فایل class آنها را تولید کردم ولی مشکلی که ایجاد شده بود مربوط به فایل weka.rar بود و با این ارور مواجه می‌شدم. (مشاهده کنید)

```
PROBLEMS 125 OUTPUT DEBUG CONSOLE TERMINAL 1: powershell
at java.base/java.lang.ClassLoader.loadClass(ClassLoader.java:521)
... 1 more
PS D:\dars\داده‌های واک\HW1\java\AssociationRulesMining> javac -classpath "C:\Program Files\Weka-3-8-5\weka.jar" .\AssociationRulesMining.java
PS D:\dars\داده‌های واک\HW1\java\AssociationRulesMining> java AssociationRulesMining
Exception in thread "main" java.lang.NoClassDefFoundError: weka/core/converters/ConverterUtils$DataSource
    at AssociationRulesMining.main(AssociationRulesMining.java:23)
Caused by: java.lang.ClassNotFoundException: weka.core.converters.ConverterUtils$DataSource
    at java.base/jdk.internal.loader.BuiltinClassLoader.loadClass(BuiltinClassLoader.java:581)
    at java.base/jdk.internal.loader.ClassLoaders$AppClassLoader.loadClass(ClassLoaders.java:178)
    at java.base/java.lang.ClassLoader.loadClass(ClassLoader.java:521)
    ... 1 more
PS D:\dars\داده‌های واک\HW1\java\AssociationRulesMining>
```

همانطور که مشاهده می‌کنیم به ConverterUtils\$DataSource گیر داده و عملیات ران را به طور انجام انجام نمی‌دهد.

لذا پس از سرچ فراوان و بررسی فایل‌های jar مختلف موجود از weka، سعی نمی‌دونم ارور را تصحیح کنم که بعد از یک روز به رفرنس پایین خوردم.

<https://forums.pentaho.com/threads/187719-Classdef-not-found-error/>

I have no Idea whats going on here as the errors seem not to be coming from my code exactly but from the weka lib. So I assume I did something fundamental wrong.

Any help is appreciated!

Thanks

edit: The same happens when I try to use other classifiers like e.g. RandomTree

Last edited by abzhilb; 06-01-2015 at 02:18 PM.

06-02-2015, 04:52 AM

#2

Mark

Pentaho WEKA Architect

Join Date: Aug 2006  
Posts: 1,741



Do you have the weka.jar file in your classpath when you run your code? And if so, is it a weka.jar from the Weka distribution or have you compiled Weka yourself in order to make the weka.jar file? The exejar target in Weka's ant build.xml script unpacks lib/packageManager.jar into build/classes before creating the weka.jar, packageManager.jar contains the org.pentaho.packageManagement.\* classes.

Cheers,  
Mark.

06-12-2015, 09:24 AM

#3

abzhilb  
Junior Member

Join Date: May 2015  
Posts: 3



yes everything was done correctly - but seems like something unrelated to weka didn't work. I copied my project to my macbook in order to be able to work on it while out of home and for some reason it started to work without any change... Thanks anyway.

همانطور که مشاهده می‌کنید راه ذکر شده در این لینک را هم کنار بقیه موارد امتحان کردم تا به جواب نهایی پرسشگر این سوال رسیدم.

مطابق با آنچه در ویندوز من رخ داده برای ایشان هم رخ داده و هیچ راه‌حلی پیدا نشده. ایشان همین فایل‌ها را بدون تغییر در سیستم‌عامل مک امتحان نموده و بدون هیچ خطایی به پاسخ رسیدند.

لذا به نظر می‌رسد خطای موجود که در این قسمت سوال با آن برخورد من مربوط به من بلکه مربوط به یکی از نقطه ضعف‌های weka است که در بعضی سیستم‌عامل‌ها در خواندن برخی از کلاس‌ها که مربوط به فایل weka.jar است، مشکل ایجاد می‌کند.

لذا تنها راهی که برای پاسخ‌گویی به این سوال برایم باقی مانده است. ذکر مراحل آن است و از پیاده‌سازی آن هر چند زمان زیادی برای تصحیح فایل weka و کلاس آن گذاشتم، معذور شدم.

مراحل حل سوال:

در ابتدا numrules را مطابق آنچه که در سوال گفته در فایل associationRulesMining.java مطابق با آنچه که در سوال گفته است تغییر می‌دهیم.

```
20      int numRules = 10000;
```

سپس minsup را به صورت یک آرایه تعریف می‌کنیم. دلیل این امر آن است که در یک لوپ الگوریتم‌ها را از نظر زمانی مقایسه می‌خواهیم کنیم.

```
double[] minSup = {0.05, 0.06, 0.07, 0.08, 0.09, 0.1, 0.11, 0.12, 0.13, 0.14, 0.15};
```

پارامترها در قسمت زیر تنظیم می‌شوند.

```
for(int j = 0; j < minSup.length; j++){  
    FPGrowth fpgrowth_model=new FPGrowth();  
    fpgrowth_model.setNumRulesToFind(numRules);  
    fpgrowth_model.setLowerBoundMinSupport(minSup[j]);  
    fpgrowth_model.setMinMetric(0.9);  
    fpgrowth_model.buildAssociations(data);
```

همچنین همانطور که از قسمت قبلی به خاطر داریم، minMetric در setMinMetric به همان minconf اشاره می‌کند و لذا این مقدار را به 0.9 تغییر می‌دهیم.

همچنین همانطور که در بالا مشاهده می‌کنید minsup به صورت یک آرایه ظاهر شده است که هر بار یکی از درایه‌های این وکتور برای پردازش اعمال می‌شود.

در نهایت با تعریف یک تایمر که به صورت یک وکتور و به اندازه‌ی minsup است، تایم هر اجرا را اندازه‌گیری می‌کنیم.

```
long[] elapsedTime={0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
```

نقطه‌ی شروع به صورت زیر است.

```
for(int j = 0; j < minSup.length; j++){  
    long start = System.currentTimeMillis();
```

و نقطه‌ی پایان اندازه‌گیری در هر iteration نیز به صورت زیر.

```
}  
long end = System.currentTimeMillis();  
elapsedTime[j] = end - start;  
}
```

نهایتاً برای هر یک از این دو الگوریتم پس از تعریف آنها این تایم‌ها را جداگانه ست کرده و با دستور put در یک فایل excel می‌ریزیم.

سپس با استفاده از ابزار مهندسی متلب فایل اکسل را به صورت زیر خوانده:

```
%% retrieving data from Data.xlsx and storing in A matrix.  
[data, text] = xlsread('data\Data.xlsx');  
A = data;
```

و در ماتریس A قرار می‌دهیم.

و در نهایت نیز با دستور plot در نرم افزار متلب آن را رسم می کنیم.

از اینکه مجبور هستم آخرین قسمت را به صورت ترکیبی از کد و توضیحی ارائه دهم عذر می خواهم. دلیل آن مشکل از سمت weka.jar بر روی سیستم عامل ها از جمله سیستم عامل من است که عرض کردم.

با تشکر - بدیعی