دانشگاه یزد

دانشکده مهندسی کامپیوتر

پایان‌نامه کارشناسی مهندسی کامپیوتر

گرایش نرم افزار

عنوان پایان‌نامه

بدست آوردن وابستگی های کالاهای فروشگاهی و رستورانی

نگارش:

امیر پورمند

استاد راهنما:

دکتر ولی درهمی

شهریور ۱۳۹۹


بسم الله الرحمن الرحیم

تشکر و قدرداني

سپاسگذار کسانی هستم که سراغاز تولد من هستند.از یکی زاده میشوم و از دیگری جاودانه.استادی که سپیدی را بر تخته سیاه زندگیم نگاشت و مادری که تار مویی از او بپای من سیاه نماند.

چکیده

در این پروژه سعی کرده ام تا یک دیتابیس واقعی را مورد تحلیل قرار بدهم والگوهای مکرر آن را از طریق الگوریتم اپریوری با پیاده سازی کامل تشریح کنم.

در این گزارش ابتدا به تعاریف مقدماتی قواعد همبستگی و معیارهایی مانند support و confidenceو liftاکه در الگوریتم یا در تحلیل داده ها از آن ها استفاده میشود میپردازیم.

سپس الگوریتم اپریوری را مرحله به مرحله در پایتون اجرا میکنم و خروجی را تحلیل میکنم. در نهایت خروجی الگوریتم در یک جدول مشخص شده و قواعد استخراج شده از این داده ها داده میشود.

کلید واژه: DataMining, Assosication Rules, Apriori Algorithmاپریوری، قواعد همبستگی، الگور های مکرر

فـهرست مطـالـب

عنوان **صفحه**

مقدمه

پیش‌گفتار

اغلب الگوریتم‌های یادگیری ماشین در داده‌کاوی با داده‌های عددی کار می‌کنند و در پیاده‌سازی و نحوه کار آن‌ها گرایش به ریاضیات محض وجود دارد. اما، «کاوش قواعد وابستگی» association rule mining که از آن با عنوان «کاوش قواعد وابستگی» نیز یاد می‌شود، برای داده‌های دسته‌ای مناسب و محاسبات آن نسبت به بسیاری از دیگر الگوریتم‌ها ساده‌تر است. این روش، یکی از راهکارهای مبتنی بر قواعد (rules)، برای کشف روابط جالب بین متغیرها در پایگاه داده‌های بزرگ محسوب می‌شود. در کاوش قواعد وابستگی، قواعد قوی با استفاده از سنجه جذابیت (interestingness) شناسایی می‌شوند.

یکی از معروف‌ترین مثال‌ها در رابطه با قواعد وابستگی، این عبارت است: «مردهای جوان آمریکایی که بعد از ظهرهای جمعه پوشک بچه تهیه می‌کنند، مستعد خرید آبجو نیز هستند». این عبارت یک مثال مشهور از کاوش قواعد وابستگی عجیب از زندگی روزمره افراد است. چنین اطلاعاتی را می‌توان به عنوان پایه‌هایی برای تصمیم‌سازی درباره فعالیت‌های بازار مانند قیمت‌گذاری تبلیغاتی یا تحلیل سبد خرید اهدف از این الگو و دستورالعمل شناسایی میشوند.

فصل ۲– تعاریف و مقدمات الگوریتم

۲– ۱– قواعد وابستگی از دیدگاه ریاضی

مساله کاوش قواعد وابستگی را می‌توان به صورت ریاضی و چنانکه در ادامه می‌آید دید.

- $I = \{I_1, I_2, I_3, ...\}$ امجموعه‌ای از ویژگی‌ها (خصیصه‌های) دودویی است که به آن‌ها اقلام گفته می‌شود.
- $D = \{t1 , t2, t3 , ...\}$ مجموعه‌ای از تراکنش‌ها است که پایگاه داده نامیده می‌شود.
- هر تراکنش در «D» شامل زیرمجموعه‌ای از اقلام موجود در «I» است.
- قواعد ساده وابستگی چنانکه در ادامه می‌آید هستند. لازم به ذکر است که در قاعده زیر، t1 مقدم و t2 نتیجه (موخر) محسوب می‌شود.
- T1 => T2 (در اینجا، t_i به طور کلی یک مورد مجزا یا مجموعه‌ای از اقلام است)

مثال سوپر مارکت مجموعه داده D (تراکنشی) به صورت زیر است.

transaction ID	milk	bread	butter	beer	diapers
1	1	1	0	0	0
2	0	0	1	0	0
3	0	0	0	1	1
4	1	1	1	0	0
5	0	1	0	0	0

شکل ۲-۱ : داده های نمونه

۲–۲– نکات کلی پیرامون مجموعه داده

- در مجموعه دادهD ، مقدار «۱» نشان‌دهنده وجود یک آیتم در یک تراکنش و «۰» نشانگر عدم وجود آن است.
- مجموعه اقلام این مجموعه داده به صورت{ I = {milk, bread, butter, beer, diapers است.

- قاعده{milk} ⇒ {butter, bread} ، بدین معناست که اگر کره و نان خریداری شوند، مشتری شیر نیز می‌خرد.

۲–۳– استفاده از آستانه‌ها برای روابط

۲–۳–۱– پشتیبان

پشتیبان شاخصی است از اینکه یک مجموعه اقلام (itemset) چند بار در یک مجموعه داده (data set) ظاهر می‌شود. پشتیبانX ، با توجه بهT ، به صورت کسر تراکنش‌های t در مجموعه داده‌ای که شامل مجموعه اقلام X است تعریف می‌شود.

$$Sup(X) = \frac{|\{t \in T ; X \subset t\}|}{|T|}$$

در مجموعه داده معرفی شده در بالا، مجموعه اقلام {beer, diapers} = X دارای پشتیبان 0.2 است. زیرا در ٪۲۰ تراکنش‌ها به وقوع پیوسته است (یک تراکنش از کل پنج تراکنش). آرگومانsupp() ، مجموعه‌ای از پیش شرط‌ها است و بدین ترتیب با رشد کردن محدودکننده‌تر می‌شود (به جای آنکه جامع‌تر شود).

البته پشتیبان به گونه دیگری نیز تعریف میشود که بدین شرح است:

$$Support(XY) = \frac{Support\ Count\ of\ XY}{Total\ Number\ Of\ transactions\ in\ D}$$

۲–۳–۲– اطمینان(Confidence)

اطمینان شاخصی است از اینکه یک قاعده چند بار درست (True) بوده. مقدار اطمینان یک قاعده(X→Y) ، با توجه به مجموعه تراکنشT ، عبارت است از کسری از تراکنش‌های شامل X که شامل Y نیز هستند. اطمینان به صورت زیر تعریف می‌شود.

$$Conf(X \rightarrow Y) = \frac{Sup(X \cup Y)}{Sup(X)}$$

برای مثال، قاعده{milk} → {butter, bread} ، در پایگاه داده، دارای اطمینان ۱ = ۰,۲/۰,۲ است. این یعنی برای ۱۰۰٪ تراکنش‌هایی که شامل کره و نان بوده‌اند این قاعده صحت داشته (۱۰۰٪ دفعاتی که مشتری نان و کره خریده، شیر نیز تهیه کرده است). لازم به ذکر است که sup(XUY)، پشتیبان اتحاد اقلام در X و Y است.

البته معیار اطمینان در برخی منابع نیز بصورت زیر تعریف شده است (فقط نوتیشن و شکل نوشتار فرق دارد و در واقع فرقی با فرمول بالا ندارد):

$$Conf(X|Y) = \frac{Sup(XY)}{Sup(X)}$$

۲-۳-۳- بالابری (Lift)

بالابریِ یک قاعده به صورت زیر تعریف می‌شود.

$$Lift(X \rightarrow Y) = \frac{Sup(X \cup Y)}{Sup(X) * Sup(Y)}$$

اگر X و Y مستقل باشند، نسبت پشتیبان مشاهده شده مورد نظر است. برای مثال، قاعده { milk, bread -> butter } دارای بالابری ۱,۲۵ = ۰,۴ × ۰,۴ / ۰,۲ است. اگر قاعده دارای بالابری ۱ باشد، به طور ضمنی دلالت بر این دارد که احتمال پیش‌آمد مقدم و نتیجه از یکدیگر مستقل هستند. هنگامی که دو رویداد مستقل از هم هستند، هیچ قاعده‌ای را نمی‌توان با این دو رویداد ایجاد کرد.

اگر بالابری بزرگ‌تر از یک باشد، به داده‌کاو اجازه می‌دهد که بداند درجه کدام دو پیش‌آمد وابسته به دیگری است و این قواعد را به طور بالقوه برای پیش‌بینی نتیجه در مجموعه داده آینده مورد استفاده قرار می‌دهد. اگر بالابری کمتر از یک باشد، به داده‌کاو اجازه می‌دهد که بداند اقلام جایگزین یکدیگر هستند. این بدین معناست که حضور یک آیتم تاثیر منفی بر حضور آیتم دیگر دارد و بالعکس.

ارزش بالابری از آنجا ناشی می‌شود که هم اطمینان یک قاعده و هم کلیت مجموعه داده را در بر دارد. برای تشریح بیشتر مساله می‌توان گفت، در داده‌کاوی و یادگیری قواعد وابستگی، بالابری سنجه‌ای است که کارایی مدل هدف (قاعده وابستگی) را در پیش‌بینی یا دسته‌بندی اقلام در راستای داشتن یک پاسخ بهبود یافته (با توجه به جمعیت کل)، که در تقابل با یک مدل هدفمند انتخاب تصادفی اندازه‌گیری شده می‌سنجد.

یک مدل هدفمند در صورتی که پاسخ (واکنش) به هدف برای جمعیت کل بهتر از میانگین باشد، عملکرد خوبی خواهد داشت. به بیان ساده، بالابری، نسبت مقادیر پاسخ هدف و میانگین پاسخ در شرایطی است که اولی بر دومی تقسیم شود. به عنوان مثالی دیگر، فرض می‌شود که یک جمعیت نرخ پاسخ میانگین ۵٪ دارد، اما یک مدل (یا قاعده) خاص بخشی با نرخ پاسخ ۲۰٪ را شناسایی کرده است. بنابراین، مقدار lift برابر با ۴,۰ (۲۰٪/۵٪) خواهد بود.

معمولا، مدل‌ساز در تلاش برای تقسیم جمعیت به چندک‌ها و رتبه‌بندی آن‌ها بر اساس بالابری است. سازمان‌ها می‌توانند هر چندک را در نظر بگیرند و با وزن‌دهی به نرخ پاسخ پیش‌بینی شده (و مزایای مالی مربوط به آن) در مقابل هزینه، تصمیم بگیرند که آیا در آن چندک بازاریابی کنند یا خیر. اگر با بالابری به مثابه «دقت (precision) « مواجهه شود که در «بازیابی اطلاعات (information retrieval) «کسر مثبت‌هایی است که مثبت صحیح (True Positive)هستند، می‌توان گفت بالابری مشابه با سنجه دقت متوسط (average precision) است.

منحنی بالابری را می‌توان به عنوان تغییری در منحنی مشخصه عملکرد سیستم (Receiver operating characteristic)محسوب کرد، که در اقتصادسنجی نیز با عنوان منحنی لورنز (Lorenz) یا منحنی توان (power curve)شناخته شده است. برای مثال دیگری از بالابری، مجموعه داده زیر مفروض است.

Antecedent	Consequent
A	0
A	0
A	1
A	0
B	1
B	0
B	1

شکل ۲-۲: مثال قواعد بالابری

در این مجموعه داده، مقدم متغیر ورودی تحت کنترل داده‌کاو و نتیجه (موخر) متغیری است که داده‌کاو سعی در پیش‌بینی آن دارد. مسائل جهان واقعی داده‌کاوی، مقدم‌های پیچیده‌تری دارند ولی معمولا تمرکز بر نتایج تک مقداری است. اغلب الگوریتم‌های کاوش، قواعد زیر را دنبال می‌کنند (مدل‌های هدفمند).

- قاعده ۱: A دلالت دارد بر ۰
- قاعده ۲: B دلالت دارد بر ۱

زیرا این موارد متداول‌ترین الگوهای یافت شده در داده‌ها هستند. انجام یک بررسی ساده روی جدول بالا، این قواعد را واضح‌تر می‌سازد. پشتیبان برای قاعده ۱ برابر با ۳,۷ است، زیرا این تعداد اقلام موجود در مجموعه داده است که در آن‌ها مقدم A و نتیجه ۰ است. پشتیبان برای قاعده ۲ برابر با ۲,۷ است زیرا دو تا از هفت رکورد به مقدم‌های B و نتایج ۱ مربوط هستند. پشتیبان‌ها را می‌توان به صورت زیر نوشت.

$$Sup(A \to 0) = P(A \wedge 0) = P(A)P(0|A) = P(0)P(A|0)$$
$$Sup(B \to 1) = P(B \wedge 1) = P(B)P(1|B) = P(1)P(B|1)$$

اطمینان برای قاعده ۱ برابر با ۴/۳ است زیرا سه تا از چهار رکورد که مقدم A دارند به نتیجه ۰ می‌رسند. اطمینان برای قاعده ۲ برابر با ۳/۲ است، زیرا دو تا از سه رکوردی که مقدم B دارند، به نتیجه ۱ می‌رسند. اطمینان را می‌توان به صورت زیر نوشت.

$$conf(A \to 0) = P(0|A)$$
$$conf(B \to 1) = P(1|B)$$

بالابری را می‌توان با تقسیم اطمینان بر احتمال غیر شرطی نتیجه‌ها یا تقسیم پشتیبان بر احتمال مقدم، ضرب در احتمال نتیجه به دست آورد.

بالابری برای قاعده ۱ برابر است با (۴/۳)/(۷/۴) = (۳*۷)/(۴ * ۴) = ۱۶/۲۱ ≈ ۱,۳۱

بالابری برای قاعده ۲ برابر است با (۳/۲)/(۷/۳) = (۲*۷)/(۳ * ۳) = ۹/۱۴ ≈ ۱,۵۶

$$lift(A \to 0) = \frac{P(0|A)}{P(0)} = \frac{P(A \wedge 0)}{P(A)P(0)}$$
$$lift(B \to 1) = \frac{P(1|B)}{P(1)} = \frac{P(B \wedge 1)}{P(B)P(1)}$$

اگر برخی از قواعد دارای بالابری ۱ باشند، به طور ضمنی دلالت بر آن دارند که احتمال وقوع مقدم و احتمال وقوع نتیجه از یکدیگر مستقل است. هنگامی که دو رویداد مستقل از هم هستند، هیچ قاعده‌ای نمی‌تواند در برگیرنده آن دو باشد. اگر بالابری بزرگ‌تر از یک باشد($lift > 1$) ، مشابه آنچه برای قواعد ۱ و ۲ وجود دارد، این امکان فراهم می‌شود که داده‌کاو بداند درجه کدام دو پیش‌آمد به یکدیگر وابسته است، و این قواعد را برای پیش‌بینی توالی در مجموعه داده‌های آتی کارآمد می‌سازد.

شایان توجه است که قاعده ۱ دارای اطمینان بالاتری است زیرا بالابری کمتری دارد. به نظر می‌رسد که قاعده ۱ به دلیل داشتن اطمینان بالاتر (صحت بالاتری دارد چون پشتیبانی بهتری نیز دارد)، ارزشمندتر است. اما صحت قاعده مستقل از داده گمراه کننده است. ارزش بالابری (lift) به آن است که دارای اطمینان برای قاعده و به طور کل مجموعه داده باشد.

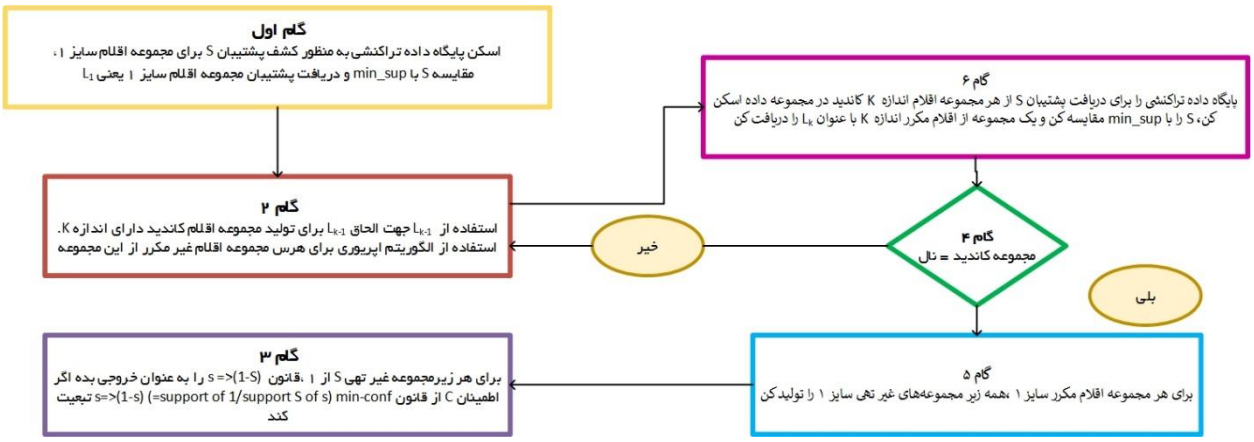
۲-۴- الگوریتم اپریوری(Apriori)

الگوریتم اپریوری(Apriori) ، روشی قابل اعمال روی رکوردهای پایگاه داده و به ویژه پایگاه داده تراکنشی یا رکوردهای حاوی تعداد مشخصی فیلد یا آیتم است. اپریوری یکی از الگوریتم‌های دارای رویکرد «پایین به بالا» است که به تدریج رکوردهای پیچیده را با یکدیگر مقایسه می‌کنند. این الگوریتم یکی از روش‌های کارآمد برای حل مسائل پیچیده کنونی موجود در داده‌کاوی و یادگیری ماشین است.

اساسا، الگوریتم اپریوری بخش‌هایی از یک پایگاه داده بزرگ‌تر را دریافت کرده و به آن‌ها «امتیازدهی» کرده و یا آن بخش‌ها را با دیگر مجموعه‌ها به شیوه مرتب شده‌ای مقایسه می‌کند. از نتایج خروجی، برای تولید مجموعه‌هایی استفاده می‌شود که مکررا در پایگاه داده اصلی به وقوع پیوسته‌اند.

برای کسب درک بهتر از الگوریتم می‌توان برخی کاربردهای آن مانند «تحلیل سبد خرید» را مورد بررسی قرار داد. در این کاربرد، داده‌کاو به دنبال کشف آن است که کدام اقلام با یکدیگر (در یک سبد خرید) خریداری شده‌اند. در دیگر مثالی که می‌توان پیرامون الگوهای مکرر زد، ابزارهای تحلیل مالی هستند که با بهره‌گیری از الگوریتم اپریوری چگونگی داغ شدن سهام‌های گوناگون با یکدیگر را نمایش می‌دهند. فلوچارت الگوریتم اپریوری (Apriori) در ادامه آورده شده است.

این روش، ممکن است به طور پیوسته با دیگر الگوریتم‌ها استفاده شود تا به طور موثری داده‌ها را مرتب‌سازی و با یکدیگر مقایسه کند. اصل اپریوری می‌تواند تعداد اقلامی که نیاز به بررسی آن‌ها است را کاهش دهد. این روش بیان می‌کند که اگر یک مجموعه اقلام فاقد تکرار است، پس همه زیرمجموعه‌های آن نیز نادر هستند. این امر بدین معناست که اگر {آبجو} فاقد تکرار بود، می‌توان انتظار داشت که {آبجو، پیتزا} هم به همان میزان و یا حتی بیشتر، نادر باشند. بنابراین، برای یکی کردن لیست مجموعه اقلام محبوب، نیازی به در نظر گرفتن {آبجو، پیتزا} و یا هیچ یک از دیگر مجموعه اقلام حاوی آبجو، نخواهد بود.



شکل ۲-۲: گام های الگوریتم اپریوری

این روش، ممکن است به طور پیوسته با دیگر الگوریتم‌ها استفاده شود تا به طور موثری داده‌ها را مرتب‌سازی و با یکدیگر مقایسه کند. اصل اپریوری می‌تواند تعداد اقلامی که نیاز به بررسی آن‌ها است را کاهش دهد. این روش بیان می‌کند که اگر یک مجموعه اقلام فاقد تکرار است، پس همه زیرمجموعه‌های آن نیز نادر هستند. این امر بدین معناست که اگر {آبجو} فاقد تکرار بود، می‌توان انتظار داشت که {آبجو، پیتزا} هم به همان میزان و یا حتی بیشتر، نادر باشند. بنابراین، برای یکی کردن لیست مجموعه اقلام محبوب، نیازی به در نظر گرفتن {آبجو، پیتزا} و یا هیچ یک از دیگر مجموعه اقلام حاوی آبجو، نخواهد بود.

۲-۴-۱- یافتن مجموعه اقلام دارای پشتیبان بالا

با استفاده از اصل اپریوری، تعداد مجموعه اقلامی که باید بررسی شوند قابل هرس شدن هستند و لیستی از مجموعه اقلام محبوب (مکرر) با انجام مراحل زیر قابل مشاهده است.

گام ۰ :آغاز با مجموعه اقلام حاوی تنها یک مورد، مانند {سیب} و {هلو}.

گام ۱ :تعیین پشتیبان برای مجموعه اقلام. حفظ کردن مجموعه اقلامی که به حداقل آستانه پشتیبان می‌رسند و حذف مواردی که مطابقت ندارند.

گام ۲ :استفاده از مجموعه اقلام گام ۱، ساخت همه پیکربندی‌های مجموعه اقلام ممکن.

گام ۳ :تکرار گام‌های ۱ و ۲ تا هنگامی که مجموعه اقلام جدیدی وجود نداشته باشد.

۲-۴-۲- یافتن قواعد با قابلیت اطمینان یا بالابری زیاد

پیش از این بیان شد که چگونه می‌توان از الگوریتم اپریوری برای شناسایی مجموعه اقلام دارای پشتیبانی بالا استفاده کرد. قاعده مشابهی برای شناسایی انجمن‌های اقلام با اطمینان یا بالابری زیاد قابل استفاده است. پیدا کردن قواعد با اطمینان یا بالابری زیاد به لحاظ محاسباتی کم هزینه است. هنگامی که اقلام با پشتیبان بالا شناسایی شدند، مقادیر بالابری با استفاده از مقادیر پشتیبان محاسبه می‌شود. برای مثال برای یافتن قواعد با اطمینان بالا، اگر قاعده به صورت زیر باشد

{beer, chips -> apple}

دارای اطمینان پایینی است. دیگر قواعد با اقلام تشکیل دهنده و سیب در سمت راست جهت نما نیز دارای اطمینان پایینی هستند. به ویژه قواعد

{beer -> apple, chips}

{chips -> apple, beer}

اطمینان بسیار پایینی دارند. مانند گذشته، قواعد اقلام کاندید با اطمینان یا بالابری پایین را می‌توان با استفاده از الگوریتم اپریوری هرس کرد، بنابراین قواعد کاندید کمتری برای بررسی باقی می‌ماند.

۲-۴-۳- محدودیت‌ها

هزینه محاسباتی بالا :روش اپریوری به لحاظ محاسباتی بسیار پر هزینه است. حتی اگر الگوریتم اپریوری تعداد اقلام کاندید برای بررسی را کاهش دهد، در صورتی که موجودی فروشگاه زیاد یا آستانه پشتیبان کم باشد میزان باقیمانده همچنان عدد بزرگی خواهد بود. یک راهکار جایگزین، کاهش تعداد مقایسه‌ها با استفاده از ساختارهای پیشرفته داده مانند جدول‌های هش برای مرتب‌سازی اقلام کاندید به شیوه موثرتر است.

انجمن‌های جعلی :تحلیل صورت کالاهای بزرگ مجموعه اقلام بیشتری را در برمی‌گیرد و آستانه پشتیبان ممکن است برای شناسایی انجمن‌های مشخصی کاهش پیدا کند. اگرچه، کاهش آستانه پشتیبان ممکن است تعداد انجمن‌های جعلی کشف شده را افزایش دهد. برای حصول اطمینان از اینکه انجمن‌های شناسایی شده قابل تعمیم هستند، می‌توان آن‌ها را از مجموعه داده آموزش به دست آورد، پیش از آنکه پشتیبان و اطمینان ارزیابی شده برای آن‌ها در یک مجموعه داده جدا قرا گیرد.

فصل ۳– پیاده سازی اپریوری در پایتون

۳–۱– آماده سازی داده ها برای الگوریتم

```
import numpy as np
import pandas as pd
from itertools import combinations
```

شکل ۳–۱: کتابخانه های استفاده شده

با توجه به شکل ۶ در پروژه از کتابخانه های نامپای و پانداز و البته ایترتولز استفاده شده که دوتای اولی از معروف ترین کتابخانه های علم داده در پایتون هستند. کتابخانه سومی هم که itertools نام دارد تمام ترکیب های چندتایی اشیا را درست میکند.

```
import pyodbc
conn = pyodbc.connect('''Driver=SQL Server;
                         SERVER=.\sql2019;
                         DATABASE=Sepand;
                         UserName=sa;Password=123;''')

```

شکل ۳–۲: اتصال به دیتابیس

سپس برای اتصال به دیتابیس از کتابخانه pyodbc استفاده شده است که کوئری مربوطه در شکل بالا نشان داده شده است. در کوئری یک Connection String نیز به چشم میخورد که درایو SqL Server استفاده شده است و از سرور و دیتابیس و یوزر پسورد مناسب برای اتصال استفاده گردیده است.

```
query1 = '''
SELECT factor as SalesOrderID,Name Name,ted as OrderQty
FROM tr INNER JOIN TblKala ON tblkala.idanbar = tr.idanbr
AND tblkala.idkala = tr.idkala
AND type = 203 AND factor>0
'''
store_data = pd.read_sql(query1, conn)
store_data.head()
```

شکل ۳-۳: کوئری دیتابیس

در کوئری بالا داده های نام و تعداد ایتم هر سفارش انتخاب شده اند و سپس کوئری در سمت دیتابیس اجرا میشود . دستور read_sql داده ها را با توجه به کانکشن مربوطه و کوئری نوشته شده از دیتابیس برمیدارد. دستور head ۵ سطر اول دیتاها را برمیگرداند و به کاربر نمایش میدهد

Out[4]:

	SalesOrderID	Name	OrderQty
0	1	گوجه کبابی 100 گرم	2.0
1	1	کباب کوبیده (1سیخ)	32.0
2	1	ظرف	4.0
3	2	جوجه مخصوص	10.0
4	2	کباب کوبیده (1سیخ)	24.0

شکل ۳-۴: خروجی کوئری اجرایی در دیتابیس

با توجه به شکل بالا ستون اول شماره فاکتور، ستون دوم نام کالا و ستون سوم تعداد محصول سفارشی را مشخص میسازد. در اینجا نیز ۵ سطر اول کوئری اجرا شده نشان داده شده اند.

```
store_data['OrderQty'] = np.where(store_data['OrderQty']>0, 1, 0)
store_data
```

شکل ۳-۵: تصحیح تعداد کالاها

در کوئری بالا تعداد خرید اقلام تمام کالا ها به ۱ اصلاح میشود زیرا تعداد خرید در یک سفارش اهمیتی ندارد و چیزی که در واقع اهمیت دارد تعداد سفارش هایی است که این کالا در آنها خریداری شده است.

	SalesOrderID	Name	OrderQty
0	1	گوجه کبابي 100 گرم	1
1	1	کباب کوبيده (1سيخ)	1
2	1	ظرف	1
3	2	جوجه مخصوص	1
4	2	کباب کوبيده (1سيخ)	1
...	...	...	...
8651	2430	خوراك جوجه چيني	1
8652	2430	فيله کباب	1
8653	2430	فيله کباب ترش	1
8654	2431	جوجه شانديزي	1
8655	2431	خوراك جوجه چيني	1

8656 rows × 3 columns

شکل ۳-۶: خروجی نهایی کالاها

همان طور که در شکل بالا مشخص است تعداد کل کالاهای خریداری شده در کل فاکتور ها ۸۶۵۶ میباشد و تعداد فاکتورها نیز ۲۴۳۱ میباشد و این بدین معنی است که در هر فاکتور تقریبا ۴ تراکنش داریم.

```
store=store_data.groupby(['SalesOrderID', 'Name'])['OrderQty']
                .sum().unstack().reset_index()
                .fillna(0).set_index('SalesOrderID')
df = store
store.head()
```

شکل ۳-۷: تبدیل جدول به فرم مناسب الگوریتم اپریوری

در کد بالا با یک دستور groupby و sum در واقع یک pivot اجرا میشود که در دنیای دیتابیس بدین معنی است که جای سطر و ستون ها به نوعی عوض میشوند. برای فهم بهتر لطفا خروجی الگوریتم را مشاهده فرمایید.

Name	موکا	چیزبرگر	آب انار	آب معدني 1.5 ليتري	آب معدني 300 سي سي	آب هويج	آب پرتقال	آبليمو	اب معدني 1/5ليتري	استانبولي	...	چلو پاچين	چلو گردن مخصوص راه وما	چلوخورشت قيمه	چلوماهي قزل	چيکن سالاد	ژله	ژله تک نفره	گردن بره	گوجه کبابي 100 گرم	گوشت چرخي
SalesOrderID																					
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

5 rows × 234 columns

شکل ۳-۸: خروجی آماده برای اجرای الگوریتم اپریوری

در شکل بالا خروجی مناسب برای الگوریتم اپریوری بدست آمده است که همان طور که مشخص است یک جدول با ۲۳۴ ستون است که هر ستون معادل یک قلم کالا است و هر سلول وجود یا عدم وجود هر کالا را در هر فاکتور نشان میدهد. بدیهی است که مقدار ۰ به معنای عدم وجود و مقدار ۱ به معنای وجود داشتن است.

۳-۲- فازهای اجرای الگوریتم

الگوریتم در ۳ مرحله یا فاز انجام میشود .خلاصه این که در فاز اول تاپل هایی که ساپورت تعیین شده را شامل میشوند انتخاب خواهند شد و در فاز دوم این تاپل ها که صرفا اندیس های آیتم ها هستند. به خود نام آیتم ها تبدیل خواهند شد و در فاز سوم بقیه اعداد مثل اطمینان و بالابری نیز محاسبه میشوند و نهایتا خروجی نمایش داده میشود.

۳-۲-۱- فاز اول الگوریتم

در فاز اول اجرای الگوریتم که در شکل زیر مشاهده میشود. ابتدا مقدار پشتیبان ۰,۰۵ در نظر گرفته شده و سپس در یک مرحله کل ساپورت های اقلام تکی با دستور np.sum محاسبه میشود سپس مرحله اصلی الگوریتم اجرا میشود که در هر مرحله کل ترکیبیات کل آیتم های مورد نظر را محاسبه می کند و آنهایی را انتخاب میکند که ساپورتشان بیشتر از مین ساپورت باشد.

```
min_support = 0.05

print("count of all:", store.shape[0])
X = df.values
out = np.sum(X, axis=0) / store.shape[0]
support = np.array(out).reshape(-1)
array_index = np.arange(X.shape[1])
support_dict = {1: support[support >= min_support]}
itemset_dict = {1: array_index[support >= min_support].reshape(-1, 1)}
max_itemset = 1
rows_count = float(X.shape[0])
all_ones = np.ones((int(rows_count), 1))
while max_itemset and max_itemset < float('inf'):
    next_max_itemset = max_itemset + 1

    combin = generate_new_combinations(itemset_dict[max_itemset])
    combin = np.fromiter(combin, dtype=int)
    combin = combin.reshape(-1, next_max_itemset)
    if combin.size == 0:
        break
    print(f"Processing {combin.size} combinations | Sampling itemset size {next_max_itemset}")
    _bools = np.all(X[:, combin], axis=2)
    support = np.sum(np.array(_bools), axis=0) / rows_count
    _mask = (support >= min_support).reshape(-1)
    if any(_mask):
        itemset_dict[next_max_itemset] = np.array(combin[_mask])
        support_dict[next_max_itemset] = np.array(support[_mask])
        max_itemset = next_max_itemset
    else:
        break
```

شکل ۳-۹: فاز اول اجرای الگوریتم

البته در اینجا از تابع generate_new_combination استفاده شده است که کد آن در زیر نوشته شده است. این کد در واقع تمام ترکیبیات لازم برای الگوریتم اپریوری را در هر مرحله تولید میکند.

```
def generate_new_combinations(old_combinations):
    items_types_in_previous_step = np.unique(old_combinations.flatten())
    for old_combination in old_combinations:
        raw_item = old_combination[-1]
        mask = items_types_in_previous_step > raw_item
        valid_items = items_types_in_previous_step[mask]
        old_tuple = tuple(old_combination)
        for item in valid_items:
            yield from old_tuple
            yield item
```

شکل ۳-۱۰: تولید ترکیبیات لازم برای الگوریتم اپریوری

خروجی فاز اول الگوریتم که در شکل زیر نشان داده شده است که در واقع یک دیکشنری است که یک های آن اقلام تکی با ساپورت مناسب را لیست میکند و دوهای آن اقلام دوتایی را لیست میکند و الی اخر ...

```
count of all: 2352
Processing 380 combinations | Sampling itemset size 2
Processing 57 combinations | Sampling itemset size 3

{1: array([[  4],
       [ 30],
       [ 43],
       [ 45],
       [ 50],
       [ 53],
       [ 55],
       [ 62],
       [ 88],
       [100],
       [115],
       [133],
       [168],
       [169],
       [170],
       [177],
       [201],
       [202],
       [215],
       [221]]), 2: array([[ 55,  88],
       [ 55, 169],
       [ 55, 170],
       [ 55, 202],
       [ 55, 215],
       [ 88, 115],
       [ 88, 169],
       [ 88, 202],
       [ 88, 215],
       [169, 215]]), 3: array([[ 55,  88, 169]])}
```

شکل ۳-۱۱: خروجی فاز اول اجرای الگوریتم

۳-۲-۲- فاز دوم الگوریتم

در فاز دوم اجرای الگوریتم که در شکل زیر نشان داده شده است. به جای ایندکس آیتم ها نام آیتم ها گذاشته میشود و در نهایت اطلاعات وارد یک Panadas DataFrame می شود که مانند یک جدول در یک دیتابیس رابطه ای است.

```
all_res = []
for k in sorted(itemset_dict):
    support_array = pd.Series(support_dict[k])
    itemsets = pd.Series([frozenset(i) for i in itemset_dict[k]],
                         dtype='object')

    res = pd.concat((support_array, itemsets), axis=1)
    all_res.append(res)

apriori_result = pd.concat(all_res)
apriori_result.columns = ['support', 'itemsets']

mapping = {idx: item for idx, item in enumerate(df.columns)}
apriori_result['itemsets'] = apriori_result['itemsets'].apply(lambda x: frozenset([
                                mapping[i] for i in x]))

apriori_result = apriori_result.reset_index(drop=True)
display(apriori_result)
```

شکل ۳-۱۲: فاز دوم اجرای الگوریتم اپریوری

در شکل زیر نیز آیتم های استخراجی این مرحله و ساپورتشان مشاهده میشود

	support	itemsets
0	0.086310	(آب معدني 300 سي سي)
1	0.050170	(جوجه مخصوص)
2	0.067602	(دوغ قوطي)
3	0.080357	(دوغ محلي)
4	0.070578	(زرشك پلو با مرغ)
5	0.059524	(زيتون پرورده)
6	0.251701	(سالاد فصل)
7	0.075255	(سوپ جو با مرغ)
8	0.275935	(ظرف)
9	0.090136	(قليان)

شکل ۳-۱۳: خروجی فاز دوم الگوریتم اپریوری

۳-۲-۳- فاز سوم اجرای الگوریتم

در فاز ۳ اجرای الگوریتم مقادیر اطمینان و بالابری و ساپورت مجموعه های مقدم و موخر در قواعد همبستگی اضافه گردیده است تا خروجی به شکل مناسبی پدید آید. نهایتا خروجی با توجه به معیارهای confidence و lift مرتب می شوند.

```
min_threshold=1
metric_dict = {
    "antecedent support": lambda _, sA, __: sA,
    "consequent support": lambda _, __, sC: sC,
    "support": lambda sAC, _, __: sAC,
    "confidence": lambda sAC, sA, _: sAC/sA,
    "lift": lambda sAC, sA, sC: metric_dict["confidence"](sAC, sA, sC)/sC
    }
columns_ordered = ["antecedent support", "consequent support",
                   "support",
                   "confidence", "lift"]
keys = apriori_result['itemsets'].values
values = apriori_result['support'].values
frozenset_vect = np.vectorize(lambda x: frozenset(x))
frequent_items_dict = dict(zip(frozenset_vect(keys), values))
rule_antecedents = []
rule_consequents = []
rule_supports = []
```

شکل۳-۱۴: فاز سوم اجرای الگوریتم اپریوری مرحله اول

همان طور که در شکل مشخص است توابع ساپورت قبلی و ساپورت بعدی و ساپورت کل و اطمینان و لیفت به صورت مناسب تعریف شده است و یک آرایه نیز به صورت مرتب برای نگهداری اسم توابع برای فراخوانی وجود دارد که در مرحله دوم فازسوم الگوریتم استفاده میشود.

از تابع zip نیز برای map کردن بین مجموعه آیتم ها و ساپورت های متناظرشان استفاده میشود. سپس از داده ساختار dict استفاده میشود تا آیتم های غالب یا همان frequent itemset ها بدست بیایند.

```
for key in frequent_items_dict.keys():
    sAC = frequent_items_dict[key]
    # پیدا کردن تمام ترکیبیات ممکن
    for idx in range(len(key)-1, 0, -1):
        # پیدا قبلی و بعدی
        for c in combinations(key, r=idx):
            antecedent = frozenset(c)

            # key and antecedent are set
            consequent = key - antecedent

            sA = frequent_items_dict[antecedent]
            sC = frequent_items_dict[consequent]

            score = metric_dict['lift'](sAC, sA, sC)
            if score >= min_threshold:
                rule_antecedents.append(antecedent)
                rule_consequents.append(consequent)
                rule_supports.append([sAC, sA, sC])

rule_supports = np.array(rule_supports).T.astype(float)
df_res = pd.DataFrame(
data=list(zip(rule_antecedents, rule_consequents)),
columns=["antecedents", "consequents"])
sAC = rule_supports[0]
sA = rule_supports[1]
sC = rule_supports[2]
for m in columns_ordered:
    df_res[m] = metric_dict[m](sAC, sA, sC)
df_res = df_res.sort_values(['confidence', 'lift'],
                            ascending =[False, False])
display(df_res)
```

شکل ۳-۱۵: فاز سوم اجرای الگوریتم مرحله دوم

در شکل بالا در حلقه اول در واقع قواعد از دل آیتم های چند تایی بیرون میآیند و سپس معیارهای مورد نیاز هر الگوریتم محاسبه میشود و سپس این آیتم ها در داخل یک دیتافریم قرار داده میشوند. در نهایت نیز بر اساس اطمینان و لیفت محاسبه میشوند.

	antecedents	consequents	antecedent support percent	antecedent support count	consequent support percent	consequent support count	support percent	support count	confidence	lift
0	(نوشابه 300 سی سی)	(سالاد فصل)	0.143282	337.0	0.251701	592.0	0.093537	220.0	0.652819	2.593632
4	(چلو جوجه کباب بدون استخوان)	(سالاد فصل)	0.120748	284.0	0.251701	592.0	0.062925	148.0	0.521127	2.070423
7	(چلو کباب کوبیده)	(سالاد فصل)	0.160714	378.0	0.251701	592.0	0.076105	179.0	0.473545	1.881381
1	(سالاد فصل)	(نوشابه 300 سی سی)	0.251701	592.0	0.143282	337.0	0.093537	220.0	0.371622	2.593632
2	(نوشابه قوطی)	(سالاد فصل)	0.205357	483.0	0.251701	592.0	0.074405	175.0	0.362319	1.439483
8	(نوشابه 300 سی سی)	(چلو کباب کوبیده)	0.143282	337.0	0.160714	378.0	0.051020	120.0	0.356083	2.215628
9	(چلو کباب کوبیده)	(نوشابه 300 سی سی)	0.160714	378.0	0.143282	337.0	0.051020	120.0	0.317460	2.215628
6	(سالاد فصل)	(چلو کباب کوبیده)	0.251701	592.0	0.160714	378.0	0.076105	179.0	0.302365	1.881381
3	(سالاد فصل)	(نوشابه قوطی)	0.251701	592.0	0.205357	483.0	0.074405	175.0	0.295608	1.439483
5	(سالاد فصل)	(چلو جوجه کباب بدون استخوان)	0.251701	592.0	0.120748	284.0	0.062925	148.0	0.250000	2.070423

شکل ۳-۱۶: خروجی فاز سوم الگوریتم اپریوری

۳-۲-۴- تحلیل نتایج بدست آمده

ابتدای امر معنای کلمات ارائه شده در جدول را بیان میکنم: کلمه antecedents به معنای مقدم است و کلمه consequents به معنای موخر یا تالی قاعده همبستگی است.

ستون سوم درصد ساپورت مقدم رابطه را در کل مجموعه بیان میکند و ستون پنجم نیز درصد ساپورت تالی را در دیتاها بیان می کند. عدد support count percent نیز درصد ساپورت هر دو آیتم باهم را در نظر میگیرد.

البته در اینجا ۳ عدد دیگر نیز بیان شده اند که به جای درصد عدد واقعی ساپورت ها را نشان میدهند.

دو معیار اطمینان و بالابری نیز قبلا بصورت جامع معرفی شده اند.

در این تحلیل قاعده اول بدست آمده تحلیل و بررسی می شود.

ابتدا به خود قاعده نگاهی بیندازیم که میگوید:

« کسانی که نوشابه ۳۰۰ سی سی سفارش میدهند سالاد فصل هم میخورند.»

در ۳۳۷ فاکتور نوشابه ۳۰۰ سی سی سفارش داده شده است و از این تعداد ۲۲۰ قلم علاوه بر نوشایه سالاد هم سفارش داده اند. همان طور که میدانیم از تقسیم تعداد ساپورت اولی بر ساپورت کل اطمینان قاعده بدست می آید که به طور خلاصه بیان میکند با احتمال ۶۵ درصد کسانی که نوشابه سفارش میدهند سالاد هم میخورند.

مقدار لیفت قاعده نیز برای بررسی همبستگی مهم است که لیفت این قاعده ۳ بدست امده که به معنی این است که وابستگی خوبی بین این ۲ آیتم وجود دارد.

اگر لیفت این دو قاعده ۱ بود به معنای عدم وابستگی مقادیر به یکدیگر است و اگر لیفت بین ۰ تا ۱ باشد رابطه معکوس را نشان میدهد که بدین معنی است که با خرید یکی احتمال خرید دیگری نه تنها افزایش نمییابد بلکه کاهش می یابد.

در ضمن با یک نگاه کلی به جدول روابط بدست آمده میتوان نتیجه گرفت افرادی که یکی از مخلفات غذا را انتخاب میکنند با احتمالی بین ۳۰ تا ۶۵ درصد یک نوع دیگر از مخلفات را نیز سفارش میدهند که نتیجه درخور توجهی است. البته این نتایج روی داده های با تعداد کوچک بدست آمده که اگر تعداد نتایج بیشتر باشد نتایج بهتری بدست خواهد آمد.

بازنویسی قواعد شکل ۳-۱۶ به صورت متنی نیز به شکل زیر است:

- کسانی که نوشابه میخورند با احتمال ۶۵ درصد سالاد فصل نیز مصرف میکنند.
- چلوجوجه کباب بدون استخوان در ۵۲ درصد موارد همراه با سالاد فصل مصرف میشود.
- چلو کباب کوبیده در ۴۷ درصد موارد همراه با سالاد فصل خورده میشود.
- کسانی که سالاد فصل میخورند با احتمال ۳۶ درصد نوشابه نیز مصرف میکنند.
- کسانی که نوشابه سفارش دهند با احتمال ۳۶ درصد کوبیده نیز سفارش میدهند.
- کسانی که کوبیده سفارش دهند با احتمال ۳۱ درصد نوشابه نیز سفارش میدهند.
- کسانی که سالاد فصل سفارش دهند با احتمال ۳۰ درصد چلوکباب کوبیده نیز سفارش میدهند.
- کسانی که سالاد فصل سفارش دهند با احتمال ۲۹ درصد نوشابه قوطی نیز سفارش میدهند.
- کسانی که سالاد فصل سفارش دهند با احتمال ۲۵ درصد چلو جوجه کباب بدون استخوان نیز سفارش میدهند.

لازم به توجه است که بعضی قواعد دوطرفه هستند مثل سالاد فصل و نوشابه قوطی و اگر دقت کنیم میبینم که لیفت آنها نیز تقریبا ۱ است.