Nefisa Hassen

MSDA 212 Task 3

**Part I: Research Question**

A. Describe the purpose of your data mining report by doing the following:

1. Propose one question relevant to a real-world organizational situation that you will answer using market basket analysis.

What common prescription combinations are frequently given together? and how can these insights improve prescription practices?

2. Define one goal of the data analysis. Ensure your goal is reasonable within the scope of the selected scenario and is represented in the available data.

The goal of the data analysis is to identify the most frequent combinations of prescriptions given to patients. hospitals can use this information to optimize prescription practices, ensuring that commonly co-prescribed medications are managed effectively to enhance patient care and streamline treatment protocols.

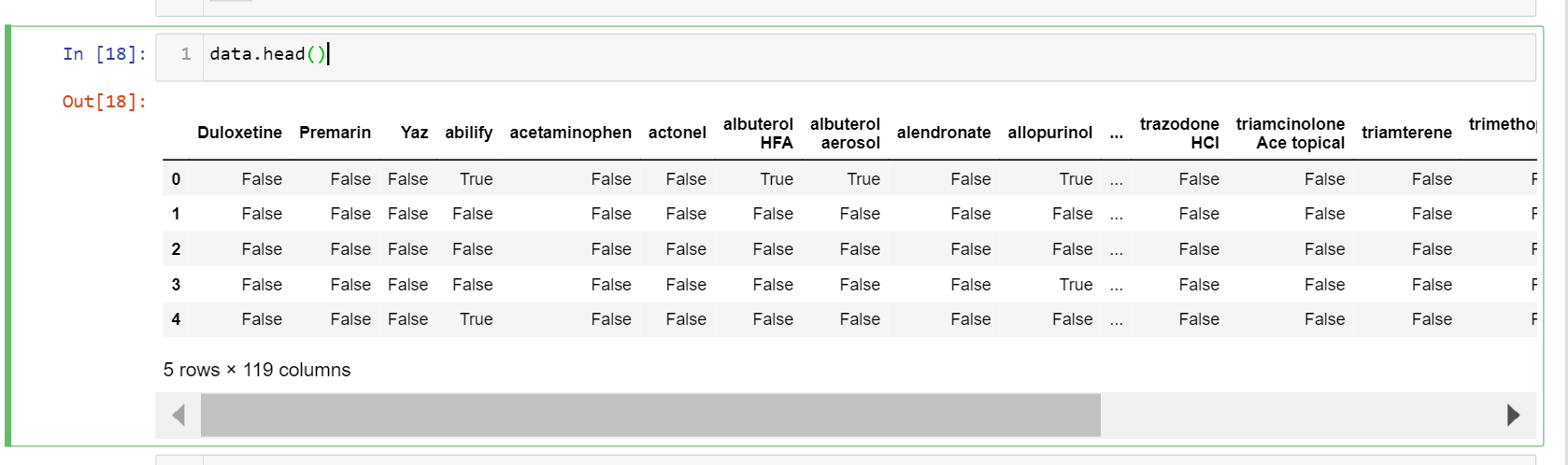
**Part II: Market Basket Justification**

B. Explain the reasons for using market basket analysis by doing the following:

1. Explain how market basket analyzes the selected data set. Include expected outcomes.

Market Basket Analysis (MBA) will examine historical prescription data to identify combinations of medications frequently prescribed together. By analyzing this dataset, which includes various prescriptions given to patients, MBA will uncover all possible prescription pairings that occur together. The analysis will then generate "association rules" to describe the relationships between these medications, highlighting how often certain prescriptions are co-prescribed and how one might suggest another. Key metrics such as support, confidence, and lift will be calculated to assess the strength and significance of these associations. The analysis is expected to reveal the most common prescription combinations among patients.

2. Provide one example of transactions in the data set.

Below is the screenshot showing transactions of the dataset. 

3. Summarize one assumption of market basket analysis.

One assumption about market basket analysis is transaction independence: Each dataset entry is assumed to be an independent transaction, which is essential for market basket analysis. This independence ensures accurate identification of item associations within individual transactions. If transactions are not independent, the reliability of the generated association rules could be compromised.

Part III: Data Preparation and Analysis

C. Prepare and perform market basket analysis by doing the following:

1. Transform the data set to make it suitable for market basket analysis. Include a copy of the cleaned data set.

**import** pandas **as** pd

2

**from** pandas **import** DataFrame

3

**import** numpy **as** np

4

**from** mlxtend.preprocessing **import** TransactionEncoder

5

**import** warnings

6

warnings.filterwarnings('ignore')

In [3]:

1

data **=** pd.read\_csv('medical\_market\_basket.csv')

2

data.head()

Out[3]:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Presc01** | **Presc02** | **Presc03** | **Presc04** | **Presc05** | **Presc06** | **Presc07** | **Presc08** | **Presc09** | **Presc10** | **Presc11** | **Presc12** | **Presc13** | **Presc14** | **Presc15** | **Presc16** | **Presc17** | **Presc18** | **Presc19** | **Presc20** |
| **0** | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| **1** | amlodipine | albuterol aerosol | allopurinol | pantoprazole | lorazepam | omeprazole | mometasone | fluconozole | gabapentin | pravastatin | cialis | losartan | metoprolol succinate XL | sulfamethoxazole | abilify | spironolactone | albuterol HFA | levofloxacin | promethazine | glipizide |
| **2** | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| **3** | citalopram | benicar | amphetamine salt combo xr | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| **4** | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |

In [4]:

1

data.shape

Out[4]:

(15002, 20)

In [5]:

1

data **=** data[data['Presc01'].notna()]

In [6]:

1

data.shape

Out[6]:

(7501, 20)

In [7]:

1

data.head()

Out[7]:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Presc01** | **Presc02** | **Presc03** | **Presc04** | **Presc05** | **Presc06** | **Presc07** | **Presc08** | **Presc09** | **Presc10** | **Presc11** | **Presc12** | **Presc13** | **Presc14** | **Presc15** | **Presc16** | **Presc17** | **Presc18** | **Presc19** | **Presc20** |
| **1** | amlodipine | albuterol aerosol | allopurinol | pantoprazole | lorazepam | omeprazole | mometasone | fluconozole | gabapentin | pravastatin | cialis | losartan | metoprolol succinate XL | sulfamethoxazole | abilify | spironolactone | albuterol HFA | levofloxacin | promethazine | glipizide |
| **3** | citalopram | benicar | amphetamine salt combo xr | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| **5** | enalapril | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| **7** | paroxetine | allopurinol | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| **9** | abilify | atorvastatin | folic acid | naproxen | losartan | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN |

In [8]:

1

*# converting dataframe to list of lists*

2

rows **=** []

3

**for** i **in** range (0,7501):

4

rows.append([str(data.values[i,j])

5

**for** j **in** range (0,20)])

In [9]:

1

*#list fed to TransactionEncoder*

In [10]:

1

DE **=**TransactionEncoder()

2

array **=**DE.fit(rows).transform(rows)

3

4

transcation**=**pd.DataFrame(array,columns **=**DE.columns\_)

In [11]:

1

transcation

Out[11]:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Duloxetine** | **Premarin** | **Yaz** | **abilify** | **acetaminophen** | **actonel** | **albuterol HFA** | **albuterol aerosol** | **alendronate** | **allopurinol** | **...** | **trazodone HCI** | **triamcinolone Ace topical** | **triamterene** | **trimethoprim DS** | **valaciclovir** | **valsartan** | **venlafaxine XR** | **verapamil SR** | **viagra** | **zolpidem** |
| **0** | False | False | False | True | False | False | True | True | False | True | ... | False | False | False | False | False | False | False | False | False | False |
| **1** | False | False | False | False | False | False | False | False | 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 | False | False | False | False | False | False | False | False |
| **3** | False | False | False | False | False | False | False | False | False | True | ... | False | False | False | False | False | False | False | False | False | False |
| **4** | False | False | False | True | False | False | False | False | False | False | ... | False | False | False | False | False | False | False | False | False | False |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **7496** | False | False | False | False | False | False | False | False | False | False | ... | False | False | False | False | False | False | False | False | False | False |
| **7497** | False | False | False | False | False | False | False | False | False | False | ... | False | False | False | False | False | False | False | False | False | False |
| **7498** | False | False | False | False | False | False | False | False | False | False | ... | False | False | False | False | False | False | False | False | False | False |
| **7499** | False | False | False | False | False | False | False | False | False | False | ... | False | False | False | False | False | False | False | False | False | False |
| **7500** | False | False | False | False | False | False | False | False | False | False | ... | False | False | False | False | False | False | False | False | False | False |

7501 rows × 120 columns

In [12]:

1

**for** col **in** transcation.columns: print (col)

Duloxetine  
Premarin  
Yaz  
abilify  
acetaminophen  
actonel  
albuterol HFA  
albuterol aerosol  
alendronate  
allopurinol  
alprazolam  
amitriptyline  
amlodipine  
amoxicillin  
amphetamine  
amphetamine salt combo  
amphetamine salt combo xr  
atenolol  
atorvastatin  
azithromycin  
benazepril  
benicar  
boniva  
bupropion sr  
carisoprodol  
carvedilol  
cefdinir  
celebrex  
celecoxib  
cephalexin  
cialis  
ciprofloxacin  
citalopram  
clavulanate K+  
clonazepam  
clonidine HCI  
clopidogrel  
clotrimazole  
codeine  
crestor  
cyclobenzaprine  
cymbalta  
dextroamphetamine XR  
diazepam  
diclofenac sodium  
doxycycline hyclate  
enalapril  
escitalopram  
esomeprazole  
ezetimibe  
fenofibrate  
fexofenadine  
finasteride  
flovent hfa 110mcg inhaler  
fluconozole  
fluoxetine HCI  
fluticasone  
fluticasone nasal spray  
folic acid  
furosemide  
gabapentin  
glimepiride  
glipizide  
glyburide  
hydrochlorothiazide  
hydrocodone  
hydrocortisone 2.5% cream  
ibuprophen  
isosorbide mononitrate  
lansoprazole  
lantus  
levofloxacin  
levothyroxine sodium  
lisinopril  
lorazepam  
losartan  
lovastatin  
meloxicam  
metformin  
metformin HCI  
methylprednisone  
metoprolol  
metoprolol succinate XL  
metoprolol tartrate  
mometasone  
nan  
naproxen  
omeprazole  
oxycodone  
pantoprazole  
paroxetine  
pioglitazone  
potassium Chloride  
pravastatin  
prednisone  
pregabalin  
promethazine  
quetiapine  
ranitidine  
rosuvastatin  
salmeterol inhaler  
sertraline HCI  
simvastatin  
spironolactone  
sulfamethoxazole  
synthroid  
tamsulosin  
temezepam  
topiramate  
tramadol  
trazodone HCI  
triamcinolone Ace topical  
triamterene  
trimethoprim DS  
valaciclovir  
valsartan  
venlafaxine XR  
verapamil SR  
viagra  
zolpidem

In [13]:

1

*#Remove empty columns*

2

3

cleaned\_df**=**transcation.drop(['nan'],axis **=**1)

4

cleaned\_df.head(7505)

Out[13]:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Duloxetine** | **Premarin** | **Yaz** | **abilify** | **acetaminophen** | **actonel** | **albuterol HFA** | **albuterol aerosol** | **alendronate** | **allopurinol** | **...** | **trazodone HCI** | **triamcinolone Ace topical** | **triamterene** | **trimethoprim DS** | **valaciclovir** | **valsartan** | **venlafaxine XR** | **verapamil SR** | **viagra** | **zolpidem** |
| **0** | False | False | False | True | False | False | True | True | False | True | ... | False | False | False | False | False | False | False | False | False | False |
| **1** | False | False | False | False | False | False | False | False | 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 | False | False | False | False | False | False | False | False |
| **3** | False | False | False | False | False | False | False | False | False | True | ... | False | False | False | False | False | False | False | False | False | False |
| **4** | False | False | False | True | False | False | False | False | False | False | ... | False | False | False | False | False | False | False | False | False | False |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **7496** | False | False | False | False | False | False | False | False | False | False | ... | False | False | False | False | False | False | False | False | False | False |
| **7497** | False | False | False | False | False | False | False | False | False | False | ... | False | False | False | False | False | False | False | False | False | False |
| **7498** | False | False | False | False | False | False | False | False | False | False | ... | False | False | False | False | False | False | False | False | False | False |
| **7499** | False | False | False | False | False | False | False | False | False | False | ... | False | False | False | False | False | False | False | False | False | False |
| **7500** | False | False | False | False | False | False | False | False | False | False | ... | False | False | False | False | False | False | False | False | False | False |

7501 rows × 119 columns

In [14]:

1

cleaned\_df.to\_csv('df\_clean1.csv',index **=** **False**)

In [15]:

1

cleaned\_df.columns

Out[15]:

Index(['Duloxetine', 'Premarin', 'Yaz', 'abilify', 'acetaminophen', 'actonel',  
 'albuterol HFA', 'albuterol aerosol', 'alendronate', 'allopurinol',  
 ...  
 'trazodone HCI', 'triamcinolone Ace topical', 'triamterene',  
 'trimethoprim DS', 'valaciclovir', 'valsartan', 'venlafaxine XR',  
 'verapamil SR', 'viagra', 'zolpidem'],  
 dtype='object', length=119)

2. Execute the code used to generate association rules with the Apriori algorithm. Provide screenshots that demonstrate that the code is error free.

**from** mlxtend.frequent\_patterns **import** apriori

2

**from** mlxtend.frequent\_patterns **import** association\_rules

3

**import** matplotlib

4

**import** seaborn **as** sns

5

**import** matplotlib.pyplot **as** plt

6

**%**matplotlib inline

In [17]:

1

data **=**pd.read\_csv('df\_clean1.csv')

In [18]:

1

data.head()

Out[18]:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Duloxetine** | **Premarin** | **Yaz** | **abilify** | **acetaminophen** | **actonel** | **albuterol HFA** | **albuterol aerosol** | **alendronate** | **allopurinol** | **...** | **trazodone HCI** | **triamcinolone Ace topical** | **triamterene** | **trimethoprim DS** | **valaciclovir** | **valsartan** | **venlafaxine XR** | **verapamil SR** | **viagra** | **zolpidem** |
| **0** | False | False | False | True | False | False | True | True | False | True | ... | False | False | False | False | False | False | False | False | False | False |
| **1** | False | False | False | False | False | False | False | False | 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 | False | False | False | False | False | False | False | False |
| **3** | False | False | False | False | False | False | False | False | False | True | ... | False | False | False | False | False | False | False | False | False | False |
| **4** | False | False | False | True | False | False | False | False | False | False | ... | False | False | False | False | False | False | False | False | False | False |

5 rows × 119 columns

In [19]:

1

data.shape

Out[19]:

(7501, 119)

*# The most comment medications*

In [54]:

1

count **=**data.loc[:,:].sum()

2

pop\_item**=**count.sort\_values(ascending**=False**).head(5)

3

pop\_item**=**pop\_item.to\_frame()

4

pop\_item**=**pop\_item.reset\_index()

5

pop\_item**=**pop\_item.rename(columns **=** {'index':'medications',0:'count'})

6

print(pop\_item)

7

medications count  
0 abilify 1788  
1 amphetamine salt combo xr 1348  
2 carvedilol 1306  
3 glyburide 1282  
4 diazepam 1229

In [ ]:

1

*#data visualization for the most prescribed meds*

In [55]:

1

plt.rcParams['figure.figsize']**=**(10,6)

2

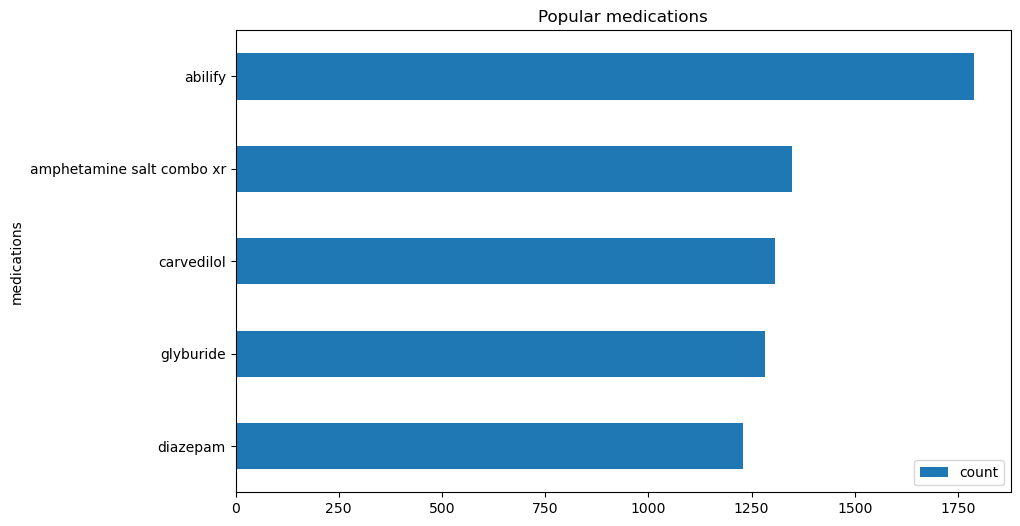
ax**=**pop\_item.plot.barh(x**=**'medications', y **=**'count')

3

plt.title('Popular medications')

4

plt.gca().invert\_yaxis()



In [34]:

1

*#creating apriori object called rules*

2

rules**=**apriori(data,min\_support **=** 0.02,use\_colnames **=True**)

3

rules.head(5)

1

*#data visualization for the most prescribed meds*

In [55]:

1

plt.rcParams['figure.figsize']**=**(10,6)

2

ax**=**pop\_item.plot.barh(x**=**'medications', y **=**'count')

3

plt.title('Popular medications')

4

plt.gca().invert\_yaxis()

1

In [34]:

1

*#creating apriori object called rules*

2

rules**=**apriori(data,min\_support **=** 0.02,use\_colnames **=True**)

3

rules.head(5)

Out[34]:

|  |  |  |
| --- | --- | --- |
|  | **support** | **itemsets** |
| **0** | 0.046794 | (Premarin) |
| **1** | 0.238368 | (abilify) |
| **2** | 0.020397 | (albuterol aerosol) |
| **3** | 0.033329 | (allopurinol) |
| **4** | 0.079323 | (alprazolam) |

In [36]:

1

2

rul\_table **=** association\_rules(rules,metric**=**'lift', min\_threshold **=** 1)

3

rul\_table.head(20)

4

Out[36]:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **antecedents** | **consequents** | **antecedent support** | **consequent support** | **support** | **confidence** | **lift** | **leverage** | **conviction** | **zhangs\_metric** |
| **0** | (amlodipine) | (abilify) | 0.071457 | 0.238368 | 0.023597 | 0.330224 | 1.385352 | 0.006564 | 1.137144 | 0.299568 |
| **1** | (abilify) | (amlodipine) | 0.238368 | 0.071457 | 0.023597 | 0.098993 | 1.385352 | 0.006564 | 1.030562 | 0.365218 |
| **2** | (amphetamine salt combo) | (abilify) | 0.068391 | 0.238368 | 0.024397 | 0.356725 | 1.496530 | 0.008095 | 1.183991 | 0.356144 |
| **3** | (abilify) | (amphetamine salt combo) | 0.238368 | 0.068391 | 0.024397 | 0.102349 | 1.496530 | 0.008095 | 1.037830 | 0.435627 |
| **4** | (abilify) | (amphetamine salt combo xr) | 0.238368 | 0.179709 | 0.050927 | 0.213647 | 1.188845 | 0.008090 | 1.043158 | 0.208562 |
| **5** | (amphetamine salt combo xr) | (abilify) | 0.179709 | 0.238368 | 0.050927 | 0.283383 | 1.188845 | 0.008090 | 1.062815 | 0.193648 |
| **6** | (atorvastatin) | (abilify) | 0.129583 | 0.238368 | 0.047994 | 0.370370 | 1.553774 | 0.017105 | 1.209650 | 0.409465 |
| **7** | (abilify) | (atorvastatin) | 0.238368 | 0.129583 | 0.047994 | 0.201342 | 1.553774 | 0.017105 | 1.089850 | 0.467950 |
| **8** | (carvedilol) | (abilify) | 0.174110 | 0.238368 | 0.059725 | 0.343032 | 1.439085 | 0.018223 | 1.159314 | 0.369437 |
| **9** | (abilify) | (carvedilol) | 0.238368 | 0.174110 | 0.059725 | 0.250559 | 1.439085 | 0.018223 | 1.102008 | 0.400606 |
| **10** | (cialis) | (abilify) | 0.076523 | 0.238368 | 0.023997 | 0.313589 | 1.315565 | 0.005756 | 1.109585 | 0.259747 |
| **11** | (abilify) | (cialis) | 0.238368 | 0.076523 | 0.023997 | 0.100671 | 1.315565 | 0.005756 | 1.026851 | 0.314943 |
| **12** | (citalopram) | (abilify) | 0.087188 | 0.238368 | 0.024397 | 0.279817 | 1.173883 | 0.003614 | 1.057552 | 0.162275 |
| **13** | (abilify) | (citalopram) | 0.238368 | 0.087188 | 0.024397 | 0.102349 | 1.173883 | 0.003614 | 1.016889 | 0.194486 |
| **14** | (clopidogrel) | (abilify) | 0.059992 | 0.238368 | 0.022797 | 0.380000 | 1.594172 | 0.008497 | 1.228438 | 0.396502 |
| **15** | (abilify) | (clopidogrel) | 0.238368 | 0.059992 | 0.022797 | 0.095638 | 1.594172 | 0.008497 | 1.039415 | 0.489364 |
| **16** | (dextroamphetamine XR) | (abilify) | 0.081056 | 0.238368 | 0.027463 | 0.338816 | 1.421397 | 0.008142 | 1.151921 | 0.322617 |
| **17** | (abilify) | (dextroamphetamine XR) | 0.238368 | 0.081056 | 0.027463 | 0.115213 | 1.421397 | 0.008142 | 1.038604 | 0.389252 |
| **18** | (abilify) | (diazepam) | 0.238368 | 0.163845 | 0.052660 | 0.220917 | 1.348332 | 0.013604 | 1.073256 | 0.339197 |
| **19** | (diazepam) | (abilify) | 0.163845 | 0.238368 | 0.052660 | 0.321400 | 1.348332 | 0.013604 | 1.122357 | 0.308965 |

In [38]:

1

top\_three\_rules **=** rul\_table.sort\_values('confidence',ascending **=False**).head(3)

2

top\_three\_rules

Out[38]:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **antecedents** | **consequents** | **antecedent support** | **consequent support** | **support** | **confidence** | **lift** | **leverage** | **conviction** | **zhangs\_metric** |
| **30** | (metformin) | (abilify) | 0.050527 | 0.238368 | 0.023064 | 0.456464 | 1.914955 | 0.011020 | 1.401255 | 0.503221 |
| **25** | (glipizide) | (abilify) | 0.065858 | 0.238368 | 0.027596 | 0.419028 | 1.757904 | 0.011898 | 1.310962 | 0.461536 |
| **28** | (lisinopril) | (abilify) | 0.098254 | 0.238368 | 0.040928 | 0.416554 | 1.747522 | 0.017507 | 1.305401 | 0.474369 |

In [39]:

1

top\_three\_rules **=** rul\_table.sort\_values('lift',ascending **=False**).head(3)

2

top\_three\_rules

Out[39]:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **antecedents** | **consequents** | **antecedent support** | **consequent support** | **support** | **confidence** | **lift** | **leverage** | **conviction** | **zhangs\_metric** |
| **75** | (carvedilol) | (lisinopril) | 0.174110 | 0.098254 | 0.039195 | 0.225115 | 2.291162 | 0.022088 | 1.163716 | 0.682343 |
| **74** | (lisinopril) | (carvedilol) | 0.098254 | 0.174110 | 0.039195 | 0.398915 | 2.291162 | 0.022088 | 1.373997 | 0.624943 |
| **73** | (glipizide) | (carvedilol) | 0.065858 | 0.174110 | 0.022930 | 0.348178 | 1.999758 | 0.011464 | 1.267048 | 0.535186 |

In [40]:

1

top\_three\_rules **=** rul\_table.sort\_values('support',ascending **=False**).head(3)

2

top\_three\_rules

Out[40]:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **antecedents** | **consequents** | **antecedent support** | **consequent support** | **support** | **confidence** | **lift** | **leverage** | **conviction** | **zhangs\_metric** |
| **8** | (carvedilol) | (abilify) | 0.174110 | 0.238368 | 0.059725 | 0.343032 | 1.439085 | 0.018223 | 1.159314 | 0.369437 |
| **9** | (abilify) | (carvedilol) | 0.238368 | 0.174110 | 0.059725 | 0.250559 | 1.439085 | 0.018223 | 1.102008 | 0.400606 |
| **19** | (diazepam) | (abilify) | 0.163845 | 0.238368 | 0.052660 | 0.321400 | 1.348332 | 0.013604 | 1.122357 | 0.308965 |

In [52]:

1

filtered\_rules **=** rul\_table[rul\_table['lift'] **>** 1]

In [53]:

1

sorted\_rules **=** filtered\_rules.sort\_values(by**=**'lift', ascending**=False**)

2

sorted\_rules

Out[53]:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **antecedents** | **consequents** | **antecedent support** | **consequent support** | **support** | **confidence** | **lift** | **leverage** | **conviction** | **zhangs\_metric** |
| **75** | (carvedilol) | (lisinopril) | 0.174110 | 0.098254 | 0.039195 | 0.225115 | 2.291162 | 0.022088 | 1.163716 | 0.682343 |
| **74** | (lisinopril) | (carvedilol) | 0.098254 | 0.174110 | 0.039195 | 0.398915 | 2.291162 | 0.022088 | 1.373997 | 0.624943 |
| **73** | (glipizide) | (carvedilol) | 0.065858 | 0.174110 | 0.022930 | 0.348178 | 1.999758 | 0.011464 | 1.267048 | 0.535186 |
| **72** | (carvedilol) | (glipizide) | 0.174110 | 0.065858 | 0.022930 | 0.131700 | 1.999758 | 0.011464 | 1.075829 | 0.605334 |
| **31** | (abilify) | (metformin) | 0.238368 | 0.050527 | 0.023064 | 0.096756 | 1.914955 | 0.011020 | 1.051182 | 0.627330 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **86** | (losartan) | (diazepam) | 0.132116 | 0.163845 | 0.023464 | 0.177598 | 1.083943 | 0.001817 | 1.016724 | 0.089231 |
| **53** | (amphetamine salt combo xr) | (losartan) | 0.179709 | 0.132116 | 0.025463 | 0.141691 | 1.072479 | 0.001721 | 1.011156 | 0.082387 |
| **52** | (losartan) | (amphetamine salt combo xr) | 0.132116 | 0.179709 | 0.025463 | 0.192735 | 1.072479 | 0.001721 | 1.016135 | 0.077869 |
| **60** | (glyburide) | (atorvastatin) | 0.170911 | 0.129583 | 0.023730 | 0.138846 | 1.071482 | 0.001583 | 1.010756 | 0.080466 |
| **61** | (atorvastatin) | (glyburide) | 0.129583 | 0.170911 | 0.023730 | 0.183128 | 1.071482 | 0.001583 | 1.014956 | 0.076645 |

94 rows × 10 columns

3. Provide values for the support, lift, and confidence of the association rules table.

I have all the values on values and here i am showing the top values for support, Lift and confidence are listed below.

Confidence: (metformin) --> (abilify) 0.456464

Support: (carvedilol)--> (abilify) 0.059725

Lift: (carvedilol)--> (lisinopril) 2.291162

4. Explain the top three relevant rules generated by the Apriori algorithm. Include a screenshot of the top three relevant rules.

Confidence: Measures the likelihood of the consequent given the antecedent. Higher confidence means a stronger association. Looking at the data, (metformin) -> (abilify) has a high confidence, it means that patients who have been prescribed metformin are likely to also be prescribed abilify.

Lift: Indicates how much more likely the consequent is given the antecedent compared to random chance. A lift greater than 1 shows a positive relationship.

Support: Reflects the frequency of the antecedent and consequent appearing together in the dataset. Higher support means the rule is more frequent. According to our data, the rule shows that (carvedilol) -> (abilify) has high support, it means this combination is common among patients.

**Part IV: Data Summary and Implications**

D. Summarize your data analysis by doing the following:

1. Summarize the significance of support, lift, and confidence from the results of the analysis.

* **Confidence** indicates the likelihood of the consequent given the antecedent. A high confidence, such as in the rule (metformin) -> (abilify), suggests that patients prescribed metformin are likely to also be prescribed abilify.
* **Lift** measures how much more likely the consequent is given the antecedent compared to random chance. A lift greater than 1, as seen in the rule (metformin) -> (abilify), indicates a positive association.
* **Support** reflects how often the antecedent and consequent appear together in the dataset. A high support, like in the rule (carvedilol) -> (abilify), indicates that this combination is common among patients.

2. Discuss the practical significance of your findings from the analysis.

The analysis shows important associations between different prescriptions, helping hospitals understand common prescription patterns among patients. For example, the strong association between medications like metformin and abilify indicates that these drugs are often prescribed together. By recognizing these frequent combinations, the hospital can ensure that they are adequately stocked with these medications and consider these patterns when developing treatment protocols, potentially leading to more personalized and efficient patient care.

3. Recommend a course of action for the real-world organizational situation from part A1 based on the results from part D1.

Hospitals need to work on making sure that commonly associated medications, like metformin and abilify, are consistently in stock. This will help avoid shortages and ensure that patients receive timely treatment. In addition, inform patients about the common combinations of prescriptions and their potential benefits or risks, helping them better understand their treatment plans.

**Part V: Attachments**

E. Provide a Panopto video

<https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=a135ac27-c19c-450c-a0e7-b1d800cee072>

F. Record *all* web sources you used to acquire data or segments of third-party code to support the application. Ensure the web sources are reliable.

G. Acknowledge sources, using in-text citations and references, for content that is quoted, paraphrased, or summarized.

Chemistry8526. (2023, January 31). *Boosting sales with data: The power of market basket analysis in retail*. Medium. <https://medium.com/@chemistry8526/boosting-sales-with-data-the-power-of-market-basket-analysis-in-retail-c79cc10a14df>

H. Demonstrate professional communication in the content and presentation of your submission.