Aalto University

PROJECT REPORT

CS-E4070 - Special Course in Machine Learning and Data Science: Learning from electronic health records

Disproportionality analysis

Author:
Rohan Chauhan

May 11, 2018

Aalto University
School of Science

1 Project Objectives

Itemset mining is used to identify frequent items in dataset having a minimum support specified by user. The problem is also sometimes viewed as "association rule mining" to discover rules that show how variables are associated in dataset. For example, onion,potato-¿burger in a supermarket dataset might imply that a customer buying onion and potato usually buys a burger too. The main objective of this project is to identify an adverse drug event which is a injury incurred to patient due to usage of drug. The main objective can be broken into a series of mini objectives to achieve the desired target as given below:

1. Implement frequent itemset mining
2. Implement association rule mining
3. Perform disproportionality analysis on association rules [3] for adverse drug event

2 Working

In this section, we describe the algorithm and library used for disproportionality analysis.

2.1 Algorithm

We used the Apriori algorithm as given by Agrawal [1] as the basis of this analysis. For learning association rules from the itemsets generated by apriori algorithm, there are different measures of interest such as support, confidence, lift and conviction. We chose **lift** for learning association rules because it is a better measure than confidence and support.

2.2 Mlxtend

Because we had to process a lot of transactions, rather than using a naive implementation for association rule learning, we decide to use mlxtend library [4].Mlxtend is a general purpose day to day use machin learning library. Mlxtend library implements frequent patterns api which uses apriori algorithm

to generate itemsets and then learns association rules from the generated itemsets.

3 Dataset

Two datasets were used in the project. Initially, a normal itemset mining data such as Dataset1 was used. After validation on Dataset1, MIMIC dataset was finally used to identify adverse drug events [2].

MIMIC is a huge dataset but we used PRESCRIPTIONS table, DIAGNOSES_ICD table, PATIENTS table and D_ICD_DIAGNOSES table. PATIENTS table contained details about patient like his gender which we thought could be used to reproduce the results in [3]. D_ICD_DIAGNOSES table contains description of ICD9 codes which was used to found adverse drug event. DIAGNOSES_ICD table was used to find patients suffering from adverse drug event and finally PRESCRIPTIONS table was used to find the drugs administrated to our target group and control group.

We chose patients diagnosed with ICD9 code **E9342** which is "Anticoagulants causing adverse effects in therapeutic use". There were 685 patients in target group and rest of the patients were in control group. No time period was set for the analysis.

4 Findings

Our findings are divided into two sections. First section discusses category of drugs with max adverse events followed by discussion on Association Rules.

4.1 Top category of Drugs with Maximum Adverse events

The top 3 adverse drug events were "Anticoagulants causing adverse effects in therapeutic use", "Antineoplastic and immunosuppressive drugs causing adverse effects in therapeutic use" and Adrenal cortical steroids causing adverse effects in therapeutic use".Top 10 adverse drug events are listed in Figure 1.

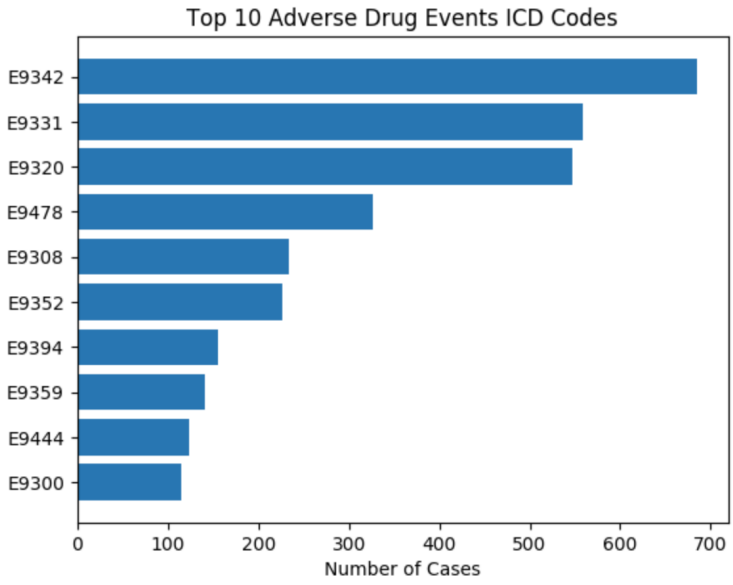

Figure 1: Adverse Drug Event

4.2 Association Rules

As no time period was set, the code is still running. So, right now , we cannot discuss association rules, however, the code is correct.

5 Conclusions

Our conclusion is that MIMIC dataset might hold a lot of scope for future research considering its detail and size. It would be also nice to see the parallel implementation of frequent itemsets algorithms which can produce faster results.

References

[1] Agrawal, R., Srikant, R., et al. Fast algorithms for mining association rules. In *Proc. 20th int. conf. very large data bases, VLDB* (1994), vol. 1215, pp. 487–499.

[2] Johnson, A. E., Pollard, T. J., Shen, L., Li-wei, H. L., Feng, M., Ghassemi, M., Moody, B., Szolovits, P., Celi, L. A., and Mark, R. G. Mimic-iii, a freely accessible critical care database. *Scientific data 3* (2016), 160035.

[3] Karlsson, I., Papapetrou, P., Asker, L., Boström, H., and Persson, H. E. Mining disproportional itemsets for characterizing groups of heart failure patients from administrative health records. In *Proceedings of the 10th International Conference on PErvasive Technologies Related to Assistive Environments* (2017), ACM, pp. 394–398.

[4] Raschka, S. Mlxtend: Providing machine learning and data science utilities and extensions to python's scientific computing stack. *The Journal of Open Source Software 3*, 24 (Apr. 2018).

6 Appendix

```
# coding: utf-8

# In[1]:


#Imports
import pandas as pd
from mlxtend.frequent_patterns import apriori
from mlxtend.frequent_patterns import association_rules
from mlxtend.preprocessing import TransactionEncoder
import operator


# In[2]:


#Importing data
prescriptions = pd.read_csv('PRESCRIPTIONS.csv')
#Converting to pandas datetime format
```

```
prescriptions['STARTDATE'] = pd.to_datetime(prescriptions['STARTDATE'])
prescriptions['ENDDATE'] = pd.to_datetime(prescriptions['ENDDATE'])
patient_diagnosis = pd.read_csv('DIAGNOSES_ICD.csv')
patients = pd.read_csv('PATIENTS.csv')
diagnosis_codes = pd.read_csv('D_ICD_DIAGNOSES.csv')


# In[3]:


#All Adverse Drug Effect
adverse_drug_effect = diagnosis_codes[diagnosis_codes['LONG_TITLE'].str

top_ade = []
#Patients having Adverse Drug Effect for each code
for ade in adverse_drug_effect:
    if (len(patient_diagnosis[patient_diagnosis['ICD9_CODE'] == ade])>0
        top_ade.append((ade,len(patient_diagnosis[patient_diagnosis['ICI

#Sorting in Desceding order
top_ade.sort(key=operator.itemgetter(1), reverse=True)
print(top_ade)


# In[4]:


#Selecting ICD9_CODE = 'E9342' as Adverse Drug Event with 685 patients
target = patient_diagnosis[patient_diagnosis['ICD9_CODE'] == 'E9342'][']
control = prescriptions[~prescriptions['SUBJECT_ID'].isin(target)]['SUB


# In[5]:


'''
Time Period Setting
new_target = prescriptions[((prescriptions['STARTDATE'] - prescriptions
```

```
new_control = prescriptions[((prescriptions['STARTDATE'] - prescriptions
'''


# In[ ]:


transaction_target = []
for patient in target:
    a=[]
    drugs =  prescriptions[prescriptions['SUBJECT_ID'] == patient]['FORM
    for drug in drugs:
        a.append(drug)
    transaction_target.append(a)

transaction_control = []
for patient in control:
    a=[]
    drugs =  prescriptions[prescriptions['SUBJECT_ID'] == patient]['FORM
    for drug in drugs:
        a.append(drug)
    transaction_control.append(a)


# In[ ]:


#Converting transaction list into Right format
te = TransactionEncoder()
te_ary = te.fit(transaction_target).transform(transaction_target)
transaction_target_final = pd.DataFrame(te_ary, columns=te.columns_)

te_ary2 = te.fit(transaction_control).transform(transaction_control)
transaction_control_final = pd.DataFrame(te_ary2, columns=te.columns_)


# In[ ]:
```

```
#Finding the frequent itemsets
frequent_itemsets_target =  apriori(transaction_target_final, min_suppo
frequent_itemsets_control =  apriori(transaction_control_final, min_sup


# In[ ]:


#Association Rules Generation from Frequent Itemsets
print("\n\n Generating Association Rules for Target Group \n\n")
association_rules_target = association_rules(frequent_itemsets_target,
print(association_rules_target)
print("\n\n Generating Association Rules for Control Group \n\n")
association_rules_control = association_rules(frequent_itemsets_control
print(association_rules_control)
```