Lab 1: Big Data

Frede Emnetu (100704524)

In [3]:
```
from apyori import apriori
import numpy as np
import pandas as pd
import matplotlib.pyplot as pl
import time
```

Output function

In [2]:
```
def output(rules):
    for item in rules:
        for x in item:
            # first index of the inner list
            # Contains base item and add item
            pair = list(x[0])
            items = [x for x in pair]
            print("Rule: " + str(items[0]) + " -> " + str(items[1]))

            #second index of the inner list
            print("Support: " + str(x[1]))

            #third index of the list located at 0th
            #of the third index of the inner list

            print("Confidence: " + str(x[2][0][2]))
            print("Lift: " + str(x[2][0][3]))
            print("=====================================")
```

Retail Data Set

In [3]:
```
# Retail dataset
retail = pd.read_csv("http://fimi.uantwerpen.be/data/retail.dat",delimiter=" ", c
retail_L = retail.values.tolist()
# retail_NN = []
```

removing Nan values

In [4]:
```
retail_NN=[]
for x in retail_L:
    retail_NN.append([i for i in x if str(i) != 'nan'])
```

creating sections in data

In [5]:
```
association = []
times = []
sections = [0.002,.005,0.01,0.05,.1]
"""
17632 = 20%
35265 = 40%
52897 = 60%
70530 = 80%
88163 = 100%
"""
for count, x in enumerate(sections):
#     new_list = retail_L[0:int(len(retail_L)*(sections[count]/100))]
    new_list = retail_L[0:int(len(retail_L)*(sections[count]))]
    association.append(new_list)
```

In [11]:
```
pairslen = []
actualpairs = []
times = []
# print(np.shape(association[5]))
for x in range(5):
    start = time.time()
    association_rules = apriori(association[x], min_support=0.01, min_confidence=
    association_results = list(association_rules)
    pairslen.append(len(association_results))
    actualpairs.append(association_results)
    end = time.time()
    times.append(end - start)
```

Pairs

In [12]:
```
for x in range(len(sections)):
    print()

    print()
#     print(pairslen[x])
    print('with %g percent of the dataset I got %d pairs.' % (sections[x], pairs]
    print()
    output(actualpairs)
```

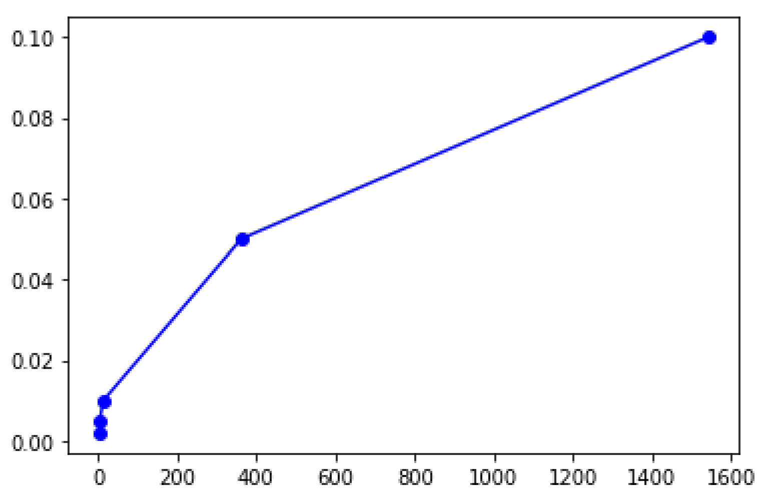
```
with 0.002 percent of the dataset I got 44 pairs.

Rule: 18.0 -> 38.0
Support: 0.011695906432748537
Confidence: 1.0
Lift: 3.489795918367347
=====================================
Rule: 32.0 -> 152.0
Support: 0.011695906432748537
Confidence: 0.5
Lift: 6.107142857142858
=====================================
Rule: 32.0 -> 178.0
Support: 0.011695906432748537
Confidence: 0.6666666666666666
Lift: 8.142857142857142
=====================================
```

Time vs Section graph

In [13]:
```
pl.plot(times, [x for x in sections], 'bo-')
```

Out[13]: [<matplotlib.lines.Line2D at 0x7fc89a400c40>]



Netflix dataset

In [ ]:
```
netflix = pd.read_csv('netflix.data',delimiter=" ", on_bad_lines='skip',skip_blar
netflix_L = netflix.values.tolist()
```

removing Nan values

In [ ]:
```
netflix_NN = []
for x in netflix_L:
    netflix_NN.append([i for i in x if str(i) != 'nan'])
```

In [ ]:
```
association = []
times = []
sections = [0.002,.005,0.01,0.05,.1]

for count, x in enumerate(sections):
    new_list = netflix_L[0:int(len(netflix_L)*(sections[count]))]
    association.append(new_list)
```

In [ ]:
```
pairs = []
times = []
for x in range(5):
    start = time.time()
    association_rules = apriori(association[x], min_support=0.0045, min_confidenc
    association_results = list(association_rules)
    pairs.append(len(association_results))
    end = time.time()
    times.append(end - start)
```

Pairs

In [ ]:
```
for x in range(len(sections)):
    print()

    print()
#     print(pairsLen[x])
    print('with %g percent of the dataset I got %d pairs.' % (sections[x], pairs]
    print()
    output(actualpairs)
```

Time vs Sections graph

In [ ]:
```
pl.plot(times, [x for x in sections], 'bo-')
```

Conclusions

The apyori algorithm is very slow when the data is turned in a list as defined by edureka. I tried using values between 2 and 50 but the algorithm for the retail data took an exceedingly long time for this reason I had to use very low values. As for the netflix data set, it was initially loading but as of finishing this lab I can't load the dataset, otherwise the kernel dies. This occured 5 times and eventually gave up (sry). The netflix dataset produced expected outputs, giving it more data allowed it to create more pairs and of course more rows meant more time spent which produce an approximately log(n) graph with some translation.