Karan Sharma 1310110159

Samkeerth laalam 1310110283

Svs karthik chowdary 1310110331

Vijay Shekhawat 1310110361

README

The dataset is obtained from `NYC Open Data Sets <http://nycopendata.socrata.com/>`_

Usage

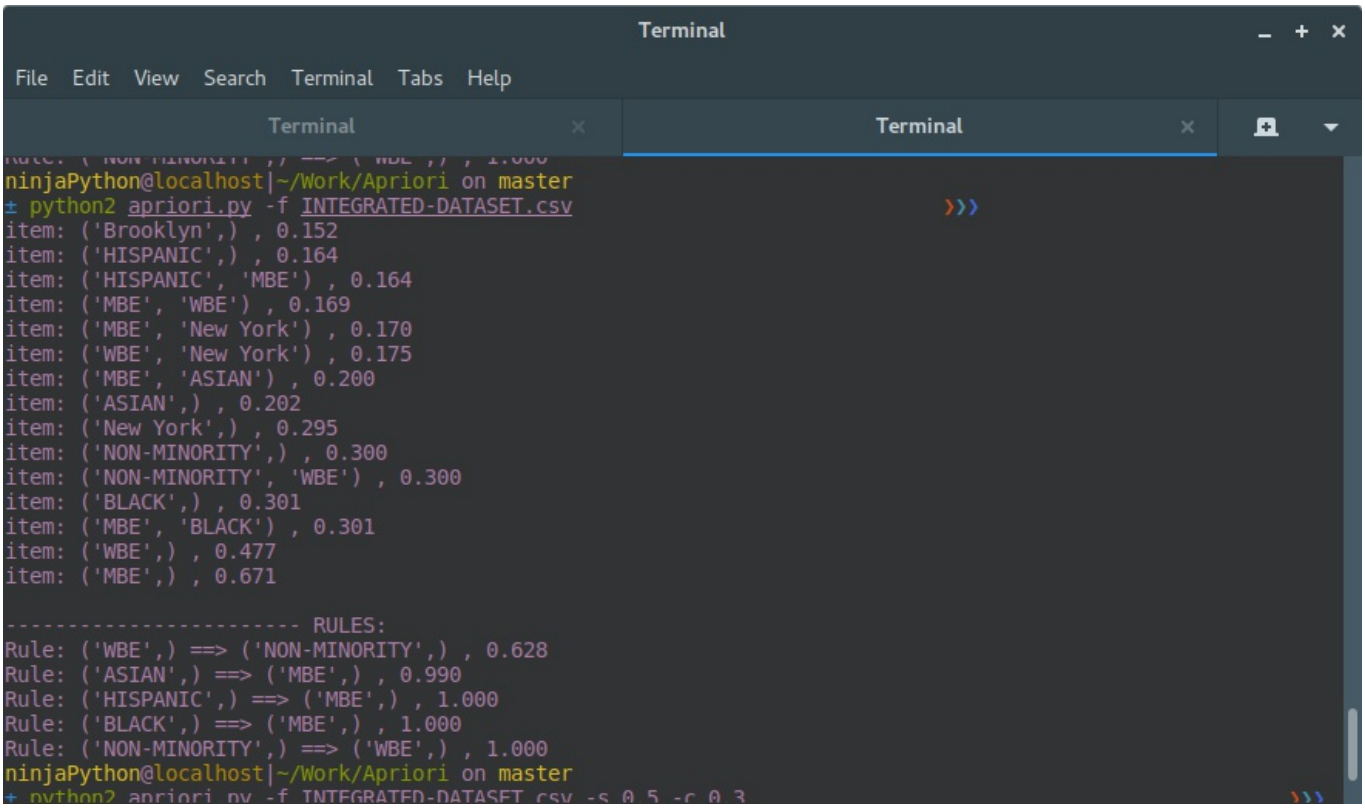
To run the program with dataset provided and default values for *minSupport* = 0.15 and *minConfidence* = 0.6

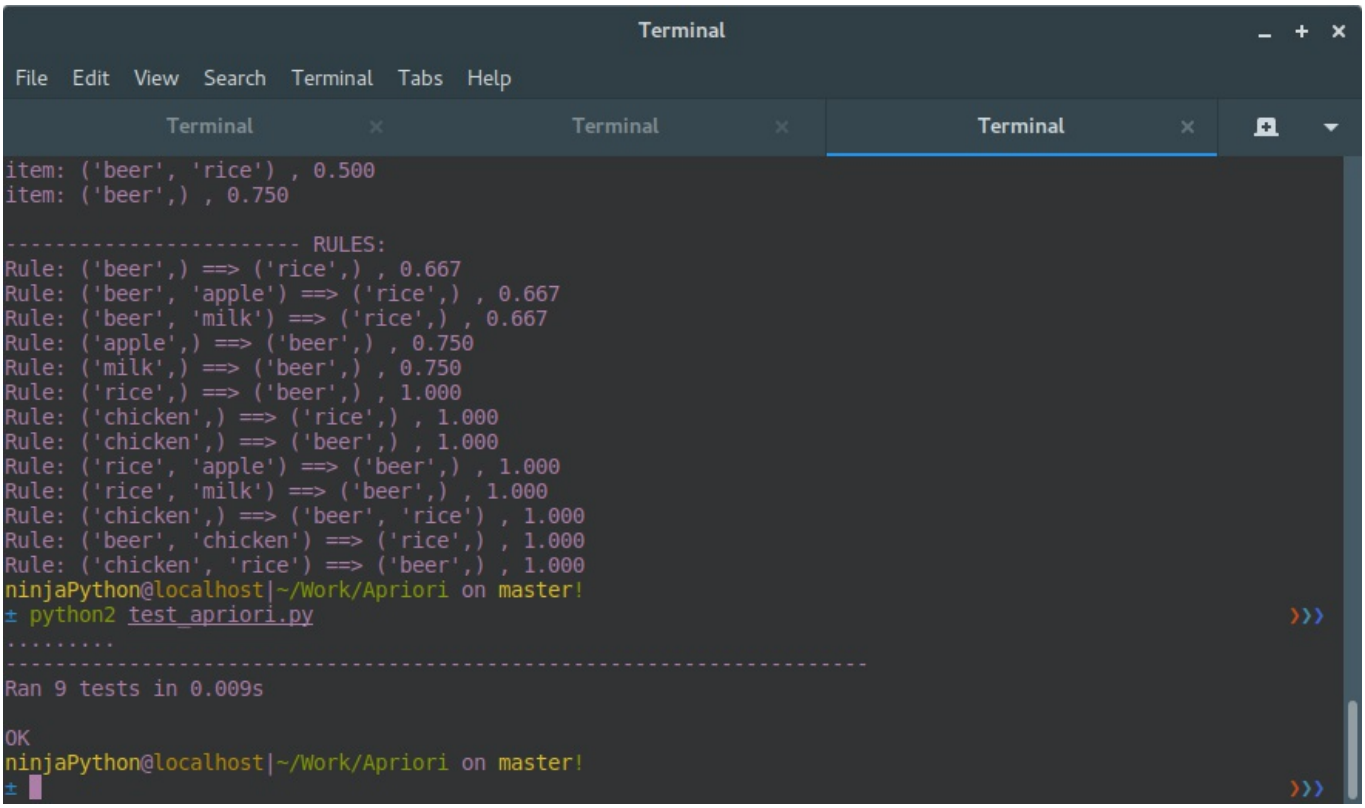
```
python apriori.py -f <<filename>>.csv
```

To run with custom values of *minSupport* and *minConfidence* provided as argument

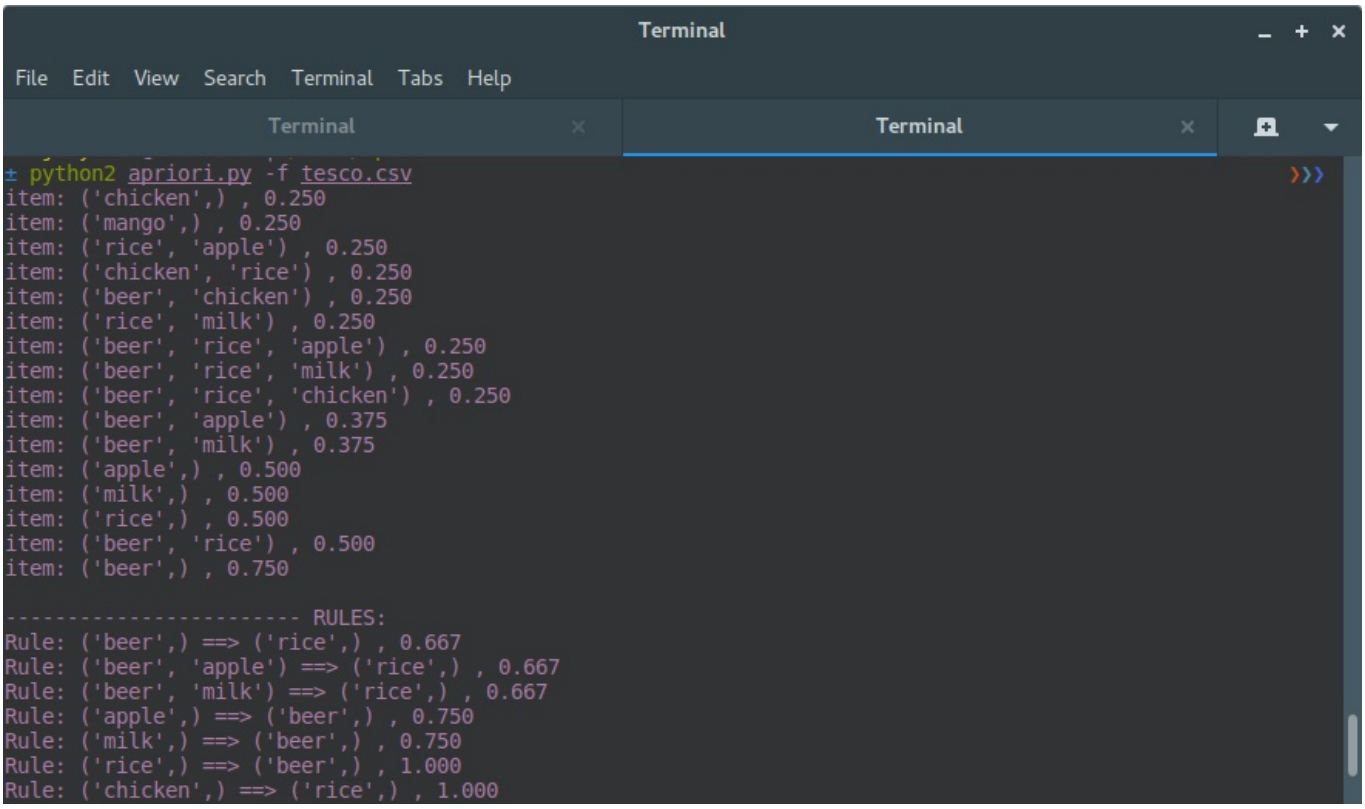
```
python apriori.py -f INTEGRATED-DATASET.csv -s 0.17 -c 0.
68
```

Screenshot


Terminal
File Edit View Search Terminal Tabs Help
Terminal
Terminal
ninjaPython@localhost|~/Work/Apriori on master
± python2 apriori.py -f INTEGRATED-DATASET.csv
item: ('Brooklyn',) , 0.152
item: ('HISPANIC',) , 0.164
item: ('HISPANIC', 'MBE') , 0.164
item: ('MBE', 'WBE') , 0.169
item: ('MBE', 'New York') , 0.170
item: ('WBE', 'New York') , 0.175
item: ('MBE', 'ASIAN') , 0.200
item: ('ASIAN',) , 0.202
item: ('New York',) , 0.295
item: ('NON-MINORITY',) , 0.300
item: ('NON-MINORITY', 'WBE') , 0.300
item: ('BLACK',) , 0.301
item: ('MBE', 'BLACK') , 0.301
item: ('WBE',) , 0.477
item: ('MBE',) , 0.671
---------- RULES:
Rule: ('WBE',) ==> ('NON-MINORITY',) , 0.628
Rule: ('ASIAN',) ==> ('MBE',) , 0.990
Rule: ('HISPANIC',) ==> ('MBE',) , 1.000
Rule: ('BLACK',) ==> ('MBE',) , 1.000
Rule: ('NON-MINORITY',) ==> ('WBE',) , 1.000
ninjaPython@localhost|~/Work/Apriori on master
± python2 apriori.py -f INTEGRATED-DATASET.csv -s 0.5 -c 0.3


Terminal
File Edit View Search Terminal Tabs Help
Terminal
Terminal
Terminal
item: ('beer', 'rice') , 0.500
item: ('beer',) , 0.750
---------- RULES:
Rule: ('beer',) ==> ('rice',) , 0.667
Rule: ('beer', 'apple') ==> ('rice',) , 0.667
Rule: ('beer', 'milk') ==> ('rice',) , 0.667
Rule: ('apple',) ==> ('beer',) , 0.750
Rule: ('milk',) ==> ('beer',) , 0.750
Rule: ('rice',) ==> ('beer',) , 1.000
Rule: ('chicken',) ==> ('rice',) , 1.000
Rule: ('chicken',) ==> ('beer',) , 1.000
Rule: ('rice', 'apple') ==> ('beer',) , 1.000
Rule: ('rice', 'milk') ==> ('beer',) , 1.000
Rule: ('chicken',) ==> ('beer', 'rice') , 1.000
Rule: ('beer', 'chicken') ==> ('rice',) , 1.000
Rule: ('chicken', 'rice') ==> ('beer',) , 1.000
ninjaPython@localhost|~/Work/Apriori on master!
± python2 test_apriori.py
....

Ran 9 tests in 0.009s
OK
ninjaPython@localhost|~/Work/Apriori on master!
±



Code

```
import sys

from itertools import chain, combinations
from collections import defaultdict
from optparse import OptionParser


def subsets(arr):
    return chain(*[combinations(arr, i + 1) for i, a in e
numerate(arr)])
```

```
def returnMin(SetI, tlist, mins, freqSet):
        SetI = set()
        localSet = defaultdict(int)

        for i in SetI:
                for t in tlist:
                        if i.issubset(t):
                                freqSet[i] += 1
                                localSet[i] += 1

        for i, count in localSet.is():
                sup = float(count)/len(tlist)

                if sup >= mins:
                        SetI.add(i)

        return SetI


def joinSet(SetI, length):
        return set([i.union(j) for i in SetI for j in Set
I if len(i.union(j)) == length])


def getSetItlist(diterator):
    tlist = list()
    SetI = set()
    for resultdata in diterator:
```

```
        t = frozenset(resultdata)
        tlist.append(t)
        for i in t:
            SetI.add(frozenset([i]))
    return SetI, tlist


def apriori(diter, mins, minc):
    SetI, tlist = getSetItlist(diter)

    freqSet = defaultdict(int)
    SetLarge = dict()
    assocRules = dict()

    oneCSet = returnMin(SetI,tlist,
                                        mins,
                                        freqSet)

    currentl = oneCSet
    k = 2
    while(currentl != set([])):
        SetLarge[k-1] = currentl
        currentl = joinSet(currentl, k)
        currentCSet = returnMin(currentl,
                                                tlist,
                                                mins,
                                                freqSet)
        currentl = currentCSet
```

```
        k = k + 1

    def supGet(i):
            return float(freqSet[i])/len(tlist)

    toreti = []
    for key, value in SetLarge.is():
        toreti.extend([(tuple(i), supGet(i))
                           for i in value])

    torule = []
    for key, value in SetLarge.is()[1:]:
        for i in value:
            _subsets = map(frozenset, [x for x in subsets
(i)])
            for elt in _subsets:
                left = i.difference(elt)
                if len(left) > 0:
                    conf = supGet(i)/supGet(elt)
                    if conf >= minc:
                        torule.append(((tuple(elt), tuple
(left)),
                                                conf))
    return toreti, torule


def result(is, rules):
    for i, sup in sorted(is, key=lambda (i, sup): sup):
```

```
        print "i: %s , %.3f" % (str(i), sup)
    print "\nRULES:"
    for rule, conf in sorted(rules, key=lambda (rule, con
f): conf):
        pre, post = rule
        print "Rule: %s ==> %s , %.3f" % (str(pre), str(p
ost), conf)


def dataFromFile(fname):
        file_iter = open(fname, 'rU')
        for line in file_iter:
                line = line.strip().rstrip(',')

                resultdata = frozenset(line.split(','))
                yield resultdata


if __name__ == "__main__":


    inputfile = None
    if options.input is None:
            inputfile = sys.stdin
    elif options.input is not None:
            inputfile = dataFromFile(options.input)
    else:
            print 'No dataset filename specified, system
```

```
with exit\n'
            sys.exit('System will exit')

    mins = options.minS
    minc = options.minC

    is, rules = apriori(inputfile, mins, minc)

    result(is, rules)
```