DHRUV GARG 16 BCE 1190

LARGE SCALE DATA PROCESSING

DIGITAL ASSIGNMENT 1

INTRODUCTION

Apriori is an algorithm for frequent item set mining and association rule learning. It proceeds by identifying the frequent individual items in the database and extending them to larger and larger item sets as long as those item sets appear sufficiently often in the dataset.

We can use the Apriori algorithm in a number of applications- most commonly used in market basket analysis. Market Basket Analysis is a modelling technique based upon the theory that if you buy a certain group of items, you are more (or less) likely to buy another group of items.

In this scenario we are implementing the Apriori algorithm on the US Census Adult Income dataset from Kaggle. The primary goal was to perform prediction task is to determine whether a person makes over $50K a year. Attribute information of the dataset is as follows: Income (<=50K, >50K), age, workclass, education, marital-status, occupation, relationship, race and gender. Instead of doing the prediction, we will use the Apriori algorithm to find strong rules in the dataset.

METHOD

Programming language used for implementation: Python 3.7

Apriori algorithm: I have used the apriori module from a python library called efficient-apriori. This repository contains an efficient, well-tested implementation of the apriori algorithm as descriped in the original paper by Agrawal et al, published in 1994.

Major commands used, with explanation:

1. **pd.read_csv:** Pandas module to import datasets in csv format
2. **df.isnull().sum() :** To check for total number of null values in columns
3. **df.replace() :** To replace one string in the dataset with another
4. **fillna(df[col].mode()):** To fill NaN values with mode of that column
5. **df.drop([cols]) :**To drop columns from dataset
6. **open(filename) :** Open a file from the system
7. **line.split(‘,’) :** Split data from the dataset, using comma delimiter
8. **apriori() :** Function call to generate rules with min_sup and min_conf values

Description of apriori():

efficient_apriori.**apriori**(*transactions: List[tuple], min_support: float = 0.5, min_confidence: float = 0.5, max_length: int = 8, verbosity: int = 0*)

The Apriori algorithm works in two phases. Phase 1 iterates over the transactions several times to build up itemsets of the desired support level. Phase 2 builds association rules of the desired confidence given the itemsets found in Phase 1. Both of these phases may be correctly implemented by exhausting the search space, i.e. generating every possible itemset and checking it’s support. Parameters description:

- **transactions** (*list of tuples, or a callable returning a generator*) – The transactions may be either a list of tuples, where the tuples must contain hashable items.
- **min_support** (*float*) – The minimum support of the rules returned.
- **min_confidence** (*float*) – The minimum confidence of the rules returned.
- **max_length** (*int*) – The maximum length of the itemsets and the rules.
- **verbosity** (*int*) – The level of detail printing when the algorithm runs. Either 0, 1 or 2.

Commands explanation and code workflow has been described through the use of comments in the Jupyter Notebook printout appended.

RESULTS

Some rules are obvious in retrospect

RULE 1: {Husband} => {Male}

RULE 2: {<=50K, Husband} => {Male}

RULE 3: {Husband, middle-aged} => {Male, Married-civ-spouse}

The above rules are pretty obvious.

Some rules are due to the skewed dataset

RULE 4: {Male} => {White}

RULE 5: {<=50K, White} -> {Male}

The above rules are due to the fact that majority records are that of males and most are of income <= 50K

Some rules are more interesting

RULE 6: {HS-grad} => {<=50K}

This means that if a person is only a high-school graduate, he/she will probably earn a salary <=50K annually.

RULE 7: {<=50K, young} => {Never-married}

Since the person is young and earns <=50K, he/she would not have married yet.

RULE 8: {Husband} => {Male, Married-civ-spouse, middle-aged}

Husbands are probably in their middle-age (35-50 years) when they are married-civ-spouse.

RULE 9: {Never-married, White} => <50K

If the person earns <=50K annually, the person is most probably a young person, and hence is unlikely to have been married or had a divorce.

CONCLUSION

The Apriori algorithm was successfully implemented on a real dataset using a built-in function from Python. We were able to extract some strong and interesting values at the minimum support of 0.2 and minimum confidence of 0.4.

LSDP - DIGITAL 1 - DHRUV GARG (16BCE1190)

Apriori algorithm implementation

Dataset chosen: US Adult Income

In [161]:
```
import numpy as np   #for linear algebra
import pandas as pd  #for data processing, CSV file I/O
```

Import dataset by giving path and file name

In [162]:
```
data = pd.read_csv('./adult_income.csv')
```

In [163]:
```
data.describe()
```

Out[163]:

	age	fnlwgt	education.num	capital.gain	capital.loss	hours.per.week
count	32561.000000	3.256100e+04	32561.000000	32561.000000	32561.000000	32561.000000
mean	38.581647	1.897784e+05	10.080679	1077.648844	87.303830	40.437456
std	13.640433	1.055500e+05	2.572720	7385.292085	402.960219	12.347429
min	17.000000	1.228500e+04	1.000000	0.000000	0.000000	1.000000
25%	28.000000	1.178270e+05	9.000000	0.000000	0.000000	40.000000
50%	37.000000	1.783560e+05	10.000000	0.000000	0.000000	40.000000
75%	48.000000	2.370510e+05	12.000000	0.000000	0.000000	45.000000
max	90.000000	1.484705e+06	16.000000	99999.000000	4356.000000	99.000000

The describe() function of pandas helps us to understand the variance of data values. A very large variance in the dataset must be controlled as it generally impacts the model negatively.

Check for missing values

In [164]:
```
data.isnull().sum()
```

Out[164]:
```
age               0
workclass         0
fnlwgt            0
education         0
education.num     0
marital.status    0
occupation        0
relationship      0
race              0
sex               0
capital.gain      0
capital.loss      0
hours.per.week    0
native.country    0
income            0
dtype: int64
```

Print the first 5 rows of the dataset

In [165]:
```
data.head(5)
```

Out[165]:

	age	workclass	fnlwgt	education	education.num	marital.status	occupation	relationship	race	s
0	90	?	77053	HS-grad	9	Widowed	?	Not-in-family	White	Fem
1	82	Private	132870	HS-grad	9	Widowed	Exec-managerial	Not-in-family	White	Fem
2	66	?	186061	Some-college	10	Widowed	?	Unmarried	Black	Fem
3	54	Private	140359	7th-8th	4	Divorced	Machine-op-inspct	Unmarried	White	Fem
4	41	Private	264663	Some-college	10	Separated	Prof-specialty	Own-child	White	Fem

We can see that there are '?' in places of incomplete data. Either these rows must be dropped or the data cells must be imputed with mean or mode value for that column.

To replace '?' with mode values

First we replace '?' with numpy's NaN, then replace it with mode of that column.

In [166]:
```
data = data.replace('?', np.nan)
```

In [167]:
```
for column in data.columns:
    data[column].fillna(data[column].mode()[0], inplace=True)
```

This for loop runs for all data records in the dataset. When it encounters a numpy-NaN value, it replaces the NaN with the mode value of that column.

In [168]:
```
data.head(5)
```

Out[168]:

	age	workclass	fnlwgt	education	education.num	marital.status	occupation	relationship	race	s
0	90	Private	77053	HS-grad	9	Widowed	Prof-specialty	Not-in-family	White	Fem
1	82	Private	132870	HS-grad	9	Widowed	Exec-managerial	Not-in-family	White	Fem
2	66	Private	186061	Some-college	10	Widowed	Prof-specialty	Unmarried	Black	Fem
3	54	Private	140359	7th-8th	4	Divorced	Machine-op-inspct	Unmarried	White	Fem
4	41	Private	264663	Some-college	10	Separated	Prof-specialty	Own-child	White	Fem

Drop the columns unlikely to produce any strong rules

In [169]:
```
data = data.drop(['age', 'workclass', 'fnlwgt', 'education.num', 'occupation
', 'capital.gain', 'capital.loss', 'hours.per.week', 'native.country'], axis
=1)
```

In [170]:
```
data.head()
```

Out[170]:

	education	marital.status	relationship	race	sex	income
0	HS-grad	Widowed	Not-in-family	White	Female	<=50K
1	HS-grad	Widowed	Not-in-family	White	Female	<=50K
2	Some-college	Widowed	Unmarried	Black	Female	<=50K
3	7th-8th	Divorced	Unmarried	White	Female	<=50K
4	Some-college	Separated	Own-child	White	Female	<=50K

Export the cleaned data as new csv file

In [171]:
```
data.to_csv('adult_income_cleaned.csv')
```

Apply the apriori algorithm

Import the apriori module from the efficient_apriori library

In [172]:
```
from efficient_apriori import apriori
```

Generating data from csv file to be passed to the built-in 'apriori' function

In [173]:
```
def data_generator(filename):
    def data_gen():
        with open(filename) as file:
            for line in file:
                yield tuple(k.strip() for k in line.split(','))
    return data_gen
```

Since we have a dataset that is large, you may pass a function returning a generator instead of a list. This will be passed to the apriori() function in a later step.

In [174]:
```
transactions = data_generator('adult_income_cleaned.csv')
```

Check that the function has been generated

In [175]:
```
print(transactions)
```

```
<function data_generator.<locals>.data_gen at 0x7f26b5a75e18>
```

Since higher values of minimum support and minimum confidence werent producing good rules, their values have been kept low.

Implementation 1: Minimum support = 0.4, Minimum confidence = 0.4

In [176]:
```
itemsets, rules = apriori(transactions, min_support=0.4, min_confidence=0.4)
```

The apriori() function takes the data, min_support and min_confidence values as input and returns the itemsets and rules which satisfy the criteria.

If the min_support is a large value, the algorithm will take more time to converge, ie. it will take a longer time to terminate.

Printing the rules

In [177]:
```
print(rules)
```

```
[{Male} -> {<=50K}, {<=50K} -> {Male}, {White} -> {<=50K}, {<=50K} -> {White}
, {Male} -> {Husband}, {Husband} -> {Male}, {Married-civ-spouse} -> {Husband}
, {Husband} -> {Married-civ-spouse}, {Married-civ-spouse} -> {Male}, {Male} -
> {Married-civ-spouse}, {White} -> {Male}, {Male} -> {White}, {White} -> {Mar
ried-civ-spouse}, {Married-civ-spouse} -> {White}, {Male, White} -> {<=50K},
{<=50K, White} -> {Male}, {<=50K, Male} -> {White}, {White} -> {<=50K, Male},
{Male} -> {<=50K, White}, {<=50K} -> {Male, White}, {Male, Married-civ-spouse
} -> {Husband}, {Husband, Married-civ-spouse} -> {Male}, {Husband, Male} -> {
Married-civ-spouse}, {Married-civ-spouse} -> {Husband, Male}, {Male} -> {Husb
and, Married-civ-spouse}, {Husband} -> {Male, Married-civ-spouse}]
```

To get more interesting rules, we lower the min_support and/or the min_confidence parameter values.

Implementation 2: Minimum support = 0.2, Minimum confidence = 0.4

In [178]:
```
itemsets, rules = apriori(transactions, min_support=0.2, min_confidence=0.4)
```

Printing the rules

In [179]:
```
print(rules)
```

```
[{Female} -> {<=50K}, {HS-grad} -> {<=50K}, {Husband} -> {<=50K}, {Male} -> {
<=50K}, {<=50K} -> {Male}, {Married-civ-spouse} -> {<=50K}, {Never-married} -
> {<=50K}, {<=50K} -> {Never-married}, {Not-in-family} -> {<=50K}, {White} ->
{<=50K}, {<=50K} -> {White}, {>50K} -> {Male}, {Married-civ-spouse} -> {>50K}
, {>50K} -> {Married-civ-spouse}, {>50K} -> {White}, {Female} -> {White}, {HS
-grad} -> {Male}, {HS-grad} -> {White}, {Male} -> {Husband}, {Husband} -> {Ma
le}, {Married-civ-spouse} -> {Husband}, {Husband} -> {Married-civ-spouse}, {W
hite} -> {Husband}, {Husband} -> {White}, {Married-civ-spouse} -> {Male}, {Ma
le} -> {Married-civ-spouse}, {White} -> {Male}, {Male} -> {White}, {White} ->
{Married-civ-spouse}, {Married-civ-spouse} -> {White}, {Never-married} -> {Wh
ite}, {Not-in-family} -> {White}, {Female, White} -> {<=50K}, {<=50K, Female}
-> {White}, {Female} -> {<=50K, White}, {HS-grad, White} -> {<=50K}, {<=50K,
HS-grad} -> {White}, {HS-grad} -> {<=50K, White}, {Husband, Male} -> {<=50K},
{<=50K, Male} -> {Husband}, {<=50K, Husband} -> {Male}, {Husband} -> {<=50K,
Male}, {Husband, Married-civ-spouse} -> {<=50K}, {<=50K, Married-civ-spouse}
-> {Husband}, {<=50K, Husband} -> {Married-civ-spouse}, {Married-civ-spouse}
-> {<=50K, Husband}, {Husband} -> {<=50K, Married-civ-spouse}, {Male, Married
-civ-spouse} -> {<=50K}, {<=50K, Married-civ-spouse} -> {Male}, {<=50K, Male}
-> {Married-civ-spouse}, {Married-civ-spouse} -> {<=50K, Male}, {Male, White}
-> {<=50K}, {<=50K, White} -> {Male}, {<=50K, Male} -> {White}, {White} -> {<
=50K, Male}, {Male} -> {<=50K, White}, {<=50K} -> {Male, White}, {Married-civ
-spouse, White} -> {<=50K}, {<=50K, Married-civ-spouse} -> {White}, {Married-
civ-spouse} -> {<=50K, White}, {Never-married, White} -> {<=50K}, {<=50K, Whi
te} -> {Never-married}, {<=50K, Never-married} -> {White}, {Never-married} ->
{<=50K, White}, {Male, Married-civ-spouse} -> {Husband}, {Husband, Married-ci
v-spouse} -> {Male}, {Husband, Male} -> {Married-civ-spouse}, {Married-civ-sp
ouse} -> {Husband, Male}, {Male} -> {Husband, Married-civ-spouse}, {Husband}
-> {Male, Married-civ-spouse}, {Male, White} -> {Husband}, {Husband, White} -
> {Male}, {Husband, Male} -> {White}, {White} -> {Husband, Male}, {Male} -> {
Husband, White}, {Husband} -> {Male, White}, {Married-civ-spouse, White} -> {
Husband}, {Husband, White} -> {Married-civ-spouse}, {Husband, Married-civ-spo
use} -> {White}, {White} -> {Husband, Married-civ-spouse}, {Married-civ-spous
e} -> {Husband, White}, {Husband} -> {Married-civ-spouse, White}, {Married-ci
v-spouse, White} -> {Male}, {Male, White} -> {Married-civ-spouse}, {Male, Mar
ried-civ-spouse} -> {White}, {White} -> {Male, Married-civ-spouse}, {Married-
civ-spouse} -> {Male, White}, {Male} -> {Married-civ-spouse, White}, {Husband
, Male, Married-civ-spouse} -> {<=50K}, {<=50K, Male, Married-civ-spouse} ->
{Husband}, {<=50K, Husband, Married-civ-spouse} -> {Male}, {<=50K, Husband, M
ale} -> {Married-civ-spouse}, {Male, Married-civ-spouse} -> {<=50K, Husband},
{Husband, Married-civ-spouse} -> {<=50K, Male}, {Husband, Male} -> {<=50K, Ma
rried-civ-spouse}, {<=50K, Married-civ-spouse} -> {Husband, Male}, {<=50K, Ma
le} -> {Husband, Married-civ-spouse}, {<=50K, Husband} -> {Male, Married-civ-
spouse}, {Married-civ-spouse} -> {<=50K, Husband, Male}, {Husband} -> {<=50K,
Male, Married-civ-spouse}, {Male, Married-civ-spouse, White} -> {<=50K}, {<=5
0K, Married-civ-spouse, White} -> {Male}, {<=50K, Male, White} -> {Married-ci
v-spouse}, {<=50K, Male, Married-civ-spouse} -> {White}, {Married-civ-spouse,
White} -> {<=50K, Male}, {Male, Married-civ-spouse} -> {<=50K, White}, {<=50K
, Married-civ-spouse} -> {Male, White}, {<=50K, Male} -> {Married-civ-spouse,
White}, {Married-civ-spouse} -> {<=50K, Male, White}, {Male, Married-civ-spou
se, White} -> {Husband}, {Husband, Married-civ-spouse, White} -> {Male}, {Hus
band, Male, White} -> {Married-civ-spouse}, {Husband, Male, Married-civ-spous
e} -> {White}, {Married-civ-spouse, White} -> {Husband, Male}, {Male, White}
-> {Husband, Married-civ-spouse}, {Male, Married-civ-spouse} -> {Husband, Whi
te}, {Husband, White} -> {Male, Married-civ-spouse}, {Husband, Married-civ-sp
ouse} -> {Male, White}, {Husband, Male} -> {Married-civ-spouse, White}, {Whit
e} -> {Husband, Male, Married-civ-spouse}, {Married-civ-spouse} -> {Husband,
Male, White}, {Male} -> {Husband, Married-civ-spouse, White}, {Husband} -> {M
ale, Married-civ-spouse, White}]
```

RESULTS AND INTERPRETATION

Some rules are obvious in retrospect

RULE 1: {Husband} => {Male}

RULE 2: {<=50K, Husband} => {Male}

RULE 3: {Husband, middle-aged} => {Male, Married-civ-spouse}

The above rules are pretty obvious.

Some rules are due to the skewed dataset

RULE 4: {Male} => {White}

RULE 5: {<=50K, White} -> {Male}

The above rules are due to the fact that majority records are that of males and most are of income <= 50K

Some rules are more interesting

RULE 6: {HS-grad} => {<=50K}

This means that if a person is only a high-school graduate, he/she will probably earn a salary <=50K annually.

RULE 7: {<=50K, young} => {Never-married}

Since the person is young and earns <=50K, he/she would not have married yet.

RULE 8: {Husband} => {Male, Married-civ-spouse, middle-aged}

Husbands are probably in their middle-age (35-50 years) when they are married-civ-spouse.

RULE 9: {Never-married, White} => <50K

If the person earns <=50K annually, the person is most probably a young person, and hence is unlikely to have been married or had a divorce.

Thus, the Apriori algorithm was implemented successfully on the adult-income dataset and the inferences were noted.