Practical-6

Aim:- WAP a program for Apriori Algo in Python

Source code:-

```
import sys
from itertools import chain, combinations
from collections import defaultdict
from optparse import optionparser

def subsets(arr):
    """ Returns non empty subsets of arr"""
    return chain(*[combinations(arr, i+1) for i,
                   a in enumerate(arr)])

def returnItemwithminsupport(itemset, transactionlist,
                             minsupport, freqset):
    itemset = set()
    localset = defaultdict(int)
    for item in itemset:
        for transaction in transactionlist:
            if item.issubset(transaction):
                freqset[item] += 1
                localset[item] += 1
    for item, count in localset.items():
        support = float(count) / len(transactionlist)
        if (support >= minsupport):
            itemset.add(item)
    return itemset
```

```
def joinset(itemset, length):
    return set([i.union(j) for i in itemset for j
                in itemset if len(i.union(j))
                == length])
def getItemSetTransactionList(data_iterator):
    transactionList = list()
    itemSet = set()
    for record in data_iterator:
        transaction = frozenset(record)
        transactionList.append(transaction)
        for item in transaction:
            itemset.add(frozenset([item]))
    return itemset, transactionList
def runApriori(data_iter, minSupport, minConfidence):
    itemset, transactionList = getItemSetTransaction
                               List(data_iter)
    freqSet = defaultdict(int)
    largeSet = dict()
    assocRules = dict()
    oneCset = returnItemWithMinSupport(itemset,
                transactionList, minSupport, freqSet)
    currentLSet = oneCset

k = 2
while(currentLSet != set([])):
    largeSet[k-1] = currentLSet
    currentLSet = joinset(currentLSet, k)
    currentCset = returnItemWithMinSupport
                  (currentLSet, transactionList, minSupport,
                   freqSet)
```

```
        currentLset = currentCset
        k = k + 1
def getSupport(item):
    return float(freqset[item]) / len(transactionList)

toRetItems = []
for key, value in LargeSet.items():
    toRetItems.extend([(tuple(item), getSupport(item))
                       for item in value])

toRetRules = []
for key, value in largeset.items()[1:]:
    for item in value:
        subsets = map(frozenset, [x for x in subset(item)])
        for element in subsets:
            remain = item.difference(element)
            if (items(remain) > 0):
                confidence = getSupport(item)/getSupport(elem
                if confidence >= min confidence:
                    toRetRules.append(((tuple(element),
                                        tuple(remain)), confidence))

return toRetItems, toRetRules
```

```
def printResults(items, rules):
    for item, support in sorted(items, key = lambda
                                (item, support):
    print("item: %s, %.3f" % (str(item), support))
for rule, confidence in sorted(rules, key =
                    lambda(rule, confidence): confidence):
    pre, post = rule
    print(str(pre), str(post), confidence)
def dataFromFile(fname):
    file_iter = open(fname, 'rU')
    for line in file_iter:
        line = line.strip().rstrip(',')
        record = frozenset(line.split(','))
        yield record
if name == 'main':
    optparser = optparser()
    optparser.add.option('-f', 'inputFile',
                         (dest = 'input'),
    help = 'filename containing csv', default
                         = None
optparser.addoption('-s', 'minsupport', 'dest = 'minS',
    help = 'minimum support value', default = 0.15,
    type = 'float')
optparser.add option('-c', 'min confidence', 'dest = 'minC',
    help: 'minimum confidence level', default = 0.6)
        type = 'float')
```

```
optparser.add_option('-c', 'min confidence', 'dest='minC',
    help='minimum confidence level', default=0.6,
    type='float')
(options, args) = optparser.parse_args()
    inFile = None
if options.input is None:
        inFile = sys.stdin
elif (options.input is not None):
        inFile = dataFormatFile(options.Input)
else:
    print 'No dataset filename specified, system
            with exit \n'

    sys.exit("system will exit")

minSupport = options.minS
minSupport = options.minC
items, rules = runApriori(inFile, minSupport,
                          minConfidence)

printResults(items, rules)
```

Practical 7

Aim :- Comparision b/w the classification results by J48 decision tree based method and naive Boysian method

- Performance criteria for evaluating the classifiers:-

There are different criteria for evaluating the performance of the classifier:-

i) classification Accuracy
ii) specificity
iii) Sensitivity recall
iv) Precision

i) Classification Accuracy :- It is the ability to predict categorial class labels. This is the simplest scoring measures.
It calculates the preparation of correctly classified instances.

$$\text{accuracy} = (TP + TN) / (TP + TN + FP + FN)$$

ii) Specificity :- It relates to the classifiers' ability to identify negative rules.

$$\text{Specificity} = TN / (FN + TP)$$

iii) Sensitivity / Recall :- Sensitivity is the proportion of actual positivities which are correctly identified positive by classifiers

$$\text{recall} = TP / (TP + FN)$$

iv) Precision :- This is the measure of retrieved instances that are relevant

$$\text{Precision} = TP / (TP + FP)$$

True Positive (TP): If the instance is positive and is classified as positive

False Negative (FN): If the instance is negative but it is classified positive

False Positive (FP): If the instance is negative but it is classified positive.

• Confusion Matrix :- In the field of machine learning, and specification, a confusion matrix, also known as an error matrix, is a layout that allows visualization of the performance of an Algorithm, typically a supervised learning

Comparision b/w Decision tree & Naive Bayes for diabetes Dataset :-

Confusion Matrix in case of J48 :-

a	b	← classified as
407	93	| a = tested-positive
108	160	| b = tested-negetive

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN}$$

$$= \frac{407+160}{407+160+90+108} = \frac{567}{768} = 0.7382$$

Confusion matrix for Naive Bayes :-

a	b	← classified as
422	78	| a = tested positive
104	164	| b = tested negetive

$$\text{Accuracy} = \frac{TP+TN}{TP+FP+FN+TN} = \frac{586}{768} = 0.7630$$

$$0.7630 > 0.7382$$

Therefore, accuracy of Naive Bayes is better than as compared to J48 for given diabetes dataset.

Practical - 8

Aim :- Demonstration of k-means clustering using weka tool.

Clustering :- Clustering is a collection of data object with the properties -
a) Similer to one another within the same cluster
b) Similar to the object in other cluster.

Cluster Analysis :- Cluster analysis is finding similarities b/w data according to the characteristics found in the data and grouping similar data object into clusters
Major clustering approaches are :-
1) Partitioning 2) Hierarchical 3) Density based
4) Grid Based 5) Model based

K-means Clustering :- K-means is one of the simplest unsupervised learning algorithm that solve the well known clustering problem.
The procedures follows a simple and easy way to classify a given dataset through a certain number of cluster (assume k clusters) fixed a priori. The main idea is to define k centroids, one for each clusters. These centroids should be placed in a cunning way because of different location causes different result. So, the choice is to place them as much possible far away from each other.

The next step is to take each point belonging to a given dataset and associate it to the nearest centroid.

A loop has been generated. As a result of this loop we may notice that the k-centroids change their location step by step until no more changes are done. In other words centroids do not move. finally this algorithm aims at minimizing an objective function, in this case a squad error function. The objective fusn

$$J = \sum_{j=1}^{k} \sum_{i=1}^{k} \| x_i(j) - c_j \|^2$$

where $\|x_i(j) - c_j\|^2$ is a chosen distance measure b/w a data point $x_i(j)$ and the cluster center c_j is an indicator of the distance of the n data points from their respective cluster centers.

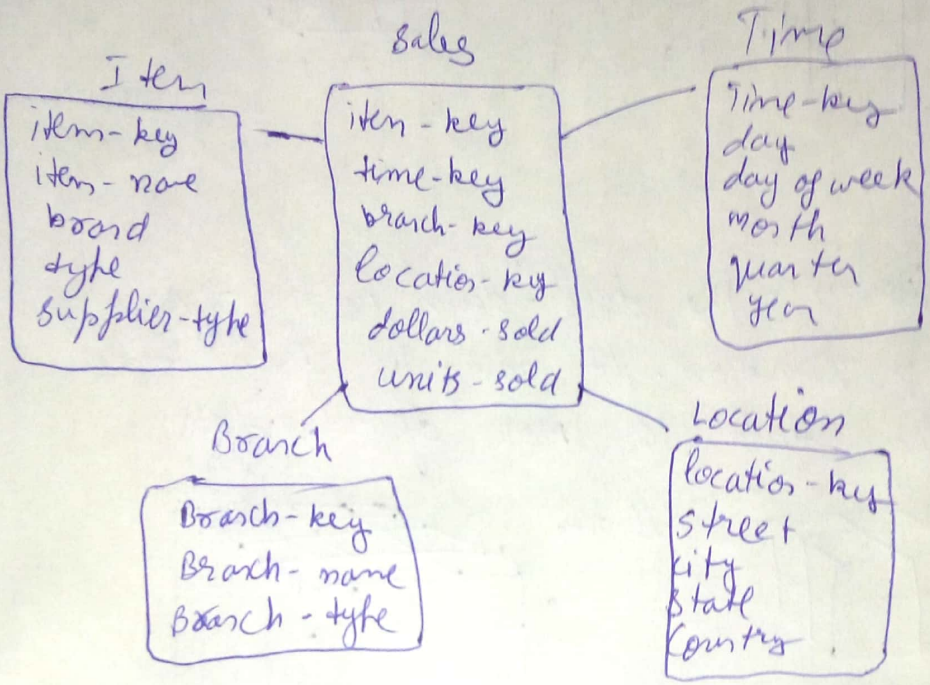
Algorithm :-

1) Place k points into space representing by the objects that are being clustered. These points represents initial group centroids
2) Assign each object to the group that has the closest centroid
3) when all objects have been arranged, recalculate position of k centroids
4) Repeat steps 2 & 3 until the centroids no longer moves.

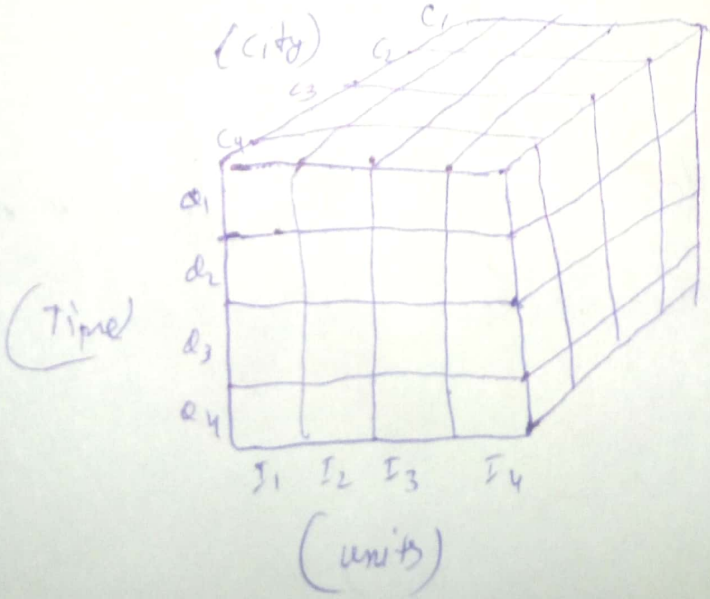
Practical-9

Aim:- Demonstration of OLAP operations on multidimensional data with example.

Let us perform OLAP operations on the given Sales schema

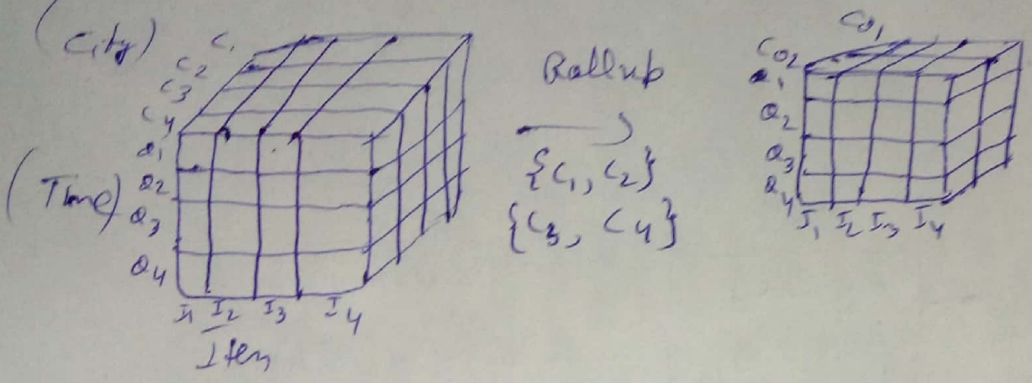


The data cube can be represented as :-

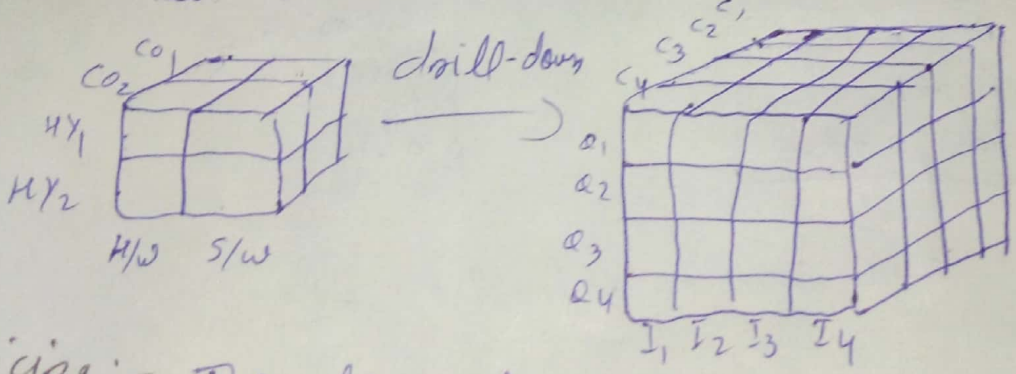


Various OLAP operations are:-

1) Roll up :- The roll up operation performs an aggregation on data cube either by climbing up a concept hierarchy for a dimension.



2) Drill Down :- The drill-down is the reverse of roll up operation. It navigates from less detailed data to more detailed data.



3) Slicing :- The slice operation performs a selection on one dimension of the given cube

4) Dicing :- The dice operation defines a subcube by performing a selection on one or more dimensions