**Big Data Analytics Assignment**

*Name*

*Clg Name*

**Question 1. Explain the Working mechanism of k-means clustering algorithm and also state its algorithm.**

**Ans:**

k-means is an analytical technique that, for a chosen value of k, identifies k clusters of objects based on the objects' proximity to the center of the k groups.

**Working Mechanism: -** Given a collection of objects each with n measurable attributes, k-means is an analytical technique that, for a chosen value of k, identifies k clusters of objects based on the objects' proximity to the center of the k groups. The center is determined as the arithmetic average (mean) of each cluster's n-dimensional vector of attributes.

**K-means Algorithm: -** The k-means algorithm to find k clusters can be described in the following four steps.

**Step-1.** Choose the value of k and the k initial guesses for the centroids.

**Step-2.** Compute the distance from each data point (, ) to each centroid. Assign each point to the closest centroid. This association defines the first k clusters.

In two dimensions, the distance, d, between any two points, (, ) and (, ), in the Cartesian plane is typically expressed by using the Euclidean distance measure formula provided in following equation:

d =

**Step-3.** Compute the centroid, the center of mass, of each newly defined cluster from Step 2.

In two dimensions, the centroid (, ) of the m points in a k-means cluster is calculated as follows:

(, ) =

Thus, (Xc,Yc) is the ordered pair of the arithmetic means of the coordinates of the m points in the cluster.

In this step, a centroid is computed for each of the k clusters.

**Step-4.** Repeat Steps 2 and 3 until the algorithm converges to the output.

a. Assign each point to the closest centroid computed in Step 3.

b. Compute the centroid of newly defined clusters.

c. Repeat until the algorithm reaches the final answer.

**Step-5.** Terminate, the data set with cluster labels is the result.

Convergence is reached when the computed centroids do not change or the centroids and the assigned points oscillate back and forth from one iteration to the next. The latter case can occur when there are one or more points that are equal distances from the computed centroid.

**Question 2. What are the challenges in the clustering?**

**Ans:** Clustering is a method often used for exploratory analysis of the data. In clustering, there are no predictions made. Clustering methods find the similarities between objects according to the object attributes and group the similar objects into clusters. Clustering techniques are utilized in marketing, economics, and various branches of science. A popular clustering method is k-means.

**Current Challenges in Clustering**

**Due to k-means:**

a) The different results via k-means with distinct random initializations are definitely a problem.

b) The number of clusters is (typically) not known a priori that’s basically the characteristic of unsupervised learning problems.

c) Measure of similarities.

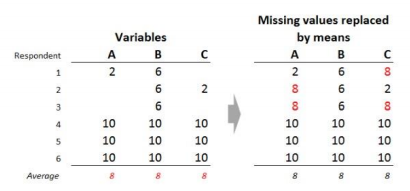
d) Cluster validity.

e) Outliers.

**Question 3. What is missing Data? And explain how it is handling?**

**Ans:** Missing data is a problem where the input table contains some null values or empty row (or column) values. Missing data are defined as values that are not available and that would be meaningful if they are observed. Missing data can be anything from missing sequence, incomplete feature, files missing, information incomplete, data entry error etc. Before you can use data with missing data fields, you need to transform those fields so they can be used for analysis and modelling.

For better understanding, see the following table:



The problem of missing values can be handled as:

**Handling Missing Values**

1. Replace the missing values by 0 (in case of numerical values).

2. Replace it by maximum possible values.

3. Fill the missing values manually based on your own knowledge.

4. Replace them by the average (mean); if numerical, or the most frequent values; if categorical.

Note: The missing values in above table are replaced by their mean.

**Question 4. What is Apriori property?**

**Ans:**

Apriori algorithm is given by R. Agrawal and R. Srikant in 1994 for finding frequent itemsets in a dataset for boolean association rule. Name of the algorithm is Apriori because it uses prior knowledge of frequent itemset properties. We apply an iterative approach or level-wise search where k-frequent itemsets are used to find k+1 itemsets.

To improve the efficiency of level-wise generation of frequent itemsets, an important property is used called Apriori property which helps by reducing the search space.

Apriori is an algorithm for frequent item set mining and association rule learning over relational databases. 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 database.

It is also an expensive method to calculate support because the calculation has to go through the entire database.

**Algorithm:**

**Step 1:** Create a frequency table of all the items that occur in all the transactions. Now, prune the frequency table to include only those items having a threshold support level over 50%. We arrive at this frequency table.

**Step 2:** Make pairs of items such as OP, OB, OM, PB, PM, BM. This frequency table is what you arrive at.

**Step 3:** Apply the same threshold support of 50% and consider the items that exceed 50%

**Step 4:** Look for a set of three items that the customers buy together. Thus, we get this combination.

**Step 5:** Determine the frequency of these two itemsets. You get this frequency table.

**Apriori Property:**

If an item set is considered frequent, then any subset of the frequent item set must also be frequent. This is referred to as the Apriori property. For example, when the support of {bread, jam} is 0.6, the support of {bread} or {jam} is at least 0.6.

**Question 5. Following is a list of 5 transaction that includes items A, B, C and D:**

**T1: {A, B, C}**

**T2: {A, C}**

**T3: {B, C}**

**T4: {A, D}**

**T5: {A, C, D}**

**Which itemset satisfies the minimum support of 0.5?**

**Ans:**

Formula: - Support of (X) = (frequency of X) / Total number of transactions

Given: Total number of transactions = 5

Support of {A, B, C} = 1 / 5 = 0.2

Support of {A, C} = 3 / 5 = 0.6

Support of {B, C} = 2 / 5 = 0.4

Support of {A, D} = 2 / 5 = 0.4

Support of {A, C, D} = 1 / 5 = 0.2

Since, the itemset which satisfies minimum support of 0.5 must have support less than or equal to 0.5.

Therefore, only the itemset {A, C} satisfied minimum support of 0.5.

**Question 6. How are interesting rules identified? How are interesting rules distinguished from coincidental rules?**

**Ans:**

A relationship may be thought of as interesting when the algorithm identifies the relationship with a measure of confidence greater than or equal to a predefined threshold. This predefined threshold is called the minimum confidence.

A higher confidence indicates that the rule (X 🡪 Y) is more interesting or more trustworthy, based on the sample dataset.

In short, Interesting rules is the trustiness or the relationship between items X and Y.

Interesting rules are differs from coincidental rules from the point that interesting rules indicates that there must be a relationship between the items, there is a certainty but the coincidental rules tells that there are the unknown chances of buying itemsets or the incidental relationships between the items.

**Question 7. Code for Apriori Algorithm**

**Ans:**

## install apyori library

## pip install apyori, numpy and pandas

Code: apriori\_implemetation.py

# importing libraries

import numpy as np

import pandas as pd

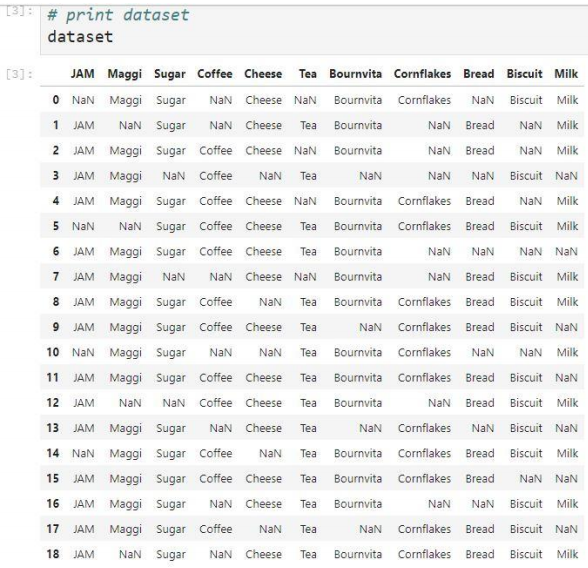
from apyori import apriori

# loading dataset

dataset = pd.read\_csv(“data.csv”)

# taking a look at dataset

dataset

****

# no of rows and columns in our dataset

dataset.shape

****

## converting dataframe to array or list

records = []

for i in range (0, 19):

records.append([str(dataset.values[i, j]) for j in range(0, 11)])

## Building apriori model on dataset

association\_rule = apriori(records, min\_support = 0.5, min\_confidence = 0.7, min\_lift = 1.2, min\_length = 2)

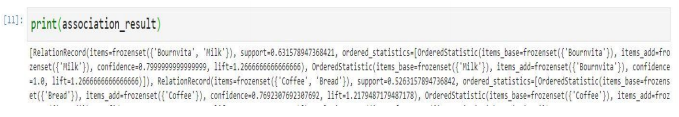
association\_result = list(association\_rule)

## Getting number of rules

print(len(association\_result))

## looking at the rule

print(len(association\_result))



We can conclude that – (by observing first rule)

1. Support value of first rule is **0.6315** This number is calculated by dividing the number of transactions containing ‘Bournvita,’ and ‘Milk’ by the total number of transactions.

2. The confidence level for the rule is **0.7999**, which shows that out of all the transactions that contain both “Milk” and “Bournvita”, **79.99** percent contain ‘Milk’.

3. The lift of **1.266** tells us that ‘Bournvita’ is **1.266** times more likely to be bought by the customers who buy both ‘Milk’.

**Question 8. Explain the ID3 algorithm in python.**

**Ans:**

def find\_entropy(df):

Class = df.keys()[-1] #To make the code generic, changing target variable class name

entropy = 0

values = df[Class].unique()

for value in values:

fraction = df[Class].value\_counts()[value]/len(df[Class])

entropy += -fraction\*np.log2(fraction)

return entropy

def find\_entropy\_attribute(df,attribute):

Class = df.keys()[-1] #To make the code generic, changing target variable class name

target\_variables = df[Class].unique() #This gives all 'Yes' and 'No'

variables = df[attribute].unique() #This gives different features in that

#attribute (like 'Hot','Cold' in Temperature)

entropy2 = 0

for variable in variables:

entropy = 0

for target\_variable in target\_variables:

num = len(df[attribute] [df[attribute] == variable][df[Class] == target\_variable])

den = len(df[attribute][df[attribute]==variable])

fraction = num/(den+eps)

entropy += -fraction\*log(fraction+eps)

fraction2 = den/len(df)

entropy2 += -fraction2\*entropy

return abs(entropy2)

def find\_winner(df):

Entropy\_att = []

IG = []

for key in df.keys()[:-1]:

# Entropy\_att.append(find\_entropy\_attribute(df,key))

IG.append(find\_entropy(df)-find\_entropy\_attribute(df,key))

return df.keys()[:-1][np.argmax(IG)]

def get\_subtable(df, node,value):

return df[df[node] == value].reset\_index(drop=True)

def buildTree(df, tree = None):

Class = df.keys()[-1] #To make the code generic, changing target variable class name

#Here we build our decision tree

#Get attribute with maximum information gain

node = find\_winner(df)

#Get distinct value of that attribute e.g Salary is node and Low,Med and High are values

attValue = np.unique(df[node])

#Create an empty dictionary to create tree

if tree is None:

tree = {}

tree[node] = {}

#We make loop to construct a tree by calling this function recursively.

#In this we check if the subset is pure and stops if it is pure.

for value in attValue:

subtable = get\_subtable(df,node,value)

clValue,counts = np.unique(subtable['Eat'],return\_counts=True)

if len(counts)==1: #Checking purity of subset

tree[node][value] = clValue[0]

else:

tree[node][value] = buildTree(subtable) #Calling the function recursively

return tree