CS 524: Data Mining

Assignment 1

Problem 1:

PCA Analysis

The **principal components** of a collection of points in a real coordinate space are a sequence of unit vectors, where the -the vector is the direction of a line that best fits the data while being orthogonal to the first vectors. Here, a best-fitting line is defined as one that minimizes the average squared distance from the points to the line. These directions constitute an orthonormal basis in which different individual dimensions of the data are linearly uncorrelated. **Principal component analysis** (**PCA**) is the process of computing the principal components and using them to perform a change of basis on the data, sometimes using only the first few principal components and ignoring the rest.

PCA is used in exploratory data analysis and for making predictive models. It is commonly used for dimensionality reduction by projecting each data point onto only the first few principal components to obtain lower-dimensional data while preserving as much of the data's variation as possible. The first principal component can equivalently be defined as a direction that maximizes the variance of the projected data. The i-th principal component can be taken as a direction orthogonal to the first principal component that maximizes the variance of the projected data.

Data Preparation:

To use the Data from movement_libras, the program uses Pandas library to read the data. The program also creates a list from 1 to 90 for the header file of the pandas dataframe. The Program divides the given data into X and Y where X represents the features and Y represents the class.

X <- Data column 0 to 89

Y <- Data Column 90

Further, the program uses the sklearn library to standardize the data.

PCA Algo:

To get EigenVectors and Eigenvalues from the given data, programs follow the following steps

1. Subtract mean from the data
2. Get covariance matrix of X_Mean
3. We now calculate the eigenvalues and eigenvectors from the data using the numpy library.
4. Sort the eigenvalues and return

Problem 2:

Introduction

In the area of association rule mining, 3 major algorithms were discussed: Apriori, FP tree and ECLAT. The purpose of this assignment is to implement these algorithms on various datasets, analyze their space and time complexity, and conclude which algorithm is better for which dataset. The scope of this project is

limited by the hardware capabilities, such as RAM, number of cores/processors, and CPU clock cycles, availability and usage of GPUs et al.

Theoretical Knowledge

Apriori Algorithm:

The Apriori algorithm uses frequent itemsets to generate association rules, and it is designed to work on the databases that contain transactions. With the help of these association rules, it determines how strongly or how weakly two objects are connected. This algorithm uses a **breadth-first search** and **Hash Tree** to calculate the itemset associations efficiently. It is the iterative process for finding the frequent itemsets from the large dataset.

This algorithm was given by R. **Agrawal** and **Srikant** in the year **1994**. It is mainly used for *market basket analysis* and helps to find those products that can be bought together. It can also be used in the healthcare field to find drug reactions for patients.

Eclat :

Eclat Algorithm is abbreviated as Equivalence class clustering and bottom up lattice transversal algorithm. It is an algorithm for finding frequent item sets in a transaction or database. It is one of the best methods of Association Rule Learning. Which means the Eclat algorithm is used to generate frequent item sets in a database.

Eclat algorithm uses a Depth first search for discovering frequent item sets, whereas Apriori algorithm uses breadth first search. It represents the data in a vertical manner unlike Apriori algorithm which represents data in horizontal pattern. This vertical pattern of the Eclat algorithm makes it a faster algorithm compared to Apriori algorithm. Hence, the Eclat algorithm is a more efficient and scalable version of the Association Rule Learning.

FP Growth:

This algorithm is an improvement to the Apriori method. A frequent pattern is generated without the need for candidate generation. FP growth algorithm represents the database in the form of a tree called a frequent pattern tree or FP tree.

This tree structure will maintain the association between the itemsets. The database is fragmented using one frequent item. This fragmented part is called a “pattern fragment”. The itemsets of these fragmented patterns are analyzed. Thus with this method, the search for frequent item sets is reduced comparatively.

Apriori Algorithm:

Data Preparation:

In this step, the program first reads the data from the dataset file. Program uses pandas library to read the dataset from txt document which is in tabular type of format. This result is in a pandas dataframe. We further convert this to numpy as it is desirable for our algo. We read transaction by transaction and store that in map called “**mapofdata**”.

Mining K itemsets:

For generating frequent datasets, the program uses support counting with hashing. We hash the itemset as a frozenset. Here we generate all the k-itemsets. First we generate size 1 and then 2, then so on.

To minimize the time complexity, we used pruning. First after generating the 1 size itemset with given minsup. We can check the current itemset we are considering if it contains itemset from previous frequent itemset or not. If it doesn't we prune it. So, whenever we find a new itemset we initialize its count with 1 or if it was already present then increase it count. At the end, after we have traversed all transactions and got all possible k size items, we check if it satisfies minsup property. If doesn't, remove it.

Eclat Algo:

Data Preparation:

First we apply the same Data preparation as what was done in the previous Apriori Algo. After we get mapofdata which is a map which holds transaction ids as key and items as its values in set. Now we will transform it in vertical order into a map tidset, where keys will be items and values will be transaction id in the form of a set.

Mining :

Now as we have a vertical dataset in the form of map, here at each level k, we will take intersections of tidsets of two itemsets at the just previous level. If the number of transactions is greater than minsup then we will add it to our frequent sets. As transaction id are stored as set, taking intersection and union is comparatively efficient. If at any level k, no sets are generated, the program will stop generating itemsets.

FP Growth:

Data Preparation

This algo requires that in each transaction items are sorted in descending values according to their support values. So I sort the values of and stored in the set as for of list

Data Mining:

The FPGrowth method indexes the database for fast support computation via the use of an augmented prefix tree called the frequent pattern tree (FP-tree). Each node in the tree is labeled with a single item, and each child node represents a different item. Each node also stores the support information for the itemset comprising the items on the path from the root to that node. The FP-tree is constructed as follows. Initially the tree contains as root the null item $\varnothing$. Next, for each tuple $\langle t,X \rangle \in D$, where $X = i(t)$, we insert the itemset X into the FP-tree, incrementing the count of all nodes along the path that represents X. If X shares a prefix with some previously inserted transaction, then X will follow

the same path until the common prefix. For the remaining items in X, new nodes are created under the common prefix, with counts initialized to 1. The FP-tree is complete when all transactions have been inserted.

Observation Tables

Dataset: T20

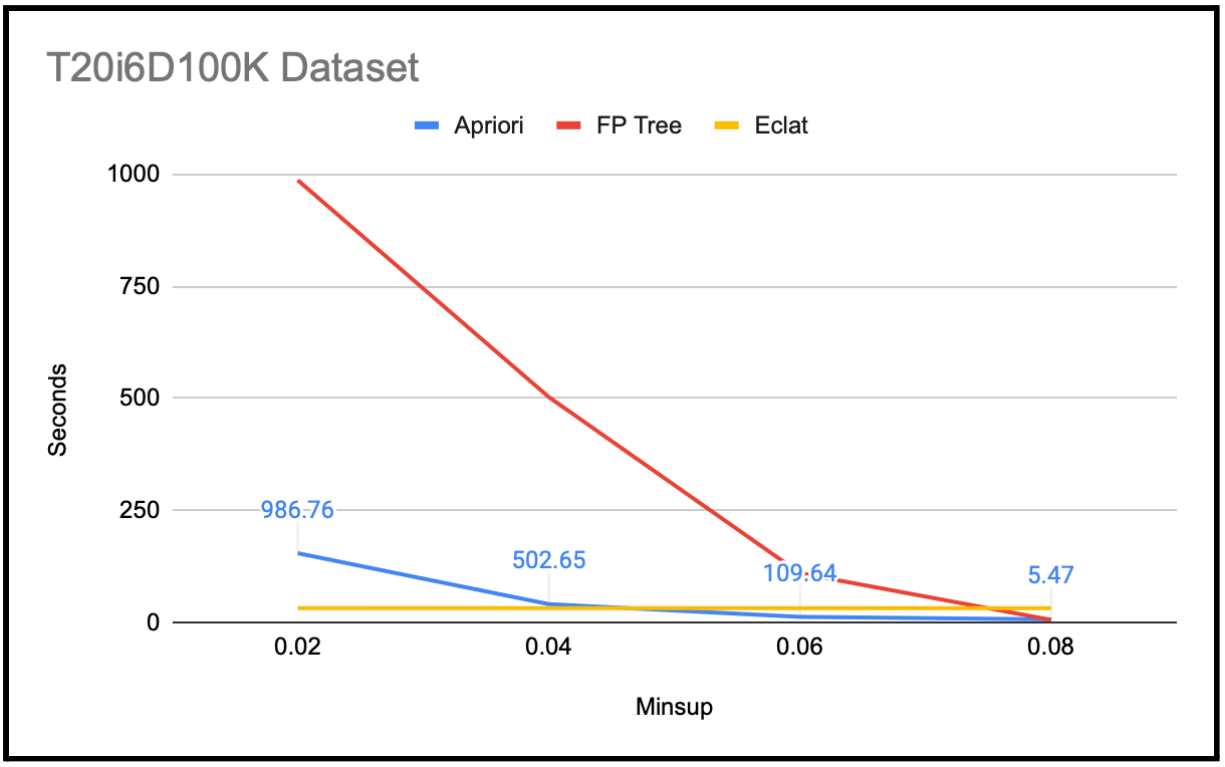
Minsup	Apriori	FP Tree	Eclat
0.02	154.3 seconds	986.76 sec	31.80 sec
0.04	40.65 sec	502.65 sec	31.79 sec
0.06	12.24 sec	109.64 sec	31.71 sec
0.08	6.67 sec	5.47	31.61 sec

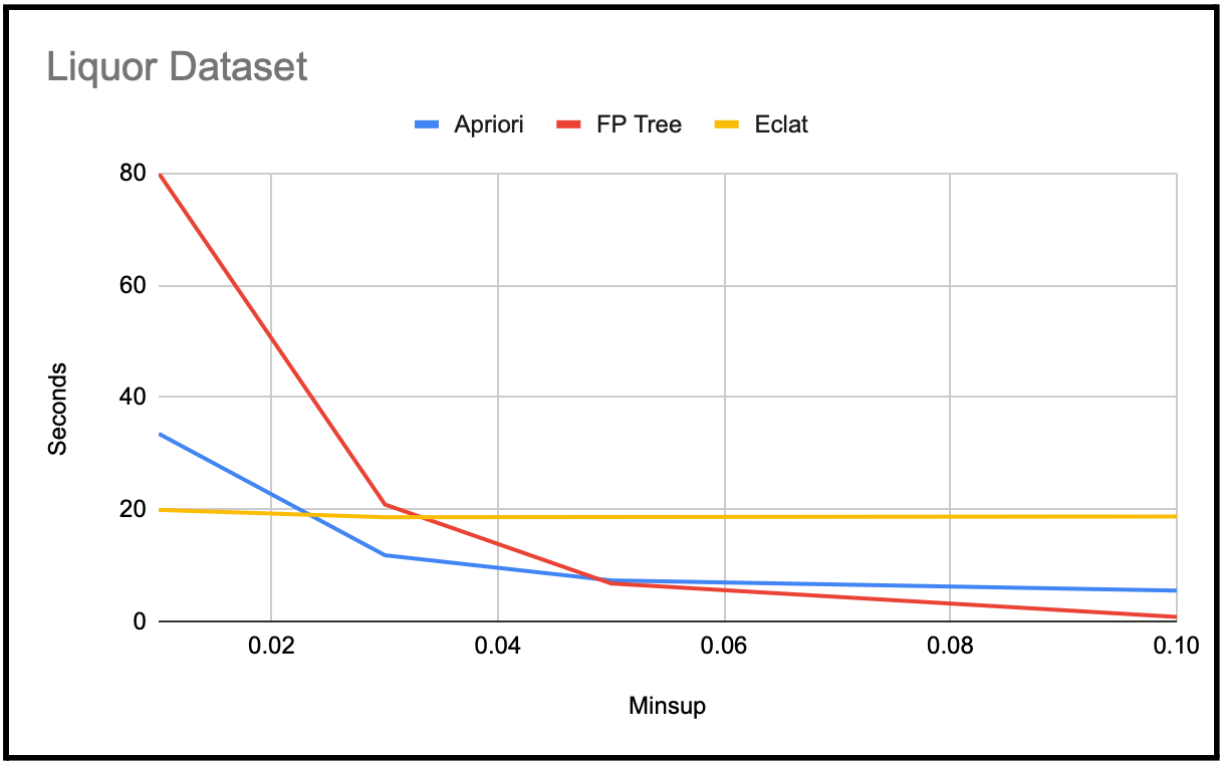
Liquor Dataset

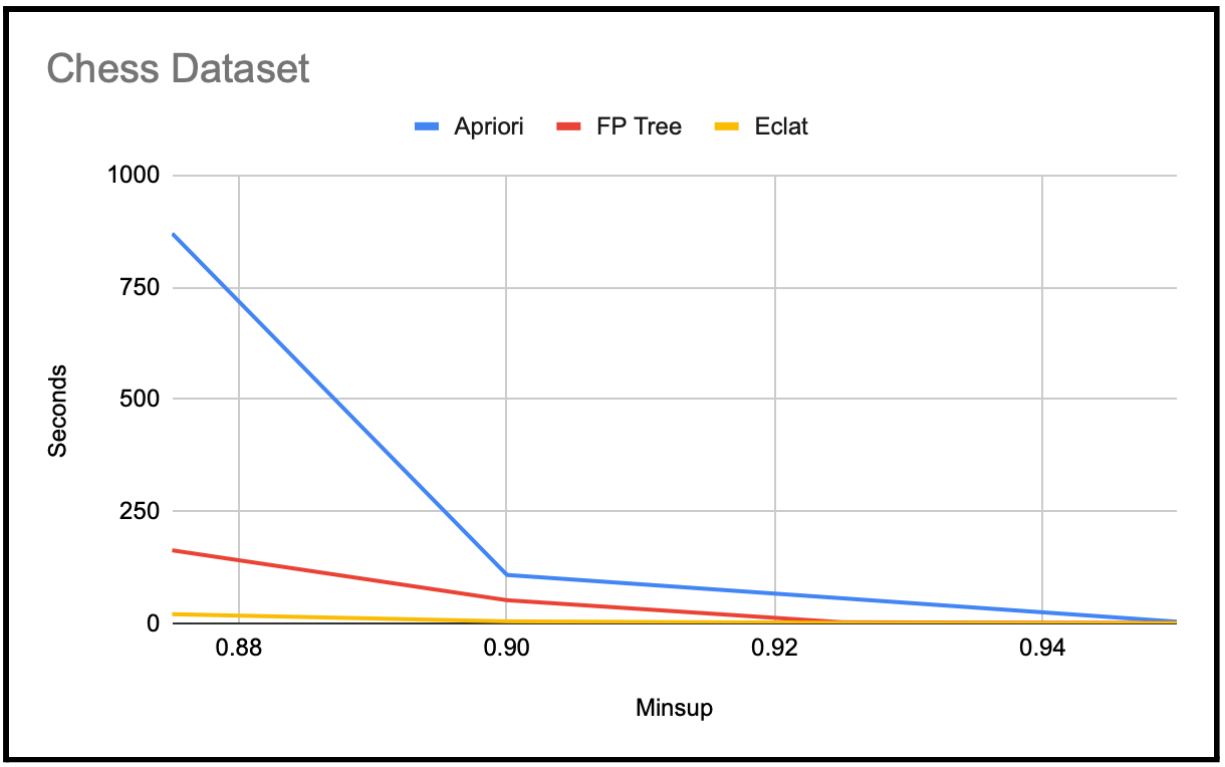
Minsup	Apriori	FP Tree	Eclat
0.01	33.5 sec	80 sec	19.90 sec
0.03	11.8 sec	20.86 sec	18.59 sec
0.05	7.31 sec	6.76 sec	18.63 sec
0.10	5.49 sec	0.79 sec	18.72 sec

Chess Dataset

Minsup	Apriori	FP Tree	Eclat
0.875	870 sec	163 sec	20.42 sec
0.9	108 sec	51.65 sec	4.52 sec
0.925	56 sec	2.47 sec	1.14 sec
0.95	3.7 sec	0.60 sec	0.4 sec


T20i6D100K Dataset
Apriori
FP Tree
Eclat
1000
750
500
250
0
Seconds
986.76
502.65
109.64
5.47
0.02
0.04
0.06
0.08
Minsup


Liquor Dataset
Apriori
FP Tree
Eclat
80
60
40
20
0
Seconds
0.02
0.04
0.06
0.08
0.10
Minsup



Dataset Information:

Properties	T20i6d100k	Liquor	Chess
# of Transaction	99,922	52131	3,196
Avg width of trans	19.90	7.88	37
Maximal Frequent itemset	[369]	[11788]	[59]
1-itemset support count ranges from	2 to 13903	1 to 17599	1 to 3195
Total # of items	893	4026	75

Algo	Apriori	FP Growth	Eclat
File Size	160kB	94kB	709kB

*File Size indicative of space complexity of the Algo

Conclusions:

With the above observation table it is clear that the FP growth algorithm and Eclatlooks fastest algorithms but it takes a amount of space. Whereas Apriori is near to brute force type of algorithm with effective pruning. Eclat using vertical transformation shows better performance than apriori. FP growth does not require candidate generations and requires only two scans of frequent item generation.

For the **T20 dataset**, which has a high number of transactions as compared to items, I will use the **Eclat** Algorithm which is most efficient here with respect to Space and Time.

For **Chess** Dataset: Which has a large number of items and average transaction size as compared to number of transactions. This makes Eclat algo less efficient on this dataset. So, FP **Growth** Algorithm seems to work better on this dataset.

For **Liquor** Dataset: This Dataset sufficiently has a high number of transactions and average transaction size, Eclat Algo seems to work better in this Dataset. I will use the Eclat **algorithm.**

References:

1. Data Mining and Analysis: Fundamental Concepts and Algorithms: Textbook by Mohammed J. Zaki and Wagner Meira
2. Introduction to Data Mining: Book by Michael Steinbach, Pang-Ning Tan, and Vipin Kumar
3. Google.com
4. Numpy and Pandas
5. Wikipedia