Report

Steps for building the recommendation system.

We first preprocessed the data, as asked in the problem statement.

First, we built the transactional dataset. For every user we mapped the movies rated by that particular user.

Then comes the mining part. We have two options Apriori or FP growth.Apriori is relatively easier to implement but it's slow whereas FPgrowth algorithm is difficult to implement but it's faster than apriori and uses less memory.

We need to find the frequent itemsets and then from frequent itemsets we need to find association rules that satisfy min_conf and min_sup

We then sorted it in descending order of their count(call it L)

Now every itemset we need to insert into the FP tree based on the L order.

For building the tree, we created a TreeNode class

```
class TreeNode:
    def __init__(self,id=-1,cnt=0):
        self.id=id
        self.cnt=cnt
        self.pointers={} #stores key id and value is reference to a node

    def check(self,node):
        if node not in self.pointers:
            return False
        return True
    def get(self,node):
        return self.pointers[node]
```

Also we defined the tree class that contains methods to insert an item list and other essential methods

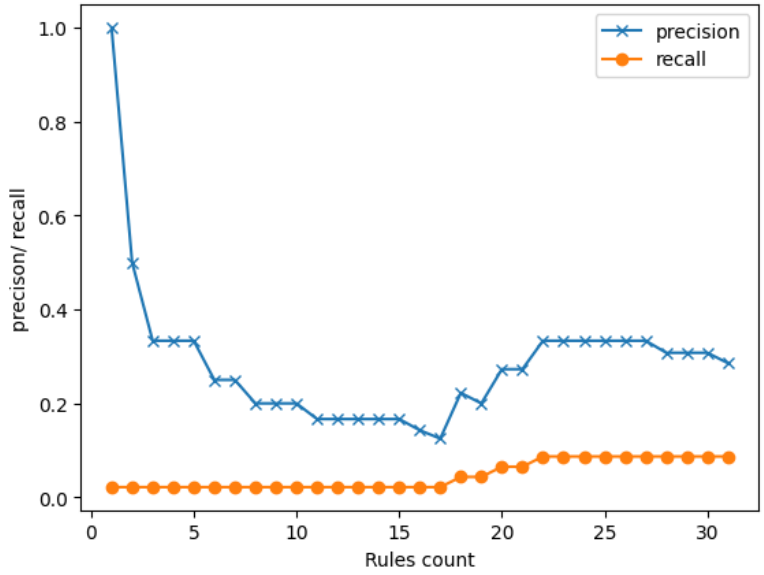
There are three main steps involved:

- We need to build the conditional pattern base
- Then using the conditional pattern base, we need to build the conditional FP-Tree
- Using the conditional FP-Tree we need to find the frequent patterns

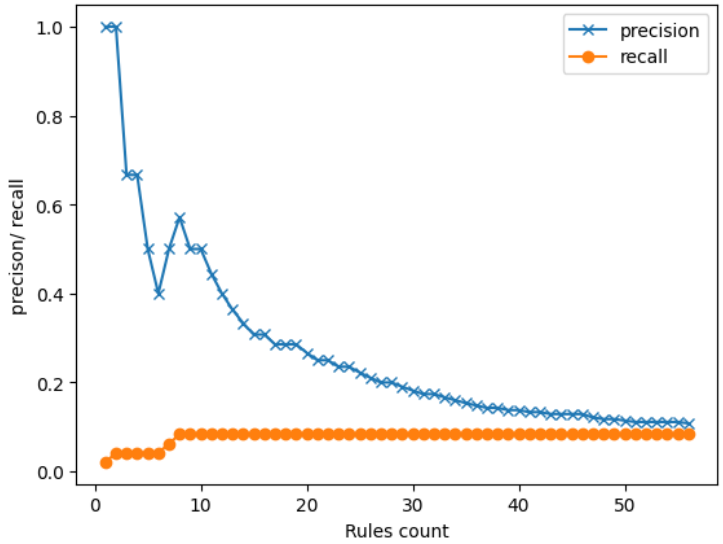
The FP-growth method transforms the problem of finding long frequent patterns into searching for shorter ones in much smaller conditional databases recursively and then concatenating with the suffix.

We then created two files one containing top 100 rules based on support and another one based on confidence (in descending order)

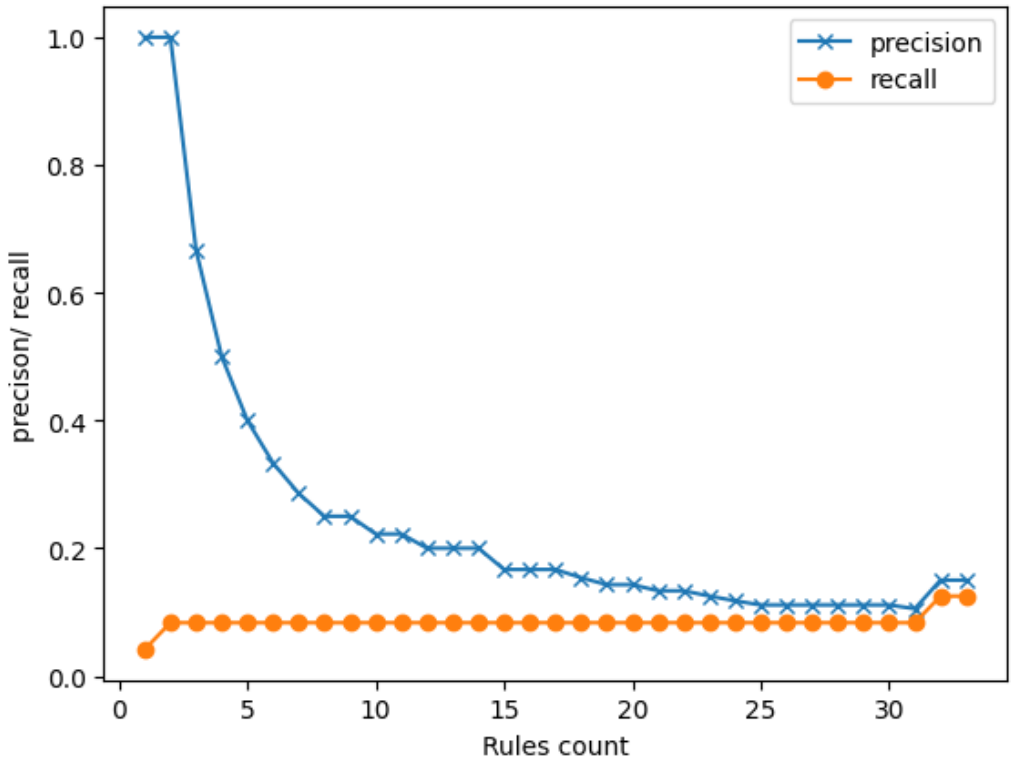
Then we plotted precision and recall graphs for 3 users.



The above is for user 1



The above is for user 50



The above is for user 10

The main observations from the above plots are:

- As the rules count increases precision decreases and recall increases slowly
- The decrease in precision with the number of rules in exponential whereas the recall increases very slowly.
- For higher number of rules precision and recall converge