AIM: Implementation of Association rule mining Using

1. Apriori Algorithm
2. FPTree

THEORY:

Association rule learning is a type of unsupervised learning technique that checks for the dependency of one data item on another data item and maps accordingly so that it can be more profitable. It tries to find some interesting relations or associations among the variables of the dataset. It is based on different rules to discover the interesting relations between variables in the database.

The association rule learning is one of the very important concepts of machine learning, and it is employed in Market Basket analysis, Web usage mining, continuous production, etc. Here market basket analysis is a technique used by the various big retailers to discover the associations between items. Association rules are created by thoroughly analysing data and looking for frequent if/then patterns. Then, depending on the following two parameters, the important relationships are observed:

1. Support: Support indicates how frequently the if/then relationship appears in the database.
2. Confidence: Confidence tells about the number of times these relationships have been found to be true.

CODE:

```
import numpy as np import
pandas as pd import
matplotlib.pyplot as plt
from apyori import apriori
store_data = pd.read_csv("store_data.csv", header=None)
#display(store_data.head()) print(store_data.shape) records =
[] for i in range(1, 7501):
records.append([str(store_data.values[i, j]) for j in range(0,
20)]) print(type(records))
```

```
x = float(input("Enter the minimum
support")) y = float(input("Enter the
minimum conidence"))
association_rules = apriori(records, min_support=x, min_confidence=y,
min_lift=3, min_length=2)
association_results = list(association_rules) print("There are {}
Relation derived.".format(len(association_results))) for i in
range(0, len(association_results)):
    print(association_results[i][0])

for item in association_results:
    # first index of the inner list
    # Contains base item and add item
    pair = item[0] items
    = [x for x in pair]
    print("Rule: " + items[0] + " -> " + items[1])

    # second index of the inner list
    print("Support: " + str(item[1]))

    # third index of the list located at 0th
    # of the third index of the inner list

    print("Confidence: " + str(item[2][0][2]))
    print("Lift: " + str(item[2][0][3]))
    print("====================================")
```

OUTPUT:

```
========== RESTART: E:/yash programming/NLP project work/test.py ==========
(7501, 20)
<class 'list'>
Enter the minimum support0.0045
Enter the minimum conidence0.2
There are 48 Relation derived.
frozenset({'chicken', 'light cream'})
frozenset({'mushroom cream sauce', 'escalope'})
frozenset({'pasta', 'escalope'})
frozenset({'herb & pepper', 'ground beef'})
frozenset({'tomato sauce', 'ground beef'})
frozenset({'whole wheat pasta', 'olive oil'})
frozenset({'shrimp', 'pasta'})
frozenset({'chicken', 'light cream', 'nan'})
frozenset({'shrimp', 'chocolate', 'frozen vegetables'})
frozenset({'spaghetti', 'cooking oil', 'ground beef'})
frozenset({'nan', 'mushroom cream sauce', 'escalope'})
frozenset({'nan', 'pasta', 'escalope'})
frozenset({'spaghetti', 'frozen vegetables', 'ground beef'})
frozenset({'milk', 'frozen vegetables', 'olive oil'})
frozenset({'shrimp', 'mineral water', 'frozen vegetables'})
frozenset({'spaghetti', 'frozen vegetables', 'olive oil'})
frozenset({'shrimp', 'spaghetti', 'frozen vegetables'})
frozenset({'spaghetti', 'frozen vegetables', 'tomatoes'})
frozenset({'spaghetti', 'grated cheese', 'ground beef'})
frozenset({'mineral water', 'herb & pepper', 'ground beef'})
frozenset({'nan', 'herb & pepper', 'ground beef'})
frozenset({'spaghetti', 'herb & pepper', 'ground beef'})
frozenset({'milk', 'olive oil', 'ground beef'})
frozenset({'tomato sauce', 'nan', 'ground beef'})
frozenset({'shrimp', 'spaghetti', 'ground beef'})
frozenset({'spaghetti', 'milk', 'olive oil'})
frozenset({'mineral water', 'olive oil', 'soup'})
frozenset({'whole wheat pasta', 'olive oil', 'nan'})
frozenset({'shrimp', 'pasta', 'nan'})
frozenset({'spaghetti', 'pancakes', 'olive oil'})
frozenset({'shrimp', 'chocolate', 'frozen vegetables', 'nan'})
frozenset({'spaghetti', 'nan', 'cooking oil', 'ground beef'})
frozenset({'spaghetti', 'nan', 'frozen vegetables', 'ground beef'})
frozenset({'spaghetti', 'milk', 'mineral water', 'frozen vegetables'})
```

```
frozenset({'spaghetti', 'milk', 'olive oil', 'nan'})
frozenset({'mineral water', 'olive oil', 'soup', 'nan'})
frozenset({'spaghetti', 'pancakes', 'olive oil', 'nan'})
frozenset({'spaghetti', 'mineral water', 'nan', 'frozen vegetables', 'milk'})
Rule: chicken -> light cream
Support: 0.004533333333333334
Confidence: 0.2905982905982906
Lift: 4.843304843304844
====================================
Rule: mushroom cream sauce -> escalope
Support: 0.005733333333333333
Confidence: 0.30069930069930073
Lift: 3.7903273197390845
====================================
Rule: pasta -> escalope
Support: 0.005866666666666667
Confidence: 0.37288135593220345
Lift: 4.700185158809287
====================================
Rule: herb & pepper -> ground beef
Support: 0.016
Confidence: 0.3234501347708895
Lift: 3.2915549671393096
====================================
Rule: tomato sauce -> ground beef
Support: 0.005333333333333333
Confidence: 0.37735849056603776
Lift: 3.840147461662528
====================================
Rule: whole wheat pasta -> olive oil
Support: 0.008
Confidence: 0.2714932126696833
Lift: 4.130221288078346
====================================
Rule: shrimp -> pasta
Support: 0.005066666666666666
Confidence: 0.3220338983050848
Lift: 4.514493901473151
====================================
Rule: chicken -> light cream
```

FP TREE :

CODE:

```
import pandas as pd
from mlxtend.preprocessing import TransactionEncoder
dataset = [ ['f', 'a', 'c', 'd', 'g', 'i', 'm', 'p'],
        ['a', 'b', 'c', 'f', 'l', 'm', 'o'],
        ['b', 'f', 'h', 'j', 'o', 'w' ],
        ['b','c', 'k', 's', 'p'],
        ['a', 'f', 'c', 'e', 'l', 'p', 'm', 'n'],]
```

```
te = TransactionEncoder() te_ary = te.fit(dataset).transform(dataset) df = pd.DataFrame(te_ary, columns=te.columns_)
# print(df) from mlxtend.frequent_patterns import fpgrowth

# print(fpgrowth(df, min_support=0.6)) print(fpgrowth(df, min_support=0.6, use_colnames=True))
```

OUTPUT:

```
============== RESTART: E:/yash programmir
    support      itemsets
0       0.8           (f)
1       0.8           (c)
2       0.6           (p)
3       0.6           (m)
4       0.6           (a)
5       0.6           (b)
6       0.6        (c, f)
7       0.6        (c, p)
8       0.6        (m, c)
9       0.6        (m, f)
10      0.6     (m, c, f)
11      0.6        (a, m)
12      0.6        (a, c)
13      0.6        (a, f)
14      0.6     (a, m, c)
15      0.6     (a, m, f)
16      0.6     (a, c, f)
17      0.6  (a, m, c, f)
>>>
```

CONCLUSION:

We learned to implement association rules using two different algorithms that can be used - Apriori and FP Tree. We implemented them using python and established association rules.