Import neccessery libraries

In [1]:

```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
from apyori import apriori as apr
from mlxtend.frequent_patterns import apriori,association_rules
from mlxtend.preprocessing import TransactionEncoder
from scipy.special import comb
import scipy as sp
from mpl_toolkits.mplot3d import Axes3D
import seaborn as sns
from itertools import combinations,permutations
```

Problem

Prepare rules for the all the data sets

1) Try different values of support and confidence. Observe the change in number of rules for different support,confidence values

2) Change the minimum length in apriori algorithm

3) Visulize the obtained rules using different plots

Import data

In [33]:

```
movie_data= pd.read_csv('my_movies.csv')
```

In [34]:

```
movie_data.head()
```

Out[34]:

	V1	V2	V3	V4	V5	Sixth Sense	Gladiator	LOTR1	Harry Potter1	Patriot	LOTR2
0	Sixth Sense	LOTR1	Harry Potter1	Green Mile	LOTR2	1	0	1	1	0	1
1	Gladiator	Patriot	Braveheart	NaN	NaN	0	1	0	0	1	0
2	LOTR1	LOTR2	NaN	NaN	NaN	0	0	1	0	0	1
3	Gladiator	Patriot	Sixth Sense	NaN	NaN	1	1	0	0	1	0
4	Gladiator	Patriot	Sixth Sense	NaN	NaN	1	1	0	0	1	0

In [54]:

```
movie_data1 = movie_data.iloc[:,5:]
```

In [61]:

```
movie_data1.head()
```

Out[61]:

	Sixth Sense	Gladiator	LOTR1	Harry Potter1	Patriot	LOTR2	Harry Potter2	LOTR	Braveheart	Green Mile
0	1	0	1	1	0	1	0	0	0	1
1	0	1	0	0	1	0	0	0	1	0
2	0	0	1	0	0	1	0	0	0	0
3	1	1	0	0	1	0	0	0	0	0
4	1	1	0	0	1	0	0	0	0	0

In [62]:

```
movie_data1.isna().sum()
```

Out[62]:

```
Sixth Sense      0
Gladiator        0
LOTR1            0
Harry Potter1    0
Patriot          0
LOTR2            0
Harry Potter2    0
LOTR             0
Braveheart       0
Green Mile       0
dtype: int64
```

In [63]:

```
movie_data.dtypes
```

Out[63]:

```
V1               object
V2               object
V3               object
V4               object
V5               object
Sixth Sense       int64
Gladiator         int64
LOTR1             int64
Harry Potter1     int64
Patriot           int64
LOTR2             int64
Harry Potter2     int64
LOTR              int64
Braveheart        int64
Green Mile        int64
dtype: object
```

In [64]:

```
movie_data.astype
```

Out[64]:

```
<bound method NDFrame.astype of               V1             V2
V3           V4      V5  \
0    Sixth Sense          LOTR1  Harry Potter1  Green Mile  LOTR2
1      Gladiator        Patriot     Braveheart         NaN    NaN
2          LOTR1          LOTR2            NaN         NaN    NaN
3      Gladiator        Patriot    Sixth Sense         NaN    NaN
4      Gladiator        Patriot    Sixth Sense         NaN    NaN
5      Gladiator        Patriot    Sixth Sense         NaN    NaN
6  Harry Potter1  Harry Potter2            NaN         NaN    NaN
7      Gladiator        Patriot            NaN         NaN    NaN
8      Gladiator        Patriot    Sixth Sense         NaN    NaN
9    Sixth Sense           LOTR      Gladiator  Green Mile    NaN

   Sixth Sense  Gladiator  LOTR1  Harry Potter1  Patriot  LOTR2  \
0            1          0      1              1        0       1
1            0          1      0              0        1       0
2            0          0      1              0        0       1
3            1          1      0              0        1       0
4            1          1      0              0        1       0
5            1          1      0              0        1       0
6            0          0      0              1        0       0
7            0          1      0              0        1       0
8            1          1      0              0        1       0
9            1          1      0              0        0       0

   Harry Potter2  LOTR  Braveheart  Green Mile
0              0     0           0           1
1              0     0           1           0
2              0     0           0           0
3              0     0           0           0
4              0     0           0           0
5              0     0           0           0
6              1     0           0           0
7              0     0           0           0
8              0     0           0           0
```

In [65]:

```
movie_data.describe().T
```

Out[65]:

	count	mean	std	min	25%	50%	75%	max
Sixth Sense	10.0	0.6	0.516398	0.0	0.00	1.0	1.0	1.0
Gladiator	10.0	0.7	0.483046	0.0	0.25	1.0	1.0	1.0
LOTR1	10.0	0.2	0.421637	0.0	0.00	0.0	0.0	1.0
Harry Potter1	10.0	0.2	0.421637	0.0	0.00	0.0	0.0	1.0
Patriot	10.0	0.6	0.516398	0.0	0.00	1.0	1.0	1.0
LOTR2	10.0	0.2	0.421637	0.0	0.00	0.0	0.0	1.0
Harry Potter2	10.0	0.1	0.316228	0.0	0.00	0.0	0.0	1.0
LOTR	10.0	0.1	0.316228	0.0	0.00	0.0	0.0	1.0
Braveheart	10.0	0.1	0.316228	0.0	0.00	0.0	0.0	1.0
Green Mile	10.0	0.2	0.421637	0.0	0.00	0.0	0.0	1.0

In [66]:

```
item_sets = {}
```

In [68]:

```
tr = TransactionEncoder()
tr_model = tr.fit(movie_data1).transform(movie_data1)
```

In [69]:

```
tr_model
```

Out[69]:

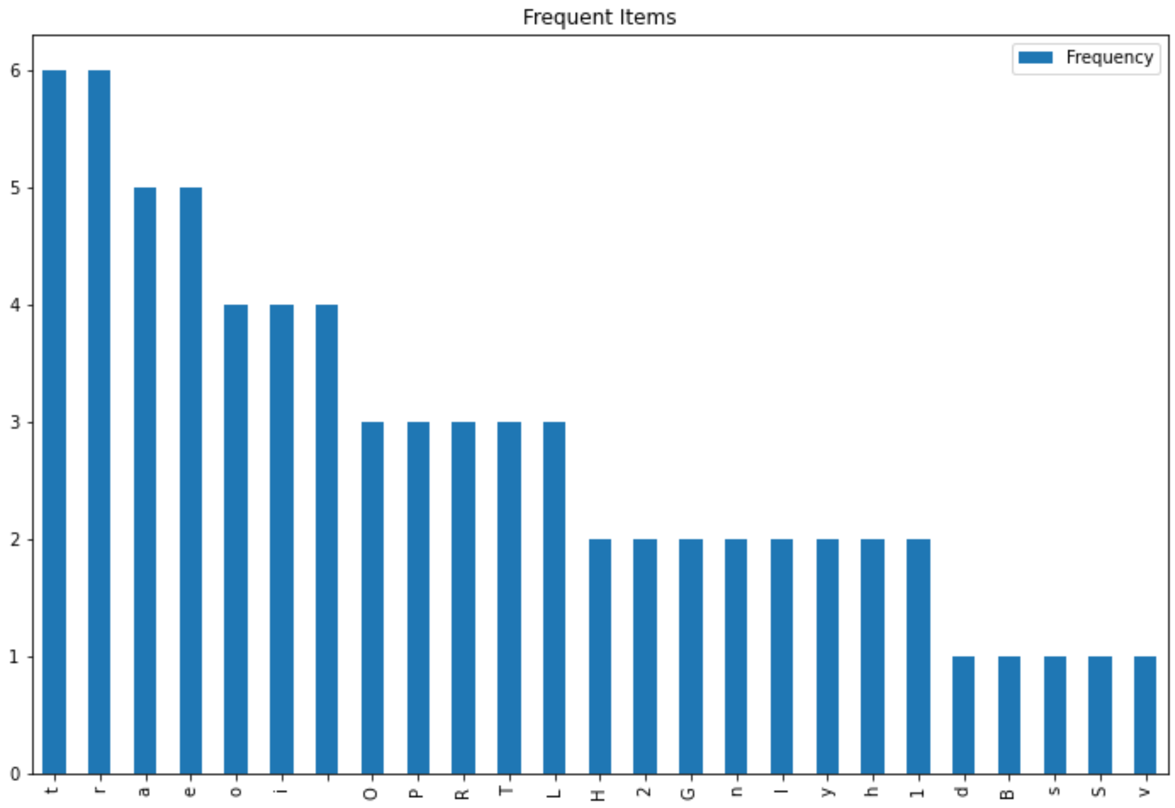
```
array([[ True, False, False, False, False, False, False, False, False,
        False, False,  True, False, False, False,  True,  True,  True,
        False,  True, False, False,  True,  True, False,  True, False],
       [False, False, False, False,  True, False, False, False, False,
        False, False, False, False,  True,  True, False, False,  True,
         True, False,  True,  True, False,  True, False, False, False],
       [False,  True, False, False, False, False,  True, False,  True,
        False,  True, False,  True, False, False, False, False, False,
        False, False, False, False, False, False, False, False, False],
       [ True,  True, False, False, False,  True, False, False, False,
         True, False, False, False,  True, False,  True, False, False,
        False, False,  True,  True, False,  True, False, False,  True],
       [False, False, False, False, False, False, False, False, False,
         True, False, False, False,  True, False, False, False,  True,
        False, False,  True,  True, False,  True, False, False, False],
       [False, False,  True, False, False, False,  True, False,  True,
        False,  True, False,  True, False, False, False, False, False,
        False, False, False, False, False, False, False, False, False],
       [ True, False,  True, False, False,  True, False, False, False,
         True, False, False, False,  True, False,  True, False, False,
        False, False,  True,  True, False,  True, False, False,  True],
       [False, False, False, False, False, False,  True, False,  True,
        False,  True, False,  True, False, False, False, False, False,
        False, False, False, False, False, False, False, False, False],
       [False, False, False,  True, False, False, False, False, False,
        False, False, False, False,  True, False,  True,  True, False,
        False, False, False,  True, False,  True,  True, False, False],
       [ True, False, False, False,  True, False, False,  True, False,
        False, False, False, False, False, False,  True, False,  True,
```

In [70]:

```
ap = pd.DataFrame(tr_model,columns=tr.columns_)
```

In [71]:

```
ap.sum().to_frame('Frequency').sort_values('Frequency',ascending=False)[:25
                                                      figsize=(12,8),title=
plt.show()
```



Apriori algorithm

In [74]:

```
ap_0_5 = {}
ap_1 = {}
ap_5 = {}
ap_1_0 = {}
```

In [75]:

```
confidence = [0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9]
```

In [81]:

```
def gen_rules(df,confidence,support):
    ap = {}
    for i in confidence:
        ap_i =apriori(movie_data1,support,True)
        rule= association_rules(ap_i,min_threshold=i)
        ap[i] = len(rule.antecedents)
    return pd.Series(ap).to_frame("Support: %s"%support)
```

In [82]:

```
confs = []
```

In [83]:

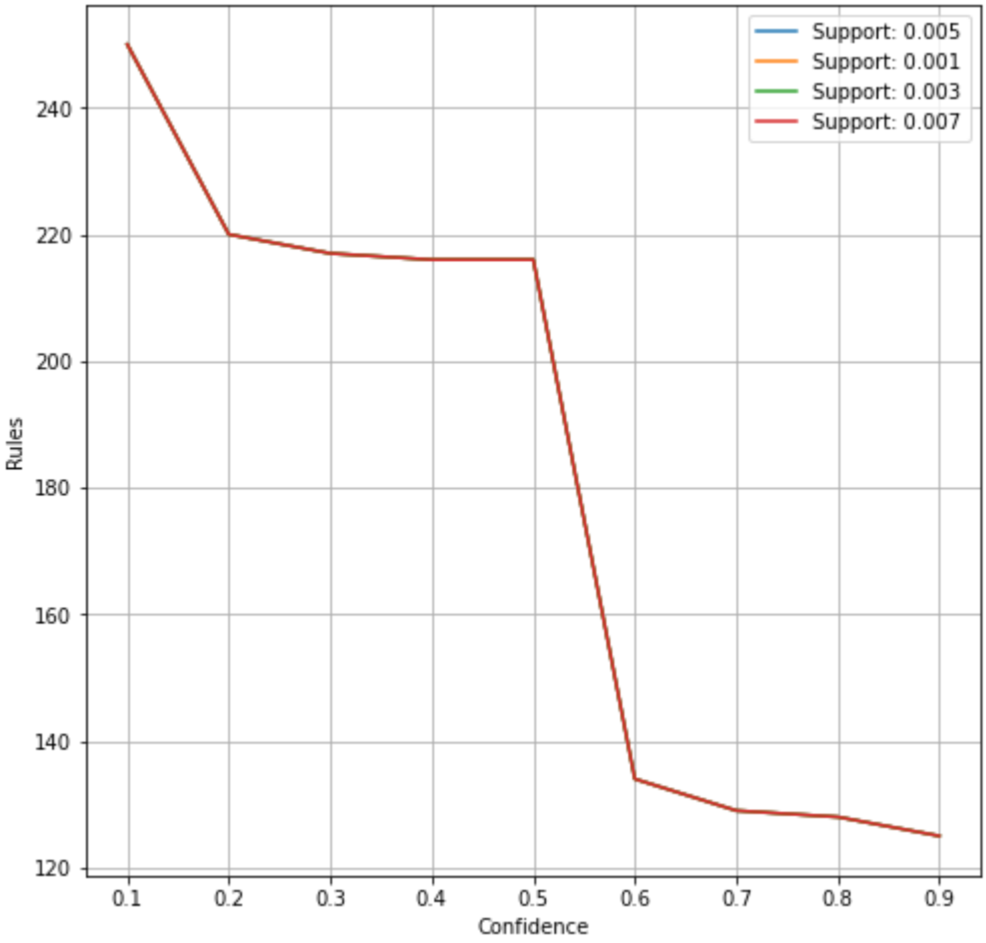
```
for i in [0.005,0.001,0.003,0.007]:
    ap_i = gen_rules(ap,confidence=confidence,support=i)
    confs.append(ap_i)
```

In [84]:

```
all_conf = pd.concat(confs,axis=1)
```

In [85]:

```
all_conf.plot(figsize=(8,8),grid=True)
plt.ylabel('Rules')
plt.xlabel('Confidence')
plt.show()
```



As shown in above graph

1.Lower the Confidence level Higher the no. of rules.

2.Higher the Support, lower the no. of rules.

In [73]:
```
frequent_items = apriori(df = movie_data1,min_support=0.03,use_colnames=Tru
frequent_items
```

Out[73]:

	support	itemsets
0	0.6	(Sixth Sense)
1	0.7	(Gladiator)
2	0.2	(LOTR1)
3	0.2	(Harry Potter1)
4	0.6	(Patriot)
5	0.2	(LOTR2)
6	0.1	(Harry Potter2)
7	0.1	(LOTR)
8	0.1	(Braveheart)
9	0.2	(Green Mile)
10	0.5	(Sixth Sense, Gladiator)

	support	itemsets
11	0.1	(Sixth Sense, LOTR1)
12	0.1	(Sixth Sense, Harry Potter1)
13	0.4	(Patriot, Sixth Sense)
14	0.1	(Sixth Sense, LOTR2)
15	0.1	(Sixth Sense, LOTR)
16	0.2	(Green Mile, Sixth Sense)
17	0.6	(Patriot, Gladiator)
18	0.1	(Gladiator, LOTR)
19	0.1	(Gladiator, Braveheart)
20	0.1	(Green Mile, Gladiator)
21	0.1	(LOTR1, Harry Potter1)
22	0.2	(LOTR2, LOTR1)
23	0.1	(Green Mile, LOTR1)
24	0.1	(LOTR2, Harry Potter1)
25	0.1	(Harry Potter1, Harry Potter2)
26	0.1	(Green Mile, Harry Potter1)
27	0.1	(Patriot, Braveheart)
28	0.1	(Green Mile, LOTR2)
29	0.1	(Green Mile, LOTR)
30	0.4	(Patriot, Sixth Sense, Gladiator)
31	0.1	(Sixth Sense, Gladiator, LOTR)
32	0.1	(Green Mile, Sixth Sense, Gladiator)
33	0.1	(Sixth Sense, LOTR1, Harry Potter1)
34	0.1	(LOTR2, Sixth Sense, LOTR1)
35	0.1	(Green Mile, Sixth Sense, LOTR1)
36	0.1	(Sixth Sense, LOTR2, Harry Potter1)
37	0.1	(Green Mile, Sixth Sense, Harry Potter1)
38	0.1	(Green Mile, Sixth Sense, LOTR2)
39	0.1	(Green Mile, Sixth Sense, LOTR)
40	0.1	(Patriot, Gladiator, Braveheart)
41	0.1	(Green Mile, Gladiator, LOTR)
42	0.1	(LOTR2, LOTR1, Harry Potter1)
43	0.1	(Green Mile, LOTR1, Harry Potter1)
44	0.1	(Green Mile, LOTR1, LOTR2)
45	0.1	(Green Mile, LOTR2, Harry Potter1)

	support	itemsets
46	0.1	(Green Mile, Sixth Sense, Gladiator, LOTR)
47	0.1	(LOTR2, Sixth Sense, LOTR1, Harry Potter1)
48	0.1	(Green Mile, Sixth Sense, LOTR1, Harry Potter1)
49	0.1	(Green Mile, Sixth Sense, LOTR1, LOTR2)

In [80]:

```
best_associates = association_rules(df = frequent_items,metric='lift',min_t
best_associates
```

Out[80]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	con
0	(LOTR1)	(Harry Potter1)	0.2	0.2	0.1	0.5	2.5	0.06	
1	(Harry Potter1)	(LOTR1)	0.2	0.2	0.1	0.5	2.5	0.06	
2	(LOTR2)	(LOTR1)	0.2	0.2	0.2	1.0	5.0	0.16	
3	(LOTR1)	(LOTR2)	0.2	0.2	0.2	1.0	5.0	0.16	
4	(Green Mile)	(LOTR1)	0.2	0.2	0.1	0.5	2.5	0.06	
...	...	...	...	...	...	...	...	...	
169	(Green Mile, Sixth Sense)	(Harry Potter1, LOTR2, LOTR1)	0.2	0.1	0.1	0.5	5.0	0.08	
170	(LOTR2)	(Green Mile, Sixth Sense, LOTR1, Harry Potter1)	0.2	0.1	0.1	0.5	5.0	0.08	
171	(LOTR1)	(Green Mile, Sixth Sense, LOTR2, Harry Potter1)	0.2	0.1	0.1	0.5	5.0	0.08	
172	(Harry Potter1)	(Green Mile, Sixth Sense, LOTR2, LOTR1)	0.2	0.1	0.1	0.5	5.0	0.08	
173	(Green Mile)	(Sixth Sense, Harry Potter1, LOTR2, LOTR1)	0.2	0.1	0.1	0.5	5.0	0.08	

174 rows × 9 columns

In [89]:

```
best_associates.shape
```

Out[89]:

```
(174, 9)
```

Lets try with Support 0.007 and Confidence at 0.7

In [107…

```
ap_final = apriori(ap,0.007,True)
```

In [108…

```
rules_final = association_rules(ap_final,min_threshold=.7,support_only=Fals
```

In [109…

```
rules_final[rules_final['confidence']>0.7]
```

Out[109…

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	c
0	(H)	( )	0.2	0.4	0.2	1.0	2.5	0.12	
1	(M)	( )	0.1	0.4	0.1	1.0	2.5	0.06	
2	(S)	( )	0.1	0.4	0.1	1.0	2.5	0.06	
3	(e)	( )	0.5	0.4	0.4	0.8	2.0	0.20	
4	( )	(e)	0.4	0.5	0.4	1.0	2.0	0.20	
...	...	...	...	...	...	...	...	...	
75755	(2, H)	(r, o, y, a, , e, P, t)	0.1	0.2	0.1	1.0	5.0	0.08	
75756	(2,)	(r, o, y, a, H, e, P, t)	0.1	0.2	0.1	1.0	5.0	0.08	
75757	(2, e)	(r, o, y, a, H, , P, t)	0.1	0.2	0.1	1.0	5.0	0.08	
75758	(2, P)	(r, o, y, a, H, , e, t)	0.1	0.2	0.1	1.0	5.0	0.08	
75759	(2, t)	(r, o, y, a, H, , e, P)	0.1	0.2	0.1	1.0	5.0	0.08	

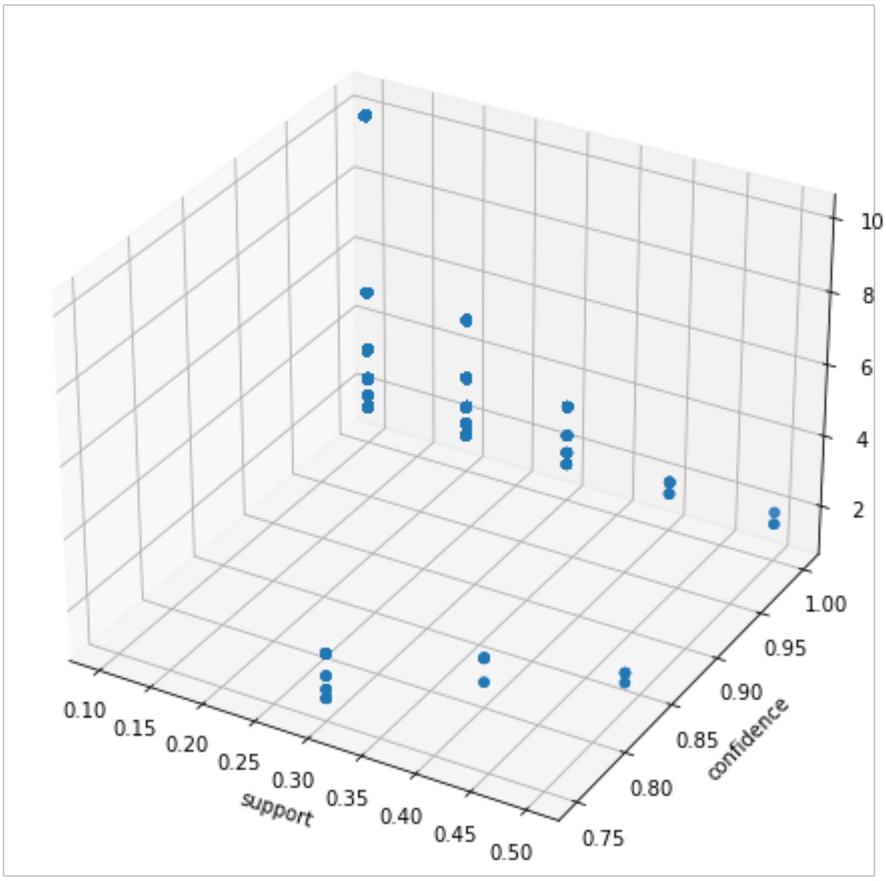
75760 rows × 9 columns

In [110…

```
support = rules_final["support"]
confidence =  rules_final["confidence"]
lift = rules_final["lift"]
```

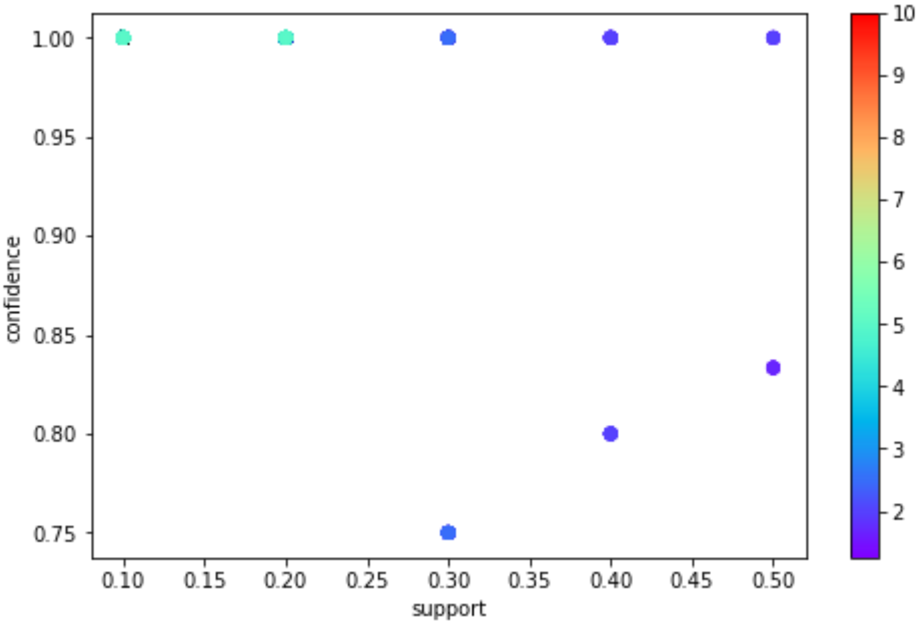
In [113…

```
fig1 = plt.figure(figsize=(8,8))
ax1 = fig1.add_subplot(111, projection = '3d')
ax1.scatter(support,confidence,lift)
ax1.set_xlabel("support")
ax1.set_ylabel("confidence")
ax1.set_zlabel("lift")
plt.show()
```



In [115…

```
fig1 = plt.figure(figsize=(8,5))
plt.scatter(support,confidence, c =lift, cmap = 'rainbow')
plt.colorbar()
plt.xlabel("support");plt.ylabel("confidence")
plt.show()
```



In [ ]:

In [ ]: