Name: Ramakrishna Raju

Batch: 09012021

ASSOCIATION RULES

Problem Statement: -

Kitabi Duniya , a famous book store in India, which was established before Independence, the growth of the company was incremental year by year, but due to online selling of books and wide spread Internet access its annual growth started to collapse, seeing sharp downfalls, you as a Data Scientist help this heritage book store gain its popularity back and increase footfall of customers and provide ways the business can improve exponentially, apply Association Rule Algorithm, explain the rules, and visualize the graphs for clear understanding of solution.

1.) Books.csv

In [1]:

```
import pandas as pd
from mlxtend.frequent_patterns import apriori, association_rules
```

In [2]:

```
book = pd.read_csv(r"F:\360\associationrules\book.csv")
```

Finding the frequent items

In [3]:

```
frequent_itemsets = apriori(book,min_support=0.005, max_len=3,use_colnames = True)
frequent_itemsets.shape
```

Out[3]:

```
(224, 2)
```

Most Frequent item sets based on support

In [4]:

```
frequent_itemsets.sort_values('support',ascending = False,inplace=True)
```

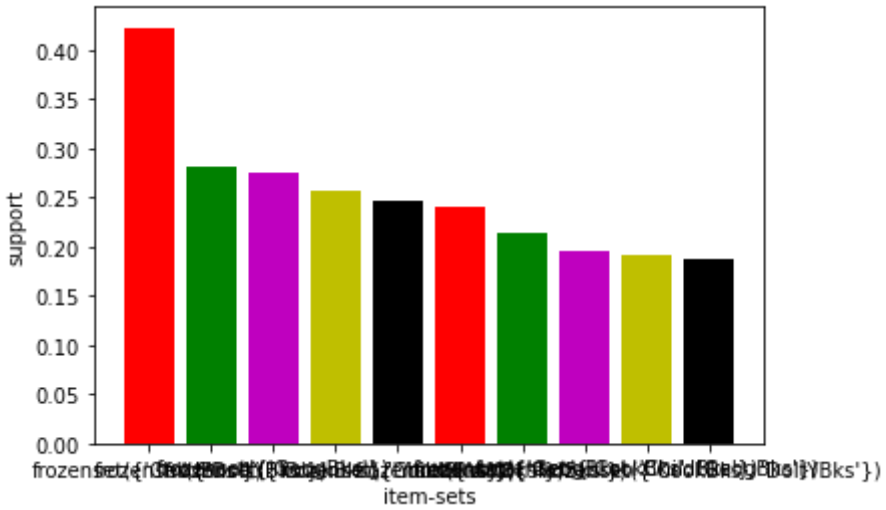
In [5]:

```
import matplotlib.pylab as plt
plt.bar(x = list(range(1,11)),height = frequent_itemsets.support[1:11],color='rgmyk');
plt.xticks(list(range(1,11)),frequent_itemsets.itemsets[1:11]);
plt.xlabel('item-sets');
plt.ylabel('support')
```

```
<ipython-input-5-48e44dd25921>:2: MatplotlibDeprecationWarning: Using a stri
ng of single character colors as a color sequence is deprecated since 3.2 an
d will be removed two minor releases later. Use an explicit list instead.
  plt.bar(x = list(range(1,11)),height = frequent_itemsets.support[1:11],col
or='rgmyk');
```

Out[5]:

```
Text(0, 0.5, 'support')
```



In [6]:

```
rules = association_rules(frequent_itemsets, metric="lift", min_threshold=1)
rules.shape
```

Out[6]:

```
(1054, 9)
```

In [7]:

```
rules.head(20)
rules.sort_values('lift',ascending = False,inplace=True)
```

To eliminate Redudancy in Rules

In [8]:

```
def to_list(i):
    return (sorted(list(i)))
```

In [9]:

```
ma_X = rules.antecedents.apply(to_list)+rules.consequents.apply(to_list)
```

In [10]:

```
ma_X = ma_X.apply(sorted)
```

In [11]:

```
rules_sets = list(ma_X)
```

In [12]:

```
unique_rules_sets = [list(m) for m in set(tuple(i) for i in rules_sets)]
index_rules = []
for i in unique_rules_sets:
    index_rules.append(rules_sets.index(i))
```

getting rules without any redudancy

In [13]:

```
rules_no_redudancy  = rules.iloc[index_rules,:]
```

In [14]:

```
# Sorting them with respect to list and getting top 10 rules
```

In [15]:

```
rules_no_redudancy.sort_values('lift',ascending=False).head(10)
```

Out[15]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leve
749	(RefBks, ItalArt)	(ItalAtlas)	0.0200	0.0370	0.0165	0.825000	22.297297	0.01
746	(ItalArt)	(ItalAtlas, ArtBks)	0.0485	0.0180	0.0165	0.340206	18.900344	0.01
351	(ArtBks, ItalCook)	(ItalArt)	0.0565	0.0485	0.0375	0.663717	13.684883	0.03
561	(RefBks, ItalCook)	(ItalAtlas)	0.0465	0.0370	0.0230	0.494624	13.368207	0.02
924	(ItalArt)	(ItalAtlas, GeogBks)	0.0485	0.0205	0.0115	0.237113	11.566507	0.01
900	(ItalArt)	(ItalAtlas, ItalCook)	0.0485	0.0230	0.0125	0.257732	11.205737	0.01
966	(ItalArt)	(Florence, ItalCook)	0.0485	0.0175	0.0095	0.195876	11.192931	0.00
812	(ItalArt, ChildBks)	(ItalAtlas)	0.0360	0.0370	0.0145	0.402778	10.885886	0.01
973	(ItalArt)	(DoItYBks, ItalAtlas)	0.0485	0.0190	0.0095	0.195876	10.309278	0.00
1004	(YouthBks, ItalAtlas)	(ItalArt)	0.0175	0.0485	0.0085	0.485714	10.014728	0.00

In [16]:

```
# Perform algorithm for different support, connfidence value and max length
```

In [17]:

```
frequent_itemsets1 = apriori(book, min_support=0.007, max_len=4,use_colnames = True)
```

In [18]:

```
# Most Frequent item sets based on support
```

In [19]:

```
frequent_itemsets1.sort_values('support',ascending = False,inplace=True)
```

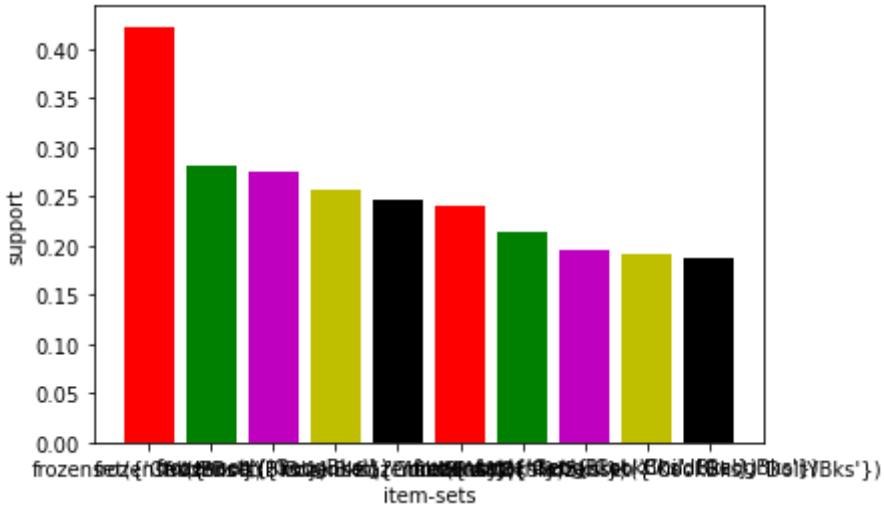
In [20]:

```
plt.bar(x = list(range(1,11)),height = frequent_itemsets1.support[1:11],color='rgmyk');
plt.xticks(list(range(1,11)),frequent_itemsets1.itemsets[1:11]);
plt.xlabel('item-sets');
plt.ylabel('support')
```

```
<ipython-input-20-298a32eca15c>:1: MatplotlibDeprecationWarning: Using a str
ing of single character colors as a color sequence is deprecated since 3.2 a
nd will be removed two minor releases later. Use an explicit list instead.
  plt.bar(x = list(range(1,11)),height = frequent_itemsets1.support[1:11],co
lor='rgmyk');
```

Out[20]:

```
Text(0, 0.5, 'support')
```



In [21]:

```
rules1 = association_rules(frequent_itemsets1, metric="lift", min_threshold=1)
rules1.head(20)
rules1.sort_values('lift',ascending = False,inplace=True)
```

In [22]:

```
frequent_itemsets2 = apriori(book, min_support=0.009, max_len=5,use_colnames = True)
```

In [23]:

```
# Most Frequent item sets based on support
```

In [24]:

```
frequent_itemsets2.sort_values('support',ascending = False,inplace=True)
```

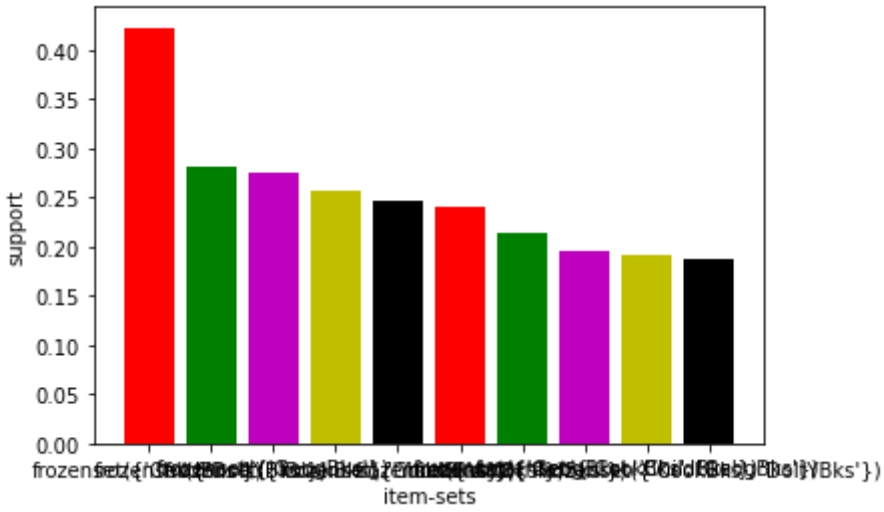
In [25]:

```
plt.bar(x = list(range(1,11)),height = frequent_itemsets2.support[1:11],color='rgmyk');
plt.xticks(list(range(1,11)),frequent_itemsets2.itemsets[1:11]);
plt.xlabel('item-sets');
plt.ylabel('support')
```

```
<ipython-input-25-ecb149ec78ec>:1: MatplotlibDeprecationWarning: Using a str
ing of single character colors as a color sequence is deprecated since 3.2 a
nd will be removed two minor releases later. Use an explicit list instead.
  plt.bar(x = list(range(1,11)),height = frequent_itemsets2.support[1:11],co
lor='rgmyk');
```

Out[25]:

```
Text(0, 0.5, 'support')
```



In [26]:

```
rules2 = association_rules(frequent_itemsets2, metric="lift", min_threshold=1)
rules2.head(20)
rules2.sort_values('lift',ascending = False,inplace=True)
```

As min lenth value is changing the rules is changing.

#rules =1054 #rules1=4556 #rules2=9164