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.

Prepare rules for the data set "Groceries.csv"

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) Visualize the obtained rules using different plots

In [11]:

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

In [ ]:

```

```

Importing Data Sets

In [2]:

```
groceries=[]
with open(r"F:\360\associationrules\groceries.csv") as f:
    groceries = f.read()
groceries = groceries.split("\n")
```

splitting the data into separate transactions

In [3]:

```
groceries_list = []
for i in groceries:
    groceries_list.append(i.split(","))
```

In [4]:

```
all_groceries_list = [i for item in groceries_list for i in item]
```

In [5]:

```
from collections import Counter
```

In [6]:

```
item_frequencies = Counter(all_groceries_list)
```

after sorting

In [7]:

```
item_frequencies = sorted(item_frequencies.items(), key = lambda x:x[1])
```

Storing frequencies and items in separate variables

In [8]:

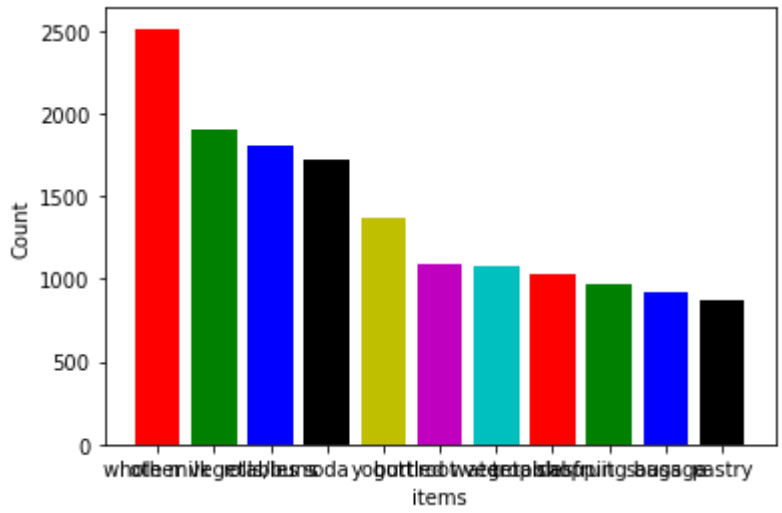
```
frequencies = list(reversed([i[1] for i in item_frequencies]))
items = list(reversed([i[0] for i in item_frequencies]))
```

barplot of top 10

In [9]:

```
import matplotlib.pyplot as plt
plt.bar(height = frequencies[0:11], x = list(range(0, 11)), color = 'rgbkymc')
plt.xticks(list(range(0, 11), ), items[0:11])
plt.xlabel("items")
plt.ylabel("Count")
plt.show()
```

```
<ipython-input-9-1a6c79c420bc>: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(height = frequencies[0:11], x = list(range(0, 11)), color = 'rgbky
mc')
```



Creating Data Frame for the transactions data

In [12]:

```
groceries_series = pd.DataFrame(pd.Series(groceries_list))
groceries_series = groceries_series.iloc[:9835, :]
```

removing the last empty transaction

In [13]:

```
groceries_series.columns = ["transactions"]
```

creating a dummy columns for the each item in each transactions ... Using column names as item name

In [14]:

```
X = groceries_series['transactions'].str.join(sep = '*').str.get_dummies(sep = '*')
frequent_itemsets = apriori(X, min_support = 0.0075, max_len = 4, use_colnames = True)
```

Most Frequent item sets based on support

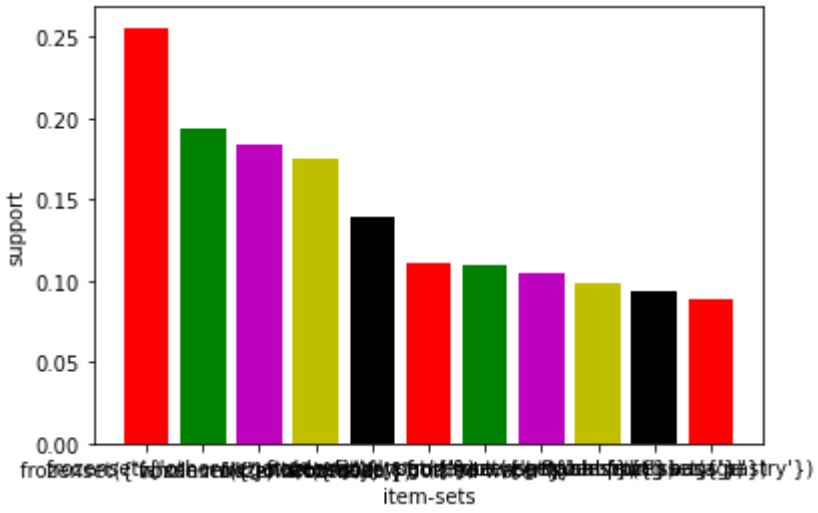
In [15]:

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

In [16]:

```
plt.bar(x = list(range(0, 11)), height = frequent_itemsets.support[0:11], color ='rgmyk')
plt.xticks(list(range(0, 11)), frequent_itemsets.itemsets[0:11])
plt.xlabel('item-sets')
plt.ylabel('support')
plt.show()
```

```
<ipython-input-16-3f6c91c30844>: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(0, 11)), height = frequent_itemsets.support[0:11],
color ='rgmyk')
```



In [17]:

```
rules = association_rules(frequent_itemsets, metric = "lift", min_threshold = 1)
rules.head(20)
rules.sort_values('lift', ascending = False).head(10)
```

Out[17]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	lev
1170	(tropical fruit, whole milk)	(other vegetables, yogurt)	0.042298	0.043416	0.007626	0.180288	4.152546	0.0
1175	(other vegetables, yogurt)	(tropical fruit, whole milk)	0.043416	0.042298	0.007626	0.175644	4.152546	0.0
1089	(root vegetables, yogurt)	(whole milk, other vegetables)	0.025826	0.074835	0.007829	0.303150	4.050919	0.0
1092	(whole milk, other vegetables)	(root vegetables, yogurt)	0.074835	0.025826	0.007829	0.104620	4.050919	0.0
792	(berries)	(whipped/sour cream)	0.033249	0.071683	0.009049	0.272171	3.796886	0.0
793	(whipped/sour cream)	(berries)	0.071683	0.033249	0.009049	0.126241	3.796886	0.0
1174	(whole milk, yogurt)	(tropical fruit, other vegetables)	0.056024	0.035892	0.007626	0.136116	3.792358	0.0
1171	(tropical fruit, other vegetables)	(whole milk, yogurt)	0.035892	0.056024	0.007626	0.212465	3.792358	0.0
1052	(root vegetables)	(beef, other vegetables)	0.108998	0.019725	0.007931	0.072761	3.688692	0.0
1049	(beef, other vegetables)	(root vegetables)	0.019725	0.108998	0.007931	0.402062	3.688692	0.0

To eliminate Redudancy in Rules

In [18]:

```
def to_list(i):
    return (sorted(list(i)))
ma_X = rules.antecedents.apply(to_list) + rules.consequents.apply(to_list)
ma_X = ma_X.apply(sorted)
rules_sets = list(ma_X)
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 [19]:

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

In [20]:

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

Out[20]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leve
792	(berries)	(whipped/sour cream)	0.033249	0.071683	0.009049	0.272171	3.796886	0.00
678	(tropical fruit, other vegetables)	(pip fruit)	0.035892	0.075648	0.009456	0.263456	3.482649	0.00
582	(whole milk, yogurt)	(curd)	0.056024	0.053279	0.010066	0.179673	3.372304	0.00
558	(other vegetables, yogurt)	(whipped/sour cream)	0.043416	0.071683	0.010168	0.234192	3.267062	0.00
1166	(tropical fruit, whole milk, other vegetables)	(yogurt)	0.017082	0.139502	0.007626	0.446429	3.200164	0.00
1064	(tropical fruit, other vegetables)	(whipped/sour cream)	0.035892	0.071683	0.007829	0.218130	3.042995	0.00
172	(beef)	(root vegetables)	0.052466	0.108998	0.017387	0.331395	3.040367	0.01
942	(root vegetables, whole milk)	(butter)	0.048907	0.055414	0.008236	0.168399	3.038910	0.00
534	(citrus fruit, root vegetables)	(other vegetables)	0.017692	0.193493	0.010371	0.586207	3.029608	0.00
360	(tropical fruit, root vegetables)	(other vegetables)	0.021047	0.193493	0.012303	0.584541	3.020999	0.00

In [21]:

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

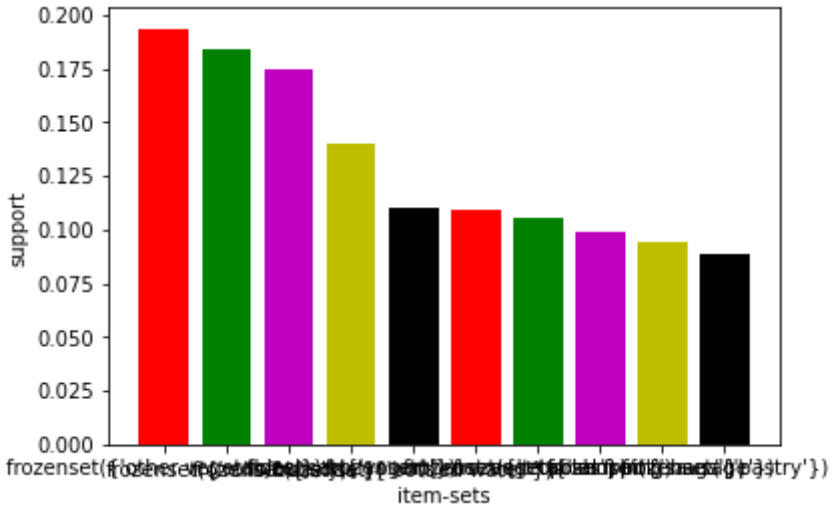
In [22]:

```
frequent_itemsets1.sort_values('support',ascending = False,inplace=True)
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-22-bc9797eccadb>:2: 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[22]:

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



In [23]:

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

In [24]:

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

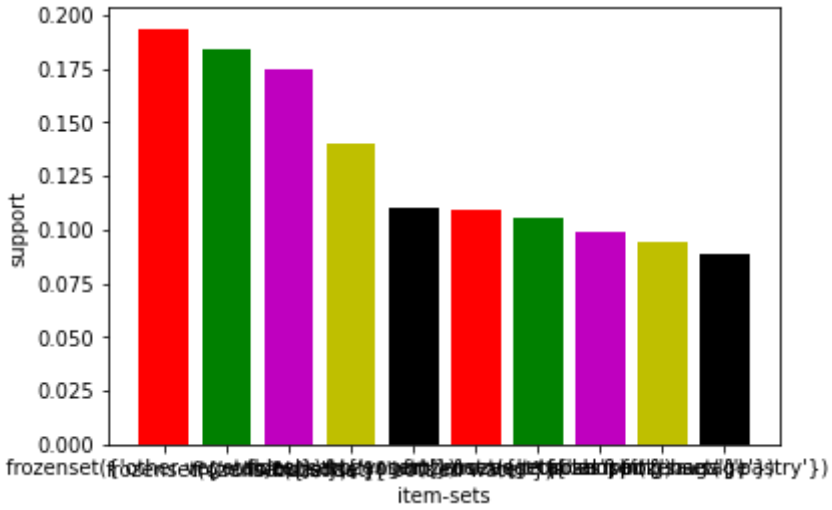
In [25]:

```
frequent_itemsets2.sort_values('support',ascending = False,inplace=True)
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-e29f79b23d96>:2: 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)
```