ASSOCIATION RULE MINING USING ECLAT ON STARS

A case study on ECLAT

Abstract

ECLAT algorithm is an association rule mining algorithm which takes transactional data in vertical format and works like Depth First Search to create frequent item sets or associative clusters. ECLAT can find associations between features and is widely used for Market basket analysis. However, it can be used in other fields also. The inspiration for this case study comes from stars being an important subject of astronomy as well as the above fact. This case study is on stars dataset. The data has been encoded to convert the data into transactional data using Python. The algorithm has been implemented on the processed data using R libraries. Associative clusters of features have been found using the algorithm.

B SHRUTI MUDALIAR

bshrutimudaliar@jklu.edu.in

Department of Computer Science and Engineering

Institute of Engineering

JK Lakshmipat University, Jaipur, India

1. Case Synopsis

Stars are the most well known astronomical objects. These are illuminating balls of gases which acted as a navigator for ancient discoverers. In the present scenario, these help the scientists and astrophysicists to unravel the mysteries of universe. Stars are the fundamental units of galaxies. The age, distribution and constitution of the stars uncover the history, dynamics and evolution of galaxies. Moreover, Stars are accountable for the production of heavy elements such as Oxygen, Nitrogen, Carbon, etc and the attributes are related to the attributes of the planetary systems that may converge about them. Therefore, Research and study on the birth, life and death is a focal point in the branch of astronomy. The subject of this case study is also based on stars. However, the perspective presented is different. This Case study is about studying an association rule mining algorithm using a dataset from the field of astronomy which is Stars.

2. Literature review

A lot of research has been done in the field of clustering stars. One of such methods is extraction of light curve sub sequences which are represented as features(vectors). Some other approaches use nearest neighbor algorithm and MST (Minimum Spanning Tree) method, one of which has been done on the star region in the small megallanic cloud.

3. Learning Objective of the case

- Understanding the vertical topology and bottom-up approach ECLAT algorithm follows to cluster the data.
- Analysing the analogy of ECLAT algorithm with DFS algorithm.
- Comparing and contrasting Apriori algorithm with ECLAT algorithm.
- Studying temperature and spectral class of stars.
- Label encoding continuous variables for creating association rules.
- Discovering clusters by creating association rules.

4. Study Questions

Q1. How will the algorithm consider the features of stars to cluster them?
Q2. Which stars are the most prominent and features do they associate with themselves?
Q3. Are main sequence stars associated with the correct range of absolute magnitude and radius?
Q4. Which association rule mining is suitable for this data?

5. Suggested Answer of Study Questions

5.1. How will the algorithm consider the features of stars to cluster them?

ECLAT algorithm takes transactional data, to cluster the stars the features have to be converted into transactions. Conditional expressions have been used to convert the data into transactional data. Every category has been created as an attribute and each record is populated with the attribute value if it exists for the particular record. The detailed steps are as follows:

1. All the continuous data has been sorted in ascending order. Each 40th, 80th, 120th, 160th, 200th and 240th value has been found. These values are the base for converting the continuous data into categorical.
2. The categories for each attribute have been created as an individual new attribute. The value for each corresponding record is filled.

5.2. Which stars are the most prominent and features do they associate with themselves?

	items	support	transIdenticalToItemsets	count
[1]	{M,red}	0.4583333	110	110
[2]	{j,o,three}	0.1666667	40	40
[3]	{j,three}	0.1666667	40	40
[4]	{j,o}	0.1666667	40	40
[5]	{o,three}	0.1666667	40	40
[6]	{g,two}	0.1666667	40	40
[7]	{five,l,m}	0.1666667	40	40
[8]	{five,m}	0.1666667	40	40
[9]	{five,l}	0.1666667	40	40
[10]	{l,m}	0.1666667	40	40

The cluster with the highest support is {M, red}. According to the available literature, the highest number of stars are of spectral class M. The above cluster shows that spectral class M star is associated with red colour. The stellar classification shows that stars of spectral class M are mostly orange red and light orange red.

5.3. Are main sequence stars associated with the correct range of absolute magnitude and radius?

	items	support	transIdenticalToItemsets	count
[1]	{M,red}	0.4583333	110	110
[2]	{j,o,three}	0.1666667	40	40
[3]	{j,three}	0.1666667	40	40
[4]	{j,o}	0.1666667	40	40
[5]	{o,three}	0.1666667	40	40
[6]	{g,two}	0.1666667	40	40
[7]	{five,l,m}	0.1666667	40	40
[8]	{five,m}	0.1666667	40	40
[9]	{five,l}	0.1666667	40	40
[10]	{l,m}	0.1666667	40	40

According to the clusters shown above, A star of type three, radius category j and magnitude category of o will be similar to each other. A main sequence star has absolute magnitude in the range of -4 to -5.8 which has been found in literature and the category o contains this range. Furthermore, the radius of main sequence star lies between 0.13 and 18 and the category j contains this range partially. This shows that this approach of ECLAT can be helpful in discovering similarities between features of stars and subsequently knowing some unknown facts about stars or research ideas related to stars and other celestial bodies.

5.4 Which association rule mining is suitable for this data?

Apriori Algorithm uses the prior knowledge of frequent itemset. A level-wise search is done where k-frequent items are used to find k+1 frequent items. It takes data in horizontal format and works like Breadth First Search (BFS).

ECLAT stands for Equivalence class Clustering and bottom-up Lattice Traversal. It recursively groups items to find intersections between item and tidset pairs according to a support value. It takes data in vertical format and mimics Depth First Search (DFS) algorithm.

According to the size of dataset, ECLAT can be used as it suits the dataset as well as it will be faster and memory efficient as compared to Apriori algorithm.
Apriori algorithm gives 1485 rules whereas ECLAT gives 596 rules. This shows that more than the half of the rules can be discarded using minimum support.

6. References

- https://rpubs.com/markloessi/500001
- https://datatofish.com/if-condition-in-pandas-dataframe/
- https://en.wikipedia.org/wiki/Association_rule_learning
- https://www.ijitee.org/wp-content/uploads/papers/v8i11/K24920981119.pdf
- https://en.wikipedia.org/wiki/Association_rule_learning
- https://www.geeksforgeeks.org/ml-eclat-algorithm/
- https://www.geeksforgeeks.org/ml-frequent-pattern-growth-algorithm/
- https://www.slideshare.net/wanaezwani/apriori-and-eclat-algorithm-in-association-rule-mining
- https://en.wikipedia.org/wiki/Stellar_classification
- http://hyperphysics.phy-astr.gsu.edu/hbase/Starlog/staspe.html
- https://www.youtube.com/watch?v=oBiq8cMkTCU
- https://www.youtube.com/watch?v=g6LBNUPNJww
- https://www.youtube.com/watch?v=p8j0jfvAvgI&t=599s
- https://en.wikipedia.org/wiki/Stellar_classification#Class_M
- Ledrew, Glenn (February 2001). "The Real Starry Sky". Journal of the Royal Astronomical Society of Canada.
- Zombeck, Martin V. (1990). Handbook of Space Astronomy and Astrophysics (2nd ed.). Cambridge University Press.
- https://iopscience.iop.org/article/10.3847/0004-637X/820/2/138/pdf
- https://iopscience.iop.org/article/10.1088/0004-637X/694/1/367/meta

Appendix A: Description of Algorithm

ECLAT

ECLAT stands for Equivalence Class Clustering and Bottom-up Lattice Traversal. It is an association rule mining algorithm. It traverses the data in vertical format like Depth first search and creates clusters of data items whose features associate together.

ASSUMPTIONS

Each record or tuple is considered as a transaction and each transaction has a transaction Id.

PROCESS

STEP 1
The data is taken in vertical format i.e., each column or attribute is considered as an item.

STEP 2
A minimum support value is set according to which the most frequent clusters can be classified.

STEP 3
Each item is paired with the transaction Id in which the particular item is occurring. This item and transactionId pairing is known as 'Item-tidset' pairing.

STEP 4
The items are grouped iteratively and item-tidset pairs are created.

STEP 5
A 'support' value is calculated for each item-tidset pairing based on the intersection of item with the transaction.

STEP 6
The item tidset pairs with support lower than minimum support are discarded.

Hence, ECLAT gives us the item sets so obtained are the associative clusters or set of attributes which occur together frequently.

Appendix B: CRISP Model Steps:

B.1. Business understanding

Define Business goal

The aim is to study ECLAT algorithm and find associative clusters of features of stars.

B.2. Data understanding

The data is of 240 stars with their 7 attributes.

The screenshot of data is shown below:

```
In [4]: star.head(5)
Out[4]:
```

	Temperature (K)	Luminosity(L/Lo)	Radius(R/Ro)	Absolute magnitude(Mv)	Star type	Star color	Spectral Class
0	3068	0.002400	0.1700	16.12	0	Red	M
1	3042	0.000500	0.1542	16.60	0	Red	M
2	2600	0.000300	0.1020	18.70	0	Red	M
3	2800	0.000200	0.1600	16.65	0	Red	M
4	1939	0.000138	0.1030	20.06	0	Red	M

```
In [5]: star.shape
Out[5]: (240, 7)
```

The statistics of data is as follows:

```
In [7]: star.describe()
Out[7]:
```

	Temperature (K)	Luminosity(L/Lo)	Radius(R/Ro)	Absolute magnitude(Mv)	Star type
count	240.000000	240.000000	240.000000	240.000000	240.000000
mean	10497.462500	107188.361635	237.157781	4.382396	2.500000
std	9552.425037	179432.244940	517.155763	10.532512	1.711394
min	1939.000000	0.000080	0.008400	-11.920000	0.000000
25%	3344.250000	0.000865	0.102750	-6.232500	1.000000
50%	5776.000000	0.070500	0.762500	8.313000	2.500000
75%	15055.500000	198050.000000	42.750000	13.697500	4.000000
max	40000.000000	849420.000000	1948.500000	20.060000	5.000000

The metadata is as follows:

Title

Star dataset to predict star types

Description

The dataset has been taken from Kaggle. The dataset is in CSV (Comma Separated Values) and contains information about 240 stars. The author of dataset has collected the data of stars from the web. The author has found the missing values using Stefan’s-Boltzmann’s law to calculate the luminosity of star, Wein’s displacement law to calculate the temperature and parallax to calculate radius.

Field description

- Temperature (K): This field contains the surface temperature of the star in Kelvin.
- Luminosity (L/Lo): This field contains the luminosity of star (L) with respect to the sun (Lo).

- Radius (R/Ro): This field contains the radius of star (R) with respect to the sun (Ro).
- Absolute magnitude (Mv): This field contains the absolute visual magnitude of star.
- Star type: This field contains type of star and is represented as a number viz.
    - Brown dwarf - 0
    - Red dwarf - 1
    - White dwarf - 2
    - Main sequence - 3
    - Supergiant - 4
    - Hypergiant – 5

- Star color: This field consists of the colour of star.
- Spectral Class: This field consists of the spectral class of star i.e., {O, B, A, F, G, K, M}.

Category or Theme

This dataset was created by the author to create a star classifier using Deep learning techniques. The theme of dataset is Astrophysics and Computer science.

Keywords

Luminosity, Temperature, Radius, Spectral class, Star type

Tags

astrophysics, computer science, physics, stars, classification, neural networks

Modification date

The data was updated by the author 7 months ago.

License

Data files © Original Authors

Data source URL

https://www.kaggle.com/deepu1109/star-dataset

B.3. Data preparation

The data consists of:

1. 4 columns with continuous data
2. 3 columns of categorical data

STEP 1: Dropping the Temperature (K) column.

According to Wein’s displacement law, the Stars can be distinguished on the basis of their surface temperatures. Spectral class also classifies the stars in another way by observing absorption lines. The absorption lines discover temperature ranges as particular absorption lines can be observed only for a range of temperatures as only in that range, the involved atomic energy levels are populated. The classes are:

Spectral Class	Temperature
O	30,000 - 60,000 K
B	10,000 - 30,000 K
A	7,500 - 10,000 K
F	6,000 - 7,500 K
G	5,000 - 6,000 K
K	3,500 - 5,000K
M	< 3,500 K

Data source: http://hyperphysics.phy-astr.gsu.edu/hbase/Starlog/staspe.html

According to the above information**,** Temperature column is redundant. Hence, the column is dropped as shown below.

In [13]:
```
#Removing Temperature column as spectral class can be used in it's place, hence it can be dropped
new=star.drop('Temperature (K)',axis=1)
```

Data before dropping Temperature column

Out[13]:

	Temperature (K)	Luminosity(L/Lo)	Radius(R/Ro)	Absolute magnitude(Mv)	Star type	Star color	Spectral Class
0	3068	0.002400	0.17000	16.120	0	Red	M
1	3042	0.000500	0.15420	16.600	0	Red	M
2	2600	0.000300	0.10200	18.700	0	Red	M
3	2800	0.000200	0.16000	16.650	0	Red	M
4	1939	0.000138	0.10300	20.060	0	Red	M
5	2840	0.000650	0.11000	16.980	0	Red	M
6	2637	0.000730	0.12700	17.220	0	Red	M
7	2600	0.000400	0.09600	17.400	0	Red	M
8	2650	0.000690	0.11000	17.450	0	Red	M
9	2700	0.000180	0.13000	16.050	0	Red	M
10	3600	0.002900	0.51000	10.690	1	Red	M

Data after dropping Temperature column

Out[15]:

	Luminosity(L/Lo)	Radius(R/Ro)	Absolute magnitude(Mv)	Star type	Star color	Spectral Class
0	0.002400	0.17000	16.120	0	Red	M
1	0.000500	0.15420	16.600	0	Red	M
2	0.000300	0.10200	18.700	0	Red	M
3	0.000200	0.16000	16.650	0	Red	M
4	0.000138	0.10300	20.060	0	Red	M
5	0.000650	0.11000	16.980	0	Red	M
6	0.000730	0.12700	17.220	0	Red	M
7	0.000400	0.09600	17.400	0	Red	M
8	0.000690	0.11000	17.450	0	Red	M
9	0.000180	0.13000	16.050	0	Red	M
10	0.002900	0.51000	10.690	1	Red	M

STEP 2: Converting the continuous data into categorical.

The columns with continuous data are first sorted to find values on the basis of which categories can be created. Consequently, categories have been created.

Luminosity column

```
In [14]: sorted_lum=new.sort_values("Luminosity(L/Lo)",ascending=True)

In [15]: print("40\n",sorted_lum.iloc[39,0:1])
         print("80\n",sorted_lum.iloc[79,0:1])
         print("120\n",sorted_lum.iloc[119,0:1])
         print("160\n",sorted_lum.iloc[159,0:1])
         print("200\n",sorted_lum.iloc[199,0:1])
         print("240\n",sorted_lum.iloc[239,0:1])

         40
          Luminosity(L/Lo)    0.00056
         Name: 82, dtype: object
         80
          Luminosity(L/Lo)    0.0013
         Name: 142, dtype: object
         120
          Luminosity(L/Lo)    0.056
         Name: 20, dtype: object
         160
          Luminosity(L/Lo)    123000
         Name: 46, dtype: object
         200
          Luminosity(L/Lo)    245000
         Name: 163, dtype: object
         240
          Luminosity(L/Lo)    849420
         Name: 233, dtype: object

In [16]: def lumina(new):
             if new['Luminosity(L/Lo)']<=0.00056: return 'a'
             elif new['Luminosity(L/Lo)']<=0.00130 and new['Luminosity(L/Lo)']>0.00056: return 'b'
             elif new['Luminosity(L/Lo)']<=0.0560 and new['Luminosity(L/Lo)']>0.00130: return 'c'
             elif new['Luminosity(L/Lo)']<=123000 and new['Luminosity(L/Lo)']>0.0560: return 'd'
             elif new['Luminosity(L/Lo)']<=245000 and new['Luminosity(L/Lo)']>123000: return 'e'
             elif new['Luminosity(L/Lo)']<=849420 and new['Luminosity(L/Lo)']>245000: return 'f'

         new['Lum'] = new.apply(lumina, axis=1)
```

Radius column

```
In [19]: sorted_rad=new.sort_values("Radius(R/Ro)",ascending=True)

In [20]: print("40\n",sorted_rad.iloc[39,1:2])
         print("80\n",sorted_rad.iloc[79,1:2])
         print("120\n",sorted_rad.iloc[119,1:2])
         print("160\n",sorted_rad.iloc[159,1:2])
         print("200\n",sorted_rad.iloc[199,1:2])
         print("240\n",sorted_rad.iloc[239,1:2])

         40
          Radius(R/Ro)    0.015
         Name: 83, dtype: object
         80
          Radius(R/Ro)    0.1542
         Name: 1, dtype: object
         120
          Radius(R/Ro)    0.73
         Name: 135, dtype: object
         160
          Radius(R/Ro)    10.6
         Name: 30, dtype: object
         200
          Radius(R/Ro)    98
         Name: 222, dtype: object
         240
          Radius(R/Ro)    1948.5
         Name: 232, dtype: object

In [21]: def rad(new):
             if new['Radius(R/Ro)']<=0.015: return 'g'
             elif new['Radius(R/Ro)']<=0.1542 and new['Radius(R/Ro)']>0.015: return 'h'
             elif new['Radius(R/Ro)']<=0.73 and new['Radius(R/Ro)']>0.1542: return 'i'
             elif new['Radius(R/Ro)']<=10.6 and new['Radius(R/Ro)']>0.73: return 'j'
             elif new['Radius(R/Ro)']<=98 and new['Radius(R/Ro)']>10.6: return 'k'
             elif new['Radius(R/Ro)']<=1948.5 and new['Radius(R/Ro)']>98: return 'l'

         new['Rad'] = new.apply(rad, axis=1)
```

Absolute magnitude column

```
In [24]: sorted_mag=new.sort_values("Absolute magnitude(Mv)",ascending=True)

In [25]: print("40\n",sorted_mag.iloc[39,2:3])
         print("80\n",sorted_mag.iloc[79,2:3])
         print("120\n",sorted_mag.iloc[119,2:3])
         print("160\n",sorted_mag.iloc[159,2:3])
         print("200\n",sorted_mag.iloc[199,2:3])
         print("240\n",sorted_mag.iloc[239,2:3])

         40
          Absolute magnitude(Mv)   -7.58
         Name: 170, dtype: object
         80
          Absolute magnitude(Mv)   -5.24
         Name: 47, dtype: object
         120
          Absolute magnitude(Mv)    6.506
         Name: 91, dtype: object
         160
          Absolute magnitude(Mv)    12.43
         Name: 194, dtype: object
         200
          Absolute magnitude(Mv)    14.94
         Name: 197, dtype: object
         240
          Absolute magnitude(Mv)    20.06
         Name: 4, dtype: object

In [26]: def mag(new):
             if new['Absolute magnitude(Mv)']<= -7.58: return 'm'
             elif new['Absolute magnitude(Mv)']<= -5.24 and new['Absolute magnitude(Mv)']> -7.58: return 'n'
             elif new['Absolute magnitude(Mv)']<=6.506 and new['Absolute magnitude(Mv)']> -5.24: return 'o'
             elif new['Absolute magnitude(Mv)']<=12.43 and new['Absolute magnitude(Mv)']>6.506: return 'p'
             elif new['Absolute magnitude(Mv)']<=14.94 and new['Absolute magnitude(Mv)']>12.43: return 'q'
             elif new['Absolute magnitude(Mv)']<=20.06 and new['Absolute magnitude(Mv)']>14.94: return 'r'

         new['Mag'] = new.apply(mag, axis=1)
```

STEP 3: Removing inconsistencies

The color column which is categorical was inconsistent as name of colour in upper case and lower case were considered differently. This problem is handled by setting one consistent spelling for each colour.

```
new.loc[(new['Star color'] == 'Blue') | (new['Star color'] == 'Blue ') , 'blue']='blue'
new.loc[(new['Star color'] == 'Blue-White') | (new['Star color'] == 'Blue-white ') , 'blue white']='blue white'
new.loc[(new['Star color'] == 'Blue white') | (new['Star color'] == 'Blue White ') , 'blue white']='blue white'
new.loc[(new['Star color'] == 'yellow-white') | (new['Star color'] == 'Yellowish white') | (new['Star color'] == 'White-Yellow')
new.loc[(new['Star color'] == 'yellowish') | (new['Star color'] == 'Yellowish') , 'yellowish']='yellowish'
```

STEP4: Converting data into transactions

Number of conditions were created to transform the data into transactions.

Luminosity column

In [17]:
```
new.loc[new['Lum'] == 'a', 'a']='a'
new.loc[new['Lum'] == 'b', 'b']='b'
new.loc[new['Lum'] == 'c', 'c']='c'
new.loc[new['Lum'] == 'd', 'd']='d'
new.loc[new['Lum'] == 'e', 'e']='e'
new.loc[new['Lum'] == 'f', 'f']='f'
new.loc[new['Lum'] != 'a', 'a']=''
new.loc[new['Lum'] != 'b', 'b']=''
new.loc[new['Lum'] != 'c', 'c']=''
new.loc[new['Lum'] != 'd', 'd']=''
new.loc[new['Lum'] != 'e', 'e']=''
new.loc[new['Lum'] != 'f', 'f']=''
```

In [18]:
```
new
```

Out[18]:

	Luminosity(L/Lo)	Radius(R/Ro)	Absolute magnitude(Mv)	Star type	Star color	Spectral Class	Lum	a	b	c	d	e	f
0	0.002400	0.17000	16.120	0	Red	M	c			c			
1	0.000500	0.15420	16.600	0	Red	M	a	a					
2	0.000300	0.10200	18.700	0	Red	M	a	a					
3	0.000200	0.16000	16.650	0	Red	M	a	a					
4	0.000138	0.10300	20.060	0	Red	M	a	a					
5	0.000650	0.11000	16.980	0	Red	M	b		b				
6	0.000730	0.12700	17.220	0	Red	M	b		b				
7	0.000400	0.09600	17.400	0	Red	M	a	a					
8	0.000690	0.11000	17.450	0	Red	M	b		b				
9	0.000180	0.13000	16.050	0	Red	M	a	a					
10	0.002900	0.51000	10.690	1	Red	M	c			c			

Radius column

In [22]:
```
new.loc[new['Rad'] == 'g', 'g']='g'
new.loc[new['Rad'] == 'h', 'h']='h'
new.loc[new['Rad'] == 'i', 'i']='i'
new.loc[new['Rad'] == 'j', 'j']='j'
new.loc[new['Rad'] == 'k', 'k']='k'
new.loc[new['Rad'] == 'l', 'l']='l'
new.loc[new['Rad'] != 'g', 'g']=''
new.loc[new['Rad'] != 'h', 'h']=''
new.loc[new['Rad'] != 'i', 'i']=''
new.loc[new['Rad'] != 'j', 'j']=''
new.loc[new['Rad'] != 'k', 'k']=''
new.loc[new['Rad'] != 'l', 'l']=''
```

In [23]:
```
new
```

Out[23]:

	Luminosity(L/Lo)	Radius(R/Ro)	Absolute magnitude(Mv)	Star type	Star color	Spectral Class	Lum	a	b	c	d	e	f	Rad	g	h	i	j	k	l
0	0.002400	0.17000	16.120	0	Red	M	c			c				i			i			
1	0.000500	0.15420	16.600	0	Red	M	a	a						h		h				
2	0.000300	0.10200	18.700	0	Red	M	a	a						h		h				
3	0.000200	0.16000	16.650	0	Red	M	a	a						i			i			
4	0.000138	0.10300	20.060	0	Red	M	a	a						h		h				
5	0.000650	0.11000	16.980	0	Red	M	b		b					h		h				
6	0.000730	0.12700	17.220	0	Red	M	b		b					h		h				
7	0.000400	0.09600	17.400	0	Red	M	a	a						h		h				
8	0.000690	0.11000	17.450	0	Red	M	b		b					h		h				
9	0.000180	0.13000	16.050	0	Red	M	a	a						h		h				
10	0.002900	0.51000	10.690	1	Red	M	c			c				i			i			

Magnitude column

In [27]:
```
new.loc[new['Mag'] == 'm', 'm']='m'
new.loc[new['Mag'] == 'n', 'n']='n'
new.loc[new['Mag'] == 'o', 'o']='o'
new.loc[new['Mag'] == 'p', 'p']='p'
new.loc[new['Mag'] == 'q', 'q']='q'
new.loc[new['Mag'] == 'r', 'r']='r'
new.loc[new['Mag'] != 'm', 'm']=''
new.loc[new['Mag'] != 'n', 'n']=''
new.loc[new['Mag'] != 'o', 'o']=''
new.loc[new['Mag'] != 'p', 'p']=''
new.loc[new['Mag'] != 'q', 'q']=''
new.loc[new['Mag'] != 'r', 'r']=''
```

In [28]:
```
new
```

Out[28]:

	Luminosity(L/Lo)	Radius(R/Ro)	Absolute magnitude(Mv)	Star type	Star color	Spectral Class	Lum	a	b	c	...	j	k	l	Mag	m	n	o	p	q	r
0	0.002400	0.17000	16.120	0	Red	M	c			c	...				r						r
1	0.000500	0.15420	16.600	0	Red	M	a	a			...				r						r
2	0.000300	0.10200	18.700	0	Red	M	a	a			...				r						r
3	0.000200	0.16000	16.650	0	Red	M	a	a			...				r						r
4	0.000138	0.10300	20.060	0	Red	M	a	a			...				r						r
5	0.000650	0.11000	16.980	0	Red	M	b		b		...				r						r
6	0.000730	0.12700	17.220	0	Red	M	b		b		...				r						r
7	0.000400	0.09600	17.400	0	Red	M	a	a			...				r						r
8	0.000690	0.11000	17.450	0	Red	M	b		b		...				r						r
9	0.000180	0.13000	16.050	0	Red	M	a	a			...				r						r
10	0.002900	0.51000	10.690	1	Red	M	c			c	...				p				p		

Spectral class column

In [34]:
```
new.loc[new['Spectral Class'] == 'M', 'M']='M'
new.loc[new['Spectral Class'] == 'B', 'B']='B'
new.loc[new['Spectral Class'] == 'A', 'A']='A'
new.loc[new['Spectral Class'] == 'F', 'F']='F'
new.loc[new['Spectral Class'] == 'O', 'O']='O'
new.loc[new['Spectral Class'] == 'K', 'K']='K'
new.loc[new['Spectral Class'] == 'G', 'G']='G'
new.loc[new['Spectral Class'] != 'M', 'M']=''
new.loc[new['Spectral Class'] != 'B', 'B']=''
new.loc[new['Spectral Class'] != 'A', 'A']=''
new.loc[new['Spectral Class'] != 'F', 'F']=''
new.loc[new['Spectral Class'] != 'O', 'O']=''
new.loc[new['Spectral Class'] != 'K', 'K']=''
new.loc[new['Spectral Class'] != 'G', 'G']=''
```

In [35]:
```
new
```

Out[35]:

	Luminosity(L/Lo)	Radius(R/Ro)	Absolute magnitude(Mv)	Star type	Star color	Spectral Class	Lum	a	b	c	...	blue white	yellow white	yellowish	M	B	A	F	O	K	G
0	0.002400	0.17000	16.120	0	Red	M	c			c	...				M						
1	0.000500	0.15420	16.600	0	Red	M	a	a			...				M						
2	0.000300	0.10200	18.700	0	Red	M	a	a			...				M						
3	0.000200	0.16000	16.650	0	Red	M	a	a			...				M						
4	0.000138	0.10300	20.060	0	Red	M	a	a			...				M						
5	0.000650	0.11000	16.980	0	Red	M	b		b		...				M						
6	0.000730	0.12700	17.220	0	Red	M	b		b		...				M						
7	0.000400	0.09600	17.400	0	Red	M	a	a			...				M						
8	0.000690	0.11000	17.450	0	Red	M	b		b		...				M						
9	0.000180	0.13000	16.050	0	Red	M	a	a			...				M						
10	0.002900	0.51000	10.690	1	Red	M	c			c	...				M						

Star type column

```
In [30]: new.loc[new['type'] == 'zero', 'zero']='zero'
         new.loc[new['type'] == 'one', 'one']='one'
         new.loc[new['type'] == 'two', 'two']='two'
         new.loc[new['type'] == 'three', 'three']='three'
         new.loc[new['type'] == 'four', 'four']='four'
         new.loc[new['type'] == 'five', 'five']='five'
         new.loc[new['type'] != 'zero', 'zero']=''
         new.loc[new['type'] != 'one', 'one']=''
         new.loc[new['type'] != 'two', 'two']=''
         new.loc[new['type'] != 'three', 'three']=''
         new.loc[new['type'] != 'four', 'four']=''
         new.loc[new['type'] != 'five', 'five']=''
```

```
In [31]: new
```

Out[31]:

	Luminosity(L/Lo)	Radius(R/Ro)	Absolute magnitude(Mv)	Star type	Star color	Spectral Class	Lum	a	b	c	...	p	q	r	type	zero	one	two	three	four	five
0	0.002400	0.17000	16.120	0	Red	M	c			c	...			r	zero	zero					
1	0.000500	0.15420	16.600	0	Red	M	a	a			...			r	zero	zero					
2	0.000300	0.10200	18.700	0	Red	M	a	a			...			r	zero	zero					
3	0.000200	0.16000	16.650	0	Red	M	a	a			...			r	zero	zero					
4	0.000138	0.10300	20.060	0	Red	M	a	a			...			r	zero	zero					
5	0.000650	0.11000	16.980	0	Red	M	b		b		...			r	zero	zero					
6	0.000730	0.12700	17.220	0	Red	M	b		b		...			r	zero	zero					
7	0.000400	0.09600	17.400	0	Red	M	a	a			...			r	zero	zero					
8	0.000690	0.11000	17.450	0	Red	M	b		b		...			r	zero	zero					
9	0.000180	0.13000	16.050	0	Red	M	a	a			...			r	zero	zero					
10	0.002900	0.51000	10.690	1	Red	M	c			c	...	p			one		one				

Star color column

```
In [32]: new.loc[new['Star color'] == 'Red', 'red']='red'
         new.loc[new['Star color'] == 'White', 'white']='white'
         new.loc[new['Star color'] == 'Orange-Red', 'orange red']='orange red'
         new.loc[new['Star color'] == 'Whitish', 'whitish']='whitish'
         new.loc[new['Star color'] == 'Orange', 'orange']='orange'
         new.loc[new['Star color'] == 'Pale yellow orange', 'pale yellow orange']='pale yellow orange'
         new.loc[(new['Star color'] == 'Blue') | (new['Star color'] == 'Blue ') , 'blue']='blue'
         new.loc[(new['Star color'] == 'Blue-White') | (new['Star color'] == 'Blue-white ') , 'blue white']='blue white'
         new.loc[(new['Star color'] == 'Blue white') | (new['Star color'] == 'Blue White ') , 'blue white']='blue white'
         new.loc[(new['Star color'] == 'yellow-white') | (new['Star color'] == 'Yellowish white') | (new['Star color'] == 'White-Yellow')
         new.loc[(new['Star color'] == 'yellowish') | (new['Star color'] == 'Yellowish') , 'yellowish']='yellowish'
         new.loc[new['Star color'] != 'Red', 'red']=''
         new.loc[new['Star color'] != 'White', 'white']=''
         new.loc[new['Star color'] != 'Orange-Red', 'orange red']=''
         new.loc[new['Star color'] != 'Whitish', 'whitish']=''
         new.loc[new['Star color'] != 'Orange', 'orange']=''
         new.loc[new['Star color'] != 'Pale yellow orange', 'pale yellow orange']=''
         new.loc[(new['Star color'] != 'Blue') | (new['Star color'] == 'Blue ') , 'blue']=''
         new.loc[(new['Star color'] != 'Blue-White') | (new['Star color'] != 'Blue-white ') , 'blue white']=''
         new.loc[(new['Star color'] != 'Blue white') | (new['Star color'] != 'Blue White ') , 'blue white']=''
         new.loc[(new['Star color'] != 'yellow-white') | (new['Star color'] != 'Yellowish white') | (new['Star color'] != 'White-Yellow')
         new.loc[(new['Star color'] != 'yellowish') | (new['Star color'] != 'Yellowish') , 'yellowish']=''
```

```
In [33]: new
```

Luminosity(L/Lo)	Radius(R/Ro)	Absolute magnitude(Mv)	Star type	Star color	Spectral Class	Lum	a	b	c	...	red	white	orange red	whitish	orange	pale yellow orange	blue	blue white	yellow white	yellowish
0.002400	0.17000	16.120	0	Red	M	c			c	...	red									
0.000500	0.15420	16.600	0	Red	M	a	a			...	red									
0.000300	0.10200	18.700	0	Red	M	a	a			...	red									
0.000200	0.16000	16.650	0	Red	M	a	a			...	red									
0.000138	0.10300	20.060	0	Red	M	a	a			...	red									
0.000650	0.11000	16.980	0	Red	M	b		b		...	red									
0.000730	0.12700	17.220	0	Red	M	b		b		...	red									
0.000400	0.09600	17.400	0	Red	M	a	a			...	red									
0.000690	0.11000	17.450	0	Red	M	b		b		...	red									

Deletion of redundant columns

In [35]:
```
new
```

Out[35]:

	Luminosity(L/Lo)	Radius(R/Ro)	Absolute magnitude(Mv)	Star type	Star color	Spectral Class	Lum	a	b	c	...	blue white	yellow white	yellowish	M	B	A	F	O	K	G
0	0.002400	0.17000	16.120	0	Red	M	c			c	...				M						
1	0.000500	0.15420	16.600	0	Red	M	a	a			...				M						
2	0.000300	0.10200	18.700	0	Red	M	a	a			...				M						
3	0.000200	0.16000	16.650	0	Red	M	a	a			...				M						
4	0.000138	0.10300	20.060	0	Red	M	a	a			...				M						
5	0.000650	0.11000	16.980	0	Red	M	b		b		...				M						
6	0.000730	0.12700	17.220	0	Red	M	b		b		...				M						
7	0.000400	0.09600	17.400	0	Red	M	a	a			...				M						
8	0.000690	0.11000	17.450	0	Red	M	b		b		...				M						
9	0.000180	0.13000	16.050	0	Red	M	a	a			...				M						
10	0.002900	0.51000	10.690	1	Red	M	c			c	...				M						

In [36]:
```
STAR=new.drop(['Luminosity(L/Lo)','Radius(R/Ro)','Absolute magnitude(Mv)','Star type','Star color','Spectral Class','Lum','Rad',
```

After Deletion

In [37]:
```
STAR
```

Out[37]:

	a	b	c	d	e	f	g	h	i	j	...	blue white	yellow white	yellowish	M	B	A	F	O	K	G
0			c						i		...				M						
1	a							h			...				M						
2	a							h			...				M						
3	a								i		...				M						
4	a							h			...				M						
5		b						h			...				M						
6		b						h			...				M						
7	a							h			...				M						
8		b						h			...				M						
9	a							h			...				M						
10			c						i		...				M						

Exporting Data to CSV file

In [38]:
```
STAR.to_csv('eclat_1.csv', header = None, index = False)
```

jupyter eclat_1.csv✓ 13 hours ago Logout

File Edit View Language current mode

```
,,c,,,,,,i,,,,,,,,,r,zero,,,,,,red,,,,,,,,,,,M,,,,,,
a,,,,,,,h,,,,,,,,,,r,zero,,,,,,red,,,,,,,,,,,M,,,,,,
a,,,,,,,h,,,,,,,,,,r,zero,,,,,,red,,,,,,,,,,,M,,,,,,
a,,,,,,,,i,,,,,,,,,r,zero,,,,,,red,,,,,,,,,,,M,,,,,,
a,,,,,,,h,,,,,,,,,,r,zero,,,,,,red,,,,,,,,,,,M,,,,,,
,b,,,,,,h,,,,,,,,,,r,zero,,,,,,red,,,,,,,,,,,M,,,,,,
,b,,,,,,h,,,,,,,,,,r,zero,,,,,,red,,,,,,,,,,,M,,,,,,
a,,,,,,,h,,,,,,,,,,r,zero,,,,,,red,,,,,,,,,,,M,,,,,,
,b,,,,,,h,,,,,,,,,,r,zero,,,,,,red,,,,,,,,,,,M,,,,,,
a,,,,,,,h,,,,,,,,,,r,zero,,,,,,red,,,,,,,,,,,M,,,,,,
,,c,,,,,,i,,,,,,,p,,,,one,,,,,red,,,,,,,,,,,M,,,,,,
,,c,,,,,,i,,,,,,,p,,,,one,,,,,red,,,,,,,,,,,M,,,,,,
a,,,,,,,,i,,,,,,,,q,,,one,,,,,red,,,,,,,,,,,M,,,,,,
,,c,,,,,,i,,,,,,,p,,,,one,,,,,red,,,,,,,,,,,M,,,,,,
,b,,,,,,h,,,,,,,,p,,,,one,,,,,red,,,,,,,,,,,M,,,,,,
,,c,,,,,,i,,,,,,,,q,,,one,,,,,red,,,,,,,,,,,M,,,,,,
,,c,,,,,,i,,,,,,,,q,,,one,,,,,red,,,,,,,,,,,M,,,,,,
,,c,,,,,,i,,,,,,,p,,,,one,,,,,red,,,,,,,,,,,M,,,,,,
,,c,,,,,,i,,,,,,,,q,,,one,,,,,red,,,,,,,,,,,M,,,,,,
,,c,,,,,,i,,,,,,,p,,,,one,,,,,red,,,,,,,,,,,M,,,,,,
,,c,,,,g,,,,,,,,,p,,,,,two,,,,,,,,,,,,,,,,B,,,,,
a,,,,,,g,,,,,,,,,,q,,,,two,,,,,white,,,,,,,,,,,,A,,,,
a,,,,,,g,,,,,,,,,,q,,,,two,,,,,white,,,,,,,,,,,,,F,,,
a,,,,,,g,,,,,,,,,,q,,,,two,,,,,white,,,,,,,,,,,,A,,,,
,,c,,,,g,,,,,,,,,p,,,,,two,,,,,,,,,,,,,,,,B,,,,,
a,,,,,,g,,,,,,,,,p,,,,,two,,,,,,,,,,,,,,,,,,F,,,
,b,,,,,g,,,,,,,,,,q,,,,two,,,,,,,,,,,,,,,,,A,,,,
a,,,,,,g,,,,,,,,,,q,,,,two,,,,,,,,,,,,,,,,,,F,,,
a,,,,,,g,,,,,,,,,,q,,,,two,,,,,,,,,,,,,,,,,,F,,,
a,,,,,,g,,,,,,,,,,q,,,,two,,,,,,,,,pale yellow orange,,,,,,,,,F,,,
,,,,e,,,,,j,,,,,o,,,,,,,three,,,,,,,,,blue,,,,,,,,,O,,
,,,d,,,,,,j,,,,,o,,,,,,,three,,,,,,,,,,,,,,,B,,,,,
,,,d,,,,,,j,,,,,o,,,,,,,three,,,,,,,,,,,,,,,B,,,,,
,,,d,,,,,,j,,,,,o,,,,,,,three,,,,,,whitish,,,,,,,,,B,,,,,
,,,d,,,,,,j,,,,,o,,,,,,,three,,,,,,,,,,,,,,,,,F,,,
,,,d,,,,,,j,,,,,o,,,,,,,three,,,,,,whitish,,,,,,,,,,A,,,,
,,,d,,,,,,j,,,,,o,,,,,,,three,,,,,,,,,,,,,,,,,F,,,
,,,d,,,,,,j,,,,,o,,,,,,,three,,,,,,,,,,,,,,,,,F,,,
,,,d,,,,,,j,,,,,o,,,,,,,three,,,,,,,,,,,,,,,,,F,,,
,,,d,,,,,,j,,,,,o,,,,,,,three,,,,,,,,,,,,,,,,,F,,,
,,,,e,,,,,,k,,,n,,,,,,,,,four,,red,,,,,,,,,,,M,,,,,,
,,,,,f,,,,,k,,,n,,,,,,,,,four,,red,,,,,,,,,,,M,,,,,,
,,,,e,,,,,,k,,,n,,,,,,,,,four,,red,,,,,,,,,,,M,,,,,,
,,,,e,,,,,,k,,,n,,,,,,,,,four,,red,,,,,,,,,,,M,,,,,,
,,,,,f,,,,,k,,,n,,,,,,,,,four,,red,,,,,,,,,,,M,,,,,,
,,,,,f,,,,,k,,,n,,,,,,,,,four,,red,,,,,,,,,,,M,,,,,,
```

Transactional data

```
In [37]: STAR
Out[37]:
```

	a	b	c	d	e	f	g	h	i	j	...	blue white	yellow white	yellowish	M	B	A	F	O	K	G
0			c						i		...				M						
1	a							h			...				M						
2	a							h			...				M						
3	a								i		...				M						
4	a							h			...				M						
5		b						h			...				M						
6		b						h			...				M						
7	a							h			...				M						
8		b						h			...				M						
9	a							h			...				M						
10			c						i		...				M						

B.4. Modelling

The processed data has been exported as a CSV file using Python. Further, the ECLAT algorithm is implemented in R and the item sets have been obtained.

The summary of transactions is shown in the figure below:

```
5:19   (Top Level) ÷                                                    R Script ÷
Console   Terminal ×
~/
>
> star_data = read.csv('eclat_1.csv', header = FALSE)
>
>
>
> star_data = read.transactions('eclat_1.csv', sep = ',', rm.duplicates = TRUE)
>
>
>
> summary(star_data)
transactions as itemMatrix in sparse format with
 240 rows (elements/itemsets/transactions) and
 38 columns (items) and a density of 0.1513158

most frequent items:
    red       M    blue       B       a (Other)
    112     111      55      46      40    1016

element (itemset/transaction) length distribution:
sizes
  5   6
 60 180

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   5.00    5.75    6.00    5.75    6.00    6.00

includes extended item information - examples:
  labels
1      a
2      A
3      b
> |
```

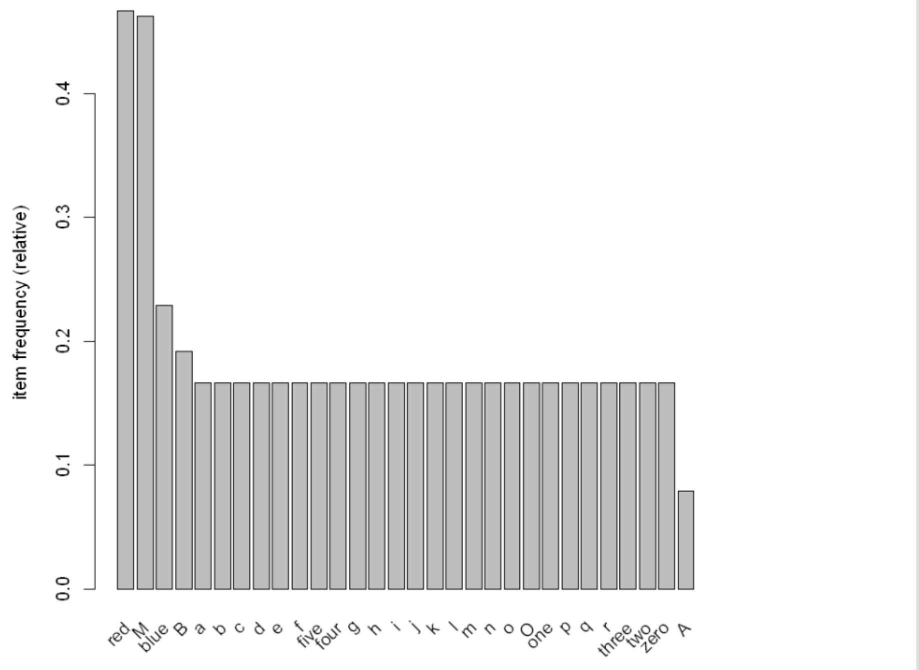
The implementation of ECLAT is as shown below:

```
>
>
> rules = eclat(data = star_data, parameter = list(support = 0.025, minlen =2))
Eclat

parameter specification:
 tidLists support minlen maxlen            target   ext
    FALSE    0.025      2     10 frequent itemsets TRUE

algorithmic control:
 sparse sort verbose
      7   -2    TRUE

Absolute minimum support count: 6

create itemset ...
set transactions ...[38 item(s), 240 transaction(s)] done [0.00s].
sorting and recoding items ... [33 item(s)] done [0.00s].
creating bit matrix ... [33 row(s), 240 column(s)] done [0.00s].
writing  ... [596 set(s)] done [0.00s].
Creating S4 object  ... done [0.00s].
> |
```

B.5. Evaluation

The results obtained are item sets along with their support and count of identical transactions which successfully meets the business goal. Also, the frequency plot gives an insight of the most frequent features.

The Frequency plot is shown below:



The item sets by Apriori algorithm are shown below:

```
In [10]: inspect(sort(rules, by = 'support')[1:10])

     lhs        rhs     support   confidence coverage  lift     count
[1]  {M}     => {red}   0.4583333 0.9909910  0.4625000 2.123552 110
[2]  {red}   => {M}     0.4583333 0.9821429  0.4666667 2.123552 110
[3]  {three} => {o}     0.1666667 1.0000000  0.1666667 6.000000  40
[4]  {o}     => {three} 0.1666667 1.0000000  0.1666667 6.000000  40
[5]  {three} => {j}     0.1666667 1.0000000  0.1666667 6.000000  40
[6]  {j}     => {three} 0.1666667 1.0000000  0.1666667 6.000000  40
[7]  {o}     => {j}     0.1666667 1.0000000  0.1666667 6.000000  40
[8]  {j}     => {o}     0.1666667 1.0000000  0.1666667 6.000000  40
[9]  {two}   => {g}     0.1666667 1.0000000  0.1666667 6.000000  40
[10] {g}     => {two}   0.1666667 1.0000000  0.1666667 6.000000  40
```

The item sets by ECLAT algorithm are shown below:

```
In [15]: inspect(sort(rules, by = 'support')[1:10])

     items        support   transIdenticalToItemsets count
[1]  {M,red}      0.4583333 110                      110
[2]  {j,o,three}  0.1666667  40                       40
[3]  {j,three}    0.1666667  40                       40
[4]  {j,o}        0.1666667  40                       40
[5]  {o,three}    0.1666667  40                       40
[6]  {g,two}      0.1666667  40                       40
[7]  {five,l,m}   0.1666667  40                       40
[8]  {five,m}     0.1666667  40                       40
[9]  {five,l}     0.1666667  40                       40
[10] {l,m}        0.1666667  40                       40
```

ANNEXURE: DATA DESCRIPTIVE ANALYTICS

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
data=pd.read_csv(r"C:\Users\Shruti\Desktop\...shruti\JKLU\VI-SEM\Predictive_Analytics\STAR-Dataset.csv")
```

In [3]:
```
star=data.iloc[:,:7]
```

In [4]:
```
star.head(5)
```

Out[4]:

	Temperature (K)	Luminosity(L/Lo)	Radius(R/Ro)	Absolute magnitude(Mv)	Star type	Star color	Spectral Class
0	3068	0.002400	0.1700	16.12	0	Red	M
1	3042	0.000500	0.1542	16.60	0	Red	M
2	2600	0.000300	0.1020	18.70	0	Red	M
3	2800	0.000200	0.1600	16.65	0	Red	M
4	1939	0.000138	0.1030	20.06	0	Red	M

In [5]:
```
star.shape
```

Out[5]: (240, 7)

In [6]:
```
star.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 240 entries, 0 to 239
Data columns (total 7 columns):
Temperature (K)           240 non-null int64
Luminosity(L/Lo)          240 non-null float64
Radius(R/Ro)              240 non-null float64
Absolute magnitude(Mv)    240 non-null float64
Star type                 240 non-null int64
Star color                240 non-null object
Spectral Class            240 non-null object
dtypes: float64(3), int64(2), object(2)
memory usage: 13.2+ KB
```

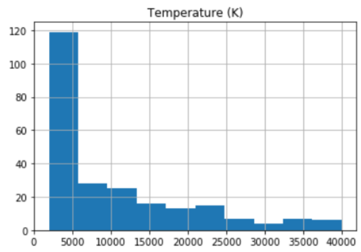
In [7]:
```
star.describe()
```

Out[7]:

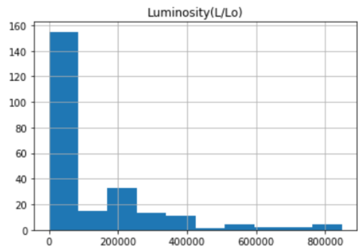
	Temperature (K)	Luminosity(L/Lo)	Radius(R/Ro)	Absolute magnitude(Mv)	Star type
count	240.000000	240.000000	240.000000	240.000000	240.000000
mean	10497.462500	107188.361635	237.157781	4.382396	2.500000
std	9552.425037	179432.244940	517.155763	10.532512	1.711394
min	1939.000000	0.000080	0.008400	-11.920000	0.000000
25%	3344.250000	0.000865	0.102750	-6.232500	1.000000
50%	5776.000000	0.070500	0.762500	8.313000	2.500000
75%	15055.500000	198050.000000	42.750000	13.697500	4.000000
max	40000.000000	849420.000000	1948.500000	20.060000	5.000000

In [8]:
```
star.hist('Temperature (K)')
```

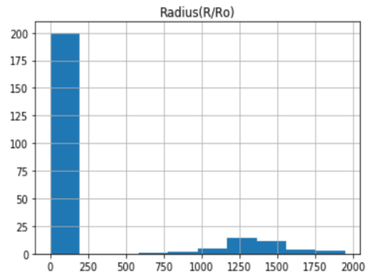
Out[8]:
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x000001E974D394E0>]],
      dtype=object)
```



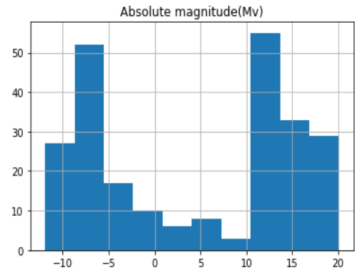
```
In [9]: star.hist('Luminosity(L/Lo)')
Out[9]: array([[<matplotlib.axes._subplots.AxesSubplot object at 0x000001E973D13E10>]],
              dtype=object)
```



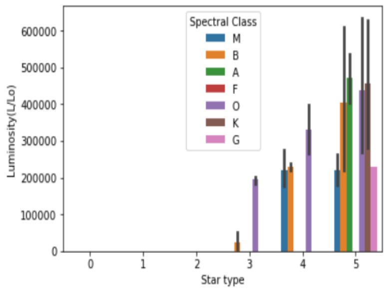

```
In [10]: star.hist('Radius(R/Ro)')
Out[10]: array([[<matplotlib.axes._subplots.AxesSubplot object at 0x000001E975062EF0>]],
               dtype=object)
```



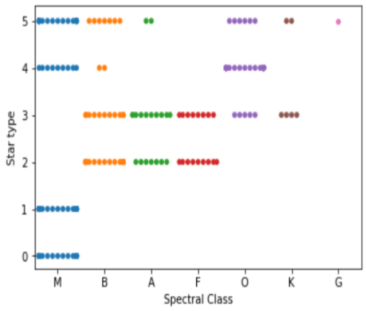

```
In [11]: star.hist('Absolute magnitude(Mv)')
Out[11]: array([[<matplotlib.axes._subplots.AxesSubplot object at 0x000001E973CE0668>]],
               dtype=object)
```



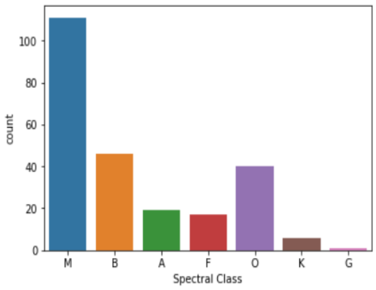

```
In [27]: sns.barplot(x="Star type",y="Luminosity(L/Lo)", hue="Spectral Class", data=star)
Out[27]: <matplotlib.axes._subplots.AxesSubplot at 0x1e9789679b0>
```



```
In [16]: sns.swarmplot(x="Spectral Class", y="Star type", data=star)
Out[16]: <matplotlib.axes._subplots.AxesSubplot at 0x1e975367860>
```




```
In [18]: sns.countplot(x="Spectral Class", data=star)
Out[18]: <matplotlib.axes._subplots.AxesSubplot at 0x1e9753d1b70>
```




```
In [19]: hmdata=star.corr()
         sns.heatmap(hmdata)
Out[19]: <matplotlib.axes._subplots.AxesSubplot at 0x1e9753b8c88>
```