Assignment 3

Made by
Swati Verma
MT19073

In this question we need to implement research paper on generating hyperchique patterns using Hyperclique miner algorithm.
This algorithm generates all the hyper clique patterns with support and h-confidence above threshold specified by the user.

Algorithm implementation:

1. Converted .dat file to .csv file so that it becomes quite easy to perform operations on data.
2. Converted data into list of lists.
3. Calculated frequency for single candidate set.
4. Calculated support values for one item dataset and pruned if the count is greater than min_support given by user.
5. Defined a function gen_set() which generate sets of k sizes.
6. Defined a function supp_prun() which prunes the data for sets which are greater than size one by comparing support count for each data item with min_support.
7. Defined function gen_cross() which calculates cross support for data item ‘x’ as support(x)* h_confidence and if this value is less than the support of any of the remaining data items then the superset which contains data item x are pruned.
8. Defined function gen_hcon_cross() which prunes the data if value of h-confidence is less than h-con threshold. H-confidence for {A-->BC} is calculated as support{ABC}/max(support of all subsets of A,B,C).
9. Calculated number of hyperclique patterns after all the pruning is finished.
10. Plotted graph between different parameters.

Graphs

1. Minimum h-confidence thresholds vs no. of hyperclique patterns keeping min_supp=0.02

h-con=0.1

```
# print(global_list2)
# print(support)
print(global_list1)

print("--%s seconds--" %(time.time()-s_time))
```

```
[[1], [2], [3], [4], [6], [7], [11], [27], [40], [55], [64], [69], [77], [83], [90], [136], [138], [148], [205], [215], [218],
[278], [294], [303], [316], [446], [490], [1, 7], [1, 148], [1, 218], [218, 148], [27, 7], [1, 218, 148]]
--449.1713216304779 seconds--
```

In [15]:
```
hyper_p=[]
hyper_p.append(len(global_list1))
print(hyper_p)
```

```
[33]
```

h-con=0.2

```
# print(global_list2)
# print(support)
print(global_list1)

print("--%s seconds--" %(time.time()-s_time))
```

```
[[1], [2], [3], [4], [6], [7], [11], [27], [40], [55], [64], [69], [77], [83], [90], [136], [138], [148], [205], [215], [218],
[278], [294], [303], [316], [446], [490], [27, 7], [218, 148]]
--454.19286036491394 seconds--
```

In [15]:
```
hyper_p=[]
hyper_p.append(len(global_list1))
print(hyper_p)
```

```
[29]
```

h-con=0.3

```
# print(support)
print(global_list1)

print("--%s seconds--" %(time.time()-s_time))
```

```
[[1], [2], [3], [4], [6], [7], [11], [27], [40], [55], [64], [69], [77], [83], [90], [136], [138], [148], [205], [215], [218],
[278], [294], [303], [316], [446], [490]]
--432.653879404068 seconds--
```

In [15]:
```
hyper_p=[]
hyper_p.append(len(global_list1))
print(hyper_p)
```

```
[27]
```

h-con=0.4

```
print(global_list1)

print("--%s seconds--" %(time.time()-s_time))
```

```
[[1], [2], [3], [4], [6], [7], [11], [27], [40], [55], [64], [69], [77], [83], [90], [136], [138], [148], [205], [215], [218],
[278], [294], [303], [316], [446], [490]]
--447.9654152393341 seconds--
```

In [15]:
```
hyper_p=[]
hyper_p.append(len(global_list1))
print(hyper_p)
```

```
[27]
```

h-con=0.6

```
print(global_list1)

print("--%s seconds--" %(time.time()-s_time))
```

```
[[1], [2], [3], [4], [6], [7], [11], [27], [40], [55], [64], [69], [77], [83], [90], [136], [138], [148], [205], [215], [218],
[278], [294], [303], [316], [446], [490]]
--431.8706855773926 seconds--
```

In [15]:
```
hyper_p=[]
hyper_p.append(len(global_list1))
print(hyper_p)
```

```
[27]
```

h-con=0.8

```
# print(support)
print(global_list1)

print("--%s seconds--" %(time.time()-s_time))
```

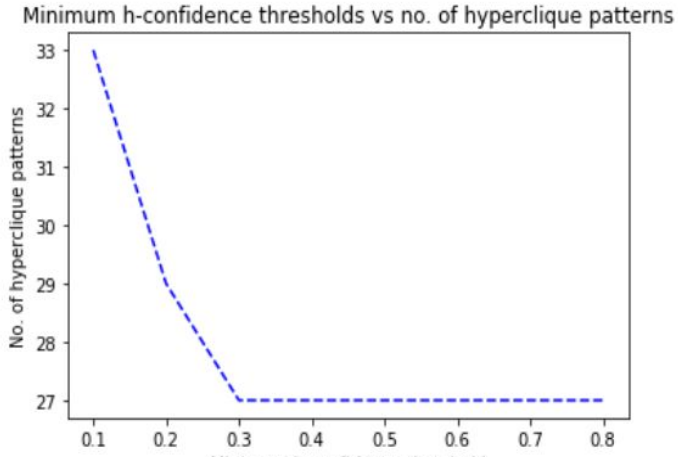
```
[[1], [2], [3], [4], [6], [7], [11], [27], [40], [55], [64], [69], [77], [83], [90], [136], [138], [148], [205], [215], [218],
[278], [294], [303], [316], [446], [490]]
--437.8841369152069 seconds--
```

In [15]:
```
hyper_p=[]
hyper_p.append(len(global_list1))
print(hyper_p)
```

```
[27]
```

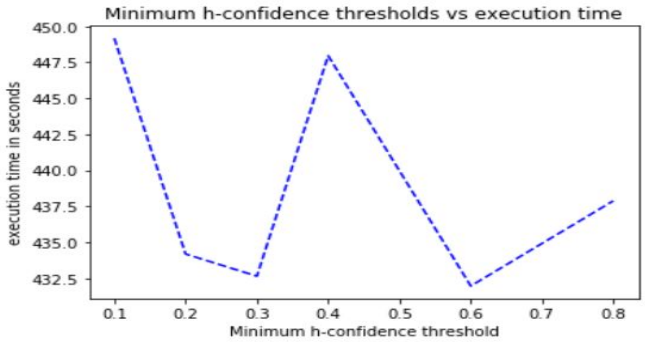
In [16]:
```
# import matplotlib.pyplot as plt
# h_confidence=[]
```

```
h_conf=[0.1,0.2,0.3,0.4,0.6,0.8]
hyper=[33,29,27,27,27,27]
plt.plot(h_conf, hyper, color='blue', linestyle='dashed', markersize=12)
plt.title('Minimum h-confidence thresholds vs no. of hyperclique patterns')

plt.xlabel('Minimum h-confidence threshold')
plt.ylabel('No. of hyperclique patterns')
plt.show()
```



2. Minimum h-confidence thresholds vs execution time

```
#https://swcarpentry.github.io/python-novice-gapminder/09-plotting/
#https://www.geeksforgeeks.org/graph-plotting-in-python-set-1/
import matplotlib.pyplot as plt
h_conf=[0.1,0.2,0.3,0.4,0.6,0.8]
exe_time=[449.171,434.192,432.653,447.96,431.9706,437.884]
plt.plot(h_conf, exe_time, color='blue', linestyle='dashed', markersize=12)
plt.title('Minimum h-confidence thresholds vs execution time')

plt.xlabel('Minimum h-confidence threshold')
plt.ylabel('execution time in seconds')
plt.show()
```



3. Minimum support thresholds vs no. of hyperclique patterns keeping min_h-con=0.03

supp=0.05

```
print(global_list1)

print("--%s seconds--" %(time.time()-s_time))
```

```
[[1], [3], [4], [6], [7], [11], [27], [55], [148], [218], [1, 11], [6, 7], [3, 6], [11, 6], [11, 148], [218, 11], [218, 148],
[27, 6], [1, 6], [11, 7], [3, 11], [1, 3], [148, 6], [218, 6], [218, 11, 148], [11, 148, 6], [1, 3, 6], [11, 6, 7], [218, 11,
6], [218, 148, 6], [11, 3, 6], [1, 11, 6], [218, 11, 148, 6]]
--104.06183815002441 seconds--
```

```
hyper_p=[]
hyper_p.append(len(global_list1))
print(hyper_p)
```

```
[33]
```

supp=0.1

```
print("--%s seconds--" %(time.time()-s_time))
```

```
[[1], [3], [6], [11], [1, 6], [3, 6], [3, 11], [11, 6], [11, 3, 6]]
--25.049624919891357 seconds--
```

```
hyper_p=[]
hyper_p.append(len(global_list1))
print(hyper_p)
```

```
[9]
```

supp=0.15

```
print(global_list1)

print("--%s seconds--" %(time.time()-s_time))
```

```
[[1], [3], [6], [11], [3, 6], [3, 11], [11, 6]]
--18.834959983825684 seconds--
```

```
hyper_p=[]
hyper_p.append(len(global_list1))
print(hyper_p)
```

```
[7]
```

supp=0.2

supp=0.5

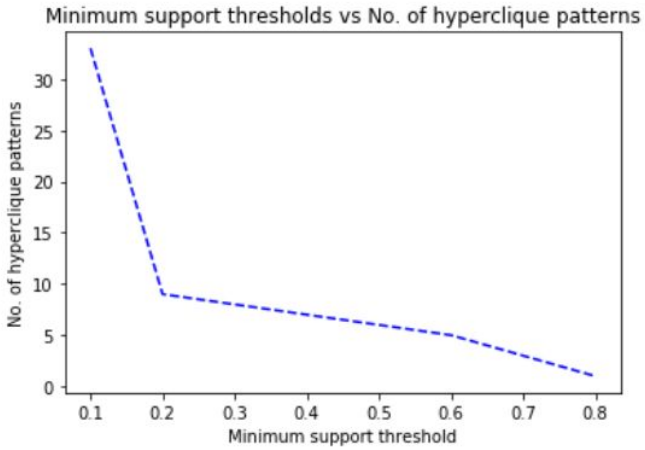
```
# print(support)
print(global_list1)

print("--%s seconds--" %(time.time()-s_time))
```

```
[[6]]
6.627076148986816 seconds
```

```
hyper_p=[]
hyper_p.append(len(global_list1))
print(hyper_p)
```

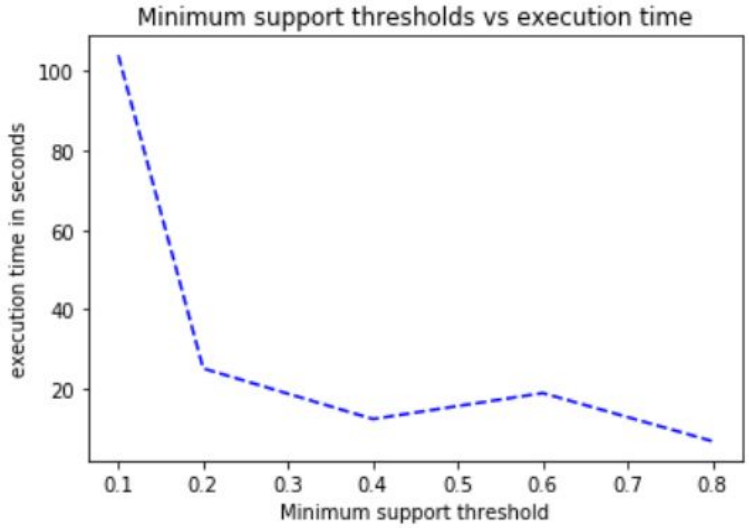
```
[1]
```

```
sup=[0.05,0.1,0.15,0.2,0.5]
hyper1=[33,9,7,5,1]
plt.plot(h_conf, hyper1, color='blue', linestyle='dashed', markersize=12)
plt.title('Minimum support thresholds vs No. of hyperclique patterns')

plt.xlabel('Minimum support threshold')
plt.ylabel('No. of hyperclique patterns')
plt.show()
```



4. Minimum support thresholds vs execution time

```
sup=[0.05,0.1,0.15,0.2,0.5]
exe_time=[104.06,25.049,12.279,18.834,6.687]
plt.plot(h_conf, exe_time, color='blue', linestyle='dashed', markersize=12)
plt.title('Minimum support thresholds vs execution time')

plt.xlabel('Minimum support threshold')
plt.ylabel('execution time in seconds')
plt.show()
```



Conclusion:

We conclude that number of hyperclique patterns decreases with increase in min_supp threshold and h_con threshold values.

Execution time also decreases with increase in threshold values.

References:

https://www.geeksforgeeks.org/frozenset-in-python/

https://stackoverflow.com/questions/36845032/how-to-convert-dat-to-csv-using-python

https://stackoverflow.com/questions/2161752/how-to-count-the-frequency-of-the-elements-in-a-list/2162045

https://stackoverflow.com/questions/2600191/how-can-i-count-the-occurrences-of-a-list-item

https://stackoverflow.com/questions/1557571/how-do-i-get-time-of-a-python-programs-execution

https://www.geeksforgeeks.org/graph-plotting-in-python-set-1/

https://stackoverflow.com/questions/13264511/typeerror-unhashable-type-dict
https://stackoverflow.com/questions/16803393/python-error-unhashable-type-list