### Bronco ID: 014429779

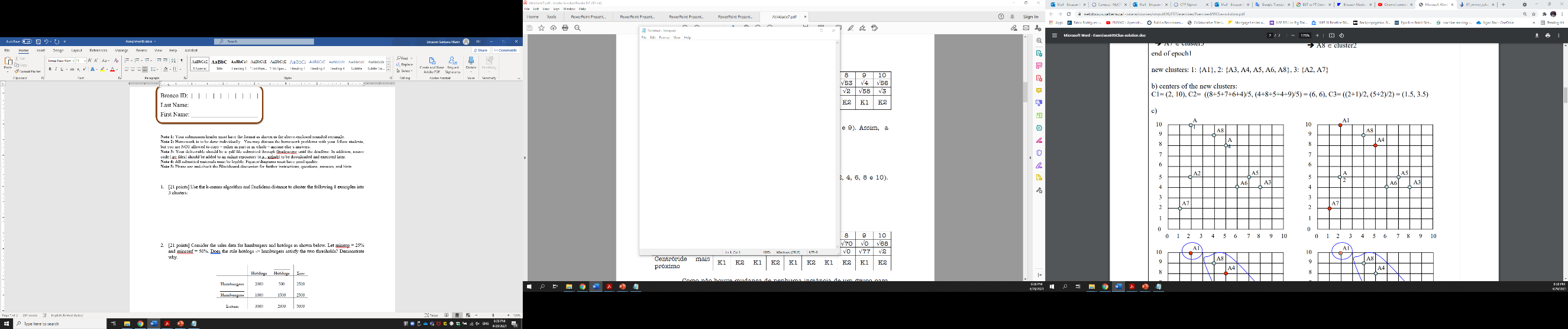
Last Name: KOEPKE **CS 4210.01 ASSIGNMENT #5**

First Name: CHRISTOPHER

1. By considering the following 8 2D data points below do:
   1. Group the points into 3 clusters by using k-means algorithm with Euclidean distance. Show the intermediate clusters (**by drawing ellipses on this 2D space**) and centroids (**by drawing marks like X on this 2D**) in each iteration until convergence. Consider the initial centroids as: C1 = A1, C2 = A4, and C3 = A7.

First Iteration: Second Iteration:

C1 = (2, 10) C2 = (5, 8) C3= (1,2) C1 = (2, 10) C2 = (6, 6) C3= (1.5, 3.5)

Graphical user interface, application

Description automatically generated

Third Iteration: Fourth Iteration:

C1 = (3, 9.5) C2 = (6.5, 5.25) C3= (1.5, 3.5) C1 = (3.67, 9) C2 = (7, 4.33) C3= (1.5, 3.5)

Graphical user interface, application

Description automatically generatedGraphical user interface, application

Description automatically generated

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1st iteration | | | | | | | | |
| Centroid: (C1, C2, C3) | | | | | | | | |
| Instance | A1 | A2 | A3 | A4 | A5 | A6 | A7 | A8 |
| C1 dist. | 0 | 5 | 8.4853 | 3.6056 | 7.0711 | 7.2111 | 8.0623 | 2.2361 |
| C2 dist. | 3.6056 | 4.2426 | 5 | 0 | 3.6056 | 4.1231 | 7.2111 | 1.4142 |
| C3 dist. | 8.0623 | 3.1623 | 7.2801 | 7.2111 | 6.7082 | 5.3852 | 0 | 7.6158 |
| Cluster Assigned | C1 | C3 | C2 | C2 | C2 | C2 | C3 | C2 |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 2st iteration | | | | | | | | |
| Centroid: (C1, C2, C3) | | | | | | | | |
| Instance | A1 | A2 | A3 | A4 | A5 | A6 | A7 | A8 |
| C1 dist. | 0 | 5 | 8.4853 | 3.6056 | 7.0711 | 7.2111 | 8.0623 | 2.2361 |
| C2 dist. | 5.6569 | 4.1231 | 2.8284 | 2.2361 | 1.1412 | 2 | 6.4031 | 3.6056 |
| C3 dist. | 6.5192 | 1.5811 | 6.5192 | 5.7009 | 5.7009 | 4.5277 | 1.5811 | 6.0415 |
| Cluster Assigned | C1 | C3 | C2 | C2 | C2 | C2 | C3 | C1 |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 3st iteration | | | | | | | | |
| Centroid: (C1, C2, C3) | | | | | | | | |
| Instance | A1 | A2 | A3 | A4 | A5 | A6 | A7 | A8 |
| C1 dist. | 1.1180 | 4.6098 | 7.4330 | 2.5 | 6.0208 | 6.2650 | 7.7621 | 1.1180 |
| C2 dist. | 6.5431 | 4.5069 | 1.9526 | 3.1325 | 0.5590 | 1.3463 | 6.3885 | 4.5070 |
| C3 dist. | 6.5192 | 1.5811 | 6.5192 | 5.7009 | 5.7009 | 4.5277 | 1.5811 | 6.0415 |
| Cluster Assigned | C1 | C3 | C2 | C1 | C2 | C2 | C3 | C1 |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 4st iteration | | | | | | | | |
| Centroid: (C1, C2, C3) | | | | | | | | |
| Instance | A1 | A2 | A3 | A4 | A5 | A6 | A7 | A8 |
| C1 dist. | 1.9465 | 4.3346 | 6.6143 | 1.6640 | 5.2047 | 5.5162 | 7.4919 | 0.33 |
| C2 dist. | 7.5597 | 2.1092 | 1.0530 | 4.1796 | 0.67 | 1.0530 | 6.4365 | 5.5506 |
| C3 dist. | 6.5192 | 1.5811 | 6.5192 | 5.7009 | 5.7009 | 4.5277 | 1.5811 | 6.0415 |
| Cluster Assigned | C1 | C3 | C2 | C1 | C2 | C2 | C3 | C1 |

**Note: Clusters do not move after the 4th iteration, thus convergence.**

* 1. Calculate the SSE (Sum of Square Errors) of the final clustering.

1. Complete the Python program (clustering.py) that will read the file training\_data.csv to cluster the data. Your goal is to run k-means multiple times and check which k value maximizes the Silhouette coefficient. You also need to plot the values of k and their corresponding Silhouette coefficients so that we can visualize and confirm the best k value found. Next, you will calculate and print the Homogeneity score (the formula of this evaluation metric is provided in the template) of this best k clustering task by using the testing\_data.csv, which is a file that includes ground truth data (classes).

https://github.com/chris-k87/CS\_4210.01/tree/main/Assignment\_5/Clustering

1. The dataset below presents the user ratings on a 1-3 scale for 6 different rock bands.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Bon Jovi | Metallica | Scorpions | AC/DC | Kiss | Guns n’ Roses |
| Fred | 1 | 3 | - | 3 | 1 | 3 |
| Lillian | 3 | - | 2 | 2 | 3 | 1 |
| Cathy | 2 | 2 | 2 | 3 | - | 2 |
| John | 3 | 2 | 2 | 2 | ? | ? |

* 1. Apply **user-based** collaborative filtering on the dataset to decide about recommending the bands Kiss and Guns n’ Roses to John. You should make a recommendation when the predicted rating is greater than or equal to 2.0. Use cosine similarity, a neutral value (1.5) for missing values, and the top 2 similar neighbors to build your model.

John [3, 2, 2, 2]

Fred [1, 3, 1.5, 3]

Lillian [3, 1.5, 2, 2]

Cathy [2, 2, 2, 3]

* 1. Now, apply **item-based** collaborative filtering to make the same decision. Use the same parameters defined before to build your model.

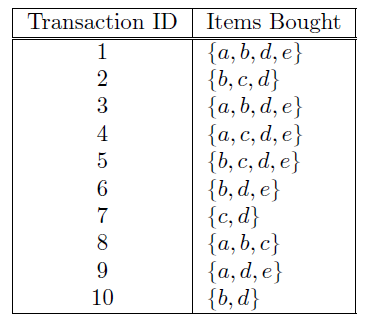
Kiss [1, 3, 1.5] Guns n’ Roses[3, 1, 2]

Bon Jovi [1, 3, 2]

Metallica [3, 1.5, 2]

Scorpions [1.5, 2, 2]

AC/DC [3, 2, 3]

1. Consider the following transaction dataset. Suppose that minimum support is set to 30% (*minsup*) and minimum confidence is set to 60%.
2. Rank all frequent itemsets according to their support (list their support values).

|  |  |  |
| --- | --- | --- |
| 1-Itemset | Support  Count | Support |
| {a} | 5 | 5/10 = 0.5 |
| {b} | 7 | 7/10 = 0.7 |
| {c} | 5 | 5/10 = 0.5 |
| {d} | 9 | 9/10 = 0.9 |
| {e} | 6 | 6/10 = 0.6 |

|  |  |  |
| --- | --- | --- |
| 2-Itemset | Support  Count | Support |
| {a,b} | 3 | 3/10 = 0.3 |
| {a,c} | 2 | 2/10 = 0.2 |
| {a,d} | 4 | 4/10 = 0.4 |
| {a,e} | 4 | 4/10 = 0.4 |
| {b,c} | 3 | 3/10 = 0.3 |
| {b,d} | 6 | 6/10 = 0.6 |
| {b,e} | 4 | 4/10 = 0.4 |
| {c,d} | 4 | 4/10 = 0.4 |
| {c,e} | 2 | 2/10 = 0.2 |
| {d,e} | 6 | 6/10 = 0.6 |

|  |  |  |
| --- | --- | --- |
| 3-Itemset | Support  Count | Support |
| {a,b,c} | 1 | 1/10 = 0.1 |
| {a,b,d} | 2 | 2/10 = 0.2 |
| {a,b,e} | 2 | 2/10 = 0.2 |
| {a,c,d} | 1 | 1/10 = 0.1 |
| {a,c,e} | 1 | 1/10 = 0.1 |
| {a,d,e} | 4 | 4/10 = 0.4 |
| {b,c,d} | 2 | 2/10 = 0.2 |
| {b,c,e} | 1 | 1/10 = 0.1 |
| {b,d,e} | 4 | 4/10 = 0.4 |
| {c,d,e} | 2 | 2/10 = 0.2 |

|  |  |  |
| --- | --- | --- |
| 4-Itemset | Support  Count | Support |
| {a,b,c,d} | 0 | 0/10 = 0.0 |
| {a,b,c,e} | 0 | 0/10 = 0.0 |
| {a,b,d,e} | 2 | 2/10 = 0.2 |
| {b,c,d,e} | 1 | 1/10 = 0.1 |

Note: Grey highlighted itemsets indicated failed *minsup* at 30%

1. For all frequent 3-itemsets, rank all association rules - according to their confidence values - which satisfy the requirements on minimum support and minimum confidence (list their confidence values).

Itemset: {a,d,e} Itemset: {b,d,e}

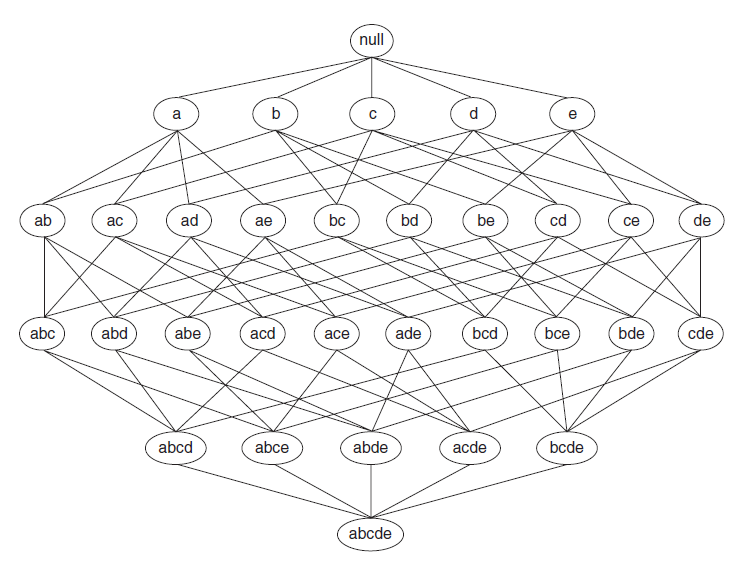
|  |  |  |
| --- | --- | --- |
| Association Rule | Support | Confidence |
| ad->e | 0.4 | 1.0 |
| ae->d | 0.4 | 1.0 |
| a->de | 0.4 | 0.8 |
| de->a | 0.4 | 0.67 |
| e->ad | 0.4 | 0.67 |
| d->ae | 0.4 | 0.44 |

|  |  |  |
| --- | --- | --- |
| Association Rule | Support | Confidence |
| be->d | 0.4 | 1.0 |
| bd->e | 0.4 | 0.67 |
| de->b | 0.4 | 0.67 |
| e->bd | 0.4 | 0.67 |
| b->de | 0.4 | 0.57 |
| d->be | 0.4 | 0.44 |

Note: Grey highlighted association rule failed *minconf* at 60%

1. Show how the 3-itemsets candidates can be generated by the X method and if these candidates will be pruned or not.

* F2 = {ab, ad, ae, bc, bd, be, cd, de}
  + Merge: {ab, ad} = abd Pruned
  + Merge: {ab, ae} = abe Pruned
  + Merge: {ad, ae} = ade Not Pruned
  + Merge: {bc, bd} = bcd Pruned
  + Merge: {bc, be} = bce Pruned
  + Merge: {bd, be} = bde Not Pruned

1. Consider the lattice structure given below. Label each node with the following letter(s): *F* if it is frequent and *I* if it is infrequent.

F F F F F

F I F F F F F F I F

I I I I I F I I F I

I I I I I

1. Complete the Python program (association\_rule\_mining.py) that will read the file retail\_dataset.csv to find strong rules related to supermarket products. You will need to install a python library this time. Just use your terminal to type: pip install mlxtend. Your goal is to output the rules that satisfy *minsup* = 0.2 and *minconf* = 0.6, as well as the priors and probability gains of the rule consequents when conditioned to the antecedents. The formulas for this math are given in the template.