**Mock Midterm Exam**

INFS4203/7203

Semester 2, 2020

**Q1. Please generally describe the procedure of k-means algorithm (50 points)**

**Answer 1:** Write your answer to Q1 here.

Fix k, minimize SSE in a greedy manner

1. Random pick up centroids

We randomly select k points as initial centroids

*(What could be the potential issues for such initialization?*

*Such initialization may make the algorithm very hard to converge)*

2. Optimize cluster assignment

Round1

Form k clusters by assigning points to the closest centroid

Calculate all points’ distances to the k centroids

Update the centroid by the “mean” of each cluster

Round2-n

Calculate the distance again and re-assign the cluster

Update the centroid

3. Optimize centroids -> go back step 2

Assign point to a closer centroid as a cluster

Until --The cluster assignment don’t change

**Q1 (add1). We have data stored in abc.csv. Please describe the general process in the plain language of using Python to do k-means clustering on this data and illustrate the results. The following packages have to be used:**

**Pandas, NumPy, Matplotlib, Sklearn.**

**Please specify what kind of factors/settings may impact the clustering result.**

**General Process:**

1. Import the necessary packages (e.g. numpy, Matplotlib.pyplot, pandas, and random) we plan to use.

2. Load the given CSV file by Pandas.read\_csv(abc.csv).

3. Use column operation in Pandas to select necessary attributes. The values of these attributes are then extracted to a Numpy array for clustering.

4. Initialize the Kmeans function in Sklearn by specifying the number of clusters. If we want the results to be reproducible, we also need to specify the random state.

5. Fit the function by feeding the Numpy array.

6. Allocate the data to the specific cluster.

7. Use Matplotlib to visualize the clustering result.

**kind of factors/settings:**

1. Parameter selection. (e.g. number of clusters and random state)

Poor initialization can cause the iterations to get stuck into an inferior local minimum.

Note that for running k-means, the first step is to randomly select several data points as the "initial" center points.

Thus, we fix the random\_state to be the random seed 1, such that the results can be reproducible.

**Q1 (add2). Please discuss in the plain language of what are the differences between running the following three methods in Python:**

**KMeans(n\_clusters=3, random\_state=1).fit(X)**

**KMeans(n\_clusters=3, random\_state=0).fit(X)**

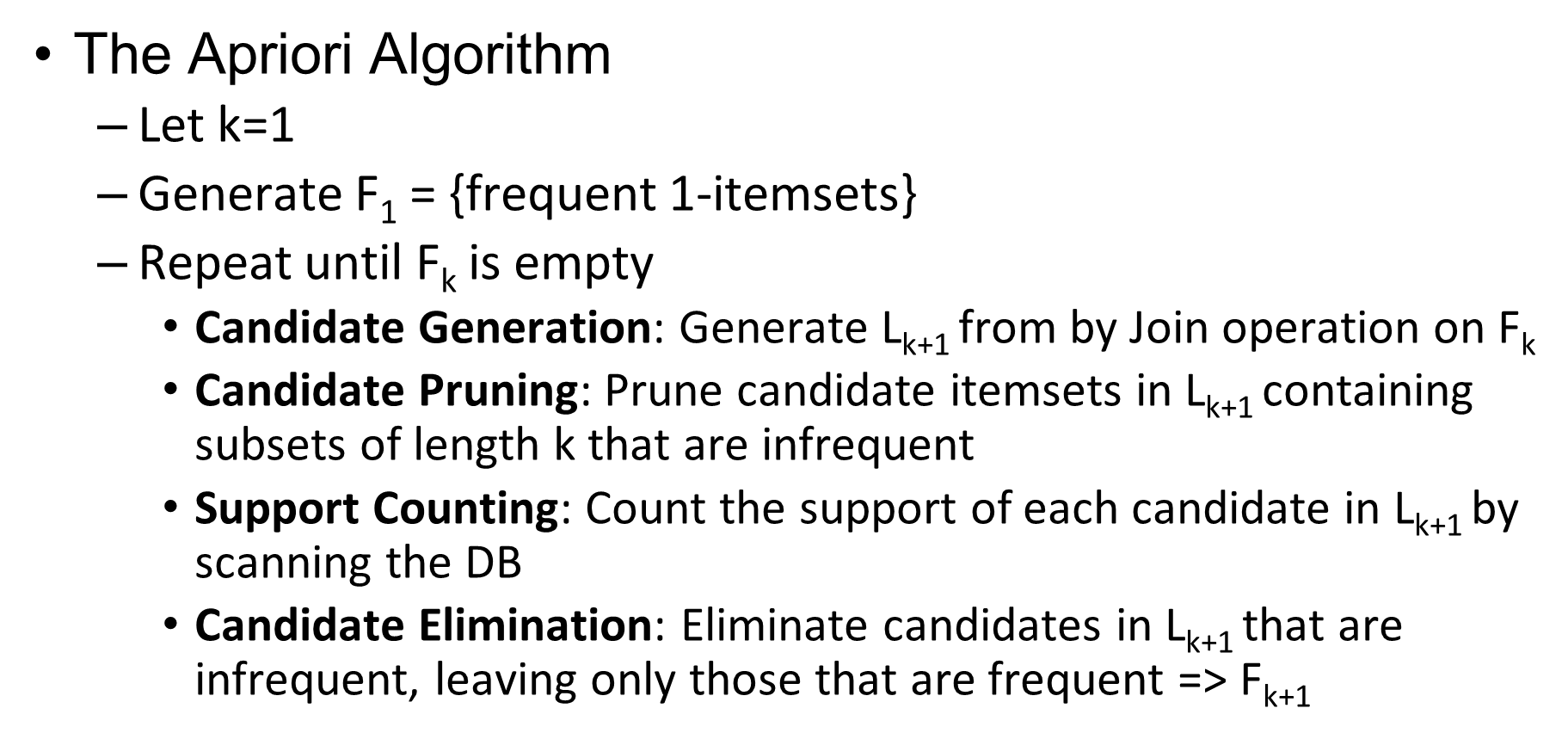
**KMeans(n\_clusters=2, random\_state=0).fit(X)**

In this Kmeans function, two different parameters are specified. In particular, the n\_clusters mean the number of clusters, and the random\_state decides the initial centroids. If we only specify these two parameters but not others, the results are reproducible for the Kmeans method on the same data.

Therefore, for the n\_clusters parameter, in both 1st and 2nd methods, we will group points into 3 clusters; but in the 3rd one, there will be only 2 clusters.

For the random\_state settings, the 1st and 2nd methods will have different 3 initial centroids, so the final clusters could be different. The 3rd method has the same random\_state as the 2nd method. However, since it only has 2 initial centroids, the final clustering result could also be different from the 2nd.

**Q2. We have the following transaction data. Please give the procedure running the given Apriori algorithm (the algorithm is shown below) on this data step-by-step.** **(50 points)**



**Answer 2:** Write your answer to Q2 here (Assume minimum support = 3)

Apriori Principle Step1: Generate frequent item set with minimum support requirement

Count the occurrences of 1 item:

Candidate 1-itemsets

|  |  |
| --- | --- |
| Itemset | Count |
| Bread | 4 |
| Milk | 4 |
| Diaper | 4 |
| Beer | 3 |
| ~~Cola~~ | ~~2~~ |
| ~~Egg~~ | ~~1~~ |

Frequent 1-itemsets

|  |  |
| --- | --- |
| Itemset | Count |
| Bread | 4 |
| Milk | 4 |
| Diaper | 4 |
| Beer | 3 |

Look for frequent occurrences of 2 items

Pairs (2-itemsets) (No need to generate candidates involving Cola or Eggs)

Candidate 2-itemsets

|  |  |
| --- | --- |
| Itemset | Count |
| Bread, Milk | 4 |
| Bread, Diapers | 3 |
| ~~Bread, Beer~~ | ~~2~~ |
| Milk, Diapers | 3 |
| ~~Milk, Beer~~ | ~~2~~ |
| Diaper, Beer | 3 |

Frequent 2-itemsets

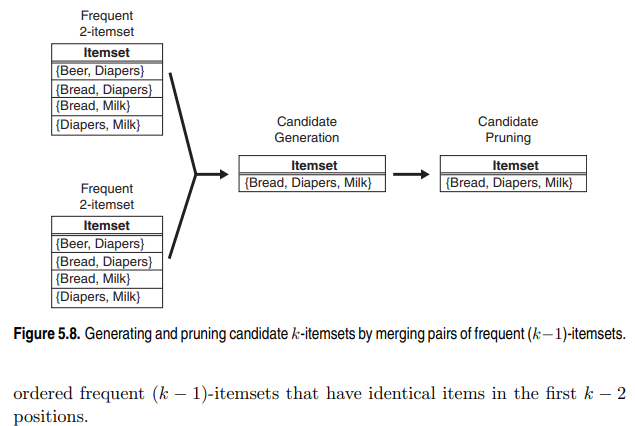
|  |  |
| --- | --- |
| Itemset | Count |
| Bread, Milk | 4 |
| Bread, Diapers | 3 |
| Milk, Diapers | 3 |
| Diaper, Beer | 3 |

Look for frequent occurrences of 3 items

Triplets (3-itemsets)

Candidate 3-itemsets (Fk−1×Fk−1 Method)

This candidate generation procedure, which is used in the candidate-gen function of the Apriori algorithm, merges a pair of frequent (k −1)-itemsets only if their first k −2 items, arranged in lexicographic order, are identical.



|  |  |
| --- | --- |
| Itemset | Count |
| Bread, Milk, Diapers | 2 |

Candidate 3-itemsets (Fk−1×F1 Method)

|  |  |
| --- | --- |
| Itemset | Count |
| Bread, Milk, Diapers | 2 |
| Bread, Diapers, Beer | 2 |
| Milk, Diapers, Beer | 2 |
| Beer, Bread, Milk | 1 |

{Beer, Bread, Diapers} and {Beer, Bread, Milk} will not be generated because {Beer, Bread} is not a frequent 2-itemset.

Frequent 3-itemsets

Null

Apriori Principle Step2: Generate rules with minimum confidence requirement

|  |  |  |
| --- | --- | --- |
| Candidate rules | | |
| B->M | B->D | B->MD |
| M->B | M->D | M->BD |
| D->B | D->M | D->BM |
| BM->D | MD->B | BD->M |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Rules | Confidence | Rules | Confidence | Rules | Confidence |
| B->M |  | B->D |  | B->MD |  |
| M->B |  | M->D |  | M->BD |  |
| D->B |  | D->M |  | D->BM |  |
| BM->D |  | MD->B |  | BD->M |  |

**T2-Q1**

**Calculate the Support and Confidence of “Rule {Apple, Banana -> Donut}”**

|  |  |  |
| --- | --- | --- |
| TID | DATE | ITEM |
| 1 | DAY1 | {Apple, Fish, Donut, Banana} |
| 2 | DAY2 | {Donut, Banana, Apple, Carrot, Egg} |
| 3 | DAY3 | {Carrot, Banana, Apple, Egg} |
| 4 | DAY4 | {Banana, Donut, Apple} |

# {Apple, Banana} = 4 # T = 4

# {Apple, Banana, Dount} = 3

Support = # {Apple, Banana, Dount} / # T = 3/4

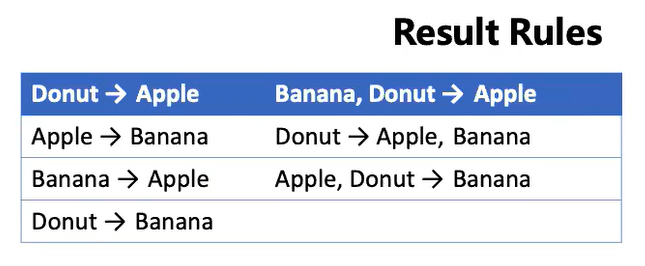
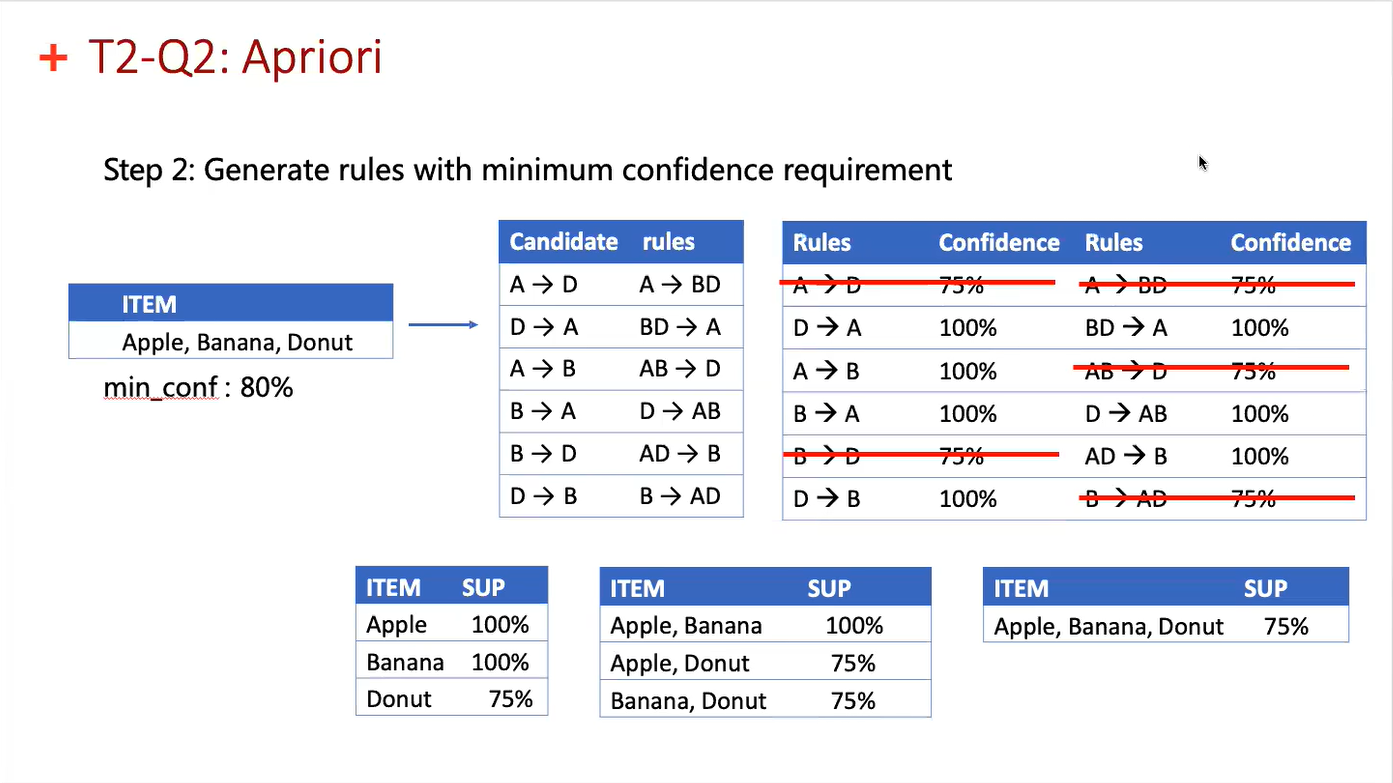
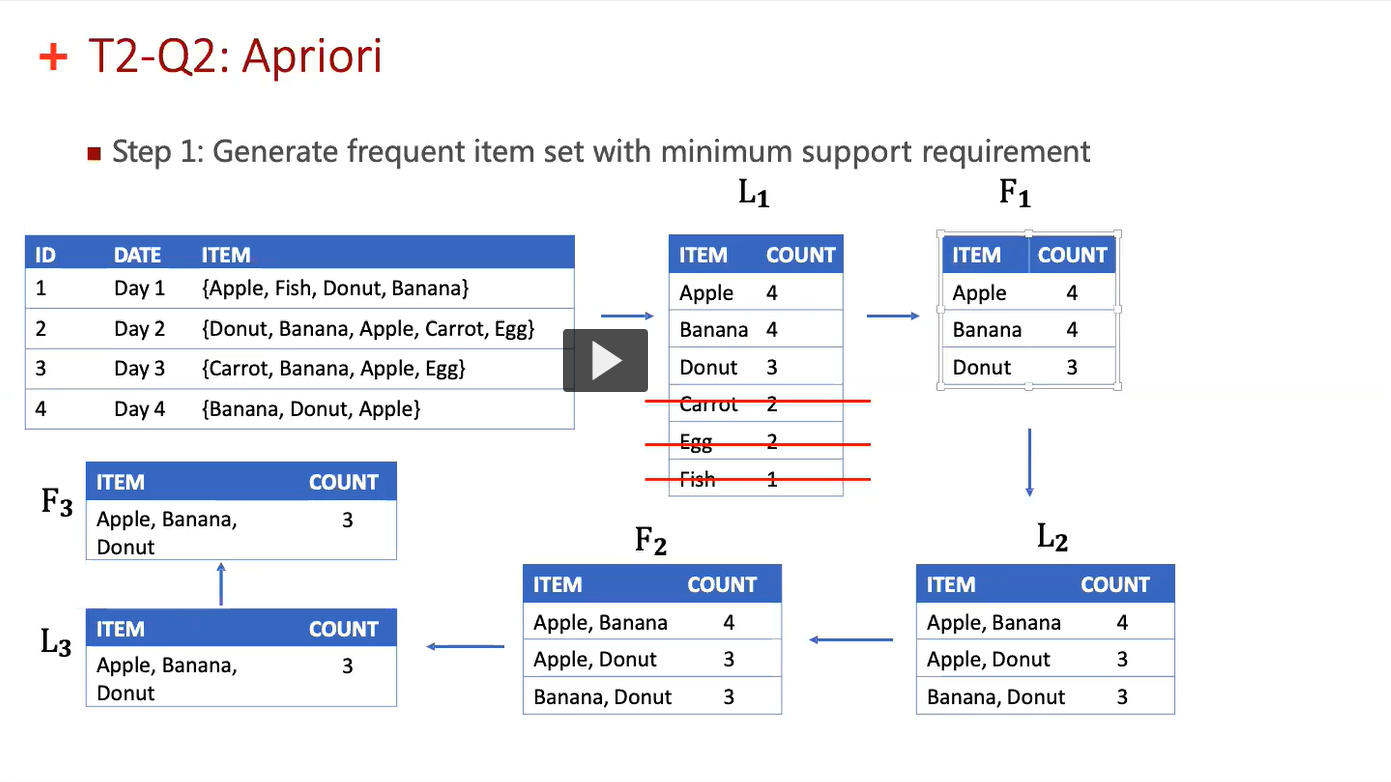
Confidence = # {Apple, Banana, Dount} / # {Apple, Banana} = 3/4

**T2-Q2**

**Finding all strong association rules with Apriori:**

|  |  |  |
| --- | --- | --- |
| TID | DATE | ITEM |
| 1 | DAY1 | {Apple, Fish, Donut, Banana} |
| 2 | DAY2 | {Donut, Banana, Apple, Carrot, Egg} |
| 3 | DAY3 | {Carrot, Banana, Apple, Egg} |
| 4 | DAY4 | {Banana, Donut, Apple} |

Requirement: min\_sup: 60% & min\_conf: 80%



**T4-Q1: Calculating distance**

**Calculate the L1, L2, L-infinite norms of two points: (1, 0 ,5) and (2, 4, 9)**

* d is number of attributes
* xiu is the u-th element of vector xi
* xju is the u-th element of vector xj

**𝐿1distance (Manhattan distance)**

𝑝=1

For (1, 0 ,5) and (2, 4, 9)

* d = 3
* 𝑥𝑖1=1, 𝑥𝑖2=0, 𝑥𝑖3=5
* 𝑥𝑗1=2, 𝑥𝑗2=4, 𝑥𝑗3=9
* dist(xi, xj)=|1−2|+|0−4|+|5−9|=9

**𝐿2distance (Euclidean distance)**

p=2

For (1, 0 ,5) and (2, 4, 9)

* d = 3
* 𝑥𝑖1=1, 𝑥𝑖2=0, 𝑥𝑖3=5
* 𝑥𝑗1=2, 𝑥𝑗2=4, 𝑥𝑗3=9
* dist(xi, xj)=(|1−2|2+|0−4|2+|5−9|2)1/2=sqrt(33)

**𝐿∞distance (Chebyshev distance)**

p=∞

For (1, 0 ,5) and (2, 4, 9)

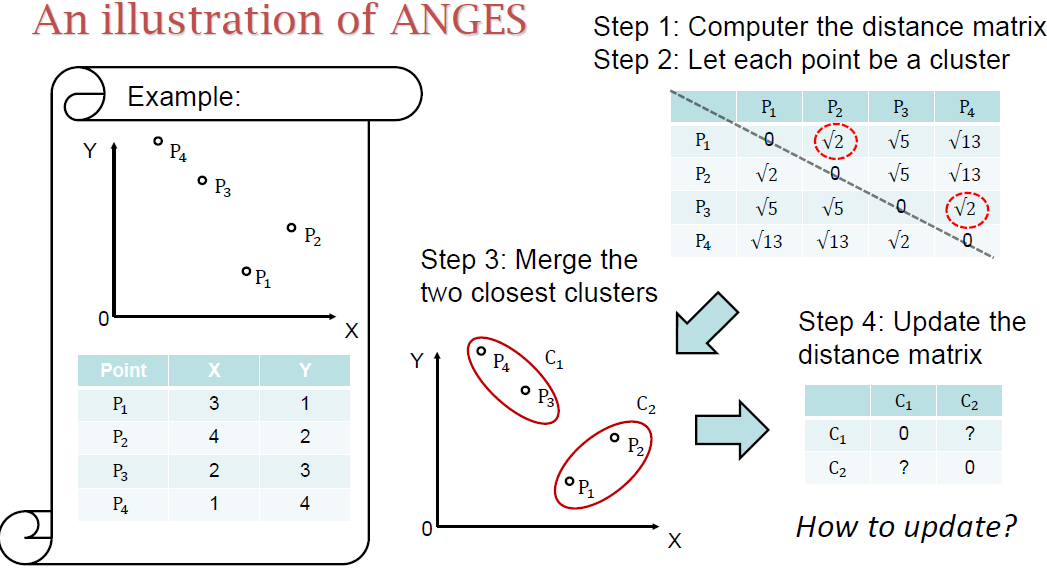
* d = 3
* 𝑥𝑖1=1, 𝑥𝑖2=0, 𝑥𝑖3=5
* 𝑥𝑗1=2, 𝑥𝑗2=4, 𝑥𝑗3=9
* dist(xi, xj)=max(|1−2|+|0−4|+|5−9|)=4

**T4-Q2: AGNES algorithm**

**Suppose the data mining task is to use agglomerative clustering to group measurements of the variable age.**

**Age = {18, 28, 22, 33, 40, 48}**

**Commute the step-by-step agglomerative grouping using:**

**a. Single linkage**

**b. Complete linkage**

**c. Average linkage**

**with Manhattan distance**

*Algorithm*

*1. Compute the distance matrix*

*2. Let each data point be a cluster*

***Repeat***

*3. merge the two closest clusters*

*4. update the distance matrix*

***Until*** *only a single cluster remains*

Step 1: Compute the distance matrix

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 18 | 28 | 22 | 33 | 40 | 48 |
| 18 | 0 | 10 | 4 | 15 | 22 | 30 |
| 28 | 10 | 0 | 6 | 5 | 12 | 20 |
| 22 | 4 | 6 | 0 | 11 | 18 | 26 |
| 33 | 15 | 5 | 11 | 0 | 7 | 15 |
| 40 | 22 | 12 | 18 | 7 | 0 | 8 |
| 48 | 30 | 20 | 26 | 15 | 8 | 0 |

Single linkage: Round 1

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 18 | 28 | 22 | 33 | 40 | 48 |
| 18 | 0 | 10 | 4 | 15 | 22 | 30 |
| 28 | 10 | 0 | 6 | 5 | 12 | 20 |
| 22 | 4 | 6 | 0 | 11 | 18 | 26 |
| 33 | 15 | 5 | 11 | 0 | 7 | 15 |
| 40 | 22 | 12 | 18 | 7 | 0 | 8 |
| 48 | 30 | 20 | 26 | 15 | 8 | 0 |

Merge 18 and 22

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 18,22 | 28 | 33 | 40 | 48 |
| 18,22 | 0 | 6 | 11 | 18 | 26 |
| 28 | 6 | 0 | 5 | 12 | 20 |
| 33 | 11 | 5 | 0 | 7 | 15 |
| 40 | 18 | 12 | 7 | 0 | 8 |
| 48 | 26 | 20 | 15 | 8 | 0 |

Merge 33 and 28

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 18,22 | 28,33 | 40 | 48 |
| 18,22 | 0 | 6 | 18 | 26 |
| 28,33 | 6 | 0 | 7 | 15 |
| 40 | 18 | 7 | 0 | 8 |
| 48 | 26 | 15 | 8 | 0 |

Merge {18, 22} and {28, 33}

|  |  |  |  |
| --- | --- | --- | --- |
|  | 18,22,28,33 | 40 | 48 |
| 18,22,28,33 | 0 | 7 | 15 |
| 40 | 7 | 0 | 8 |
| 48 | 15 | 8 | 0 |

Merge {18, 22, 28, 33} and 40

|  |  |  |
| --- | --- | --- |
|  | 18,22,28,33,40 | 48 |
| 18,22,28,33,40 | 0 | 8 |
| 48 | 8 | 0 |

**T4-Q3: k-means and SSE calculation**

**Suppose the data mining task is to cluster the following measurements**

**Age = {18, 22, 25, 42, 27, 43, 33, 35, 56, 28} into three groups. For initial centroid of {18, 27, 35}:**

**a. Use the k-means algorithm to show the clustering procedures step by step;**

**b. Calculate the final SSE using Manhattan distance**

*K-means algorithm*

*1. --Select k points as the initial centroids*

*2. Repeat*

*3. --Form k clusters by assigning points to the closest centroid*

*4. --Update the centroid of each cluster*

*5. Until --The cluster assignment don’t change*

k-means: Round 1

|  |  |  |  |
| --- | --- | --- | --- |
| Cluster | Centroid | Cluster elements | Updated centroid |
| 1 | 18 | 18,22 | 20 |
| 2 | 27 | 25,27,28 | 26.7 |
| 3 | 35 | 33,35,42,43,56 | 41.8 |

k-means: Round 2

|  |  |  |  |
| --- | --- | --- | --- |
| Cluster | Centroid | Cluster elements | Updated centroid |
| 1 | 20 | 18,22 | 20 |
| 2 | 26.7 | 25,27,28,33 | 28.25 |
| 3 | 41.8 | 35,42,43,56 | 44 |

k-means: Round 3

|  |  |  |  |
| --- | --- | --- | --- |
| Cluster | Centroid | Cluster elements | Updated centroid |
| 1 | 20 | 18,22 | 20 |
| 2 | 28.25 | 25,27,28,33,35 | 29.6 |
| 3 | 44 | 42,43,56 | 47 |

k-means: Round 4

|  |  |  |  |
| --- | --- | --- | --- |
| Cluster | Centroid | Cluster elements | Updated centroid |
| 1 | 20 | 18,22 | 20 |
| 2 | 29.6 | 25,27,28,33,35 | 29.6 |
| 3 | 47 | 42,43,56 | 47 |

Calculate the SSE

= (20 - 18)2 + (22 - 20)2 + (25 – 29.6)2 + (27 – 29.6)2 + … + (56 – 47)2 = 201.2

**The density-based clustering algorithm: DBSCAN**

1. Label all points as noise, core or border points

2. Eliminate all noise points

3. **Initialization**: cluster\_index= 0

4. **for** all core points

5. **if** it is not assigned a cluster\_index

6. cluster\_index= cluster\_index+1

7. assign the core point the cluster\_index

8. **for** all points within the Eps circle of the core point

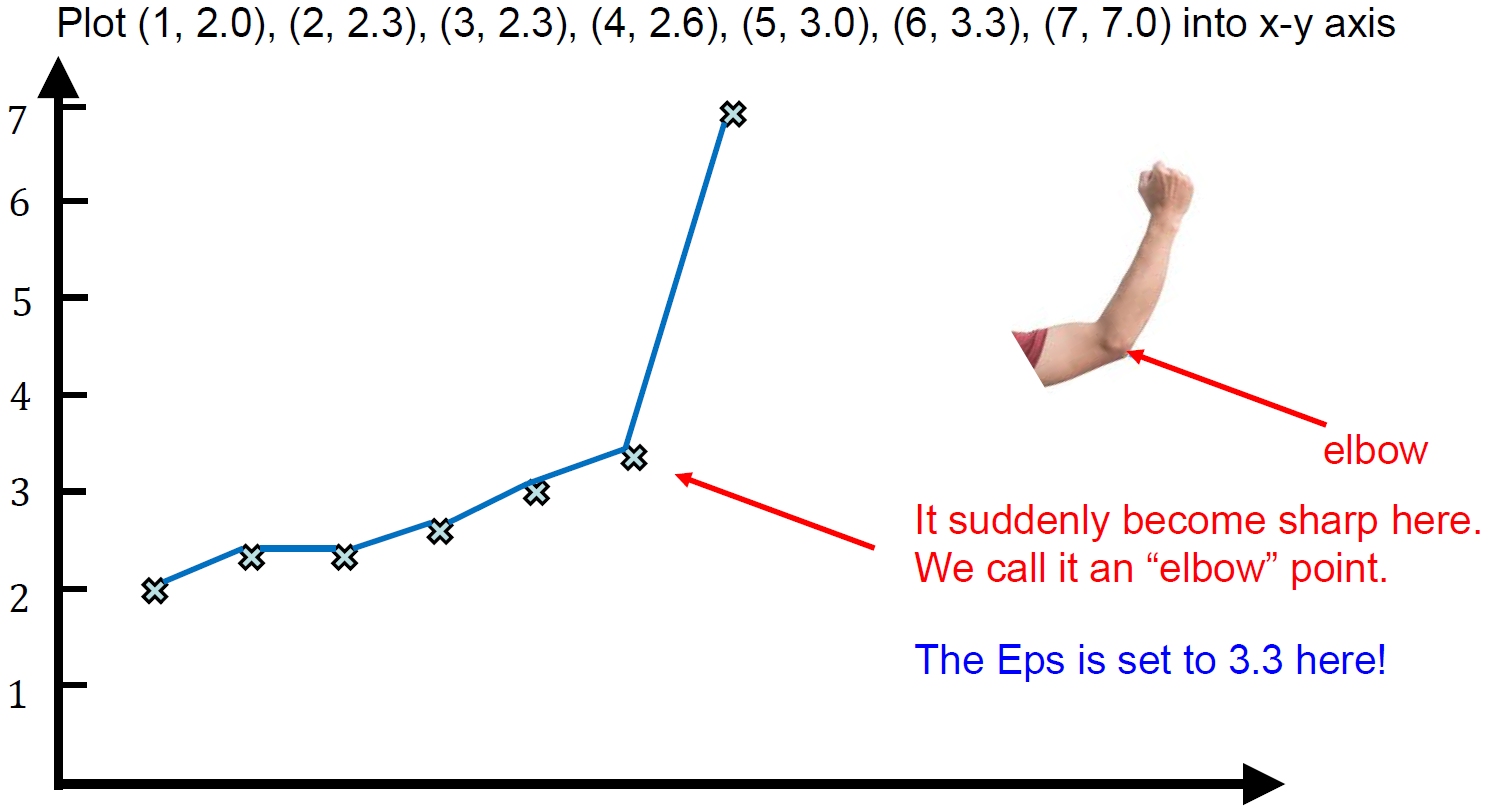
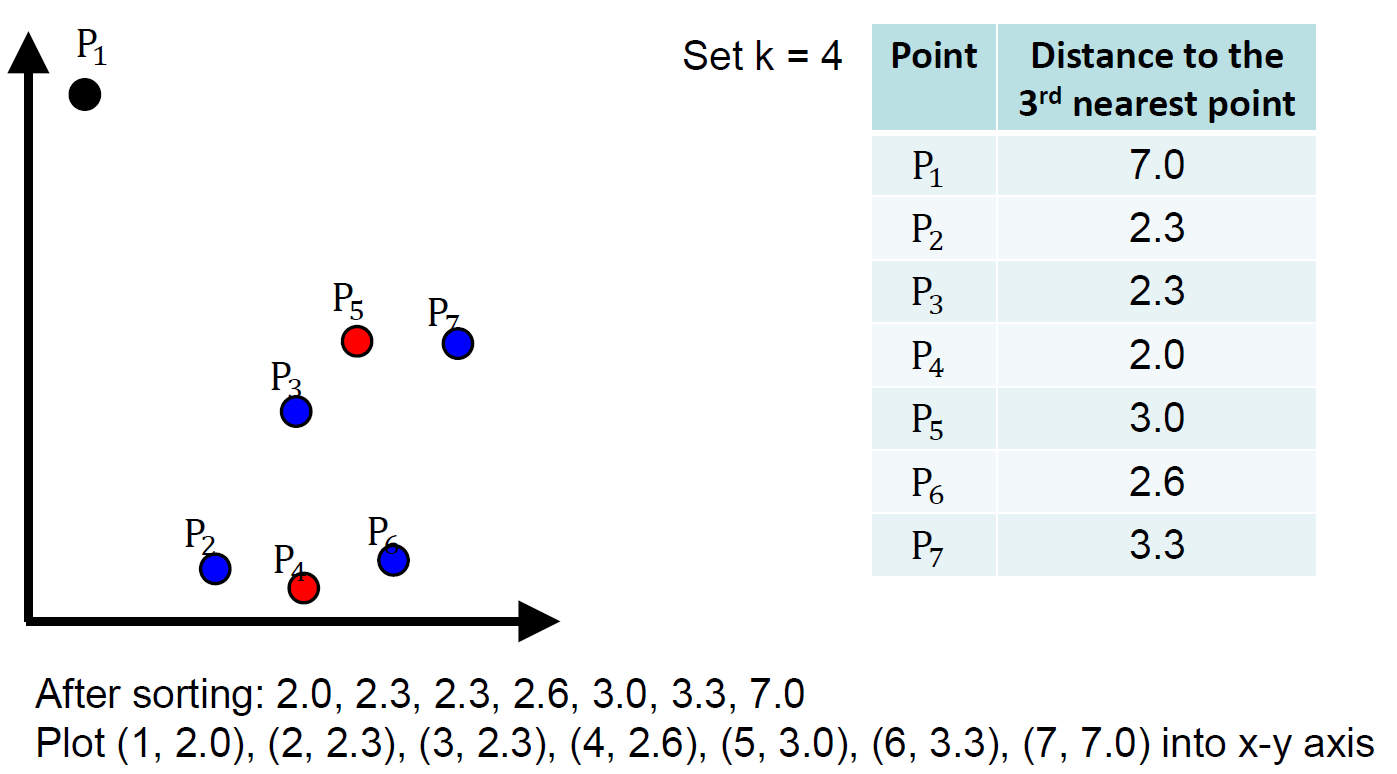
9. **if** it is not assigned a cluster\_index

10. assign the point the cluster\_index

**How to determine Eps and MinPts**

One recommended Elbow method:

* Fix MinPts to be k, (e.g., k=4)
* Calculate all points’ distances to their (k-1)th nearest point
* Sort the distance in ascending order and plot them
* Find the “elbow” point, whose corresponding distance is Eps



**Model-based approach: statistical method**

***univariate***

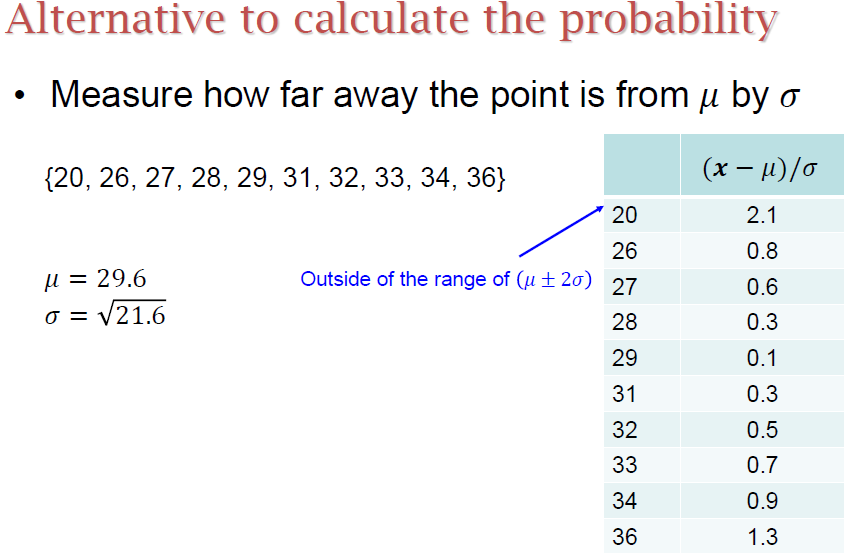
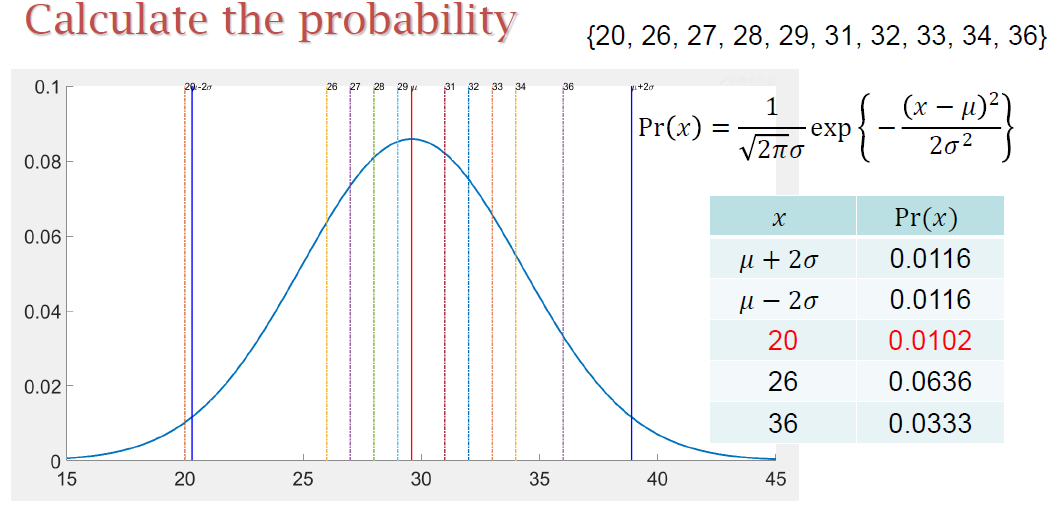
**Data:**

* **10 temperature data points collected in the summer**
* **{20, 26, 27, 28, 29, 31, 32, 33, 34, 36}**

Statistical model: Gaussian model

* : mean(29.6):
* : variance (21.6):

Outlier:



***multivariate***

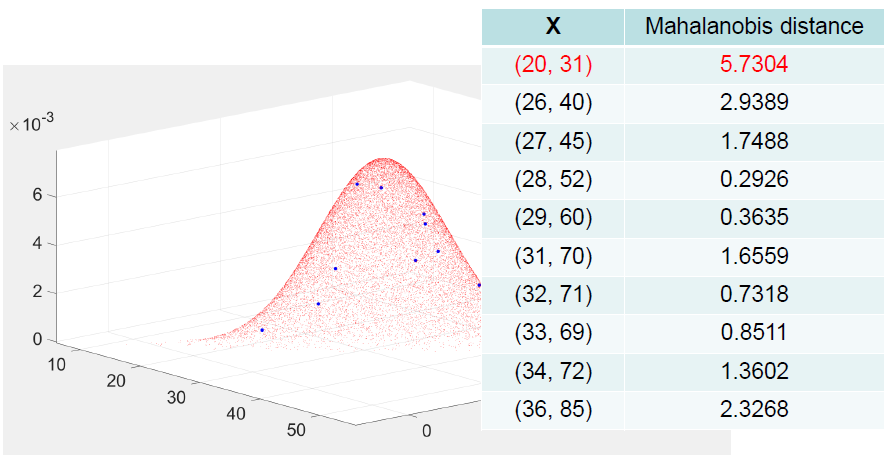
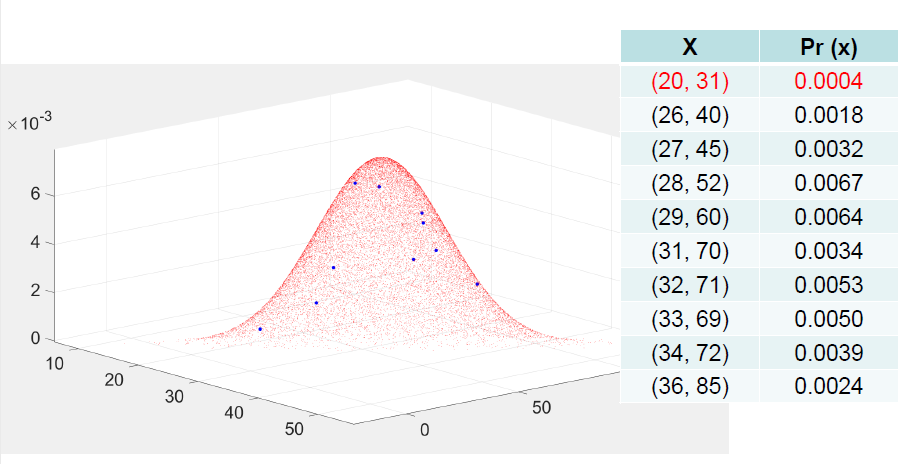
**Data:**

* **10 (temperature, humidity\*100) data collected: {(20, 31), (26, 40), (27, 45), (28, 52), (29, 60), (31, 70), (32, 71), (33, 69), (34, 72), (36, 85)}**

Statistical model

: mean: = (29.6, 59.5)

Outlier:



Alternative to calculate the probability

What really do calculation with x? also called “mahalanobis distance”

If the mahalanobis distance to “mean” is large (larger than a threshold): outlier

*Anomaly detection by statistical method for {0,12,15,27,46}*

* *Using the univariate gaussian distribution*
* *If is counted as “anomaly”, which points are anomaly points?*

Mean: (0+12+15+27+46)/5=20

Variance (0−20)2 + (12−20)2 + (15−20)2 + (27−20)2 + (46−20)2) / 4 = 303.5

Distance to mean measured by variance:

**Density-based technique**

**1/(the average distance from any point in the k-nearest neighborhood of 𝒙 to 𝒙)**

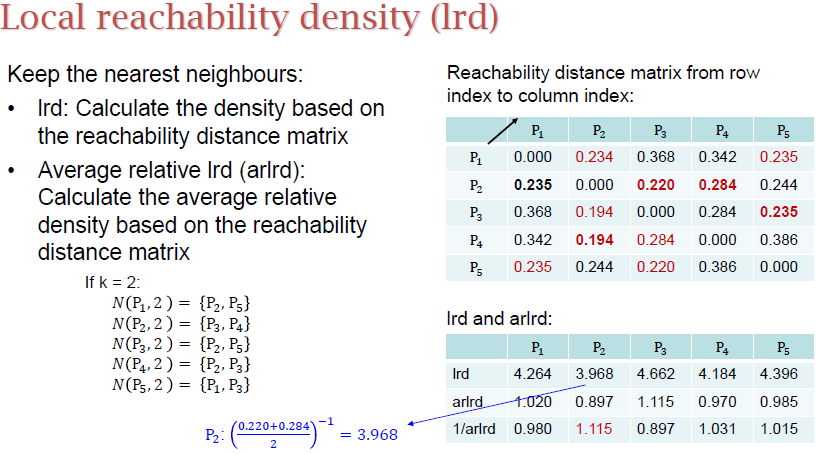
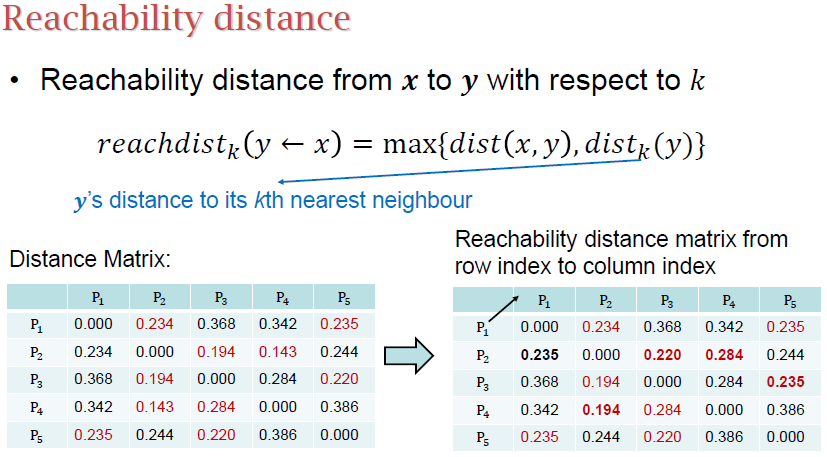
Distance Matrix:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | P1 | P2 | P3 | P4 | P5 |
| P1 | 0.000 | 0.234 | 0.368 | 0.342 | 0.235 |
| P2 | 0.234 | 0.000 | 0.194 | 0.143 | 0.244 |
| P3 | 0.368 | 0.194 | 0.000 | 0.284 | 0.220 |
| P4 | 0.342 | 0.143 | 0.284 | 0.000 | 0.386 |
| P5 | 0.235 | 0.244 | 0.220 | 0.386 | 0.000 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | P1 | P2 | P3 | P4 | P5 |
| Density | 4.264 | 5.935 | 4.831 | 4.684 | 4.396 |
| ard | 0.825 | 1.248 | 0.935 | 0.870 | 0.967 |

Average relative density

* The density relative to the average density within the neighborhood



**Distance-based technique – to kth nearest**

**Cluster-based technique – to centroid**

**Limitation of k-means**

NFL theorem:

* No free lunch theorem
* If you work well on some data, there must exist some data you cannot work well (you need to pay for what you achieve).
* Prototype-based clustering (k-means)
* Easy and simple
* Need to specify k, cannot deal with non-globular shape
* Density-based clustering (DBSCAN)
* Can deal with non-globular shape
* Need to specify MinPts and Eps, density-sensitive
* Hierarchical-based clustering(AGNES)
* Hierarchical structure, deal with non-globular shape
* Select within Min, Max, Average, each has its own disadvantage
  + Min is best at handling non-elliptical shapes
  + Max tends to form globular shape
  + Min is sensitive to noise
  + Max tends to break large clusters

**Density-based Technique**

**Calculate the Local Outlier Factor (LOF)**

*Given four points: 𝑃1(1,0), 𝑃2(2,0), 𝑃3(1,1), 𝑃4(2,2.5). Calculate the Local Outlier Factor (LOF) for each point and find the top-1 outliers. Use a k value of 2 and Euclidean Distance as the distance function.*

Distance Matrix:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | P1 (1,0) | P2 (2,0) | P3 (1,1) | P4 (2,2.5) |
| P1 (1,0) | 0 | 1 | 1 | 2.693 |
| P2 (2,0) | 1 | 0 | 1.414 | 2.5 |
| P3 (1,1) | 1 | 1.414 | 0 | 1.803 |
| P4 (2,2.5) | 2.693 | 2.5 | 1.803 | 0 |

If k = 2:

N(P1, 2) = {P2, P3}

N(P2, 2) = {P1, P3}

N(P3, 2) = {P1, P2}

N(P4, 2) = {P2, P3}

Distance Matrix:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | P1 (1,0) | P2 (2,0) | P3 (1,1) | P4 (2,2.5) |
| density | 1 | 0.8290 | 0.8290 | 0.4650 |