Northeastern University, Khoury College of Computer Science

CS 6220 Data Mining | Assignment 4

Due: March 1, 2023 (100 points)

Mengfei Zhang
ZMF01
zhang.mengfei@northeastern.edu

K-Means

The normalized automobile distributor timing speed and ignition coil gaps for production F-150 trucks over the years of 1996, 1999, 2006, 2015, and 2022. We have stripped out the labels for the five years of data.

Each sample in the dataset is two-dimensional, i.e. $x_i \in \mathbb{R}^2$(one dimension for timing speed and the other for coil gaps), and there are N = 5000 instances in the data.

Question 1 [20 pts total]

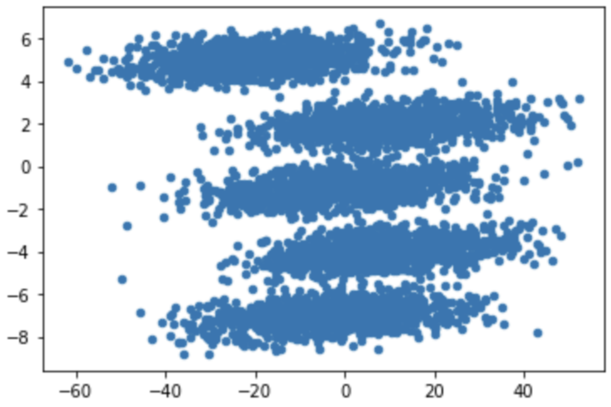
[10 pts] Question 1a.) Implement a simple k-means algorithm in Python on Colab with the following initialization:

$$x_1 = \begin{pmatrix}10\\10\end{pmatrix}, x_2 = \begin{pmatrix}-10\\-10\end{pmatrix}, x_3 = \begin{pmatrix}2\\2\end{pmatrix}, x_4 = \begin{pmatrix}3\\3\end{pmatrix}, x_5 = \begin{pmatrix}-3\\-3\end{pmatrix},$$

You need only 100 iterations, maximum, and your algorithm should run very quickly to get the results.

```python
import numpy as np
import matplotlib.pyplot as plt
import matplotlib as mpl
```

```python
X = np.loadtxt("f150_motor_distributors.txt",delimiter=",")
plt.scatter(X[:, 0], X[:, 1], s=20)
plt.show()
```



```python
def findClosestCentroids(X, centroids):
    idx = np.zeros(len(X))
    for i in range(len(X)):
        minDistance=float('inf')
        index=0

        for k in range(len(centroids)):
            # Euclidean distance
            distance=np.sum(np.power(X[i]-centroids[k],2))
            if(distance<minDistance):
                minDistance=distance
                index=k
                # Minimum distance from the k-th center point
        idx[i]=index
    return idx

# Recalculate center point position
def computeCentroids(X, idx):
    # Find all cluster center indexes
    k = set(np.ravel(idx).tolist())
    k = list(k)
    centroids = np.ndarray((len(k),X.shape[1]))

    for i in range(len(k)):
    # Select the data with category k [i] in data X and select by row
        data = X[np.where(idx==k[i])[0]]
        #Recalculate the cluster center, axis=0 is row compression,
        #add each column of the matrix to form a one-dimensional array,
        #and divide by the number of rows of the original matrix to get the cluster center
        centroids[i] = np.sum(data,axis=0)/len(data)
    return centroids
```

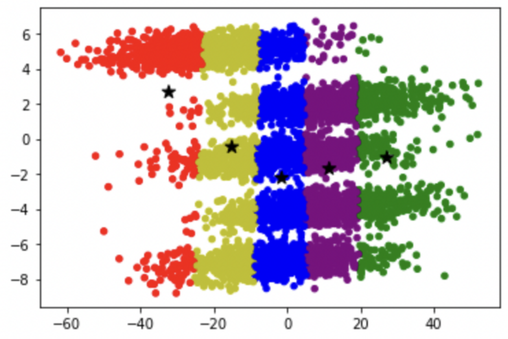
```
def k_means(X, k, max_iters):

  # UsingUsing given initialization
    initial_centroids = np.array([[10, 10], [-10, -10], [2, 2], [3, 3], [-3, -3]])

    for i in range(max_iters):
        if(i==0):
            centroids=initial_centroids
        #Calculate the distance from the sample to the cluster center,
        #and return the cluster center to which each sample belongs
        idx=findClosestCentroids(X,centroids)
        #Recalculate cluster center
        centroids=computeCentroids(X,idx)
    return idx,centroids
```

```
# 100 iterations
idx,centroids = k_means(X, 5, 100)
print(idx)
print(centroids)
```

```
[4. 1. 4. ... 3. 1. 1.]
[[ 27.26677403  -1.08848482]
 [-32.27032272   2.65984149]
 [ -1.45401192  -2.23932918]
 [ 11.48737759  -1.70898344]
 [-15.05895772  -0.47281415]]
```

[5 pts] Question 1b.) Scatter the results in two dimensions with different clusters as different colors.

```
colors = mpl.colors.ListedColormap(['g', 'r', 'b','purple','y'])
c=mpl.colors.ListedColormap('k')
#colors = ['red', 'blue', 'green', 'yellow', 'purple']
plt.scatter(X[:, 0], X[:, 1], c=np.ravel(idx), cmap=colors, s=20)
plt.scatter(centroids[:, 0], centroids[:, 1], c=np.arange(len(centroids)), cmap=c, marker='*', s=100)
plt.show()
```



[5 pts] Question 1c.) You will notice that in the above, there are only five initialization clusters. Why is $k = 5$ a logical choice for this dataset? After plotting your resulting clusters, what do you notice? Did it cluster very well? Is there an initialization that would make it cluster well?

- Why is $k = 5$ a logical choice for this dataset?

  Because based on the scatter plot of our dataset, we can notice that these data points make 5 clusters clearly. That's why k = 5 is a logical choice for this dataset.

- After plotting your resulting clusters, what do you notice? Did it cluster very well?

  No, it did not cluster very well. Because after plotting, I notice that each cluster is sparsely distributed, not dense.

- Is there an initialization that would make it cluster well?

  I don't think there is an initialization that would make it cluster well. Because it will always show a similar result by using this method, which shows each cluster is vertically sparsely distributed, not horizontally densely distributed.

Question 2 [30 pts total]

In the data from Question 1, let x and y be two instances, i.e., they are each truck with separate measurements. A common distance metric is the Mahalanobis Distance with a specialized matrix $P \in \mathbb{R}^{2\times 2}$ that is written as follows:

$$R = \left(P^T P\right)^{-1}$$

$$d(x, y) = (x - y)^T R(x - y)$$

In scalar format (non-matrix format), the Mahalanobis Distance can be expressed as:

$$d(x, y) = \sum_{i=1}^{2} \sum_{j=1}^{2} (x_i - y_i) \cdot P_{i,j}^{-1} \cdot (x_j - y_j)$$

where x and y are two instances of dimensionality 2, and d(x, y) is the distance between them. In the case of the F150 engine components, P is a known relationship through Ford's quality control analysis each year, where it is numerically shown as below:

$$P = \begin{pmatrix} 10 & 0.5 \\ -10 & 0.25 \end{pmatrix}$$

[15 pts] Question 2a.) Using the same data as Question 1 and the same initialization instances $\{x_1, x_2, x_3, x_4, x_5\}$implement a specialized k-means with the above Mahalanobis Distance. Scatter the results with the different clusters as different colors.

What do you notice? You may want to pre-compute P^{-1} so that you aren't calculating an inverse every single loop of the k-Means algorithm.

```python
P = np.array([[10, 0.5], [-10, 0.25]])
factor = np.linalg.inv(P.T @ P)

def mahalanobis_distance(x, y, factor):
    return (x - y).T @ factor @ (x - y)
```

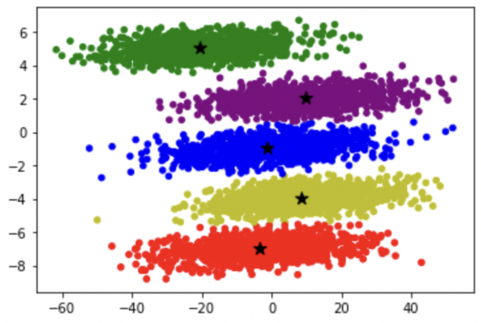
```python
def findClosestCentroids_MD(X, centroids):
    idx = np.zeros(len(X))

    for i in range(len(X)):
        minDistance=float('inf')
        index=0
        for k in range(len(centroids)):
            # mahalanobis distance
            distance = mahalanobis_distance(centroids[k],X[i],factor)
            if(distance<minDistance):
                minDistance=distance
                index=k
        idx[i]=index
    return idx

def computeCentroids(X, idx):
    k = set(np.ravel(idx).tolist())
    k = list(k)
    centroids = np.ndarray((len(k),X.shape[1]))

    for i in range(len(k)):
        data = X[np.where(idx==k[i])[0]]
        centroids[i] = np.sum(data,axis=0)/len(data)
    return centroids
```

```python
def k_means_MD(X, k):
    internation = 1000
    centroids = np.array([[10, 10], [-10, -10], [2, 2], [3, 3], [-3, -3]])

    for i in range(internation):
        idx=findClosestCentroids_MD(X,centroids)
        new_centroids=computeCentroids(X,idx)
        if np.array_equal(new_centroids, centroids):
            break
        centroids = new_centroids
    return idx,centroids
```

```python
idx1,centroids1 = k_means_MD(X, 5)
print(idx1)
print(centroids1)
```

```
[1. 1. 2. ... 2. 0. 2.]
[[-20.40310599   5.00655283]
 [ -3.24764066  -7.01766444]
 [ -1.09670369  -0.9992032 ]
 [  9.97451285   2.00207148]
 [  8.71408722  -4.00981947]]
```

```
[27] colors = mpl.colors.ListedColormap(['g', 'r', 'b','purple','y'])
     star=mpl.colors.ListedColormap('k')
     plt.scatter(X[:, 0], X[:, 1], c=np.ravel(idx1), cmap=colors, s=20)
     plt.scatter(centroids1[:, 0], centroids1[:, 1], c=np.arange(len(centroids1)), cmap=star, marker='*', s=100)
     plt.show()
```

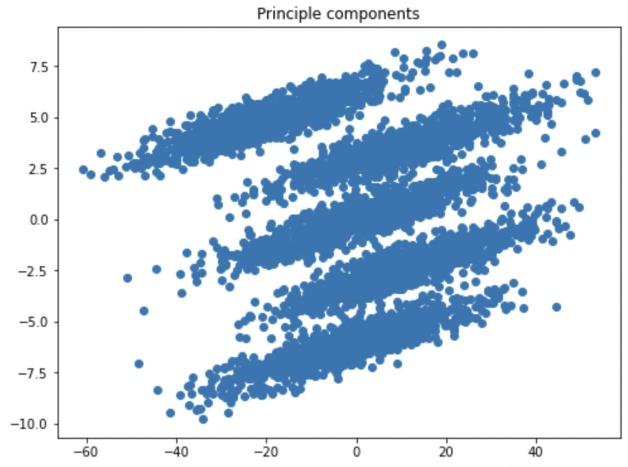


Answer: from the figure above, I notice that the result is much better than Question 1.

[5 pts] Question 2b.) Calculate and print out the principle components of the aggregate data.

```python
def pca(X,k):
  #mean of each feature
  n_samples, n_features = X.shape
  mean=np.array([np.mean(X[:,i]) for i in range(n_features)])

  norm_X=X-mean
  scatter_matrix=np.dot(np.transpose(norm_X),norm_X)

  #Calculate the eigenvectors and eigenvalues
  eig_val, eig_vec = np.linalg.eig(scatter_matrix)

  eig_pairs = [(np.abs(eig_val[i]), eig_vec[:,i]) for i in range(n_features)]
  eig_pairs.sort(reverse=True)

  feature=np.array([ele[1] for ele in eig_pairs[:k]])
  #get new data
  data = np.dot(norm_X,np.transpose(feature))
  #print(eig_pairs)

  print("eig_vec:")
  print(eig_vec)
  print("eig_val:")
  print(eig_val)
  print("data:")
  print(data)
  fig = plt.figure(figsize=(8, 6))
  plt.scatter(data[:, 0], data[:, 1])
  plt.title('Principle components')
  plt.show()

X = np.loadtxt("f150_motor_distributors.txt",delimiter=",")
print(pca(X,2))
```

```
eig_vec:
[[ 0.99838317  0.05684225]
 [-0.05684225  0.99838317]]
eig_val:
[1612213.15649735   86924.89062219]
data:
[[-10.33254906  -7.6356903 ]
 [-25.36097916  -7.41205973]
 [-11.65548133  -1.03953719]
 ...
 [ 14.07157308   1.70302537]
 [-33.3705919    3.51819072]
 [-25.51506544  -2.09006015]]
```

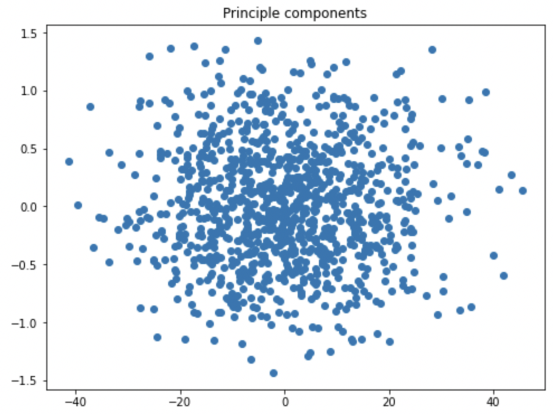


[5 pts] Question 2c.) Calculate and print out the principle components of each cluster. Are they the same as the aggregate data? Are they the same as each other?

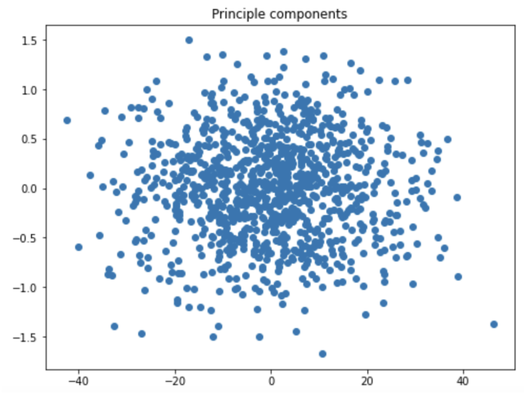
Answer: the principle components of each cluster are not the same as the aggregate data. They are similar to each other.

```
# For each cluster:
for i in range(5):
    indices = np.where(idx1 == i)
    pca(X[indices],2)
```

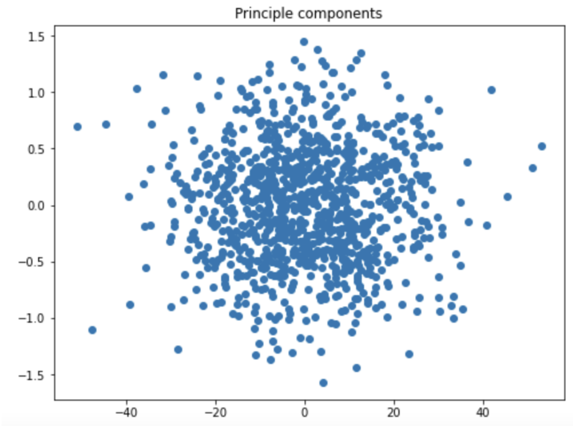
```
eig_vec:
[[ 0.99993527 -0.01137789]
 [ 0.01137789  0.99993527]]
eig_val:
[194830.89058105    271.93249514]
data:
[[  7.25405752  -0.25378307]
 [-18.11417956  -0.68829451]
 [ 27.23950349  -0.76561297]
 ...
 [  5.98078814   0.21199726]
 [  8.49049436   0.21735037]
 [-13.9396063   -0.44222502]]
```



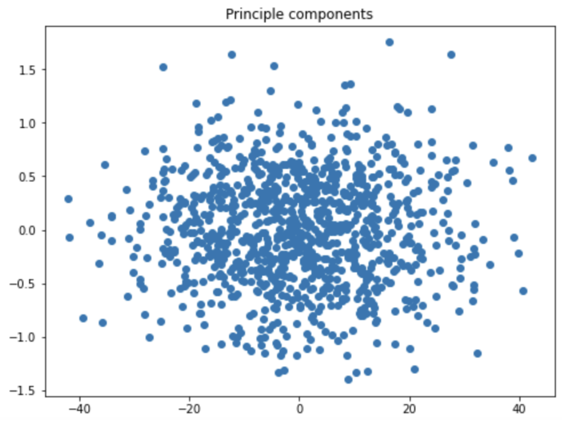
```
eig_vec:
[[ 0.99992533 -0.01222027]
 [ 0.01222027  0.99992533]]
eig_val:
[204172.43019429    283.10966196]
data:
[[ -8.73419248  -0.91529791]
 [-23.71132555   0.34533961]
 [  1.94266087   0.75298475]
 ...
 [  6.37240317  -0.70787872]
 [-15.07539706   0.44106933]
 [  0.90286284  -0.33686259]]
```



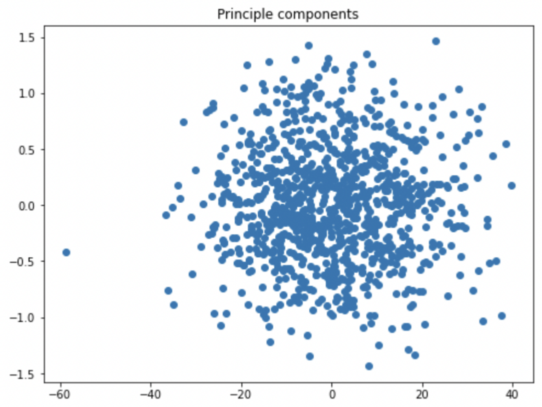
```
eig_vec:
[[ 0.99990986 -0.01342629]
 [ 0.01342629  0.99990986]]
eig_val:
[218001.47457169    277.53733718]
data:
[[-11.82317836  -0.22101684]
 [ -7.97251269  -0.07352572]
 [-21.79151461   0.27086357]
 ...
 [-15.03445302  -0.59867838]
 [ 14.03297252   0.70765947]
 [-25.72231983  -0.29542422]]
```



```
eig_vec:
[[ 0.99993306 -0.01157047]
 [ 0.01157047  0.99993306]]
eig_val:
[203808.38379012    268.96095496]
data:
[[ 12.50154109  -0.080114  ]
 [ 26.88083768   0.15701845]
 [-12.34056372  -0.61900941]
 ...
 [ 28.29556699  -0.14442832]
 [ -8.76609494   0.96638809]
 [ -3.12687729  -0.41684377]]
```



```
eig_vec:
[[ 0.99989374 -0.01457781]
 [ 0.01457781  0.99989374]]
eig_val:
[190961.02031395    262.76085431]
data:
[[-25.99638961  -0.96756674]
 [-58.71301722  -0.41374627]
 [ -8.42892305  -0.14539332]
 ...
 [  1.42435118  -0.82899984]
 [-24.31180609  -0.48532642]
 [ -0.59071656  -0.68142565]]
```



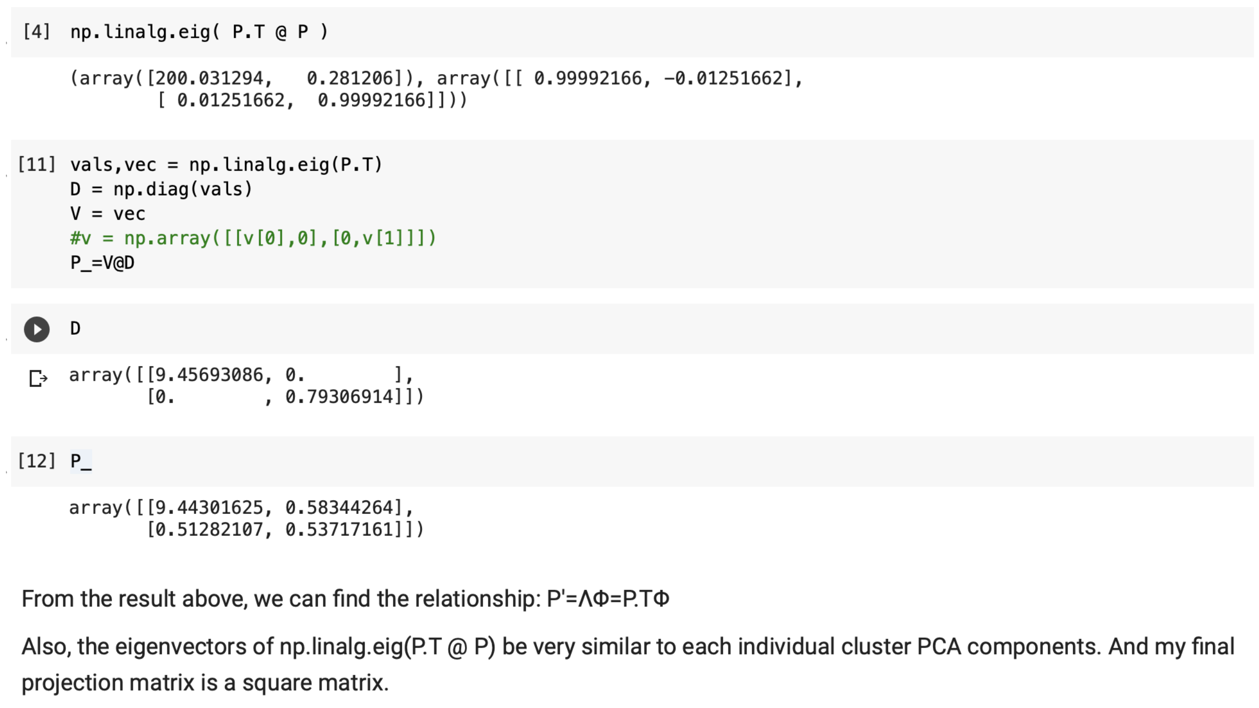
[5 pts] Question 2d.) Take the eigenvector / eigenvalue decomposition of P^T and subsequently, take their product. That is to say,

$$\{\Lambda, \Phi\} = eig\big(P^T\big)$$

Where $\Lambda = \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}$ and Φ is a 2×2 matrix with $\varphi_i \in \mathbb{R}^2$, a column in Φ. Calculate new P' such that

$$P' = \Lambda\Phi$$

What is the relationship between P' and the data?



```
[4]  np.linalg.eig( P.T @ P )

     (array([200.031294,   0.281206]), array([[ 0.99992166, -0.01251662],
             [ 0.01251662,  0.99992166]]))
```

```
[11] vals,vec = np.linalg.eig(P.T)
     D = np.diag(vals)
     V = vec
     #v = np.array([[v[0],0],[0,v[1]]])
     P_=V@D
```

```
     D

     array([[9.45693086, 0.        ],
            [0.        , 0.79306914]])
```

```
[12] P_

     array([[9.44301625, 0.58344264],
            [0.51282107, 0.53717161]])
```

From the result above, we can find the relationship: P'=ΛΦ=P.TΦ

Also, the eigenvectors of np.linalg.eig(P.T @ P) be very similar to each individual cluster PCA components. And my final projection matrix is a square matrix.

Market Basket Analysis and Algorithms

Consider F_3 as the following set of frequent 3-itemsets:

$$\{1,2,3\},\ \{1,2,4\},\ \{1,2,5\},\ \{1,3,4\}$$
$$\{2,3,4\},\ \{2,3,5\},\ \{3,4,5\}\,.$$

Assume that there are only five items in the data set.

Question 3 [25 pts total]

[10 pts] Question 3a.) List all candidate 4-itemsets obtained by a candidate generation procedure using the $F_{k-1} \times F_1$ merging strategy.

Using $F_{k-1} \times F_1$ merging strategy to get all candidate 4-itemsets, we need to merge the frequent itemsets of size 3 with all frequent itemsets of size 1 to generate F_4.

So, we merge the given sets of frequent 3-itemsets with F_1, which is {1}, {2}, {3}, {4}, {5}.

1. {1,2,3} merge with F_1, we get {1,2,3,4} and {1,2,3,5}
2. {1,2,4} merge with F_1, we get {1,2,4,5}
3. {1,2,5} merge with F_1, we can't get new 4-itemsets
4. {1,3,4} merge with F_1, we get {1,3,4,5}
5. {2,3,4} merge with F_1, we get {2,3,4,5}
6. {2,3,5} merge with F_1, we can't get new 4-itemsets
7. {3,4,5} merge with F_1, we can't get new 4-itemsets

Thus, F_4 = {1,2,3,4}, {1,2,3,5}, {1,2,4,5}, {1,3,4,5}, {2,3,4,5}

[10 pts] Question 3b.) List all candidate 4-itemsets obtained by the candidate generation procedure in A Priori, using $F_{k-1} \times F_{k-1}$.

Using $F_{k-1} \times F_{k-1}$ merging strategy to get all candidate 4-itemsets, we need to merge $F_3 \times F_3$ with the same first two items.

1. {1,2,3} merge with {1,2,4} = {1,2,3,4}
2. {1,2,3} merge with {1,2,5} = {1,2,3,5}
3. {1,2,4} merge with {1,2,5} = {1,2,4,5}
4. {2,3,4} merge with {2,3,5} = {2,3,4,5}

Thus, F_4 = {1,2,3,4}, {1,2,3,5}, {1,2,4,5}, {2,3,4,5}.

[5 pts] Question 3c.) List all candidate 4-itemsets that survive the candidate pruning step of the Apriori algorithm.

From Question 3b, we get F_4 = {1,2,3,4}, {1,2,3,5}, {1,2,4,5}, {2,3,4,5}. Then, we need to check if its subsets of size 3 are frequent. If it has a subset that is not frequent, it will be pruned.

1. {1,2,3,4} subsets: {1,2,3}, {1,2,4}, {1,3,4}, {2,3,4} are all frequent, so don't prune
2. {1,2,3,5} subsets: {1,2,3}, {1,2,5}, {1,3,5}, {2,3,5}. {1,3,5} is not frequent, so {1,2,3,5} is pruned.
3. {1,2,4,5} subsets: {1,2,4}, {1,2,5}, {1,4,5}, {2,4,5}. {2,4,5} is not frequent, so {1,2,4,5} is pruned.
4. {2,3,4,5} subsets: {2,3,4}, {2,3,5}, {2,4,5}, {3,4,5}. {2,4,5} is not frequent, so {1,2,3,5} is pruned.

Thus, {1,2,3,4} survives.

Question 4 [25 pts total]

Consider the following table for questions 4a) to 4c):

Transaction ID	Items
1	**{Beer, Diapers}**
2	**{Milk, Diapers, Bread, Butter}**
3	**{Milk, Diapers, Cookies}**
4	**{Bread, Butter, Cookies}**
5	**{Milk, Beer, Diapers, Eggs}**
6	**{Beer, Cookies, Diapers}**
7	**{Milk, Diapers, Bread, Butter}**
8	**{Bread, Butter, Diapers}**
9	**{Bread, Butter, Milk}**
10	**{Beer, Butter, Cookies}**

[3 pts] Question 4a.) What is the maximum number of association rules that can be extracted from this data (including rules that have zero support)?

From the above table, we can see that there are 7 items, which are {Beer, Diapers, Milk, Bread, Butter, Cookies, Eggs} .

So, by using the association rules $R = 3^n - 2^{n+1} + 1$, we can calculate that the maximum number of association rules that can be extracted from the data is equal to $3^7 - 2^{7+1} + 1$ = 1932.

[3 pts] Question 4b.) What is the confidence of the rule {Milk, Diapers} ⇒ {Butter}?

$$Confidence = \frac{\sigma(\{Milk,\ Diapers,\ Butter\})}{\sigma(\{Milk,\ Diapers\})}$$

From the given table, we can see that Transactions 2, 3, 5, 7 contain {Milk, Diapers}. Transactions 2, 7 contain {Milk, Diapers, Butter}.

Thus, Confidence = 2/4 = 0.5

[3 pts] Question 4c.) What is the support for the rule {Milk, Diapers} ⇒ {Butter}?

$$Support = \frac{\sigma(\{Milk,\ Diapers,\ Butter\})}{\left|T\right|}$$

The number of {Milk, Diapers, Butter} is 2, and it has 10 transactions.

Thus, Support = 2/10 = 0.2

[3 pts] Question 4d.) True or False with an explanation: Given that $\{a,\ b,\ c,\ d\}$ is a frequent itemset, $\{a,\ b\}$is always a frequent itemset.

True.

Because {a, b} is a subset of {a, b, c, d}. If {a, b, c, d} is a frequent itemset, its subsets are also a frequent itemset. Thus, it’s true.

[3 pts] Question 4e.) True or False with an explanation: Given that $\{a, b\}$ $\{b, c\}$ and $\{a, c\}$ are frequent itemsets, $\{a, b, c\}$ is always frequent.

False.

Just because {a, b}, {b, c}, and {a, c} are frequent, it's not necessarily assuming that {a, b, c} is always frequent. This is because {a, b, c} might not show in the same transactions as any one of the subsets.

[3 pts] Question 4f.) True or False with an explanation: Given that the support of $\{a, b\}$ is 20 and the support of $\{b, c\}$ is 30, the support of $\{b\}$ is larger than 20 but smaller than 30.

False.

Because if the support of {a, b} is 20 and the support of {b, c} is 30, the support of {b} would be larger or equal than 30. Thus, the support of {b} is larger than 20 but smaller than 30 is false.

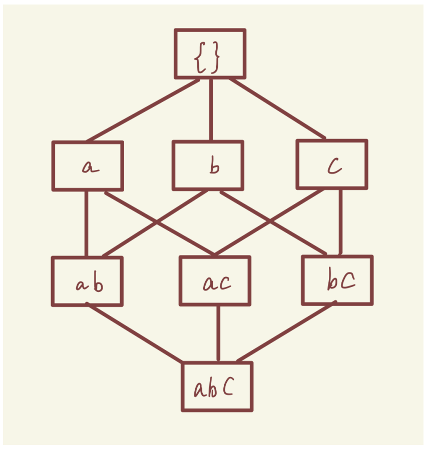
[3 pts] Question 4g.) True or False with an explanation: In a dataset that has 5 items, the maximum number of size-2 frequent itemsets that can be extracted (assuming minsup > 0) is 20.

False.

The maximum number of size-2 frequent itemsets that can be extracted is 10.

Because C(5, 2) = 10.

[4 pts] Question 4h.) Draw the itemset lattice for the set of unique items I = {a, b, c}.


{}
a
b
c
ab
ac
bc
abc