CS 584-04: Machine Learning

Spring 2020 Assignment 2

Question 1 (35 points)

The file Groceries.csv contains market basket data. The variables are:

1. Customer: Customer Identifier
2. Item: Name of Product Purchased

After you have imported the CSV file, please discover association rules using this dataset. For your information, the observations have been sorted in ascending order by Customer and then by Item. Also, duplicated items for each customer have been removed.

a) (5 points) Create a data frame that contains the number of unique items in each customer's market basket. Draw a histogram of the number of unique items. What are the 25th, 50th, and the 75th percentiles of the histogram?

Ans:

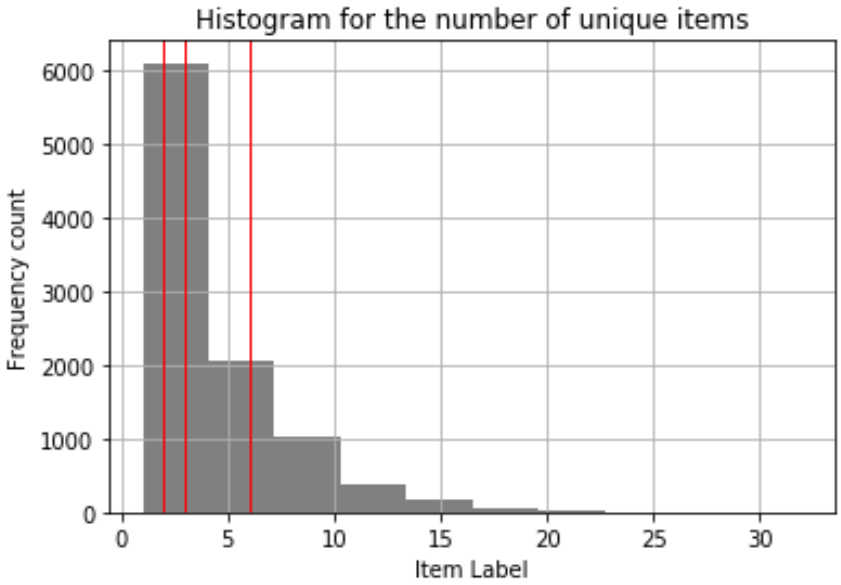
Input:

```
Q1,Q2,Q3= np.percentile(new_dataset, [25 ,50 ,75])
print("the 25th percentiles of the histogram is:",Q1)
print("Median of the histogram is",Q2)
print("the 75th percentiles of the histogram is:",Q3)

#histogram
plt.hist(new_dataset,color="gray")
plt.title('Histogram for the number of unique items')
plt.xlabel('Item Label')
plt.ylabel('Frequency count')
plt.axvline(Q1, color='red', linewidth=1, alpha=1)
plt.axvline(Q2, color='red', linewidth=1, alpha=1)
plt.axvline(Q3, color='red', linewidth=1, alpha=1)
plt.grid(True)
plt.show()
```

Output:

```
Distinct items :
 Customer
4918     1
3937     1
3932     1
1819     1
1820     1
        ..
5611    28
9002    29
2974    29
2939    29
1217    32
Name: Item, Length: 9835, dtype: int64
the 25th percentiles of the histogram is: 2.0
Median of the histogram is 3.0
the 75th percentiles of the histogram is: 6.0
```



In the above histogram, first red line shows the 25th percentiles of the histogram which is at 2.0, Second red line shows the 50th percentiles(i.e., median) of the histogram which is at 3.0 and the third red line shows the 75th percentiles of the histogram which is at 6.0.

b) (10 points) We are only interested in the *k*-itemsets that can be found in the market baskets of at least seventy five (75) customers. How many itemsets can we find? Also, what is the largest *k* value among our itemsets?
Ans:

Input:

```
listItem = dataset.groupby(['Customer'])['Item'].apply(list).values.tolist()
# Convert the Item List format to the Item Indicator format
te = TransactionEncoder()
te_ary = te.fit(listItem).transform(listItem)
itemIndicator = pd.DataFrame(te_ary, columns=te.columns_)
# Find the frequent itemsets
min_supp = 75 / len(new_dataset)
print("\nMinimum support is:",min_supp)
frequent_itemset = apriori(itemIndicator, min_support=min_supp, use_colnames=True)
k_largest_itemset = len(frequent_itemset['itemsets'][len(frequent_itemset) - 1])
print("frequent item sets : \n",frequent_itemset['itemsets'])
print("\ntotal number of item sets found = ",frequent_itemset.shape[0])
print("\nthe largest value of k =",k_largest_itemset)
```

Output:

```
Minimum support is: 0.007625826131164209
frequent item sets :
 0                                (Instant food products)
1                                           (UHT-milk)
2                                      (baking powder)
3                                                (beef)
4                                             (berries)
                              ...
519     (whole milk, whipped/sour cream, tropical fruit)
520                 (whole milk, yogurt, tropical fruit)
521             (whole milk, whipped/sour cream, yogurt)
522    (yogurt, whole milk, root vegetables, other ve...
523    (yogurt, whole milk, tropical fruit, other veg...
Name: itemsets, Length: 524, dtype: object

total number of item sets found =  524

the largest value of k = 4
```

c) (10 points) Find out the association rules whose Confidence metrics are greater than or equal to 1%. How many association rules can we find? Please be reminded that a rule must have a non-empty antecedent and a non-empty consequent. Please **do not** display those rules in your answer.

Ans:

Input:

```
assoc_rules = association_rules(frequent_itemset, metric="confidence", min_threshold=0.01)
print("\nTotal number of association rules found = ",assoc_rules.shape[0])
```

Output:

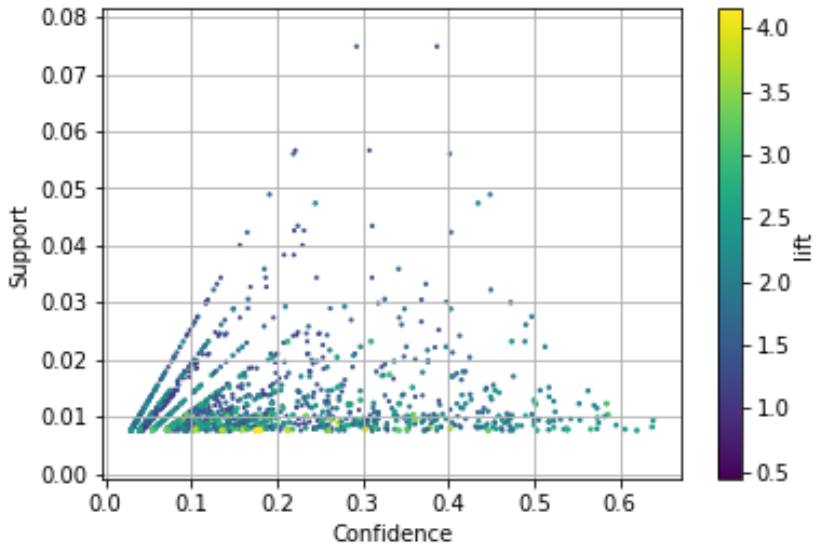
```
Total number of association rules found =  1228
```

d) (5 points) Plot the Support metrics on the vertical axis against the Confidence metrics on the horizontal axis for the rules you have found in (c). Please use the Lift metrics to indicate the size of the marker.

Input:

```
plt.scatter(assoc_rules['confidence'], assoc_rules['support'], c=assoc_rules['lift'], s=assoc_rules['lift'])
plt.ylabel("Support")
plt.xlabel("Confidence")
cbar = plt.colorbar()
cbar.set_label('lift', labelpad=+1)
plt.grid(True)
plt.show()
```

Output:



e) (5 points) List the rules whose Confidence metrics are greater than or equal to 60%. Please include their Support and Lift metrics.

Input:

```
assoc_rules = association_rules(frequent_itemset, metric="confidence", min_threshold=0.6)
print("association rules are: \n",assoc_rules.to_string())
```

Output:

```
association rules are:
                              antecedents   consequents  antecedent support  consequent support   support  confidence      lift  leverage  conviction
0                 (butter, root vegetables)  (whole milk)            0.012913            0.255516  0.008236    0.637795  2.496107  0.004936    2.055423
1                         (yogurt, butter)  (whole milk)            0.014642            0.255516  0.009354    0.638889  2.500387  0.005613    2.061648
2  (yogurt, root vegetables, other vegetables)  (whole milk)            0.012913            0.255516  0.007829    0.606299  2.372842  0.004530    1.890989
3   (yogurt, tropical fruit, other vegetables)  (whole milk)            0.012303            0.255516  0.007626    0.619835  2.425816  0.004482    1.958317
```

Question 2 (30 points)

The K-means algorithm works only with interval features. One way to apply the k-means algorithm to categorical features is to transform them into a new interval feature space. However, this approach can be very inefficient, and it does not produce good results.

For clustering categorical features, we should consider the K-modes clustering algorithm which extends the K-means algorithm by using different dissimilarity measures and a different method for computing cluster centers. See this article for more details. Huang, Z. (1997). "A Fast Clustering Algorithm to Cluster Very Large Categorical Data Sets in Data Mining." In Proceedings of the *SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery*, 1–8. New York: ACM Press.

Please implement the K-modes clustering method in Python and then apply the method to the cars.csv. Your input fields are these four categorical features: Type, Origin, DriveTrain, and Cylinders. **Please do not remove the missing or blank values in these four features**. Instead, consider these values as a separate category.

The cluster centroids are the modes of the input fields. In the case of tied modes, choose the lexically or numerically lowest one.

Suppose a categorical feature has observed values $v_1, \ldots, v_p$. Their frequencies (i.e., number of observations) are $f_1, \ldots, f_p$. The distance metric between two values is $d(v_i, v_j) = 0$ if $v_i = v_j$. Otherwise, $d(v_i, v_j) = \frac{1}{f_i} + \frac{1}{f_j}$. The distance between any two observations is the sum of the distance metric of the four categorical features.

a) (5 points) What are the frequencies of the categorical feature Type?

Ans:

Input:

```
print('\nThe frequencies of the categorical feature Type is ')
print(dataset['Type'].value_counts())
```

Output:

```
The frequencies of the categorical feature Type is
Sedan     262
SUV        60
Sports     49
Wagon      30
Truck      24
Hybrid      3
```

b) (5 points) What are the frequencies of the categorical feature DriveTrain?

Ans:

Input:

```
print('\nThe frequencies of the categorical feature DriveTrain is ')
print(dataset['DriveTrain'].value_counts())
```

Output:

```
The frequencies of the categorical feature DriveTrain is
FWD    226
RWD    110
AWD     92
```

c) (5 points) What is the distance between Origin = 'Asia' and Origin = 'Europe'?

Ans:

Input:

```
val = dataset['Origin'].value_counts(dropna = False).to_dict()
dist = 1/(val['Asia'])+1/(val['Europe'])
print('\ndistance between Origin = 'Asia' and Origin = 'Europe'is ',dist)
```

Output:

```
distance between Origin = 'Asia' and Origin = 'Europe'is  0.014459195224863643
```

d) (5 points) What is the distance between Cylinders = 5 and Cylinders = Missing?

Ans:

Input:

```
val1 = dataset['Cylinders'].fillna(0.0).value_counts(dropna = False).to_dict()
print(val1)
dist1 = 1/(val1[5.0])+1/(val1[0.0])
print('\nThe the distance between Cylinders = 5 and Cylinders = Missing is ',dist1)
```

Output:

```
The the distance between Cylinders = 5 and Cylinders = Missing is  0.6428571428571428
```

e) (5 points) Apply the K-modes method with **three clusters**. How many observations in each of these three clusters? What are the centroids of these three clusters?

Ans:

k-means algorithm cannot be used efficiently when the data is categorical. So, we can use K-modes algorithm when data is categorical. In K-modes algorithm, instead of considering mean of each cluster, we consider mode of element.

Input:

```
km = KModes(n_clusters=3,init='Huang')
clusters = km.fit_predict(cardata)
print('\nThe number of observations in each clusters are: ')
print("\t0:",list(clusters).count(0))
print("\t1:",list(clusters).count(1))
print("\t2:",list(clusters).count(2))
print('The centroids for the 3 clusters are:')
print(km.cluster_centroids_)
```

Output:

```
The number of observations in each clusters are:
        0: 253
        1: 109
        2: 66
The centroids for the 3 clusters are:
[['Sedan' 'USA' 'FWD' '6.0']
 ['Sedan' 'Asia' 'FWD' '4.0']
 ['Sports' 'Europe' 'RWD' '8.0']]
```

f) (5 points) Display the frequency distribution table of the Origin feature in each cluster.

Ans:

Input:

```
cardata['Cluster_number']=pd.Series(clusters)
print("\n",pd.crosstab(index = cardata["Origin"], columns = cardata["Cluster_number"]))
```

Output:

```
 Cluster_number    0   1   2
Origin
Asia              66  86   6
Europe            49  23  51
USA              138   0   9
```

Question 3 (35 points)

Apply the Spectral Clustering method to the FourCircle.csv. Your input fields are x and y. Wherever needed, specify random_state = 60616 in calling the KMeans function.

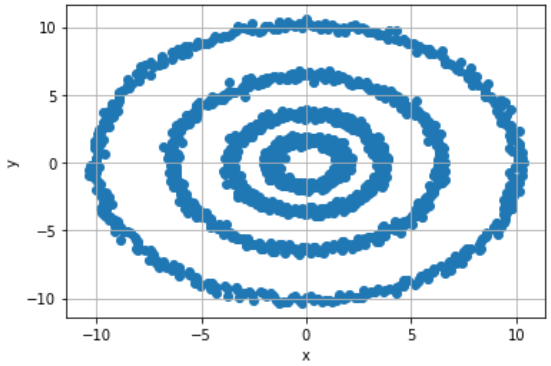
g) (5 points) Plot y on the vertical axis versus x on the horizontal axis. How many clusters are there based on your visual inspection?

Ans:

Input:

```
plt.scatter(dataset['x'], dataset['y'])
plt.xlabel('x')
plt.ylabel('y')
plt.grid(True)
plt.show()
```

Output:



Based on visual Inspection, we can say that there are **Four clusters**.

h) (5 points) Apply the K-mean algorithm directly using your number of clusters that you think in (a). Regenerate the scatterplot using the K-mean cluster identifiers to control the color scheme. Please comment on this K-mean result.

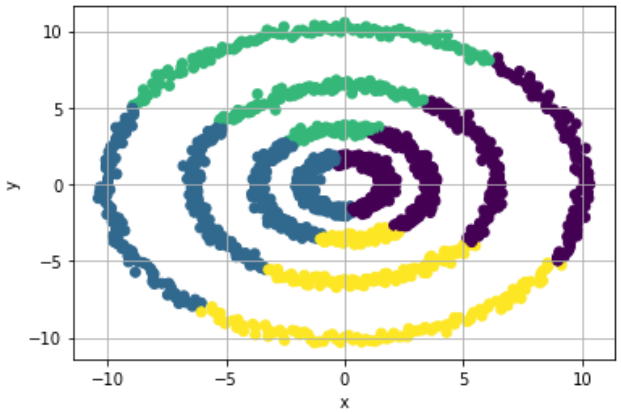
Ans:

Input:

```
k_means_model = cluster.KMeans(n_clusters=num_clusters, random_state=60(
k_means=k_means_model.fit(trainData)
print("Centroids of the Cluster are= \n", k_means.cluster_centers_)

# printing cluster data
dataset['k_mean_cluster'] = k_means.labels_
for i in range(num_clusters):
    print("\ncluster label = ", i)
    print(dataset.loc[dataset['k_mean_cluster'] == i])
#plot
plt.scatter(dataset['x'], dataset['y'], c=dataset['k_mean_cluster'])
plt.xlabel('x')
plt.ylabel('y')
plt.grid(True)
plt.show()
```

Output:



In the above scatter plot, we can see that the number of clusters chosen are four. But the clusters are not formed properly by using K-means algorithm as we can observe that although the data is almost equally distributed to each of the four clusters but we can observe that there are many data points which are closer to each other but still belong to different clusters.

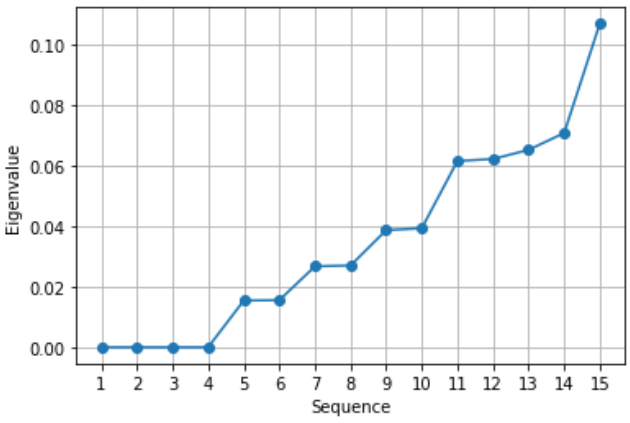
i) (10 points) Apply the nearest neighbor algorithm using the Euclidean distance. We will consider the number of neighbors from 1 to 15. What is the smallest number of neighbors that we should use to discover the clusters correctly? Remember that we may need to try a couple of values first and use the eigenvalue plot to validate our choice.

Ans:

Input:

```
sequence = np.arange(1,16,1)
plt.plot(sequence, evals[0:15,], marker = "o")
plt.xlabel('Sequence')
plt.ylabel('Eigenvalue')
plt.grid("both")
plt.xticks(sequence)
plt.show()
```

Output:



We have applied nearest neighbor algorithm using the Euclidean distance by considering neighbors from 1 to 15 and we have observed that the smallest number of neighbors that we should use are **6 neighbors** as by choosing six neighbors we were able to discover the clusters correctly.

j) (5 points) Using your choice of the number of neighbors in (c), calculate the Adjacency matrix, the Degree matrix, and finally the Laplacian matrix. How many eigenvalues do you determine are practically zero? Please display their calculated values in scientific notation.

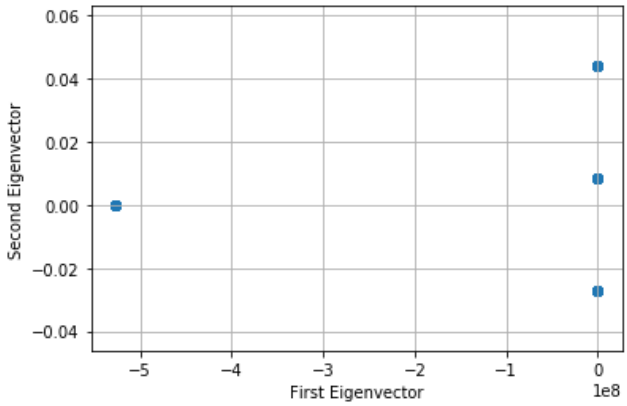
Ans: Total of **Four** eigenvalues we have determine are practically zero.

Input:

```
for i in range(4):
    print("{:e}".format(evals[i]))
```

Output:

```
-2.730702e-15
-8.206377e-17
8.307187e-15
1.477970e-14
```



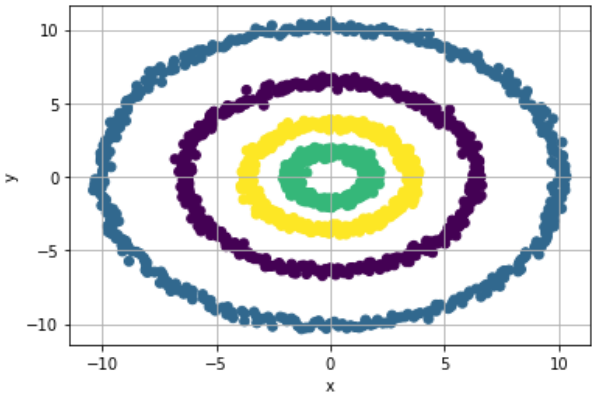
k) (10 points) Apply the K-mean algorithm on the eigenvectors that correspond to your "practically" zero eigenvalues. The number of clusters is the number of your "practically" zero eigenvalues. Regenerate the scatterplot using the K-mean cluster identifier to control the color scheme.

Ans:

Input:

```
Z = evecs[:,[0,3]]
kmeans_spectral = cluster.KMeans(n_clusters=4, random_state=60616).fit(Z)
dataset['ring'] = kmeans_spectral.labels_
plt.scatter(dataset['x'], dataset['y'], c=dataset['ring'])
plt.xlabel('x')
plt.ylabel('y')
plt.grid(True)
plt.show()
```

Output:


10
5
0
−5
−10
y
−10
−5
0
5
10
x