CO544: Machine Learning and Data Mining

Lab 04: Clustering and Association Rule Learning

E/20/197 Kawya A.H.D.

Exercise 01

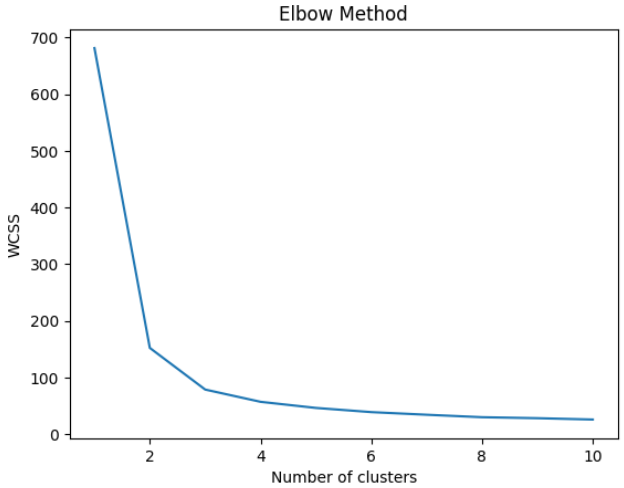
Clutering – Iris Dataset

1. Import the iris dataset from scikit-learn. Convert it into an unlabeled data set by removing the class attribute.

```
from sklearn.cluster import KMeans
from sklearn.datasets import load_iris
```

```
# Work only with the iris.data
X = iris.data
```

2. Use the Elbow method to identify the best value for k (minimizing WCSS).



As the result got, the rapid change stops at k=3. That means the best value for k is 3.

3. Fit the K-Means algorithm with the k found in part(b).

```
kmeans = KMeans(n_clusters=3, random_state=0) # from Elbow method
# random_state=0 This gives the same result in every time we run this
closest_cluster_index = kmeans.fit_predict(X)
cluster_centers = kmeans.cluster_centers_
```

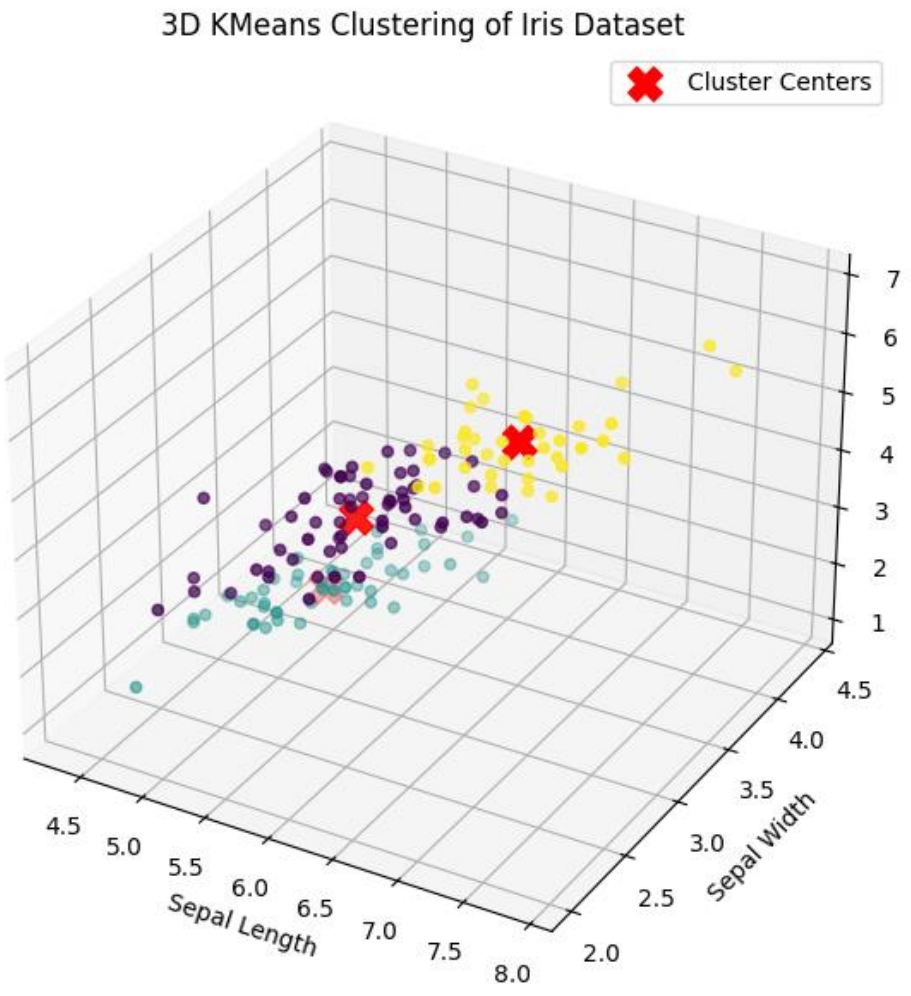
4. Explain the output of:

   kmeans.cluster_centers_

   where k means is your fitted KMeans object.

It gives the final centers of the clusters.

5. Visualize the data points and cluster centers in a 3D plot using the first three features as axes.



Exercise 02

Association Rule Learning

1. Import the provided groceries.csv dataset.

```
data = pd.read_csv('groceries.csv', header=None)
```

As above the data set has been loaded. 'header=None' means no column names in csv file given. Each of the rows represents one transaction.

2. Explore the dataset and build the frequent-item Data Frame.

```
transactions = data.apply(lambda row: row.dropna().tolist(), axis=1).tolist()
te = TransactionEncoder()
te_data = te.fit(transactions).transform(transactions)
df = pd.DataFrame(te_data, columns=te.columns_)
df.head()
```

Here the first line drops the NaNs in the transactions, otherwise, it will produce an error because mix of float (NaN) and strings break sorting.

Then the second line turns the items lists into a one-hot table. '.fit()' learns all unique items and '.transform()' creates a binary matrix to present the items in each of the transaction.

	Instant food products	UHT-milk	abrasive cleaner	artif. sweetener	baby cosmetics	baby food	bags	baking powder
0	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False

3. Apply the Apriori algorithm to find item sets with support > 8%.

```
import warnings
warnings.filterwarnings("ignore", category=RuntimeWarning)
freq = apriori(df, min_support=0.08, use_colnames=True)
freq.head()
```

	support	itemsets
0	0.080529	(bottled beer)
1	0.110524	(bottled water)
2	0.082766	(citrus fruit)
3	0.193493	(other vegetables)
4	0.088968	(pastry)

Here it finds the frequent item combinations that appear in at least 8% of all transactions. In simple terms, we only care about item sets seen in at least 8% of the transactions

4. Generate association rules using the lift metric.

```
rules = association_rules(freq, metric="lift", min_threshold=1)
rules.head()
```

antecedents	consequents	antecedent support	consequent support	support	confidence	lift

With 0.08 support, there is no rule generated. Here lift matric means,

$$If\ \{LHS\ Items\}\ then\ \{RHS\ Items\}$$

How much more likely RHS is given LHS compared to random chance.

5. Select one rule and interpret it in your own words.

As above mentioned, with the support given, there is not any rule generated. But I have tried to with support 0.70. Then it has been given the following rules.

```
import warnings
warnings.filterwarnings("ignore", category=RuntimeWarning)
freq = apriori(df, min_support=0.070, use_colnames=True)
rules = association_rules(freq, metric="lift", min_threshold=1)
rules.head()
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	representativity	leverage	conviction	zhangs_metric	jaccard	certainty	kulczynski
0	(other vegetables)	(whole milk)	0.193493	0.255516	0.074835	0.386758	1.513634	1.0	0.025394	1.214013	0.420750	0.2	0.176286	0.339817
1	(whole milk)	(other vegetables)	0.255516	0.193493	0.074835	0.292877	1.513634	1.0	0.025394	1.140548	0.455803	0.2	0.123228	0.339817

Interpret a rule

```
rules.iloc[0]
```

	0
antecedents	(other vegetables)
consequents	(whole milk)
antecedent support	0.193493
consequent support	0.255516
support	0.074835
confidence	0.386758
lift	1.513634
representativity	1.0
leverage	0.025394
conviction	1.214013
zhangs_metric	0.42075
jaccard	0.2
certainty	0.176286
kulczynski	0.339817

According to this output,

The customers buy other vegetables, mostly buy whole milk as well.

If I explain this furthermore, out of all the customers, 19% buy other vegetables. When they do, 39% of the time, they also buy whole milk as well. Because the confidence level is 39%. And this is 1.5 times more likely than a random customer buying whole milk. Hence, there a considerable association between buying other vegetables and buying whole milk.

6. How many rules satisfy both lift > 4 and confidence > 0.8?

```
strong_rules = rules[(rules['lift'] > 4) & (rules['confidence'] > 0.8)]
print(f"Number of strong rules: {len(strong_rules)}")
strong_rules
```

```
Number of strong rules: 0
```

antecedents	consequents	antecedent support	consequent support	support	confidence	lift

With the given conditions, there is no any rule generated.