

University of Central Missouri
Department of Computer Science & Cybersecurity

CS5710 Machine Learning

Fall 2025

Home Assignment 4.

Student name:

Submission Requirements:

- Once finished your assignment push your source code to your repo (GitHub) and explain the work through the ReadMe file properly. Make sure you add your student info in the ReadMe file.
- Comment your code appropriately ***IMPORTANT***.
- Any submission after provided deadline is considered as a late submission.

Part A: Calculation

Q1. Find the cluster using the Average and MIN technique. Use Euclidean distance to build the complete distance matrix, updated the distance matrix to the final step and draw the dendrogram for each.

	X	Y
P1	0.4	0.5
P2	0.2	0.3
P3	0.1	0.08
P4	0.21	0.12
P5	0.6	0.16
P6	0.33	0.28
P7	0.11	0.15

Solution:

Complete pairwise Euclidean distance matrix (rounded to 6 decimals):

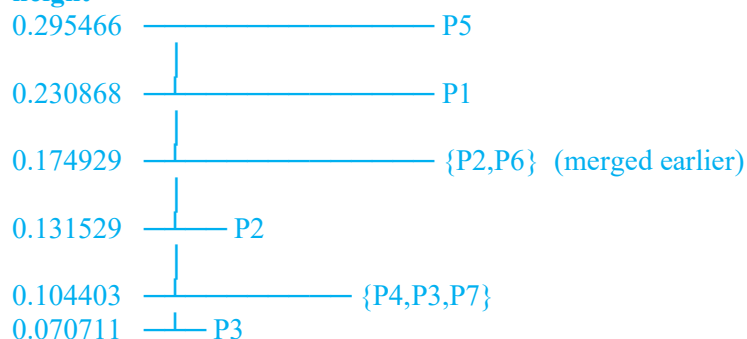
	P1	P2	P3	P4	P5	P6	P7
P1	0.000000	0.282843	0.516140	0.424853	0.394462	0.230868	0.454533
P2	0.282843	0.000000	0.241661	0.180278	0.423792	0.131529	0.174929
P3	0.516140	0.241661	0.000000	0.117047	0.506360	0.304795	0.070711
P4	0.424853	0.180278	0.117047	0.000000	0.392046	0.200000	0.104403
P5	0.394462	0.423792	0.506360	0.392046	0.000000	0.295466	0.490102
P6	0.230868	0.131529	0.304795	0.200000	0.295466	0.000000	0.255539
P7	0.454533	0.174929	0.070711	0.104403	0.490102	0.255539	0.000000

Single-link (MIN) clustering:

Merge sequence (cluster A merged with cluster B at height = distance):

1. Merge P3 and P7 at distance **0.070711**
(smallest pairwise: $d(P3, P7) = \sqrt{((0.10 - 0.11)^2 + (0.08 - 0.15)^2)} = 0.0707107...$)
2. Merge P4 with cluster {P3, P7} at distance **0.104403**
(min distance between P4 and {P3, P7} is $d(P4, P7) = 0.104403$)
3. Merge P2 and P6 at distance **0.131529**
4. Merge cluster {P4, P3, P7} with cluster {P2, P6} at distance **0.174929**
(single-link uses the minimum pairwise distance between any member of the two clusters; min here is $d(P2, P7) = 0.174929$)
5. Merge P1 with the large cluster {P4, P3, P7, P2, P6} at distance **0.230868**
(min $d(P1, P6) = 0.2308679...$)
6. Merge P5 with the rest at distance **0.295466**

height

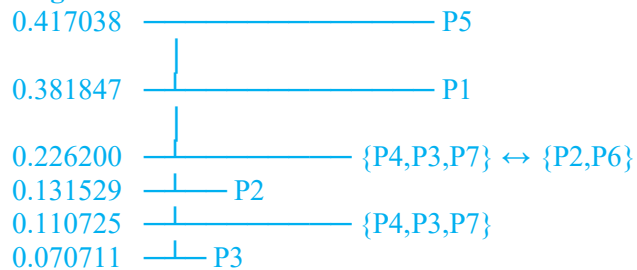


Average-link clustering:

Merge sequence with average distances:

1. Merge P3 and P7 at **0.070711** (same first merge)
2. Merge P4 with {P3,P7} at **0.110725**
(average of $d(P4,P3)=0.117047$ and $d(P4,P7)=0.104403 \rightarrow (0.117047+0.104403)/2 = 0.110725$)
3. Merge P2 and P6 at **0.131529**
4. Merge {P4,P3,P7} with {P2,P6} at **0.226200**
(average across all $3 \times 2 = 6$ pairwise distances — computed exactly; result ≈ 0.2262)
5. Merge P1 with the large cluster {P4,P3,P7,P2,P6} at **0.381847**
(average distance of P1 to all five points in that cluster)
6. Merge P5 with the rest at **0.417038**

height



Q2. We have the following 2D data points:

Points: (2,1), (3,1), (3, 3), (4, 1), (5, 1), (6,7), (1,3), (2,5)

for $K=3$:

Centroid1: (2,1)

Centroid 2: (4, 1)

Centroid 3: (5, 1)

Euclidean Distance:

$$\text{Distance} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Solution:

Distances from each point to each centroid (Euclidean)

- (2,1) : [0.0, 2.0, 3.0]
- (3,1) : [1.0, 1.0, 2.0] (tie between C1 and C2; by convention assign to smallest-index centroid $\rightarrow C1$)
- (3,3) : [2.236068, 2.236068, 2.828427]
- (4,1) : [2.0, 0.0, 1.0]
- (5,1) : [3.0, 1.0, 0.0]
- (6,7) : [7.211103, 6.324555, 6.082763]
- (1,3) : [2.236068, 3.605551, 4.472136]
- (2,5) : [4.0, 4.472136, 5.0]

Assignments (nearest centroid; ties \rightarrow choose lowest index)

- Cluster 1 (C1) receives: (2,1), (3,1), (3,3), (1,3), (2,5)

- Cluster 2 (C2) receives: (4,1)
- Cluster 3 (C3) receives: (5,1), (6,7)

New centroids (mean of assigned points)

- New Centroid 1 = average of $\{(2,1),(3,1),(3,3),(1,3),(2,5)\} = (2.2, 2.6)$
- New Centroid 2 = (4,1) (single point) = **(4.0, 1.0)**
- New Centroid 3 = average of $\{(5,1),(6,7)\} = (5.5, 4.0)$

Second iteration: distances to the **new** centroids and reassign

- (2,1) : [1.612451, 2.0, 4.609772]
- (3,1) : [1.788854, 1.0, 3.905124]
- (3,3) : [0.894427, 2.236068, 2.69258]
- (4,1) : [2.40831, 0.0, 3.354101]
- (5,1) : [3.224903, 1.0, 3.041381]
- (6,7) : [5.813776, 6.324555, 3.041381]
- (1,3) : [1.264911, 3.605551, 4.609772]
- (2,5) : [2.408318, 4.472136, 3.6400549]

New assignments after iteration 2

Cluster 1 (C1) now has: (2,1), (3,3), (1,3), (2,5)

Cluster 2 (C2) now has: (3,1), (4,1), (5,1)

Cluster 3 (C3) now has: (6,7)

Compute centroids after iteration 2

Cluster 1 with points (2,1), (3,3), (1,3), (2,5): C1=(2.0,3.0)

Cluster 2 with points (3,1), (4,1), (5,1): C2=(4.0,1.0)

Cluster 3 with single point (6,7): C3=(6.0,7.0)

Third iteration: distances to the **new** centroids and reassign

- (2,1) : [2.0, 2.0, larger]
- (3,1) : [larger, 1, larger]
- (3,3) : [1, larger, larger]
- (4,1) : [larger, 0.0, larger]
- (5,1) : [larger, 1.0, larger]
- (6,7) : [larger, larger, 0]
- (1,3) : [1, larger, larger]
- (2,5) : [2, larger, larger]

New assignments after iteration 3

Cluster	Points	Final Centroid
C ₁	(2,1), (3,3), (1,3), (2,5)	(2.0, 3.0)
C ₂	(3,1), (4,1), (5,1)	(4.0, 1.0)
C ₃	(6,7)	(6.0, 7.0)

Part B — Short-Answer

Q1.

a) Describe agglomerative hierarchical clustering.

Answer: *Agglomerative hierarchical clustering* starts with each object as its own cluster and repeatedly merges the two closest clusters until all objects are in one cluster (or a stopping criterion). It produces a dendrogram (bottom-up).

b) Describe divisive hierarchical clustering.

Answer: *Divisive hierarchical clustering* starts with all objects in one cluster and recursively splits clusters into smaller ones (top-down) until each object is alone (or stopping criterion).

c) Which one is more commonly used and why?

Answer: Agglomerative is more commonly used because it is simpler to implement, widely available in libraries, and computationally often easier for typical dataset sizes (divisive methods can require evaluating many possible splits). Agglomerative methods also produce intuitive dendrograms and flexible linkage choices.

Q2.

a) To improve clustering quality, should inter-cluster distance be maximized or minimized?

Answer: To improve clustering quality, **inter-cluster distance should be maximized** — we want clusters to be far apart (well separated).

b) Same question for intra-cluster distance — explain the reasoning.

Answer: **Intra-cluster distance should be minimized** — members of a cluster should be close/similar. Maximizing inter-cluster distance and minimizing intra-cluster distance jointly leads to compact and well-separated clusters.

Q3.

a) Define single link, complete link, and average link.

Answer:

- **Single link (minimum link):** cluster distance = minimum distance between any pair of points (one from each cluster).
- **Complete link (maximum link):** cluster distance = maximum distance between any pair of points (one from each cluster).
- **Average link:** cluster distance = average of all pairwise distances between points in the two clusters.

b) Explain one strength and one weakness of single-link clustering.

Answer:

- **Strength:** Good at finding non-spherical clusters and chaining structure; can connect cluster members by nearest neighbors.
- **Weakness:** Very sensitive to noise and outliers; prone to the “chain” effect where clusters can be linked through single intermediate points (producing elongated clusters).

Q4.

a) What is the role of tokenization and give one example.

Answer: *Role of tokenization:* Tokenization splits raw text into meaningful units (tokens) such as words or subwords; it's the first preprocessing step in most NLP pipelines. Example: the sentence "John's book." tokenized to ["John", "'s", "book", "."] or (word tokens) ["John", " 's", "book", "."] depending on tokenizer.

b) Compare stemming vs. lemmatization in terms of speed and accuracy.

Answer:

- **Speed:** Stemming is usually faster (rule-based chopping of word endings).

- **Accuracy:** Lemmatization is more accurate (returns valid dictionary forms/lemmas using POS and morphological analysis), but slower because it often requires POS tagging or morphological lookup.

Q5.

a) Explain what word sense ambiguity is and provide an example.

Answer: *Word-sense ambiguity* (polysemy): a word has multiple meanings. Example: bank — "river bank" vs. "financial bank". The correct sense depends on context.

b) Explain why pronoun reference ambiguity can confuse a model.

Answer: *Pronoun reference ambiguity:* Pronouns (he/she/they/it) can have unclear antecedents. Example: "Chris met Alex at Apple headquarters in California. He told him about the new iPhone." — who does "He" refer to? A model can be confused because resolving pronouns requires world/ discourse context, coreference resolution, and sometimes real-world knowledge.

Q6.

a) Why can't NLP tasks like POS tagging be solved by predicting each token independently?

Answer: *Why not predict each token independently for POS tagging?* Because POS tags are context dependent and adjacent tags are correlated. Tagging decisions influence neighbors (e.g., a word could be noun or verb depending on neighboring words). Modeling tokens independently ignores these dependencies and yields worse accuracy.

b) Give one example where decisions are mutually dependent in a sentence.

Answer: *Example of mutually dependent decisions:* In the phrase "Visiting relatives can be annoying", deciding whether "Visiting" is a verb (gerund) or adjective/noun depends on the parse and on "relatives" — the POS and parse decisions are interdependent. Similarly, subject-verb agreement decisions depend on the subject and verb tokens jointly.

Part C — Coding

Q1.

Write Python code to perform the following steps:

1. Segment into tokens
2. Remove stopwords
3. Apply lemmatization (not stemming)
4. Keep only verbs and nouns (use POS tags)

Input text:

"John enjoys playing football while Mary loves reading books in the library."

Code link:

<https://colab.research.google.com/drive/1b1IGHFeeerghzwHuj8MA7O9zN388jKAp#scrollTo=aX0oddXdfaXt&line=37&uniqifier=1>

Github Link:

https://github.com/saisravan98/MachineLearning_HomeAssignment4.git

Code:

```
# q1_nlp.py
```

```

"""
Requires:
    pip install spacy
    python -m spacy download en_core_web_sm
"""

import spacy

nlp = spacy.load("en_core_web_sm", disable=["parser", "ner"]) # we only
need tokenizer, tagger, lemmatizer
nlp.enable_pipe("tagger")

text = "John enjoys playing football while Mary loves reading books in the
library."

doc = nlp(text)

# 1. Tokenize (spaCy doc)
tokens = [t for t in doc]

# 2. Remove stopwords and punctuation, and lemmatize
filtered = []
for t in tokens:
    if t.is_stop or t.is_punct or t.like_num:
        continue
    # keep only alphabetic tokens (drop punctuation, etc.)
    if not t.is_alpha:
        continue
    # 3. Keep only verbs and nouns (POS tags)
    # spaCy POS tags: t.pos_ is coarse-grained (e.g., 'VERB', 'NOUN',
    'PROPN', ...)
    if t.pos_ in ("VERB", "NOUN", "PROPN"):
        filtered.append((t.text, t.lemma_, t.pos_))

# Output
print("Tokens (text, lemma, POS):")
for item in filtered:
    print(item)

```

Output:

```

Tokens (text, lemma, POS):
('John', 'John', 'PROPN')
('enjoys', 'enjoy', 'VERB')
('playing', 'play', 'VERB')
('football', 'football', 'NOUN')
('Mary', 'Mary', 'PROPN')

```

```
('loves', 'love', 'VERB')
('reading', 'read', 'VERB')
('books', 'book', 'NOUN')
('library', 'library', 'NOUN')
```

Q2.

Use Python and any NLP model to perform:

1. Named Entity Recognition (NER)
2. Disambiguation prompt:

If the text contains a pronoun ("he", "she", "they"), print:

"Warning: Possible pronoun ambiguity detected!"

Input text:

"Chris met Alex at Apple headquarters in California. He told him about the new iPhone launch."

Code link:

https://colab.research.google.com/drive/1b1IGHFeeerghzwHuj8MA7O9zN388jKAp#scrollTo=mAz_wh7BgIP9&line=26&uniqifier=1

Github Link:

https://github.com/saisravan98/MachineLearning_HomeAssignment4.git

Code:

```
# q2_ner_pronoun_warning.py
"""
Requires:
    pip install spacy
    python -m spacy download en_core_web_sm
"""

import spacy

nlp = spacy.load("en_core_web_sm") # includes NER

text = "Chris met Alex at Apple headquarters in California. He told him about the new iPhone launch."

doc = nlp(text)

# Named Entities
print("Named Entities (text, label):")
for ent in doc.ents:
    print(f"{ent.text} - {ent.label}")
```

```
# Pronoun ambiguity check (simple heuristic)
pronouns =
{"he", "she", "they", "him", "her", "them", "his", "hers", "their", "it", "its"}
tokens_lower = [t.text.lower() for t in doc]
if any(p in tokens_lower for p in pronouns):
    print("\nWarning: Possible pronoun ambiguity detected!")
```

Output:

```
Named Entities (text, label):
Chris - PERSON
Alex - PERSON
Apple - ORG
California - GPE
iPhone - ORG
```

```
Warning: Possible pronoun ambiguity detected!
```