Exercise Set 3

- Submit your answer to Moodle by **Wednesday, 03 December 2025**, no later than 23:59.
- You may answer anonymously or include your name, as you prefer.
- Each assignment must be written independently. Collaborative discussion is encouraged, and using external resources—including web searches—is permitted, but all answers must be your own work and not copied.
- All work will be peer-reviewed (by yourself and randomly assigned peers). For expectations, grading rubric, and further details, see the Exercise and Peer Review Instructions. A good answer should:
  - **Clearly explain the reasoning and process**, not just present final results.
  - **Be well-organised and concise**, with logical structure, precise technical language, and properly labelled code snippets/figures/tables as needed.
  - **Demonstrate factual correctness** with relevant, sufficient evidence and well-justified claims.
  - **Show critical analysis**, including justification of methods, discussion of alternatives or limitations, and integration of course concepts to the extend applicable.
  - **Reflect master's-level understanding**: not just procedural work, but explanation, synthesis, and professional communication.
  - **Not look like a code dump**: unless explicitly requested, you do not need to include program code in your answer.
- The language of submission and review is English.
- Upload a single, well-formatted PDF. Double-check before submitting that all figures and sections are present; Jupyter Notebooks can produce incomplete PDFs. We recommend R Markdown (or Quarto). For help, visit https://rmarkdown.rstudio.com/lesson-1.html and https://bookdown.org/yihui/rmarkdown/. It is a good idea to learn a system for writing proper reports, also for future studies and life after graduation. Incomplete or unreadable submissions may result in no points awarded.
- Answer the problems in the order given.
- Consult Moodle's general instructions and grading criteria before starting.
- Main textbook: ISLR_v2 ("An Introduction to Statistical Learning with Applications in R", 2nd edition, James et al.) or ISLP (Python version)—either is acceptable. You may use other materials as well.
- Submit as early as possible: you can revise and resubmit until the deadline. Due to peer review, we cannot accept late submissions. If you miss the deadline (and don't even submit empty PDF) you will loose points from the exercise set, including peer review points. Check in advance, well before the deadline date, that you can produce legible PDF files (especially with Jupyter notebooks students have often had last-minute surprises as PDF generation has not worked as expected). You have been warned.
- Work at your own pace, but refer to the study schedule on Moodle to avoid last-day rush.
- If you require special arrangements (e.g., sudden illness), inform staff promptly. By default, we assume you have followed the study schedule up to the time of notification; if you contact us at the last moment, we assume you have had time to do all exercises before the force majeure reason.
- If you find the problems difficult:
  - Follow the study schedule. Solving tasks after corresponding lectures reduces difficulty and stress.
  - Attend lectures and read the recommended textbook chapters.
  - Participate in exercise workshops and ask for help in workshops or the Team.
  - Talk with fellow students.

Full details on peer review, grading expectations, and evaluation criteria for master's-level work are found in the separate Exercise and Peer Review Instructions file.

Suggested study schedule for Block 3

Please refer to the suggested study schedule on the course Moodle page.

Please remember to book enough time in your schedule to solve the problems. It helps if you have read the respective book chapters recommended in the lecture schedule above and attended the classes and the exercise workshops. Avoid leaving everything to the last day!

Default policy in the case of force majeure reasons

If you need a special arrangement due to a sudden illness or other force majeure reason, contact us as soon as possible. As a default policy, we assume that in such cases, you have followed the schedule recommended above up until you inform us of the force majeure reason. For example, if you contact us on the last day before the deadline, we assume you have had time to solve most of the problems. Therefore, please solve the problems as suggested above and submit your answer early (you can update the answer until the deadline)!

Problem 17

[9 points]

Objectives: k-means loss and Lloyd's algorithm

In this problem, we will study the (naive) k-means algorithm (Lloyd's algorithm). You should be able to do this problem with a pen and paper. See Section 12.4.1 of ISLR_v2.

Task a

Answer the following:

1. For what kinds of tasks can we use the k-means algorithm?
2. What are the algorithm's *inputs* and *outputs*?
3. How should you interpret the results?

Task b

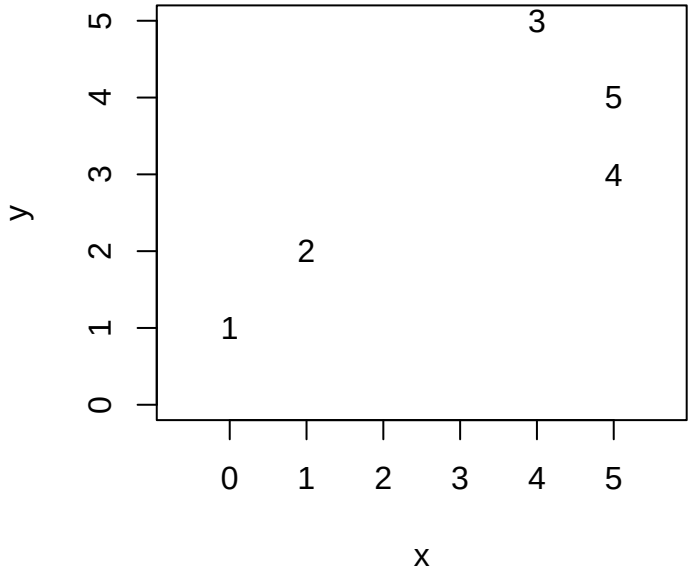
Define the objective (or cost) function the k-means algorithm tries to minimise.

What can you say about the objective function's value during the algorithm iteration?

Task c

Consider the following toy data set:

	1	2	3	4	5
x	0	1	4	5	5
y	1	2	5	3	4



Sketch a run of the (naive) k-means algorithm using $K = 2$ and initial prototype (mean) vectors $\mu_1 = (1, 4)$ and $\mu_2 = (4, 1)$. Write down the calculation procedure at each iteration and report the cluster memberships, the prototype vectors, and the value of the objective function at each iteration.

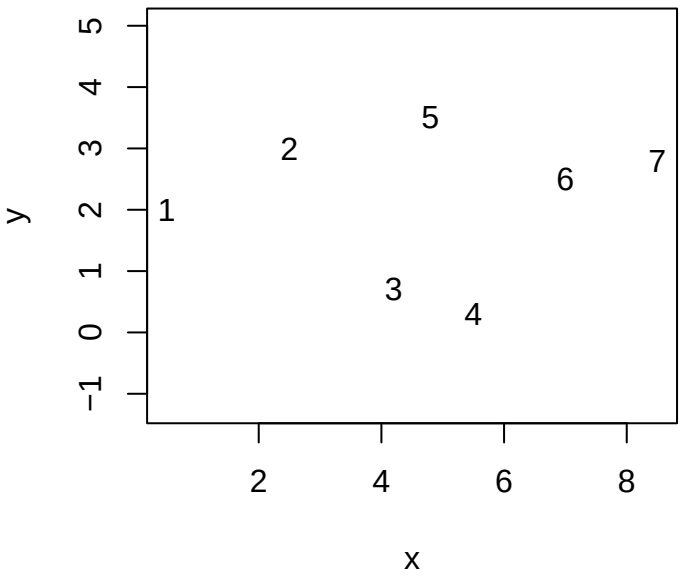
Problem 18

[9 points]

Objectives: understanding hierarchical clustering algorithms

In this problem, you will apply *hierarchical clustering* on a toy data set. You should be able to do this problem with a pen and paper. The toy data are shown below in a table and a figure.

	1	2	3	4	5	6	7
x	0.5	2.5	4.2	5.5	4.8	7.0	8.5
y	2.0	3.0	0.7	0.3	3.5	2.5	2.8



We show the pairwise Euclidean distances between the data points in the table below.

	1	2	3	4	5	6	7
1	0.00	2.24	3.92	5.28	4.55	6.52	8.04
2	2.24	0.00	2.86	4.04	2.35	4.53	6.00
3	3.92	2.86	0.00	1.36	2.86	3.33	4.79
4	5.28	4.04	1.36	0.00	3.28	2.66	3.91
5	4.55	2.35	2.86	3.28	0.00	2.42	3.77
6	6.52	4.53	3.33	2.66	2.42	0.00	1.53
7	8.04	6.00	4.79	3.91	3.77	1.53	0.00

Task a

Sketch a run of the basic agglomerative hierarchical clustering algorithm using the *single link (min)* notion of dissimilarity between clusters.

Use the above figure in your answer sheet to indicate step-by-step how the algorithm forms clusters. You don't need to show detailed calculations but indicate the cost of the join you select.

Visualise the result as a dendrogram.

Task b

Repeat Task a using the *complete link (max)* dissimilarity. Compare the results.

Problem 19

[10 points]

Objectives: practical application of k-means and hierarchical clustering

For this problem, you will apply clustering on the term project dataset (`train.csv`). See Section 12.5.3 of ISLR_v2.

We will use only the real-valued mean measurements (variable names ending with `.mean`) and the class variable (`class4`), resulting in 50+1 data columns. Unless otherwise instructed, scale the variables to zero mean and unit variance.

Task a

You can use a library function such as `kmeans` in R or `KMeans` in `scikit-learn`. Cluster the rows of the data matrix. Plot the k-means loss as a function of the number of clusters from 1 to 20.

Should you normalise the columns? What effect does the normalisation of the columns have?

Task b

Using Lloyd's algorithm and scaled variables, cluster the rows into four clusters and make a confusion matrix (contingency table where the rows are the actual classes, columns the cluster indices, and entries the counts of rows). Order the rows and columns so that the sum of diagonal entries in your contingency table is maximised (see the Instructions below).

Where are most "errors" made if you use your clusters as a rudimentary classifier (i.e., if you associate each cluster with a class)? See the hint below!

Task c

(i) Repeat the clustering of Task b with 1000 different random initialisations, make a histogram of the losses, and then answer the following:

- What are the minimum and maximum k-means losses for your 1000 random initialisations?
- How many initialisations would you expect to have to obtain one reasonably good loss for this dataset and number of clusters? A reasonably good loss here is a solution with a loss within 1% of the best loss out of your 1000 losses.

(ii) Repeat (i) using a smarter initialisation, such as kmeans++. Discuss how this initialisation affected your results.

Task d

(i) Cluster the same data with agglomerative hierarchical clustering using two different linkage functions. Produce a dendrogram and the corresponding flat clustering (e.g., by splitting the dendrogram with `cutree` in R).

(ii) Compare their properties. For example, you can compare the sizes of clusters by looking at confusion matrices.

(iii) Find and report at least one interesting feature or reproduce some properties of hierarchical clustering discussed in the lecture (e.g., differences between the linkage functions). *Hint*: See Sect. 12.5.3 of James et al. for examples in R.

Instructions and hints

In R, you can run Lloyd's algorithm with one random initialisation on a scaled dataset as follows:

```r
# R
cl <- kmeans(scale(npf[, vars]), centers = 4, algorithm = "Lloyd", nstart = 1, iter.max = 100)
```

In Python `KMeans` from `sklearn.cluster` uses kmeans++ as the default initialisation. To get random initialisation use `init="random"`:

```python
# Python
from sklearn.cluster import KMeans
cl = KMeans(n_clusters = 4, n_init = 1, init="random").fit(npf[vars])
```

In Task b, you need to combine the class variables and the cluster indices; in Task d, you must compare clusterings. Because all permutations of cluster indices are equally good, finding the best match between the known classes (`class4`) and the cluster indices is helpful. One way to do this is the Hungarian algorithm which finds a permutation of cluster indices that maximises the sum of the diagonal entries in the confusion matrix. We give an example in R below, using `solve_LSAP` from the `clue` library.

```r
# R
library(clue)
## Create confusion matrix between the known classes (class 4) and cluster indices.
tt <- table(npf$class4, cl$cluster)
## Find a permutation of cluster indices that best matches the classes in class4.
tt <- tt[, solve_LSAP(tt, maximum = TRUE)]
```

In Scipy, you can use `linear_sum_assignment` from `scipy.optimize`.

```python
# Python
from scipy.optimize import linear_sum_assignment
tt = pd.DataFrame({"class": npf.class4, "cluster": cl.labels_})
tt = tt.groupby(["class", "cluster"]).size().unstack()
lsa = linear_sum_assignment(tt, maximize = True)
pairs = {tt.index[i]: tt.columns[j] for i, j in zip(*lsa)}
```

In R, you can use the following function to use the kmeans++ algorithm to find the initial cluster centroids. You can give the resulting centroids directly as a parameter to the `kmeans` function (parameter `centers`).

```r
# R

#' Find initial kmeans++ centroids
#'
#' Reference: Arthur & Vassilivitskii (2007) k-means++: the
#' advantages of careful seeding. In Proc SODA '07, 1027-1035.
#'
#' @param x numeric matrix, rows correspond to data items
#' @param k integer, number of clusters
#'
#' @return a matrix of cluster centroids, which can be fed to `kmeans`
kmeansppcenters <- function(x, k) {
  x <- as.matrix(x)
  n <- nrow(x)
  centers <- matrix(NA, k, ncol(x))
  p <- rep(1 / n, n)
  for (i in 1:k) {
    centers[i, ] <- x[sample.int(n, size = 1, prob = p), ]
    dd <- rowSums((x - (rep(1, n) %o% centers[i, ]))^2)
    d <- if (i > 1) pmin(d, dd) else dd
    if (max(d) > 0) {
      p <- d / sum(d)
```

```
    }
  }
  centers
}

cl <- kmeans(scale(npf[, vars]),
  centers = kmeansppcenters(scale(npf[, vars]), 4), algorithm = "Lloyd", iter.max = 100
)
```

In Python, `KMeans` from `sklearn.cluster` uses kmeans++ as the default initialisation.

Problem 20

[10 points]

Objectives: uses of PCA

In this problem, you will apply Principal Component Analysis (PCA) on the same dataset as in the problem above (`train.csv`).

Task a

Compute and show a PCA projection of the data into two dimensions. Indicate the class index (`class4`), by colour and the glyph's shape. Remember to include a legend that indicates which colour/shape corresponds to which class.

Task b

(i) Compute and plot the proportion of variance explained (PVE), and the cumulative PVE for the principal components.

(ii) Study the effects of different normalisations. Compare the difference between not scaling the data and normalising each variable to zero mean and unit variance. Why does it seem for unnormalised data that fewer components explain a large proportion of the variance compared to the normalised data? (Hint: See Sect. 12.2.4 of ISLR_v2)

Task c

Pick one classification algorithm that would work with this data and choose one of the challenge performance measures (binary accuracy, multiclass accuracy, or perplexity). Split the data at random into training and validation sets of equal sizes.

(i) Train your classification algorithm without the dimensionality reduction on the training set and report your chosen performance measure on the validation set.

(ii) Do the same on the data after reducing the dimensionality with PCA (see Task b above). How does the performance of your classifier vary with the (reduced) dimensionality? Is there an "optimal" dimensionality which gives you the best performance on the validation set?

Hint: Notice that you can do PCA on the combined training and validation sets; this is a simple form of semi-supervised learning: you utilise the structure of the validation/test set even if you don't know the class labels!

Task d (optional)

This is a bonus task for which no points are awarded.

Repeat the task a above, but this time with isometric multidimensional scaling (MDS) and t-distributed stochastic neighbour embedding (t-SNE).

If you use R, you can use `isoMDS` from library `MASS` and `tsne` from library `tsne`. Notice that neither MDS nor t-SNE algorithms are guaranteed to converge to a local optimum. Therefore, a good initial position is essential (one common choice is the PCA solution).

```
# R
library(MASS)
library(tsne)
d <- dist(scale(npf[, vars]))
## cmdscale essentially does PCA. Both MDS and t-SNE are sensitive to initial
## configuration and the PCA initialisation generally leads to reasonable convergence.
## (Even though R isoMDS and tsne can handle bad initial config fine,
```

```
## we are being explicit here.)
y <- cmdscale(d, 2)
## isometric MDS embedding coordinates
x.mds <- isoMDS(d, y = y)$points
## t-SNE embedding coordinates
x.tsne <- tsne(d, initial_config = y, k = 2)
```

In Python, you can use `MDS` and `TSNE` from `sklearn.manifold`:

```
from sklearn.manifold import MDS, TSNE
x_mds = MDS().fit_transform(npf[vars])
x_tsne = TSNE(init="pca").fit_transform(npf[vars])
```

Problem 21

[2 points]

Objectives: self-reflection, giving feedback on the course

Tasks

- Write a learning diary of the topics of lectures 9-12 and this exercise set.

Instructions

Guiding questions: What did I learn? What did I not understand? Was there something relevant for other studies or (future) work? The length of your reply should be 1-3 paragraphs of text.