(name of your school)

A I P R O J E C T R E P O R T

on

“GROUPING HOMES BY WATER USE”

Statistical Data · SDG 6: Clean Water and Sanitation · SDG 11: Sustainable Cities and Communities

Class X

Artificial Intelligence — Subject Code 417
Part C: Practical and Project Work

Session _____

Submitted by:

Name _____

Roll No. _____

Under the guidance of:

_____

Teacher (Artificial Intelligence)

CERTIFICATE

This is to certify that _____ of Class X, Roll No. _____, has satisfactorily completed the Artificial Intelligence project entitled **“Grouping Homes by Water Use”** during the academic session _____, under my guidance and supervision.

The work has been carried out as the Project Work of Part C of Artificial Intelligence (Subject Code 417), in accordance with the curriculum prescribed by the Central Board of Secondary Education.

_____ _____

Teacher-in-charge External Examiner

_____

Principal

Date: _____

ACKNOWLEDGEMENT

I would like to thank _____, my Artificial Intelligence teacher, for the guidance and encouragement given throughout this project.

I am grateful to the Principal of _____ for the computer lab and the facilities that made this work possible.

My thanks go to _____, who helped me understand the problem this project tries to solve and how it is handled today.

I thank my classmates, who tried the model and told me where it went wrong, and my family for their support and patience while the project was built.

_____

(Candidate)

INDEX

SL. NO.	TOPIC	PAGE NO.
1	Introduction	5
2	Problem Scoping	5
3	Data Acquisition	6
4	Data Exploration	7
5	Modelling	9
6	Evaluation	10
7	Deployment	11
8	Ethical Concerns	11
9	The No-Code Version	12
10	Technical Details	12
11	Source Code	13
12	Sample Output	17
13	Testing	18
14	Results and Future Scope	19
15	Conclusion	19
16	References	20

> Page numbers refer to the pages of this report as printed. The cover page, certificate, acknowledgement and this index are not numbered.

INTRODUCTION

Every summer the society's tank runs low, and every summer the same notice goes up: "Please save water." Nobody reads it, because it speaks to everyone and so to no one.

The flats are not all the same. Some already use little. Some water a garden for half an hour a day. Some wash a car every other day. Each needs different advice.

But nobody has sorted the flats into groups, and there is no column saying who is careful and who is not. This project asks the computer to find those groups on its own.

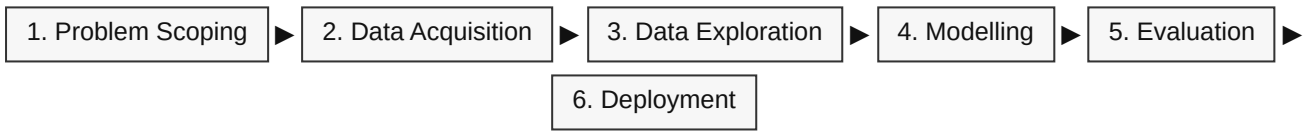
Who would use it: The water committee of a housing society with a meter in every flat, or a municipal water department planning a saving campaign.

Link to the Sustainable Development Goals

SDG 6 — Clean Water and Sanitation; SDG 11 — Sustainable Cities and Communities. SDG 6 asks for water to be used efficiently, and many Indian cities run short every summer. A housing society that knows which homes use the most water, and why, can give each group advice that fits — and save far more than a single notice on the board ever does.

The AI Project Cycle

This is a **Statistical Data** project. It follows the six stages of the AI Project Cycle, and the report that follows takes them in order, one section each.



PROBLEM SCOPING

The problem was scoped with the **4Ws Problem Canvas**: who has the problem, what it is, where it happens, and why solving it matters.

Who? STAKEHOLDERS • The society's water committee • Every family in the society, who share one tank	**What?** THE PROBLEM • Water runs short every summer • One general notice does not change anybody's habits
Where? CONTEXT • A housing society with a water meter in every flat • In the summer months	**Why?** BENEFIT • Advice that fits a household is more likely to be followed • Saving water keeps the tank from running dry

Problem statement

> Our **the water committee of our housing society** have a problem of **getting families to save water** when **the tank runs low every summer and a general notice is ignored**. An ideal solution would **find groups of similar households, so each group can be given advice that fits it**.

How it is done today

A notice on the board. Cheap and quick, and almost always ignored, because it is written for everyone at once.

The monthly water bill. Some societies split the bill by meter. It tells a family how much they used, but not whether that is a lot, or what to change.

Fixed rules. For example, "no garden watering after 8 am". Useful, but the committee has to guess which habits matter most.

Objectives

1. To read a month of water-meter data for 300 flats from a CSV file
2. To remove blank and impossible (negative) meter readings
3. To compare the society's use with the city norm of 135 litres a person a day
4. To find groups of similar flats with k-means clustering, without any labels
5. To check how many groups is the right number, and whether the groups are stable
6. To place new flats into a group and suggest a tip for each

DATA ACQUISITION

Each row is one flat: the litres used per person per day over the month (from the meter, divided by the number of people), the minutes a day the garden is watered, and the car washes in a week. There is no label — no column says which group a flat belongs to.

The file has 300 rows.

For a real project, ask the society office for a month of meter readings and send a short form to every flat asking about gardens and cars. Use flat numbers only, and get the committee's permission first.

> **About this dataset.** The data used here is a LambdaLab sample, made by a program to behave like the real thing so that the project could be run and tested. It is not real data. If you submit this project, replace it with data you collected or a public dataset, and say here where it came from and that you had permission to use it.

Features and label

The features are what the model is given; the label is the answer it must learn to give.

COLUMN	TYPE	WHAT IT HOLDS	ROLE
`flat_no`	text	Flat number, never a name	not used
`litres_per_person`	integer	Litres used per person per day, from the meter	feature
`garden_minutes`	integer	Minutes a day the garden or plants are watered	feature
`car_washes_week`	integer	Car or scooter washes in a week	feature

The file water_use.csv

Three hundred flats and one month of water use. There is no label column. Some meter readings are blank and two are negative. The whole file is **300 rows, 3.9 KB**.

The first eight rows of water_use.csv

FLAT_NO	LITRES_PER_PERSON	GARDEN_MINUTES	CAR_WASHES_WEEK
F101	91	9	2
F102	196	39	4
F103	234	36	4
F104	146	10	1
F105	136	9	2
F106	131	9	1
F107	103	4	0
F108	91	4	0

DATA EXPLORATION

Before any model was built, the data was checked for problems and then drawn, to find the patterns a model could use.

Problems found in the data

WHAT WAS WRONG	HOW MUCH	WHAT WAS DONE
Blank meter readings	3 (removed)	A flat with no reading cannot be grouped, so it is removed and counted.
Negative litres	2 (removed)	Water used cannot be less than zero. A negative number means the meter was reset in the middle of the month, so the reading is wrong and is removed. A reading of exactly 0 is kept — the flat may have been empty.
Features on different scales	litres in hundreds, car washes under 5	Each feature is rescaled to run from 0 to 1, so litres do not decide the groups on their own.

Figure 1: Water used by one person in a day

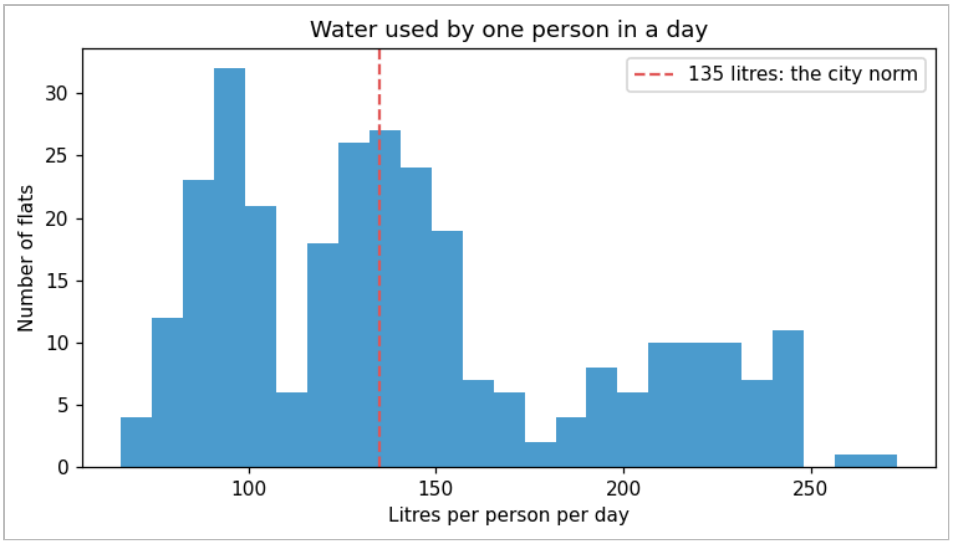


Figure 1 — Water used by one person in a day, drawn by the program as a histogram and saved as chart1_litres.png.

How to read it. Each bar counts the flats in a range of litres per person per day. The red dashed line is 135 litres, the amount city planners use for one person in a day.

145 of 295 flats are above the norm. The histogram is not one smooth hill — there is a separate bump on the right, around 200 litres. A bump like that is a sign that the data holds more than one kind of household.

Figure 2: Every flat, before grouping

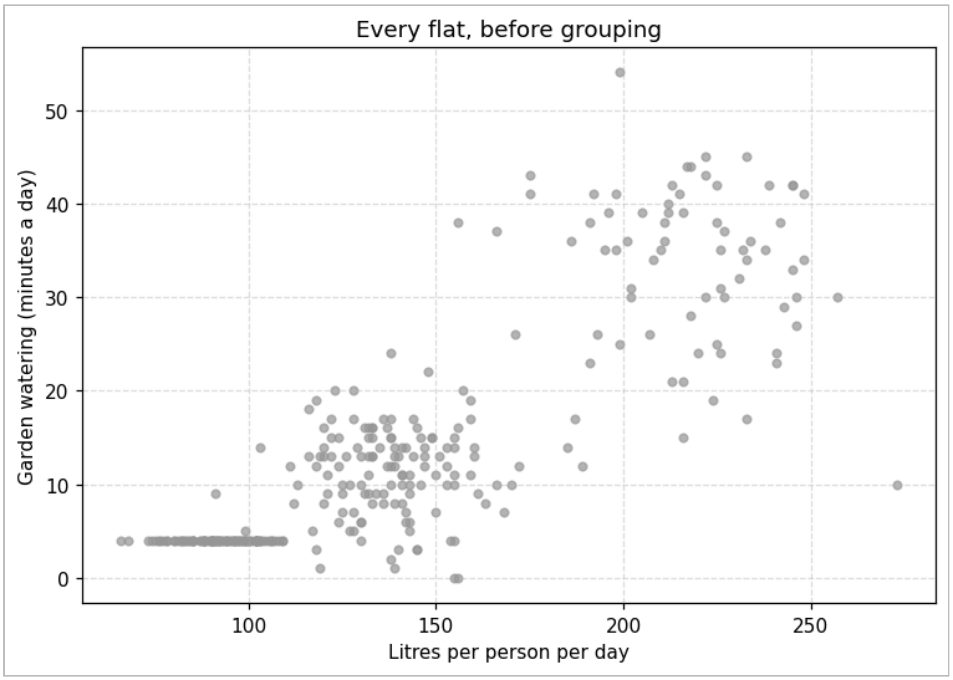


Figure 2 — Every flat, before grouping, drawn by the program as a scatter plot and saved as chart2_before.png.

How to read it. Each dot is one flat: litres per person across, garden minutes up. All the dots are grey because there are no labels.

Even in grey, the dots are not spread evenly. There is a crowd at the bottom left, another in the middle, and a looser group at the top right. Our eyes see groups — the question is whether the computer finds the same ones.

Figure 3: The three groups k-means found

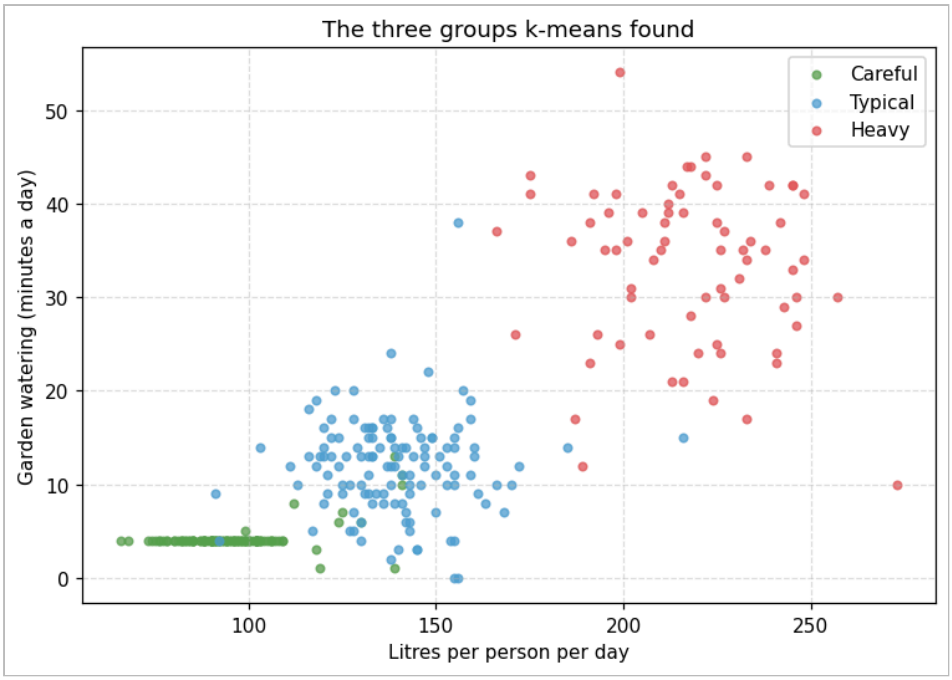


Figure 3 — The three groups k-means found, drawn by the program as a scatter plot and saved as chart3_groups.png.

How to read it. The same scatter plot, now coloured by the group each flat was put in.

The computer found the three crowds. Careful flats use about 96 litres a person and hardly water a garden. Typical flats use about 139. Heavy flats use about 217 litres and water a garden for over half an hour a day — the garden is a large part of why they are heavy.

The names Careful, Typical and Heavy were given by the program after the groups were found, by sorting them by average litres. K-means itself only says group 0, 1 and 2. Giving the groups meaning is a person's job.

MODELLING

AI model ▶ Learning based ▶ Unsupervised learning ▶ Clustering

There is no label: no column says who is careful and who is heavy. When a model has to find patterns in data without being told the answers, it is unsupervised learning. Finding groups of similar items is clustering.

How the model works

The model is k-means, with k = 3 groups. It works in four steps. One: pick three flats at random as the first group centres. Two: put every flat in the group whose centre is nearest. Three: move each centre to the middle (the average) of the flats in its group. Four: repeat steps two and three until the groups stop changing.

K-means has one known weakness, and the program shows it. If two of the random first centres happen to land in the same crowd, it can get stuck with a poor grouping. In this run, one of the five starts got stuck. So the program runs k-means from 5 different random starts and keeps the grouping with the smallest spread — 4 of 5 starts found that same best grouping. Orange's k-Means widget does exactly this, and calls it re-runs.

EVALUATION

An unsupervised model cannot have an accuracy, because there are no right answers to compare with. It has to be judged in other ways, and this project uses two.

First, is 3 the right number of groups? The elbow chart answers that: the spread falls a lot up to k = 3 and only a little after it.

Second, are the groups real, or did the computer invent them? If they are real, k-means should find the same groups from different random starts. Four of the five starts ended with exactly the same grouping, and the fifth got stuck in a clearly worse one (a larger spread). That is strong evidence the three groups are really in the data.

Finally, a person has to check that the groups make sense. Here they do: each group has a clear story — careful users, typical users, and users with a garden and a car to wash.

Figure 4: Choosing k: how tight the groups are

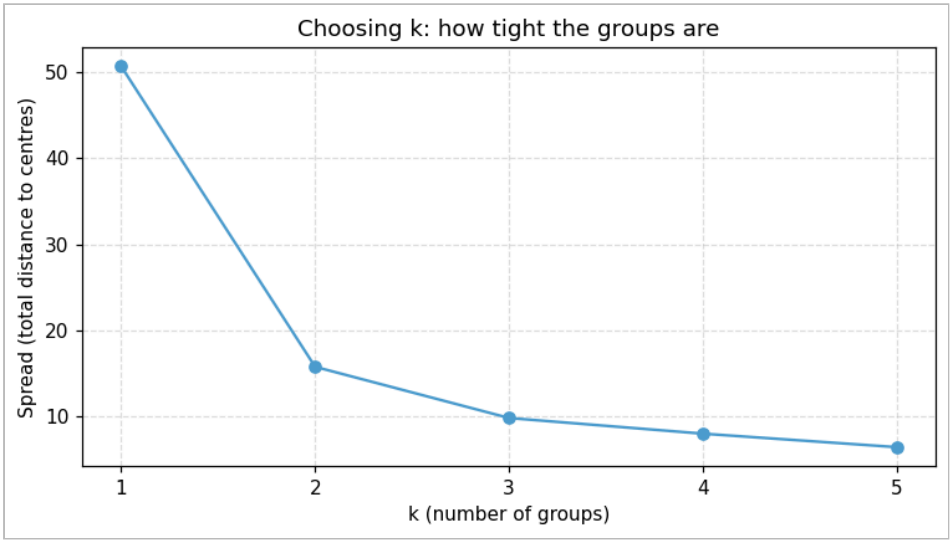


Figure 4 — Choosing k: how tight the groups are, drawn by the program as a line chart and saved as chart4_elbow.png.

How to read it. Across is k, the number of groups. Up is the spread: the total distance from every flat to the centre of its group. Lower means tighter groups.

Going from 2 groups to 3 cuts the spread by 5.94. Going from 3 to 4 cuts it by only 1.82. The line bends — like an elbow — at k = 3. After that, extra groups only split real groups into smaller pieces. So 3 is the right number.

Figure 5: Flats in each group

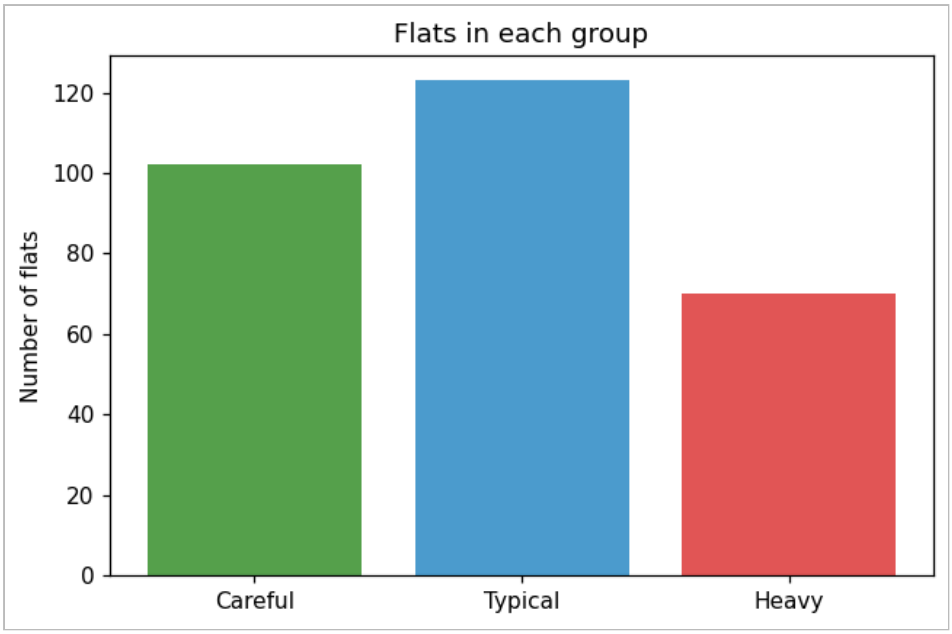


Figure 5 — Flats in each group, drawn by the program as a bar chart and saved as chart5_group_sizes.png.

How to read it. One bar per group: how many flats are in it.

The heavy group is the smallest, at about a quarter of the society. That is useful: the committee can talk to 70 households about gardens and cars instead of 295 about everything.

DEPLOYMENT

When a new family moves in, its first month of readings is placed in the group whose centre is nearest, and the program prints the tip for that group: a thank-you for careful users, help with taps for typical users, and garden and car advice for heavy users.

In real use the committee would run this every month and send each group its own message. Next summer, running it again would show whether flats have moved from the heavy group to the typical one.

ETHICAL CONCERNS

A model that works is not finished until it has been checked for fairness (bias), for whether it can explain itself (transparency), and for who answers for its mistakes (accountability).

Labels can hurt

"Heavy" is a description of water use, not a judgement of a family. A large family or someone who is unwell may need more water. Messages to each group should be helpful and private, never a list on the notice board.

Privacy

Water use shows when a family is at home. Readings should be stored with flat numbers, seen only by the committee, and used only for saving water.

Transparency

The groups are explained by their averages, which anyone can read. A family can see why it was put in a group.

THE NO-CODE VERSION

Orange's k-Means widget does the same clustering, including the re-runs from several random starts, and its Silhouette Plot is another way to check the groups. The tool used is **Orange Data Mining**.

STEP	WIDGET	WHAT IT DOES
1	**File**	Load water_use.csv. Set flat_no as a meta attribute. Leave the target empty — there is no label.
2	**Select Rows**	Keep rows where litres_per_person is at least 0.
3	**Preprocess**	Normalise the features to the 0–1 range.
4	**k-Means**	Try 2 to 5 clusters. Orange scores each k; choose 3. Its re-runs setting is the same idea as this program's 5 starts.
5	**Scatter Plot**	Plot litres_per_person against garden_minutes and colour by Cluster — the same as chart 3.
6	**Box Plot**	Split litres_per_person by Cluster to describe each group.

TECHNICAL DETAILS

Domain of AI	Statistical Data
Programming language	Python 3
Libraries	pandas, NumPy and matplotlib (Unit 7 of the curriculum)
Where to run it	Jupyter Notebook (water_groups.ipynb), or any Python editor (water_groups.py)
No-code tool	Orange Data Mining
Model type	Learning based → Unsupervised learning → Clustering
Charts drawn	5

Setting it up

1. Install Python 3 (Anaconda includes Jupyter Notebook and all three libraries).
2. If they are missing, install the libraries: **pip install pandas numpy matplotlib**
3. Keep the program and its data together in one folder.
4. Run **python water_groups.py**, or open **water_groups.ipynb** and run each cell in order.

The library calls it uses

CALL	STAGE	WHAT IT DOES
`pd.read_csv()`	Stage 2	Reads the CSV file
`data[data["litres_per_person"] >= 0]`	Stage 3	Removes impossible negative readings
`plt.hist / plt.axvline`	Stage 3	The histogram and the 135-litre line

`scaled.sample(k, random_state=...)`	Stage 4	Picks k random flats as the first centres
`distance.idxmin(axis=1)`	Stage 4	Puts each flat in the group of its nearest centre
`scaled.groupby(group).mean()`	Stage 4	Moves each centre to the middle of its group
`distance.min(axis=1).sum()`	Stages 4, 5	The spread: total distance from each flat to its centre
`replace({...})`	Stage 4	Gives the groups names after they are found
`plt.scatter / plt.plot / plt.bar`	Stages 3, 5	Scatter plots, the elbow chart and the group sizes

SOURCE CODE

The complete program. Every figure and number in this report came from running it.

`water_groups.py`

```
# ------------------------------------------------------------------------
# water_groups.py
#
# AI project on Statistical Data: finding groups of households by how they
# use water, so that a housing society can give each group the right advice.
#
# CBSE Class X, Artificial Intelligence (417), Part C project work.
# SDG 6 (Clean Water and Sanitation) and SDG 11 (Sustainable Cities).
#
# This is UNSUPERVISED learning. Nobody has told the program which flats are
# careful and which are wasteful - there is no label column at all. The
# model, k-means clustering, finds the groups by itself:
#   1. pick k flats at random as the first group centres
#   2. put every flat in the group whose centre is nearest
#   3. move each centre to the middle (the mean) of its group
#   4. repeat steps 2 and 3 until nothing changes
#
# Libraries: pandas (the data), numpy (the arithmetic), matplotlib (charts).
# ------------------------------------------------------------------------

# %% Stage 1 - Problem scoping
print("=" * 60)
print("GROUPING HOMES BY WATER USE")
print("=" * 60)
print("Who   : the housing society's water committee")
print("What  : one notice for everyone is ignored by everyone")
print("Where : a society with a water meter in every flat")
print("Why   : advice that fits each group saves more water")
print()

# %% Stage 2 - Data acquisition
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

pd.set_option("display.width", 120)
pd.set_option("display.max_columns", 20)

data = pd.read_csv("water_use.csv")
```

```
print("--- Stage 2: Data acquisition ---")
print("Rows read    :", len(data))
print("Columns read :", len(data.columns))
print()
print("The first 10 rows:")
print(data.head(10))
print()

# %% Stage 3 - Data exploration
print("--- Stage 3: Data exploration ---")
features = ["litres_per_person", "garden_minutes", "car_washes_week"]

missing = data["litres_per_person"].isnull().sum()
data = data.dropna(subset=["litres_per_person"])
print("Rows with no meter reading     :", missing, "(removed)")

# Water used cannot be negative; a negative number means the meter was reset.
negative = (data["litres_per_person"] < 0).sum()
data = data[data["litres_per_person"] >= 0]
print("Rows with negative litres      :", negative, "(removed)")
print("Rows left                      :", len(data))
print()

litres = data["litres_per_person"]
print("Litres per person a day - mean   :", round(np.mean(litres), 1))
print("Litres per person a day - median :", np.median(litres))
over = (litres > 135).sum()
print("Flats above 135 litres a person  :", over, "of", len(data))
print()

# Chart 1: how much water a person uses in a day, across the society.
plt.figure(figsize=(7, 4))
plt.hist(litres, bins=25, color="#4e9fd1")
plt.axvline(135, color="#e15759", linestyle="--", label="135 litres: the city norm")
plt.title("Water used by one person in a day")
plt.xlabel("Litres per person per day")
plt.ylabel("Number of flats")
plt.legend()
plt.tight_layout()
plt.savefig("chart1_litres.png")
plt.close()

# Chart 2: litres against garden watering, before any grouping.
plt.figure(figsize=(7, 5))
plt.scatter(data["litres_per_person"], data["garden_minutes"], color="#999999", alpha=0.7, s=18)
plt.title("Every flat, before grouping")
plt.xlabel("Litres per person per day")
plt.ylabel("Garden watering (minutes a day)")
plt.grid(True, linestyle="--", alpha=0.4)
plt.tight_layout()
plt.savefig("chart2_before.png")
plt.close()

# %% Stage 4 - Modelling
print("--- Stage 4: Modelling ---")

# Rescale every feature to run from 0 to 1, so litres (in hundreds) do not
# outweigh car washes (under 5).
low = data[features].min()
high = data[features].max()
scaled = (data[features] - low) / (high - low)
```

```
k = 3
starts = 5

# k-means can get stuck if two of its random first centres land in the same
# group. So it is run from 5 different random starts, and the grouping with
# the smallest SPREAD (the total distance from every flat to its own centre)
# is kept. Orange's k-Means widget does the same thing ("re-runs").
best_spread = None
same_as_best = 0
for start in range(starts):
    # Step 1: k flats chosen at random are the first centres.
    centres_try = scaled.sample(k, random_state=start).reset_index(drop=True)
    for step in range(20):
        # Step 2: distance from every flat to each centre; join the nearest.
        distance = pd.DataFrame()
        for c in range(k):
            distance[c] = ((scaled - centres_try.iloc[c]) ** 2).sum(axis=1)
        group_try = distance.idxmin(axis=1)
        # Step 3: move each centre to the mean of the flats in its group.
        centres_try = scaled.groupby(group_try).mean().reset_index(drop=True)
    spread = round(distance.min(axis=1).sum(), 4)
    print("Start", start + 1, "- spread :", spread)
    if best_spread is None or spread < best_spread:
        best_spread = spread
        group = group_try
        centres = centres_try
        same_as_best = 1
    elif spread == best_spread:
        same_as_best = same_as_best + 1
print("Starts that found the best grouping :", same_as_best, "of", starts)
print()

# Name the groups by how much water they use, smallest first.
order = data.groupby(group)["litres_per_person"].mean().sort_values().index
names = ["Careful", "Typical", "Heavy"]
data["group"] = group.replace({order[0]: names[0], order[1]: names[1], order[2]: names[2]})
print()

summary = data.groupby("group")[features].mean().round(1).reindex(names)
summary["flats"] = data["group"].value_counts().reindex(names)
print("What each group looks like (averages):")
print(summary)
print()

colours = ["#59a14f", "#4e9fd1", "#e15759"]

# Chart 3: the same scatter, now coloured by the group k-means found.
plt.figure(figsize=(7, 5))
for i in range(k):
    rows = data[data["group"] == names[i]]
    plt.scatter(rows["litres_per_person"], rows["garden_minutes"],
                color=colours[i], alpha=0.75, s=18, label=names[i])
plt.title("The three groups k-means found")
plt.xlabel("Litres per person per day")
plt.ylabel("Garden watering (minutes a day)")
plt.legend()
plt.grid(True, linestyle="--", alpha=0.4)
plt.tight_layout()
plt.savefig("chart3_groups.png")
plt.close()
```

```
# %% Stage 5 - Evaluation
print("--- Stage 5: Evaluation ---")

# There is no label, so there is no accuracy. Two other checks are used.

# There is no label, so there is no accuracy. Instead the program checks
# how tight the groups are for different values of k. Spread falls as k
# grows, because more centres means every flat has one closer by. The k
# after which it stops falling quickly is a good choice - the "elbow".
spreads = []
for kk in range(1, 6):
    best_kk = None
    for start in range(starts):
        c_kk = scaled.sample(kk, random_state=start).reset_index(drop=True)
        for step in range(20):
            d = pd.DataFrame()
            for c in range(kk):
                d[c] = ((scaled - c_kk.iloc[c]) ** 2).sum(axis=1)
            g = d.idxmin(axis=1)
            c_kk = scaled.groupby(g).mean().reset_index(drop=True)
        s_kk = d.min(axis=1).sum()
        if best_kk is None or s_kk < best_kk:
            best_kk = s_kk
    spreads.append(best_kk)
for kk in range(1, 6):
    print("Spread with k =", kk, ":", round(spreads[kk - 1], 2))
print()
print("Drop in spread from k = 2 to k = 3 :", round(spreads[1] - spreads[2], 2))
print("Drop in spread from k = 3 to k = 4 :", round(spreads[2] - spreads[3], 2))
print()

# Chart 4: the elbow chart.
plt.figure(figsize=(7, 4))
plt.plot(range(1, 6), spreads, marker="o", color="#4e9fd1")
plt.title("Choosing k: how tight the groups are")
plt.xlabel("k (number of groups)")
plt.xticks(range(1, 6))
plt.ylabel("Spread (total distance to centres)")
plt.grid(True, linestyle="--", alpha=0.4)
plt.tight_layout()
plt.savefig("chart4_elbow.png")
plt.close()

# Chart 5: how many flats in each group.
plt.figure(figsize=(6, 4))
plt.bar(names, summary["flats"], color=colours)
plt.title("Flats in each group")
plt.ylabel("Number of flats")
plt.tight_layout()
plt.savefig("chart5_group_sizes.png")
plt.close()

# %% Stage 6 - Deployment
print("--- Stage 6: Deployment ---")
tips = {
    "Careful": "Thank you note - keep it up",
    "Typical": "Fix dripping taps; use a bucket, not a shower",
    "Heavy":   "Water the garden at dawn; wash the car with a bucket",
}
new_flats = pd.DataFrame({
```

```
    "litres_per_person": [95, 140, 230],
    "garden_minutes":    [0, 12, 35],
    "car_washes_week":   [0, 1, 4],
})
new_scaled = (new_flats - low) / (high - low)
# Give each centre its group's name, so the nearest centre IS the answer.
centres = centres.rename(index={order[0]: names[0], order[1]: names[1], order[2]: names[2]})
for i in range(len(new_flats)):
    d = ((centres - new_scaled.iloc[i]) ** 2).sum(axis=1)
    g = d.idxmin()
    f = new_flats.iloc[i]
    print(int(f["litres_per_person"]), "litres,", int(f["garden_minutes"]), "garden min,",
          int(f["car_washes_week"]), "car washes ->", g, "->", tips[g])
```

SAMPLE OUTPUT

An actual run of the program, exactly as it appeared on screen.

Output — python water_groups.py

```
============================================================
GROUPING HOMES BY WATER USE
============================================================
Who   : the housing society's water committee
What  : one notice for everyone is ignored by everyone
Where : a society with a water meter in every flat
Why   : advice that fits each group saves more water

--- Stage 2: Data acquisition ---
Rows read    : 300
Columns read : 4

The first 10 rows:
  flat_no  litres_per_person  garden_minutes  car_washes_week
0    F101               91.0               9                2
1    F102              196.0              39                4
2    F103              234.0              36                4
3    F104              146.0              10                1
4    F105              136.0               9                2
5    F106              131.0               9                1
6    F107              103.0               4                0
7    F108               91.0               4                0
8    F109               90.0               4                0
9    F110              210.0              35                4

--- Stage 3: Data exploration ---
Rows with no meter reading     : 3 (removed)
Rows with negative litres      : 2 (removed)
Rows left                      : 295

Litres per person a day - mean   : 142.6
Litres per person a day - median : 133.0
Flats above 135 litres a person  : 145 of 295

--- Stage 4: Modelling ---
Start 1 - spread : 9.8589
Start 2 - spread : 9.8589
Start 3 - spread : 13.9302
Start 4 - spread : 9.8589
Start 5 - spread : 9.8589
Starts that found the best grouping : 4 of 5


What each group looks like (averages):
```

```
         litres_per_person  garden_minutes  car_washes_week  flats
group
Careful               95.7             4.3              0.3    102
Typical              138.9            11.9              1.4    123
Heavy                217.4            33.4              3.2     70

--- Stage 5: Evaluation ---
Spread with k = 1 : 50.63
Spread with k = 2 : 15.8
Spread with k = 3 : 9.86
Spread with k = 4 : 8.04
Spread with k = 5 : 6.48

Drop in spread from k = 2 to k = 3 : 5.94
Drop in spread from k = 3 to k = 4 : 1.82

--- Stage 6: Deployment ---
95 litres, 0 garden min, 0 car washes -> Careful -> Thank you note - keep it up
140 litres, 12 garden min, 1 car washes -> Typical -> Fix dripping taps; use a bucket, not a shower
230 litres, 35 garden min, 4 car washes -> Heavy -> Water the garden at dawn; wash the car with a bucket
```

TESTING

The program was tested by running it. Every case gives the whole program a small input of its own and records what actually came back.

TEST CASE	EXPECTED	ACTUAL RESULT	RESULT
The full dataset	`Rows read : 300`	`Rows read : 300`	Pass
Blank meter readings are removed	`Rows with no meter reading : 3 (removed)`	`Rows with no meter reading : 3 (removed)`	Pass
Negative litres (a meter reset) are removed	`Rows with negative litres : 2 (removed)`	`Rows with negative litres : 2 (removed)`	Pass
One of the five random starts gets stuck; the other four agree	`Starts that found the best grouping : 4 of 5`	`Starts that found the best grouping : 4 of 5`	Pass
The elbow: going from 2 to 3 groups helps far more than 3 to 4	`Drop in spread from k = 3 to k = 4 : 1.82`	`Drop in spread from k = 3 to k = 4 : 1.82`	Pass
Three obvious groups of 15: the careful group is found	`95 litres, 0 garden min, 0 car washes -> Careful -> Thank you note - keep it up`	`95 litres, 0 garden min, 0 car washes -> Careful -> Thank you note - keep it up`	Pass
...and the heavy group is found	`230 litres, 35 garden min, 4 car washes -> Heavy -> Water the garden at dawn; wash the car with a bucket`	`230 litres, 35 garden min, 4 car washes -> Heavy -> Water the garden at dawn; wash the car with a bucket`	Pass
Boundary: 0 litres is kept (an empty flat), -1 is removed	`Rows with negative litres : 1 (removed)`	`Rows with negative litres : 1 (removed)`	Pass
The data file is missing	`FileNotFoundError: [Errno 2] No such file or directory: 'water_use.csv'`	`FileNotFoundError: [Errno 2] No such file or directory: 'water_use.csv'`	Pass

RESULTS AND FUTURE SCOPE

What the project found

1. 145 of 295 flats use more than the city norm of 135 litres a person a day.
2. With no labels at all, k-means found three clear groups: careful (about 96 litres), typical (about 139) and heavy (about 217).
3. Heavy users water their gardens for over half an hour a day — the clearest single habit to change.
4. The elbow chart shows three is the right number of groups.
5. One random start in five got stuck in a worse grouping; running from several starts solves it.

Limitations

- One month only; a family on holiday looks careful
- Litres per person depends on the family size being reported correctly
- Only three features; bathing habits, washing machines and leaks are not included
- K-means finds round groups; a group with an odd shape would be split wrongly

Future scope

- Add a fourth feature, such as whether the flat has a washing machine
- Collect three months and see which flats move between groups
- Send each group its tip and measure the change the next month
- Try k-means on your own school's water or electricity use

What had to be learnt along the way

- The difference between supervised and unsupervised learning
- How k-means works, step by step, and why it is run from several starts
- How to choose k with the elbow method
- Why an unsupervised model cannot have an accuracy, and how it is judged instead

CONCLUSION

The project found three groups of households from meter readings alone, with nobody telling the computer what the groups should be. The groups are clear and they make sense: careful users, typical users, and heavy users who water gardens and wash cars.

Judging an unsupervised model needed different tools. There was no accuracy to report, so the project checked the number of groups with the elbow chart and checked that the groups were real by finding them again from different random starts.

The testing also found a real weakness of k-means: a single random start can get stuck. Running it from five starts fixed that, and seeing it go wrong was more useful than if it had always worked.

REFERENCES

1. CPHEEO, Manual on Water Supply and Treatment, Ministry of Housing and Urban Affairs — the 135 litres per person per day norm for cities with sewerage.
2. Jal Shakti Abhiyan, Ministry of Jal Shakti — https://jsactr.mowr.gov.in/
3. Orange Data Mining, k-Means widget — https://orangedatamining.com/widget-catalog/unsupervised/kmeans/
4. CBSE, Artificial Intelligence (Subject Code 417), Class X curriculum, 2026-27 — Part C, Practical and Project Work.
5. CBSE, Artificial Intelligence Facilitator Handbook, Class X.
6. pandas documentation — https://pandas.pydata.org/docs/
7. NumPy documentation — https://numpy.org/doc/
8. Matplotlib documentation — https://matplotlib.org/stable/
9. LambdaLab — https://www.lambdalab.in