_____

(name of your school)

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

on

“WHO NEEDS A HELPING HAND?”

Statistical Data · SDG 4: Quality Education · SDG 10: Reduced Inequalities

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 **“Who Needs a Helping Hand?”** 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	8
6	Evaluation	8
7	Deployment	10
8	Ethical Concerns	10
9	The No-Code Version	11
10	Technical Details	11
11	Source Code	12
12	Sample Output	16
13	Testing	17
14	Results and Future Scope	18
15	Conclusion	18
16	References	19

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

INTRODUCTION

Every year some students fail the final exam who could have passed with a little extra help. The trouble is timing: by the time a teacher is sure who is struggling, there is no time left.

Schools often use a simple rule of thumb to decide who to worry about — low attendance. But is attendance really the best early warning? Or is something else a better sign?

This project lets the computer test every simple rule it can make from the school's records and find the one that would have worked best. Because it is about children, it also asks carefully how such a model should and should not be used.

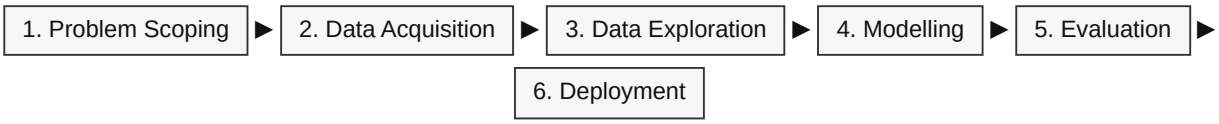
Who would use it: Class teachers and the school's academic coordinator, deciding who should be invited to remedial classes after the first unit tests.

Link to the Sustainable Development Goals

SDG 4 — Quality Education; SDG 10 — Reduced Inequalities. SDG 4 asks that every child completes school with real learning, not only the ones who find it easy. Students who fall behind are often the ones with the least help at home. Finding them early, so extra classes reach them before the exam, is a direct way to reduce that gap — SDG 10.

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 • Students who are falling behind • Class teachers and the academic coordinator	**What?** THE PROBLEM • Struggling students are often noticed only after the exam • Remedial classes have limited places
Where? CONTEXT • A secondary school • In the first months of the school year, after the first unit tests	**Why?** BENEFIT • Help that comes early can change the result • Places in remedial classes go to the students who need them

Problem statement

> Our **class teachers** have a problem of **finding students who need extra help** when **struggling students are usually noticed only after the final exam**. An ideal solution would **point out, early in the year, the students most likely to need support, so that help reaches them in time**.

How it is done today

The attendance rule. Flag any student below 75% attendance. It is simple and familiar, but it assumes that students who come to school are doing fine. This project tests that assumption.

The teacher's eye. Teachers often know. But one teacher sees a class for a few periods a week, and a quiet student who is lost can go unnoticed.

Waiting for the half-yearly exam. Clear evidence, but it arrives halfway through the year, when much of the time to help has gone.

Objectives

1. To read the records of 360 students from a CSV file, using roll numbers only
2. To remove records with missing marks or impossible attendance
3. To explore which records differ most between students who needed support and those who did not
4. To let the computer find the single best rule from the training data
5. To compare it with the attendance rule, using recall, precision and the F1 score
6. To set out the ethical rules any such system in a school must follow

DATA ACQUISITION

Each row is one student's record from the first months of the year: attendance, the share of homework handed in, the average of the unit tests, and hours of self-study a week from a short survey. The label says whether the student later scored below 40% in the final exam.

The file has 360 rows. Students are known only by roll number — there are no names anywhere.

Real student records are private and protected. A student must never collect classmates' marks for a project. A real version could only be built by the school itself, with permission, and with the data kept by the school. For your own project, use made-up data like this, or anonymous survey data that you have permission to collect.

> **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
`roll_no`	text	Roll number — never a name	not used
`attendance_pct`	integer	Attendance so far, as a percentage	feature
`homework_pct`	integer	Share of homework handed in, as a percentage	feature
`unit_test_avg`	decimal	Average of the unit tests so far, as a percentage	feature
`study_hours_week`	integer	Hours of self-study a week (from a survey)	feature
`needs_support`	Yes / No	Whether the student later scored below 40% in the final exam — the label	label

The file student_records.csv

Records of 360 students, by roll number. A few unit-test averages are blank, and two attendance figures are over 100%. The whole file is **360 rows, 7.6 KB**.

The first eight rows of student_records.csv

ROLL_NO	ATTENDANCE_PCT	HOMEWORK_PCT	UNIT_TEST_AVG	STUDY_HOURS_WEEK	NEEDS_SUPPORT
9001	85	42	49.1	2	No
9002	87	53	45.0	1	No
9003	83	44	46.1	4	Yes
9004	78	75	60.4	6	No
9005	67	56	57.9	5	No
9006	90	100	58.7	6	No
9007	88	73	72.1	9	No
9008	96	58	72.2	9	No

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 unit-test average	3 (removed)	These students missed the tests or the marks were not entered. Their rows are removed and counted; inventing a mark would be wrong.
Attendance above 100%	2 (removed)	A percentage cannot go above 100, so 910 and 108 are typing mistakes. The rows are removed. Exactly 100% is kept — it is possible.

Figure 1: Attendance and unit test average

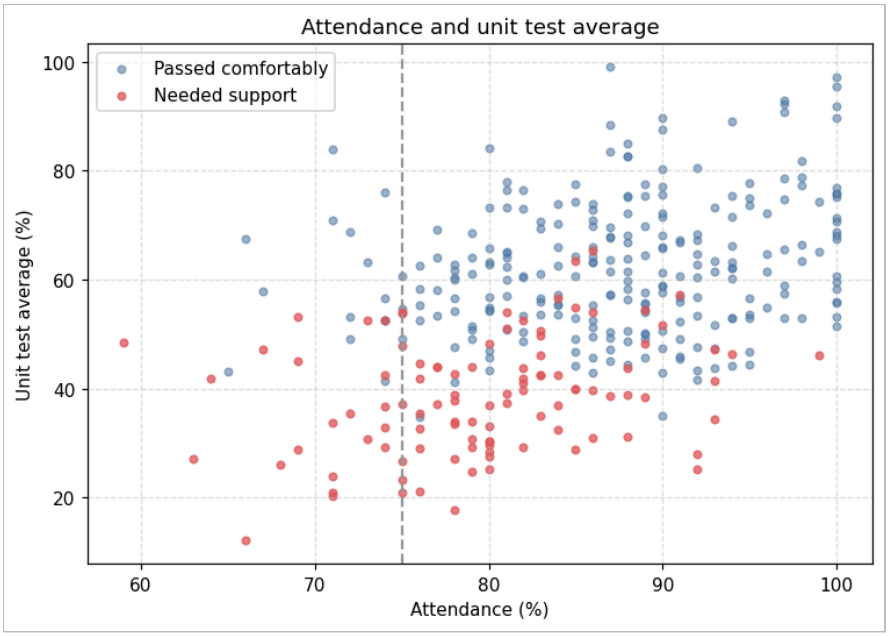


Figure 1 — Attendance and unit test average, drawn by the program as a scatter plot and saved as chart1_attendance_tests.png.

How to read it. Each dot is one student: attendance across, unit-test average up. Red dots needed support in the end. The dashed line is 75% attendance.

Look where the red dots are. Most of them are to the right of the dashed line — these students came to school regularly. What they share is that they are low down: their unit-test averages are low. Attendance alone would miss them.

Figure 2: Unit test average in the two groups

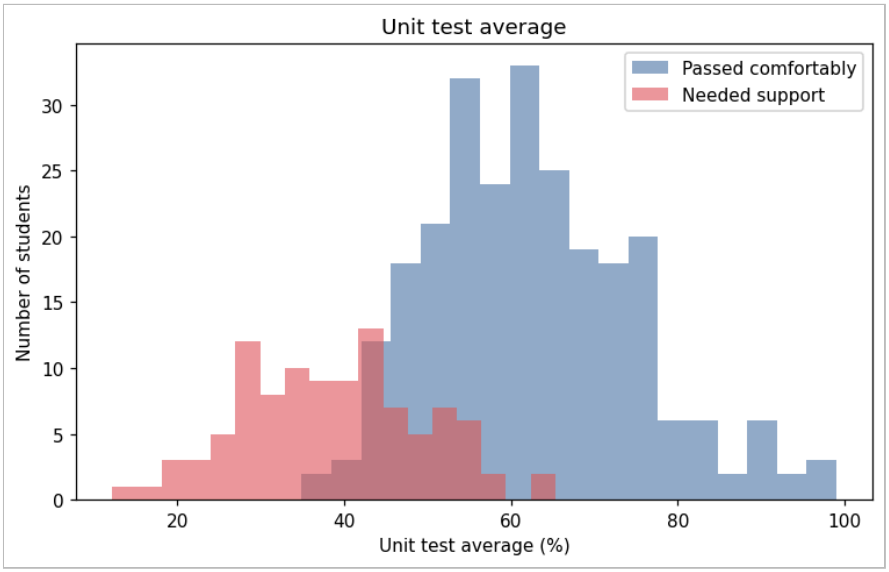


Figure 2 — Unit test average in the two groups, drawn by the program as a histogram and saved as chart2_unit_tests.png.

How to read it. Two histograms on top of each other, one for each group of students.

The two groups separate well on unit-test average, with only a small overlap around 40 to 50%. This chart suggests the unit tests are the best early warning — but the computer has not been asked yet.

MODELLING

AI model ▶ Learning based ▶ Supervised learning ▶ Classification

The answer is Yes or No — will this student need support? — so it is classification. The training records come with what really happened in the exam, so it is supervised learning.

How the model works

The records are split into 284 for training and 71 for testing.

The teacher's rule is written by a person: flag the student if attendance is below 75%.

The learned rule is found by the computer. It has the form "flag the student if this feature is below this cut-off". The program tries every feature and every value as the cut-off — 349 rules in all — counts the mistakes each would make on the training students, and keeps the one with the fewest. That is learning in its simplest form: the rule comes from the data, not from a person.

A rule with one question like this is the first branch of a decision tree, and is called a decision stump. Orange's Tree widget grows the full tree by asking the same kind of question again and again.

EVALUATION

Both rules were tested on the same 71 students that neither had seen.

The key question is which mistake is worse. A false negative is a student who needed help and did not get it: they may fail. A false positive is a student invited to a few extra classes they did not strictly need. Missing a student is worse, so recall matters most here.

On recall the two rules are far apart: 10.5 % for the attendance rule against 68.4 % for the learned rule. The learned rule's precision is also high, at 92.9 %, so the places in remedial classes would mostly go to the right students.

The most useful result is not the model at all. It is the evidence that the familiar rule of thumb does not work: in this data, most struggling students come to school regularly. A school relying on attendance alone would be looking in the wrong place.

Figure 3: How the computer chose its cut-off

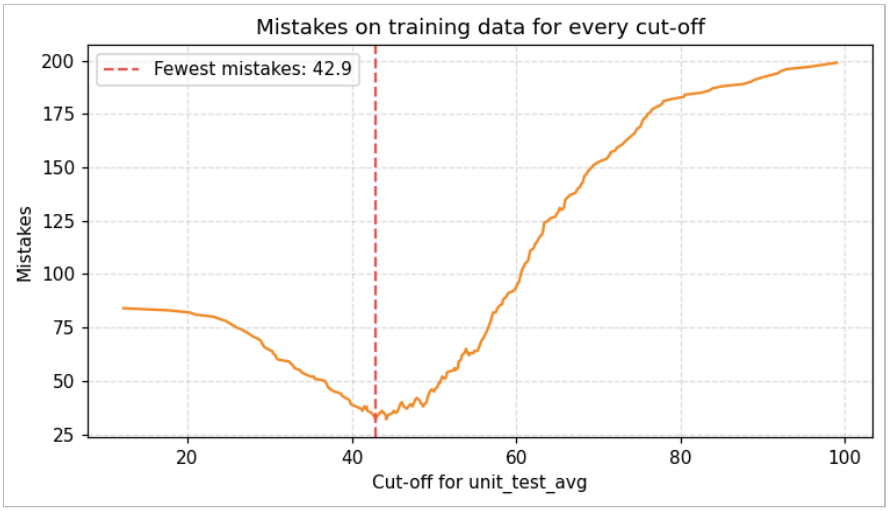


Figure 3 — How the computer chose its cut-off, drawn by the program as a line chart and saved as chart3_choosing_cutoff.png.

How to read it. Across: every possible cut-off for the feature the computer picked. Up: how many training students that rule would have got wrong.

The line falls to a lowest point and rises again. Too low a cut-off misses struggling students; too high flags students who were fine. The computer simply picked the bottom of the curve — flag if unit_test_avg < 42.9.

Figure 4: Confusion matrix of the learned rule

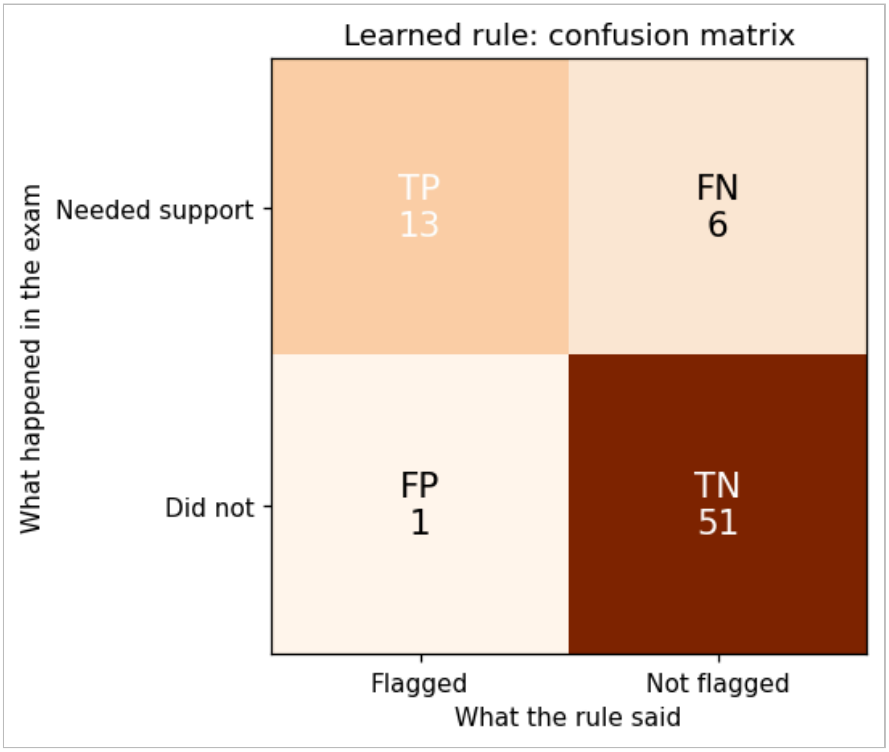


Figure 4 — Confusion matrix of the learned rule, drawn by the program as a confusion matrix and saved as chart4_matrix.png.

How to read it. TP: flagged and did need support. FN: needed support but was not flagged. FP: flagged but did not need support. TN: not flagged, did not need support.

On the testing students the learned rule's numbers are 13 6 1 51 (TP, FN, FP, TN). It flagged only one student who did not need help, and caught most of those who did.

Figure 5: The teacher's rule and the learned rule

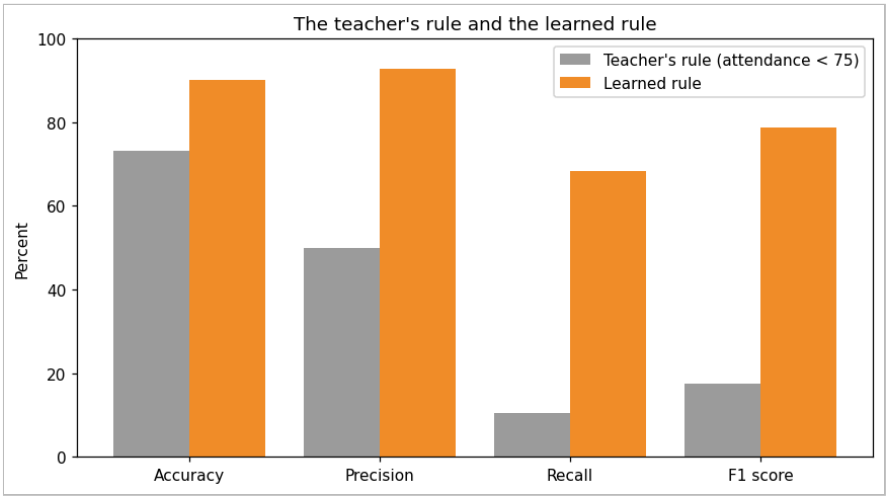


Figure 5 — The teacher's rule and the learned rule, drawn by the program as a bar chart and saved as chart5_rules_compared.png.

How to read it. Four pairs of bars — accuracy, precision, recall and F1 — for each rule.

The attendance rule has an accuracy of 73.2 %, which sounds fine. Its recall is 10.5 %: it would have missed almost every student who needed help. Its accuracy looks good only because most students do not need support, and a rule that flags almost nobody is right about them.

The learned rule's recall is 68.4 % and its accuracy 90.1 %. It is better on every measure.

DEPLOYMENT

After the first unit tests, the program checks three students and suggests whether each should be offered extra classes. It ends with a reminder printed in the output itself: the model only suggests.

In real use, the coordinator would get a short list of suggested students. Each class teacher would look at the list, add or remove names from what they know, talk to each student, and offer support privately — never as a public list.

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).

A label can hurt a child

Being marked as "needs support" can embarrass a student or lower a teacher's expectations of them. The list must stay private, be used only to offer help, and never follow the student around. It is a suggestion for this term, not a judgement about the student.

Accountability

If a student is missed, a person must answer for it — not the program. That is why the class teacher makes every final decision and can add students the rule did not flag.

Bias

If the data comes from one school, the rule fits that school. A cut-off learned from one school's unit tests means little in another school with harder or easier tests. It must be retrained, and checked, wherever it is used.

Privacy

Marks and attendance are personal. Real records must stay with the school, use roll numbers only, and be used only for the purpose they were collected for.

Transparency

This model is one sentence long: flag if unit-test average is below 42.9. A teacher, a parent or the student can read it and question it. For decisions about children, a model that can explain itself is far better than one that cannot.

THE NO-CODE VERSION

Orange's Tree widget builds decision trees. Limited to one level, it finds exactly the kind of single rule this program finds. The tool used is **Orange Data Mining**.

STEP	WIDGET	WHAT IT DOES
1	**File**	Load student_records.csv. Set needs_support as the target and roll_no as a meta attribute.
2	**Select Rows**	Keep rows where attendance_pct is at most 100 and unit_test_avg is defined.
3	**Scatter Plot**	Plot attendance_pct against unit_test_avg, coloured by needs_support — chart 1.
4	**Data Sampler**	Keep 80% for training; the rest for testing.
5	**Tree**	Set the maximal tree depth to 1. This gives a single rule.
6	**Tree Viewer**	See the question the tree chose. It should be about unit_test_avg.
7	**Test and Score**	Test on the testing data with target class Yes, and read the recall.
8	**Confusion Matrix**	Count the students who were missed (the false negatives).

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 (student_support.ipynb), or any Python editor (student_support.py)
No-code tool	Orange Data Mining
Model type	Learning based → Supervised learning → Classification
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 student_support.py**, or open **student_support.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
`dropna() / data[... <= 100]`	Stage 3	Removes blank marks and impossible attendance
`groupby(label).mean()`	Stage 3	Average of each feature in the two groups
`plt.scatter / plt.hist`	Stage 3	The scatter plot and the two histograms
`sorted(train[f].unique())`	Stage 4	Every value a feature takes — each is tried as a cut-off
`(flagged != truth).sum()`	Stage 4	Counts the mistakes one rule makes
`nested for loops`	Stage 4	Try every feature, and every cut-off for it
`TP, FN, FP, TN counts`	Stage 5	The confusion matrix, counted with a loop
`plt.plot / plt.imshow / plt.bar`	Stages 4, 5	The cut-off curve, the confusion matrix and the comparison

SOURCE CODE

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

`student_support.py`

```
# ------------------------------------------------------------------------
# student_support.py
#
# AI project on Statistical Data: spotting, early in the year, the students
# who may need extra help to pass - so that help reaches them in time.
#
# CBSE Class X, Artificial Intelligence (417), Part C project work.
# SDG 4 (Quality Education) and SDG 10 (Reduced Inequalities).
#
# Two models are compared:
#   - the TEACHER'S RULE, written by a person: flag a student whose
#     attendance is below 75%
#   - a LEARNED RULE: the computer tries every feature and every cut-off
#     on the training data, and keeps the one rule that makes the fewest
#     mistakes. (This is a decision tree with a single question, which is
#     called a decision stump.)
#
# Libraries: pandas (the data), numpy (the arithmetic), matplotlib (charts).
# ------------------------------------------------------------------------

# %% Stage 1 - Problem scoping
print("=" * 60)
print("WHO NEEDS A HELPING HAND?")
print("=" * 60)
print("Who   : class teachers, and students who are falling behind")
print("What  : struggling students are often noticed only after the exam")
print("Where : a school, in the first months of the year")
print("Why   : help given early can change the result")
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("student_records.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 = ["attendance_pct", "homework_pct", "unit_test_avg", "study_hours_week"]
label = "needs_support"

missing = data["unit_test_avg"].isnull().sum()
data = data.dropna(subset=["unit_test_avg"])
print("Rows with no unit test average   :", missing, "(removed)")

# Attendance is a percentage, so anything above 100 is a typing mistake.
wrong = (data["attendance_pct"] > 100).sum()
data = data[data["attendance_pct"] <= 100]
print("Rows with attendance above 100%  :", wrong, "(removed)")
print("Rows left                        :", len(data))
print()

print("Students who needed support and who did not:")
print(data[label].value_counts())
print()
print("Average of each feature in the two groups:")
print(data.groupby(label)[features].mean().round(1))
print()

yes = data[data[label] == "Yes"]
no = data[data[label] == "No"]

# Chart 1: attendance against unit test average.
plt.figure(figsize=(7, 5))
plt.scatter(no["attendance_pct"], no["unit_test_avg"], color="#4e79a7", alpha=0.55, s=18, label="Passed
comfortably")
plt.scatter(yes["attendance_pct"], yes["unit_test_avg"], color="#e15759", alpha=0.75, s=18, label="Needed
support")
plt.axvline(75, color="#999999", linestyle="--")
plt.title("Attendance and unit test average")
plt.xlabel("Attendance (%)")
plt.ylabel("Unit test average (%)")
plt.legend()
plt.grid(True, linestyle="--", alpha=0.4)
plt.tight_layout()
plt.savefig("chart1_attendance_tests.png")
plt.close()

# Chart 2: unit test average in the two groups.
plt.figure(figsize=(7, 4.5))
plt.hist(no["unit_test_avg"], bins=18, alpha=0.6, color="#4e79a7", label="Passed comfortably")
plt.hist(yes["unit_test_avg"], bins=18, alpha=0.6, color="#e15759", label="Needed support")
plt.title("Unit test average")
plt.xlabel("Unit test average (%)")
plt.ylabel("Number of students")
plt.legend()
plt.tight_layout()
plt.savefig("chart2_unit_tests.png")
plt.close()
```

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

data = data.sample(frac=1, random_state=2)
cut = int(len(data) * 0.8)
train = data[:cut]
test = data[cut:]
print("Training rows :", len(train))
print("Testing rows  :", len(test))
print()

# The learned rule: "flag the student if <feature> is below <cut-off>".
# Try every feature, and every value it takes in the training data as the
# cut-off. Count the mistakes each rule makes on the training students, and
# keep the rule with the fewest.
truth = train[label] == "Yes"
best_mistakes = len(train) + 1
rules_tried = 0
for f in features:
    for cutoff in sorted(train[f].unique()):
        rules_tried = rules_tried + 1
        flagged = train[f] < cutoff
        mistakes = (flagged != truth).sum()
        if mistakes < best_mistakes:
            best_mistakes = mistakes
            best_feature = f
            best_cutoff = cutoff

print("Rules tried                :", rules_tried)
print("Learned rule               : flag if", best_feature, "<", best_cutoff)
print("Mistakes on training data  :", best_mistakes, "of", len(train))
print("Teacher's rule             : flag if attendance_pct < 75")
print()

# For the chart: mistakes made by every cut-off of the chosen feature.
cutoffs = sorted(train[best_feature].unique())
cutoff_mistakes = []
for cutoff in cutoffs:
    cutoff_mistakes.append(((train[best_feature] < cutoff) != truth).sum())

# Chart 3: how the computer chose its cut-off.
plt.figure(figsize=(7, 4))
plt.plot(cutoffs, cutoff_mistakes, color="#f28e2b")
plt.axvline(best_cutoff, color="#e15759", linestyle="--", label="Fewest mistakes: " + str(best_cutoff))
plt.title("Mistakes on training data for every cut-off")
plt.xlabel("Cut-off for " + best_feature)
plt.ylabel("Mistakes")
plt.legend()
plt.grid(True, linestyle="--", alpha=0.4)
plt.tight_layout()
plt.savefig("chart3_choosing_cutoff.png")
plt.close()

teacher = []
learned = []
for i in range(len(test)):
    row = test.iloc[i]
    if row["attendance_pct"] < 75:
        teacher.append("Yes")
    else:
        teacher.append("No")
    if row[best_feature] < best_cutoff:
        learned.append("Yes")
    else:
```

```
        learned.append("No")

# %% Stage 5 - Evaluation
print("--- Stage 5: Evaluation ---")
actual = list(test[label])

results = []
for name, predicted in [["Teacher's rule", teacher], ["Learned rule", learned]]:
    tp = fn = fp = tn = 0
    for i in range(len(actual)):
        if actual[i] == "Yes" and predicted[i] == "Yes":
            tp = tp + 1
        elif actual[i] == "Yes":
            fn = fn + 1
        elif predicted[i] == "Yes":
            fp = fp + 1
        else:
            tn = tn + 1
    accuracy = (tp + tn) / len(actual) * 100
    precision = 0
    if tp + fp > 0:
        precision = tp / (tp + fp) * 100
    recall = 0
    if tp + fn > 0:
        recall = tp / (tp + fn) * 100
    f1 = 0
    if precision + recall > 0:
        f1 = 2 * precision * recall / (precision + recall)
    results.append([name, tp, fn, fp, tn, accuracy, precision, recall, f1])
    print(name)
    print("  " + name + " TP, FN, FP, TN :", tp, fn, fp, tn)
    print("  " + name + " accuracy  :", round(accuracy, 1), "%")
    print("  " + name + " precision :", round(precision, 1), "%")
    print("  " + name + " recall    :", round(recall, 1), "%")
    print("  " + name + " F1 score  :", round(f1, 1), "%")
    print()

# Chart 4: confusion matrix of the learned rule.
r = results[1]
grid = [[r[1], r[2]], [r[3], r[4]]]
plt.figure(figsize=(5, 4.3))
plt.imshow(grid, cmap="Oranges")
names = [["TP", "FN"], ["FP", "TN"]]
for row in range(2):
    for col in range(2):
        if grid[row][col] > max(max(grid)) / 2:
            ink = "white"
        else:
            ink = "black"
        plt.text(col, row, names[row][col] + "\n" + str(grid[row][col]),
                 ha="center", va="center", fontsize=14, color=ink)
plt.xticks([0, 1], ["Flagged", "Not flagged"])
plt.yticks([0, 1], ["Needed support", "Did not"])
plt.xlabel("What the rule said")
plt.ylabel("What happened in the exam")
plt.title("Learned rule: confusion matrix")
plt.tight_layout()
plt.savefig("chart4_matrix.png")
plt.close()

# Chart 5: the two rules side by side.
x = np.arange(4)
plt.figure(figsize=(8, 4.5))
plt.bar(x - 0.2, results[0][5:9], width=0.4, label="Teacher's rule (attendance < 75)", color="#9c9c9c")
plt.bar(x + 0.2, results[1][5:9], width=0.4, label="Learned rule", color="#f28e2b")
```

```
plt.xticks(x, ["Accuracy", "Precision", "Recall", "F1 score"])
plt.ylim(0, 100)
plt.ylabel("Percent")
plt.title("The teacher's rule and the learned rule")
plt.legend()
plt.tight_layout()
plt.savefig("chart5_rules_compared.png")
plt.close()

# %% Stage 6 - Deployment
print("--- Stage 6: Deployment ---")
print("Three students after the first unit tests:")
students = pd.DataFrame({
    "roll_no":          ["9401", "9402", "9403"],
    "attendance_pct":   [92, 70, 88],
    "homework_pct":     [55, 85, 90],
    "unit_test_avg":    [36.5, 64.0, 71.5],
    "study_hours_week": [3, 7, 8],
})
for i in range(len(students)):
    s = students.iloc[i]
    if s[best_feature] < best_cutoff:
        advice = "suggest to the class teacher for extra classes"
    else:
        advice = "no flag"
    print("Roll", s["roll_no"], ": attendance", s["attendance_pct"], "%, unit tests",
          s["unit_test_avg"], "% ->", advice)
print("(The model only suggests. The class teacher decides, and talks to the student first.)")
```

SAMPLE OUTPUT

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

Output — python student_support.py

```
============================================================
WHO NEEDS A HELPING HAND?
============================================================
Who   : class teachers, and students who are falling behind
What  : struggling students are often noticed only after the exam
Where : a school, in the first months of the year
Why   : help given early can change the result

--- Stage 2: Data acquisition ---
Rows read    : 360
Columns read : 6

The first 10 rows:
   roll_no  attendance_pct  homework_pct  unit_test_avg  study_hours_week needs_support
0     9001             85            42           49.1                 2            No
1     9002             87            53           45.0                 1            No
2     9003             83            44           46.1                 4           Yes
3     9004             78            75           60.4                 6            No
4     9005             67            56           57.9                 5            No
5     9006             90           100           58.7                 6            No
6     9007             88            73           72.1                 9            No
7     9008             96            58           72.2                 9            No
8     9009             76            91           62.5                 7            No
9     9010             67            61           47.1                 5           Yes

--- Stage 3: Data exploration ---
Rows with no unit test average   : 3 (removed)
Rows with attendance above 100%  : 2 (removed)
Rows left                        : 355

Students who needed support and who did not:
```

```
needs_support
No     252
Yes    103
Name: count, dtype: int64

Average of each feature in the two groups:
               attendance_pct  homework_pct  unit_test_avg  study_hours_week
needs_support
No                       87.4          75.1           62.3               6.6
Yes                      79.7          55.0           38.6               3.4

--- Stage 4: Modelling ---
Training rows : 284
Testing rows  : 71

Rules tried                : 349
Learned rule               : flag if unit_test_avg < 42.9
Mistakes on training data  : 32 of 284
Teacher's rule             : flag if attendance_pct < 75

--- Stage 5: Evaluation ---
Teacher's rule
  Teacher's rule TP, FN, FP, TN : 2 17 2 50
  Teacher's rule accuracy  : 73.2 %
  Teacher's rule precision : 50.0 %
  Teacher's rule recall    : 10.5 %
  Teacher's rule F1 score  : 17.4 %

Learned rule
  Learned rule TP, FN, FP, TN : 13 6 1 51
  Learned rule accuracy  : 90.1 %
  Learned rule precision : 92.9 %
  Learned rule recall    : 68.4 %
  Learned rule F1 score  : 78.8 %

--- Stage 6: Deployment ---
Three students after the first unit tests:
Roll 9401 : attendance 92 %, unit tests 36.5 % -> suggest to the class teacher for extra classes
Roll 9402 : attendance 70 %, unit tests 64.0 % -> no flag
Roll 9403 : attendance 88 %, unit tests 71.5 % -> no flag
(The model only suggests. The class teacher decides, and talks to the student first.)
```

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 : 360`	`Rows read : 360`	Pass
Blank unit test averages are removed	`Rows with no unit test average : 3 (removed)`	`Rows with no unit test average : 3 (removed)`	Pass
Attendance of 910% and 108% is removed	`Rows with attendance above 100% : 2 (removed)`	`Rows with attendance above 100% : 2 (removed)`	Pass
The computer picks the unit tests, not attendance	`Learned rule : flag if unit_test_avg < 42.9`	`Learned rule : flag if unit_test_avg < 42.9`	Pass
The attendance rule misses most students who need help	`Teacher's rule recall : 10.5 %`	`Teacher's rule recall : 10.5 %`	Pass
Struggling students with perfect attendance: the learned rule still finds them	`Learned rule recall : 100.0 %`	`Learned rule recall : 100.0 %`	Pass

...while the attendance rule flags nobody	`Teacher's rule recall : 0.0 %`	`Teacher's rule recall : 0.0 %`	Pass
Boundary: 100% attendance is kept, 101% is removed	`Rows with attendance above 100% : 1 (removed)`	`Rows with attendance above 100% : 1 (removed)`	Pass
The data file is missing	`FileNotFoundError: [Errno 2] No such file or directory: 'student_records.csv'`	`FileNotFoundError: [Errno 2] No such file or directory: 'student_records.csv'`	Pass

RESULTS AND FUTURE SCOPE

What the project found

1. The computer's best rule is about the unit tests, not attendance: flag if unit_test_avg < 42.9.
2. The common attendance rule had an accuracy of 73.2 % but a recall of only 10.5 % — it misses almost every student who needs help.
3. Most students who needed support in this data came to school regularly.
4. The learned rule caught 68.4 % of them, with a precision of 92.9 %.
5. A high accuracy can hide a useless rule when most of the data is in one group.

Limitations

- Made-up data; the cut-off is an example, not advice for any real school
- A single rule cannot use two clues together, such as low marks and little homework
- It still misses some students who needed support
- Self-study hours come from a survey, which students may not answer honestly

Future scope

- Let the computer find a second question for the students the first rule does not flag (a decision tree with two levels)
- Add a "just above the cut-off" list for teachers to keep an eye on
- Repeat after the half-yearly exam and see how the rule changes
- Build the same project on anonymous survey data from your class, with permission

What had to be learnt along the way

- How a computer can learn a rule by trying every cut-off and counting mistakes
- Decision trees and decision stumps
- Why accuracy can mislead when one group is much larger than the other
- The ethics of using AI to make decisions about people, and children especially

CONCLUSION

The project found that the unit-test average is a far better early warning than attendance. The rule the computer learned caught 68.4 % of the students who needed support; the attendance rule caught 10.5 %.

The model is simple on purpose. A single readable rule is easy for teachers, parents and students to understand and to challenge, which matters more than a few extra percent of accuracy when the decisions are about children.

The biggest lesson is about evidence. Everybody assumed attendance was the warning sign. Testing that belief against data showed it was wrong — and that is worth more than the model itself.

REFERENCES

1. UNESCO, SDG 4 — Education 2030 — https://www.unesco.org/sdg4education2030/
2. Ministry of Education, Government of India, National Education Policy 2020 — on early identification and support for students who fall behind.
3. Orange Data Mining, Tree widget — https://orangedatamining.com/widget-catalog/model/tree/
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