(name of your school)

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

on

“WHO SHOULD GET TESTED?”

Statistical Data · SDG 3: Good Health and Well-being

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 Should Get Tested?"** 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	9
7	Deployment	12
8	Ethical Concerns	12
9	The No-Code Version	13
10	Technical Details	13
11	Source Code	14
12	Sample Output	19
13	Testing	20
14	Results and Future Scope	21
15	Conclusion	21
16	References	22

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

INTRODUCTION

Many people with diabetes do not know they have it. A free health camp can find them — but a camp may see hundreds of people and have test kits for only some of them.

Every person can be weighed and asked a few questions: age, family history, how much they exercise. A quick finger-prick gives a glucose reading. The question is who should be sent for the full test.

This project builds a model that answers that question. Its real lesson is about mistakes: in health, the two kinds of mistake are not equal, and the model has to be judged with that in mind.

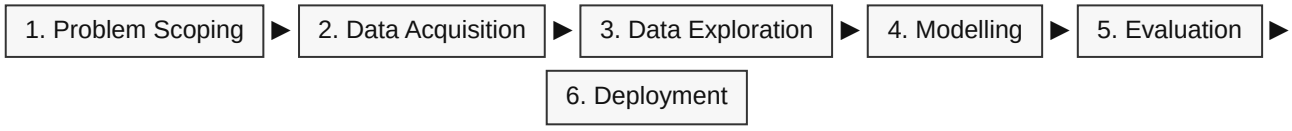
Who would use it: Doctors and health workers running a free health camp, who must decide who gets one of a limited number of blood tests.

Link to the Sustainable Development Goals

SDG 3 — Good Health and Well-being. SDG 3 includes reducing early deaths from diseases like diabetes. Diabetes found early can be controlled with diet, exercise and medicine; diabetes found late damages the eyes, kidneys and heart. A screening model helps a camp with few test kits find the people most likely to have it.

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 • Doctors and health workers at a community health camp • People at the camp who may have diabetes without knowing it	**What?** THE PROBLEM • Not enough test kits to test everyone fully • People with diabetes who are sent home go untreated
Where? CONTEXT • Free health camps in towns and villages • On the day of the camp, with a long queue	**Why?** BENEFIT • Diabetes caught early can be controlled • Test kits go to the people who need them most

Problem statement

> Our **health workers at a free health camp** have a problem of **deciding who should get one of a few full diabetes tests** when **hundreds of people come and test kits are limited**. An ideal solution would **pick out**

> **the people most likely to have diabetes, missing as few of them as possible**.

How it is done today

Testing everyone. The best answer, when there are enough kits, money and time. At a large free camp there usually are not.

Testing by age only. A simple rule such as "test everyone over 45". Easy, but it misses younger people with a high risk and tests many older people with none.

The doctor's judgement. Very good, but a doctor at a busy camp has a minute or two per person. A model can sort the queue so the doctor's time goes where it matters.

Objectives

1. To read the health check-up records of 600 people from a CSV file
2. To find and remove readings that are impossible, such as a glucose of 0
3. To explore which measurements differ between people with and without diabetes
4. To build a k nearest neighbours model that counts how many similar people have diabetes
5. To compare a majority rule with a cautious rule, using precision, recall and the F1 score
6. To decide which rule a health camp should use, and explain why

DATA ACQUISITION

Each row is one adult at the health camp: age, BMI (body mass index), a fasting glucose reading, blood pressure, whether a parent has diabetes, and minutes of exercise in a week. The last column says whether a full test later showed diabetes — that is the label.

The file has 602 rows. No names are stored; each person has an ID such as P5001.

Health data is private, which is why this project uses a made-up sample. A student should never collect real medical records. A real version would be built by a hospital, with the patients' permission, under a doctor's supervision. For practice, public datasets such as the Pima Indians Diabetes dataset are often used — credit it if you do.

> **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
`person_id`	text	An ID number, never a name	not used
`age`	integer	Age in years	feature
`bmi`	decimal	Body mass index: weight compared with height	feature
`glucose`	integer	Fasting blood sugar, mg/dL	feature
`blood_pressure`	integer	Diastolic blood pressure, mmHg	feature
`family_history`	0 or 1	1 if a parent has diabetes	feature

activity_min	integer	Minutes of exercise in a week	feature
diabetic	Yes / No	Whether the full test showed diabetes — what the model must predict	label

The file health_checkup.csv

Six hundred check-up records. Some glucose and blood pressure readings are recorded as 0, which is impossible in a living person. The whole file is **602 rows, 17.5 KB**.

The first eight rows of health_checkup.csv

PERSON_ID	AGE	BMI	GLUCOSE	BLOOD_PRESSURE	FAMILY_HISTORY	ACTIVITY_MIN	DIABETIC
P5001	32	28.5	96	92	0	140	No
P5002	68	17.0	120	69	0	210	No
P5003	32	25.6	86	61	0	160	No
P5004	44	17.3	80	75	0	300	No
P5005	43	24.1	108	96	0	130	No
P5006	62	25.1	170	91	0	170	Yes
P5007	41	25.1	99	76	0	20	No
P5008	43	17.0	94	70	0	110	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
Glucose recorded as 0	`5 (removed)`	Nobody alive has a blood sugar of 0. The reading was simply not taken. Treating it as a real 0 would teach the model that a very low sugar means no diabetes, so these rows are removed.
Blood pressure recorded as 0	`4 (removed)`	The same mistake, the same fix.
The same person typed in twice	`2 rows`	drop_duplicates() removes exact repeats, so one person does not count as two neighbours.
Features on different scales	`glucose in hundreds, family history 0 or 1`	Every feature is rescaled to run from 0 to 1 before people are compared.

Figure 1: Fasting glucose, with and without diabetes

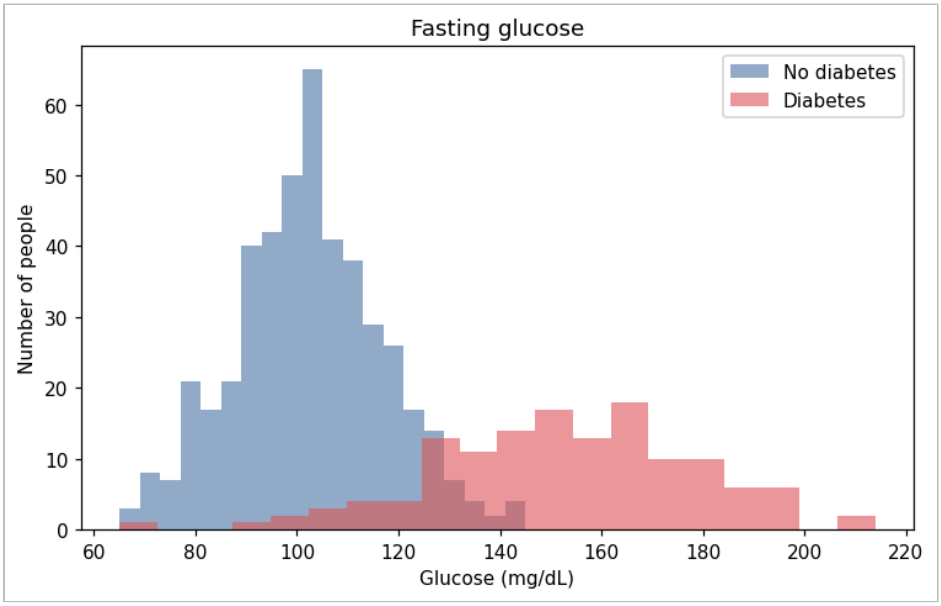


Figure 1 — Fasting glucose, with and without diabetes, drawn by the program as a histogram and saved as chart1_glucose.png.

How to read it. Two histograms on top of each other. Each bar counts how many people had a glucose reading in that range — blue for people without diabetes, red for people with it.

Glucose is the strongest clue. Most people without diabetes sit around 100; people with diabetes are spread out higher, around 150. But the two shapes overlap between about 110 and 130. In that overlap, glucose alone cannot decide — which is why the model uses five more features.

Figure 2: BMI and fasting glucose

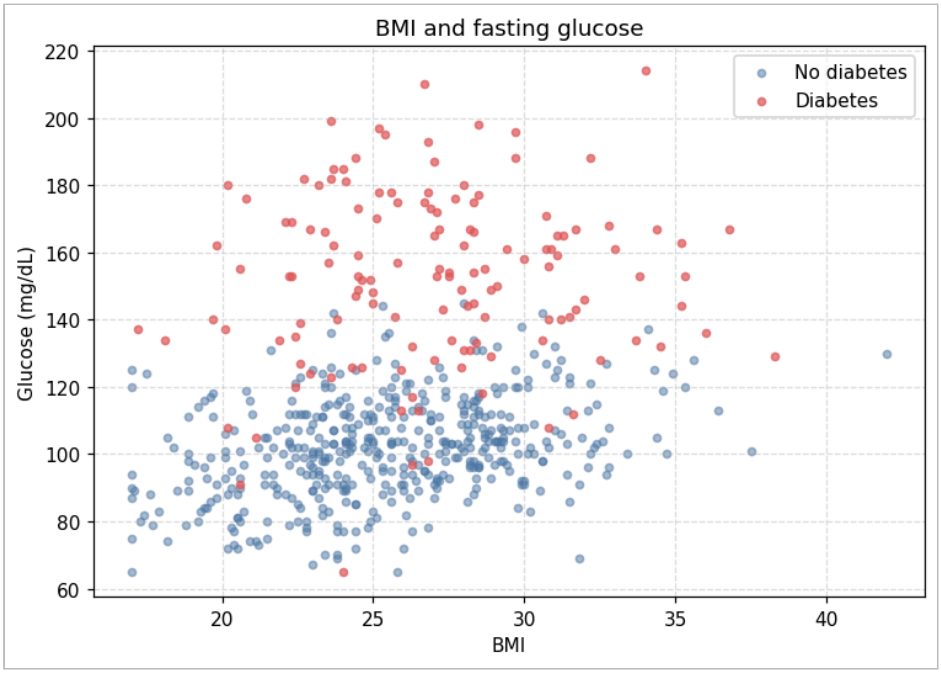


Figure 2 — BMI and fasting glucose, drawn by the program as a scatter plot and saved as chart2_bmi_glucose.png.

How to read it. Each dot is one person: BMI across, glucose up. Red dots have diabetes.

Most red dots are high up, and there are more of them on the right, where BMI is higher. But some red dots sit down among the blue ones, with normal glucose. Those people will be hard for any model to catch, and they are the reason the next stage matters.

MODELLING

AI model ▶ Learning based ▶ Supervised learning ▶ Classification

The answer is Yes or No — diabetes or not — so this is classification. The training records come with the answer from a full test, so it is supervised learning.

How the model works

The data is split into 472 training records and 119 testing records.

The model is k nearest neighbours with k = 7. For each person it finds the 7 most similar people in the training data and counts how many of them have diabetes. That count, from 0 to 7, is how worried the model is.

The count is then turned into a decision in two ways. The majority model refers the person if 4 or more of the 7 have diabetes. The cautious model refers the person if 2 or more do. Both use the same neighbours; only the bar is different.

EVALUATION

Both models were tested on the same 119 people. The majority model had an accuracy of 91.6 % and the cautious model 86.6 %. Judged by accuracy alone, the majority model wins.

But think about the two kinds of mistake. A false positive sends a healthy person for one blood test: a little time and money. A false negative sends a person with diabetes home believing they are fine: years of damage before anyone notices. These mistakes are not equal.

Recall measures exactly the mistake that hurts — what share of people with diabetes were caught. The cautious model's recall is 81.5 % against 77.8 %. Its precision is lower (66.7 % against 84.0 %), meaning more healthy people get tested. For a screening camp that is the right trade, so the cautious model is the one to use. The F1 score, which balances precision and recall, would have picked the other model — a good example of why the metric must be chosen to fit the problem.

Even the cautious model misses some people. Chart 2 shows why: a few people with diabetes have ordinary readings. A screening model is a first sieve, not a diagnosis.

Figure 3: Confusion matrix: majority model

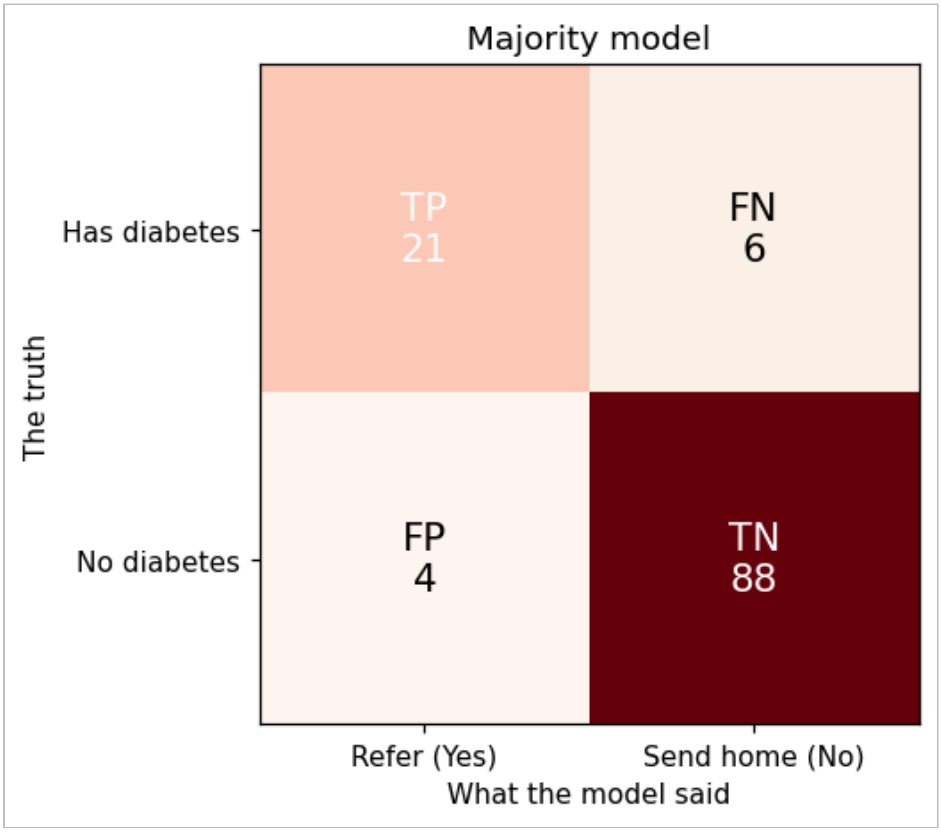


Figure 3 — Confusion matrix: majority model, drawn by the program as a confusion matrix and saved as chart3_matrix_majority.png.

How to read it. Top left (TP): has diabetes, and was referred. Top right (FN): has diabetes, but was sent home. Bottom left (FP): no diabetes, but referred. Bottom right (TN): no diabetes, sent home.

The majority model refers a person only when at least 4 of their 7 nearest neighbours have diabetes. Its numbers are 21 6 4 88 (TP, FN, FP, TN). It wastes very few tests — but it misses 6 people with diabetes, who are sent home untested.

Figure 4: Confusion matrix: cautious model

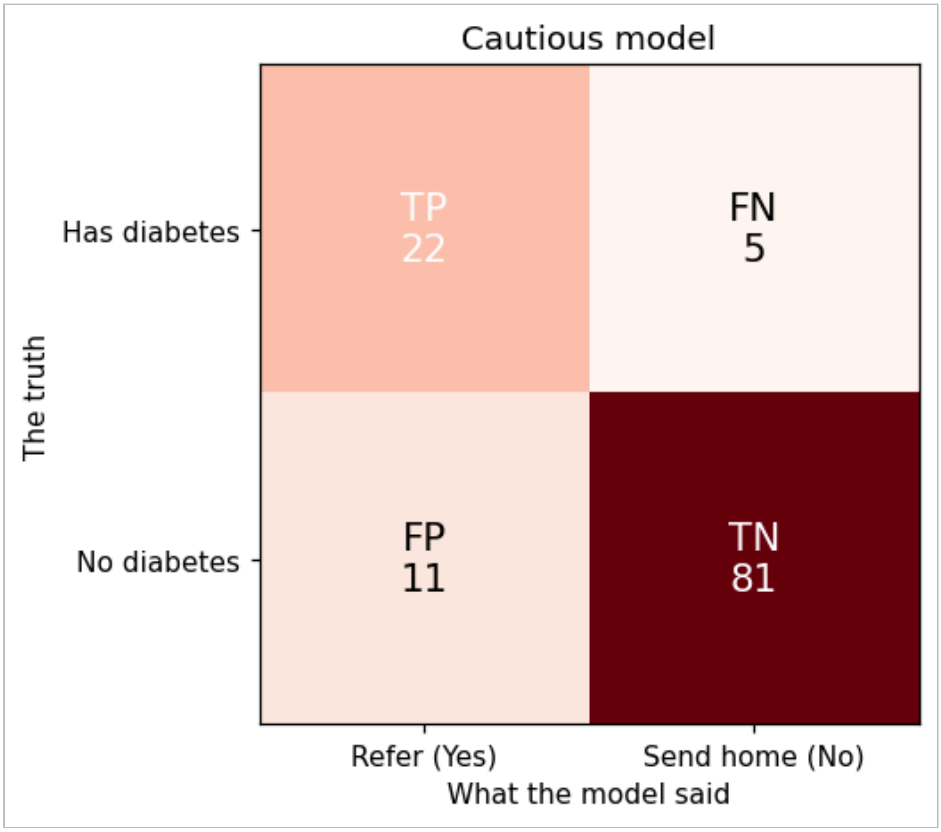


Figure 4 — Confusion matrix: cautious model, drawn by the program as a confusion matrix and saved as chart4_matrix_cautious.png.

How to read it. The same four boxes, for the cautious model, which refers a person when 2 or more of their 7 neighbours have diabetes.

Its numbers are 22 5 11 81. One more person with diabetes is caught. The price is more false alarms: more healthy people sent for a test they did not need.

Figure 5: The two models, metric by metric

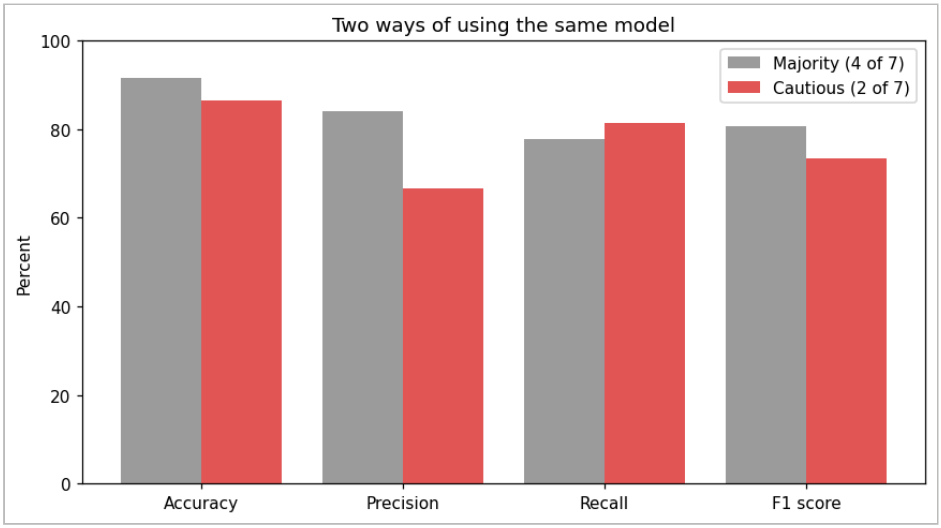


Figure 5 — The two models, metric by metric, drawn by the program as a bar chart and saved as chart5_metrics.png.

How to read it. Four pairs of bars: accuracy, precision, recall and F1 score, for each model.

By accuracy, precision and F1, the majority model looks better. Only recall — the share of people with diabetes who are caught — is higher for the cautious model. If accuracy were the only number in this report, it would pick the model that

sends more sick people home.

Figure 6: The trade-off between recall and precision

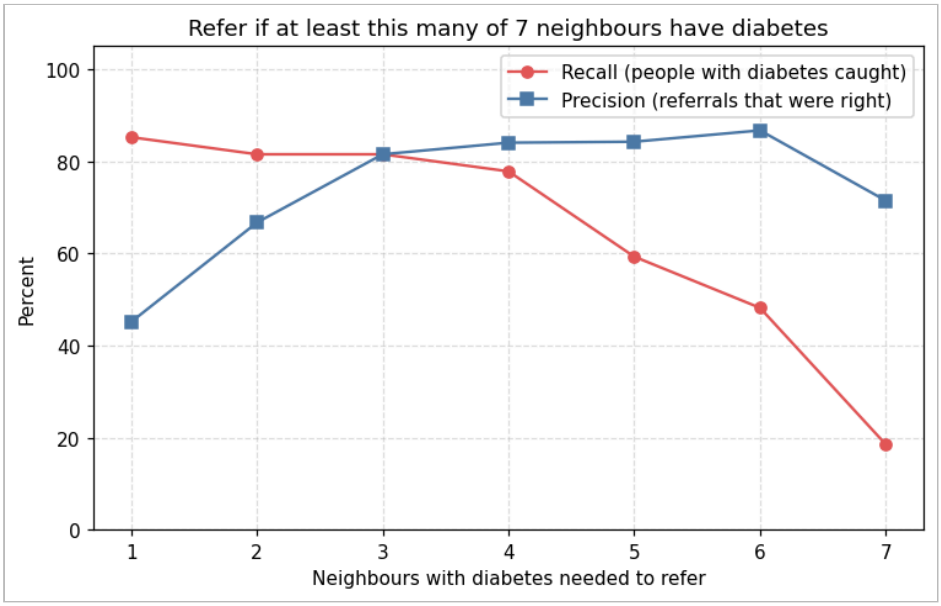


Figure 6 — The trade-off between recall and precision, drawn by the program as a line chart and saved as chart6_tradeoff.png.

How to read it. Across: how many of the 7 neighbours must have diabetes before the person is referred. Red: recall. Blue: precision.

The two lines move in opposite directions. Make the bar lower and more people with diabetes are caught (recall rises), but more referrals turn out to be healthy (precision falls). Make it higher and the opposite happens. There is no bar that gives both — the camp has to choose, and the choice is about people, not numbers.

DEPLOYMENT

The program ends by screening three new people with the cautious model. For each it prints how many of the 7 neighbours have diabetes and what to do: refer for a blood test, or come back next year.

In a real camp this would sort the queue. The health worker enters the readings, the model suggests who should be tested, and the doctor makes the final decision.

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).

Non-maleficence: do no harm

This is the bioethics principle the whole evaluation turns on. A model that sends sick people home does harm, however accurate it looks. That is why recall, not accuracy, decided which model to use.

Autonomy and privacy

Health data belongs to the patient. Records must be collected with consent and stored with an ID number, never a name. A student project must use public or made-up data, never real patients' records.

Accountability

The model suggests; a doctor decides. If a person is wrongly sent home, a person — not the program — is responsible, so the model must never be the last word.

Justice and bias

If the training data came mostly from one city or one age group, the model may work worse for others. A real screening model must be checked on every group it will be used on.

THE NO-CODE VERSION

The same project in Orange uses the kNN widget, and Orange's Confusion Matrix and Test and Score widgets show precision and recall directly. The tool used is **Orange Data Mining**.

STEP	WIDGET	WHAT IT DOES
1	**File**	Load health_checkup.csv. Set diabetic as the target and person_id as a meta attribute.
2	**Select Rows**	Keep only rows where glucose is greater than 0 and blood_pressure is greater than 0.
3	**Distributions**	Show glucose split by diabetic to see the overlap in chart 1.
4	**Data Sampler**	Keep 80% for training; the rest is for testing.
5	**kNN**	Set the number of neighbours to 7.
6	**Test and Score**	Test on the testing data. Set the target class to Yes so that precision and recall are about people with diabetes.
7	**Confusion Matrix**	Read TP, FN, FP and TN. Count the false negatives — the people sent home.

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 (diabetes_screening.ipynb), or any Python editor (diabetes_screening.py)
No-code tool	Orange Data Mining
Model type	Learning based → Supervised learning → Classification
Charts drawn	6

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 diabetes_screening.py**, or open **diabetes_screening.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["glucose"] > 0]`	Stage 3	Keeps only the rows with a real glucose reading
`drop_duplicates()`	Stage 3	Removes repeated rows
`groupby(label).mean()`	Stage 3	Average of each feature with and without diabetes
`plt.hist`	Stage 3	Draws the two glucose histograms
`nsmallest(7).index`	Stage 4	The 7 most similar people in the training data
`(... == "Yes").sum()`	Stage 4	Counts how many of the neighbours have diabetes
`TP, FN, FP, TN counts`	Stage 5	The four boxes of the confusion matrix, counted with a loop
`2 × P × R / (P + R)`	Stage 5	The F1 score from precision and recall
`plt.imshow / plt.plot / plt.bar`	Stage 5	Confusion matrices, the trade-off line and the metric bars

SOURCE CODE

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

`diabetes_screening.py`

```
# ------------------------------------------------------------------------
# diabetes_screening.py
#
# AI project on Statistical Data: a screening model that picks out the people
# at a health camp who should be sent for a proper blood test for diabetes.
#
# CBSE Class X, Artificial Intelligence (417), Part C project work.
# SDG 3 (Good Health and Well-being).
#
# The model is k nearest neighbours: it looks at the 7 people in the training
# data most like this person and counts how many of them have diabetes.
#
# Two ways of using those votes are compared:
#   - MAJORITY : refer the person if 4 or more of the 7 have diabetes
#   - CAUTIOUS : refer the person if 2 or more of the 7 have diabetes
# The point of the project is the difference between them. In screening, a
# person with diabetes who is sent home is a much worse mistake than a
# healthy person who is sent for one extra test.
#
# Libraries: pandas (the data), numpy (the arithmetic), matplotlib (charts).
# ------------------------------------------------------------------------

# %% Stage 1 - Problem scoping
print("=" * 60)
print("DIABETES SCREENING AT A HEALTH CAMP")
print("=" * 60)
print("Who   : doctors and health workers at a free health camp")
print("What  : too many people to blood-test every one of them")
print("Where : community health camps with limited test kits")
print("Why   : diabetes found early can be controlled")
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("health_checkup.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 = ["age", "bmi", "glucose", "blood_pressure", "family_history", "activity_min"]
label = "diabetic"

before = len(data)
data = data.drop_duplicates()
print("Repeated rows removed            :", before - len(data))

# Nobody alive has a glucose or blood pressure of 0. A 0 here means "not
# measured", so those rows cannot be used.
zero_glucose = (data["glucose"] == 0).sum()
zero_bp = (data["blood_pressure"] == 0).sum()
data = data[(data["glucose"] > 0) & (data["blood_pressure"] > 0)]
print("Rows with glucose recorded as 0  :", zero_glucose, "(removed)")
print("Rows with BP recorded as 0       :", zero_bp, "(removed)")
print("Rows left                        :", len(data))
print()

print("People with and without diabetes:")
print(data[label].value_counts())
print()

print("Average of each feature, with and without diabetes:")
print(data.groupby(label)[features].mean().round(1))
print()

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

# Chart 1: glucose, for people with and without diabetes.
plt.figure(figsize=(7, 4.5))
plt.hist(no["glucose"], bins=20, alpha=0.6, color="#4e79a7", label="No diabetes")
plt.hist(yes["glucose"], bins=20, alpha=0.6, color="#e15759", label="Diabetes")
plt.title("Fasting glucose")
plt.xlabel("Glucose (mg/dL)")
plt.ylabel("Number of people")
plt.legend()
plt.tight_layout()
plt.savefig("chart1_glucose.png")
plt.close()
```

```
# Chart 2: BMI against glucose.
plt.figure(figsize=(7, 5))
plt.scatter(no["bmi"], no["glucose"], color="#4e79a7", alpha=0.5, s=16, label="No diabetes")
plt.scatter(yes["bmi"], yes["glucose"], color="#e15759", alpha=0.7, s=16, label="Diabetes")
plt.title("BMI and fasting glucose")
plt.xlabel("BMI")
plt.ylabel("Glucose (mg/dL)")
plt.legend()
plt.grid(True, linestyle="--", alpha=0.4)
plt.tight_layout()
plt.savefig("chart2_bmi_glucose.png")
plt.close()

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

data = data.sample(frac=1, random_state=11)
cut = int(len(data) * 0.8)
train = data[:cut]
test = data[cut:]
print("Training rows :", len(train))
print("Testing rows  :", len(test))
print()

low = train[features].min()
high = train[features].max()
train_scaled = (train[features] - low) / (high - low)
test_scaled = (test[features] - low) / (high - low)

k = 7

# For every testing person: how many of their 7 nearest neighbours have diabetes?
votes = []
for i in range(len(test)):
    distance = ((train_scaled - test_scaled.iloc[i]) ** 2).sum(axis=1)
    nearest = distance.nsmallest(k).index
    votes.append((train.loc[nearest, label] == "Yes").sum())

majority = []
cautious = []
for v in votes:
    if v >= 4:
        majority.append("Yes")
    else:
        majority.append("No")
    if v >= 2:
        cautious.append("Yes")
    else:
        cautious.append("No")

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

# The four boxes of the confusion matrix, counted for each model.
# "Yes" (has diabetes) is the positive class.
results = []
for name, predicted in [["Majority", majority], ["Cautious", cautious]]:
    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" and predicted[i] == "No":
            fn = fn + 1
        elif actual[i] == "No" and predicted[i] == "Yes":
            fp = fp + 1
        else:
            tn = tn + 1
    accuracy = (tp + tn) / len(actual) * 100
    # Each formula is only worked out when it would not divide by zero.
    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, "model")
    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("  " + name + " missed    :", fn, "people with diabetes")
    print()

# Charts 3 and 4: the two confusion matrices.
chart_no = 3
for r in results:
    grid = [[r[1], r[2]], [r[3], r[4]]]
    plt.figure(figsize=(5, 4.3))
    plt.imshow(grid, cmap="Reds")
    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], ["Refer (Yes)", "Send home (No)"])
    plt.yticks([0, 1], ["Has diabetes", "No diabetes"])
    plt.xlabel("What the model said")
    plt.ylabel("The truth")
    plt.title(r[0] + " model")
    plt.tight_layout()
    plt.savefig("chart" + str(chart_no) + "_matrix_" + r[0].lower() + ".png")
    plt.close()
    chart_no = chart_no + 1

# Chart 5: the four metrics, both models side by side.
metric_names = ["Accuracy", "Precision", "Recall", "F1 score"]
x = np.arange(4)
plt.figure(figsize=(8, 4.5))
plt.bar(x - 0.2, results[0][5:9], width=0.4, label="Majority (4 of 7)", color="#9c9c9c")
plt.bar(x + 0.2, results[1][5:9], width=0.4, label="Cautious (2 of 7)", color="#e15759")
```

```
plt.xticks(x, metric_names)
plt.ylim(0, 100)
plt.ylabel("Percent")
plt.title("Two ways of using the same model")
plt.legend()
plt.tight_layout()
plt.savefig("chart5_metrics.png")
plt.close()

# The trade-off in full: try every possible bar, from "1 of 7 neighbours"
# to "all 7", and see what happens to precision and recall.
bars = [1, 2, 3, 4, 5, 6, 7]
bar_precision = []
bar_recall = []
for b in bars:
    tp = fp = fn = 0
    for i in range(len(actual)):
        if votes[i] >= b and actual[i] == "Yes":
            tp = tp + 1
        elif votes[i] >= b and actual[i] == "No":
            fp = fp + 1
        elif votes[i] < b and actual[i] == "Yes":
            fn = fn + 1
    if tp + fp > 0:
        bar_precision.append(tp / (tp + fp) * 100)
    else:
        bar_precision.append(0)
    if tp + fn > 0:
        bar_recall.append(tp / (tp + fn) * 100)
    else:
        bar_recall.append(0)

# Chart 6: precision and recall for every bar.
plt.figure(figsize=(7, 4.5))
plt.plot(bars, bar_recall, marker="o", color="#e15759", label="Recall (people with diabetes
caught)")
plt.plot(bars, bar_precision, marker="s", color="#4e79a7", label="Precision (referrals that were
right)")
plt.title("Refer if at least this many of 7 neighbours have diabetes")
plt.xlabel("Neighbours with diabetes needed to refer")
plt.ylabel("Percent")
plt.ylim(0, 105)
plt.xticks(bars)
plt.legend()
plt.grid(True, linestyle="--", alpha=0.4)
plt.tight_layout()
plt.savefig("chart6_tradeoff.png")
plt.close()

# %% Stage 6 - Deployment
print("--- Stage 6: Deployment ---")
print("Three people at the next camp (the cautious model is used):")

people = pd.DataFrame({
    "age":            [58, 34, 47],
    "bmi":            [31.0, 22.5, 27.8],
    "glucose":        [142, 92, 118],
    "blood_pressure": [88, 74, 82],
    "family_history": [1, 0, 1],
    "activity_min":   [30, 240, 90],
})
```

```
people_scaled = (people - low) / (high - low)

for i in range(len(people)):
    distance = ((train_scaled - people_scaled.iloc[i]) ** 2).sum(axis=1)
    nearest = distance.nsmallest(k).index
    v = (train.loc[nearest, label] == "Yes").sum()
    if v >= 2:
        advice = "REFER for a blood test"
    else:
        advice = "low risk - advise a check-up next year"
    p = people.iloc[i]
    print("Age", int(p["age"]), " BMI", p["bmi"], " glucose", int(p["glucose"]),
          "->", v, "of 7 neighbours have diabetes ->", advice)
```

SAMPLE OUTPUT

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

Output — python diabetes_screening.py

```
============================================================
DIABETES SCREENING AT A HEALTH CAMP
============================================================
Who   : doctors and health workers at a free health camp
What  : too many people to blood-test every one of them
Where : community health camps with limited test kits
Why   : diabetes found early can be controlled

--- Stage 2: Data acquisition ---
Rows read    : 602
Columns read : 8

The first 10 rows:
  person_id  age   bmi  glucose  blood_pressure  family_history  activity_min diabetic
0     P5001   32  28.5       96              92               0           140       No
1     P5002   68  17.0      120              69               0           210       No
2     P5003   32  25.6       86              61               0           160       No
3     P5004   44  17.3       80              75               0           300       No
4     P5005   43  24.1      108              96               0           130       No
5     P5006   62  25.1      170              91               0           170      Yes
6     P5007   41  25.1       99              76               0            20       No
7     P5008   43  17.0       94              70               0           110       No
8     P5009   33  35.2      144              74               1            70      Yes
9     P5010   53  28.3      116              67               0           230       No

--- Stage 3: Data exploration ---
Repeated rows removed            : 2
Rows with glucose recorded as 0  : 5 (removed)
Rows with BP recorded as 0       : 4 (removed)
Rows left                        : 591

People with and without diabetes:
diabetic
No     456
Yes    135
Name: count, dtype: int64

Average of each feature, with and without diabetes:
           age   bmi  glucose  blood_pressure  family_history  activity_min
diabetic
No        45.4  25.5    102.0            78.2             0.2         154.6
Yes       54.0  27.0    152.8            84.4             0.4         116.1

--- Stage 4: Modelling ---
```

```
Training rows : 472
Testing rows  : 119

--- Stage 5: Evaluation ---
Majority model
  Majority TP, FN, FP, TN : 21 6 4 88
  Majority accuracy  : 91.6 %
  Majority precision : 84.0 %
  Majority recall    : 77.8 %
  Majority F1 score  : 80.8 %
  Majority missed    : 6 people with diabetes

Cautious model
  Cautious TP, FN, FP, TN : 22 5 11 81
  Cautious accuracy  : 86.6 %
  Cautious precision : 66.7 %
  Cautious recall    : 81.5 %
  Cautious F1 score  : 73.3 %
  Cautious missed    : 5 people with diabetes

--- Stage 6: Deployment ---
Three people at the next camp (the cautious model is used):
Age 58  BMI 31.0  glucose 142 -> 6 of 7 neighbours have diabetes -> REFER for a blood test
Age 34  BMI 22.5  glucose 92 -> 0 of 7 neighbours have diabetes -> low risk - advise a check-up next year
Age 47  BMI 27.8  glucose 118 -> 1 of 7 neighbours have diabetes -> low risk - advise a check-up next
year
```

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 : 602`	`Rows read : 602`	Pass
Glucose of 0 means not measured: those rows are removed	`Rows with glucose recorded as 0 : 5 (removed)`	`Rows with glucose recorded as 0 : 5 (removed)`	Pass
Blood pressure of 0 is removed too	`Rows with BP recorded as 0 : 4 (removed)`	`Rows with BP recorded as 0 : 4 (removed)`	Pass
The cautious model misses fewer people with diabetes	`Cautious missed : 5 people with diabetes`	`Cautious missed : 5 people with diabetes`	Pass
...and the majority model misses more	`Majority missed : 6 people with diabetes`	`Majority missed : 6 people with diabetes`	Pass
Two groups that never overlap: nobody with diabetes is missed	`Majority recall : 100.0 %`	`Majority recall : 100.0 %`	Pass
Nobody has diabetes: precision cannot divide by zero, so it is reported as 0	`Majority precision : 0 %`	`Majority precision : 0 %`	Pass
Boundary: a single glucose of 0 is removed, not treated as very low sugar	`Rows with glucose recorded as 0 : 1 (removed)`	`Rows with glucose recorded as 0 : 1 (removed)`	Pass

The data file is missing	`FileNotFoundError: [Errno 2] No such file or directory: 'health_checkup.csv'`	`FileNotFoundError: [Errno 2] No such file or directory: 'health_checkup.csv'`	Pass

RESULTS AND FUTURE SCOPE

What the project found

1. Glucose is the strongest single clue, but it overlaps between about 110 and 130 mg/dL.
2. The majority model is more accurate (91.6 %) but misses more people with diabetes.
3. The cautious model catches more of them (recall 81.5 %) at the cost of more unneeded tests.
4. Recall and precision pull in opposite directions; choosing the bar is a decision about people.
5. A few people with diabetes have normal readings, so no model here catches everyone.

Limitations

- Made-up data — the numbers show the method, not real medical facts
- Only six features; a real screening would also use tests such as HbA1c
- A small testing set, so one person more or less changes the percentages
- It cannot diagnose — it can only suggest who should be tested

Future scope

- Try every value of k as well as every bar, and pick the pair with the best recall for an acceptable number of tests
- Show the doctor why a person was referred — which of their readings were high
- Practise on a public dataset such as the Pima Indians Diabetes dataset, credited
- Add a third answer, "see a doctor today", for people whose neighbours all have diabetes

What had to be learnt along the way

- True and false positives and negatives, and which one hurts most in a medical problem
- Precision, recall and F1, and why the F1 score is not always the right judge
- The four principles of bioethics from Unit 1, and how they apply to a health model
- Why a reading of 0 can mean "missing" rather than a real zero

CONCLUSION

The project built a working screening model, but its most important result is not a number. On accuracy, precision and F1 the majority model wins. On recall — the people with diabetes who are caught — the cautious model wins, and in a health camp that is the measure that matters.

Choosing the metric is part of solving the problem. A person sent home with undetected diabetes is a far worse outcome than a healthy person sent for one extra test, so the evaluation was built around that fact.

The model should help a doctor sort a queue, never replace one. It still misses some people, and saying so plainly is part of using AI responsibly.

REFERENCES

1. World Health Organization, Diabetes fact sheet — https://www.who.int/news-room/fact-sheets/detail/diabetes
2. Smith, J. W. et al. (1988). Using the ADAP learning algorithm to forecast the onset of diabetes mellitus — the source of the public Pima Indians Diabetes dataset often used for practice.
3. Orange Data Mining, Test and Score widget — https://orangedatamining.com/widget-catalog/evaluate/testandscore/
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