(name of your school)

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

on

“SPOTTING CORAL BLEACHING”

Computer Vision · SDG 14: Life below Water · SDG 13: Climate Action

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 **“Spotting Coral Bleaching”** 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	11
8	Ethical Concerns	11
9	The No-Code Version	11
10	Technical Details	12
11	Source Code	13
12	Sample Output	17
13	Testing	19
14	Results and Future Scope	19
15	Conclusion	20
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

Divers and underwater drones take thousands of photos of a reef on every survey. A scientist then has to look at each one and decide: is this coral healthy or bleached?

That takes weeks, and by the time the photos are sorted the bleaching may have spread. The colour of the coral is the main clue, and colour is something a computer can measure exactly.

This is the Computer Vision sample project in the CBSE curriculum, which builds it in Orange Data Mining. Here it is built in Python, so that every step — from pixels to prediction — can be seen.

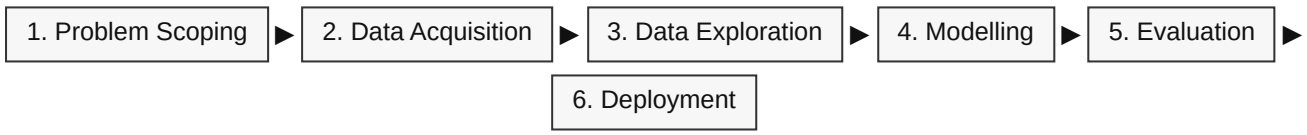
Who would use it: Marine scientists and trained divers who photograph reefs, such as those around Lakshadweep, the Andaman and Nicobar Islands and the Gulf of Mannar.

Link to the Sustainable Development Goals

SDG 14 — Life below Water; SDG 13 — Climate Action. Coral reefs are home to about a quarter of all sea life, though they cover a tiny part of the ocean. When the sea gets too warm, corals push out the tiny algae that live in them and give them colour, and turn white — this is bleaching. A bleached coral can recover if the water cools in time, but repeated bleaching kills reefs. Finding it early tells scientists where reefs need protection, which is SDG 14; the warming behind it is SDG 13.

The AI Project Cycle

This is a **Computer Vision** 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 • Marine scientists studying coral reefs • Divers and reef-protection groups who take the photos	**What?** THE PROBLEM • Thousands of reef photos are too many to check by eye quickly • Bleaching found late may be too late to protect a reef
Where? CONTEXT • Coral reefs, such as Lakshadweep and the Gulf of Mannar • After each survey dive, especially in hot summers	**Why?** BENEFIT • Bleached areas can be found and protected sooner • Scientists' time goes to the reefs that need it

Problem statement

> Our **marine scientists** have a problem of **sorting thousands of reef photos into healthy and bleached coral** when **every survey brings back more photos than they can check quickly**. An ideal solution would **sort the photos automatically, so bleached areas are found early**.

How it is done today

Scientists checking every photo. Accurate, but very slow. A single survey can take weeks to sort.

Coral colour charts. Divers hold a card of coral colours up to the reef and note the closest match. Clever and cheap, but done by hand, one coral at a time.

Satellite sea-temperature alerts. Satellites warn when the sea is hot enough to cause bleaching. They predict the risk; they cannot see whether a particular coral has actually bleached.

Objectives

1. To read reef photos into Python as arrays of pixel values
2. To see how a colour photo is stored as red, green and blue layers
3. To separate coral pixels from water pixels
4. To measure two features of the coral — brightness and colourfulness — from every photo
5. To build a model that learns the average healthy and bleached coral, and test it on unseen photos
6. To look at the photos it got wrong and understand why
7. To label new photos from a survey

DATA ACQUISITION

The data is a folder of photos, one folder for each class: images/healthy/ and images/bleached/. This is the same layout that Teachable Machine and Orange use.

There are 160 photos, each 64 × 64 pixels. A second folder, new_photos/, holds four photos from a new survey that the model has never seen.

These pictures were drawn by a program, not taken in the sea, so that the project can be shared freely and run anywhere. The curriculum's own coral dataset (linked from the CBSE curriculum) uses real photos. For your project, use that dataset or another public one, and credit it — never present drawn pictures as real reef photos.

> **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
`photo`	file	The photo, read as a 64 × 64 × 3 array of numbers	not used

coral_pixels	integer	How many pixels were counted as coral	not used
brightness	decimal	Average of the coral pixels' R, G and B values (0 = black, 255 = white)	feature
colourfulness	decimal	Average gap between each coral pixel's largest and smallest colour value (0 = grey or white)	feature
label	text	healthy or bleached — the folder the photo came from	label

The images/ folder

Reef photos sorted into a healthy folder and a bleached folder. Lighting and the blue cast of deep water change from photo to photo, and some corals are part-way, so the job is not trivially easy. It holds **160 images**, one folder per class:

FOLDER	IMAGES
images/bleached/	80
images/healthy/	80

The new_photos/ folder

Four photos from a new survey, used in the deployment stage. It holds **4 images**.

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
Most of each photo is water, not coral	`about 70% of the pixels`	Averaging the whole photo would mostly measure the water. So the program first finds the coral: water has very little red, coral has a lot, so a pixel with red above 77 (out of 255) counts as coral. Deciding which pixels belong to the object is called segmentation.
Lighting changes from photo to photo	`every photo`	Deep or cloudy water makes everything darker and bluer. This is why the project measures colourfulness as well as brightness: a dim white coral is still grey, not brown.
A photo is 12,288 numbers	`64 × 64 × 3`	Too many for a simple model to use directly. Feature extraction turns each photo into just two numbers that describe what matters — how bright and how colourful the coral is.

Figure 1: Eight photos of each kind

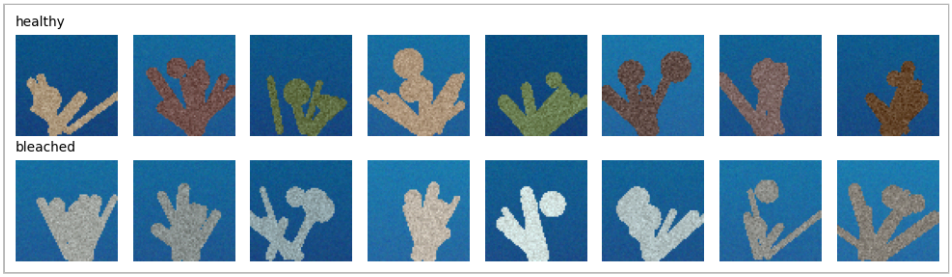


Figure 1 — Eight photos of each kind, drawn by the program as a set of images and saved as chart1_sample_photos.png.

How to read it. The top row is healthy coral; the bottom row is bleached. They were read from the folders with plt.imread() and shown with plt.imshow() — the practical-file program "read an image and display it using Python".

To our eyes the difference is colour: healthy coral is brown, olive or orange, and bleached coral is white or grey. But some healthy corals are pale and some bleached ones still have a little colour, and dim lighting makes everything look darker. The model will find those photos hardest.

Figure 2: One photo, split into red, green and blue

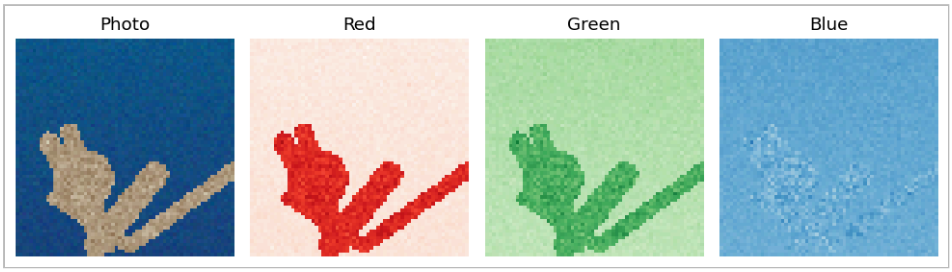


Figure 2 — One photo, split into red, green and blue, drawn by the program as a set of images and saved as chart2_rgb_layers.png.

How to read it. The same photo four times: as we see it, then each of its three colour layers on its own. In each layer, a darker colour means a bigger number.

This is how the computer stores a colour photo: three grids of numbers, one each for red, green and blue. The water is strong in blue and weak in red. The coral is strong in red. That one difference is what lets the program separate coral from water.

Figure 3: Two features measured from every photo

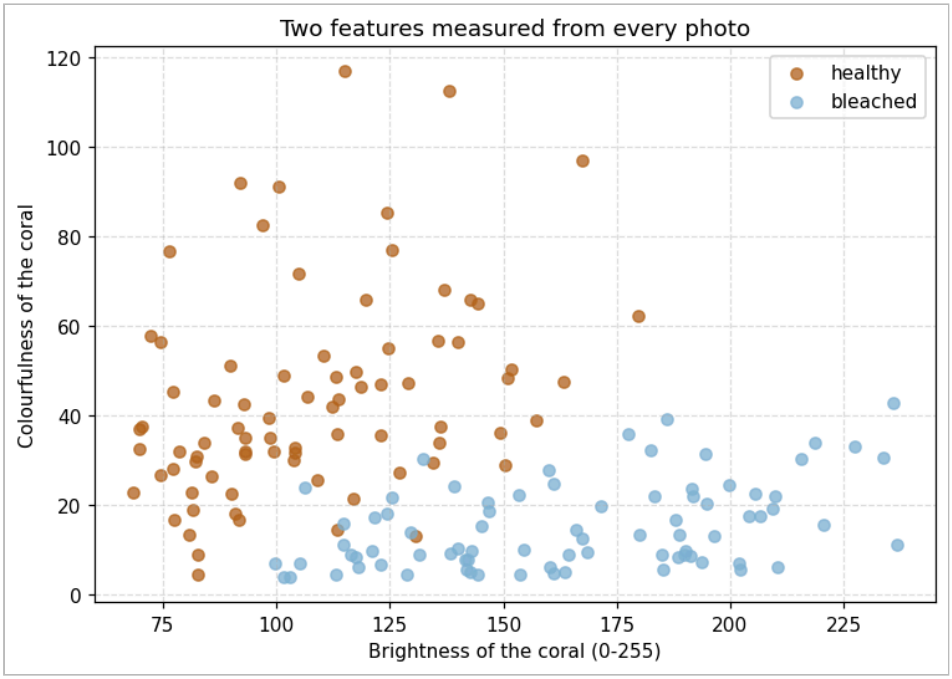


Figure 3 — Two features measured from every photo, drawn by the program as a scatter plot and saved as chart3_features.png.

How to read it. Each dot is one photo: how bright its coral is (across) and how colourful (up).

The two kinds form two clouds. Healthy coral sits up and to the left — colourful and darker. Bleached coral sits low and to the right — grey or white, and brighter. Where they meet, dim bleached corals sit close to pale healthy ones. That overlap is where the mistakes will be.

MODELLING

AI model ▶ Learning based ▶ Supervised learning ▶ Classification

The answer is one of two classes — healthy or bleached — so this is classification. Every training photo comes from a labelled folder, so it is supervised learning. Because the input is images, it belongs to the Computer Vision domain.

How the model works

The photos are shuffled and split into 128 for training and 32 for testing.

Each photo has already been turned into two features. The model learns the average brightness and colourfulness of healthy coral, and the same for bleached coral, from the training photos. To label a new photo, it measures the photo's two features and picks whichever average is closer.

This is the classic way computer vision worked before deep learning: a person decides which features matter (here, colour) and writes code to measure them. A Convolutional Neural Network, from Unit 5, learns its own features from the pixels instead. That is what Teachable Machine and Orange's Image Embedding do behind the scenes.

EVALUATION

On 32 testing photos the model reached an accuracy of 84.4 %.

For bleaching, the costly mistake is a missed bleached coral (a false negative): a reef that needs help goes unnoticed. A false alarm only means a scientist looks at one extra photo. So recall matters most. The recall was 75.0 % and the precision 100.0 %.

That recall is the honest weak point of this model. The mistakes chart shows why: dimly lit bleached corals look darker and slightly coloured. A better feature — for example, comparing the coral's colour with the colour of the water around it, to cancel out the lighting — is the obvious next step.

Figure 4: Confusion matrix

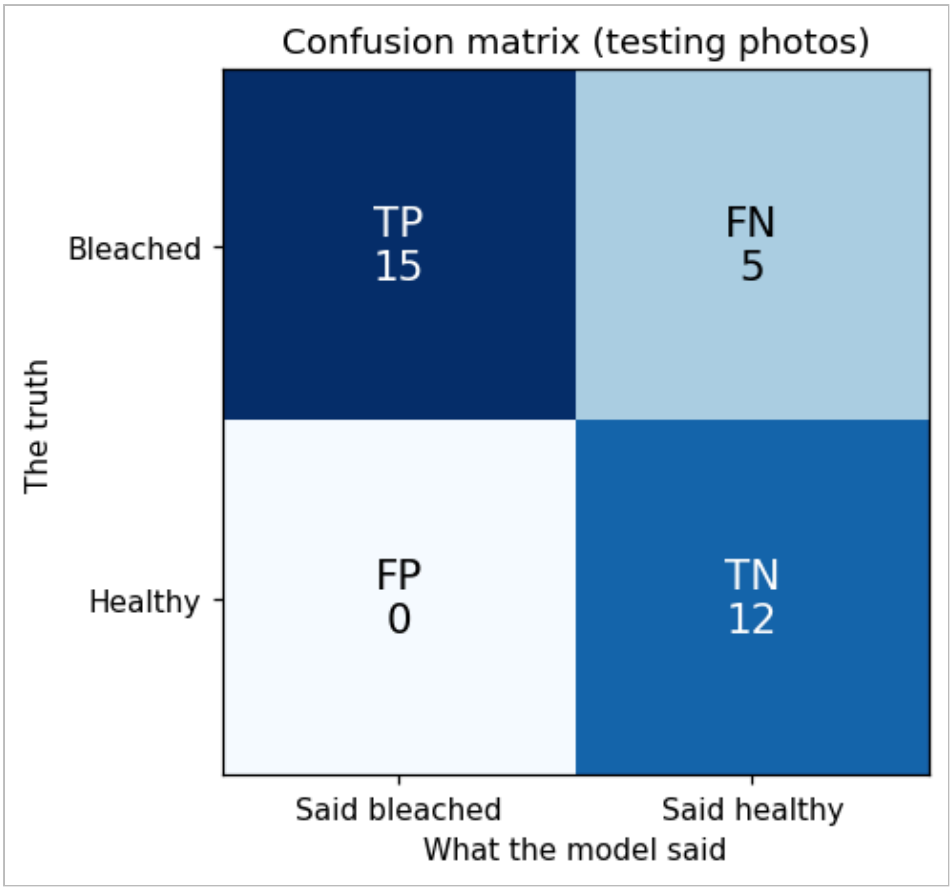


Figure 4 — Confusion matrix, drawn by the program as a confusion matrix and saved as chart4_matrix.png.

How to read it. Bleached is the positive class, because it is what the scientists are looking for. TP: bleached, and found. FN: bleached, but called healthy. FP: healthy, but called bleached. TN: healthy, and called healthy.

On the testing photos the counts are 15 5 0 12 (TP, FN, FP, TN). The model never raised a false alarm, but it missed some bleached coral.

Figure 5: The photos it got wrong

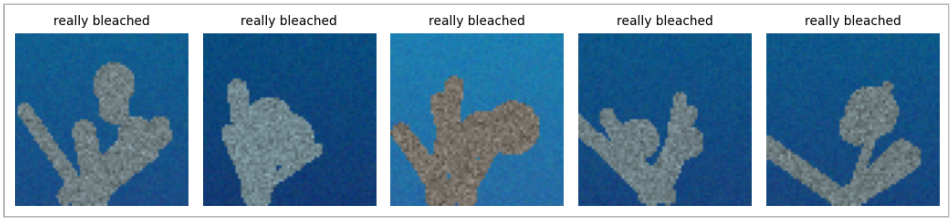


Figure 5 — The photos it got wrong, drawn by the program as a set of images and saved as chart5_mistakes.png.

How to read it. Every testing photo the model labelled wrongly, with its true label.

All the mistakes are bleached corals that it called healthy, and they have something in common: they are dimly lit, with brightness between about 115 and 130, and they still carry a little colour. In dark water, a bleached coral is not bright white any more. Looking at the mistakes tells you exactly what to fix next — which a single accuracy number never does.

DEPLOYMENT

The program reads four new survey photos it has never seen, measures their features, and labels each one. It saves them with their labels as a picture.

In real use, a scientist would point the program at a whole folder of survey photos and get back a list of the ones labelled bleached, to check first. The program does the sorting; the scientist does the judging.

Figure 6: Four photos from a new survey

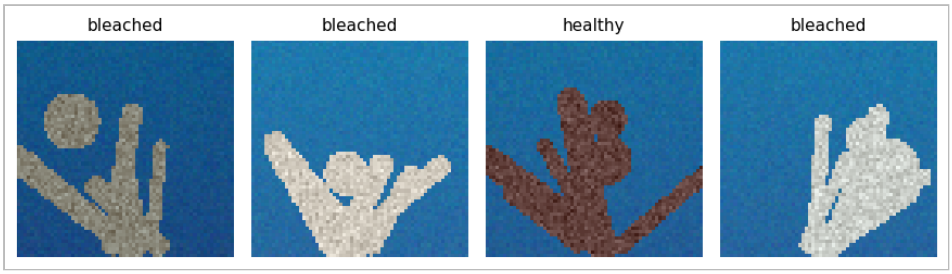


Figure 6 — Four photos from a new survey, drawn by the program as a set of images and saved as chart6_new_photos.png.

How to read it. The four new photos, each with the label the model gave it.

Three are labelled correctly. The first was drawn as a healthy but very pale coral, and the model calls it bleached. That is a false alarm — but in a real reef, a coral that has gone pale is often starting to bleach, so it is exactly the photo a scientist would want to look at anyway.

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).

Where the photos come from

Real reef photos belong to the scientists or divers who took them. Use public datasets, credit them, and never present drawn or edited pictures as real survey photos.

Bias in the data

A model trained on one reef, in one kind of water, may fail on another — different coral species, different water colour. It must be tested on photos from the reef where it will be used.

Accountability

A missed bleaching event could mean a reef is not protected in time. The model should help scientists check faster, not replace their checking.

THE NO-CODE VERSION

The curriculum names both tools for Computer Vision. Teachable Machine trains an image model in the browser; Orange does the same with widgets. The tool used is **Teachable Machine (or Orange with the Image Analytics add-on)**.

STEP	SCREEN	WHAT IT DOES

1	**Teachable Machine: New Image Project**	Open teachablemachine.withgoogle.com and choose an Image Project, standard image model.
2	**Class 1 and Class 2**	Rename them healthy and bleached, and upload the photos from each folder.
3	**Train Model**	Click Train. It learns its own features from the pixels, using a neural network.
4	**Preview**	Upload a photo from new_photos and read the confidence for each class.
5	**Orange: Import Images**	Point it at the images folder. The folder names become the class labels.
6	**Orange: Image Embedding**	Turns each photo into features, using a CNN (it needs an internet connection).
7	**Orange: Test and Score + Logistic Regression**	Train a model on the embeddings and read the accuracy, precision and recall.
8	**Orange: Confusion Matrix → Image Viewer**	Select the wrong cells in the matrix and view those photos — the same as chart 5.

TECHNICAL DETAILS

Domain of AI	Computer Vision
Programming language	Python 3
Libraries	pandas, NumPy and matplotlib (Unit 7 of the curriculum)
Where to run it	Jupyter Notebook (coral_bleaching.ipynb), or any Python editor (coral_bleaching.py)
No-code tool	Teachable Machine (or Orange with the Image Analytics add-on)
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 coral_bleaching.py**, or open **coral_bleaching.ipynb** and run each cell in order.

The library calls it uses

CALL	STAGE	WHAT IT DOES
`os.listdir()`	Stage 2	Lists the photos in each folder
`plt.imread()`	Stage 2	Reads a photo into an array of numbers
`img.shape`	Stage 2	The size of the array: height, width, colours
`img[40, 32]`	Stage 2	One pixel's red, green and blue values
`plt.imshow() / plt.subplot()`	Stages 2, 3, 5, 6	Shows photos side by side

`img[:, :, 0]`	Stage 3	The red layer on its own
`img[:, :, 0] > 77`	Stage 3	Marks which pixels are coral (segmentation)
`pixels.mean()`	Stage 3	Brightness of the coral
`pixels.max(axis=1) - pixels.min(axis=1)`	Stage 3	How colourful each coral pixel is
`groupby(...).mean()`	Stage 4	The average healthy and bleached coral
`idxmin()`	Stage 4	The closer of the two averages

SOURCE CODE

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

`coral_bleaching.py`

```
# ------------------------------------------------------------------------
# coral_bleaching.py
#
# AI project on Computer Vision: telling healthy coral from bleached coral in
# underwater photographs, so that bleaching can be spotted early.
#
# CBSE Class X, Artificial Intelligence (417), Part C project work.
# SDG 14 (Life below Water) and SDG 13 (Climate Action).
#
# A computer sees a photo as a grid of pixels, and each pixel as three
# numbers from 0 to 255 - how much red, green and blue it holds. The program:
#   1. reads every photo into those numbers,
#   2. finds which pixels are coral and which are water (segmentation),
#   3. measures two FEATURES of the coral: how bright it is, and how
#      colourful it is, and
#   4. learns the average healthy coral and the average bleached coral from
#      the training photos, and labels a new photo by whichever it is closer to.
#
# Libraries: numpy (the pixel arithmetic), matplotlib (reading and showing
# images, and the charts), pandas (the table of features), os (the folders).
# ------------------------------------------------------------------------

# %% Stage 1 - Problem scoping
print("=" * 60)
print("SPOTTING CORAL BLEACHING")
print("=" * 60)
print("Who   : marine scientists and reef divers")
print("What  : thousands of reef photos, too many to check by eye")
print("Where : coral reefs, such as those around Lakshadweep")
print("Why   : bleaching caught early shows where reefs need protection")
print()

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

pd.set_option("display.width", 120)
```

```
classes = ["healthy", "bleached"]
paths = []
labels = []
for c in classes:
    for name in sorted(os.listdir(os.path.join("images", c))):
        paths.append(os.path.join("images", c, name))
        labels.append(c)

print("--- Stage 2: Data acquisition ---")
print("Healthy photos  :", labels.count("healthy"))
print("Bleached photos :", labels.count("bleached"))
print("Total photos    :", len(paths))

# Read one photo and look at what the computer actually sees.
first = plt.imread(paths[0])
print("One photo is stored as an array of shape :", first.shape)
print("Height x width x colours                 : 64 x 64 x 3 = 12288 numbers")
pixel = (first[40, 32] * 255).round()
print("The pixel at row 40, column 32 (R, G, B) :", int(pixel[0]), int(pixel[1]), int(pixel[2]))
print()

# Chart 1: eight photos of each kind, spread through each folder.
healthy_paths = []
bleached_paths = []
for i in range(len(paths)):
    if labels[i] == "healthy":
        healthy_paths.append(paths[i])
    else:
        bleached_paths.append(paths[i])
plt.figure(figsize=(9, 2.6))
for i in range(8):
    plt.subplot(2, 8, i + 1)
    plt.imshow(plt.imread(healthy_paths[i * len(healthy_paths) // 8]))
    plt.axis("off")
    if i == 0:
        plt.title("healthy", fontsize=9, loc="left")
    plt.subplot(2, 8, i + 9)
    plt.imshow(plt.imread(bleached_paths[i * len(bleached_paths) // 8]))
    plt.axis("off")
    if i == 0:
        plt.title("bleached", fontsize=9, loc="left")
plt.tight_layout()
plt.savefig("chart1_sample_photos.png")
plt.close()

# %% Stage 3 - Data exploration
print("--- Stage 3: Data exploration ---")

# Chart 2: one photo split into its red, green and blue layers.
plt.figure(figsize=(9, 2.8))
plt.subplot(1, 4, 1)
plt.imshow(first)
plt.title("Photo")
plt.axis("off")
channel_names = ["Red", "Green", "Blue"]
channel_maps = ["Reds", "Greens", "Blues"]
for ch in range(3):
    plt.subplot(1, 4, ch + 2)
    plt.imshow(first[:, :, ch], cmap=channel_maps[ch], vmin=0, vmax=1)
    plt.title(channel_names[ch])
    plt.axis("off")
```

```
plt.tight_layout()
plt.savefig("chart2_rgb_layers.png")
plt.close()

# Feature extraction. Water is blue-green with very little red, and coral -
# healthy or bleached - has a lot of red. So a pixel with red above 0.3
# (about 77 out of 255) is counted as coral. This is segmentation: deciding
# which pixels belong to the object.
rows = []
for i in range(len(paths)):
    img = plt.imread(paths[i])[:, :, :3].astype(float) * 255
    coral = img[:, :, 0] > 77
    pixels = img[coral]                                  # only the coral pixels
    brightness = pixels.mean()                            # 0 = black, 255 = white
    colourful = (pixels.max(axis=1) - pixels.min(axis=1)).mean()   # grey/white = 0
    rows.append([paths[i], labels[i], coral.sum(), round(brightness, 1), round(colourful, 1)])

features = pd.DataFrame(rows, columns=["photo", "label", "coral_pixels", "brightness",
"colourfulness"])
print("Features measured from each photo (first 5 of each kind):")
print(features[features["label"] == "healthy"].head(5).to_string(index=False))
print(features[features["label"] == "bleached"].head(5).to_string(index=False))
print()
print("Average features of each kind:")
print(features.groupby("label")[["coral_pixels", "brightness", "colourfulness"]].mean().round(1))
print()

# Chart 3: the two features, one dot per photo.
plt.figure(figsize=(7, 5))
colours = ["#b5651d", "#7fb3d5"]
for k in range(2):
    part = features[features["label"] == classes[k]]
    plt.scatter(part["brightness"], part["colourfulness"], color=colours[k], label=classes[k],
alpha=0.75)
plt.title("Two features measured from every photo")
plt.xlabel("Brightness of the coral (0-255)")
plt.ylabel("Colourfulness of the coral")
plt.legend()
plt.grid(True, linestyle="--", alpha=0.4)
plt.tight_layout()
plt.savefig("chart3_features.png")
plt.close()

# %% Stage 4 - Modelling
print("--- Stage 4: Modelling ---")
use = ["brightness", "colourfulness"]

features = features.sample(frac=1, random_state=8)
cut = int(len(features) * 0.8)
train = features[:cut]
test = features[cut:]
print("Training photos :", len(train))
print("Testing photos  :", len(test))

low = train[use].min()
high = train[use].max()
averages = ((train[use] - low) / (high - low)).groupby(train["label"]).mean()
print()
print("What the model learned - the average of each kind (rescaled 0 to 1):")
print(averages.round(2))
print()
```

```
test_scaled = (test[use] - low) / (high - low)
predictions = []
for i in range(len(test)):
    distance = ((averages - test_scaled.iloc[i]) ** 2).sum(axis=1)
    predictions.append(distance.idxmin())

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

# "bleached" is the positive class: it is what the scientists are looking for.
tp = fn = fp = tn = 0
for i in range(len(actual)):
    if actual[i] == "bleached" and predictions[i] == "bleached":
        tp = tp + 1
    elif actual[i] == "bleached":
        fn = fn + 1
    elif predictions[i] == "bleached":
        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)
print("TP, FN, FP, TN :", tp, fn, fp, tn)
print("Accuracy       :", round(accuracy, 1), "%")
print("Precision      :", round(precision, 1), "%")
print("Recall         :", round(recall, 1), "%")
print("F1 score       :", round(f1, 1), "%")
print()

# Looking at the mistakes is the most useful check in computer vision.
wrong = []
for i in range(len(actual)):
    if predictions[i] != actual[i]:
        wrong.append(i)
print("Photos the model got wrong :", len(wrong))
for i in wrong:
    row = test.iloc[i]
    print("  " + os.path.basename(row["photo"]).ljust(18), "really", row["label"].ljust(8),
          " brightness", row["brightness"], " colourfulness", row["colourfulness"])
print()

# Chart 5: the photos it got wrong.
if len(wrong) > 0:
    plt.figure(figsize=(1.8 * len(wrong) + 0.5, 2.4))
    for n in range(len(wrong)):
        row = test.iloc[wrong[n]]
        plt.subplot(1, len(wrong), n + 1)
        plt.imshow(plt.imread(row["photo"]))
        plt.title("really " + row["label"], fontsize=9)
        plt.axis("off")
    plt.tight_layout()
```

```
    plt.savefig("chart5_mistakes.png")
    plt.close()

# Chart 4: confusion matrix.
grid = [[tp, fn], [fp, tn]]
plt.figure(figsize=(5, 4.3))
plt.imshow(grid, cmap="Blues")
names = [["TP", "FN"], ["FP", "TN"]]
for r in range(2):
    for c in range(2):
        if grid[r][c] > max(max(grid)) / 2:
            ink = "white"
        else:
            ink = "black"
        plt.text(c, r, names[r][c] + "\n" + str(grid[r][c]), ha="center", va="center", fontsize=14,
color=ink)
plt.xticks([0, 1], ["Said bleached", "Said healthy"])
plt.yticks([0, 1], ["Bleached", "Healthy"])
plt.xlabel("What the model said")
plt.ylabel("The truth")
plt.title("Confusion matrix (testing photos)")
plt.tight_layout()
plt.savefig("chart4_matrix.png")
plt.close()

# %% Stage 6 - Deployment
print("--- Stage 6: Deployment ---")
new_names = sorted(os.listdir("new_photos"))
plt.figure(figsize=(8, 2.6))
for i in range(len(new_names)):
    img = plt.imread(os.path.join("new_photos", new_names[i]))[:, :, :3].astype(float) * 255
    pixels = img[img[:, :, 0] > 77]
    point = pd.Series({
        "brightness": pixels.mean(),
        "colourfulness": (pixels.max(axis=1) - pixels.min(axis=1)).mean(),
    })
    distance = ((averages - (point - low) / (high - low)) ** 2).sum(axis=1)
    answer = distance.idxmin()
    print(new_names[i], "-> brightness", round(point["brightness"]), " colourfulness",
          round(point["colourfulness"]), "->", answer.upper())
    plt.subplot(1, len(new_names), i + 1)
    plt.imshow(img.astype("uint8"))
    plt.title(answer, fontsize=10)
    plt.axis("off")
plt.tight_layout()
plt.savefig("chart6_new_photos.png")
plt.close()
```

SAMPLE OUTPUT

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

Output — python coral_bleaching.py

```
============================================================
SPOTTING CORAL BLEACHING
============================================================
Who   : marine scientists and reef divers
What  : thousands of reef photos, too many to check by eye
```

```
Where : coral reefs, such as those around Lakshadweep
Why   : bleaching caught early shows where reefs need protection

--- Stage 2: Data acquisition ---
Healthy photos  : 80
Bleached photos : 80
Total photos    : 160
One photo is stored as an array of shape : (64, 64, 3)
Height x width x colours                 : 64 x 64 x 3 = 12288 numbers
The pixel at row 40, column 32 (R, G, B) : 30 81 139

--- Stage 3: Data exploration ---
Features measured from each photo (first 5 of each kind):
                         photo   label  coral_pixels  brightness  colourfulness
images/healthy/healthy_001.png healthy           866       151.0           48.2
images/healthy/healthy_002.png healthy          1414       117.0           21.5
images/healthy/healthy_003.png healthy          1074       123.1           35.6
images/healthy/healthy_004.png healthy           992        72.2           57.9
images/healthy/healthy_005.png healthy          1301       142.8           65.9
                          photo    label  coral_pixels  brightness  colourfulness
images/bleached/bleached_001.png bleached          1267       163.6            5.1
images/bleached/bleached_002.png bleached           813       113.0            4.6
images/bleached/bleached_003.png bleached          1448       144.4            4.6
images/bleached/bleached_004.png bleached           902       204.2           17.6
images/bleached/bleached_005.png bleached          1352       141.8            5.5

Average features of each kind:
          coral_pixels  brightness  colourfulness
label
bleached        1178.8       164.8           15.3
healthy         1066.4       108.1           43.8

--- Stage 4: Modelling ---
Training photos : 128
Testing photos  : 32

What the model learned - the average of each kind (rescaled 0 to 1):
          brightness  colourfulness
label
bleached        0.58           0.09
healthy         0.24           0.35

--- Stage 5: Evaluation ---
TP, FN, FP, TN : 15 5 0 12
Accuracy       : 84.4 %
Precision      : 100.0 %
Recall         : 75.0 %
F1 score       : 85.7 %

Photos the model got wrong : 5
  bleached_079.png   really bleached  brightness 124.4  colourfulness 18.0
  bleached_019.png   really bleached  brightness 132.4  colourfulness 30.4
  bleached_036.png   really bleached  brightness 114.7  colourfulness 15.8
  bleached_006.png   really bleached  brightness 125.4  colourfulness 21.7
  bleached_057.png   really bleached  brightness 121.5  colourfulness 17.4

--- Stage 6: Deployment ---
photo_1.png -> brightness 127  colourfulness 17 -> BLEACHED
photo_2.png -> brightness 200  colourfulness 19 -> BLEACHED
photo_3.png -> brightness 76  colourfulness 38 -> HEALTHY
photo_4.png -> brightness 203  colourfulness 6 -> BLEACHED
```

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
All 160 photos are read	`Total photos : 160`	`Total photos : 160`	Pass
A photo is a grid of numbers	`One photo is stored as an array of shape : (64, 64, 3)`	`One photo is stored as an array of shape : (64, 64, 3)`	Pass
Water pixels are dark red, not coral	`The pixel at row 40, column 32 (R, G, B) : 30 81 139`	`The pixel at row 40, column 32 (R, G, B) : 30 81 139`	Pass
Every bleached photo it calls bleached really is (no false alarms)	`Precision : 100.0 %`	`Precision : 100.0 %`	Pass
Pure brown against pure white: no mistakes	`Accuracy : 100.0 %`	`Accuracy : 100.0 %`	Pass
A new brown coral is called healthy	`photo_1.png -> brightness 100 colourfulness 95 -> HEALTHY`	`photo_1.png -> brightness 100 colourfulness 95 -> HEALTHY`	Pass
A new white coral is called bleached	`photo_2.png -> brightness 232 colourfulness 10 -> BLEACHED`	`photo_2.png -> brightness 232 colourfulness 10 -> BLEACHED`	Pass
The images folder is missing	`FileNotFoundError: [Errno 2] No such file or directory: 'images/healthy'`	`FileNotFoundError: [Errno 2] No such file or directory: 'images/healthy'`	Pass

RESULTS AND FUTURE SCOPE

What the project found

1. A colour photo is three grids of numbers; coral and water separate cleanly on the red layer.
2. Two features — brightness and colourfulness — are enough to tell most healthy and bleached coral apart.
3. On unseen photos the model reached an accuracy of 84.4 %, with no false alarms.
4. Its recall was 75.0 %: it misses some bleached coral, and every miss was a dimly lit photo.
5. Looking at the wrong photos showed the cause — lighting — which the accuracy figure alone could not.

Limitations

- Pictures drawn by a program, small (64 × 64) and simpler than real reef photos
- Lighting still fools it: dim bleached coral looks healthy
- The red-pixel rule for finding coral would fail on red sponges or a sandy sea floor
- Only two classes; a real survey also needs "partly bleached" and "dead"

Future scope

- Compare the coral's colour with the water around it, to cancel out lighting
- Train the same project in Teachable Machine and compare the results
- Use the curriculum's real coral dataset

- Add a third class for partly bleached coral

What had to be learnt along the way

- How a colour image is stored as red, green and blue values from 0 to 255
- Reading and showing images with plt.imread() and plt.imshow()
- Feature extraction and segmentation
- How a CNN differs from hand-made features — Unit 5

CONCLUSION

The project turned reef photos into numbers, picked out the coral, measured its colour, and used that to tell healthy coral from bleached. On photos it had never seen, the model was right 84.4 % of the time and never raised a false alarm.

Its weak point is recall, and the project found out exactly why by looking at the mistakes: every missed coral was a bleached one photographed in dim light. That is the most valuable result, because it says what the next version must do.

Building the features by hand made it clear what a computer is actually measuring when it "sees" — and why a CNN, which learns its own features, is used for harder problems.

REFERENCES

1. CBSE, Artificial Intelligence curriculum, Class X — Unit 5 practical: Build a real-world classification model: Coral Bleaching (use case walkthrough).
2. NOAA Coral Reef Watch — https://coralreefwatch.noaa.gov/
3. Teachable Machine — https://teachablemachine.withgoogle.com/
4. Orange Data Mining, Image Analytics add-on — https://orangedatamining.com/widget-catalog/image-analytics/
5. CBSE, Artificial Intelligence (Subject Code 417), Class X curriculum, 2026-27 — Part C, Practical and Project Work.
6. CBSE, Artificial Intelligence Facilitator Handbook, Class X.
7. pandas documentation — https://pandas.pydata.org/docs/
8. NumPy documentation — https://numpy.org/doc/
9. Matplotlib documentation — https://matplotlib.org/stable/
10. LambdaLab — https://www.lambdalab.in