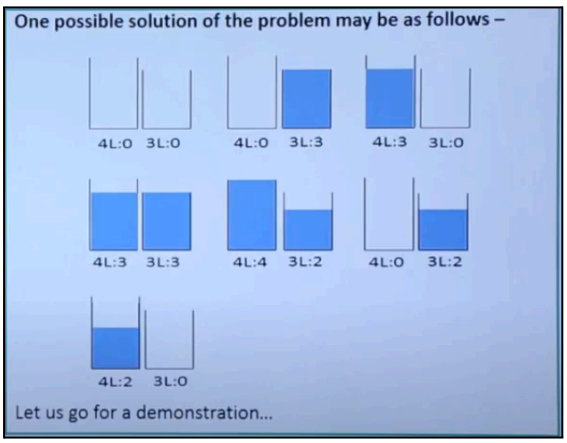
NANDITHA PS - U18EV23S0096

AI LAB

Write a program to solve the Water Jug Problem using Breadth First Search (BFS).

```
from collections import deque

def water_jug_bfs(a, b, target):
    q = deque([(0, 0)])
    visited = {(0, 0)}
    parent = {(0, 0): None}

    while q:
        x, y = q.popleft()
        if x == target:  # stop only when jug1 has target
            path = []
            while (x, y) is not None:
                path.append((x, y))
                prev = parent[(x, y)]
                if prev is None:
                    break
                x, y = prev
            return path[::-1]

        moves = [
            (a, y), (x, b), (0, y), (x, 0),
            (x - min(x, b - y), y + min(x, b - y)),
            (x + min(y, a - x), y - min(y, a - x))
        ]
        for m in moves:
            if m not in visited:
                visited.add(m)
                q.append(m)
                parent[m] = (x, y)
    return None

# Example
path = water_jug_bfs(4, 3, 2)
for step in path:
    print(step)
```

```
(0, 0)
(4, 0)
(1, 3)
(1, 0)
(0, 1)
(4, 1)
(2, 3)
```



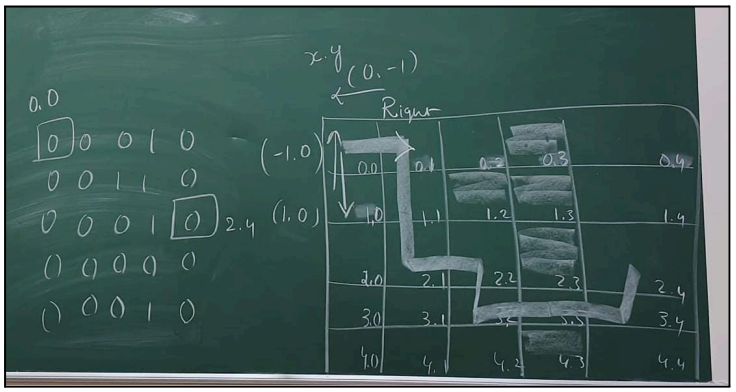
2)Write a program to find the optimum path from Source to Destination using A* search technique in python

```
import heapq

def astar(grid, start, goal):
    h = lambda a,b: abs(a[0]-b[0]) + abs(a[1]-b[1])
    q, cost = [(0, start, [start])], {start: 0}
    while q:
        _, curr, path = heapq.heappop(q)
        if curr == goal: return path
        for dx, dy in [(0,1),(1,0),(0,-1),(-1,0)]:
            x, y = curr[0]+dx, curr[1]+dy
            if 0<=x<len(grid) and 0<=y<len(grid[0]) and grid[x][y]==0:
                nc = cost[curr]+1
                if (x,y) not in cost or nc < cost[(x,y)]:
                    cost[(x,y)] = nc
                    heapq.heappush(q, (nc+h((x,y),goal), (x,y),
path+[(x,y)]))
    return None

grid = [
    [0,0,0,1,0],
    [0,0,1,1,0],
    [0,0,0,1,0],
    [0,0,0,0,0],
```

```
    [0,0,0,1,0]
]
print("Path:", astar(grid,(0,0),(2,4)) or "No path")
```

```
Path: [(0, 0), (0, 1), (1, 1), (2, 1), (2, 2), (3, 2), (3, 3), (3, 4), (2, 4)]
```

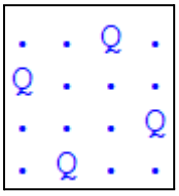
Note: Change co-ordinates to find the shortest path

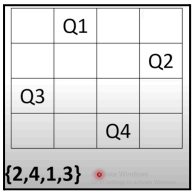


3) Write a program to solve the 4 – Queens Problem.

```
def is_safe(board, row, col):
    # Check left in the same row
    for i in range(col):
        if board[row][i] == 1:
            return False
    # Check upper diagonal
    for i, j in zip(range(row, -1, -1), range(col, -1, -1)):
        if board[i][j] == 1:
            return False
    # Check lower diagonal
    for i, j in zip(range(row, len(board)), range(col, -1, -1)):
        if board[i][j] == 1:
            return False
    return True

def solve(board, col=0):
    if col == len(board):  # All queens placed
        return True
    for row in range(len(board)):
        if is_safe(board, row, col):
            board[row][col] = 1
```

```
            if solve(board, col + 1):
                return True
            board[row][col] = 0  # Backtrack
    return False

board = [[0]*4 for _ in range(4)]
if solve(board):
    for row in board:
        print(" ".join("Q" if x else "." for x in row))
else:
    print("No solution")
```





Note: Backtracking concept is used

4. Write a program to implement Minimax search for 2 Player games.

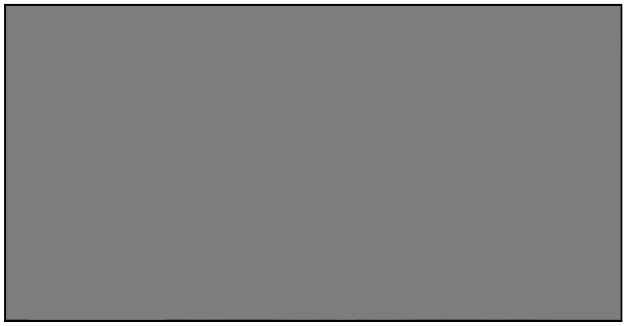
```
W=[(0,1,2),(3,4,5),(6,7,8),(0,3,6),(1,4,7),(2,5,8),(0,4,8),(2,4,6)]
win=lambda b,p:any(all(b[i]==p for i in l)for l in W)
def mm(b,t):
    if win(b,'O'):return 1
    if win(b,'X'):return-1
    if ' 'not in b:return 0
    s=[mm(b[:i]+[t]+b[i+1:],'OX'[t=='O'])for i in range(9)if b[i]==' ']
    return(max if t=='O'else min)(s)

b=[' ']*9
while True:
    [print(b[i:i+3])for i in range(0,9,3)]
    if win(b,'X'):print("You win!");break
    if win(b,'O'):print("AI wins!");break
    if ' 'not in b:print("Draw!");break
    p=int(input("Your move (0-8): "))
    if b[p]!=' ':continue
    b[p]='X'
    if ' 'not in b or win(b,'X'):continue
    m=max((mm(b[:i]+['O']+b[i+1:],'X'),i)for i in range(9)if b[i]==' ')[1]
    b[m]='O'
```

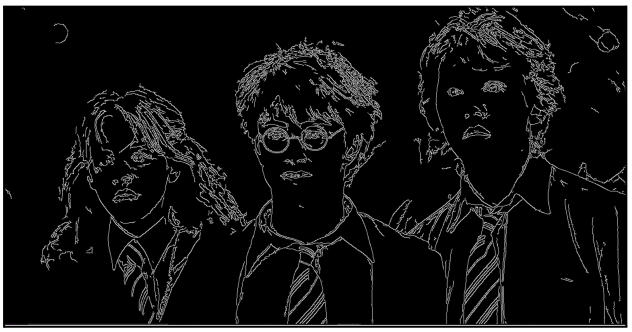
```
[' ', ' ', ' ']
[' ', ' ', ' ']
[' ', ' ', ' ']
Your move (0-8): 3
[' ', ' ', ' ']
['X', ' ', ' ']
['O', ' ', ' ']
Your move (0-8): 8
[' ', ' ', ' ']
['X', ' ', 'O']
['O', ' ', 'X']
Your move (0-8): 0
['X', ' ', ' ']
['X', 'O', 'O']
['O', ' ', 'X']
Your move (0-8): 2
['X', 'O', 'X']
['X', 'O', 'O']
['O', ' ', 'X']
Your move (0-8): 7
['X', 'O', 'X']
['X', 'O', 'O']
['O', 'X', 'X']
Draw!
```

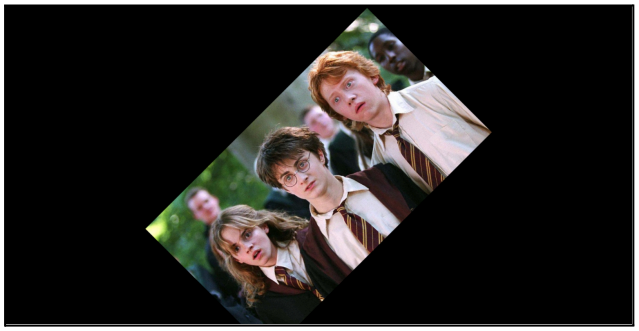
5)Using OpenCV python library capture an image and perform the following image processing operations: a) Image Resizing b) Blurring of Image c) Grayscaling of image d) Scaling and rotation e) Edge Detection f) Segmentation using thresholding g) Background subtraction h) Morphological operations

```
import cv2
import numpy as np

# Load static image
image_path = r"C:\Users\user\Desktop\pexels-divinetechygirl-1181623.jpg"   # Change to your image path
captured_image = cv2.imread(image_path)

if captured_image is None:
    print("Error: Could not load the image.")
    exit()

# a) Image Resizing
resized_img = cv2.resize(captured_image, (300, 300))

# b) Blurring of Image
blurred_img = cv2.GaussianBlur(captured_image, (11, 11), 0)
```

```
# c) Grayscaling of image
gray_img = cv2.cvtColor(captured_image, cv2.COLOR_BGR2GRAY)

# d) Scaling and rotation
(h, w) = captured_image.shape[:2]
center = (w // 2, h // 2)
M = cv2.getRotationMatrix2D(center, 45, 0.5)  # rotate 45° and scale to 50%
rotated_scaled_img = cv2.warpAffine(captured_image, M, (w, h))

# e) Edge Detection (Canny)
edges = cv2.Canny(gray_img, 100, 200)

# f) Segmentation using Thresholding
_, threshold_img = cv2.threshold(gray_img, 127, 255, cv2.THRESH_BINARY)

# g) Background subtraction
fgbg = cv2.createBackgroundSubtractorMOG2()
bg_subtracted = fgbg.apply(captured_image)

# h) Morphological operations
kernel = np.ones((5, 5), np.uint8)
morph_img = cv2.morphologyEx(threshold_img, cv2.MORPH_CLOSE, kernel)

# Display results using OpenCV's cv2.imshow() for local execution
cv2.imshow("Original Image", captured_image)       # Original
cv2.imshow("Resized Image", resized_img)          # Resized
cv2.imshow("Blurred Image", blurred_img)          # Blurred
cv2.imshow("Grayscale Image", gray_img)            # Grayscale
cv2.imshow("Rotated & Scaled Image", rotated_scaled_img)   # Rotated & Scaled
cv2.imshow("Edges", edges)                # Edges
cv2.imshow("Threshold Image", threshold_img)        # Threshold
cv2.imshow("Background Subtraction", bg_subtracted)        # Background
Subtraction
cv2.imshow("Morphology Image", morph_img)           # Morphology

# Wait until a key is pressed to close all windows
cv2.waitKey(0)
cv2.destroyAllWindows()
```







6)Write a program with two menu options 1) Capture Image and 2) Recognise Image. This program should capture pictures of five students and save them. The program should identify/recognise the student and display the student name.

```
import os
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.preprocessing.image import load_img, img_to_array
```

```
def load_images_from_folder(folder, target_size=(28, 28)):
    images = []
    image_paths = []


    for filename in os.listdir(folder):
        img_path = os.path.join(folder, filename)
        if os.path.isfile(img_path):
            img = load_img(img_path, target_size=target_size,
color_mode='grayscale')
            img_array = img_to_array(img) / 255.0
            images.append(img_array)
            image_paths.append(img_path)
    return np.array(images), image_paths


def display_image(image_path):
    img = load_img(image_path, color_mode='grayscale')
    plt.imshow(img, cmap='gray')
    plt.title(f'Selected Image: {os.path.basename(image_path)}')
    plt.axis('off')
    plt.show()


# Modified function to directly use the provided path
def display_selected_image(image_paths, selected_image_name):
    # Find the full path of the selected image
    selected_image_path = next((path for path in image_paths if
os.path.basename(path) == selected_image_name), None)

    if selected_image_path:
        display_image(selected_image_path)
        print(f'Selected Image Name: {selected_image_name}')
    else:
        print("Selected image is not in the dataset.")


folder_path = '/content/ai '
images, image_paths = load_images_from_folder(folder_path)


print("Loaded images shape:", images.shape)
```

```
print("Available images:", [os.path.basename(path) for path in
image_paths])


# Hardcode the image name here
selected_image_name = 'image.13.38 (2).jpeg'  # Replace with your
desired image name


display_selected_image(image_paths, selected_image_name)
```


```
Selected Image Name: image.13.38 (2).jpeg
```

7) Using Keras/any standard dataset write the programs for the following Machine learning tasks: 7. Use the Decision tree classifier to classify the dataset.

```
!pip install scikit-learn
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn import datasets
from sklearn.metrics import accuracy_score
# Load the Iris dataset
iris = datasets.load_iris()
X = iris.data
y = iris.target
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=42)
clf = DecisionTreeClassifier()
clf.fit(X_train, y_train)
```

```
y_pred = clf.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")
```

```
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.12/dist-packages (1.6.1)
Requirement already satisfied: numpy>=1.19.5 in /usr/local/lib/python3.12/dist-packages (from scikit-learn) (2.0.2)
Requirement already satisfied: scipy>=1.6.0 in /usr/local/lib/python3.12/dist-packages (from scikit-learn) (1.16.1)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.12/dist-packages (from scikit-learn) (1.5.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.12/dist-packages (from scikit-learn) (3.6.0)
Accuracy: 1.00
```

8) Use the Naïve Bayes classifier to classify the dataset.

```
!pip install scikit-learn
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn import datasets
from sklearn.metrics import accuracy_score
# Load the Iris dataset
iris = datasets.load_iris()
X = iris.data
y = iris.target
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=42)
clf = GaussianNB()
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")
```

```
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.12/dist-packages (1.6.1)
Requirement already satisfied: numpy>=1.19.5 in /usr/local/lib/python3.12/dist-packages (from scikit-learn) (2.0.2)
Requirement already satisfied: scipy>=1.6.0 in /usr/local/lib/python3.12/dist-packages (from scikit-learn) (1.16.1)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.12/dist-packages (from scikit-learn) (1.5.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.12/dist-packages (from scikit-learn) (3.6.0)
Accuracy: 0.98
```

9. Implement K-Means clustering Algorithm.

```
from sklearn import datasets
from sklearn.cluster import KMeans
from sklearn.metrics import accuracy_score
import numpy as np

# Load the Iris dataset
iris = datasets.load_iris()
X = iris.data
y = iris.target

# Apply KMeans clustering
kmeans = KMeans(n_clusters=4, random_state=42)  # 3 clusters for 3
classes
y_pred = kmeans.fit_predict(X)
```

```
# Since clustering doesn't use labels, we compare clusters with true
labels
# But clusters may not match label numbers directly -> need mapping
from scipy.stats import mode

labels = np.zeros_like(y_pred)
for i in range(4):  # 3 clusters
    mask = (y_pred == i)
    labels[mask] = mode(y[mask], keepdims=True)[0]

# Calculate accuracy
accuracy = accuracy_score(y, labels)
print(f"KMeans Clustering Accuracy: {accuracy:.2f}")
```

```
KMeans Clustering Accuracy: 0.84
```

10. Using Python NLTK, perform the following Natural Language Processing (NLP) tasks for any textual content. a) Tokenizing b) Filtering Stop Words c) Stemming d) Part of Speech tagging e) Chunking f) Named Entity Recognition (NER)

```
import nltk
nltk.download('averaged_perceptron_tagger_eng')
import nltk
from nltk.tokenize import word_tokenize, sent_tokenize
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from nltk import pos_tag, RegexpParser, ne_chunk

# --- Download necessary NLTK data (first time only) ---
nltk.download('punkt')
nltk.download('punkt_tab')    # needed in new NLTK versions
nltk.download('stopwords')
nltk.download('averaged_perceptron_tagger')
nltk.download('averaged_perceptron_tagger_eng')  # for English tagging
nltk.download('maxent_ne_chunker')
nltk.download('words')

# --- Input text ---
text = """MSRCASC is our college. it is in MSR Nagar!"""

# a) Tokenizing
print("\n--- Tokenizing ---")
word_tokens = word_tokenize(text)
sent_tokens = sent_tokenize(text)
```

```
print("Word Tokens:", word_tokens)
print("Sentence Tokens:", sent_tokens)

# b) Filtering Stop Words
print("\n--- Filtering Stop Words ---")
stop_words = set(stopwords.words('english'))
filtered_tokens = [w for w in word_tokens if w.lower() not in
stop_words]
print("Filtered Tokens:", filtered_tokens)

# c) Stemming
print("\n--- Stemming ---")
stemmer = PorterStemmer()
stemmed_tokens = [stemmer.stem(w) for w in filtered_tokens]
print("Stemmed Tokens:", stemmed_tokens)

# d) Part of Speech tagging
print("\n--- POS Tagging ---")
pos_tags = pos_tag(word_tokens)
print(pos_tags)

# e) Chunking (Noun Phrase Example)
print("\n--- Chunking ---")
grammar = "NP: {<DT>?<JJ>*<NN>}"  # Rule: optional determiner +
adjectives + noun
cp = RegexpParser(grammar)
chunked = cp.parse(pos_tags)
print(chunked)
# chunked.draw()  # Uncomment to visualize tree (opens a window)

# f) Named Entity Recognition (NER)
print("\n--- Named Entity Recognition (NER) ---")
ner_tree = ne_chunk(pos_tags)
print(ner_tree)
# ner_tree.draw()  # Uncomment to visualize NER tree
```

```
[nltk_data] Downloading package averaged_perceptron_tagger_eng to
[nltk_data]     /root/nltk_data...
[nltk_data]   Unzipping taggers/averaged_perceptron_tagger_eng.zip.
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt_tab.zip.
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data...
[nltk_data]   Unzipping taggers/averaged_perceptron_tagger.zip.
[nltk_data] Downloading package averaged_perceptron_tagger_eng to
[nltk_data]     /root/nltk_data...
[nltk_data]   Package averaged_perceptron_tagger_eng is already up-to-
[nltk_data]       date!
[nltk_data] Downloading package maxent_ne_chunker to
[nltk_data]     /root/nltk_data...
[nltk_data]   Unzipping chunkers/maxent_ne_chunker.zip.
[nltk_data] Downloading package words to /root/nltk_data...
[nltk_data]   Unzipping corpora/words.zip.

--- Tokenizing ---
Word Tokens: ['MSRCASC', 'is', 'our', 'college', '.', 'it', 'is', 'in', 'MSR', 'Nagar', '!']
Sentence Tokens: ['MSRCASC is our college.', 'it is in MSR Nagar!']

--- Filtering Stop Words ---
Filtered Tokens: ['MSRCASC', 'college', '.', 'MSR', 'Nagar', '!']

--- Stemming ---
Stemmed Tokens: ['msrcasc', 'colleg', '.', 'msr', 'nagar', '!']


--- POS Tagging ---
[('MSRCASC', 'NNP'), ('is', 'VBZ'), ('our', 'PRP$'), ('college', 'NN'), ('.', '.'), ('it', 'PRP'), ('is', 'VBZ'), ('in', 'IN'), ('MSR', 'NNP'), ('Nagar', 'NNP'), ('!', '.')]

--- Chunking ---
(S
  MSRCASC/NNP
  is/VBZ
  our/PRP$
  (NP college/NN)
  ./.
  it/PRP
  is/VBZ
  in/IN
  MSR/NNP
  Nagar/NNP
  !/.)

--- Named Entity Recognition (NER) ---
---------------------------------------------------------------------------
LookupError                               Traceback (most recent call last)
/tmp/ipython-input-2665826628.py in <cell line: 0>()
     53 # f) Named Entity Recognition (NER)
     54 print("\n--- Named Entity Recognition (NER) ---")
---> 55 ner_tree = ne_chunk(pos_tags)
     56 print(ner_tree)
     57 # ner_tree.draw()  # Uncomment to visualize NER tree

                                 3 frames
/usr/local/lib/python3.12/dist-packages/nltk/data.py in find(resource_name, paths)
    577     sep = "*" * 70
    578     resource_not_found = f"\n{sep}\n{msg}\n{sep}\n"
--> 579     raise LookupError(resource_not_found)
    580
    581

LookupError:
**********************************************************************
  Resource maxent_ne_chunker_tab not found.
  Please use the NLTK Downloader to obtain the resource:

  >>> import nltk
  >>> nltk.download('maxent_ne_chunker_tab')

  For more information see: https://www.nltk.org/data.html

  Attempted to load chunkers/maxent_ne_chunker_tab/english_ace_multiclass/

  Searched in:
    - '/root/nltk_data'
    - '/usr/nltk_data'
    - '/usr/share/nltk_data'
    - '/usr/lib/nltk_data'
    - '/usr/share/nltk_data'
    - '/usr/local/share/nltk_data'
    - '/usr/lib/nltk_data'
    - '/usr/local/lib/nltk_data'
**********************************************************************
```

11) Write a program that uses Neural networks for image classification using Keras Iris dataset

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.utils import to_categorical

# Load the Iris dataset
from sklearn.datasets import load_iris
data = load_iris()

# Features and labels
X = data['data']
y = data['target']

# Preprocess the data
scaler = StandardScaler()
X = scaler.fit_transform(X)  # Standardizing the features

# Convert labels to one-hot encoding
encoder = LabelEncoder()
y_encoded = encoder.fit_transform(y)
y_categorical = to_categorical(y_encoded)

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y_categorical,
test_size=0.2, random_state=42)

# Build a Neural Network model using Keras
model = Sequential()
model.add(Dense(64, input_shape=(X_train.shape[1],),
activation='relu'))  # Input layer
model.add(Dense(32, activation='relu'))  # Hidden layer
model.add(Dense(3, activation='softmax'))  # Output layer (3 classes)

# Compile the model
model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])
```

```python
# Train the model
history = model.fit(X_train, y_train, epochs=50, batch_size=5,
validation_split=0.2)

# Evaluate the model
loss, accuracy = model.evaluate(X_test, y_test)
print(f"Test Loss: {loss}")
print(f"Test Accuracy: {accuracy}")

# Predicting on test set
y_pred = model.predict(X_test)
predicted_classes = np.argmax(y_pred, axis=1)
true_classes = np.argmax(y_test, axis=1)

# Display the predictions and true values
print(f"Predicted classes: {predicted_classes}")
print(f"True classes: {true_classes}")
```

```
Epoch 49/50
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 15ms/step - accuracy: 0.9479 - loss: 0.1069 - val_accuracy: 0.9167 - val_loss: 0.2794
Epoch 50/50
20/20 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.9447 - loss: 0.1112 - val_accuracy: 0.9583 - val_loss: 0.2465
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 63ms/step - accuracy: 1.0000 - loss: 0.0371
Test Loss: 0.037054650485515594
Test Accuracy: 1.0
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 112ms/step
Predicted classes: [1 0 2 1 1 0 1 2 1 1 2 0 0 0 0 1 2 1 1 2 0 2 0 2 2 2 2 2 0 0]
True classes: [1 0 2 1 1 0 1 2 1 1 2 0 0 0 0 1 2 1 1 2 0 2 0 2 2 2 2 2 0 0]
```