AI Learning Game:

Classification: An animal data set was acquired that contained a picture of a variety of animals (Dogs, Cats, Chicken, Horse, etc). the images were preprocessed in python using a library called Pillow, the preprocessing includes

```
from PIL import Image, ImageFilter
from random import randint
import os
import uuid

dogphotos = os.listdir(r"C:\\Users\\yasse\Downloads\\cat")
folderNumber = 0
for photo in dogphotos:
    im = Image.open(r"C:\\Users\\yasse\Downloads\\cat\\"+ photo)
    Foldername = str(uuid.uuid4())
    os.mkdir("cat/"+Foldername)
    width, height = im.size
    increment_top = height/3
    increment_side = width/3
    top = 0
    botom = increment_top
    photonumber =0

    for rows in range(3):
        left = 0
        right = increment_side
        for column in range(3):
            im1 = im.crop((left, top, right, botom))
            newimage= im1.rotate(randint(0,360))
            blurImage = newimage.filter(ImageFilter.BLUR)
            blurImage.save(f"dog/{Foldername}/rotatedDog{photonumber}.png")
            left += increment_side
            right += increment_side
            photonumber +=1
        botom += increment_top
        top += increment_top
```

this code imports the image for the location on my computer, segments it into 9 sections and applies a random rotation and blur masking layer onto each segmented image it then exports it and stores it in a random directory at the location of my choosing.

From here images were imported as assets into the unity asset manager and converted into sprites.

A Unity 2D environment was created along with the games UI layout which included buttons and RawImage gameobjects to render the sprites. The most important aspect of the scene is the manage game object. This is what contains all the scripts that have the main game loop. The NextButton.cs controls what happens when the “Next” button is pressed.

The order of events proceeds like this:

1. Sets all the button colors to white
2. Selects a random Animal from the Dog, cat, Butterfly, Chicken, Elephant, Horse
3. selects the appropriate button to be the correct one and the rest to be wrong
4. selects a random image folder inside of the animal folder selected
5. Takes the images and renders them in the game View.
6. Once an animal button is clicked it informs the user if they made the correct choice by turning green, or red otherwise.

Each button contains an Answrs script that simply contains a color that that the button will turn into when clicked in step 6 from above. In step 3 the NextButton script will edit this color value turning it either green or red.

Linear Classification:

The CreateLine.cs contains one main function called Update, this is run once every frame inorder to get the precise moment the user clicks the mouse on the game view. It retrieves the location of the click in world space and stores the value in the StartMousePos which contains the X, Y coordinates. It then waits for the user to release the mouse button and creates a new game object to which a LineRenderer Component is immediately added, this is what creates the line that’s seen on screen. Once the user lifts the mouse button the world space coordinates and once again recorded and the line is drawn but setting the 0 and 1 positions of the line renderer components to the start and end coordinates.

Since a new game object is drawn each time we can draw many lines on screen without the previous lines disappearing.

The GeneratePoints.cs will simply generate points on the game view within the bounds on the graph in batches of 4 up to a total of 20. This method is linked to the main camera and is called every time the Get Points button is pressed