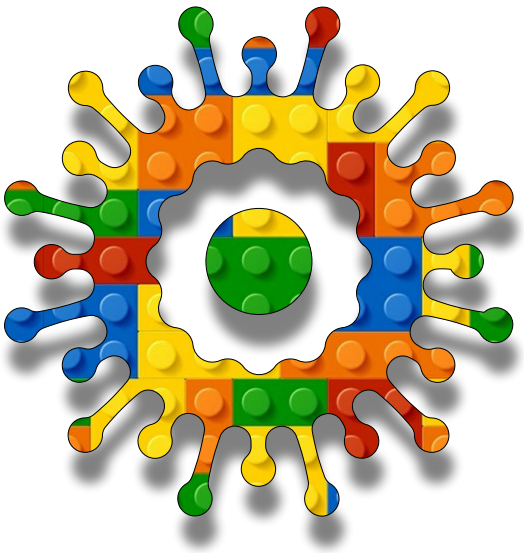
Algorithmic Intelligence Module A Unit #13 save and load



Module A Unit #13 save and load

Introduction to save and load (data and model)

We are going to save to, and then load the data points from, a JSON file rather than generating those data points each time. This is so that we can then see the impact of changing the hyperparameters without changing the dataset every time.

There are four parts to this unit:
Part 1️⃣ saving the data
Part 2️⃣ loading the saved data
Part 3️⃣ saving the model
Part 4️⃣ loading the saved model

part #1
saving the data

Saving the data

We have used new synthetic data generated each time we run the code. It was good for an initial demonstration of machine learning, but if we want to see how the hyperparameters impact the training, we need a fixed dataset, not one that changes every time we run the model.

In this unit, we will save some data and then load it so that we always use the same data when experimenting with our model. You can use this approach for any synthetic dataset that you might generate.

Sketch A13.1 this is our starting sketch

We have the data points drawn and a `console.log()` of the data so we can see what's inside the `data` array. Notice we have used the `abs()` function inside the `floor()` function. This removes the negative values and keeps everything on the canvas.

```
const number = 30
const spread = 30
let data = []

function setup()
{
  createCanvas(400, 400)
  background(220)
  trainingData()
}

function trainingData()
{
  for (let i = 0; i < width; i += 10)
  {
    for (let j = 0; j < number; j++)
    {
      fill(0, 0, 255)
      noStroke()
      let x = floor(abs(i + random(-spread, spread)))
      let y = floor(abs(height - (i + random(-spread, spread))))
      data.push(x, y)
      circle(x, y, 5)
    }
  }
  console.log(data)
}
```

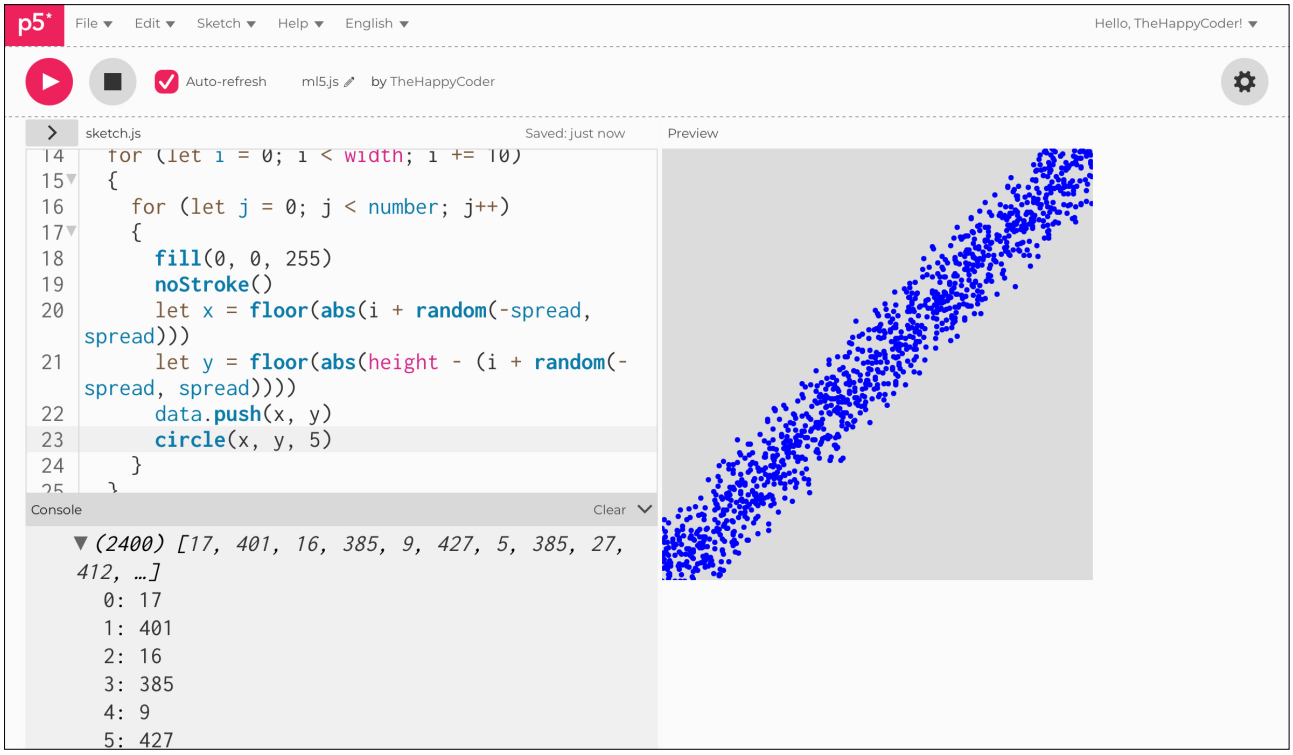
🗒️ Notes

Recap: The `data` is an empty array that is filled with the `x` and `y` values from the nested loop using the `push()` function. One of the benefits of the `console.log()` function is that we can see the size of the data array. In this case, it is `2400` elements, or `1200` pairs of co-ordinates.

🌻 Challenge

Try `console.log(data.length)`

Figure A13.1



Sketch A13.2 we need a button to press

We are going to add a button so that we can click on it and save a JSON file with all the data in it. First, let's create the button.

```
const number = 30
const spread = 30
let data = []
let button

function setup()
{
  createCanvas(400, 400)
  background(220)
  button = createButton('save data')
  button.style('font-size', '20px')
  button.style('background-color', color('darkred'))
  button.style('color', color('yellow'))
  trainingData()
}

function trainingData()
{
  for (let i = 0; i < width; i += 10)
  {
    for (let j = 0; j < number; j++)
    {
      fill(0, 0, 255)
      noStroke()
      let x = floor(abs(i + random(-spread, spread)))
      let y = floor(abs(height - (i + random(-spread, spread))))
      data.push(x, y)
      circle(x, y, 5)
    }
  }
  console.log(data)
}
```

🗒️ Notes

We can create a simple button, but I have included some styling to make it a bit more interesting. At this point, however, nothing happens when you click the button; that part is yet to come.

🌻 Challenge

Try different colours and sizes of font

🛠️ Code Explanation

`let button`	The button variable
`button = createButton('save data')`	Creating the button and giving it some text
`button.style('font-size', '20px')`	Increasing the size of the font
`button.style('background-color', color('darkred'))`	Give the button a background colour
`button.style('color', color('yellow'))`	Give the text some colour

Sketch A13.3 adding a function

When we press the button, we want it to do something. We use a `mousePressed()` function, which leads to the `saveData()` function when the button is pressed. The data is then saved as a `JSON` file called `trainingData.json` using the `save()` function.

! You may encounter a brown error message. For the moment it is not critical.

```
const number = 30
const spread = 30
let data = []
let button

function setup()
{
  createCanvas(400, 400)
  background(220)
  button = createButton('save data')
  button.style('font-size', '20px')
  button.style('background-color', color('darkred'))
  button.style('color', color('yellow'))
  trainingData()
}

function trainingData()
{
  for (let i = 0; i < width; i += 10)
  {
    for (let j = 0; j < number; j++)
    {
      fill(0, 0, 255)
      noStroke()
      let x = floor(abs(i + random(-spread, spread)))
      let y = floor(abs(height - (i + random(-spread, spread))))
      data.push(x, y)
      circle(x, y, 5)
    }
  }
  button.mousePressed(saveData)
```

```
  console.log(data)
}
```

```
function saveData()
{
  console.log('button pressed')
  save(data, 'trainingData.json', true)
}
```

🗒️ Notes

When you click on the button, you should get a request to download the file. It will be different for different machines and browsers. The reason for the boolean true in the file arguments is whether to trim unneeded whitespace. You can open the file to see what's inside it. In some instances, it might just download without a prompt.

🛠️ Code Explanation

`button.mousePressed(saveData)`	When the button is pressed it activates the function saveData()
`save(data, 'trainingData.json', true)`	The save() function needs the first argument to be the data, the second is the name of the file to be saved and the third is a boolean optimiser.

Figure A13.3a

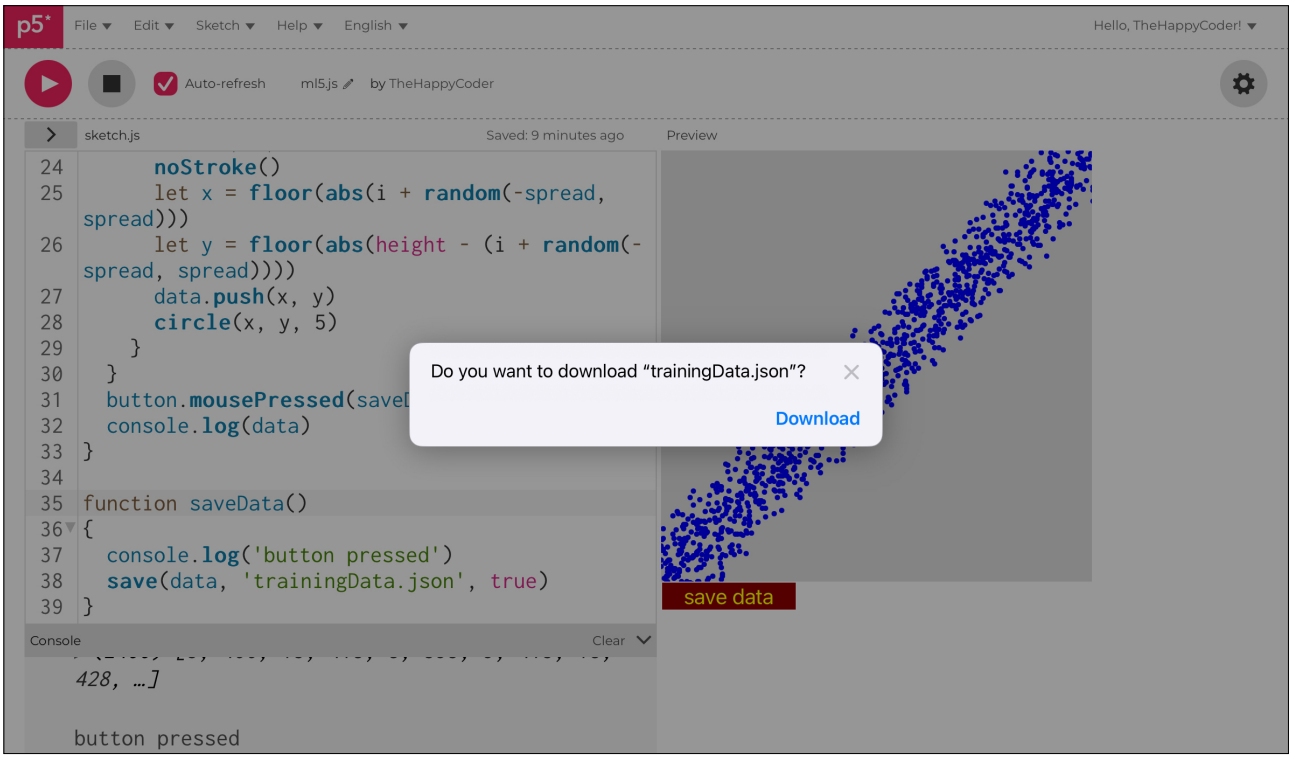


Figure A13.3b having a quick peak inside
the trainingData.json file

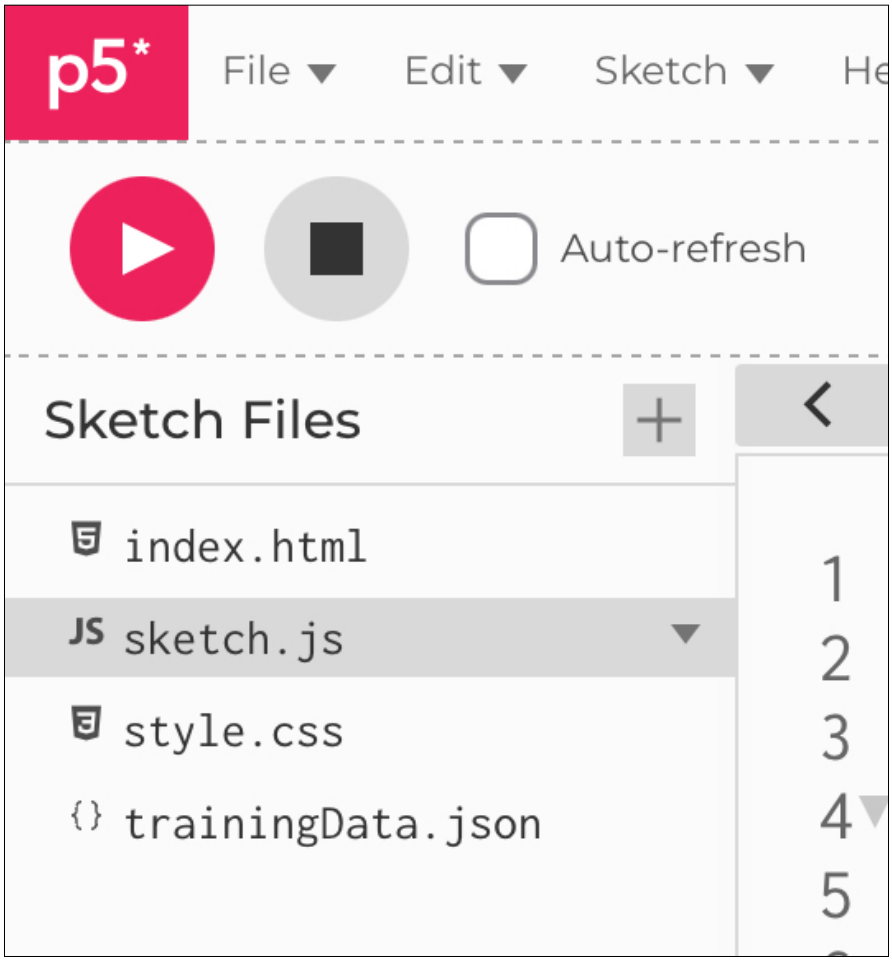
[4,422,14,383,20,390,0,392,20,382,4,415,7,396,15,405,6,411,6,425,20,420,22,394,6,397,26,419,19,424,10,415,3,406,3,424,12,381,13,373,15,384,17,395,29,396,26,376,5,371,24,403,24,422,22,408,20,411,26,383,19,400,4,379,10,379,3,375,28,403,38,377,13,399,1,369,35,414,1,369,18,376,2,368,17,388,22,417,18,398,8,390,0,396,11,401,2,370,13,360,37,375,33,384,20,360,18,407,29,370,10,409,1,382,9,400,7,381,0,416,5,394,1,360,1,373,21,384,48,368,32,389,9,380,44,361,9,359,15,395,39,369,2,370,22,408,15,377,10,370,7,350,5,394,28,370,21,383,38,351,2,376,11,403,13,376,41,383,44,365,36,374,42,363,6,408,7,365,14,398,16,347,32,369,17,392,30,384,35,379,12,355,25,354,22,395,53,342,11,348,9,372,18,385,20,347,40,368,37,391,20,399,34,383,17,375,50,362,20,341,6,393,53,371,5,383,0,374,36,371,53,361,29,364,59,354,32,341,8,372,23,339,49,346,52,357,19,356,29,347,44,344,35,368,57,362,55,364,20,361,42,388,34,368,55,349,42,349,33,366,40,336,23,384,44,388,35,369,25,355,64,356,47,379,27,365,22,362,35,382,56,361,40,388,30,334,45,368,63,363,42,330,41,375,65,361,75,379,33,334,59,351,72,332,41,365,49,351,47,368,48,328,67,335,40,347,41,346,38,355,46,352,59,357,24,341,69,332,76,331,71,352,35,370,67,358,34,343,74,372,33,351,60,341,31,323,53,346,34,357,69,347,56,338,35,340,80,360,39,351,57,358,75,367,60,363,37,324,52,315,38,329,73,353,59,328,41,334,43,357,52,320,43,356,40,318,32,345,79,344,78,353,31,364,51,340,75,329,42,324,38,356,40,346,60,333,77,330,51,334,58,354,80,335,48,333,97,348,94,332,79,359,66,333,73,349,98,308,66,303,90,350,49,338,77,337,62,312,90,355,41,350,53,307,81,328,70,353,48,303,82,357,48,337,44,349,83,359,43,315,62,340,99,334,55,335,59,347,81,316,94,323,98,335,50,291,92,330,53,293,94,338,80,335,90,348,56,305,109,304,100,336,76,344,63,315,73,342,73,331,54,332,53,340,62,298,107,304,68,348,59,315,61,292,63,306,86,317,64,337,90,297,95,324,83,333,64,316,80,295,108,326,114,294,66,325,89,319,99,291,82,318,64,335,97,324,60,321,112,299,106,296,64,298,108,308,62,316,104,323,116,315,97,280,109,305,111,332,71,298,73,280,83,305,115,329,66,300,86,289,92,324,105,308,88,296,109,297,119,309,79,302,120,311,101,284,116,275,127,314,81,273,123,270,114,311,126,308,83,277,110,276,91,292,85,314,98,297,96,274,106,283,95,281,115,313,127,319,117,277,116,329,119,300,118,298,106,291,113,303,129,281,128,278,101,275,117,303,90,290,122,316,112,311,122,267,116,292,106,268,84,297,124,310,136,261,95,280,104,308,121,274,90,310,100,279,107,315,109,286,128,301,86,309,106,311,115,270,132,269,102,293,127,269,138,278,110,294,103,295,81,261,126,283,80,286,86,315,119,310,90,270,124,260,125,288,106,291,104,302,126,268,137,293,141,286,111,287,95,283,90,309,94,258,135,287,117,259,146,301,114,258,102,298,119,305,113,273,115,299,119,258,100,304,134,250,104,304,132,256,104,284,138,270,127,254,110,255,133,254,109,285,155,258,114,266,124,295,136,289,129,250,108,274,131,270,140,284,103,242,142,287,136,271,151,276,100,271,149,267,157,291,121,276,144,270,157,271,144,252,149,261,106,248,155,252,126,292,155,282,159,286,106,291,110,277,107,276,143,283,128,268,163,286,168,249,153,235,167,275,130,235,127,262,111,250,123,286,111,254,145,257,134,248,127,230,128,275,119,288,162,277,161,281,117,246,140,252,127,283,113,289,155,285,150,247,125,247,133,244,161,259,124,281,110,289,160,281,137,236,153,229,164,225,128,223,128,261,155,221,147,258,160,220,158,223,159,263,137,254,120,261,136,270,136,268,174,226,148,249,140,257,140,275,132,235,154,255,173,255,138,228,161,253,166,264,157,259,158,247,138,273,149,249,150,230,173,238,173,258,188,268,133,261,166,233,173,242,174,248,130,230,182,227,135,254,183,252,173,266,165,220,130,230,187,243,189,218,183,237,175,266,172,267,148,236,158,263,140,210,140,249,188,268,150,268,137,230,150,258,184,233,173,249,177,216,149,257,169,265,153,224,154,236,148,231,166,258,153,245,142,204,194,255,149,216,173,238,154,256,180,242,196,237,181,248,163,200,174,206,158,209,180,215,198,243,162,207,162,223,179,253,182,239,167,238,168,248,152,222,161,244,183,227,169,226,150,255,140,240,178,214,205,202,179,244,156,243,163,223,180,222,158,213,181,194,184,213,175,205,160,199,171,202,201,194,201,245,189,225,165,229,175,246,151,195,197,197,179,238,174,192,185,228,182,225,188,204,159,199,164,243,183,221,195,208,158,235,188,228,202,210,216,189,185,190,205,190,188,208,201,186,172,192,160,234,179,201,208,216,196,181,204,225,187,196,203,206,169,188,206,208,196,205,183,193,162,226,176,190,213,219,213,203,168,202,205,232,183,225,173,212,209,235,169,212,176,231,189,195,207,205,181,227,174,177,191,193,203,180,211,211,179,199,204,205,222,229,205,215,184,204,197,184,209,206,189,218,176,213,225,172,172,194,202,209,171,185,217,219,185,175,215,211,220,178,196,198,215,183,224,197,179,222,192,218,227,204,197,214,212,216,208,182,234,174,188,167,192,163,212,165,198,174,214,209,189,192,235,176,207,182,192,212,214,167,214,207,214,167,212,179,180,211,225,162,202,201,202,170,187,213,229,209,187,200,205,205,226,167,216,186,223,172,193,180,214,188,223,163,236,151,242,166,207,184,202,156,202,201,211,158,213,183,218,182,202,182,247,162,209,193,191,207,242,202,215,157,232,156,242,190,234,158,190,202,209,195,216,187,243,191,219,200,205,169,221,184,205,184,209,189,197,162,239,165,221,198,238,203,227,156,216,150,214,157,208,160,205,158,209,154,248,155,247,165,224,173,235,184,212,169,201,154,210,192,248,194,222,173,210,164,216,170,200,153,200,170,216,145,258,180,243,165,236,197,212,183,230,185,215,169,221,196,246,190,232,181,217,185,225,137,243,152,254,165,214,155,264,150,211,146,239,151,243,138,237,153,256,154,239,144,226,182,241,171,224,151,256,150,247,167,232,152,263,174,214,139,235,165,251,146,227,154,250,136,237,139,247,181,256,186,242,139,252,144,243,150,258,130,267,139,239,165,241,160,247,163,257,163,238,152,262,128,278,124,220,124,273,124,222,133,258,126,232,158,273,163,247,147,240,160,238,150,242,178,262,160,254,123,259,130,223,179,241,177,225,169,243,174,255,135,239,146,223,130,235,149,257,160,278,143,230,133,274,115,232,125,244,127,240,122,230,136,279,146,249,139,276,148,274,155,254,141,277,151,288,110,236,140,255,135,239,147,285,138,240,154,267,161,230,143,271,130,242,110,254,167,287,126,242,156,256,163,231,159,232,137,232,116,248,150,296,112,295,118,283,102,263,108,248,100,293,110,263,129,294,158,294,155,288,150,287,118,276,133,289,156,296,135,298,114,282,110,258,107,290,155,266,107,288,146,274,111,298,112,248,136,272,125,288,112,272,154,271,155,291,124,247,152,278,99,301,130,293,101,261,96,282,148,271,106,262,113,260,113,271,142,251,111,254,111,301,146,281,107,254,145,304,105,271,148,290,108,264,131,260,142,273,110,276,98,309,101,297,124,261,100,264,109,293,132,299,126,307,131,279,145,309,132,317,128,272,135,315,135,295,133,305,136,276,134,310,90,283,112,318,111,264,92,313,122,282,134,281,122,304,119,292,126,317,83,265,109,268,110,276,124,273,117,279,81,266,108,299,80,311,101,303,91,288,103,260,131,264,111,288,88,278,95,299,107,319,74,317,98,293,98,298,106,312,121,316,104,322,108,270,77,284,91,325,112,283,75,283,83,275,110,298,111,304,109,280,119,297,104,283,95,275,110,292,103,299,74,310,129,307,108,317,111,313,111,270,90,284,76,299,91,281,93,309,82,301,116,312,119,291,116,336,94,321,105,314,97,282,90,336,93,282,97,290,106,299,108,295,81,319,92,320,98,319,118,323,88,332,91,331,115,312,108,297,114,293,74,306,89,312,101,320,87,306,97,296,78,290,83,298,118,286,117,312,64,337,75,316,85,326,78,314,71,336,109,339,102,303,90,301,57,303,106,293,81,335,77,335,83,310,69,329,74,312,68,341,87,331,59,311,89,329,50,299,52,311,91,317,78,316,50,3

part #2
loading the data

Loading the data

We need to upload the file to the web editor before we can access it in the sketch. The steps are detailed in the unit called: A Quick Exploration of p5.js. If you are unfamiliar with uploading files, images, videos, or music, then please refer to that section of the unit; it is not difficult. Needless to say, this is what you should have if you open the sidebar of files.

Figure 1: loading the data



Sketch A13.4 loading the file

❗ Starting a completely new sketch.

Firstly, we need to load the data. The `loadJSON()` function does exactly that. We have to do this asynchronously. This is why we have the keywords `async` and `await`. It is a better way of preloading the data. In version 1.11.x of p5.js we used the `preload()` function, but that will not work with version 2.1.x. Async and await will work with version 1.11.x however.

```
let data

async function setup()
{
  data = await loadJSON('trainingData.json')
  createCanvas(400, 400)
  console.log(data)
}

function draw()
{
  background(220)
}
```

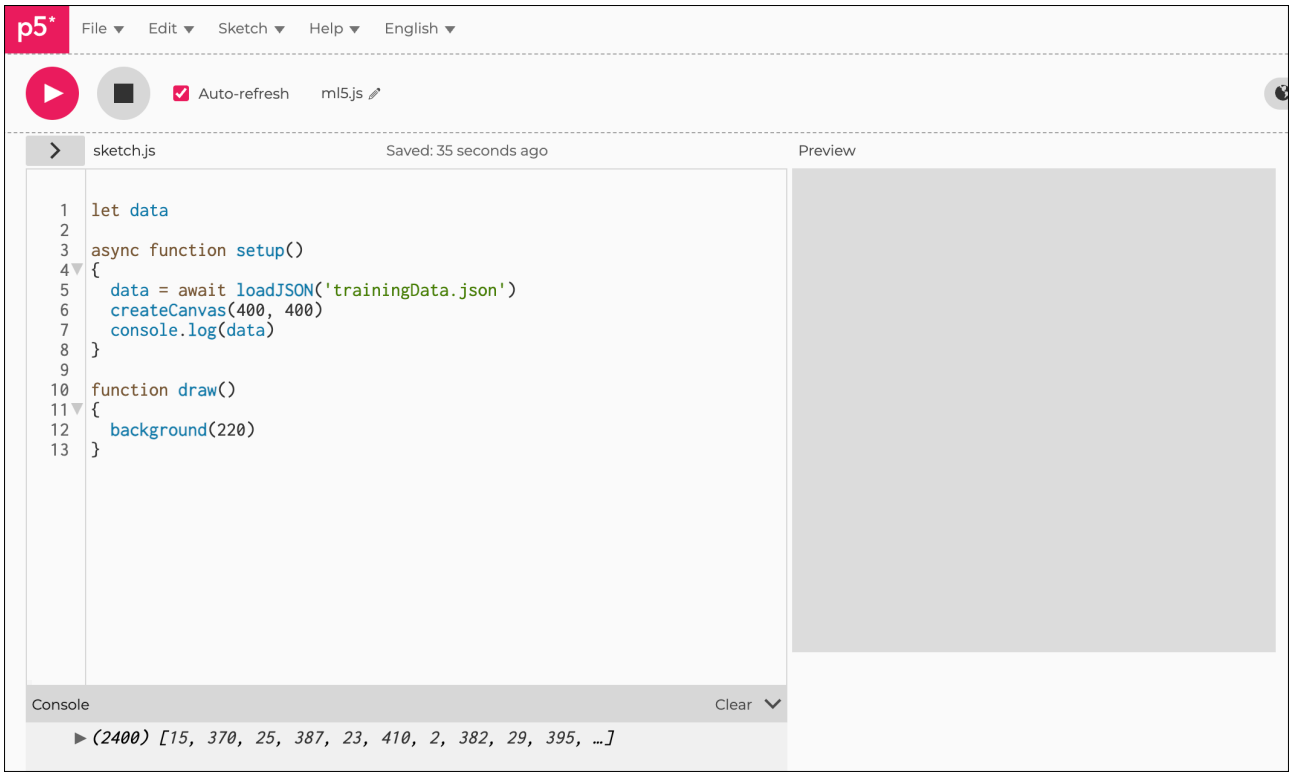
🗒️ Notes

We `console.log(data)` just to make sure that it has loaded. You should get an object array in the console, but nothing on the canvas (yet).

🛠️ Code Explanation

`async function setup()`	The setup() function needs to complete a task before anything else happens
`data = await loadJSON('trainingData.json')`	In setup() nothing else will happen until all the data is loaded

Figure A13.4


p5*
File
Edit
Sketch
Help
English
Auto-refresh
ml5.js
sketch.js
Saved: 35 seconds ago
Preview
let data
async function setup()
{
data = await loadJSON('trainingData.json')
createCanvas(400, 400)
console.log(data)
}
function draw()
{
background(220)
}
Console
Clear
▶ (2400) [15, 370, 25, 387, 23, 410, 2, 382, 29, 395, …]

Sketch A13.5 checking the size

We have a small loop that goes through all the data points in the `data` array. We have a counter called `num` which adds `1` each time for each data point. When it has gone through the whole dataset (`data` array), we call `noLoop()` which stops the `draw()` function looping. We console log to see what `num` finally comes up with; this will give us the size of the `data` array.

❗ Remove the console log in the `setup()` function; we don't need it anymore.

```
let data
let num = 0

async function setup()
{
  data = await loadJSON('trainingData.json')
  createCanvas(400, 400)
}

function draw()
{
  background(220)
  for (let datapoints in data)
  {
    num++
  }
  noLoop()
  console.log(num)
}
```

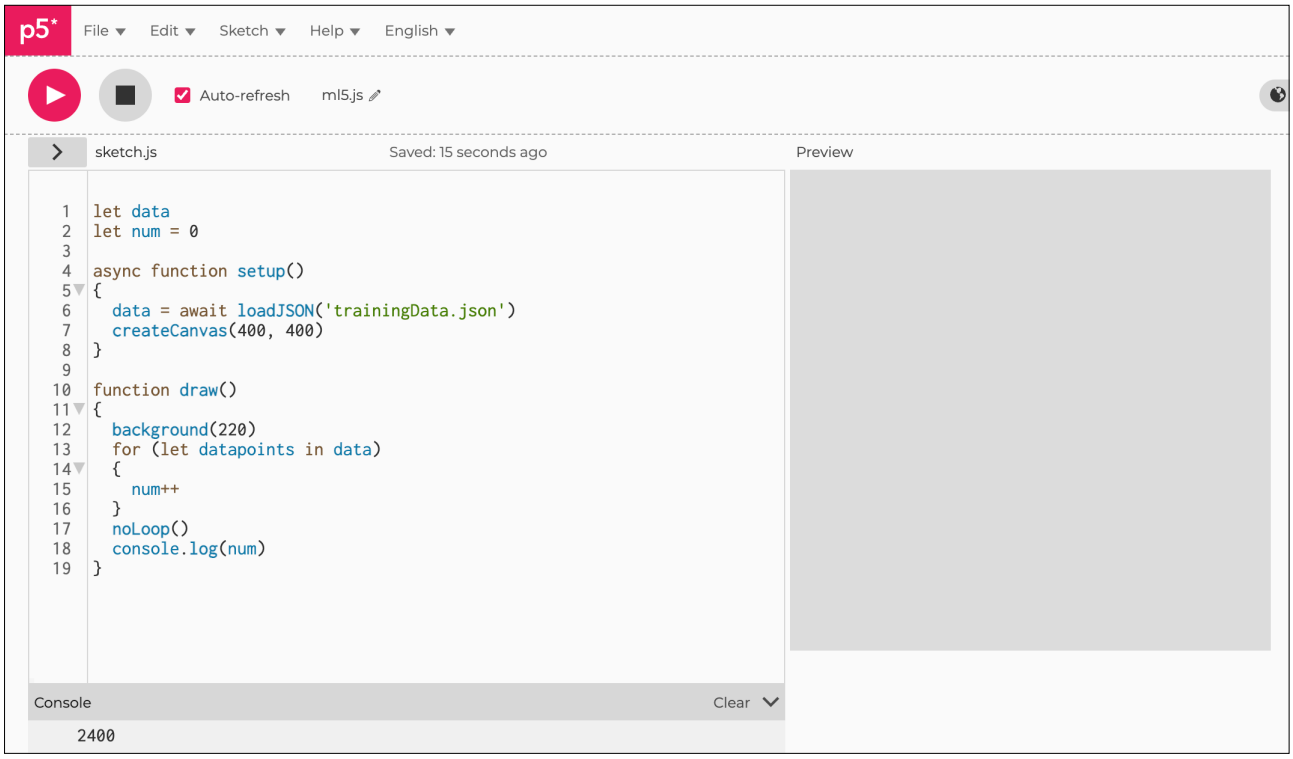
🗒️ Notes

As expected, the size of the array is `2400` data points. This means we can use that number to draw all the data points.

🛠️ Code Explanation

`let num = 0`	Declare and initialise a counter variable
`noLoop()`	Stops a loop if there is one running
`for (let datapoints in data)`	A for loop that iterates through all the elements in an array and copies them to a new array.

Figure A13.5


p5*
File
Edit
Sketch
Help
English
Auto-refresh
ml5.js
sketch.js
Saved: 15 seconds ago
Preview
let data
let num = 0
async function setup()
{
data = await loadJSON('trainingData.json')
createCanvas(400, 400)
}
function draw()
{
background(220)
for (let datapoints in data)
{
num++
}
noLoop()
console.log(num)
}
Console
Clear
2400

Sketch A13.6 drawing the points

We jump every two elements in the array, as they are `pairs` of elements (`x, y`). Hence, we use the index for `x` as `data[i]` and for `y` as `data[i+1]`.

! Remove the `console.log(num)`, we don't need it anymore.

```
let data
let num = 0

async function setup()
{
  data = await loadJSON('trainingData.json')
  createCanvas(400, 400)
}

function draw()
{
  background(220)
  for (let datapoints in data)
  {
    num++
  }
  noLoop()
  for (let i = 0; i < num; i += 2)
  {
    fill(0, 0, 255)
    noStroke()
    circle(data[i], data[i + 1], 5)
  }
}
```

Notes

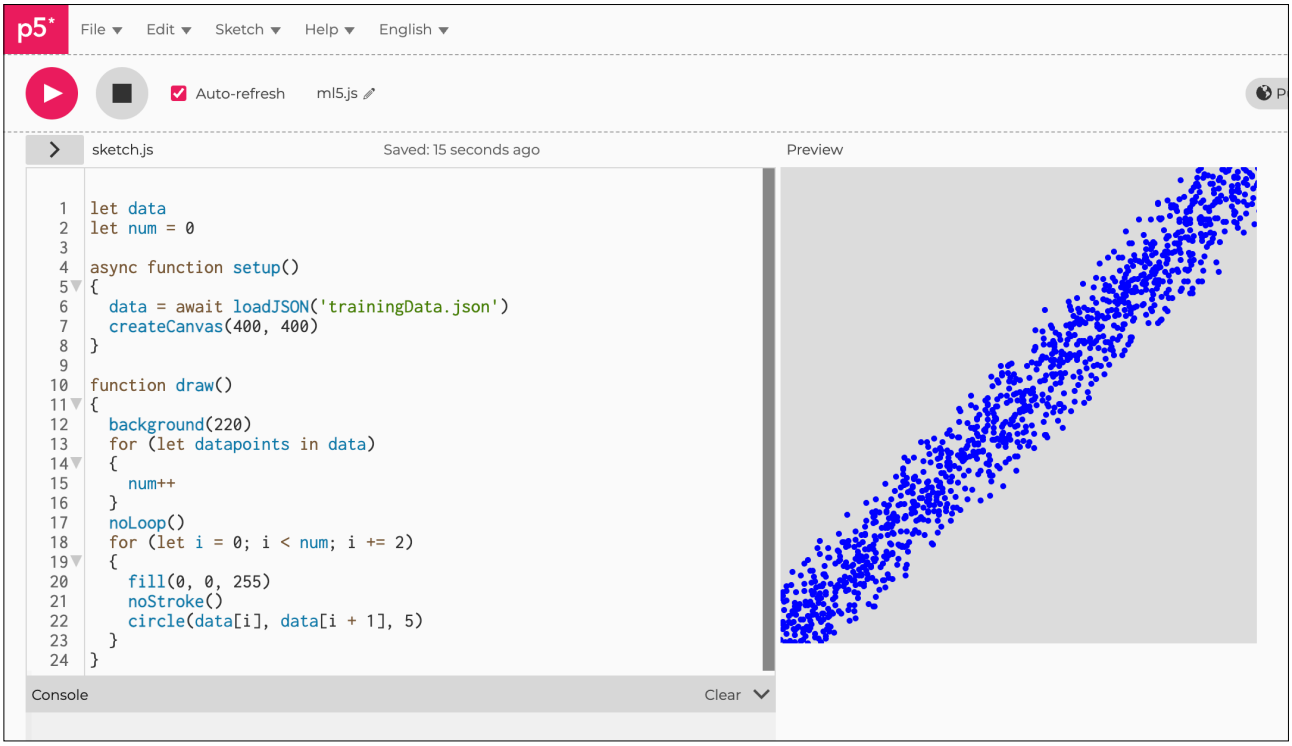
There are `2400` data points, but only `1200` pairs of data points. We get the same result as when we saved the data in the first instance.

! Keep the file in the sketch for later; we will use it with a model.

🛠️ Code Explanation

`for (let i = 0; i < num; i += 2)`	We work through the size of the array by placing the variable index[i] every second element i += 2
`circle(data[i], data[i + 1], 5)`	This is where i is the x index data point and i + 1 is the y data point

Figure A13.6

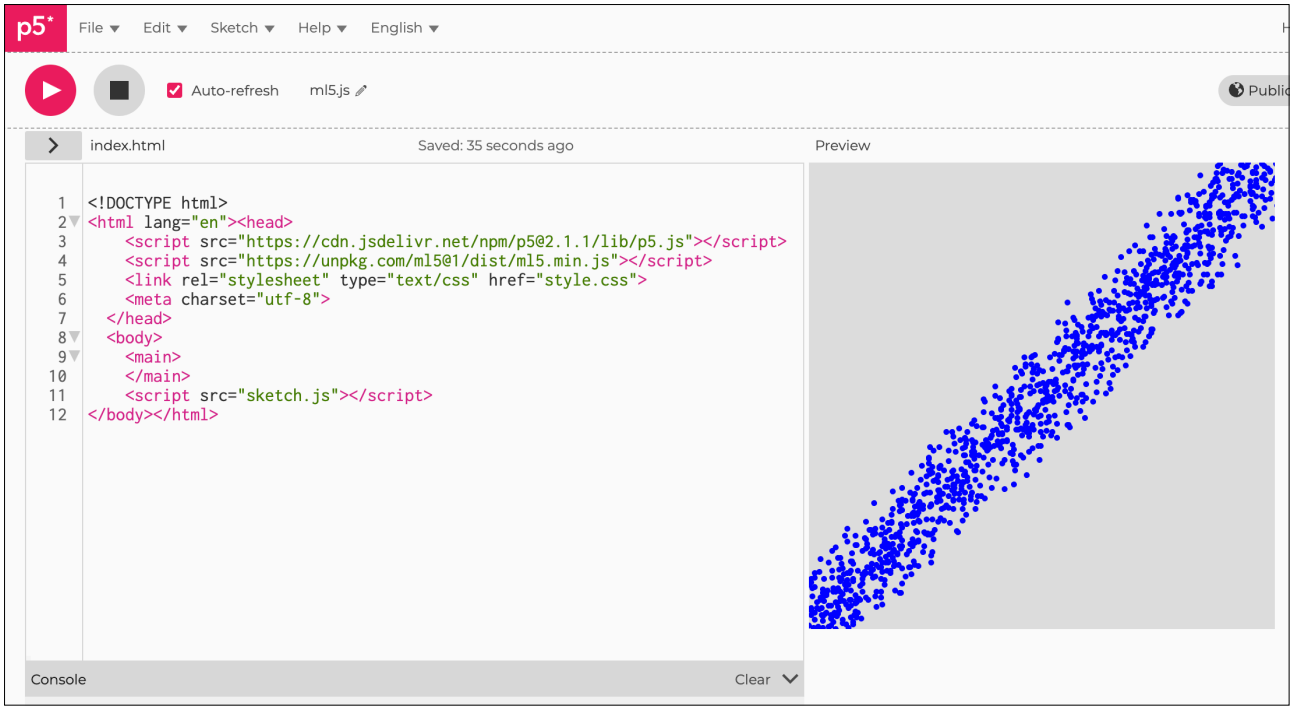


part #3
saving the model

The index.html and training data json files

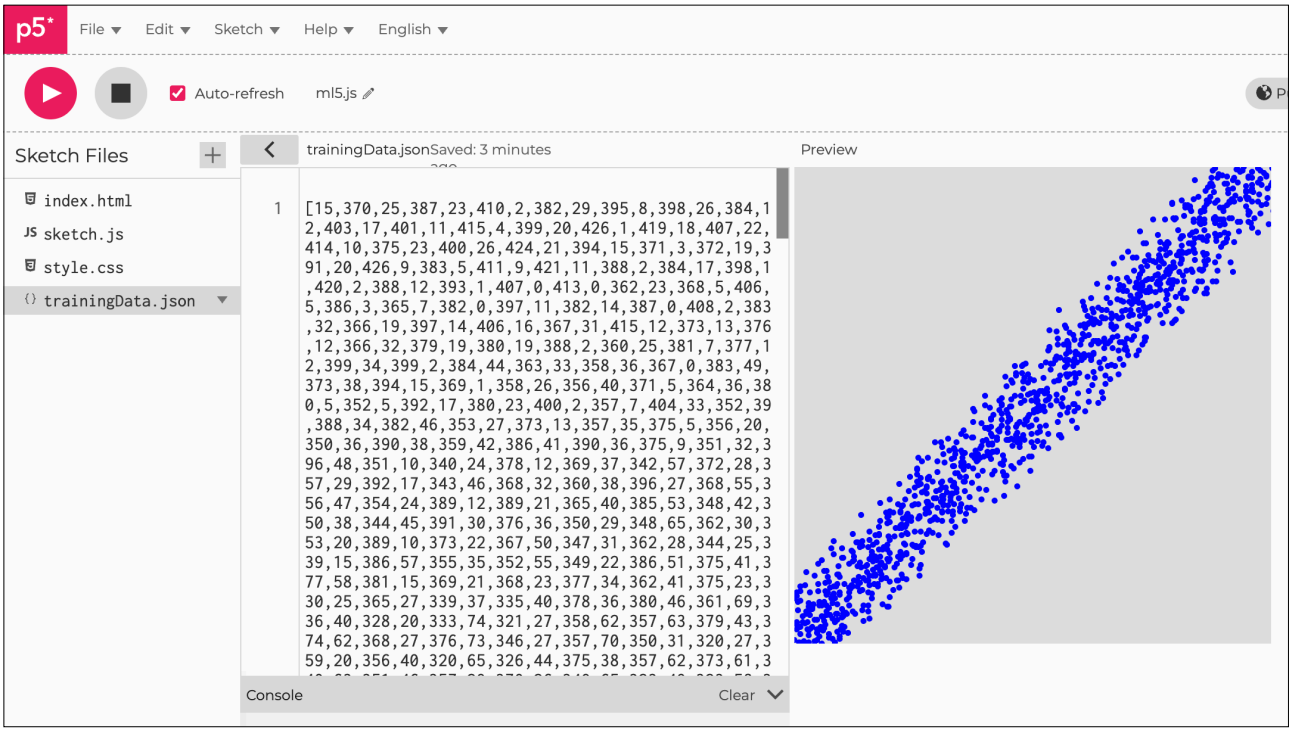
Make sure you have the ml5.js line of code in the index.html file.

Figure 2: ml5.js in the index.html file



Also, make sure you have uploaded the file `trainingData.json`.

Figure 3: Training data file



Sketch A13.7 we start again

This is our starting sketch; it isn't exactly the same as where we left off because we will be making major changes and additions. We won't be generating the data but loading it instead. So you will need to write the code below as is, then we can add in everything else as we go along.

```
let nn
let counter = 0
let data
const number = 30
const spread = 30
const options = {
  task: 'regression',
  debug: false
}

function setup()
{
  createCanvas(400, 400)
  ml5.setBackend("webgl")
  nn = ml5.neuralNetwork(options)
  trainingData()
  nn.normalizeData()
  const trainingOptions = {
    batchSize: 512
  }
  nn.train(trainingOptions, finishedTraining)
}

function trainingData()
{
  background(220)
}

function finishedTraining()
{
  if (counter < 400)
  {
```

```
    nn.predict([counter], gotResults)
  }
}

function gotResults(results)
{
  let prediction = results[0]
  let x = counter
  let y = prediction.value
  stroke(255, 0, 0)
  strokeWeight(5)
  point(x, y)
  counter++
  finishedTraining()
}
```

🗒️ Notes

❗ Be aware, if you run the sketch at this stage, you will get an error.

Sketch A13.8 loading the data

We now use the `preload()` function to load the data into the sketch. We use the `for()` loop to get the number of data points (`num`) from the `data` array.

```
let nn
let counter = 0
let data
let num = 0
const number = 30
const spread = 30
const options = {
  task: 'regression',
  debug: false
}

async function setup()
{
  data = await loadJSON('trainingData.json')
  createCanvas(400, 400)
  ml5.setBackend("webgl")
  nn = ml5.neuralNetwork(options)
  trainingData()
  nn.normalizeData()
  const trainingOptions = {
    batchSize: 512
  }
  nn.train(trainingOptions, finishedTraining)
}

function trainingData()
{
  background(220)
  for (let datapoints in data)
  {
    num++
  }
}
```

```
function finishedTraining()
{
  if (counter < 400)
  {
    nn.predict([counter], gotResults)
  }
}

function gotResults(results)
{
  let prediction = results[0]
  let x = counter
  let y = prediction.value
  stroke(255, 0, 0)
  strokeWeight(5)
  point(x, y)
  counter++
  finishedTraining()
}
```

🗒️ Notes

This gives us the data points. Now we want to add those data points to the neural network (and draw them).

Sketch A13.9 adding the data and drawing

We will now add the data to the neural network to predict the line and also draw the data points (circles).

```
let nn
let counter = 0
let data
let num = 0
const number = 30
const spread = 30
const options = {
  task: 'regression',
  debug: false
}

async function setup()
{
  data = await loadJSON('trainingData.json')
  createCanvas(400, 400)
  ml5.setBackend("webgl")
  nn = ml5.neuralNetwork(options)
  trainingData()
  nn.normalizeData()
  const trainingOptions = {
    batchSize: 512
  }
  nn.train(trainingOptions, finishedTraining)
}

function trainingData()
{
  background(220)
  for (let datapoints in data)
  {
    num++
  }
  for (let i = 0; i < num; i += 2)
```

```
  {
    fill(0, 0, 255)
    noStroke()
    circle(data[i], data[i + 1], 5)
    nn.addData([data[i]], [data[i + 1]])
  }
}

function finishedTraining()
{
  if (counter < 400)
  {
    nn.predict([counter], gotResults)
  }
}

function gotResults(results)
{
  let prediction = results[0]
  let x = counter
  let y = prediction.value
  stroke(255, 0, 0)
  strokeWeight(5)
  point(x, y)
  counter++
  finishedTraining()
}
```

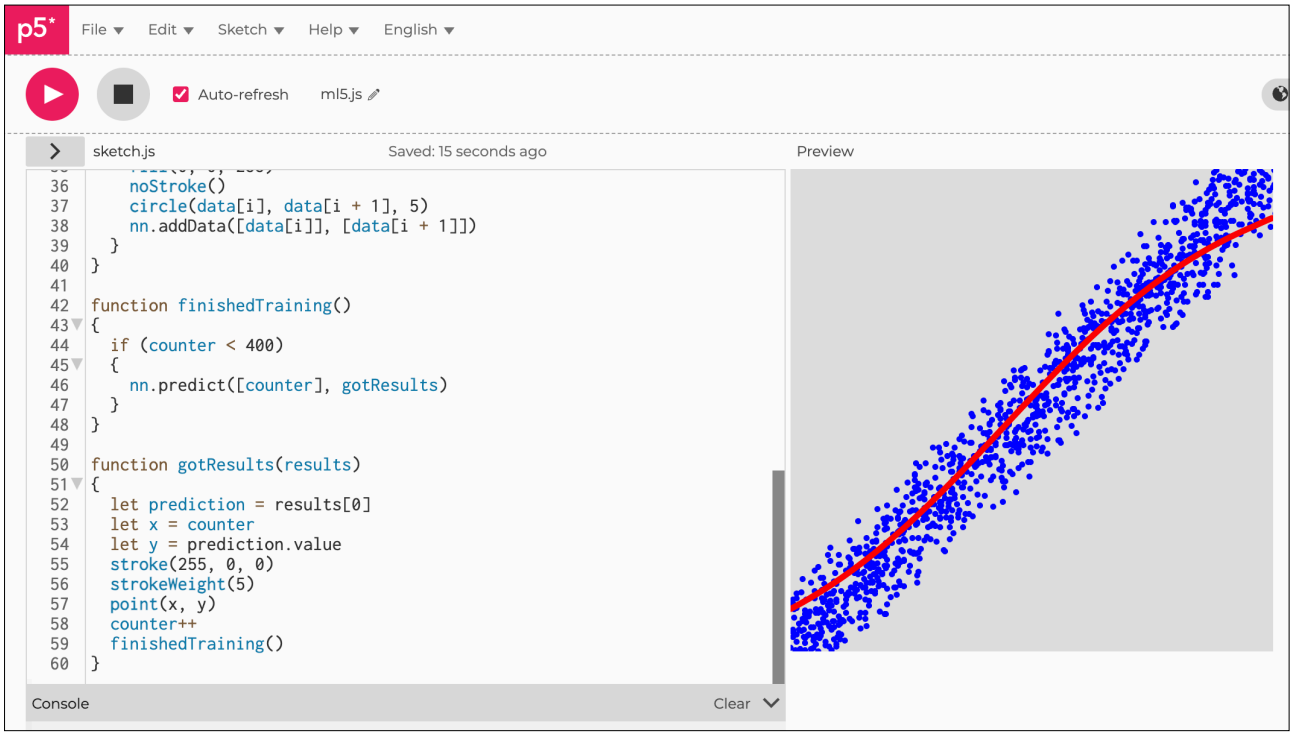
🗒️ Notes

What I get is not a very straight line at all, but this is really to illustrate the process. The next section is about saving the model, at least one we think is good enough.

🛠️ Code Explanation

`circle(data[i], data[i + 1], 5)`	The first element is the x coordinate and the second is the y coordinates.
`nn.addData([data[i]], [data[i + 1]])`	Adding the x and y coordinates to the neural network

Figure A13.9


p5*
File
Edit
Sketch
Help
English
Auto-refresh
ml5.js
sketch.js
Saved: 15 seconds ago
Preview
noStroke()
circle(data[i], data[i + 1], 5)
nn.addData([data[i]], [data[i + 1]])
}
}
function finishedTraining()
{
if (counter < 400)
{
nn.predict([counter], gotResults)
}
}
function gotResults(results)
{
let prediction = results[0]
let x = counter
let y = prediction.value
stroke(255, 0, 0)
strokeWeight(5)
point(x, y)
counter++
finishedTraining()
}
Console
Clear

Sketch A13.10 creating another button

We want a button to click on when we want to save the model. The button variable is declared, creating the button and also styling it with a bit of colour. We incorporate the button into the `gotResults()` function and call the function `saveModel()` to do just that. When the model is trained, we click on the button and the `save()` function acts on the neural network (`nn`). It should download three files, and those three files are:

- `model.json`
- `model.weights.bin`
- `model_meta.json`

```
let nn
let counter = 0
let data
let num = 0
let button
const number = 30
const spread = 30
const options = {
  task: 'regression',
  debug: false
}

async function setup()
{
  data = await loadJSON('trainingData.json')
  createCanvas(400, 400)
  button = createButton('save model')
  button.style('font-size', '20px')
  button.style('background-color', color('darkblue'))
  button.style('color', color('yellow'))
  ml5.setBackend("webgl")
  nn = ml5.neuralNetwork(options)
  trainingData()
  nn.normalizeData()
  const trainingOptions = {
    batchSize: 512
```

```
  }
  nn.train(trainingOptions, finishedTraining)
}

function trainingData()
{
  background(220)
  for (let datapoints in data)
  {
    num++
  }
  for (let i = 0; i < num; i += 2)
  {
    fill(0, 0, 255)
    noStroke()
    circle(data[i], data[i + 1], 5)
    nn.addData([data[i]], [data[i + 1]])
  }
}

function finishedTraining()
{
  if (counter < 400)
  {
    nn.predict([counter], gotResults)
  }
}

function gotResults(results)
{
  let prediction = results[0]
  let x = counter
  let y = prediction.value
  stroke(255, 0, 0)
  strokeWeight(5)
  point(x, y)
  counter++
  finishedTraining()
```

```
  button.mousePressed(saveModel)
}

function saveModel()
{
  nn.save()
}
```

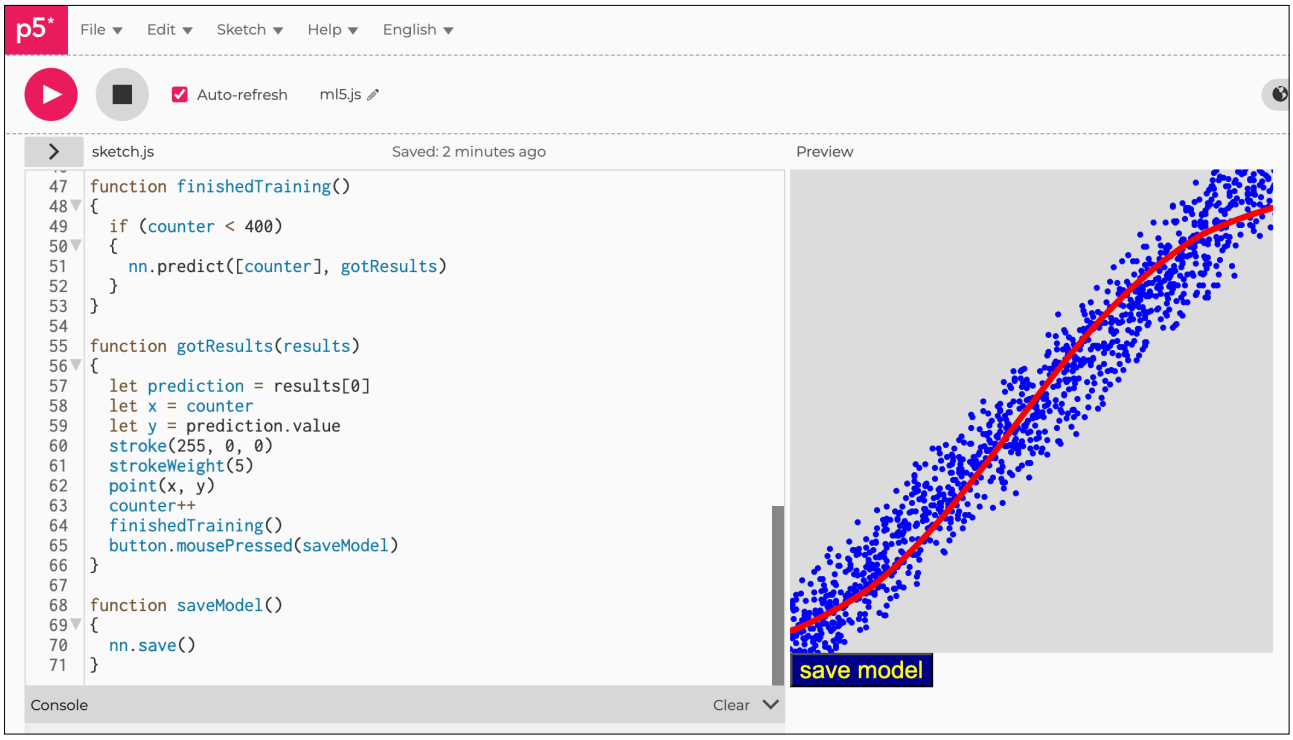
🗒️ Notes

If you want to give your saved model a different name, then you can (the default is `model`). You can also have a callback function if you want: `nn.save('differentName', callback)`. If, for some reason, you do not get three files, then I suggest using a different machine. The iPad I use doesn't download three files for some reason.

🛠️ Code Explanation

`nn.save()`	This saves the nn model. It gives it the default name 'model', you can specify another name.

Figure A13.10



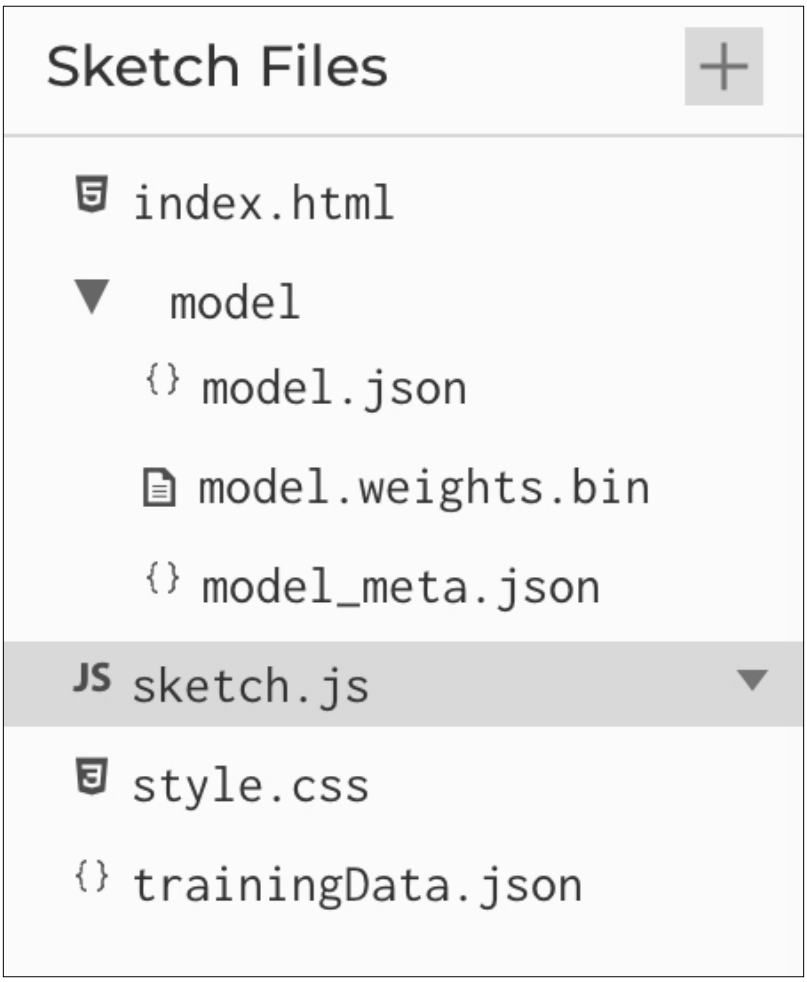
part #4
loading the model

Creating a folder for the model

We have saved the model, so it is now ready for deployment. We are going to see how well it really performs against straight-line data, but remember it was trained on data that had some variance to it, so it won't be exact like the examples in module A.

To store the saved model files, we have to put them into the same folder. Firstly, create that folder; I have called it `model`. Now upload (or drag and drop) the three files into that folder; you can do it one at a time or all three at once. You should have a folder called `model`, and inside that folder, you should have your three files, see Fig. 1 below.

Figure 4: model folder



Sketch A13.11 loading the model

❗ We are starting almost a completely new sketch

We are now going to load the model into our sketch. What we are going to do is a straight predict on some inputs (`x` values). But first, we load the model with the `load()` function.

```
let nn
let counter = 0
let data

const modelInfo = {
  model: "model/model.json",
  metadata: "model/model_meta.json",
  weights: "model/model.weights.bin"
}

function setup()
{
  createCanvas(400, 400)
  background(220)
  ml5.setBackend("webgl")
  nn = ml5.neuralNetwork()
  nn.load(modelInfo)
}
```

🗒️ Notes

We load the model with the `load()` function, giving it the folder and clear path details. This is very important; they are all case-sensitive. The three files do need to be in the same folder, and they do need the name of the folder followed by a forward slash and then the name of the file. Each one is prefixed by `model`, `metadata`, and `weights`.

🛠️ Code Explanation

`nn.load(modelInfo)`	Loading the model with the details of the model reference

Sketch A13.12 a callback

We want to make sure that it is loaded before we do the predictions. The boolean variable is `modelLoaded` and is initialised to `false`. Once the model is loaded, we then move back to the callback function `modelLoadedCallback()`. Here, we console log it and set the `modelLoaded` boolean to `true`.

```
let nn
let modelLoaded = false
let counter = 0
let data

const modelInfo = {
  model: "model/model.json",
  metadata: "model/model_meta.json",
  weights: "model/model.weights.bin"
}

function setup()
{
  createCanvas(400, 400)
  background(220)
  ml5.setBackend("webgl")
  nn = ml5.neuralNetwork()
  nn.load(modelInfo, modelLoadedCallback)
}

function modelLoadedCallback()
{
  modelLoaded = true
}
```

Notes

We are nearly there. We have loaded the model and created a callback function. We have yet to predict and draw the line.

🛠️ Code Explanation

`let modelLoaded = false`	Boolean is nitialised to false
`nn.load(modelInfo, modelLoadedCallback)`	We add a callback in the load function
`modelLoaded = true`	Now it ii set to true

Sketch A13.13 predicting

We create a function called `predicting()`. This function checks to see if the model is loaded and that the counter is less than `400`. This uses the counter as the `x` input, and `y` is the predicted output value. We draw the red line as before.

```
let nn
let modelLoaded = false
let counter = 0
let data

const modelInfo = {
  model: "model/model.json",
  metadata: "model/model_meta.json",
  weights: "model/model.weights.bin"
}

function setup()
{
  createCanvas(400, 400)
  background(220)
  ml5.setBackend("webgl")
  nn = ml5.neuralNetwork()
  nn.load(modelInfo, modelLoadedCallback)
}

function modelLoadedCallback()
{
  modelLoaded = true
  predicting()
}



function predicting()
{
  if (modelLoaded && counter < 400)
  {
    nn.predict([counter], gotResults)
```

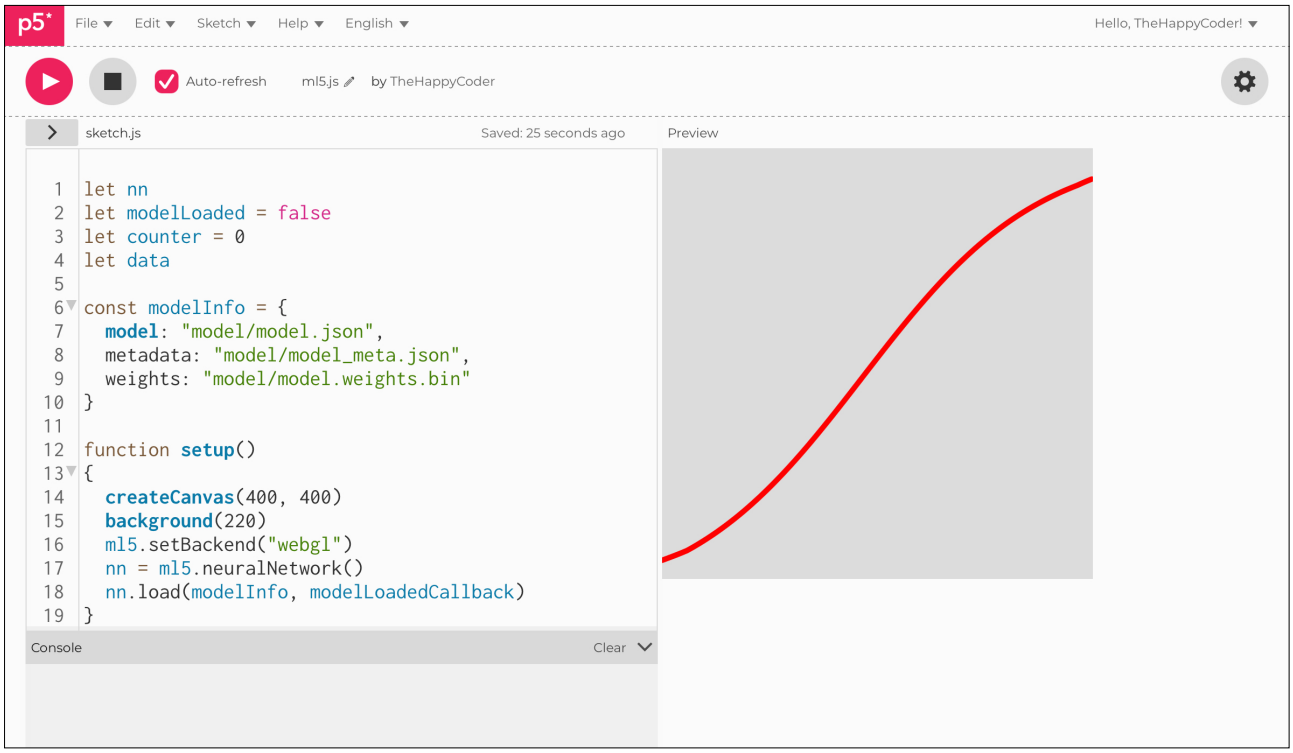
```
  }
}

function gotResults(results)
{
  let prediction = results[0]
  let x = counter
  let y = prediction.value
  stroke(255, 0, 0)
  strokeWeight(5)
  point(x, y)
  counter++
  predicting()
}
```

🗒️ Notes

We can see the predictions. It should draw the same line as before. It is a trained model.

Figure A13.13



Sketch A13.14 comparing

We will now draw what the line should look like as a comparison to the true relationship between the x input values (400) and the corresponding y values (y = x straight line equation).

```
let nn
let modelLoaded = false
let counter = 0
let data

const modelInfo = {
  model: "model/model.json",
  metadata: "model/model_meta.json",
  weights: "model/model.weights.bin"
}

function setup()
{
  createCanvas(400, 400)
  background(220)
  ml5.setBackend("webgl")
  nn = ml5.neuralNetwork()
  nn.load(modelInfo, modelLoadedCallback)
}

function modelLoadedCallback()
{
  modelLoaded = true
  predicting()
}

function predicting()
{
  if (modelLoaded && counter < 400)
  {
    nn.predict([counter], gotResults)
  }
```

```
}
```

```
function draw()
{
  for (let i = 0; i < width; i++)
  {
    data = [i, height - i]
    fill(0, 0, 255)
    noStroke()
    circle(data[0], data[1], 5)
  }
}
```

```
function gotResults(results)
{
  let prediction = results[0]
  let x = counter
  let y = prediction.value
  stroke(255, 0, 0)
  strokeWeight(5)
  point(x, y)
  counter++
  predicting()
}
```

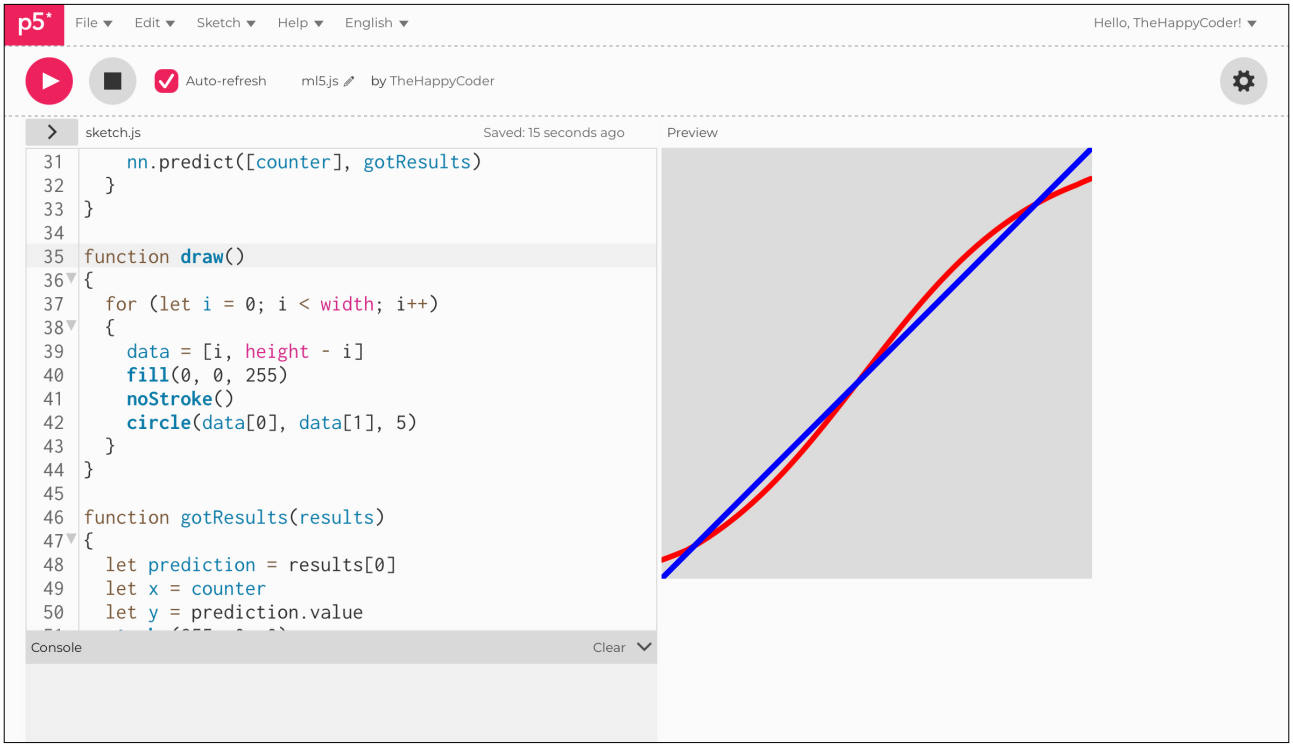
🗒️ Notes

You can see it is reasonably close to the blue line. Not bad but not perfect. We haven't used many of the other `hyperparameters` we could alter, so there is plenty of room for improvement.

🌻 Challenges

1. Try the line but with more of the hyperparameters.
2. Try the sine wave

Figure A13.14



Additional notes

Although we know the length of the dataset, it isn't always obvious, so here are three options for collecting that bit of information when looping through the dataset. We can get the length of a .json file with the following code snippets and button styling.

Option 1:

```
let num = Object.keys(data).length
```

Option 2:

The one we used.

```
let num = 0
for (let key in data)
{
  num++
}
```

Option 3:

```
let num = Object.entries(data).length
```

Option 4:

Code for changing the font colour and background of the button.

```
let button

function setup()
{
  createCanvas(400, 400)
  button = createButton("Click Me")
  button.position(190, 200)
  button.style('background-color', color(255, 0, 0))
  button.style('color', color(255))
}
```