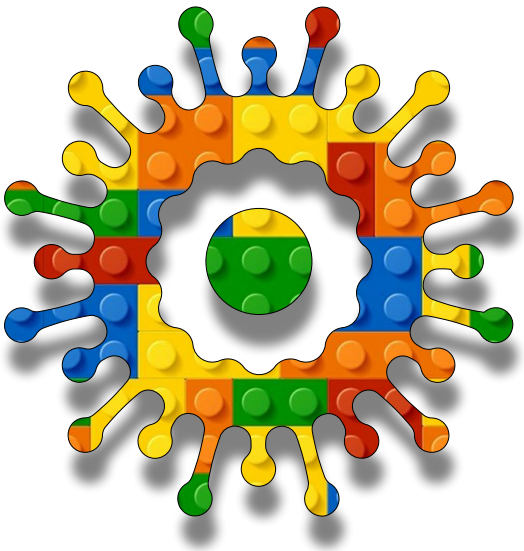
Algorithmic Intelligence Module B Unit #7 mobileNet Image Classification



Module B Unit #7 mobileNet image classification

Introduction to mobileNet image classification

The pre-trained model we are going to use is called mobileNet. It is a very handy model that you can use online. It uses a dataset from ImageNet, which has nearly 15 million images. So it is quite comprehensive but not complete, as you may well find out when you try to look at objects.

This unit, we just upload images to classify.

Sketch B7.1 index.html

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

```
<!DOCTYPE html>
<html lang="en"><head>
    <script src="https://cdn.jsdelivr.net/npm/p5@2.2.2/lib/p5.js"></script>
    <script src="https://unpkg.com/ml5@1/dist/ml5.min.js"></script>
    <link rel="stylesheet" type="text/css" href="style.css">
    <meta charset="utf-8">
  </head>
  <body>
    <main>
    </main>
    <script src="sketch.js"></script>
</body></html>
```

Sketch B7.2 callback

We have now access to another pre-trained model with `ml5.js` called `mobileNet`. We are going to check that everything is working by having a callback. The callback tells us when it is loaded in the console.

```
let mobilenet


function setup()
{
  createCanvas(400, 400)
  mobilenet = ml5.imageClassifier('MobileNet', modelReady)
}


function modelReady()
{
  console.log('model is ready')
}
```

Notes

We get the callback console log: `model is ready`. It may take a few seconds.

A banana

You can use any object you like, but choose something that doesn't have any other object in it. Just make sure it works OK from the start. I have selected a banana image from Unsplash.

Figure 1: Source: 'Mockup Graphics' on Unsplash

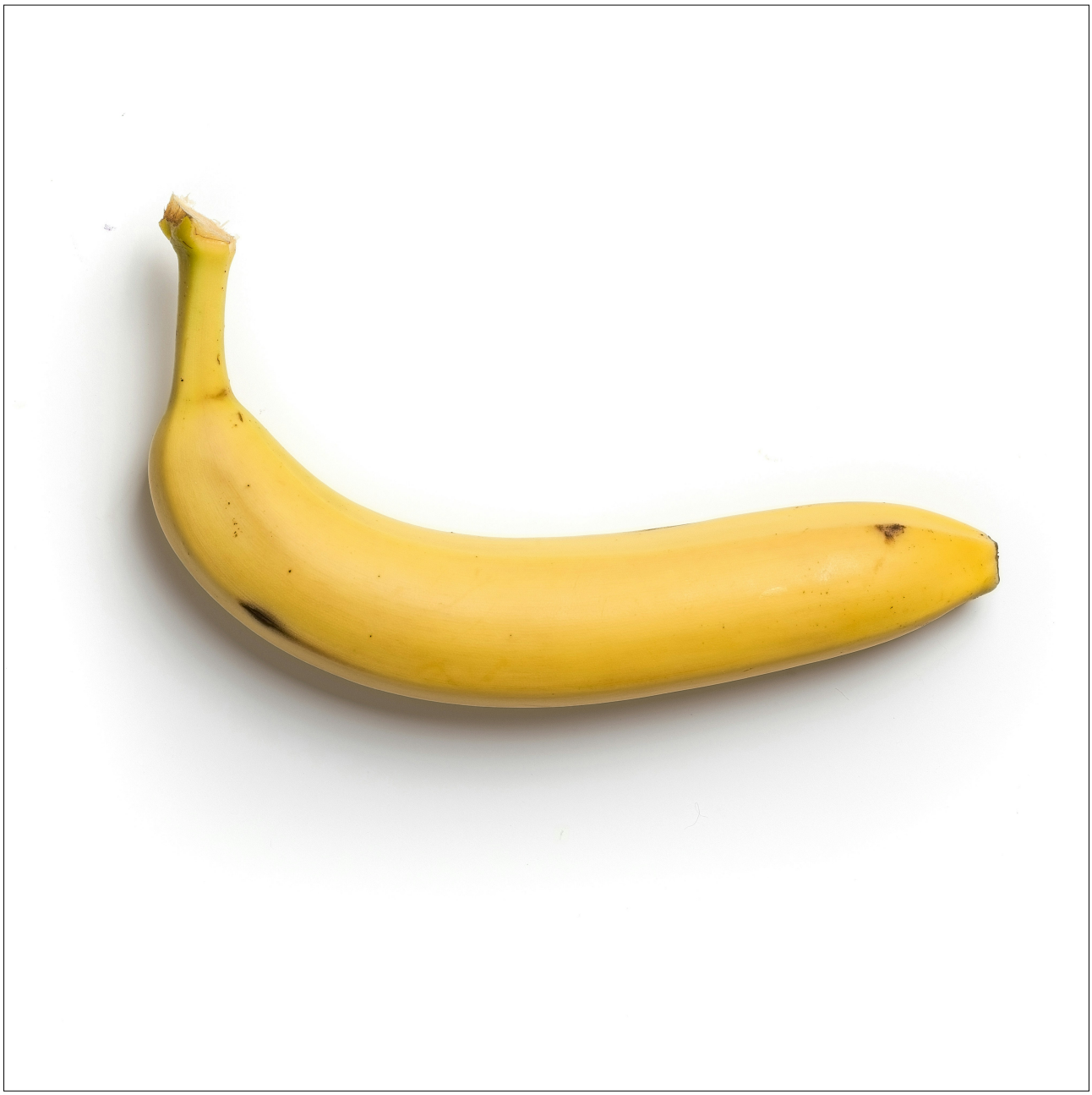


Figure 2: Upload it to the sketch

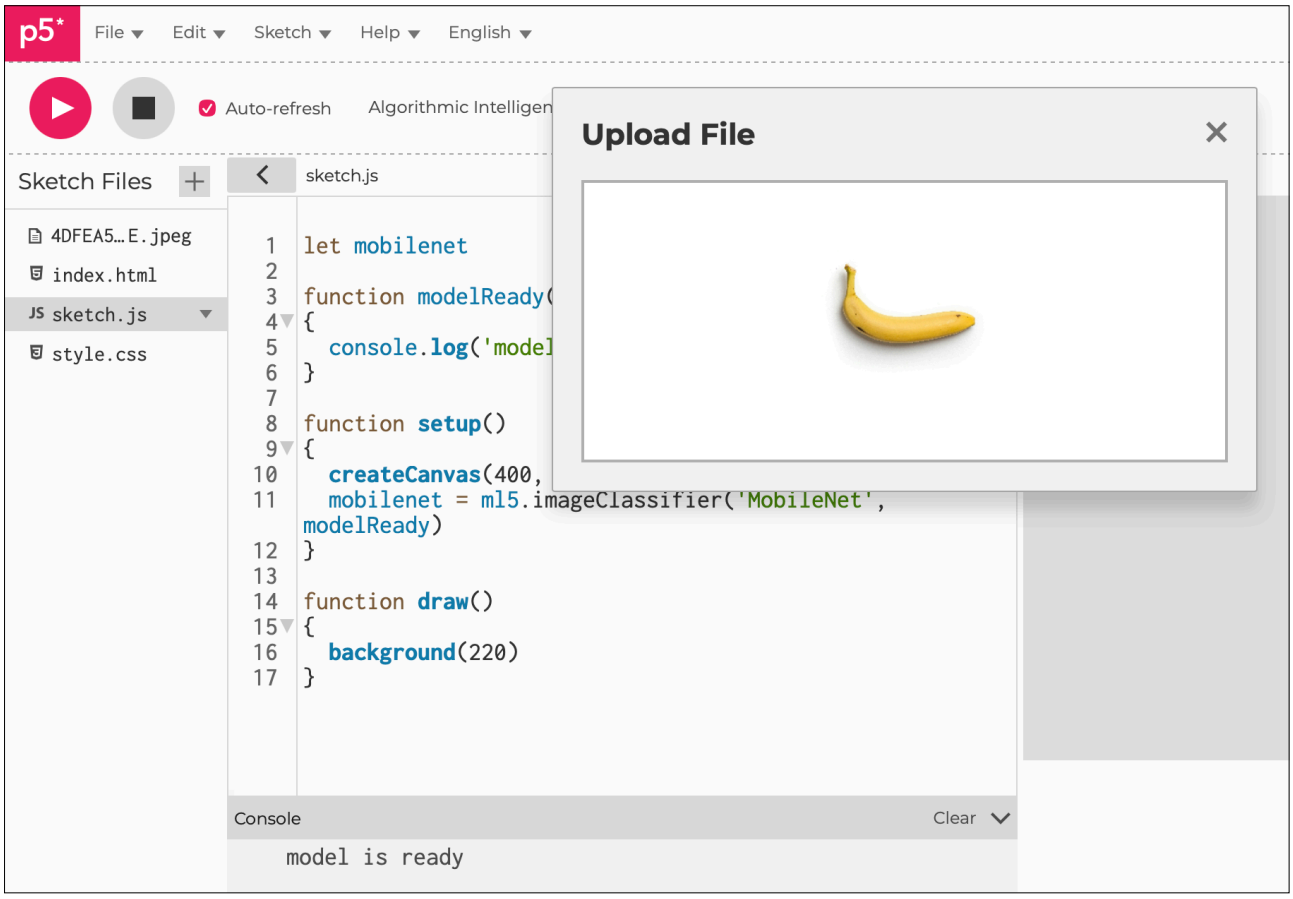
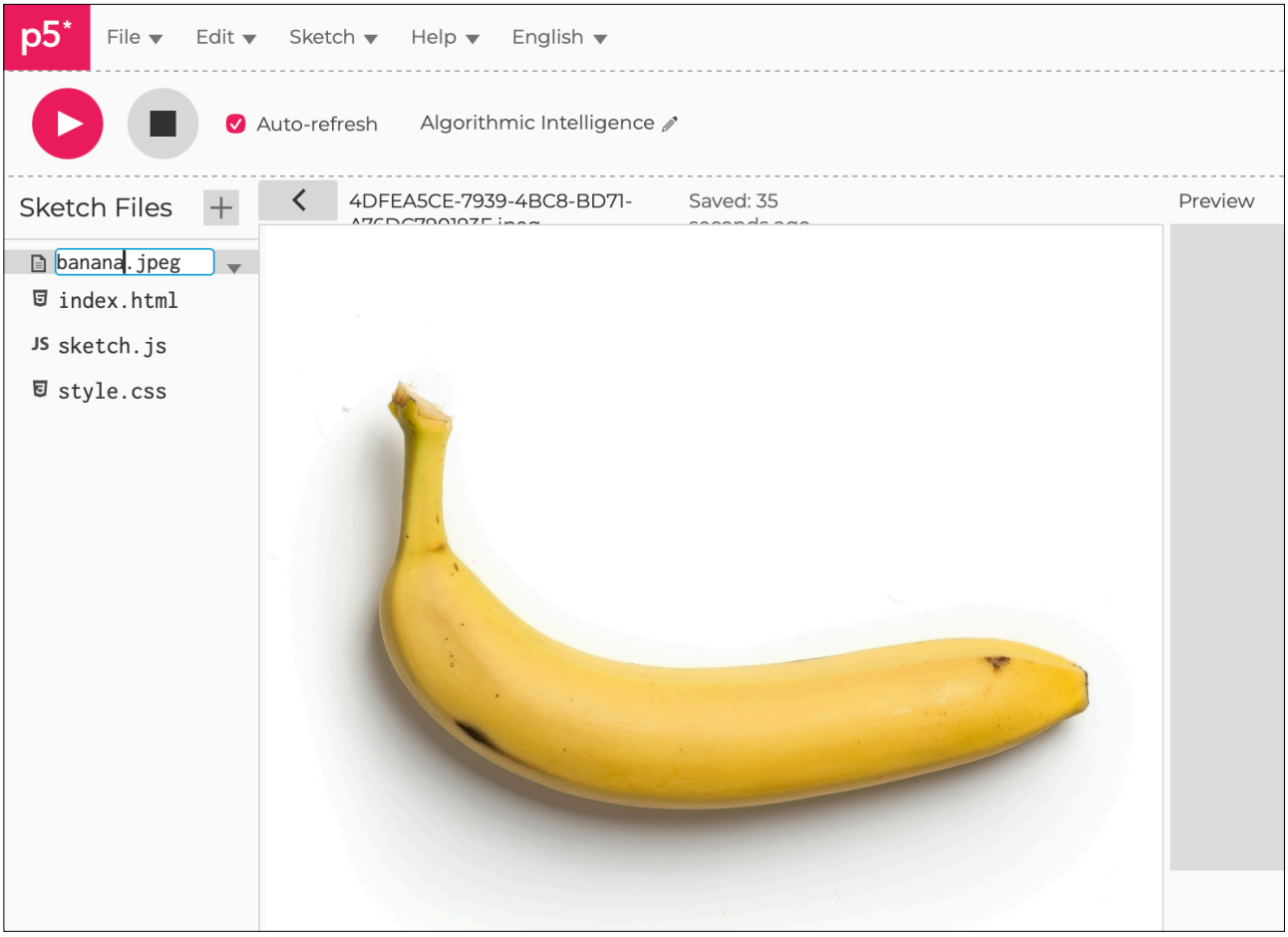


Figure 3: rename 'banana'



Sketch B7.3 the appearance of the banana

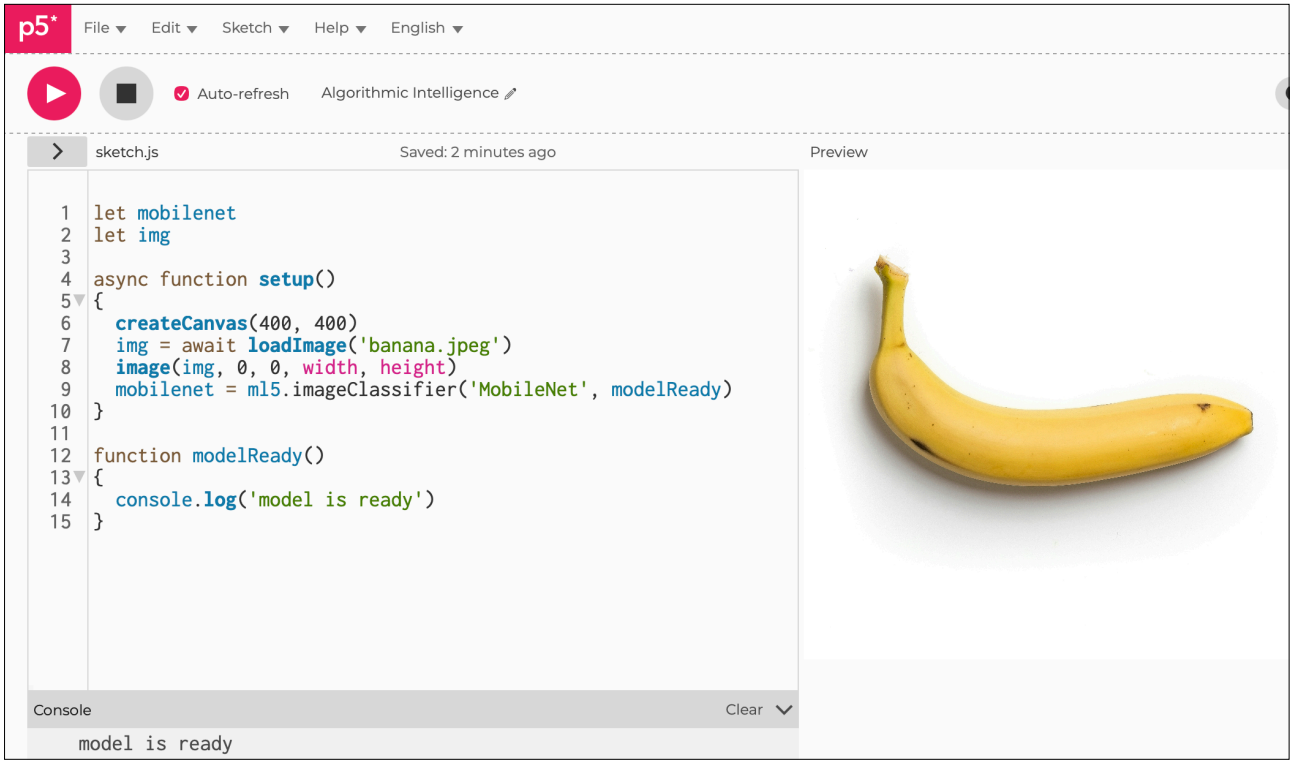
Now that you have successfully uploaded an image and renamed it, then we can display it on the canvas.

```
let mobilenet
let img

async function setup()
{
  createCanvas(400, 400)
  img = await loadImage('banana.jpeg')
  image(img, 0, 0, width, height)
  mobilenet = ml5.imageClassifier('MobileNet', modelReady)
}

function modelReady()
{
  console.log('model is ready')
}
```

Notes

We use `async` and `await` because we want the image to be downloaded first, before the model is loaded. The banana should magically appear on the canvas. I deliberately chose a square image so there's no squashing or stretching.

Figure B7.3


p5*
File
Edit
Sketch
Help
English
Auto-refresh
Algorithmic Intelligence
sketch.js
Saved: 2 minutes ago
Preview
let mobilenet
let img
async function setup()
{
createCanvas(400, 400)
img = await loadImage('banana.jpeg')
image(img, 0, 0, width, height)
mobilenet = ml5.imageClassifier('MobileNet', modelReady)
}
function modelReady()
{
console.log('model is ready')
}
Console
Clear
model is ready

Sketch B7.4 the classification

We can make a few changes to the sketch to get the model to `classify` (predict) what the object is in the image. The first change is that we need the model to load before we start to classify. We get rid of the callback and change its name (to `gotResults`) so we can repurpose it for collecting the results of the classification.

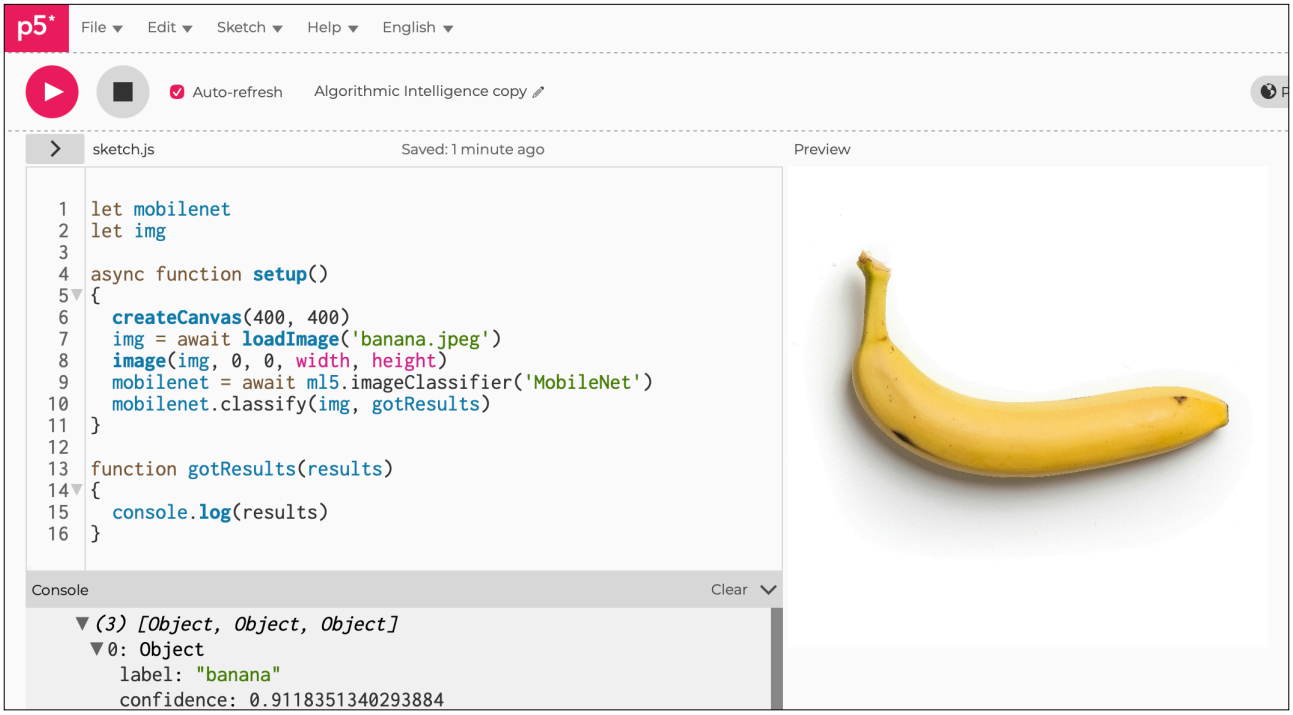
```
let mobilenet
let img


async function setup()
{
  createCanvas(400, 400)
  img = await loadImage('banana.jpeg')
  image(img, 0, 0, width, height)
  mobilenet = await ml5.imageClassifier('MobileNet')
  mobilenet.classify(img, gotResults)
}


function gotResults(results)
{
  console.log(results)
}
```

🗒️ Notes

You will notice that we get the top three guesses (predictions), and they are scored as a probability. Clearly, this is a banana with almost `100%` (`0.91`) score; the other two are interesting. A slug, I could just about get!

! If you get an error message, just check you have removed the redundant code.

Figure B7.4


p5*
File
Edit
Sketch
Help
English
Auto-refresh
Algorithmic Intelligence copy
sketch.js
Saved: 1 minute ago
Preview
let mobilenet
let img
async function setup()
{
createCanvas(400, 400)
img = await loadImage('banana.jpeg')
image(img, 0, 0, width, height)
mobilenet = await ml5.imageClassifier('MobileNet')
mobilenet.classify(img, gotResults)
}
function gotResults(results)
{
console.log(results)
}
Console
Clear
▼(3) [Object, Object, Object]
▼0: Object
label: "banana"
confidence: 0.9118351340293884

Sketch B7.5 get the results

We want the top result, which is the `label` at `index [0]`, the first one. This is also true for probability (confidence). We create two variables to hold these values as strings, as we will want to display them on the canvas as a fixed value or name.

! Remove the console log.

```
let mobilenet
let img
let label = ''
let confidence = ''

async function setup()
{
  createCanvas(400, 400)
  img = await loadImage('banana.jpeg')
  image(img, 0, 0, width, height)
  mobilenet = await ml5.imageClassifier('MobileNet')
  mobilenet.classify(img, gotResults)
}

function gotResults(results)
{
  label = results[0].label
  confidence = results[0].confidence
}
```

Notes

Nothing new to see.

Sketch B7.6 display

! Move the image into the `draw()` function from `setup()`.

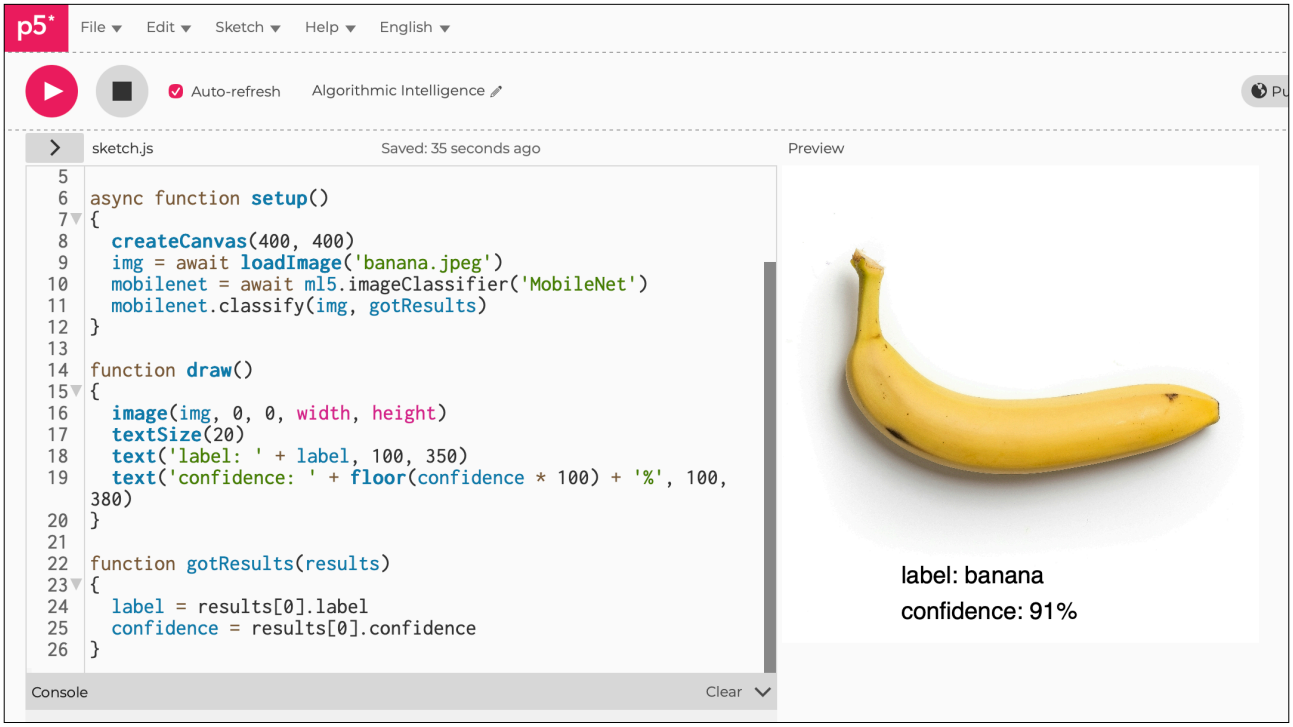
We multiply the confidence by `100` (to get the percentage) and take the floor value. We then print the text onto the canvas. We use the `draw()` function for this.

```
let mobilenet
let img
let label = ''
let confidence = ''

async function setup()
{
  createCanvas(400, 400)
  img = await loadImage('banana.jpeg')
  mobilenet = await ml5.imageClassifier('MobileNet')
  mobilenet.classify(img, gotResults)
}

function draw()
{
  image(img, 0, 0, width, height)
  textSize(20)
  text('label: ' + label, 100, 350)
  text('confidence: ' + floor(confidence * 100) + '%', 100, 380)
}

function gotResults(results)
{
  label = results[0].label
  confidence = results[0].confidence
}
```

Notes

A very confident banana

Figure B7.6


p5*
File
Edit
Sketch
Help
English
Auto-refresh
Algorithmic Intelligence
sketch.js
Saved: 35 seconds ago
Preview
async function setup()
{
createCanvas(400, 400)
img = await loadImage('banana.jpeg')
mobilenet = await ml5.imageClassifier('MobileNet')
mobilenet.classify(img, gotResults)
}
function draw()
{
image(img, 0, 0, width, height)
textSize(20)
text('label: ' + label, 100, 350)
text('confidence: ' + floor(confidence * 100) + '%', 100, 380)
}
function gotResults(results)
{
label = results[0].label
confidence = results[0].confidence
}
Console
Clear
label: banana
confidence: 91%