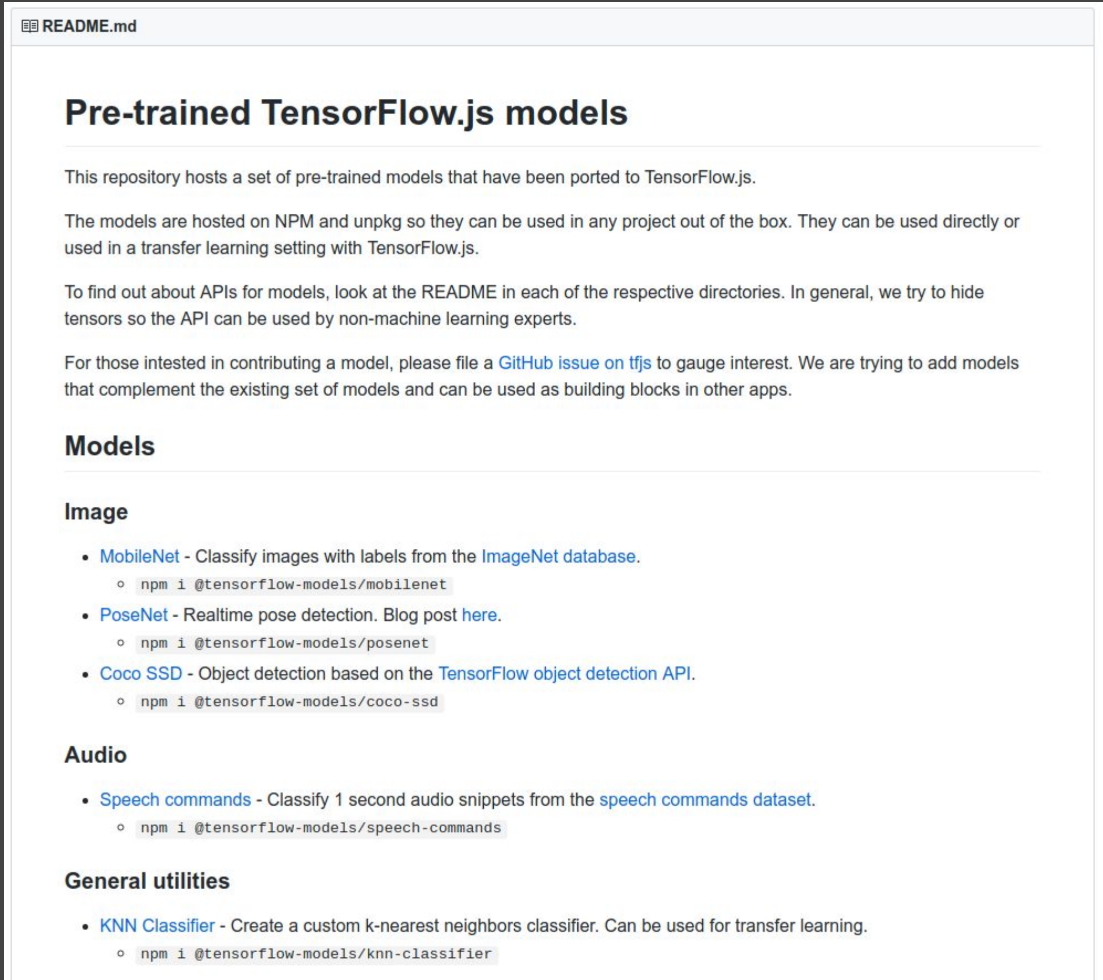
Copyright Notice

https://github.com/tensorflow/tfjs-models



README.md

Pre-trained TensorFlow.js models

This repository hosts a set of pre-trained models that have been ported to TensorFlow.js.

The models are hosted on NPM and unpkg so they can be used in any project out of the box. They can be used directly or used in a transfer learning setting with TensorFlow.js.

To find out about APIs for models, look at the README in each of the respective directories. In general, we try to hide tensors so the API can be used by non-machine learning experts.

For those intested in contributing a model, please file a GitHub issue on tfjs to gauge interest. We are trying to add models that complement the existing set of models and can be used as building blocks in other apps.

Models

Image

- MobileNet - Classify images with labels from the ImageNet database.
  - `npm i @tensorflow-models/mobilenet`
- PoseNet - Realtime pose detection. Blog post here.
  - `npm i @tensorflow-models/posenet`
- Coco SSD - Object detection based on the TensorFlow object detection API.
  - `npm i @tensorflow-models/coco-ssd`

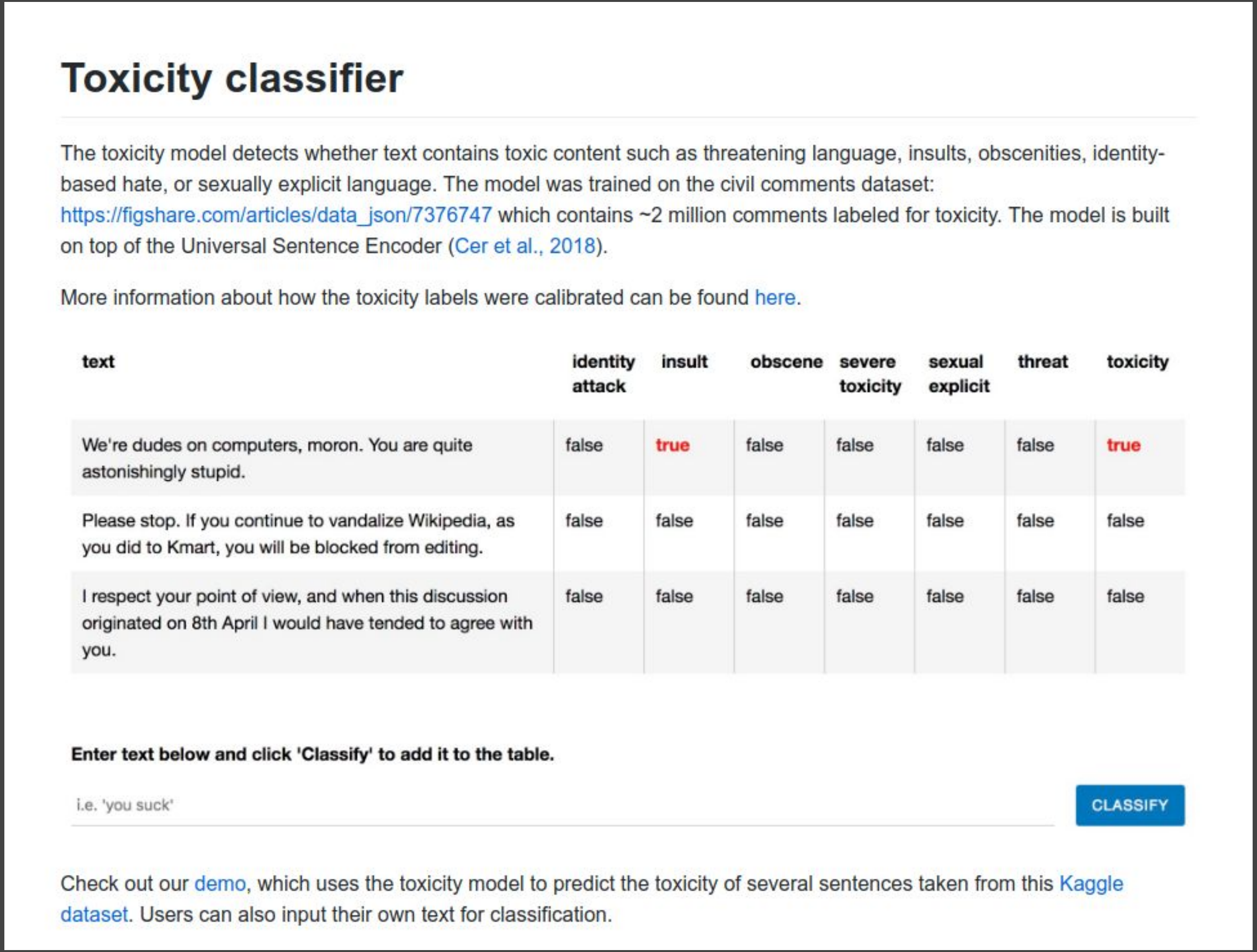
Audio

- Speech commands - Classify 1 second audio snippets from the speech commands dataset.
  - `npm i @tensorflow-models/speech-commands`

General utilities

- KNN Classifier - Create a custom k-nearest neighbors classifier. Can be used for transfer learning.
  - `npm i @tensorflow-models/knn-classifier`

https://github.com/tensorflow/tfjs-models/tree/master/toxicity



Toxicity classifier

The toxicity model detects whether text contains toxic content such as threatening language, insults, obscenities, identity-based hate, or sexually explicit language. The model was trained on the civil comments dataset: https://figshare.com/articles/data_json/7376747 which contains ~2 million comments labeled for toxicity. The model is built on top of the Universal Sentence Encoder (Cer et al., 2018).

More information about how the toxicity labels were calibrated can be found here.

text	identity attack	insult	obscene	severe toxicity	sexual explicit	threat	toxicity
We're dudes on computers, moron. You are quite astonishingly stupid.	false	true	false	false	false	false	true
Please stop. If you continue to vandalize Wikipedia, as you did to Kmart, you will be blocked from editing.	false	false	false	false	false	false	false
I respect your point of view, and when this discussion originated on 8th April I would have tended to agree with you.	false	false	false	false	false	false	false

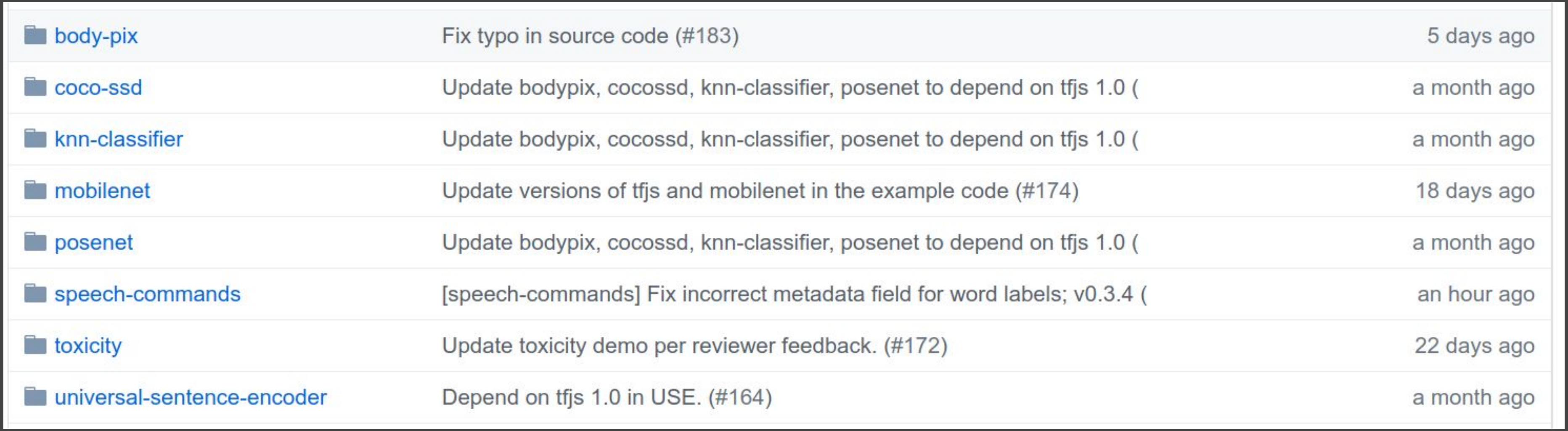
Enter text below and click 'Classify' to add it to the table.

Check out our demo, which uses the toxicity model to predict the toxicity of several sentences taken from this Kaggle dataset. Users can also input their own text for classification.

```html
<script src="https://cdn.jsdelivr.net/npm/@tensorflow/tfjs@latest"></script>
<script src="https://cdn.jsdelivr.net/npm/@tensorflow-models/toxicity"></script>
```

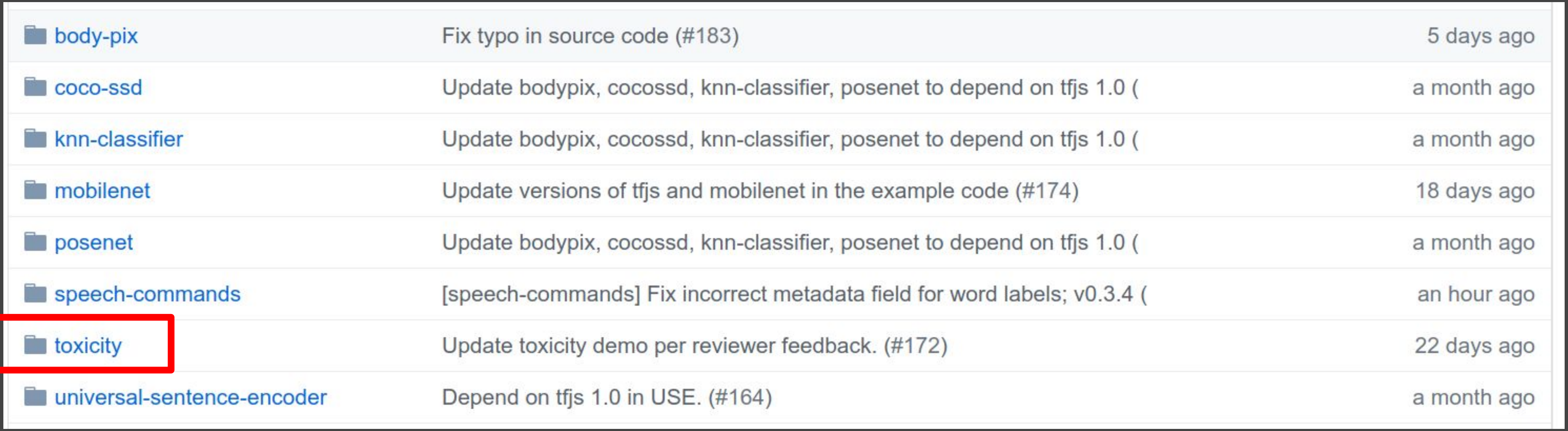
```html
<script src="https://cdn.jsdelivr.net/npm/@tensorflow/tfjs@latest"></script>
<script src="https://cdn.jsdelivr.net/npm/@tensorflow-models/toxicity"></script>
```

https://github.com/tensorflow/tfjs-models



body-pix	Fix typo in source code (#183)	5 days ago
coco-ssd	Update bodypix, cocossd, knn-classifier, posenet to depend on tfjs 1.0 (	a month ago
knn-classifier	Update bodypix, cocossd, knn-classifier, posenet to depend on tfjs 1.0 (	a month ago
mobilenet	Update versions of tfjs and mobilenet in the example code (#174)	18 days ago
posenet	Update bodypix, cocossd, knn-classifier, posenet to depend on tfjs 1.0 (	a month ago
speech-commands	[speech-commands] Fix incorrect metadata field for word labels; v0.3.4 (	an hour ago
toxicity	Update toxicity demo per reviewer feedback. (#172)	22 days ago
universal-sentence-encoder	Depend on tfjs 1.0 in USE. (#164)	a month ago

https://github.com/tensorflow/tfjs-models



body-pix	Fix typo in source code (#183)	5 days ago
coco-ssd	Update bodypix, cocossd, knn-classifier, posenet to depend on tfjs 1.0 (	a month ago
knn-classifier	Update bodypix, cocossd, knn-classifier, posenet to depend on tfjs 1.0 (	a month ago
mobilenet	Update versions of tfjs and mobilenet in the example code (#174)	18 days ago
posenet	Update bodypix, cocossd, knn-classifier, posenet to depend on tfjs 1.0 (	a month ago
speech-commands	[speech-commands] Fix incorrect metadata field for word labels; v0.3.4 (	an hour ago
toxicity	Update toxicity demo per reviewer feedback. (#172)	22 days ago
universal-sentence-encoder	Depend on tfjs 1.0 in USE. (#164)	a month ago

```html
<html>
<head>
<script src="https://cdn.jsdelivr.net/npm/@tensorflow/tfjs@latest"></script>
<script src="https://cdn.jsdelivr.net/npm/@tensorflow-models/toxicity"></script>
</head>
<body></body>
</html>
```

```html
<html>
<head>
<script src="https://cdn.jsdelivr.net/npm/@tensorflow/tfjs@latest"></script>
<script src="https://cdn.jsdelivr.net/npm/@tensorflow-models/toxicity"></script>
<script>
// Your code here
</script>
</head>
<body></body>
</html>
```

```
const threshold = 0.9;
```

Value for not-insult

```
const threshold = 0.9;

"label": "insult",
    "results": [{
      "probabilities": [0.9187529683113098, 0.0812470614910125 7],
        "match": false
      }]
```

Value for insult

```
const threshold = 0.9;

"label": "insult",
    "results": [{
      "probabilities": [0.9187529683113098, 0.0812470614910125 7],
        "match": false
      }]
```

Value for not-insult > threshold
So match = false

```
const threshold = 0.9;

"label": "insult",
    "results": [{
      "probabilities": [0.08124706149101257, 0.9187529683113098],
        "match": true
      }]
```

Value for insult > threshold
So match = true

```
const threshold = 0.9;

"label": "insult",
    "results": [{
      "probabilities": [0.5, 0.5],
        "match": null
      }]
```

Neither value > threshold
So match = null

```
toxicity.load(threshold).then(model => {
  const sentences = ['you suck!'];
  model.classify(sentences).then(predictions => {
  // Handle Results
  });
});
```

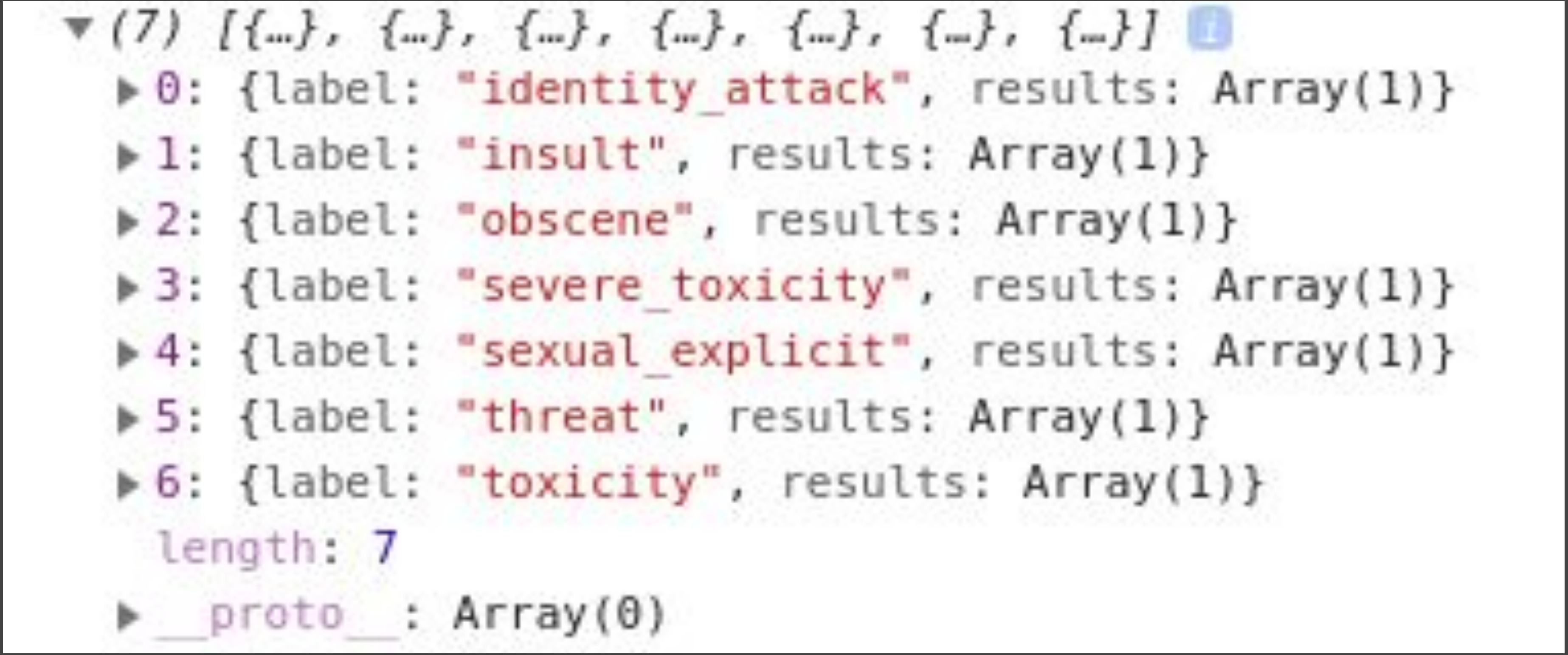
```
toxicity.load(threshold).then(model => {
  const sentences = ['you suck!'];
  model.classify(sentences).then(predictions => {
  // Handle Results
  });
});
```

```
toxicity.load(threshold).then(model => {
  const sentences = ['you suck!'];
  model.classify(sentences).then(predictions => {
  // Handle Results
  });
});
```

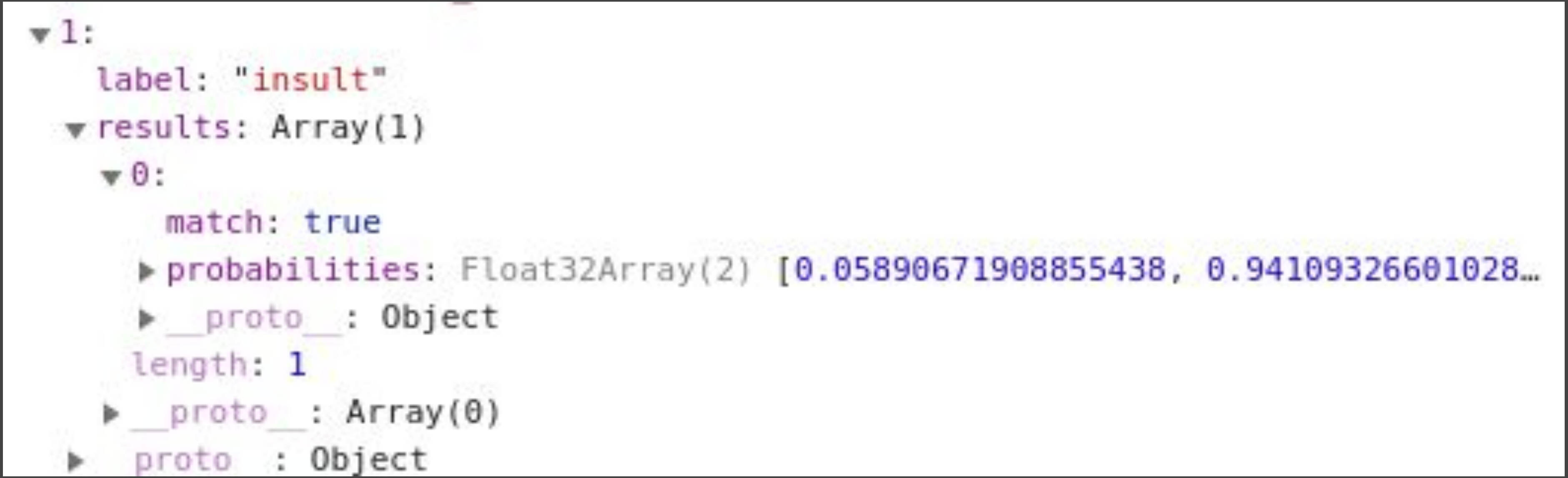
```
toxicity.load(threshold).then(model => {
  const sentences = ['you suck!'];
  model.classify(sentences).then(predictions => {
  // Handle Results
  });
});
```

```
toxicity.load(threshold).then(model => {
  const sentences = ['you suck!'];
  model.classify(sentences).then(predictions => {
  // Handle Results
  });
});
```

```
toxicity.load(threshold).then(model => {
  const sentences = ['you suck!'];
  model.classify(sentences).then(predictions => {
  // Handle Results
  });
});
```



```
▼ (7) [{…}, {…}, {…}, {…}, {…}, {…}, {…}]
  ▶ 0: {label: "identity_attack", results: Array(1)}
  ▶ 1: {label: "insult", results: Array(1)}
  ▶ 2: {label: "obscene", results: Array(1)}
  ▶ 3: {label: "severe_toxicity", results: Array(1)}
  ▶ 4: {label: "sexual_explicit", results: Array(1)}
  ▶ 5: {label: "threat", results: Array(1)}
  ▶ 6: {label: "toxicity", results: Array(1)}
    length: 7
  ▶ __proto__: Array(0)
```



```
▼1:
   label: "insult"
 ▼results: Array(1)
   ▼0:
      match: true
     ▶probabilities: Float32Array(2) [0.05890671908855438, 0.9410932660102…
     ▶__proto__: Object
    length: 1
   ▶__proto__: Array(0)
 ▶ __proto__: Object
```

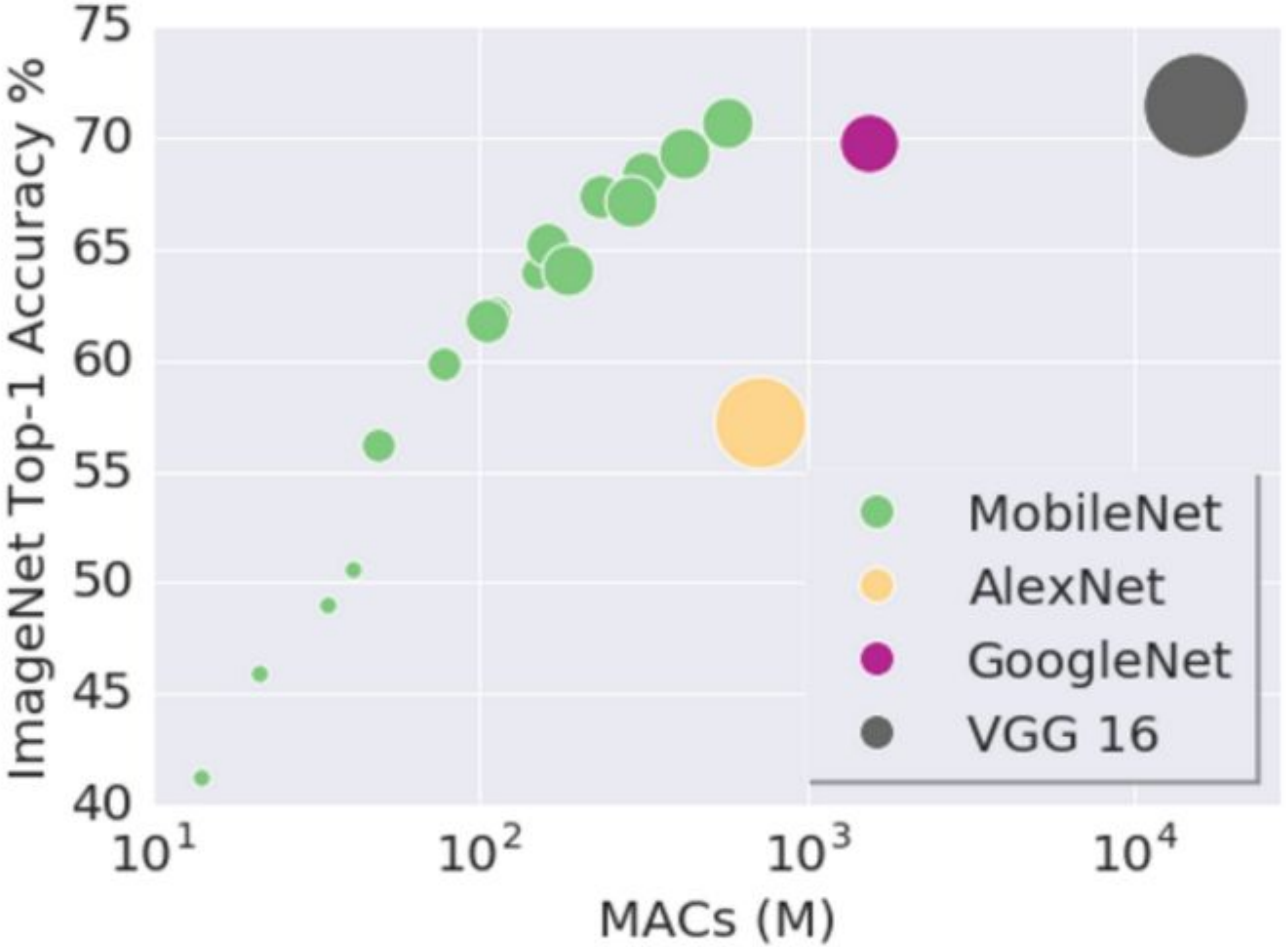
```
predictions[1].label;
predictions[1].results[0].probabilities[0];
predictions[1].results[0].probabilities[1];
predictions[1].results[0].match;
```

```
"label": "insult",
 "results": [{
  "probabilities": [0.5, 0.5],
   "match": null
  }]
```

```
for(i=0; i<7; i++){
    if(predictions[i].results[0].match){
        console.log(predictions[i].label + " was found with a probability of " +
                    predictions[i].results[0].probabilities[1]);
    }
}
```

```
insult was found with a probability of 0.9410932660102844
toxicity was found with a probability of 0.9766321778297424
```

```
predictions[1].label;
predictions[1].results[n].probabilities[0];
predictions[1].results[n].probabilities[1];
predictions[1].results[n].match;
```



ImageNet Top-1 Accuracy %
75
70
65
60
55
50
45
40
10^1
10^2
10^3
10^4
MACs (M)
MobileNet
AlexNet
GoogleNet
VGG 16

http://bit.ly/mobilenet-labels

```
00: background
01: tench
02: goldfish
03: great white shark
04: tiger shark
05: hammerhead
06: electric ray
07: stingray
08: cock
09: hen
10: ostrich
11: brambling
12: goldfinch
13: house finch
...
```

```html
<script src="https://cdn.jsdelivr.net/npm/@tensorflow/tfjs@1.0.1"> </script>
<script src="https://cdn.jsdelivr.net/npm/@tensorflow-models/mobilenet@1.0.0"> </script>
```

```html
<body>
    <img id="img" src="coffee.jpg"></img>
    <div id="output" style="font-family:courier;font-size:24px;height=300px"></div>
</body>
```

```
<body>
    <img id="img" src="coffee.jpg"></img>
    <div id="output" style="font-family:courier;font-size:24px;height=300px"></div>
</body>
```

```javascript
const img = document.getElementById('img');
mobilenet.load().then(model => {
  model.classify(img).then(predictions => {
    console.log(predictions);
  });
});
```

```
const img = document.getElementById('img');
mobilenet.load().then(model => {
  model.classify(img).then(predictions => {
    console.log(predictions);
  });
});
```

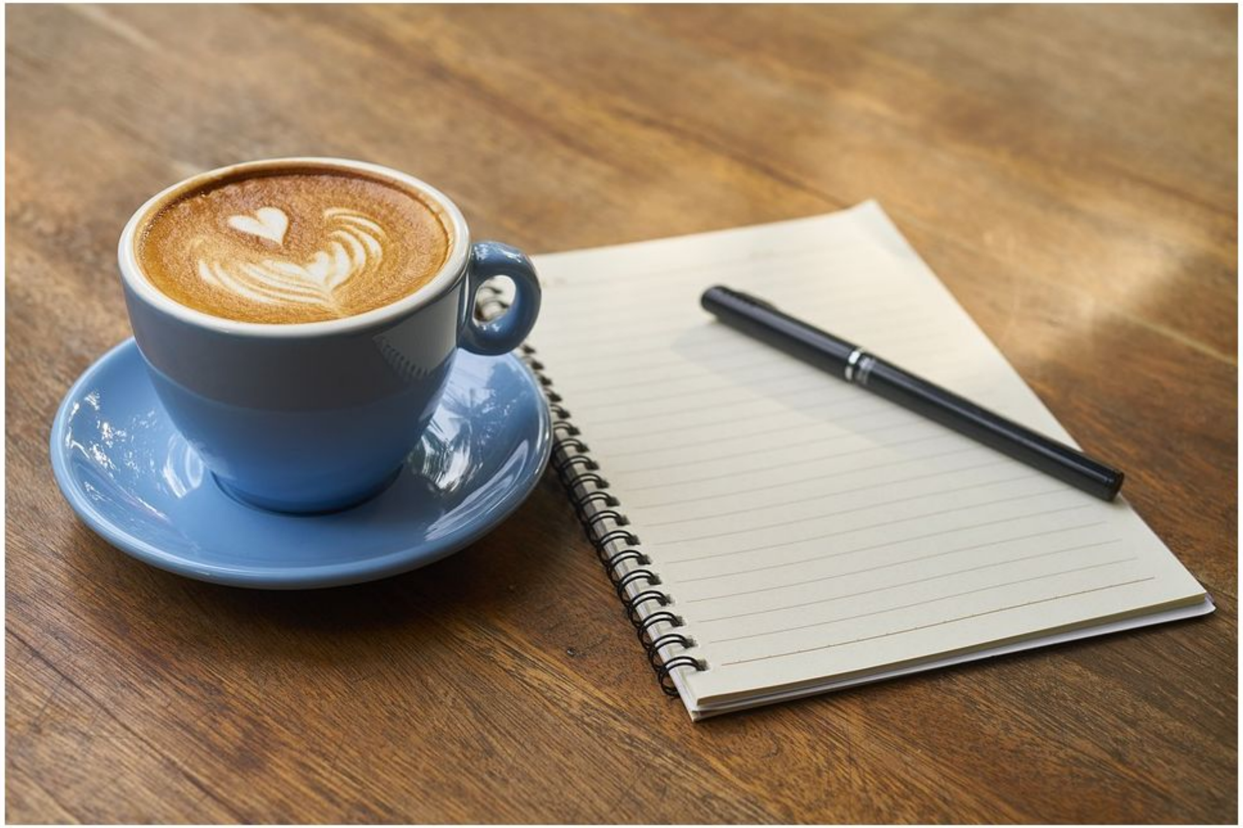
```
const img = document.getElementById('img');
mobilenet.load().then(model => {
  model.classify(img).then(predictions => {
    console.log(predictions);
  });
});
```

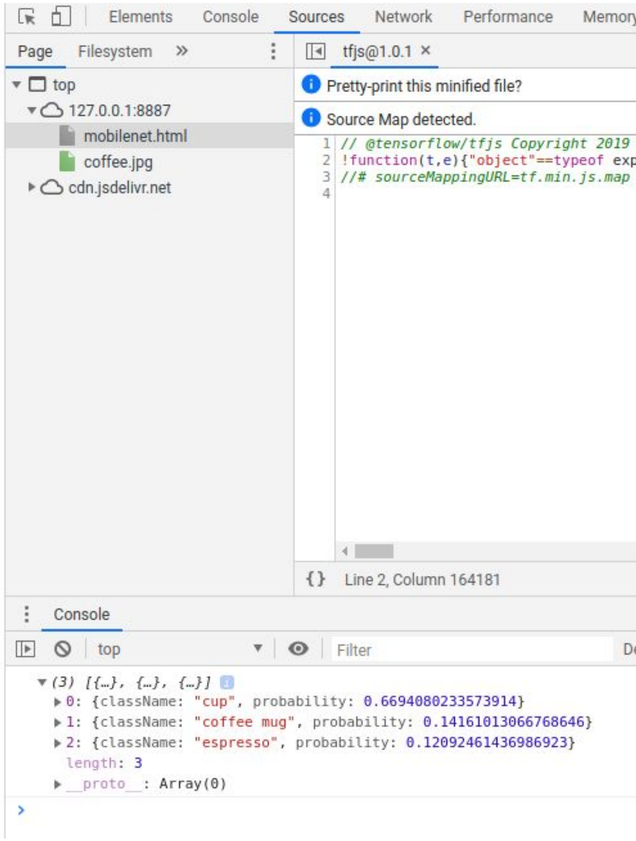
```
const img = document.getElementById('img');
mobilenet.load().then(model => {
  model.classify(img).then(predictions => {
    console.log(predictions);
  });
});
```

```
const img = document.getElementById('img');
mobilenet.load().then(model => {
  model.classify(img).then(predictions => {
    console.log(predictions);
  });
});
```

```
const img = document.getElementById('img');
mobilenet.load().then(model => {
  model.classify(img).then(predictions => {
    console.log(predictions);
  });
});
```

```
const img = document.getElementById('img');
mobilenet.load().then(model => {
  model.classify(img).then(predictions => {
    console.log(predictions);
  });
});
```





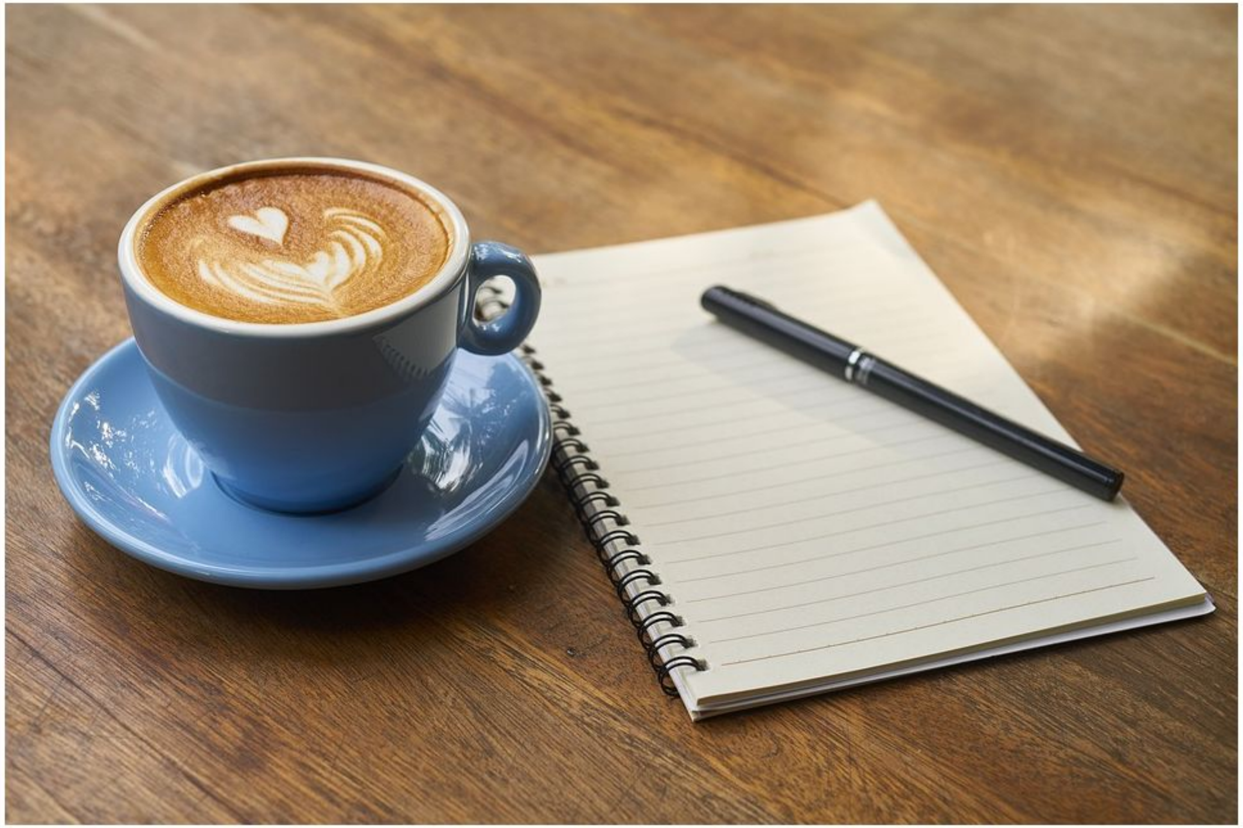
Pretty-print this minified file?

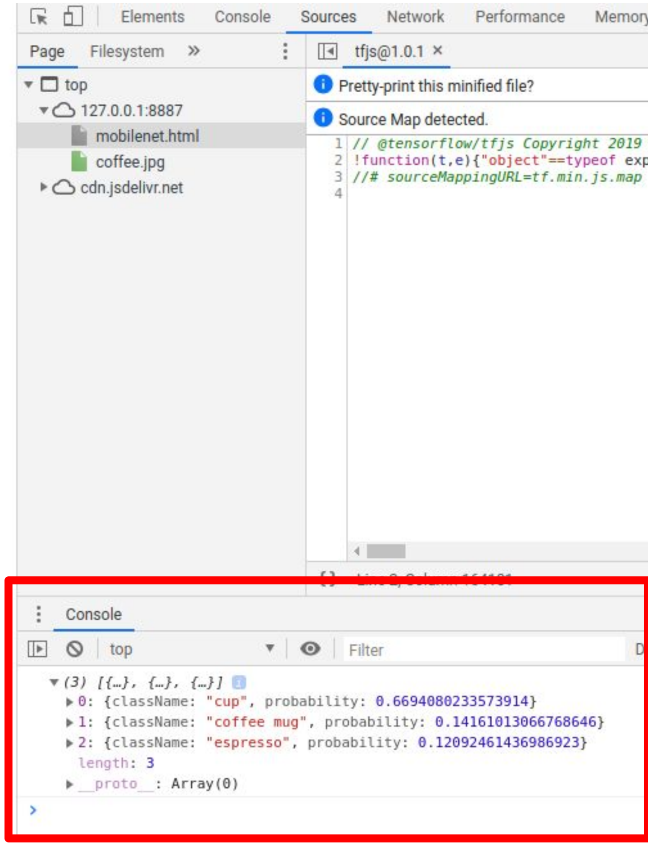
Source Map detected.

```
// @tensorflow/tfjs Copyright 2019
!function(t,e){"object"==typeof exp
//# sourceMappingURL=tf.min.js.map
```

Line 2, Column 164181

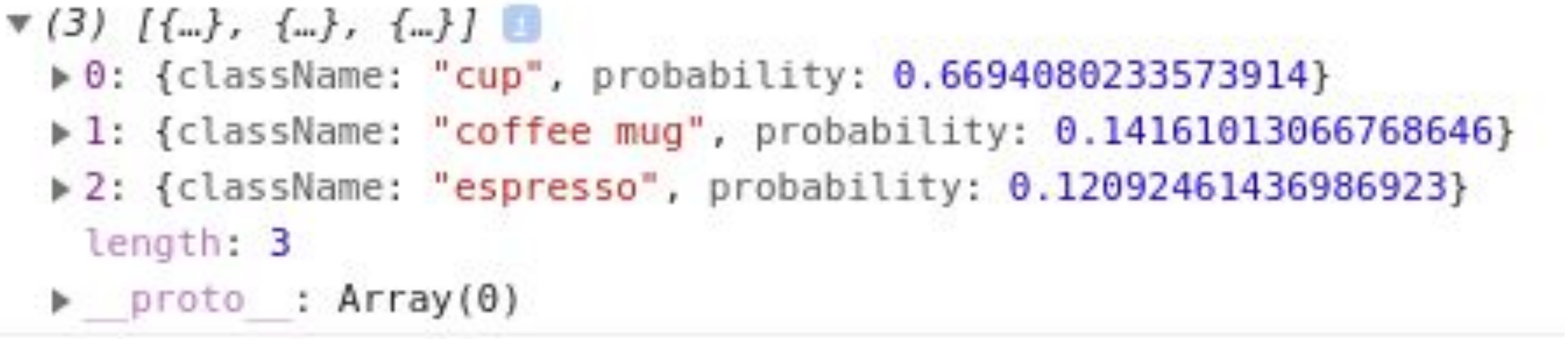
```
▼(3) [{…}, {…}, {…}]
  ▶0: {className: "cup", probability: 0.6694080233573914}
  ▶1: {className: "coffee mug", probability: 0.14161013066768646}
  ▶2: {className: "espresso", probability: 0.12092461436986923}
   length: 3
  ▶__proto__: Array(0)
>
```





```
// @tensorflow/tfjs Copyright 2019
!function(t,e){"object"==typeof exp
//# sourceMappingURL=tf.min.js.map
```

```
▼(3) [{…}, {…}, {…}]
  ▶0: {className: "cup", probability: 0.6694080233573914}
  ▶1: {className: "coffee mug", probability: 0.1416101306676846}
  ▶2: {className: "espresso", probability: 0.1209246143698692}
   length: 3
  ▶__proto__: Array(0)
>
```



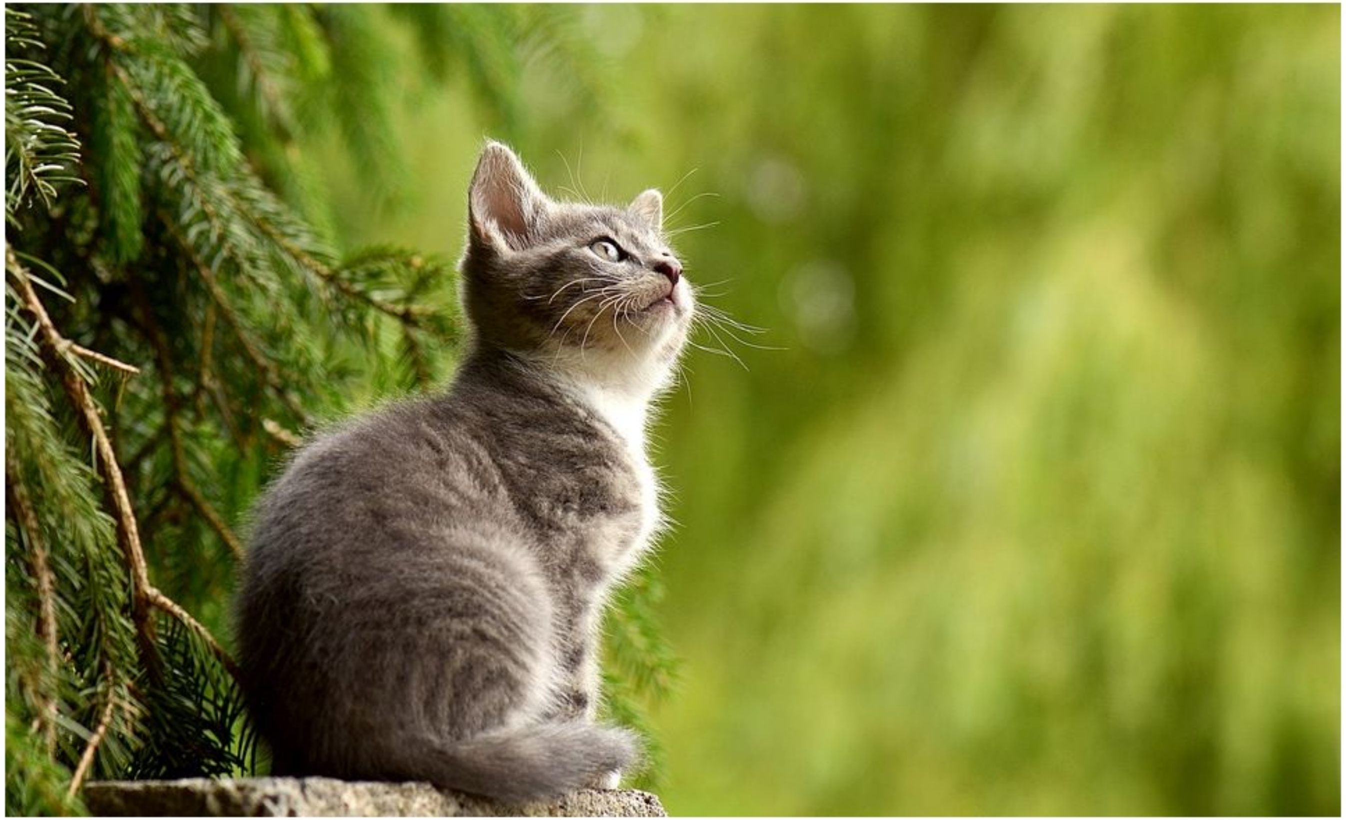
```
▼(3) [{…}, {…}, {…}]
  ▶0: {className: "cup", probability: 0.6694080233573914}
  ▶1: {className: "coffee mug", probability: 0.1416101306676846}
  ▶2: {className: "espresso", probability: 0.1209246143698692}
   length: 3
  ▶__proto__: Array(0)
```

```javascript
const img = document.getElementById('img');
const outp = document.getElementById('output');
mobilenet.load().then(model => {
  model.classify(img).then(predictions => {
    console.log(predictions);
    for(var i = 0; i<predictions.length; i++){
      outp.innerHTML += "<br/>" + predictions[i].className
                        + " : " + predictions[i].probability;
    }
  });
});
```

```
const img = document.getElementById('img');
const outp = document.getElementById('output');
mobilenet.load().then(model => {
  model.classify(img).then(predictions => {
    console.log(predictions);
    for(var i = 0; i<predictions.length; i++){
      outp.innerHTML += "<br/>" + predictions[i].className
                        + " : " + predictions[i].probability;
    }
  });
});
```

```
const img = document.getElementById('img');
const outp = document.getElementById('output');
mobilenet.load().then(model => {
  model.classify(img).then(predictions => {
    console.log(predictions);
    for(var i = 0; i<predictions.length; i++){
      outp.innerHTML += "<br/>" + predictions[i].className
                       + " : " + predictions[i].probability;
    }
  });
});
```

```
const img = document.getElementById('img');
const outp = document.getElementById('output');
mobilenet.load().then(model => {
  model.classify(img).then(predictions => {
    console.log(predictions);
    for(var i = 0; i<predictions.length; i++){
      outp.innerHTML += "<br/>" + predictions[i].className
                       + " : " + predictions[i].probability;
    }
  });
});
```



```
tiger cat : 0.44101303815841675
leopard, Panthera pardus : 0.18026068806648254
Madagascar cat, ring-tailed lemur, Lemur catta : 0.14320671558380127
```

```
!pip install tensorflowjs
```

```python
import numpy as np
import tensorflow as tf
from tensorflow import keras
print(tf.__version__)
model = tf.keras.Sequential([keras.layers.Dense(units=1, input_shape=[1])])
model.compile(optimizer='sgd', loss='mean_squared_error')
xs = np.array([-1.0,  0.0, 1.0, 2.0, 3.0, 4.0], dtype=float)
ys = np.array([-3.0, -1.0, 1.0, 3.0, 5.0, 7.0], dtype=float)
model.fit(xs, ys, epochs=500)
```

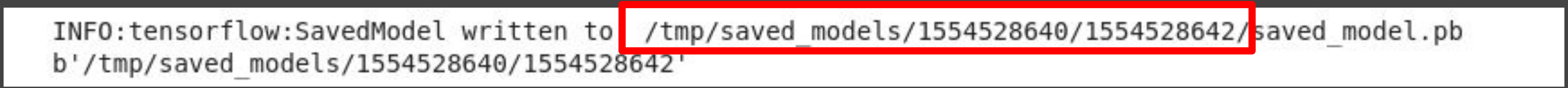
```python
print(model.predict([10.0]))
```

```
import time
saved_model_path = "/tmp/saved_models/{}".format(int(time.time()))

# For TensorFlow 2.0 use this:
# tf.keras.experimental.export_saved_model(model, saved_model_path)

# For TensorFlow 1.x use this:
tf.contrib.saved_model.save_keras_model(model, saved_model_path)
```

```python
import time
saved_model_path = "/tmp/saved_models/{}".format(int(time.time()))

# For TensorFlow 2.0 use this:
# tf.keras.experimental.export_saved_model(model, saved_model_path)

# For TensorFlow 1.x use this:
tf.contrib.saved_model.save_keras_model(model, saved_model_path)
```

```python
import time
saved_model_path = "/tmp/saved_models/{}".format(int(time.time()))

# For TensorFlow 2.0 use this:
# tf.keras.experimental.export_saved_model(model, saved_model_path)

# For TensorFlow 1.x use this:
tf.contrib.saved_model.save_keras_model(model, saved_model_path)
```

```python
import time
saved_model_path = "/tmp/saved_models/{}".format(int(time.time()))

# For TensorFlow 2.0 use this:
# tf.keras.experimental.export_saved_model(model, saved_model_path)

# For TensorFlow 1.x use this:
tf.contrib.saved_model.save_keras_model(model, saved_model_path)
```



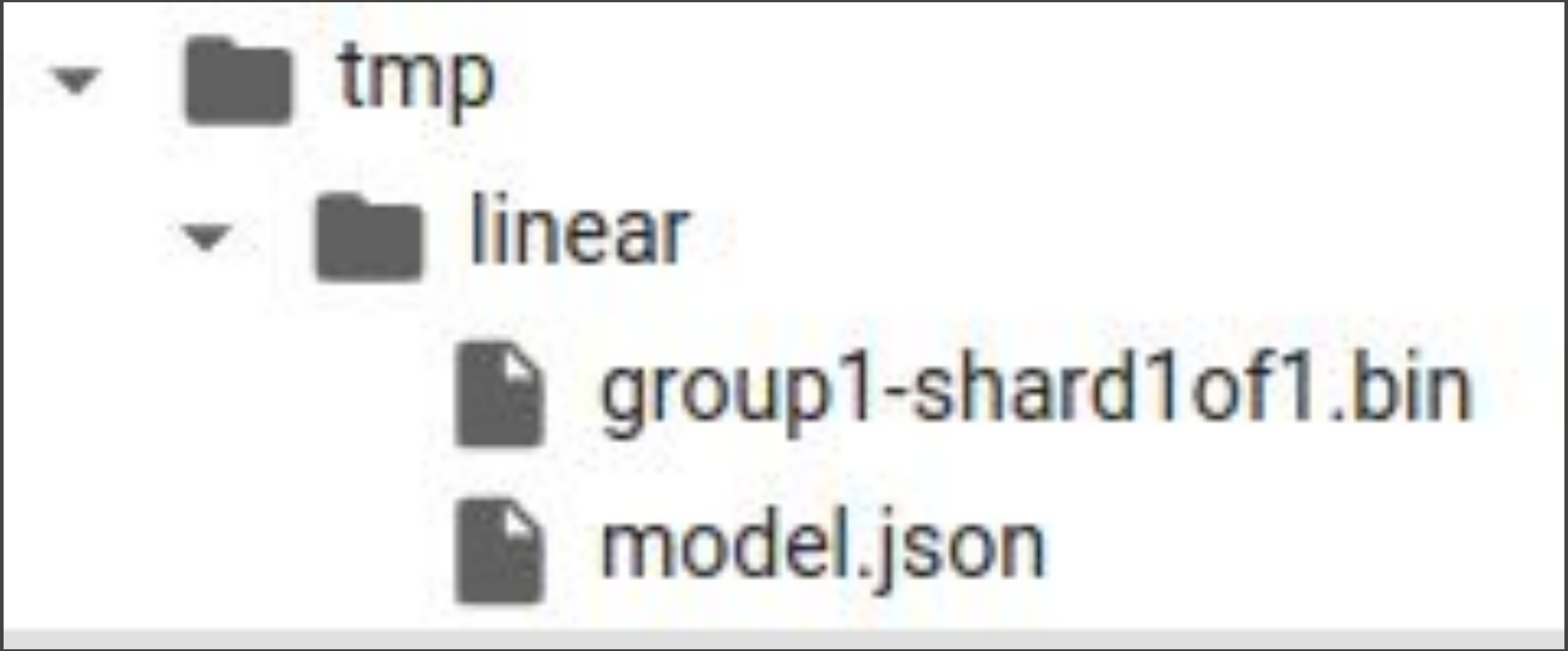
```
INFO:tensorflow:SavedModel written to  /tmp/saved_models/1554528640/1554528642/saved_model.pb
b'/tmp/saved_models/1554528640/1554528642'
```

```
!tensorflowjs_converter \
    --input_format=keras_saved_model \
    /tmp/saved_models/1554528640/1554528642 \
    /tmp/linear
```

```
!tensorflowjs_converter \
    --input_format=keras_saved_model \
    /tmp/saved_models/1554528640/1554528642 \
    /tmp/linear
```

```
!tensorflowjs_converter \
    --input_format=keras_saved_model \
    /tmp/saved_models/1554528640/1554528642 \
    /tmp/linear
```

```
!tensorflowjs_converter \
    --input_format=keras_saved_model \
    /tmp/saved_models/1554528640/1554528642 \
    /tmp/linear
```



```
▾ tmp
  ▾ linear
      group1-shard1of1.bin
      model.json
```

```html
<html>
<head>
<script src="https://cdn.jsdelivr.net/npm/@tensorflow/tfjs@latest"></script>
<script>
    async function run(){
        const MODEL_URL = 'http://127.0.0.1:8887/model.json';
        const model = await tf.loadLayersModel(MODEL_URL);
        console.log(model.summary());
        const input = tf.tensor2d([10.0], [1, 1]);
        const result = model.predict(input);
        alert(result);
    }
    run();
</script>
<body>
</body>
</html>
```

```
<html>
<head>
<script src="https://cdn.jsdelivr.net/npm/@tensorflow/tfjs@latest"></script>
<script>
    async function run(){
        const MODEL_URL = 'http://127.0.0.1:8887/model.json';
        const model = await tf.loadLayersModel(MODEL_URL);
        console.log(model.summary());
        const input = tf.tensor2d([10.0], [1, 1]);
        const result = model.predict(input);
        alert(result);
    }
    run();
</script>
<body>
</body>
</html>
```

```
<html>
<head>
<script src="https://cdn.jsdelivr.net/npm/@tensorflow/tfjs@latest"></script>
<script>
    async function run(){
        const MODEL_URL = 'http://127.0.0.1:8887/model.json';
        const model = await tf.loadLayersModel(MODEL_URL);
        console.log(model.summary());
        const input = tf.tensor2d([10.0], [1, 1]);
        const result = model.predict(input);
        alert(result);
    }
    run();
</script>
<body>
</body>
</html>
```

```
<html>
<head>
<script src="https://cdn.jsdelivr.net/npm/@tensorflow/tfjs@latest"></script>
<script>
    async function run(){
        const MODEL_URL = 'http://127.0.0.1:8887/model.json';
        const model = await tf.loadLayersModel(MODEL_URL);
        console.log(model.summary());
        const input = tf.tensor2d([10.0], [1, 1]);
        const result = model.predict(input);
        alert(result);
    }
    run();
</script>
<body>
</body>
</html>
```

```
<html>
<head>
<script src="https://cdn.jsdelivr.net/npm/@tensorflow/tfjs@latest"></script>
<script>
    async function run(){
        const MODEL_URL = 'http://127.0.0.1:8887/model.json';
        const model = await tf.loadLayersModel(MODEL_URL);
        console.log(model.summary());
        const input = tf.tensor2d([10.0], [1, 1]);
        const result = model.predict(input);
        alert(result);
    }
    run();
</script>
<body>
</body>
</html>
```

```
<html>
<head>
<script src="https://cdn.jsdelivr.net/npm/@tensorflow/tfjs@latest"></script>
<script>
    async function run(){
        const MODEL_URL = 'http://127.0.0.1:8887/model.json';
        const model = await tf.loadLayersModel(MODEL_URL);
        console.log(model.summary());
        const input = tf.tensor2d([10.0], [1, 1]);
        const result = model.predict(input);
        alert(result);
    }
    run();
</script>
<body>
</body>
</html>
```

```
<html>
<head>
<script src="https://cdn.jsdelivr.net/npm/@tensorflow/tfjs@latest"></script>
<script>
    async function run(){
        const MODEL_URL = 'http://127.0.0.1:8887/model.json';
        const model = await tf.loadLayersModel(MODEL_URL);
        console.log(model.summary());
        const input = tf.tensor2d([10.0], [1, 1]);
        const result = model.predict(input);
        alert(result);
    }
    run();
</script>
<body>
</body>
</html>
```