Neural Network based multiclass classification of iris flower

Aim:

The aim of this project is to predict the classification of the iris flower on the basis of its sepal length, sepal width, petal length, petal width using tensorflow.js.

Requirements:

Google chrome

Download link: https://www.google.com/intl/en_in/chrome/

Brackets (recommended) or you can use notepad++ or any other editor of your choice.

Brackets download link: http://brackets.io/

A web server

Download as chrome extension: https://chrome.google.com/webstore/detail/web-server-for-chrome/ofhbbkphhbklhfoeikjpcbhemlocgigb/related?hl=en

Dataset:

You can download the iris.csv dataset from this link:

https://github.com/Harshderp/Neural-Network-based-multiclass-classification-of-iris-flower/tree/master

Now let us code:

Open brackets software and start typing this code or you can download the iris.html file from the same link from where you have downloaded the dataset

Code:

```
<html>
        <head></head>
            <script src="https://cdn.jsdelivr.net/npm/@tensorflow/tfjs@latest"></script>
            <script lang="js">
                async function run(){
                    const csvUrl = 'iris.csv';
                    const trainingData = tf.data.csv(csvUrl, {
                        columnConfigs: {
                            species: {
                                isLabel: true
```

```
                }
            }
        });

        const numOfFeatures = (await trainingData.columnNames()).length - 1;
        const numOfSamples = 150;
        const convertedData =
              trainingData.map(({xs, ys}) => {
                  const labels = [
                        ys.species == "setosa" ? 1 : 0,
                        ys.species == "virginica" ? 1 : 0,
                        ys.species == "versicolor" ? 1 : 0
                  ]
                  return{ xs: Object.values(xs), ys: Object.values(labels)};
              }).batch(10);

        const model = tf.sequential();
        model.add(tf.layers.dense({inputShape: [numOfFeatures], activation:
"sigmoid", units: 5}))
        model.add(tf.layers.dense({activation: "softmax", units: 3}));

        model.compile({loss: "categoricalCrossentropy", optimizer:
tf.train.adam(0.06)});

        await model.fitDataset(convertedData,
                        {epochs:100,
                         callbacks:{
                             onEpochEnd: async(epoch, logs) =>{
                                 console.log("Epoch: " + epoch + " Loss: " +
logs.loss);
                             }
                         }});

        // Test Cases:

        // Setosa
        //const testVal = tf.tensor2d([4.4, 2.9, 1.4, 0.2], [1, 4]);

        // Versicolor
        const testVal = tf.tensor2d([6.4, 3.2, 4.5, 1.5], [1, 4]);

        // Virginica
        // const testVal = tf.tensor2d([5.8,2.7,5.1,1.9], [1, 4]);
```

```
                const prediction = model.predict(testVal);
                const pIndex = tf.argMax(prediction, axis=1).dataSync();

                const classNames = ["Setosa", "Virginica", "Versicolor"];

                // alert(prediction)
                alert(classNames[pIndex])

            }
            run();
        </script>
    <body>
    </body>
    </html>
```

****Copy the above code and paste in a file and name it as iris.html in a new folder and name this folder as tensorflow.js and keep it on your desktop (keep the iris.csv dataset also in the tensorflow.js folder)****

Code Explanation:

```
<html>
        <head></head>
            <script src="https://cdn.jsdelivr.net/npm/@tensorflow/tfjs@latest"></script>
            <script lang="js">
                async function run(){
                    const csvUrl = 'iris.csv';
                    const trainingData = tf.data.csv(csvUrl, {
                        columnConfigs: {
                            species: {
                                isLabel: true
                            }
                        }
                    });
```

In the above cell we are creating an html file and an html file is always start with <html> tag and <head>tag is used to add content to header and it is closed by </head> tag. We are using this script tag under which we are using an existing script i.e. tensorflow.js and we are importing it from internet.

After importing tensorflow.js we are creating our own function “async function run()” which will be our main function and it will help to load the data using tensorflow.js and we are declaring that our data is labeled.

```
const numOfFeatures = (await trainingData.columnNames()).length - 1;
const numOfSamples = 150;
const convertedData =
      trainingData.map(({xs, ys}) => {
          const labels = [
                ys.species == "setosa" ? 1 : 0,
                ys.species == "virginica" ? 1 : 0,
                ys.species == "versicolor" ? 1 : 0
          ]
          return{ xs: Object.values(xs), ys: Object.values(labels)};
      }).batch(10);
```

In this cell we are declaring the value for number of features, number of samples. Number of features means the number of columns in our data and number of samples means the number of rows in our data.

Then we are separating our training data where xs will be the independent features and ys be the dependent features.

```
const model = tf.sequential();
model.add(tf.layers.dense({inputShape: [numOfFeatures], activation: "sigmoid",
units: 5}))
model.add(tf.layers.dense({activation: "softmax", units: 3}));

model.compile({loss: "categoricalCrossentropy", optimizer: tf.train.adam(0.06)});

await model.fitDataset(convertedData,
                {epochs:100,
                 callbacks:{
                     onEpochEnd: async(epoch, logs) =>{
                         console.log("Epoch: " + epoch + " Loss: " + logs.loss);
                     }
                 }});
```

Here we are declaring that our model will be sequential with activation function sigmoid and input will be number of features and then after setting the layers we are training our model.

```
            // Test Cases:

            // Setosa
            //const testVal = tf.tensor2d([4.4, 2.9, 1.4, 0.2], [1, 4]);

            // Versicolor
            const testVal = tf.tensor2d([6.4, 3.2, 4.5, 1.5], [1, 4]);

            // Virginica
            // const testVal = tf.tensor2d([5.8,2.7,5.1,1.9], [1, 4]);

            const prediction = model.predict(testVal);
            const pIndex = tf.argMax(prediction, axis=1).dataSync();

            const classNames = ["Setosa", "Virginica", "Versicolor"];

            // alert(prediction)
            alert(classNames[pIndex])

        }
        run();
    </script>
<body>
</body>
</html>
```

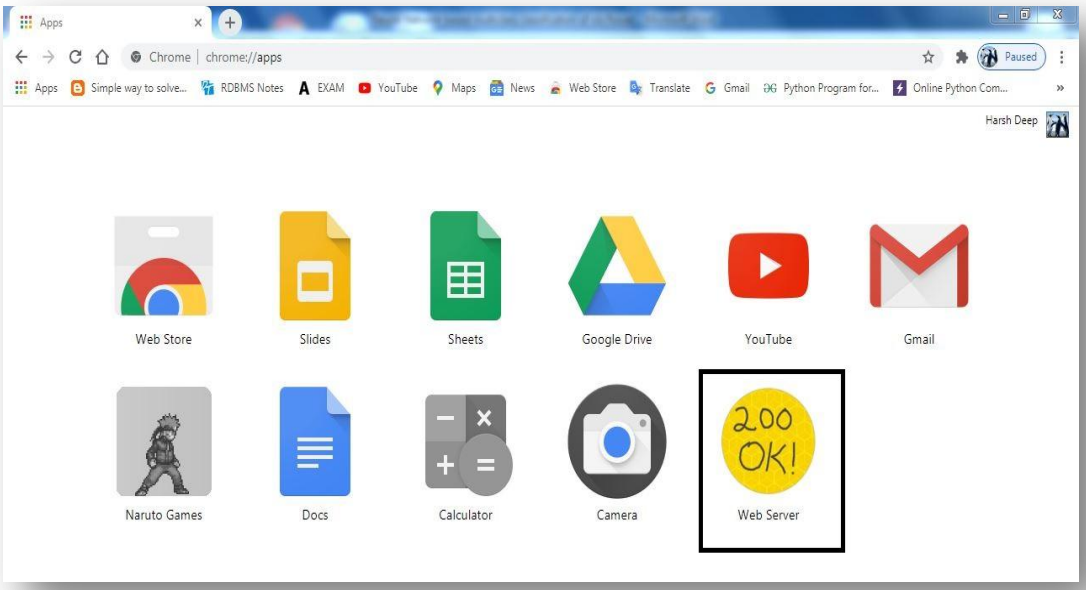
Here we are defining some test cases to test our model and you can remove // sign to test a test case and after that we are sending this prediction to be shown as alert message in the web browser, then the run function will run our async function run() and then we are closing our script using </script> and then <body> Between these tags you can write/add any paragraph using tags and you can add any image using <img> tag</body> and to end this page simply use </html> tag.

How to run this project:

****Keep your iris.html (code) and iris.csv (dataset) in a folder let say tensorflow.js (folder name) on desktop****

Below are the steps required to run this project successfully:

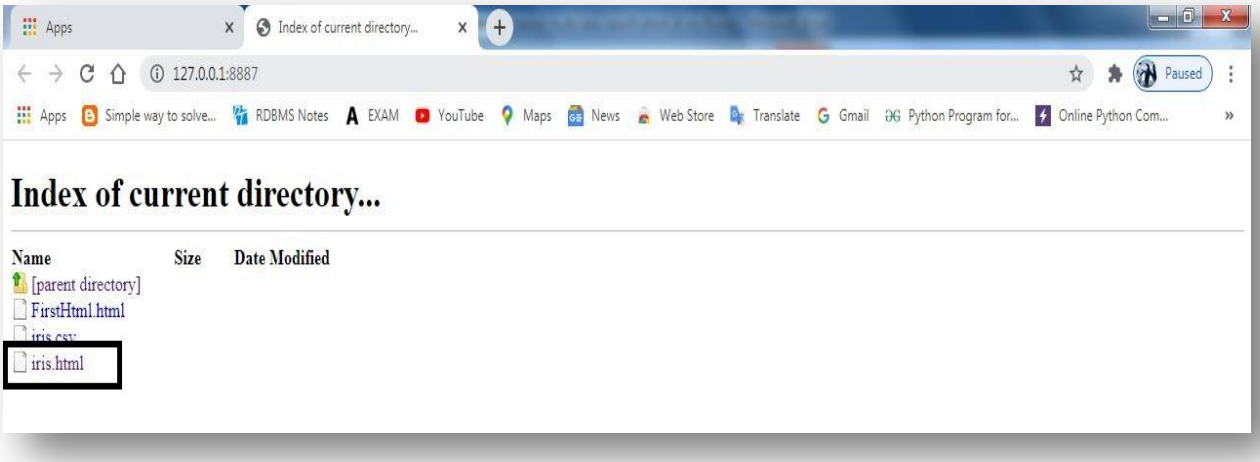
1. Open google chrome and search chrome://apps and click on this highlighted icon (this is the web server that we have downloaded and added as chrome extension.



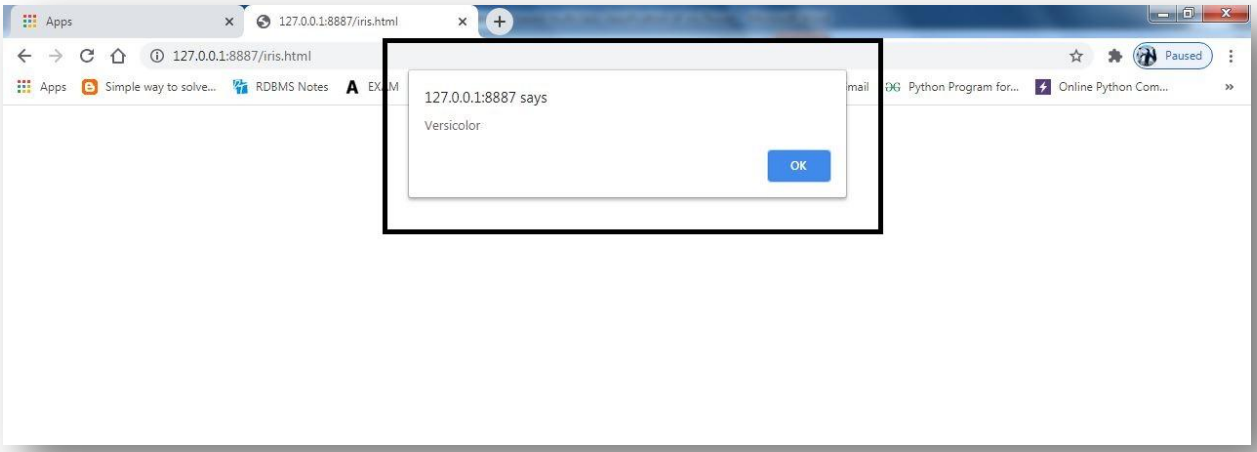
2. Now click on the above highlighted icon and choose the folder in which you have saved the dataset and html file.

3. Now click on the link which is highlighted with a circle

4. After clicking on the above link you will be redirected to a webpage

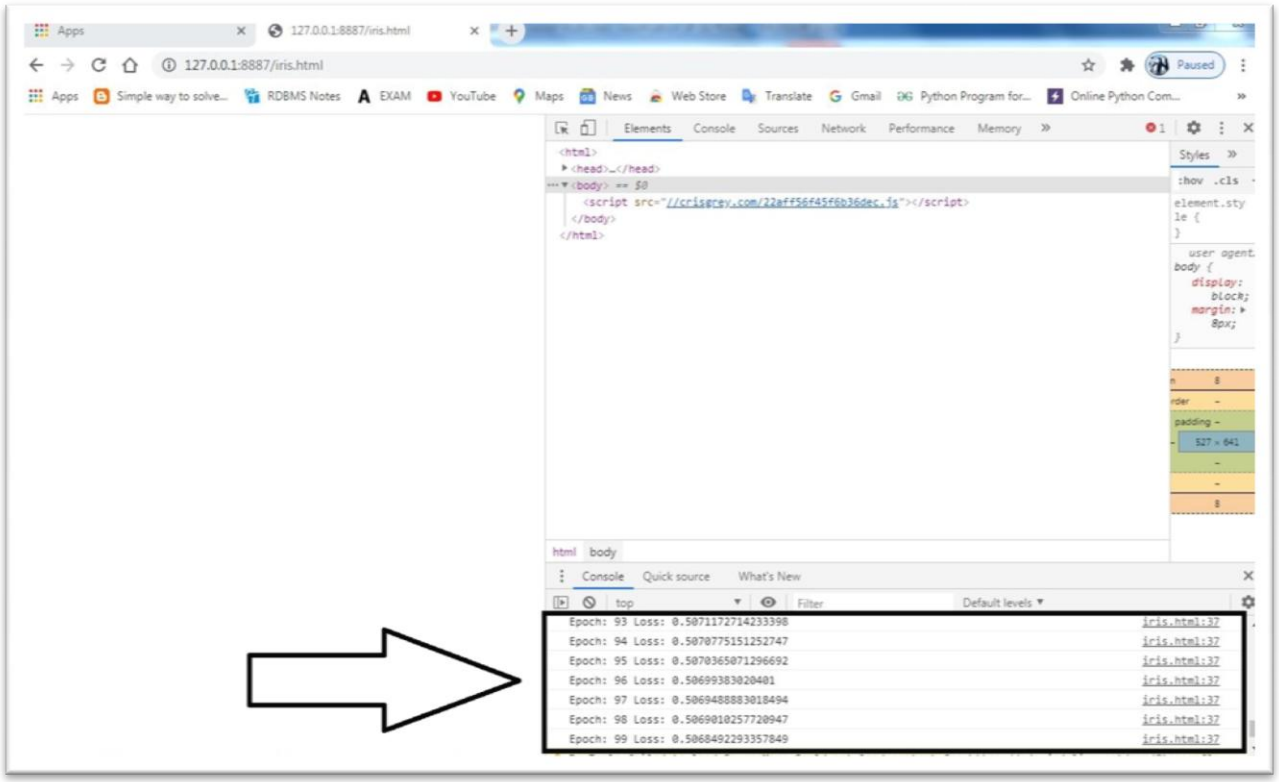


5. Now just click on the iris.html file and you will be redirecting to a new webpage and after waiting for 10 seconds you can see a message will be there like this:



This alerted message is the result of your prediction.

6. You can see the epochs running in the background by going to Settings->Developer mode



What else you can do to make this project more innovative and interactive?

As a developer I always want that my project should be interactive and innovative.

I always try finding new things/ideas by which I can make my project awesome.

Until now you may have seen that our project is working fine but there is nothing on the page (iris.html) like pictures, paragraphs, heading or anything else which makes this project boring.

Also w have to open chrome the open the web server and then open the file and it is time consuming as you see.

Now, if you are a curious developer/student like me then I think you should do two improvements in this project to make it so interactive and innovative:

- You can develop a beautiful frontend of this project to make it interactive
- Secondly, you should find out the solution that how can we reduce the steps to run this project/ how can you automate the process of running of this project.