University of Science and Technology Bannu

Deep Learning

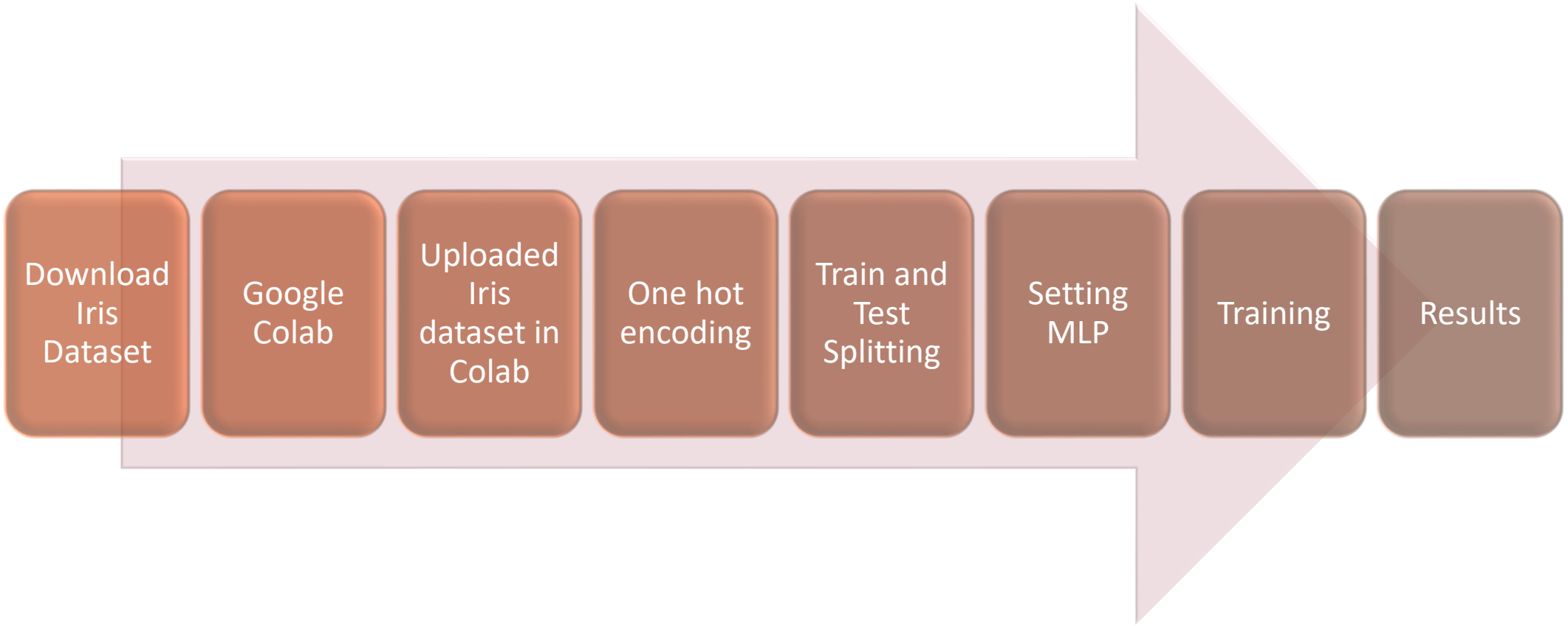
Lesson 4

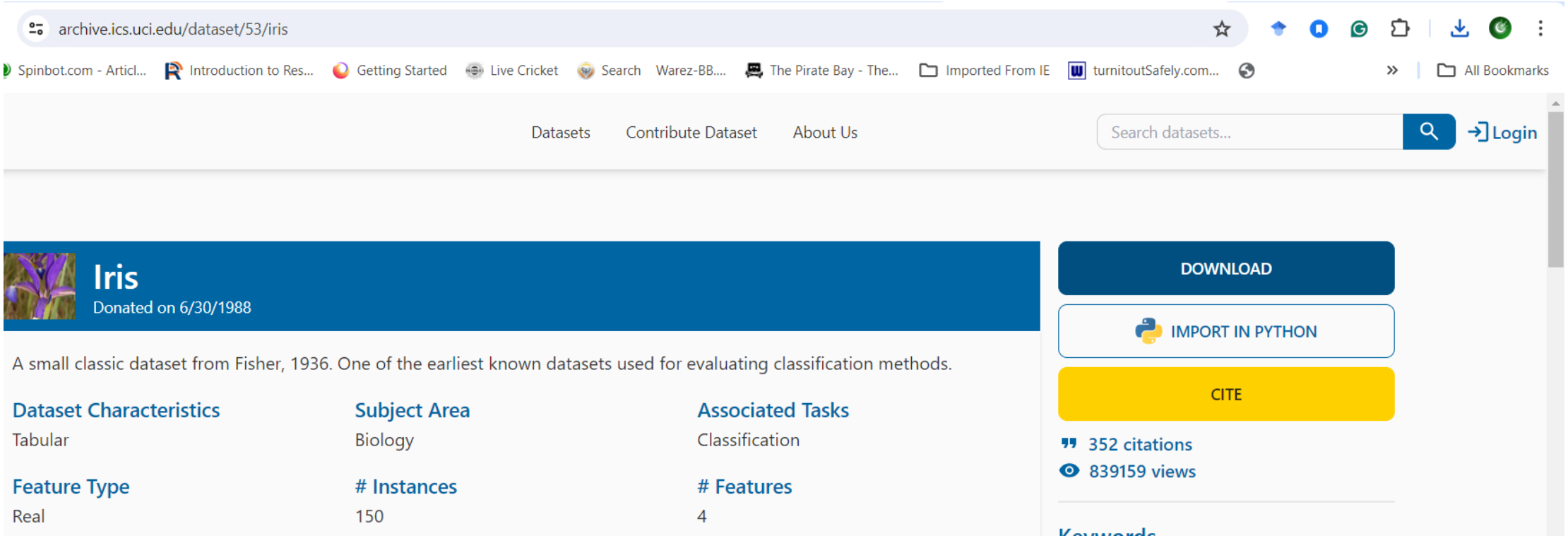
May 14, 2024

Learning Objectives

Practicable Demo of MLP on Iris Dataset

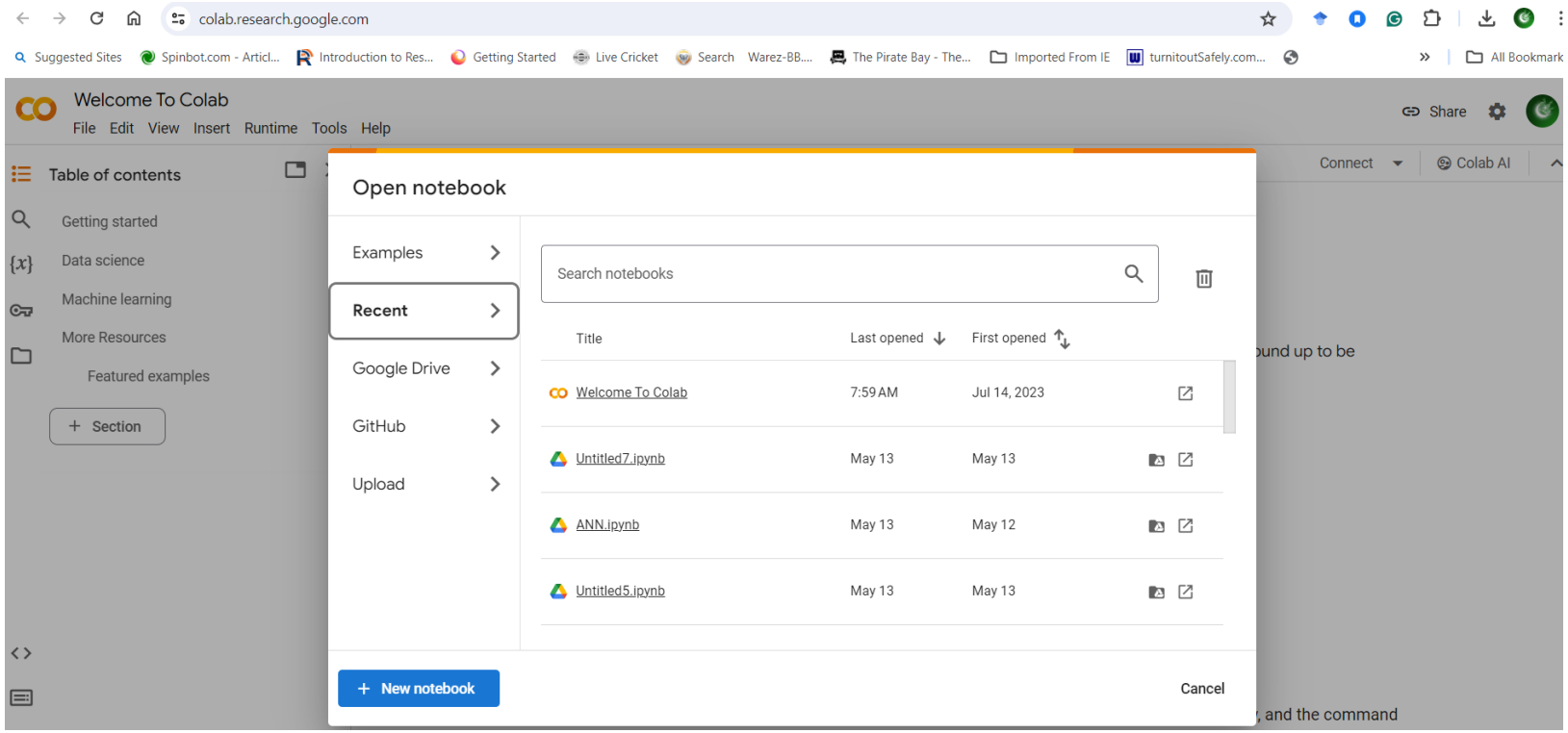
Steps


Download Iris Dataset
Google Colab
Uploaded Iris dataset in Colab
One hot encoding
Train and Test Splitting
Setting MLP
Training
Results



Step-1

DOWNLOAD THE IRIS DATASET UCI OR KAGGLE

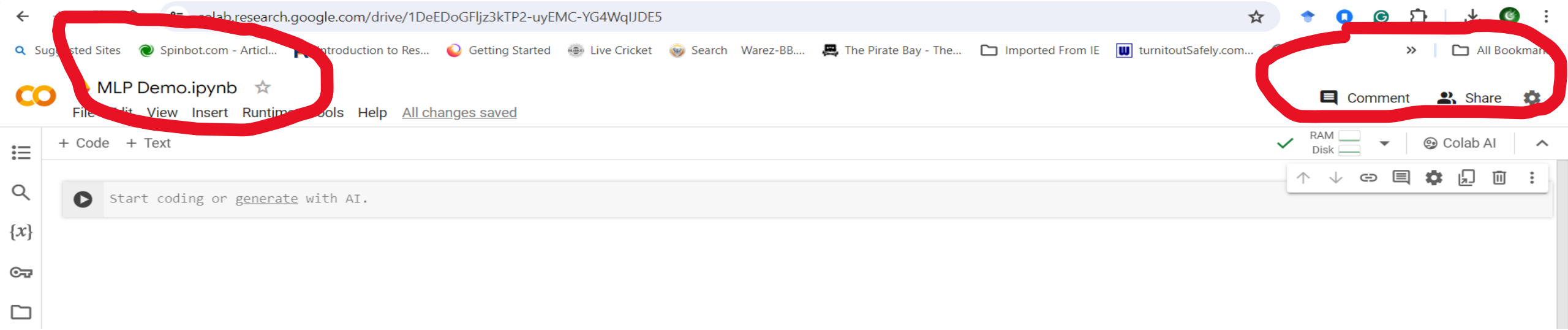


Step-2

SETTING COLAB ENVIRNMENT

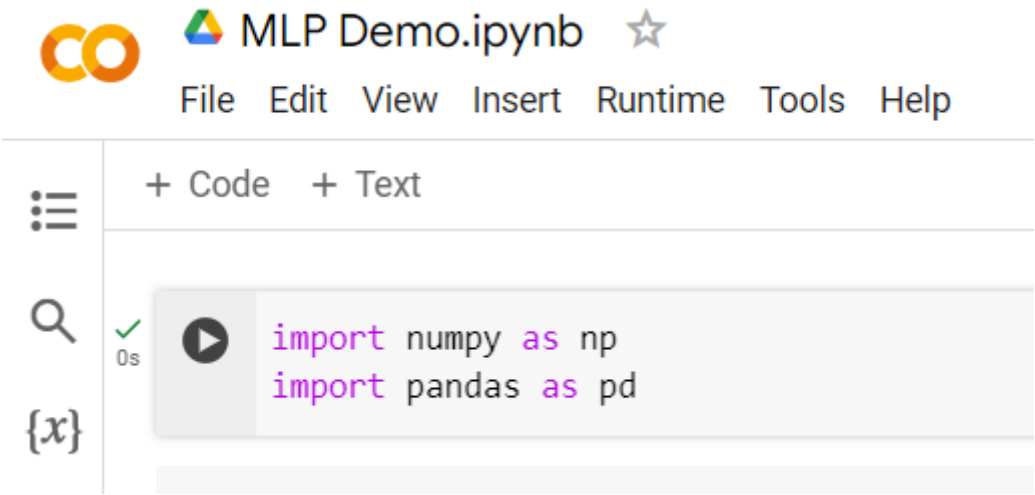
Step-3

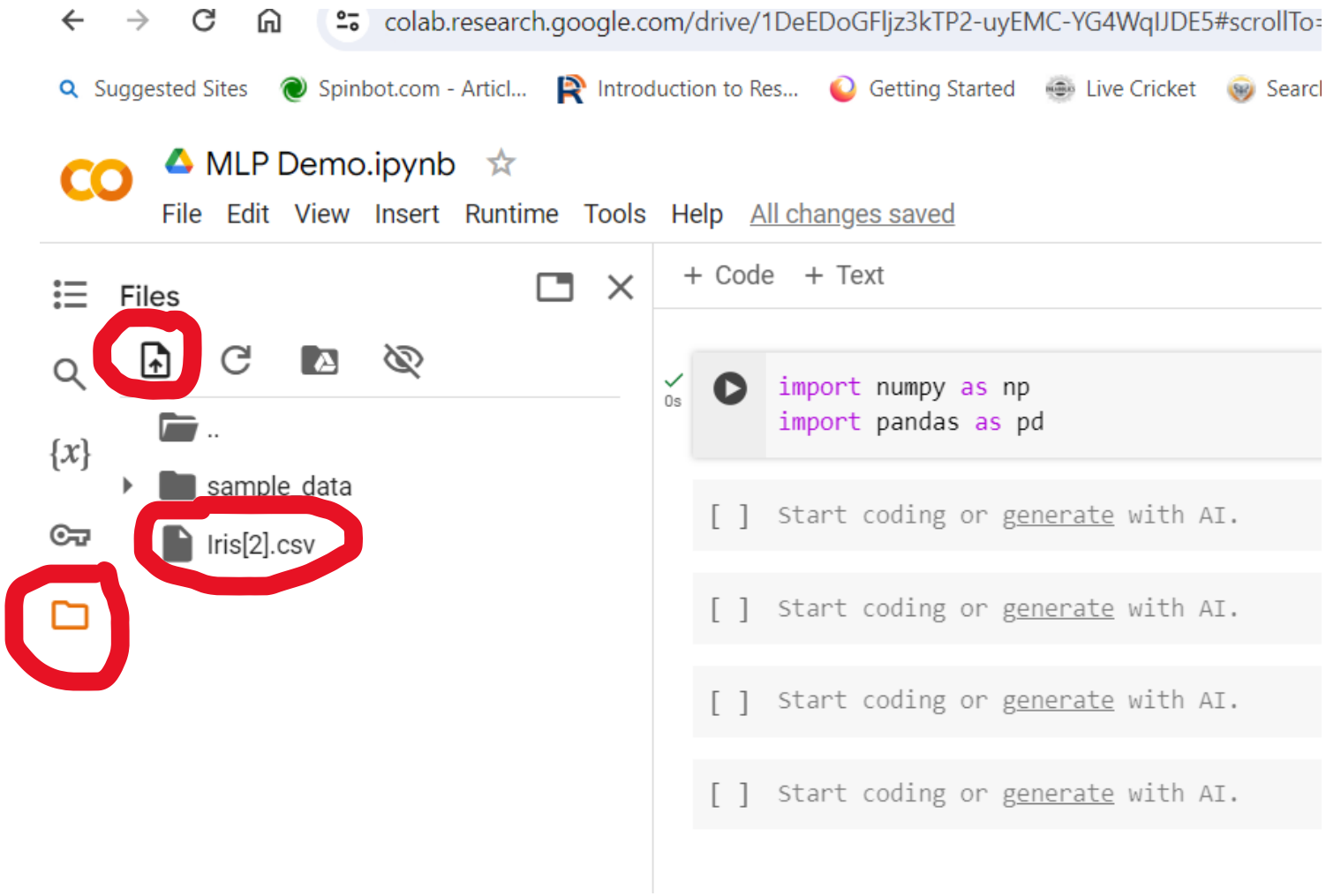
Give name and connect the colab



Step-4

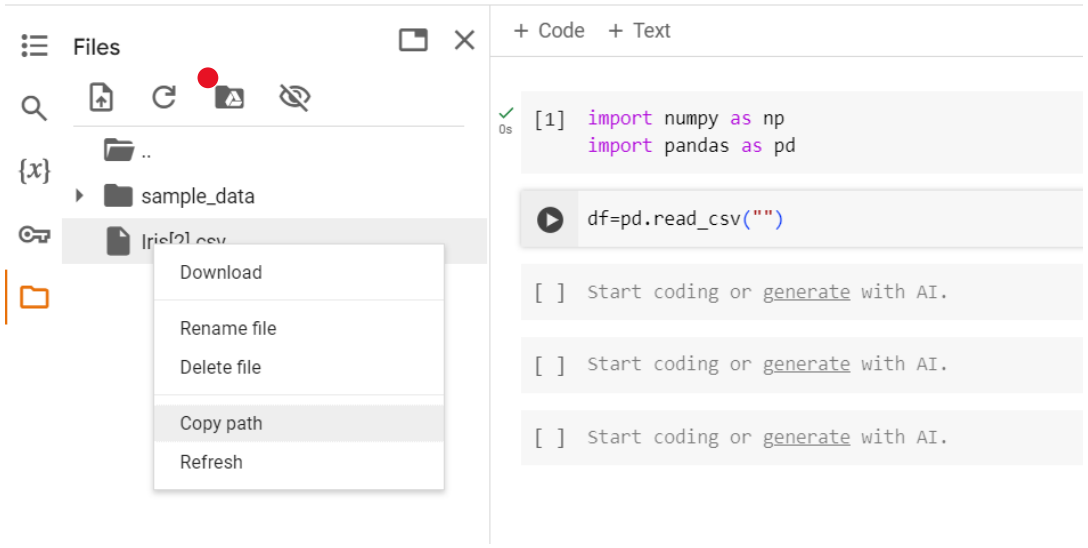
Start Coding





Step-5

UPLOAD THE IRIS DATASET



Step-6

READ THE DATASET

```
df.head(15)
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa
5	6	5.4	3.9	1.7	0.4	Iris-setosa

Step-7

DISPLAY THE DATA

```python
#one Hot Encoding
y=pd.get_dummies(df["Species"])
y
```

	Iris-setosa	Iris-versicolor	Iris-virginica
131	False	False	True
60	False	True	False
139	False	False	True
96	False	True	False
10	True	False	False
...	...	...	...

```python
x=df.drop(["Id","Species"],axis=1)
x
```

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
131	7.9	3.8	6.4	2.0
60	5.0	2.0	3.5	1.0
139	6.9	3.1	5.4	2.1
96	5.7	2.9	4.2	1.3
10	5.4	3.7	1.5	0.2
...	...	...	...	...
129	7.2	3.0	5.8	1.6
114	5.8	2.8	5.1	2.4

Step-8

ONE HOT ENCODING

Step-9

Setting Train and Test Split using Sklearn

Next steps: Generate code with x | View recommended plots

0s

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test=train_test_split(x,y,test_size=0.3)
```

step=10

Setting the Neural Network (MLP)

```
[11] from sklearn.model_selection import train_test_split
     x_train, x_test, y_train, y_test=train_test_split(x,y,test_size=0.3)
```

```
from keras.models import Sequential
from keras.layers import Dense
```

Step-11

```
model=Sequential()
model.add(Dense(6,activation="sigmoid")) #Hidden Layer
model.add(Dense(3,activation="softmax"))#uotput Layer
#model.compile(loss="categorical_crossentropy", matrics=["accuracy"])
model.compile(loss="categorical_crossentropy",metrics=["accuracy"])
model.fit(x_train,y_train,epochs=25, batch_size=5)
```

```
Epoch 1/25
21/21 [==============================] - 1s 2ms/step - loss: 1.3570 - accuracy: 0.2857
Epoch 2/25
21/21 [==============================] - 0s 2ms/step - loss: 1.2883 - accuracy: 0.2857
Epoch 3/25
21/21 [==============================] - 0s 2ms/step - loss: 1.2343 - accuracy: 0.2857
Epoch 4/25
21/21 [==============================] - 0s 2ms/step - loss: 1.1832 - accuracy: 0.5048
Epoch 5/25
21/21 [==============================] - 0s 2ms/step - loss: 1.1389 - accuracy: 0.6857
Epoch 6/25
21/21 [==============================] - 0s 2ms/step - loss: 1.1045 - accuracy: 0.6857
Epoch 7/25
21/21 [==============================] - 0s 2ms/step - loss: 1.0781 - accuracy: 0.6857
Epoch 8/25
21/21 [==============================] - 0s 2ms/step - loss: 1.0552 - accuracy: 0.6857
```

Step-12

Printing the Result

```
score=model.evaluate(x_test,y_test)
print("Accuracy:",score)
```

```
2/2 [==============================] - 0s 7ms/step - loss: 1.0690 - accuracy: 0.5333
Accuracy: [1.0689753293991089, 0.5333333611488342]
```