



# University of Science and Technology Bannu

---

## Deep Learning

### Lesson 4

May 14, 2024

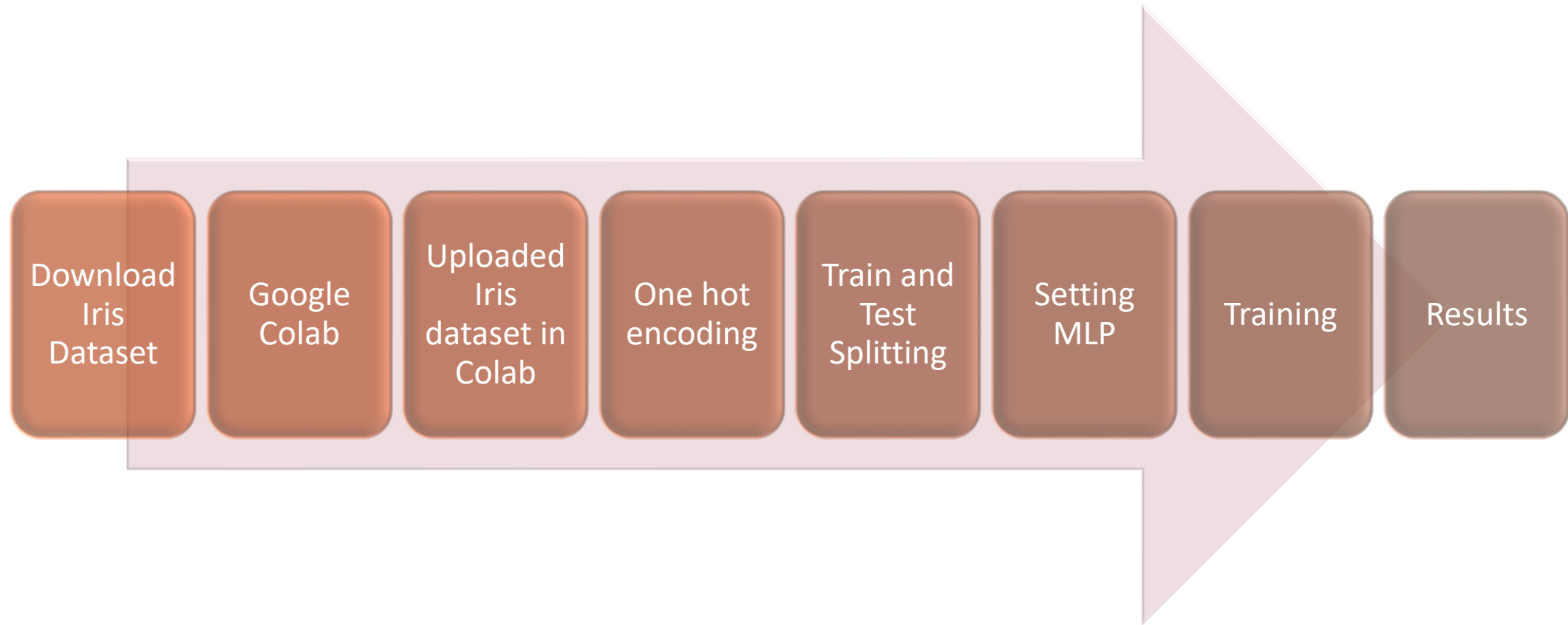
---

**Practicable Demo of MLP on Iris Dataset**

# Learning Objectives

# Steps

---



archive.ics.uci.edu/dataset/53/iris

Spinbot.com - Articl... Introduction to Res... Getting Started Live Cricket Search Warez-BB... The Pirate Bay - The... Imported From IE turnoutSafely.com... All Bookmarks

Datasets Contribute Dataset About Us

Search datasets... Login

# Iris

Donated on 6/30/1988

A small classic dataset from Fisher, 1936. One of the earliest known datasets used for evaluating classification methods.

Dataset Characteristics	Subject Area	Associated Tasks
Tabular	Biology	Classification
Feature Type	# Instances	# Features
Real	150	4

**DOWNLOAD**

**IMPORT IN PYTHON**

**CITE**

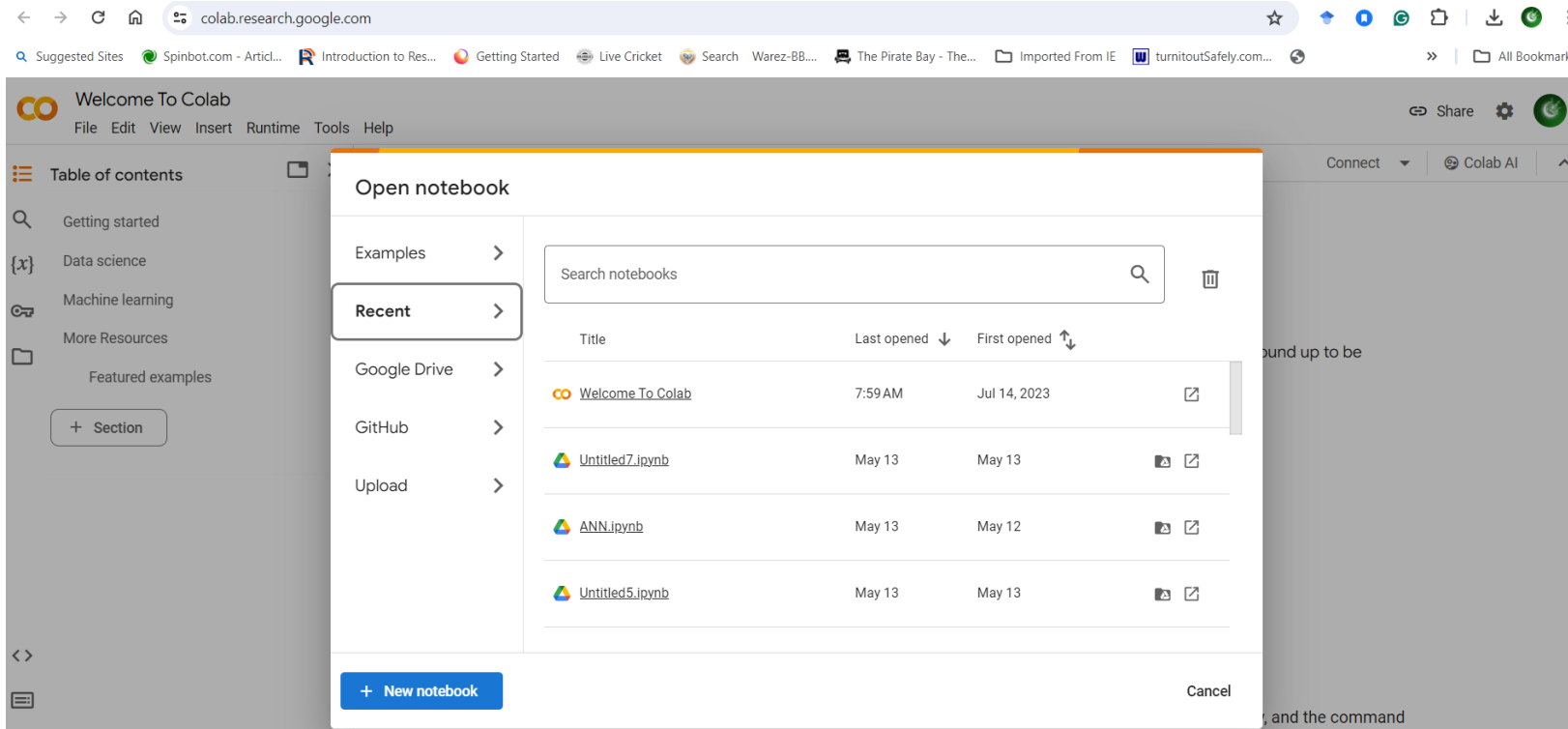
352 citations

839159 views

Keywords

# Step-1

DOWNLOAD THE IRIS DATASET UCI OR KAGGLE



# Step-2

## SETTING COLAB ENVIRNMENT

# Step-3

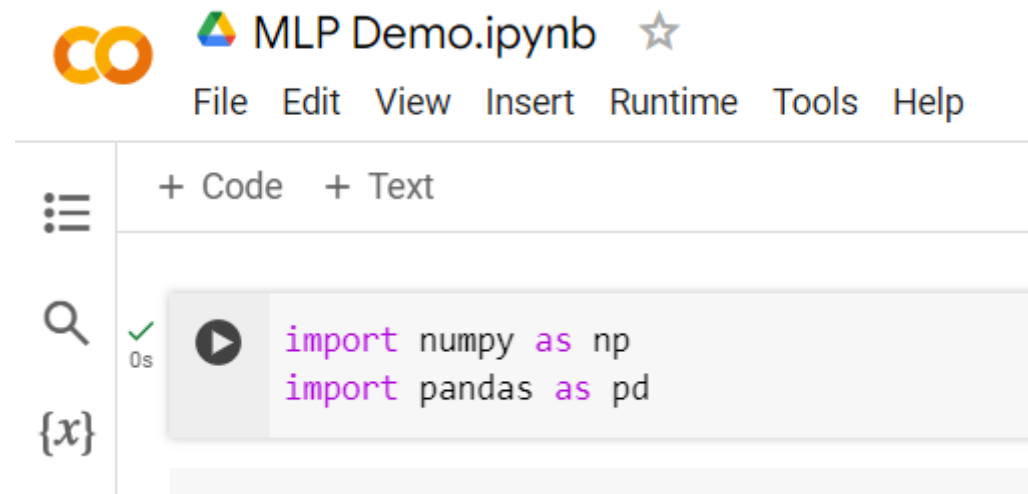
Give name and connect the colab

The screenshot displays the Google Colab web interface. The browser's address bar shows the URL: `colab.research.google.com/drive/1DeEDoGFljz3kTP2-uyEMC-YG4WqIJDE5`. The file name **MLP Demo.ipynb** is visible in the top left of the Colab editor. The menu bar includes **File**, **Edit**, **View**, **Insert**, **Runtime**, **Tools**, **Help**, and **All changes saved**. On the right side, the **Comment** and **Share** buttons are highlighted with a red circle. The bottom right corner shows the **RAM** and **Disk** usage indicators, along with the **Colab AI** logo. The main code area contains a prompt: **start coding or generate with AI.**

# Step-4

---

Start Coding





← → ↻ 🏠 [colab.research.google.com/drive/1DeEDoGFljz3kTP2-uyEMC-YG4WqIJDE5#scrollTo=](https://colab.research.google.com/drive/1DeEDoGFljz3kTP2-uyEMC-YG4WqIJDE5#scrollTo=)

🔍 Suggested Sites 🌐 Spinbot.com - Articl... 📄 Introduction to Res... 🌈 Getting Started 🏏 Live Cricket 🔍 Search

CO MLP Demo.ipynb ☆

File Edit View Insert Runtime Tools Help [All changes saved](#)

Files

🔍 📁 🔄 🗑️

{x}

..

sample data

🔑 Iris[2].csv

📁

+ Code + Text

✓ 0s ▶ `import numpy as np`  
`import pandas as pd`

[ ] Start coding or [generate](#) with AI.

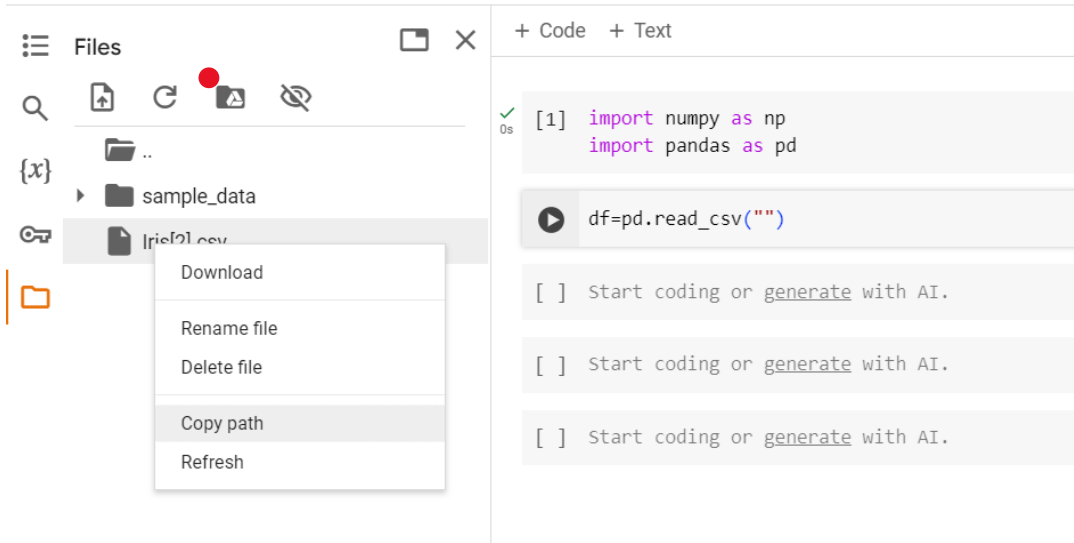
[ ] Start coding or [generate](#) with AI.

[ ] Start coding or [generate](#) with AI.

[ ] Start coding or [generate](#) with AI.

# Step-5

UPLOAD THE IRIS  
DATASET



# Step-6

READ THE DATASET

✓  
0s



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



	<b>Id</b>	<b>SepalLengthCm</b>	<b>SepalWidthCm</b>	<b>PetalLengthCm</b>	<b>PetalWidthCm</b>	<b>Species</b>
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

✓  
0s

```
#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
...	...	...	...



✓  
0s

```
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)

✓  
0s [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)`

✓  
0s  `from keras.models import Sequential`  
`from keras.layers import Dense`

# Step-11

✓  
31s



```
model=Sequential()  
model.add(Dense(6,activation="sigmoid")) #Hidden Layer  
model.add(Dense(3,activation="softmax"))#uotput Layer  
#model.compile(loss="categorical_crossentropy", metrics=["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

✓  
js



```
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]
```