

i love pdf - Yahoo India Search | X | Download file | iLovePDF | X | FundamentalsOfMachineLearn | X | Harishma0523/FOML | X | +

colab.research.google.com/drive/1SdqBF8cPKJS7sJqrzuJM-Os1HcENosz?authuser=1#scrollTo=oLp9svLobhTd

Commands + Code + Text Run all ⌂

Connect ⌂

```
Training size: (1097, 4)
Testing Size: (275, 4)

• Activation Function: RELU

Confusion Matrix:
[[148  0]
 [ 0 127]]

Classification Report:
precision    recall   f1-score   support
          0       1.00      1.00      1.00      148
          1       1.00      1.00      1.00      127

accuracy                           1.00      275
macro avg       1.00      1.00      1.00      275
weighted avg    1.00      1.00      1.00      275

• Activation Function: LOGISTIC

Confusion Matrix:
[[148  0]
 [ 1 126]]

Classification Report:
precision    recall   f1-score   support
          0       0.99      1.00      1.00      148
          1       1.00      0.99      1.00      127

accuracy                           1.00      275
macro avg       1.00      1.00      1.00      275
weighted avg    1.00      1.00      1.00      275
```

Variables Terminal

Finance headline  
India Posts Reco...

Search

ENG IN 15:41 18-11-2025

```
i love pdf - Yahoo India Search | X | Download file | iLovePDF | X | FundamentalsOfMachineLearn | X | Harishma0523/FOML | X | +  
colab.research.google.com/drive/1SdqBF8cPKJS7sJqrzuJM-Os1HcENosz?authuser=1#scrollTo=oLp9svLobhTd  
Commands + Code + Text Run all ⚡ Connect 🔍
```

Activation Function: TANH

Confusion Matrix:  
[[148 0]  
 [ 0 127]]

Classification Report:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	148
1	1.00	1.00	1.00	127

accuracy 1.00  
macro avg 1.00 1.00 1.00 275  
weighted avg 1.00 1.00 1.00 275

Activation Function: IDENTITY

Confusion Matrix:  
[[146 2]  
 [ 2 125]]

Classification Report:

	precision	recall	f1-score	support
0	0.99	0.99	0.99	148
1	0.98	0.98	0.98	127

accuracy 0.99  
macro avg 0.99 0.99 0.99 275  
weighted avg 0.99 0.99 0.99 275

Execution Completed Successfully!

```
Variables Terminal 15:41  
Finance headline ENG  
India Posts Reco... IN 18-11-2025
```

The screenshot shows a Jupyter Notebook interface with a code cell containing Python code for a logistic regression model. The code imports numpy, defines inputs (0, 1, 1, 0) and outputs (0, 0, 1, 1), and initializes weights (0.1, 0.3). It includes sigmoid and sigmoid derivative functions, a training loop for 1500 epochs, and a predict function. The notebook also displays variables and a terminal.

```
import numpy as np
inputs = np.array([[0, 0],
                  [0, 1],
                  [1, 0],
                  [1, 1]])
outputs = np.array([0, 0, 1, 1])
weights = np.array([0.1,
                   0.3])
bias = 0.2
learning_rate = 0.05
def sigmoid(x):
    return 1 / (1 + np.exp(-x))
def sigmoid_derivative(x):
    return x * (1 - x)
for epoch in range(15000):
    weighted_sum = np.dot(inputs, weights) + bias
    predicted_output = sigmoid(weighted_sum)
    error = outputs - predicted_output
    d_predicted = error * sigmoid_derivative(predicted_output)
    weights += learning_rate * np.dot(inputs.T, d_predicted)
    bias += learning_rate * np.sum(d_predicted)
print("Training Complete")
print("Final Weights:", weights)
print("Final Bias:", bias)
def predict(x1, x2):
    x = np.array([x1, x2])
    result = sigmoid(np.dot(x, weights) + bias)
    return result
print("\n--- Predictions ---")
print("[0, 0] ->", predict(0, 0))
print("[0, 1] ->", predict(0, 1))
print("[1, 0] ->", predict(1, 0))
```

The screenshot shows a Jupyter Notebook interface with the following code and output:

```
return x * (1 - x)
for epoch in range(15000):
    weighted_sum = np.dot(inputs, weights) + bias
    predicted_output = sigmoid(weighted_sum)
    error = outputs - predicted_output
    d_predicted = error * sigmoid_derivative(predicted_output)
    weights += learning_rate * np.dot(inputs.T, d_predicted)
    bias += learning_rate * np.sum(d_predicted)
print("Training Complete")
print("Final Weights:", weights)
print("Final Bias:", bias)
def predict(x1, x2):
    x = np.array([x1, x2])
    result = sigmoid(np.dot(x, weights) + bias)
    return result
print("\n--- Predictions ---")
print("[0, 0] =>", predict(0, 0))
print("[0, 1] =>", predict(0, 1))
print("[1, 0] =>", predict(1, 0))
print("[1, 1] =>", predict(1, 1))
```

Output:

```
Training Complete
Final Weights:
[[5.13667423]
 [5.13667425]]
Final Bias: -7.7984793971029855
--- Predictions ---
[0, 0] => [0.00041019]
[0, 1] => [0.06522853]
[1, 0] => [0.06522853]
[1, 1] => [0.92227519]
```



15:40

18-11-2025