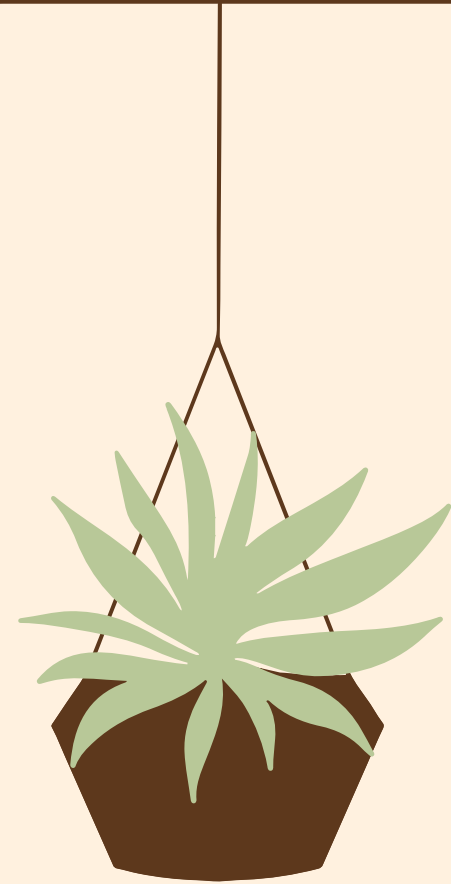


NEURO COMPUTING

HEART DISEASE

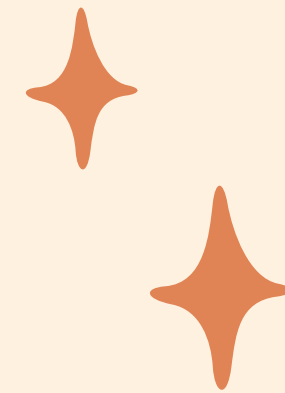
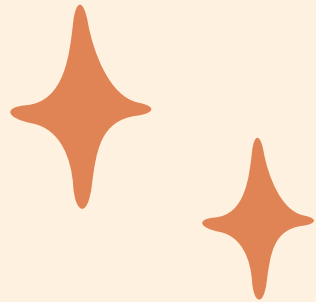
Reyna Aisyana | 3321600002





PEMODELAN 1

Single Layer Perceptron



IMPORT LIBRARIES

```
# IMPORT LIBRARY
import math
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import warnings
from sklearn.linear_model import Perceptron
from sklearn.neural_network import MLPClassifier as MLP
from sklearn.metrics import classification_report, accuracy_score
from importlib import reload

warnings.filterwarnings('ignore')
```

✓ 0.4s

Python

IMPORT DATASET

```
# READ DATASET
dataset = pd.read_csv('heart.csv')
data = dataset.iloc[:,0:-1]
label = dataset.iloc[:, -1]
```

DATA :

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2
...
298	57	0	0	140	241	0	1	123	1	0.2	1	0	3
299	45	1	3	110	264	0	1	132	0	1.2	1	0	3
300	68	1	0	144	193	1	1	141	0	3.4	1	2	3
301	57	1	0	130	131	0	1	115	1	1.2	1	1	3
302	57	0	1	130	236	0	0	174	0	0.0	1	1	2

303 rows × 13 columns

LABEL :

0	1
1	1
2	1
3	1
4	1
...	...
298	0
299	0
300	0
301	0
302	0

PEMODELAN

```
# MODEL SINGLE LAYER PERCEPTRON
model_slf = Perceptron(tol=None, random_state=60, penalty='l1')
model_slf.fit(data, label)

# Memprediksi Label
predicted_label = model_slf.predict(data)

# Evaluasi Model
print("Score : ", model_slf.score(data, label))
report = classification_report(label, predicted_label)
print("Classification Report:")
print(report)
```

Score : 0.834983498349835


Classification Report:

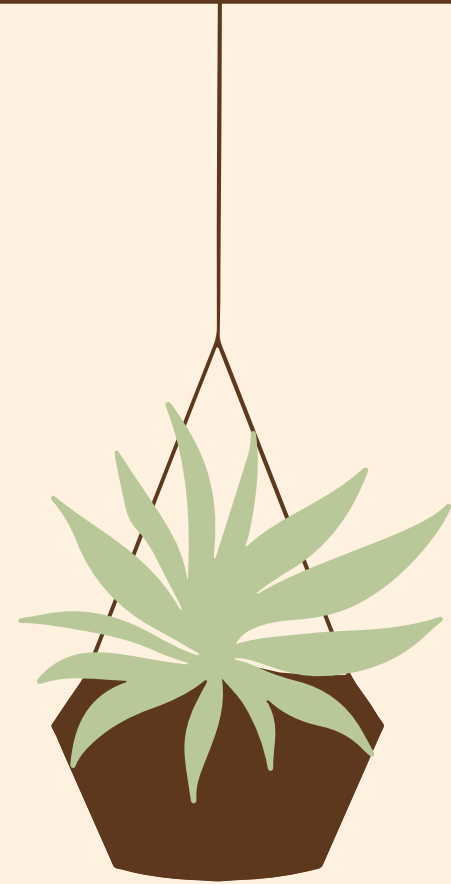
	precision	recall	f1-score	support
0	0.91	0.71	0.80	138
1	0.79	0.94	0.86	165
accuracy			0.83	303
macro avg	0.85	0.82	0.83	303
weighted avg	0.85	0.83	0.83	303



ANALISIS

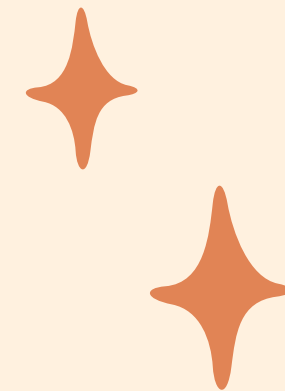
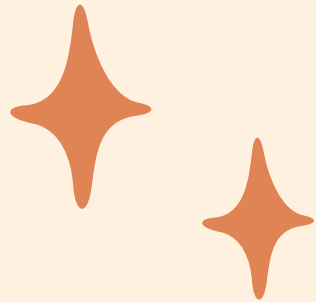
Model Single Layer Perceptron merupakan model yang sederhana dan linear dalam pemisahan kelas. Akurasi 83% menunjukkan bahwa model Single Layer Perceptron berhasil mengklasifikasikan sebagian besar data dengan benar, tetapi perlu diperhatikan untuk tugas klinis yang sensitif seperti ini, akurasi yang lebih tinggi mungkin diperlukan. Sehingga untuk masalah klasifikasi penyakit jantung yang kompleks, mungkin diperlukan model yang lebih canggih seperti jaringan saraf tiruan dengan beberapa hidden layer.





PEMODELAN 2

Multi Layer Perceptron : dilakukan dengan menggunakan $\mu=0.1$, layer = 5, 10, 15, 20, 25





SOURCE CODE


```
# MODEL MLP
model_mlp = MLP(hidden_layer_sizes=(5), activation = 'logistic',
                 learning_rate_init=0.1, tol=0, random_state=15)
model_mlp.fit(data, label)

# Menghitung akurasi
print('Accuracy Score :', model_mlp.score(data, label))
score = classification_report(label, model_mlp.predict(data))
print("Classification Report:")
print(score)

# Membuat plot loss curve
loss_values = model_mlp.loss_curve_
plt.figure(figsize=(10, 6))
plt.title("Loss per Iteration")
plt.xlabel("Iteration")
plt.ylabel("Loss")
plt.grid()
plt.plot(loss_values, '-', color="b", label="Loss")
plt.legend(loc="best")
plt.show()
```

✓ 0.5s

Python



5 HIDDEN LAYER

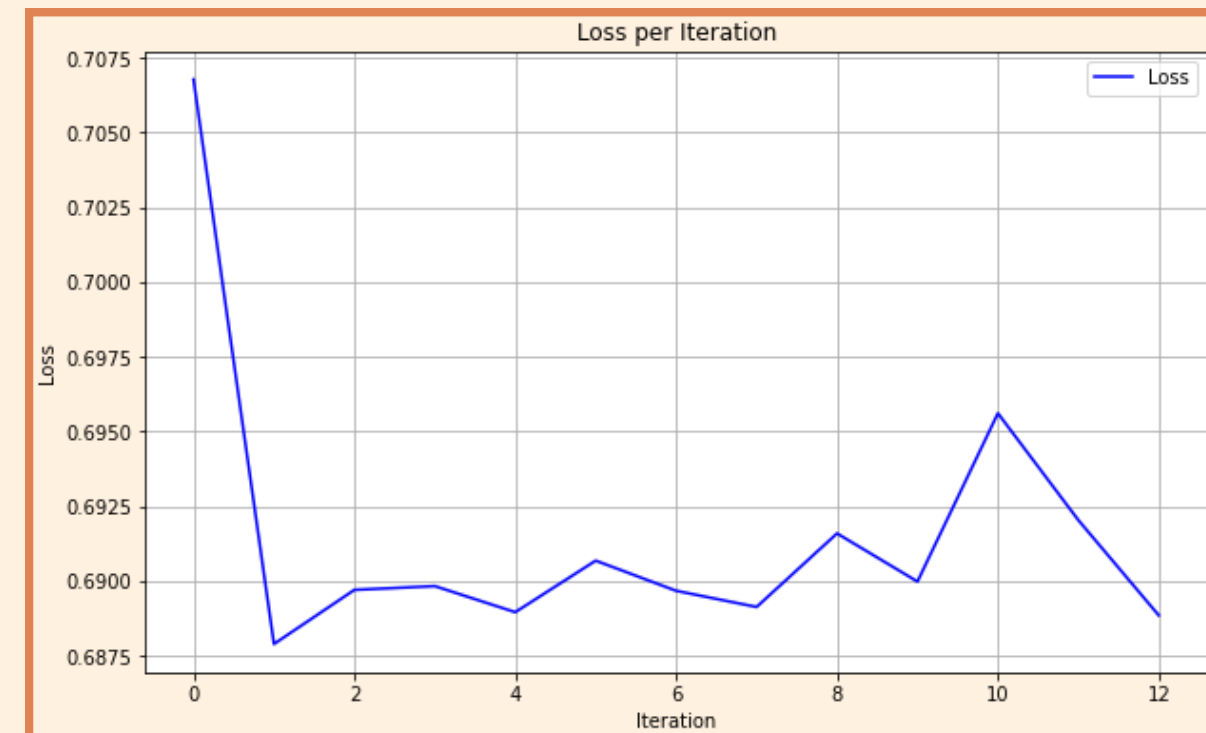
AKURASI :

Accuracy Score : 0.5445544554455446

Classification Report:

	precision	recall	f1-score	support
0	0.00	0.00	0.00	138
1	0.54	1.00	0.71	165
accuracy			0.54	303
macro avg	0.27	0.50	0.35	303
weighted avg	0.30	0.54	0.38	303

GRADIENT DESCENT :



10 HIDDEN LAYER

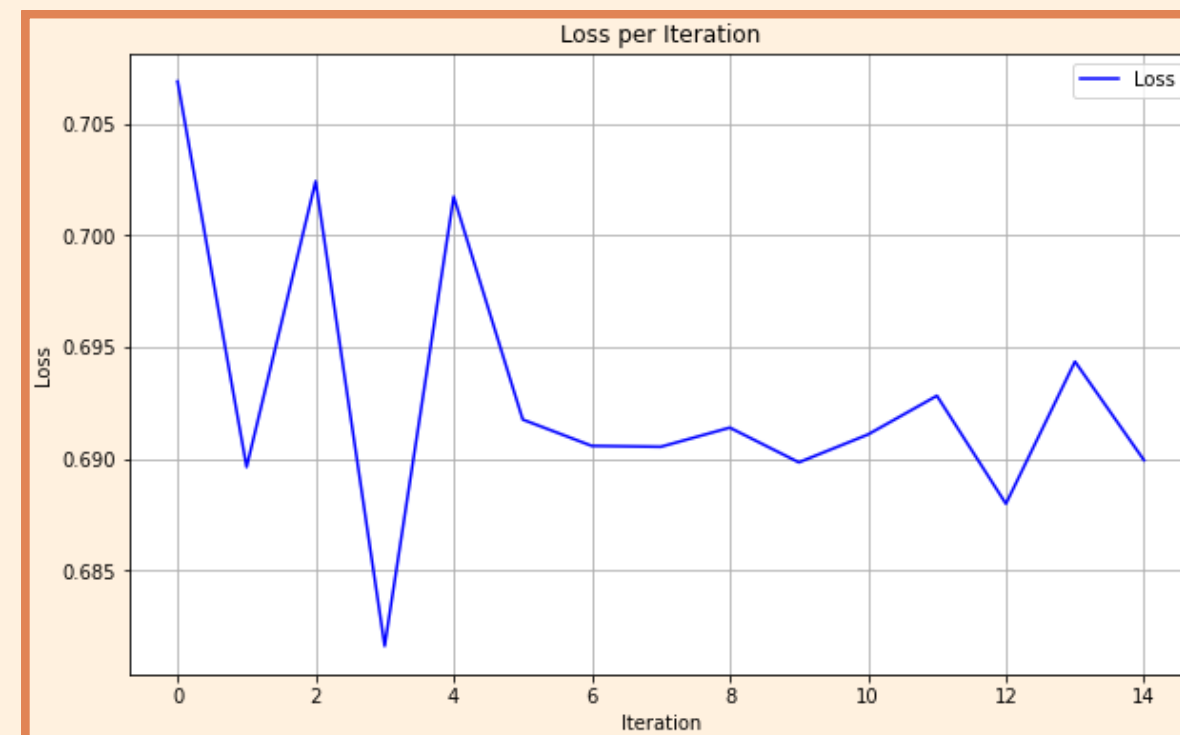
AKURASI :

Accuracy Score : 0.5445544554455446

Classification Report:

	precision	recall	f1-score	support
0	0.00	0.00	0.00	138
1	0.54	1.00	0.71	165
accuracy			0.54	303
macro avg	0.27	0.50	0.35	303
weighted avg	0.30	0.54	0.38	303

GRADIENT DESCENT :



15 HIDDEN LAYER

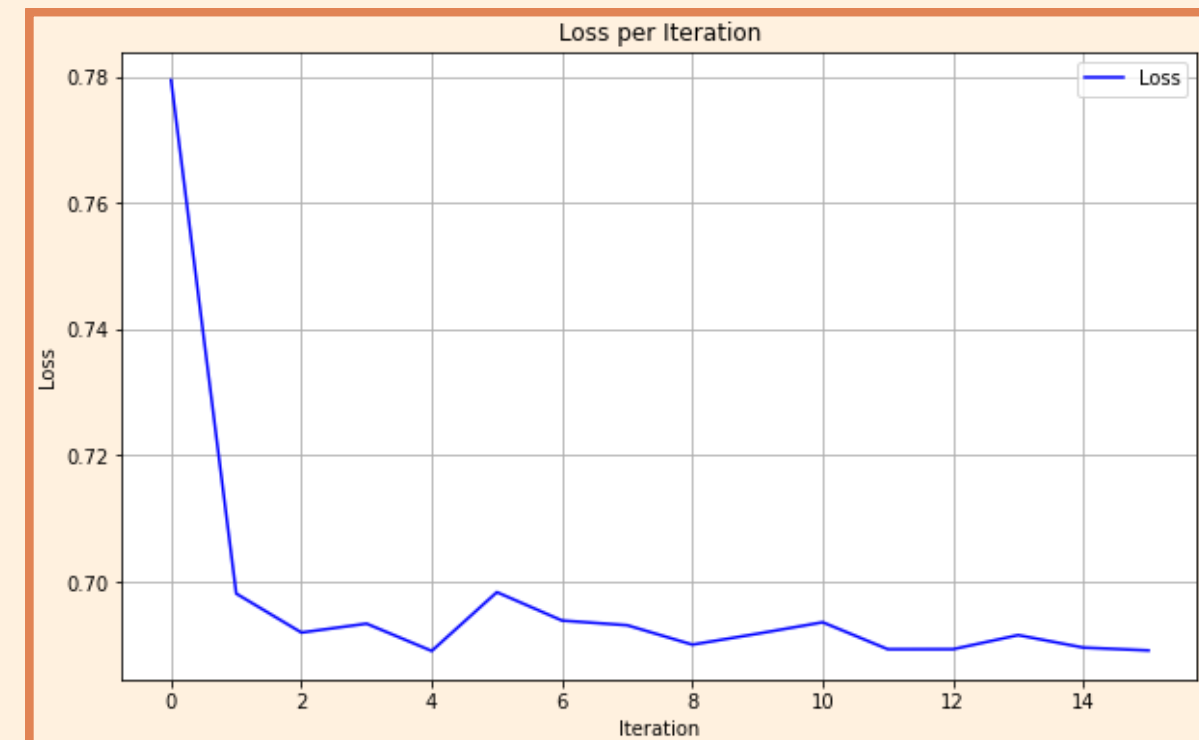
AKURASI :

Accuracy Score : 0.5445544554455446

Classification Report:

	precision	recall	f1-score	support
0	0.00	0.00	0.00	138
1	0.54	1.00	0.71	165
accuracy			0.54	303
macro avg	0.27	0.50	0.35	303
weighted avg	0.30	0.54	0.38	303

GRADIENT DESCENT :



20 HIDDEN LAYER

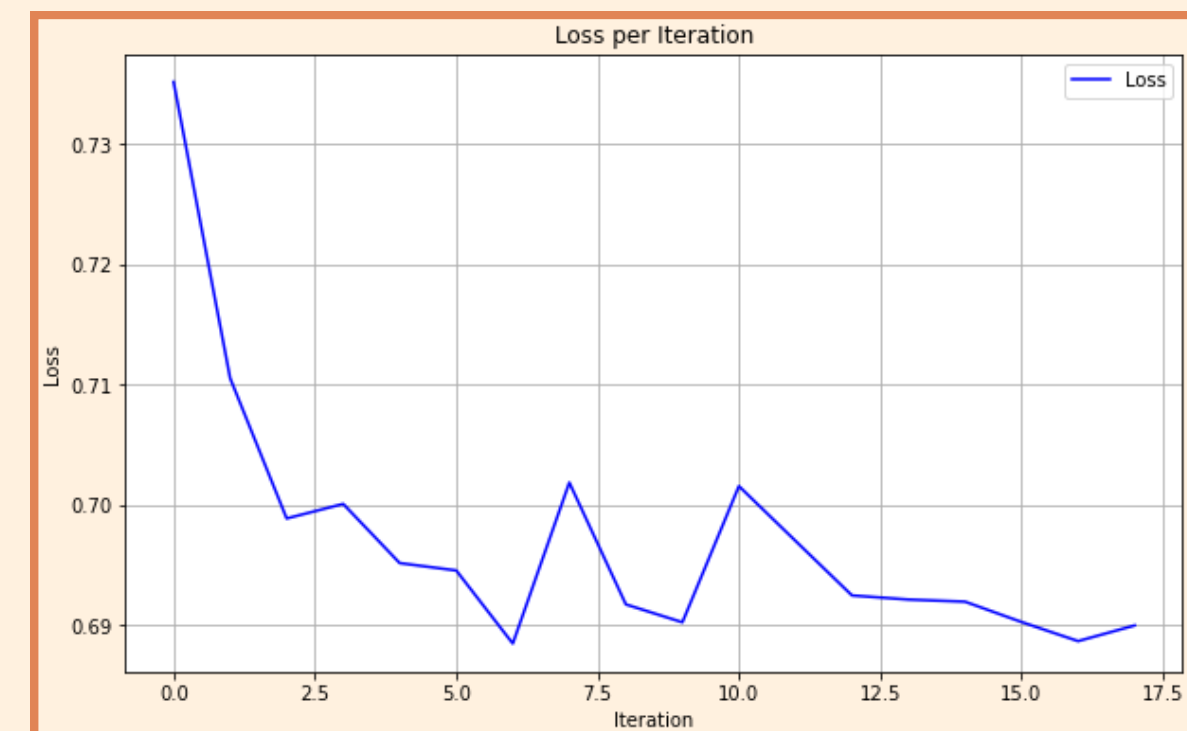
AKURASI :

Accuracy Score : 0.5445544554455446

Classification Report:

	precision	recall	f1-score	support
0	0.00	0.00	0.00	138
1	0.54	1.00	0.71	165
accuracy			0.54	303
macro avg	0.27	0.50	0.35	303
weighted avg	0.30	0.54	0.38	303

GRADIENT DESCENT :

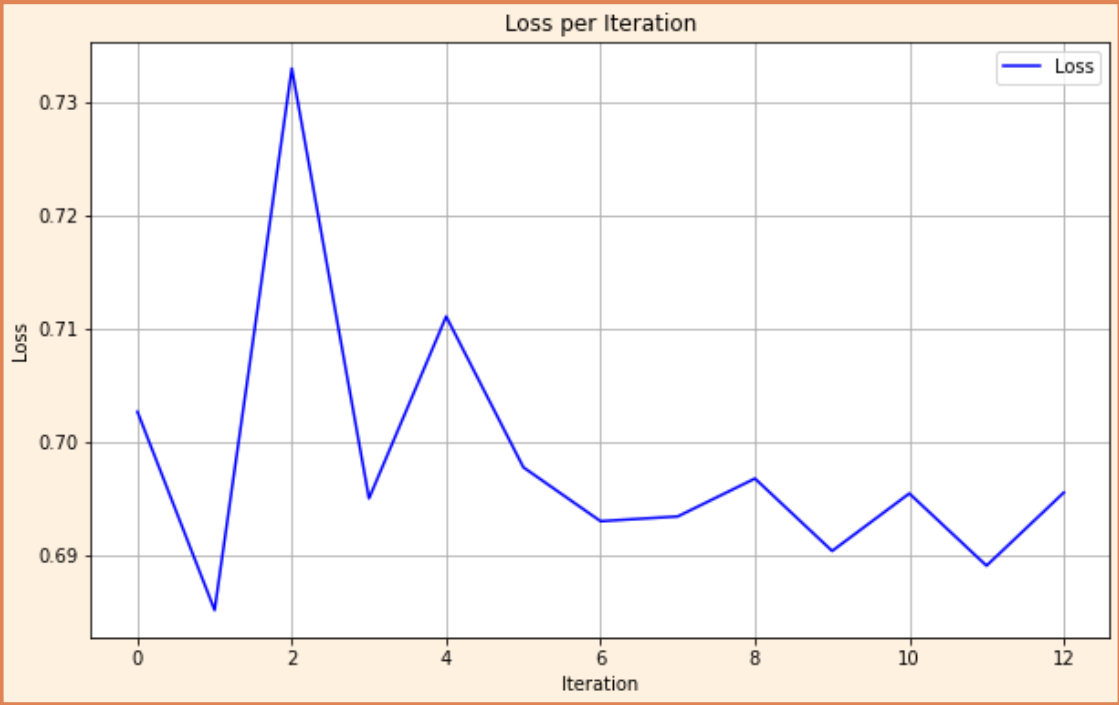


25 HIDDEN LAYER

AKURASI :

Accuracy Score : 0.5445544554455446				
Classification Report:				
	precision	recall	f1-score	support
0	0.00	0.00	0.00	138
1	0.54	1.00	0.71	165
accuracy			0.54	303
macro avg	0.27	0.50	0.35	303
weighted avg	0.30	0.54	0.38	303


GRADIENT DESCENT :

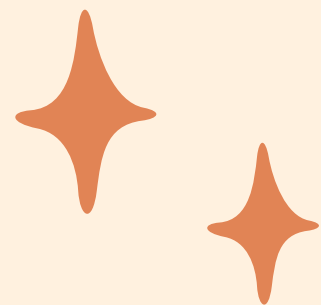
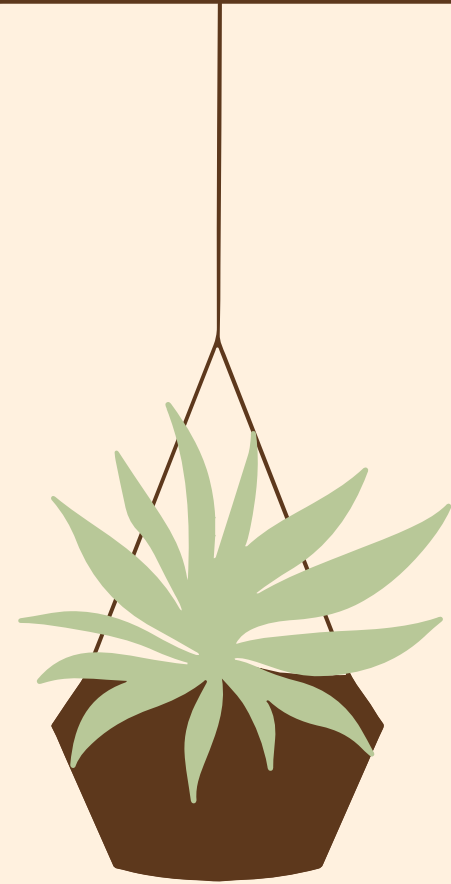




ANALISIS

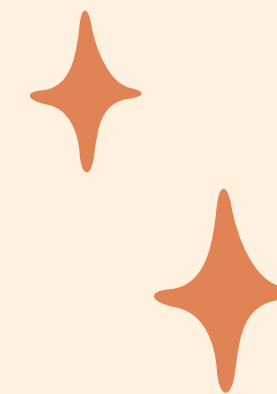
Dari kelima percobaan yang dengan menggunakan nilai Miu sebesar 0.1 dan mengganti jumlah hidden layer, grafik gradient descent yang menunjukkan hasil paling baik ditunjukkan dengan grafik yang menggunakan 15 hidden layer. Meskipun terjadi bounce berulang kali dan nilai loss tidak mencapai 0, grafik dengan 15 hidden layer lebih cepat dalam mencapai nilai loss yang rendah pada data pengujian dibandingkan dengan grafik lainnya. Pada percobaan ini, dengan hasil akurasi yang hampir sama pada keseluruhan percobaan menunjukkan bahwa jumlah hidden layer masih belum mampu dalam memecahkan masalah yang sulit atau menangkap pola yang sangat rumit dalam data. Agar didapatkan hasil pemodelan yang optimal diperlukan penambahan, perbaikan parameter atau bahkan reduksi dimensi pada data.





PEMODELAN 3

Multi Layer Perceptron : dilakukan dengan menggunakan layer 10 dengan $\mu = 0.01, 0.05, 0.1, 0.5, 1$





SOURCE CODE

```
# MODEL MLP
model_mlp = MLP(hidden_layer_sizes=(10), activation = 'logistic',
                learning_rate_init=0.01, tol=0, random_state=10)
model_mlp.fit(data, label)

# Menghitung akurasi
print('Accuracy Score :', model_mlp.score(data, label))
score = classification_report(label, model_mlp.predict(data))
print("Classification Report:")
print(score)

# Membuat plot loss curve
loss_values = model_mlp.loss_curve_
plt.figure(figsize=(10, 6))
plt.title("Loss per Iteration")
plt.xlabel("Iteration")
plt.ylabel("Loss")
plt.grid()
plt.plot(loss_values, '-', color="b", label="Loss")
plt.legend(loc="best")
plt.show()
```

✓ 0.5s

Python



MIU 0.01

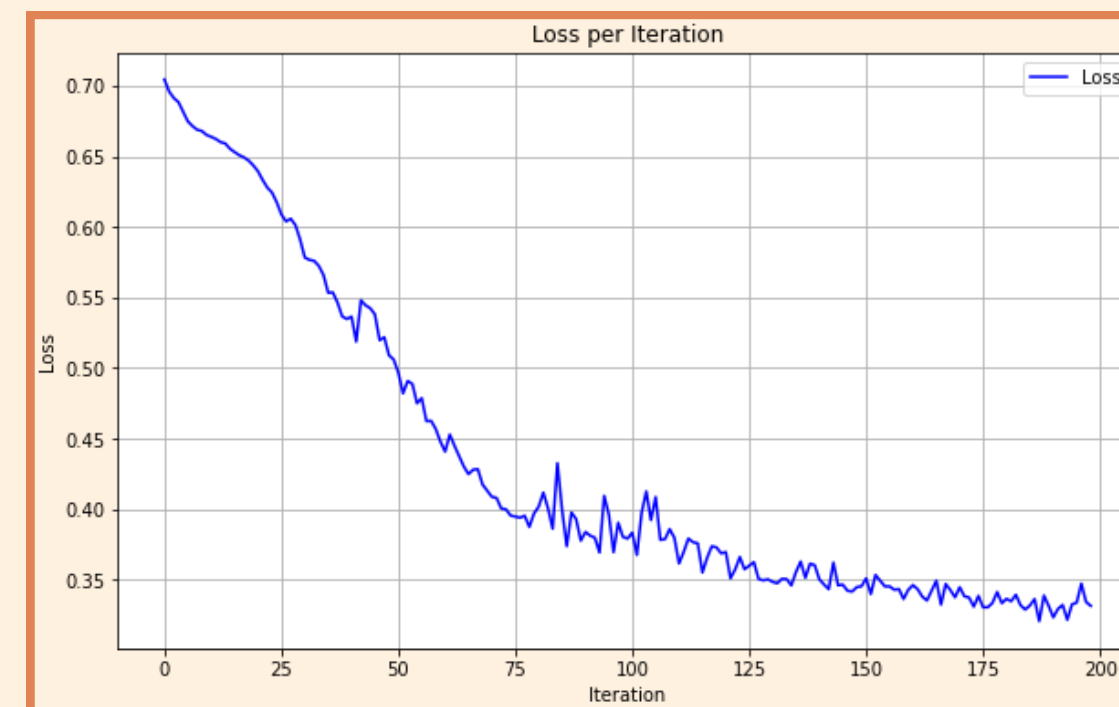
AKURASI :

Accuracy Score : 0.8514851485148515

Classification Report:

	precision	recall	f1-score	support
0	0.80	0.91	0.85	138
1	0.91	0.81	0.86	165
accuracy			0.85	303
macro avg	0.85	0.86	0.85	303
weighted avg	0.86	0.85	0.85	303

GRADIENT DESCENT :



MIU 0.05

AKURASI :

Accuracy Score : 0.5445544554455446

Classification Report:

	precision	recall	f1-score	support
0	0.00	0.00	0.00	138
1	0.54	1.00	0.71	165
accuracy			0.54	303
macro avg	0.27	0.50	0.35	303
weighted avg	0.30	0.54	0.38	303

GRADIENT DESCENT :



MIU 0.1

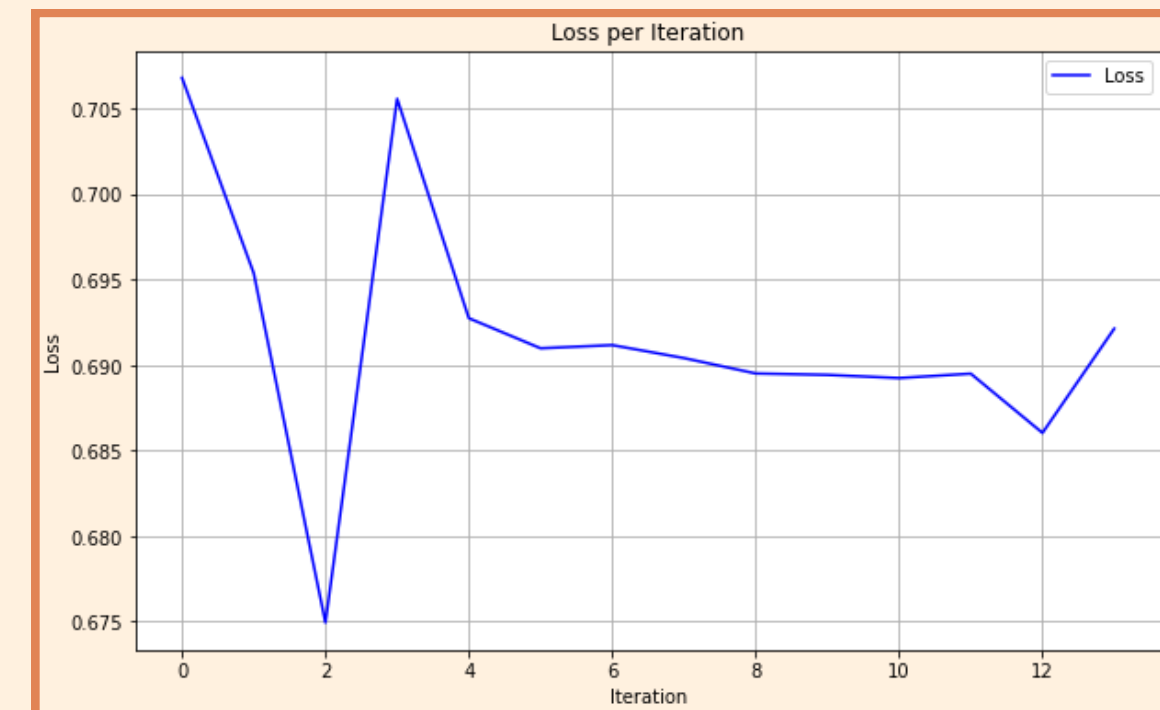
AKURASI :

Accuracy Score : 0.5445544554455446

Classification Report:

	precision	recall	f1-score	support
0	0.00	0.00	0.00	138
1	0.54	1.00	0.71	165
accuracy			0.54	303
macro avg	0.27	0.50	0.35	303
weighted avg	0.30	0.54	0.38	303

GRADIENT DESCENT :



MIU 0.5

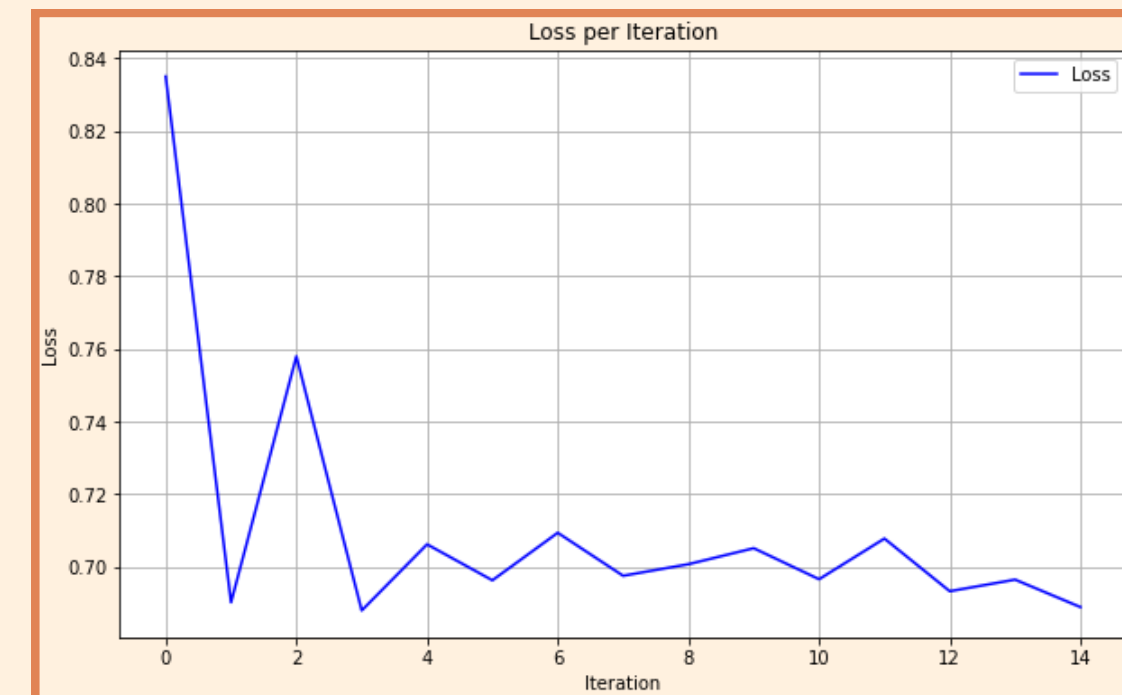
AKURASI :

Accuracy Score : 0.5445544554455446

Classification Report:

	precision	recall	f1-score	support
0	0.00	0.00	0.00	138
1	0.54	1.00	0.71	165
accuracy			0.54	303
macro avg	0.27	0.50	0.35	303
weighted avg	0.30	0.54	0.38	303

GRADIENT DESCENT :



MIU 1

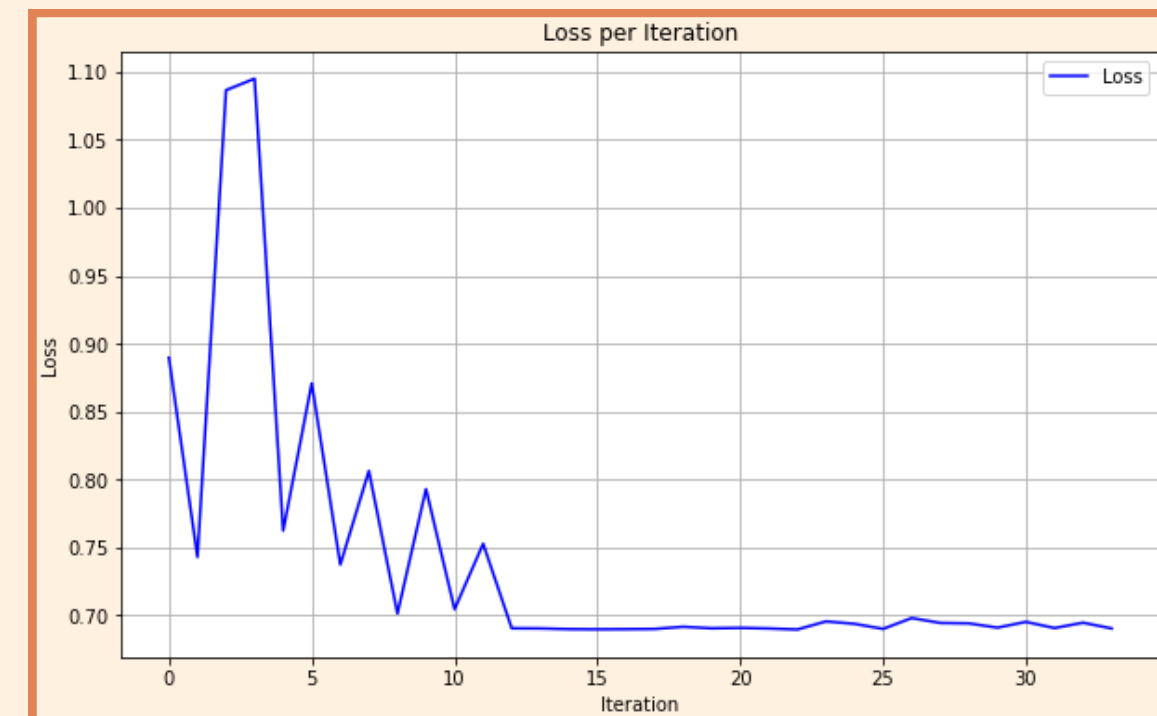
AKURASI :

Accuracy Score : 0.5445544554455446

Classification Report:

	precision	recall	f1-score	support
0	0.00	0.00	0.00	138
1	0.54	1.00	0.71	165
accuracy			0.54	303
macro avg	0.27	0.50	0.35	303
weighted avg	0.30	0.54	0.38	303


GRADIENT DESCENT :





ANALISIS

Dari kelima percobaan yang dengan menggunakan hidden layer sejumlah 10 dan mengganti nilai Miu, grafik gradient descent yang menunjukkan hasil paling baik ditunjukkan dengan grafik yang menggunakan nilai Miu sebesar 0.01. Meskipun terjadi bounce berulang kali dalam proses pelatihannya dan nilai loss tidak mencapai 0, grafik dengan Miu 0.01 lebih cepat dan konsisten dalam mencapai nilai loss yang rendah pada data pengujian dibandingkan dengan grafik lainnya. Pada percobaan ini, dengan hasil akurasi yang hampir sama pada beberapa percobaan menunjukkan bahwa nilai Miu saja masih belum mampu dalam memecahkan masalah yang sulit atau menangkap pola yang sangat rumit dalam data. Agar didapatkan hasil pemodelan yang optimal diperlukan penambahan, perbaikan parameter atau bahkan reduksi dimensi pada data.






KESIMPULAN

Ketiga percobaan menunjukkan perbandingan antara model Single Layer Perceptron (SLP) dan model Multilayer Perceptron (MLP) dalam pemodelan data penyakit jantung (Heart Disease).

1. **Percobaan 1 (SLP):** Model SLP, meskipun sederhana, berhasil menghasilkan akurasi sebesar 83%. Namun, keterbatasan SLP dalam menangani masalah klasifikasi yang kompleks seperti penyakit jantung. Dalam konteks tugas klinis yang sensitif, akurasi yang lebih tinggi mungkin diperlukan.
2. **Percobaan 2 dan 3 (MLP):** Penggunaan model Multilayer Perceptron (MLP) dengan variasi jumlah hidden layer dan nilai learning rate (μ) menunjukkan bahwa meskipun terdapat peningkatan dalam grafik gradient descent, model MLP juga mengalami kendala dalam mencapai nilai loss yang rendah pada data pengujian. Sejumlah hidden layer dan nilai μ saja tidak cukup untuk mengatasi masalah yang sulit dalam data.

Kesimpulan yang dapat diambil adalah bahwa, meskipun model MLP memiliki kapabilitas yang lebih tinggi dalam menangani masalah yang kompleks dibandingkan dengan SLP, hasil dari ketiga percobaan menunjukkan bahwa pemodelan penyakit jantung tetap menjadi tantangan yang signifikan. Oleh karena itu, diperlukan eksperimen lebih lanjut, perbaikan parameter, atau pendekatan yang lebih canggih untuk mencapai performa yang memadai dalam mendeteksi penyakit jantung dengan akurasi dan keandalan yang tinggi.



TERIMAKASIH

