Nama : Andreas Hasiholan Sinaga

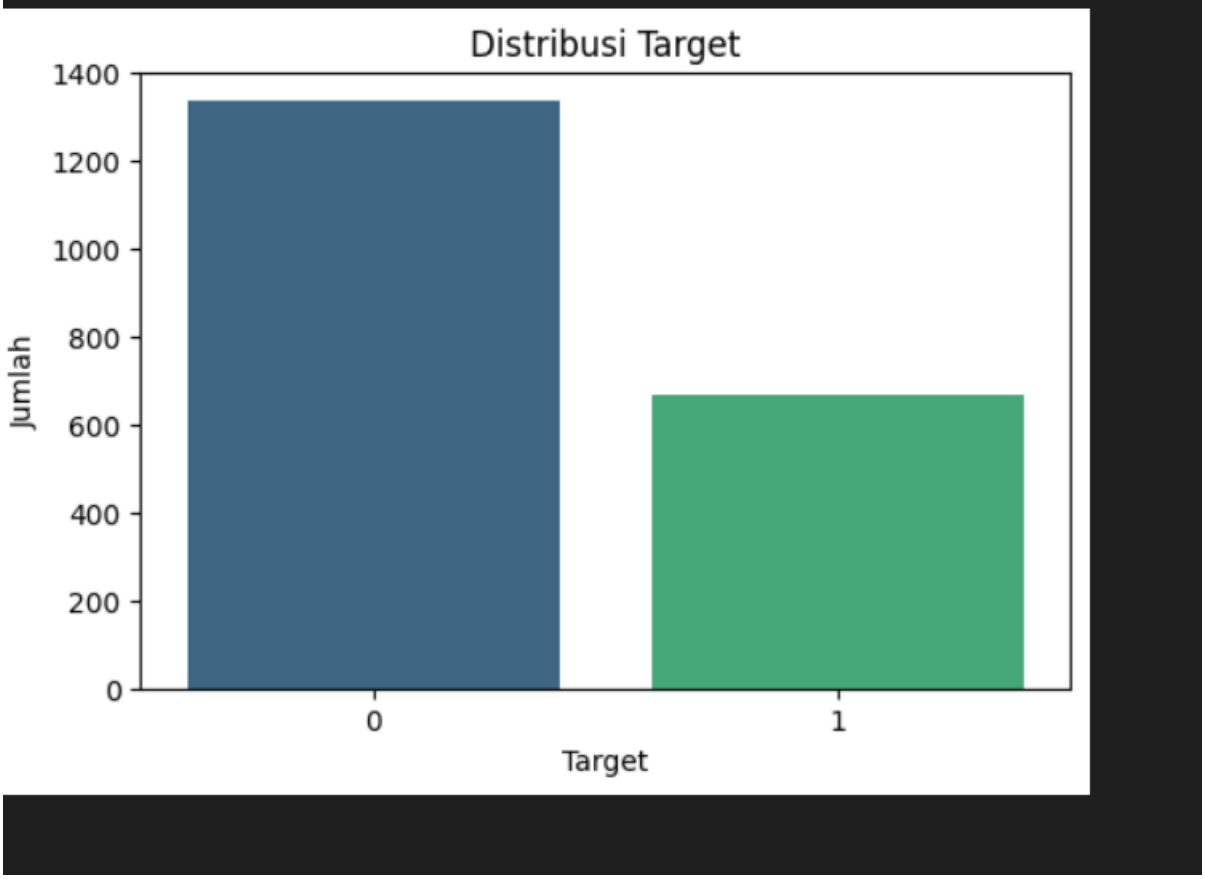
NIM: 110321303

Model Classification dengan Deep Learning Dataset Dummy Data

```python
# ====================================
# 1. Membuat Dataset Dummy dengan Target
# ====================================
np.random.seed(42)
num_samples = 2000  # Jumlah sampel
data_features = 6  # Jumlah fitur

# Membuat fitur dengan distribusi normal
X = np.random.randn(num_samples, data_features)

# Membuat target biner dengan logika sederhana (dummy)
# Jika jumlah nilai fitur positif > jumlah negatif, target = 1, sebaliknya = 0
y = (np.sum(X > 0, axis=1) > (data_features // 2)).astype(int)

# Konversi menjadi DataFrame untuk analisis lebih lanjut
data = pd.DataFrame(X, columns=[f'feature_{i+1}' for i in range(data_features)])
data['target'] = y

# Menyimpan dataset ke file CSV
data.to_csv("Task11DummyData/dummy_dataset.csv", index=False)
print("Dataset dummy disimpan sebagai 'dummy_dataset.csv'")
```

Pada modeling ini menggunakan dataset dummy yang dimana data sampel sebanyak 2000 dan menggunakan 6 feauture.Tipe data feature maupun target adalah float

```python
# Mengecek distribusi target
plt.figure(figsize=(6, 4))
sns.countplot(x='target', data=data, palette='viridis')
plt.title('Distribusi Target')
plt.xlabel('Target')
plt.ylabel('Jumlah')
plt.savefig("target_distribution.png")
plt.show()
```

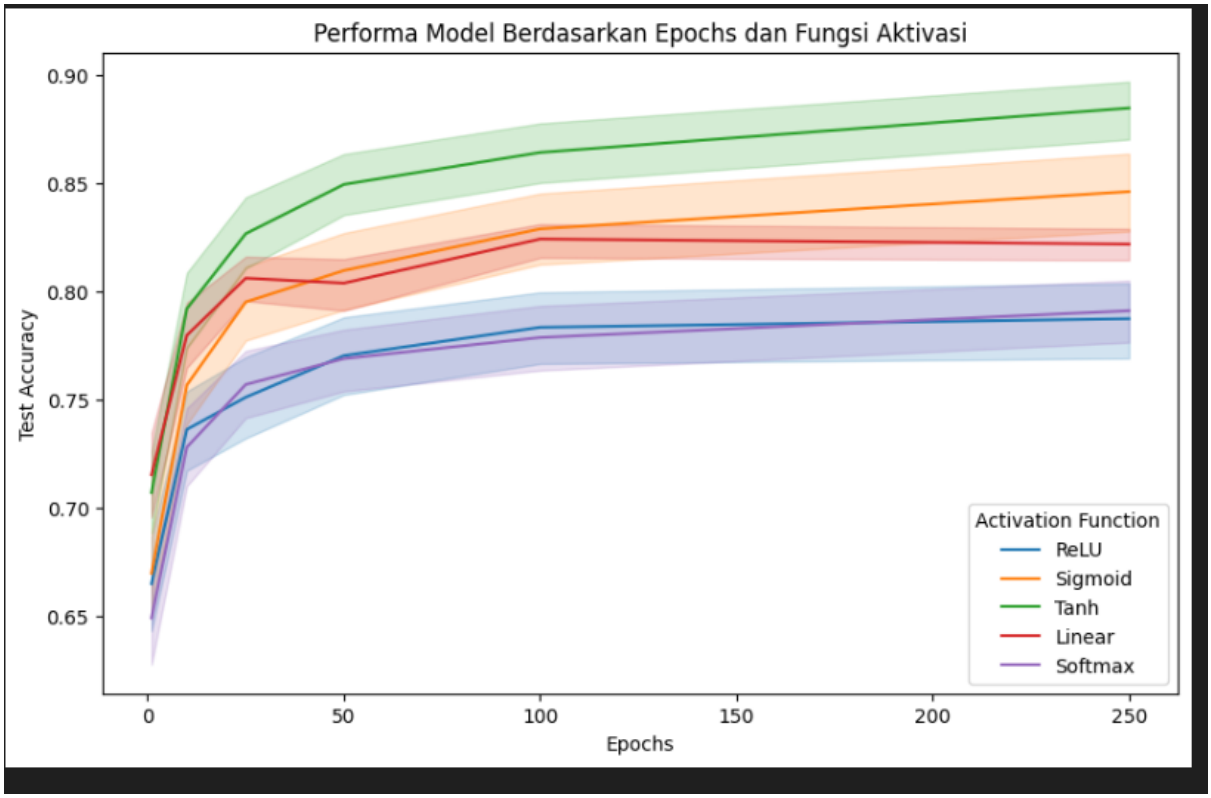



Dari grafik ini kita dapat melihat jumlaah target 0 yaitu sebanyak 1300 sedangkan target 1 sebanya 700

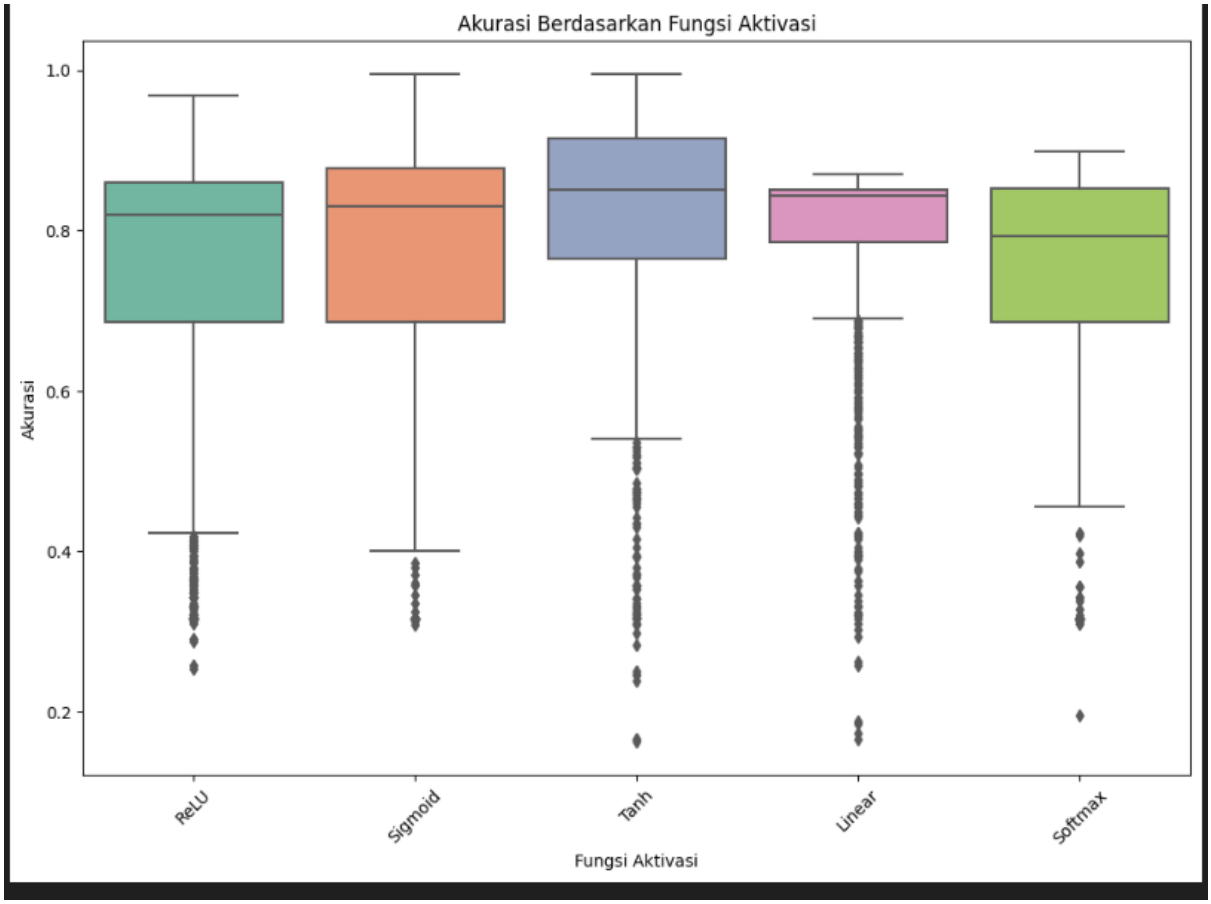
```python
# 5. Eksperimen Hyperparameter
hyperparameters = {
    "hidden_layers": [[4], [8], [16], [32], [64], [16, 8], [32, 16, 8]],
    "activation_fn": ["ReLU", "Sigmoid", "Tanh", "Linear", "Softmax"],
    "learning_rate": [10, 1, 0.1, 0.01, 0.001, 0.0001],
    "batch_size": [16, 32, 64, 128, 256, 512],
    "epochs": [1, 10, 25, 50, 100, 250]
}

```

Kode ini  mendefinisikan konfigurasi hyperparameter untuk eksperimen pelatihan model klasifikasi menggunakan berbagai kombinasi parameter. Hyperparameter yang dicoba meliputi jumlah neuron pada lapisan tersembunyi (hidden_layers), dengan variasi mulai dari satu

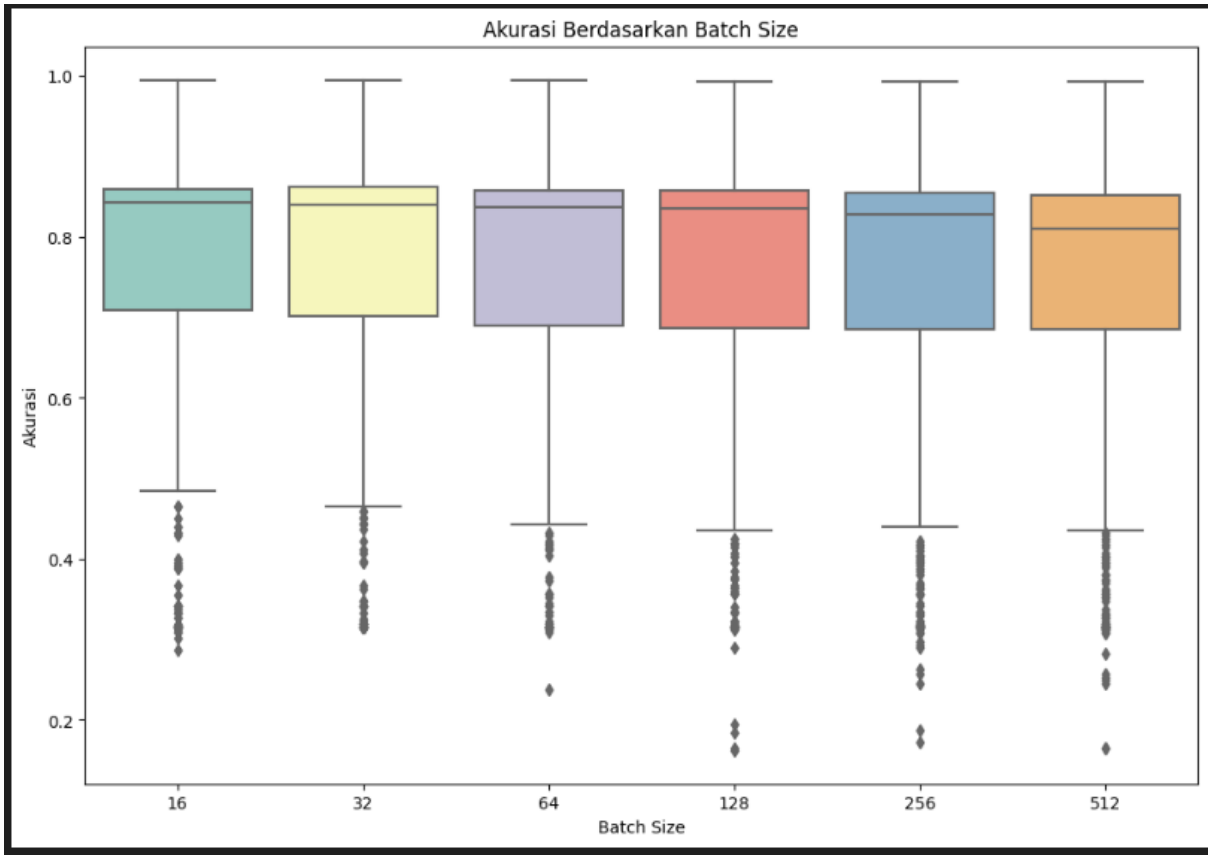
lapisan sederhana seperti [4] hingga struktur multi-lapisan seperti [32, 16, 8]. Fungsi aktivasi (activation_fn) mencakup opsi ReLU, Sigmoid, Tanh, Linear, dan Softmax untuk menangani pola non-linearitas dalam data. Selain itu, nilai learning_rate diatur dari 10 hingga 0.0001 untuk mengeksplorasi kecepatan pembaruan bobot, batch_size bervariasi antara 16 hingga 512 untuk mengelola jumlah data yang diproses per iterasi, dan jumlah epochs mulai dari 1 hingga 250 untuk menentukan jumlah iterasi pelatihan. Tujuan dari konfigurasi ini adalah mengevaluasi performa model dengan berbagai kombinasi hyperparameter untuk menemukan pengaturan terbaik yang menghasilkan akurasi tertinggi.



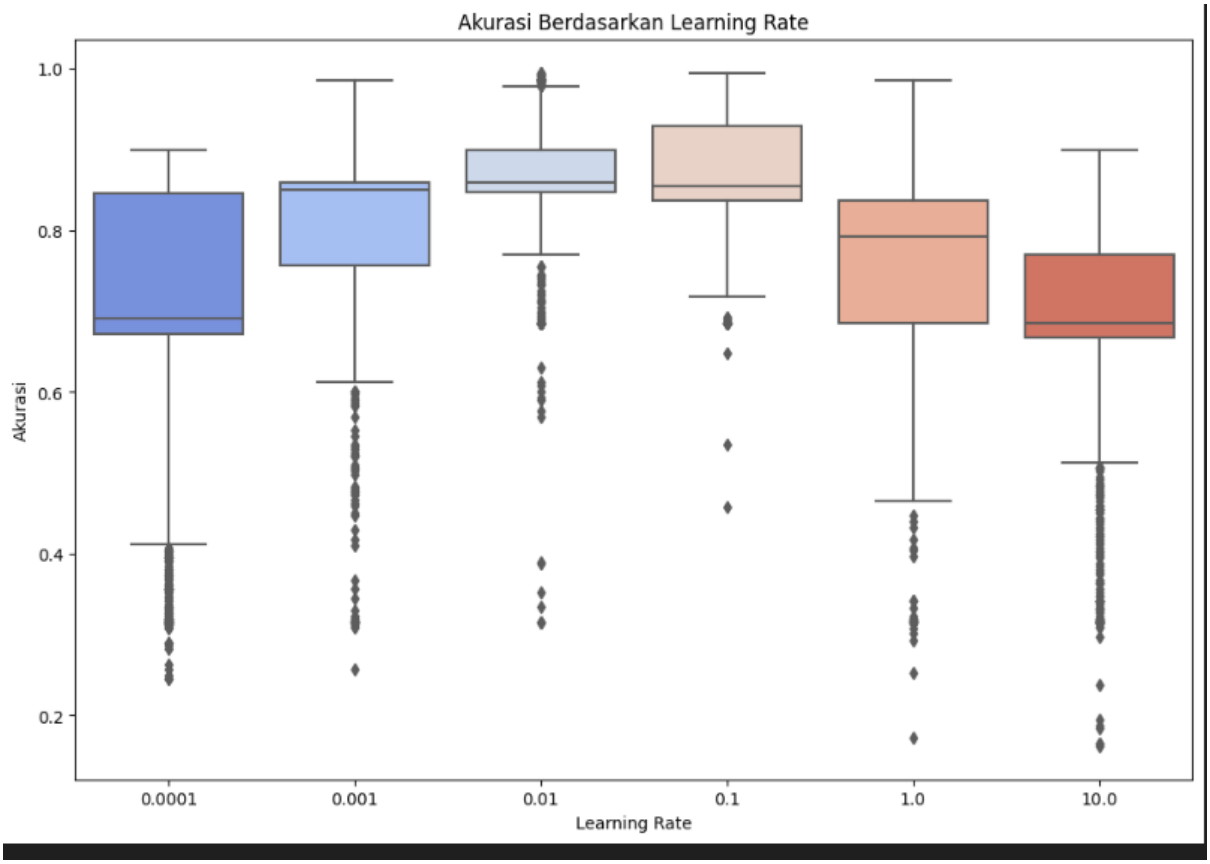

Dari hasil beberapa kombinasi parameter, diperoleh grafik yang menunjukkan performa model berdasarkan jumlah epochs dan jenis fungsi aktivasi yang digunakan. Grafik ini menggambarkan bahwa fungsi aktivasi Tanh menghasilkan akurasi tertinggi dibandingkan fungsi aktivasi lainnya, mencapai nilai stabil mendekati 0.9 setelah 100 epochs. Fungsi aktivasi Sigmoid dan Softmax juga menunjukkan performa yang cukup baik, tetapi berada sedikit di bawah Tanh. ReLU memiliki tren peningkatan akurasi yang konsisten namun lebih lambat dibandingkan fungsi lainnya, sedangkan Linear memiliki performa terendah di antara semua fungsi aktivasi. Area berwarna di sekitar setiap kurva menunjukkan tingkat variasi (confidence interval) untuk setiap fungsi aktivasi. Dari grafik ini, dapat disimpulkan bahwa pemilihan fungsi aktivasi memiliki pengaruh signifikan terhadap performa model, dengan Tanh menjadi pilihan terbaik dalam konfigurasi eksperimen ini.




Dari hasil beberapa kombinasi parameter, diperoleh grafik boxplot yang menunjukkan distribusi akurasi berdasarkan jenis fungsi aktivasi yang digunakan. Grafik ini mengungkapkan bahwa fungsi aktivasi Tanh memiliki median akurasi tertinggi dengan distribusi yang cukup stabil, meskipun terdapat beberapa outlier di rentang akurasi rendah. Sigmoid dan ReLU juga menunjukkan performa yang kompetitif dengan median akurasi mendekati Tanh, tetapi dengan sedikit lebih banyak outlier pada nilai akurasi rendah. Fungsi aktivasi Softmax memiliki distribusi yang serupa dengan Tanh, namun median akurasinya sedikit lebih rendah. Sebaliknya, fungsi aktivasi Linear menunjukkan performa yang paling rendah dengan rentang distribusi yang sempit dan median yang lebih rendah dibandingkan fungsi lainnya. Dari grafik ini, dapat dilihat bahwa fungsi aktivasi Tanh menjadi pilihan terbaik untuk mencapai performa akurasi yang tinggi dan stabil dalam eksperimen ini.




Dari hasil beberapa kombinasi parameter, diperoleh grafik boxplot yang menunjukkan distribusi akurasi berdasarkan ukuran batch (batch size) yang digunakan. Grafik ini menunjukkan bahwa ukuran batch memiliki pengaruh terhadap distribusi akurasi, meskipun median akurasi relatif stabil di sekitar 0.8 untuk semua ukuran batch. Ukuran batch kecil seperti 16 dan 32 menunjukkan distribusi yang lebih luas dengan lebih banyak outlier pada akurasi rendah. Sebaliknya, ukuran batch yang lebih besar, seperti 256 dan 512, cenderung memiliki distribusi yang lebih sempit tetapi tetap menyisakan beberapa outlier pada rentang akurasi rendah. Grafik ini mengindikasikan bahwa ukuran batch tidak memiliki dampak signifikan pada median akurasi, tetapi ukuran batch kecil dapat menghasilkan variabilitas performa yang lebih besar dibandingkan ukuran batch yang lebih besar.




Dari hasil beberapa kombinasi parameter, diperoleh grafik boxplot yang menunjukkan distribusi akurasi berdasarkan nilai learning rate yang digunakan. Grafik ini menunjukkan bahwa learning rate yang lebih kecil, seperti 0.0001 dan 0.001, cenderung menghasilkan median akurasi yang lebih tinggi dan stabil, meskipun terdapat beberapa outlier pada akurasi rendah. Nilai learning rate menengah seperti 0.01 dan 0.1 juga menunjukkan performa yang baik, tetapi distribusinya mulai meluas, menandakan peningkatan variabilitas. Sebaliknya, learning rate yang besar, seperti 1.0 dan 10.0, menghasilkan performa yang kurang stabil dengan median akurasi yang lebih rendah dan lebih banyak outlier, menunjukkan bahwa learning rate terlalu besar dapat menghambat konvergensi model. Dari grafik ini, dapat disimpulkan bahwa learning rate kecil hingga menengah (0.0001 hingga 0.01) memberikan hasil yang lebih baik dan konsisten untuk akurasi model.