Nama : Hero Kartiko

NIM : 1103210205

Kelas : TK-45-G04

TUGAS WEEK 10 MLP CLASSIFICATION

```
# 1. Importing Libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.metrics import mean_squared_error, mean_absolute_error, accuracy_score, r2_score, classification_report
import torch
import torch.nn as nn
import torch.optim as optim
from torch.utils.data import DataLoader, TensorDataset
import itertools
%matplotlib inline
```

✓ 0.0s

```
df = pd.read_csv('default of credit card clients.csv')
df.head(10)
```

✓ 0.0s

```
df.info()
```

✓ 0.0s

Output:

	ID	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_0	PAY_2	PAY_3	PAY_4	...	BILL_AMT4	BILL_AMT5	BILL_AMT6	PAY_AMT1	PAY_AMT2	PAY_AMT3	PAY_AMT4	PAY_AMT5	PAY_AMT6	default payment next month
0	1	20000	2	2	1	24	2	2	-1	-1	...	0	0	0	0	689	0	0	0	0	1
1	2	120000	2	2	2	26	-1	2	0	0	...	3272	3455	3261	0	1000	1000	1000	0	2000	1
2	3	90000	2	2	2	34	0	0	0	0	...	14331	14948	15549	1518	1500	1000	1000	1000	5000	0
3	4	50000	2	2	1	37	0	0	0	0	...	28314	28959	29547	2000	2019	1200	1100	1069	1000	0
4	5	50000	1	2	1	57	-1	0	-1	0	...	20940	19146	19131	2000	36681	10000	9000	689	679	0
5	6	50000	1	1	2	37	0	0	0	0	...	19394	19619	20024	2500	1815	657	1000	1000	800	0
6	7	500000	1	1	2	29	0	0	0	0	...	542653	483003	473944	55000	40000	38000	20239	13750	13770	0
7	8	100000	2	2	2	23	0	-1	-1	0	...	221	-159	567	380	601	0	581	1687	1542	0
8	9	140000	2	3	1	28	0	0	2	0	...	12211	11793	3719	3329	0	432	1000	1000	1000	0
9	10	20000	1	3	2	35	-2	-2	-2	-2	...	0	13007	13912	0	0	0	13007	1122	0	0

```
Data columns (total 25 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   ID                          30000 non-null  int64
 1   LIMIT_BAL                   30000 non-null  int64
 2   SEX                         30000 non-null  int64
 3   EDUCATION                   30000 non-null  int64
 4   MARRIAGE                    30000 non-null  int64
 5   AGE                         30000 non-null  int64
 6   PAY_0                       30000 non-null  int64
 7   PAY_2                       30000 non-null  int64
 8   PAY_3                       30000 non-null  int64
 9   PAY_4                       30000 non-null  int64
 10  PAY_5                       30000 non-null  int64
 11  PAY_6                       30000 non-null  int64
 12  BILL_AMT1                   30000 non-null  int64
 13  BILL_AMT2                   30000 non-null  int64
 14  BILL_AMT3                   30000 non-null  int64
 15  BILL_AMT4                   30000 non-null  int64
 16  BILL_AMT5                   30000 non-null  int64
 17  BILL_AMT6                   30000 non-null  int64
 18  PAY_AMT1                    30000 non-null  int64
 19  PAY_AMT2                    30000 non-null  int64
 20  PAY_AMT3                    30000 non-null  int64
 21  PAY_AMT4                    30000 non-null  int64
 22  PAY_AMT5                    30000 non-null  int64
 23  PAY_AMT6                    30000 non-null  int64
 24  default payment next month  30000 non-null  int64
dtypes: int64(25)
```

Analisis:

Kode diatas terdiri dari library yang akan digunakan untuk melakukan machine learning, menampilkan visualisasi dari dataset yang akan dipakai, serta model yang akan digunakan untuk melakukan pemodelan dari dataset tersebut. Dataset yang digunakan adalah default credit of client.csv dengan menampilkan 10 baris pertama dan 25 kolom.

```
df.describe()
```
✓ 0.0s

```
df.isnull().sum()
```
✓ 0.0s

Output:

	ID	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_0	PAY_2	PAY_3	PAY_4	...	BILL_AMT4	BILL_AMT5	BILL_AMT6	PAY_AMT1	PAY_AMT2	PAY
count	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000	...	30000.000000	30000.000000	30000.000000	30000.000000	3.000000e+04	30000
mean	15000.500000	167484.322667	1.603733	1.853133	1.551867	35.485500	-0.016700	-0.133767	-0.166200	-0.220667	...	43262.948967	40311.400967	38871.760400	5663.580500	5.921163e+03	5225
std	8660.398374	129747.661567	0.489129	0.790349	0.521970	9.217904	1.123802	1.197186	1.196868	1.169139	...	64332.856134	60797.155770	59554.107537	16563.280354	2.304087e+04	17606
min	1.000000	10000.000000	1.000000	0.000000	0.000000	21.000000	-2.000000	-2.000000	-2.000000	-2.000000	...	-170000.000000	-81334.000000	-339603.000000	0.000000	0.000000e+00	0
25%	7500.750000	50000.000000	1.000000	1.000000	1.000000	28.000000	-1.000000	-1.000000	-1.000000	-1.000000	...	2326.750000	1763.000000	1256.000000	1000.000000	8.330000e+02	390
50%	15000.500000	140000.000000	2.000000	2.000000	2.000000	34.000000	0.000000	0.000000	0.000000	0.000000	...	19052.000000	18104.500000	17071.000000	2100.000000	2.009000e+03	1800
75%	22500.250000	240000.000000	2.000000	2.000000	2.000000	41.000000	0.000000	0.000000	0.000000	0.000000	...	54506.000000	50190.500000	49198.250000	5006.000000	5.000000e+03	4505
max	30000.000000	1000000.000000	2.000000	6.000000	3.000000	79.000000	8.000000	8.000000	8.000000	8.000000	...	891586.000000	927171.000000	961664.000000	873552.000000	1.684259e+06	896040

```
ID                          0
LIMIT_BAL                   0
SEX                         0
EDUCATION                   0
MARRIAGE                    0
AGE                         0
PAY_0                       0
PAY_2                       0
PAY_3                       0
PAY_4                       0
PAY_5                       0
PAY_6                       0
BILL_AMT1                   0
BILL_AMT2                   0
BILL_AMT3                   0
BILL_AMT4                   0
BILL_AMT5                   0
BILL_AMT6                   0
PAY_AMT1                    0
PAY_AMT2                    0
PAY_AMT3                    0
PAY_AMT4                    0
PAY_AMT5                    0
PAY_AMT6                    0
default payment next month  0
dtype: int64
```

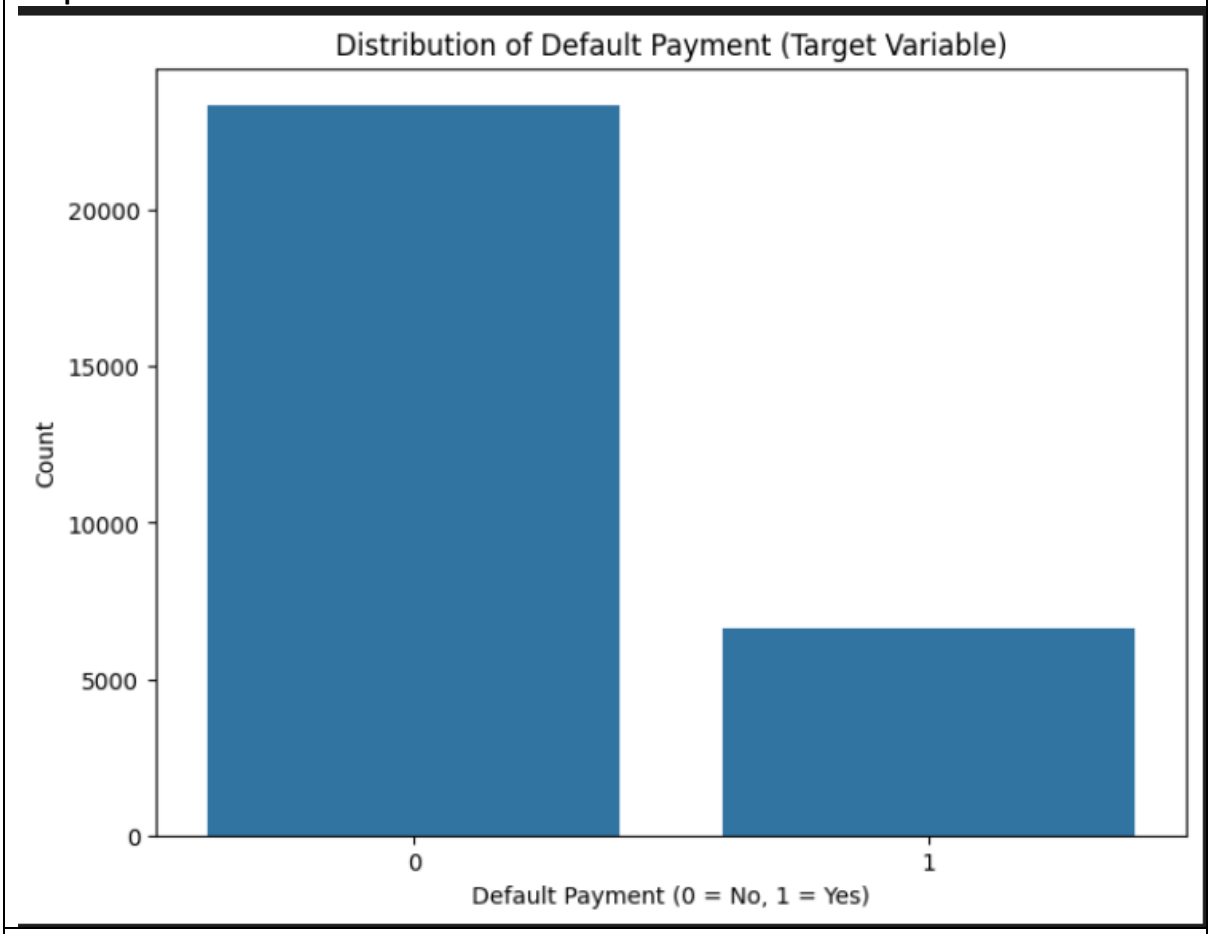
Analisis:

ditampilkan bahwa data tersebut terdiri dari ID, LIMIT_BAL, SEX, EDUCATION, MARRIAGE, AGE, PAY_0 hingga PAY_6, BILL_AMT1 hingga BILL_AMT6, dan PAY_AMT1 hingga PAY_AMT6, serta default payment next month dimana data ini mengindikasikan informasi terkait limit saldo, pembayaran, dan tagihan, sedangkan gambar kedua menampilkan fitur yang memiliki missing value dan tidak ada missing value dari dataset tersebut. Informasi ini akan digunakan untuk analisis resiko default pembayaran

```
# Plot the distribution of the target variable
plt.figure(figsize=(8, 6))
sns.countplot(x='default payment next month', data=df)
plt.title('Distribution of Default Payment (Target Variable)')
plt.xlabel('Default Payment (0 = No, 1 = Yes)')
plt.ylabel('Count')
plt.show()
```

✓ 0.2s

Output:



Analisis:

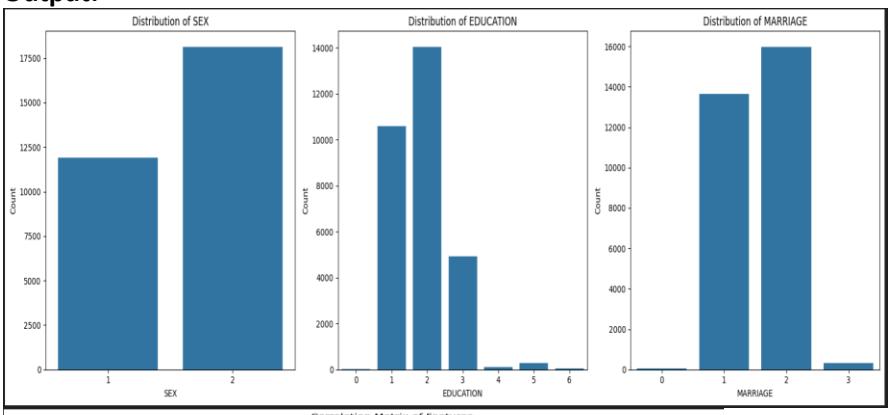
Kode ini memvisualisasikan distribusi variabel target default payment next month menggunakan bar plot untuk memahami proporsi kategori 0 and 1. Tujuannya untuk mendeteksi potensi ketidakseimbangan kelas dalam dataset. Pada visualisasi dataset diatas terlihat adanya ketidakseimbangan dataset dimana kelas 0 memiliki sekitar 20000 sampel kelas sedangkan kelas 1 memiliki 6000 sampel yang menunjukkan rasio 4:1.

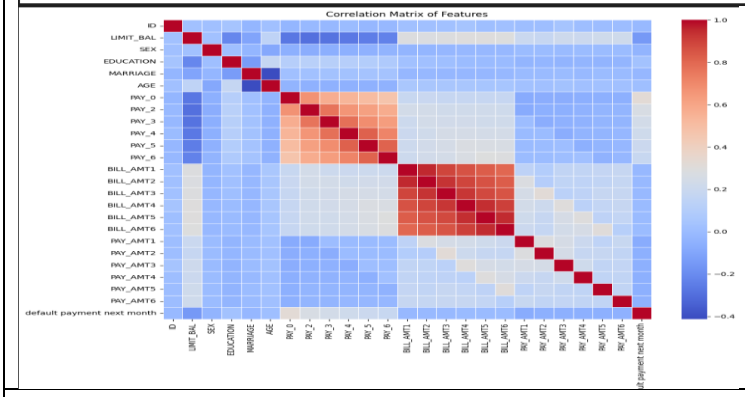
```python
# Bar plot for categorical variables (SEX, EDUCATION, MARRIAGE)
plt.figure(figsize=(18, 6))
categorical_features = ['SEX', 'EDUCATION', 'MARRIAGE']
for i, feature in enumerate(categorical_features, 1):
    plt.subplot(1, 3, i)
    sns.countplot(x=feature, data=df)
    plt.title(f'Distribution of {feature}')
    plt.xlabel(feature)
    plt.ylabel('Count')
plt.tight_layout()
plt.show()
```

```python
# Correlation matrix
plt.figure(figsize=(12, 10))
corr_matrix = df.corr()
sns.heatmap(corr_matrix, annot=False, cmap='coolwarm', linewidths=0.5)
plt.title('Correlation Matrix of Features')
plt.show()
```

✓ 0.6s

Output:





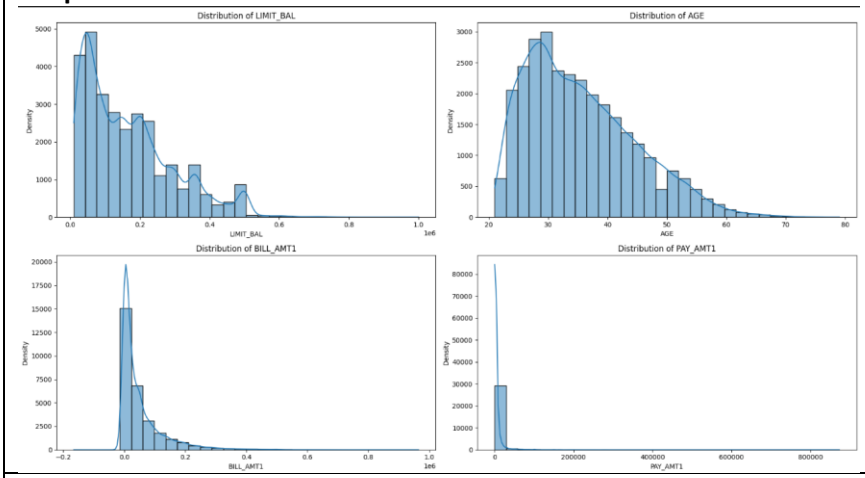
Analisis:

Visualisasi bar plot menunjukkan distribusi dari variabel kategorikal SEX, EDUCATION, dan MARRIAGE. Pada variabel SEX, kategori 2 mendominasi, sementara kategori 1 lebih sedikit. Untuk variabel EDUCATION, mayoritas data berada pada kategori 1 (graduate school), 2 (university), dan 3 (high school), sedangkan kategori lain memiliki jumlah yang sangat kecil. Pada variabel MARRIAGE, kategori 1 (married) dan 2 (single) mendominasi, dengan kategori 3 (others) yang jauh lebih sedikit. Untuk heatmap menunjukkan hubungan fitur dimana nilai BILL_AMT dan PAY_AMT memiliki korelasi positif yang kuat, sementara target default pay next month menujukkan korelasi yang lemah dengan sebagian besar fitur.

```python
# Distribution of numerical features
numerical_features = ['LIMIT_BAL', 'AGE', 'BILL_AMT1', 'PAY_AMT1']
plt.figure(figsize=(18, 10))
for i, feature in enumerate(numerical_features, 1):
    plt.subplot(2, 2, i)
    sns.histplot(df[feature], kde=True, bins=30)
    plt.title(f'Distribution of {feature}')
    plt.xlabel(feature)
    plt.ylabel('Density')
plt.tight_layout()
plt.show()
```

✓ 2.2s

Output :



Analisis:

Kode ini digunakan untuk memvisualisasikan distribusi beberapa fitur numerik seperti LIMIT_BAL, AGE, BILL_AMT1, dan PAY_AMT1 dalam dataset. Grafik histogram menunjukkan bahwa fitur LIMIT_BAL dan BILL_AMT1 memiliki distribusi yang condong ke kanan, menandakan adanya data dengan nilai besar tetapi jarang muncul. Fitur AGE memiliki distribusi mendekati normal, dengan mayoritas berada dalam rentang usia 30 – 40 tahun. Fitur PAY_AMT1 menunjukkan mayoritas pembayaran berada di angka kecil, dengan beberapa outlier pembayaran besar.

```python
# Separate features (X) and target (y)
X = df.drop(columns=['default payment next month', 'ID'])
y = df['default payment next month']

# Standardize numeric features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42, stratify=y)

# Verify shapes of the splits
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

Output:


```python
# Separate features (X) and target (y)
X = df.drop(columns=['default payment next month', 'ID'])
y = df['default payment next month']

# Standardize numeric features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42, stratify=y)

# Verify shapes of the splits
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```


Analisis:

Kode ini bertujuan untuk mempersiapkan data sebelum diproses ke dalam model machine learning. Fitur X dipisahkan dari target, sementara kolom ID dihapus karena tidak relevan untuk analisis. Fitur numerik dalam X akan dinormalisasi menggunakan StandardScaler untuk memastikan distribusi yang seragam. Dataset yang telah dinormalisasi kemudian dibagi menjadi data train dan data test dengan rasio 80:20 menggunakan train_test_split, dengan statifikasi berdasarkan target agar distribusi kelas tetap seimbang di kedua set. Hasil pembagian ini diverifikasi dengan mencetak bentuk dari data latih (X_train, y_train) dan data uji (X_test, y_test).

```
y_train = y_train.values if isinstance(y_train, pd.Series) else y_train
y_test = y_test.values if isinstance(y_test, pd.Series) else y_test
```
✓ 0.0s

```
# Convert data to tensors
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
# Convert data to tensors
X_train_tensor = torch.tensor(X_train, dtype=torch.float32).to(device)
y_train_tensor = torch.tensor(y_train, dtype=torch.long).to(device)  # Long for classification
X_test_tensor = torch.tensor(X_test, dtype=torch.float32).to(device)
y_test_tensor = torch.tensor(y_test, dtype=torch.long).to(device)
```
✓ 0.0s

Output:

-

Analisis:

Kode ini mempersiapkan agar data latih dan uji kompatibel dengan model berbasis PyTorch. Langkah pertama memastikan bahwa target (y_train dan y_test) dikonversi menjadi array NumPy jika sebelumnya dalam bentuk pd.Series. Selanjutnya, data latih dan uji (X_train, X_test, y_train, y_test) diubah menjadi tensor PyTorch dengan tipe data float32 untuk fitur (X) dan long untuk target (y) yang sesuai dengan klasifikasi. Data tensor ini kemudian dipindahkan ke GPU (cuda) jika tersedia, atau ke CPU jika GPU tidak ada.

```python
# Define the MLP Classification model
class MLPClassification(nn.Module):
    def __init__(self, input_size, hidden_layers, activation):
        super(MLPClassification, self).__init__()
        layers = []
        for neurons in hidden_layers:
            layers.append(nn.Linear(input_size, neurons))
            if activation == 'relu':
                layers.append(nn.ReLU())
            elif activation == 'sigmoid':
                layers.append(nn.Sigmoid())
            elif activation == 'tanh':
                layers.append(nn.Tanh())
            elif activation == 'softmax':
                layers.append(nn.Softmax(dim=1))
            elif activation == 'linear':
                pass  # Linear activation is identity
            input_size = neurons
        layers.append(nn.Linear(input_size, 2))  # Output layer for 2 classes
        self.model = nn.Sequential(*layers)

    def forward(self, x):
        return self.model(x)

# Hyperparameter options
hidden_layers_options = [
    [4], [16], [32], [64],             # 1 hidden layer
    [4, 16], [16, 32], [32, 64],       # 2 hidden layers
    [4, 16, 32], [16, 32, 64]          # 3 hidden layers
]
activation_options = ["relu", "sigmoid", "tanh", "linear"]
epochs_options = [1, 10, 25, 50, 100, 250]
learning_rate_options = [10, 1, 0.1, 0.01, 0.001, 0.0001]
batch_size_options = [16, 32, 64, 128, 256, 512]

# Store results
results = []

# Device setup
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
```


```python
# Define the MLP Classification model
class MLPClassification(nn.Module):
    def __init__(self, input_size, hidden_layers, activation):
        super(MLPClassification, self).__init__()
        layers = []
        for neurons in hidden_layers:
            layers.append(nn.Linear(input_size, neurons))
            if activation == 'relu':
                layers.append(nn.ReLU())
            elif activation == 'sigmoid':
                layers.append(nn.Sigmoid())
            elif activation == 'tanh':
                layers.append(nn.Tanh())
            elif activation == 'softmax':
                layers.append(nn.Softmax(dim=1))
            elif activation == 'linear':
                pass  # Linear activation is identity
            input_size = neurons
        layers.append(nn.Linear(input_size, 2))  # Output layer for 2 classes
        self.model = nn.Sequential(*layers)

    def forward(self, x):
        return self.model(x)

# Hyperparameter options
hidden_layers_options = [
    [4], [16], [32], [64],             # 1 hidden layer
    [4, 16], [16, 32], [32, 64],       # 2 hidden layers
    [4, 16, 32], [16, 32, 64]          # 3 hidden layers
]
activation_options = ["relu", "sigmoid", "tanh", "linear"]
epochs_options = [1, 10, 25, 50, 100, 250]
learning_rate_options = [10, 1, 0.1, 0.01, 0.001, 0.0001]
batch_size_options = [16, 32, 64, 128, 256, 512]

# Store results
results = []

# Device setup
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
```


```python
                    model.eval()
                    with torch.no_grad():
                        y_pred_tensor = model(X_test_tensor)
                        y_pred_prob = torch.softmax(y_pred_tensor, axis=1).cpu().numpy()[:, 1]
                        y_pred = torch.argmax(y_pred_tensor, axis=1).cpu().numpy()
                        accuracy = accuracy_score(y_test, y_pred)
                        mae = mean_absolute_error(y_test, y_pred)
                        mse = mean_squared_error(y_test, y_pred)
                        r2 = r2_score(y_test, y_pred)

                        # Store results
                        results.append({
                            'hidden_layers': hidden_layers,
                            'activation': activation,
                            'epochs': epochs,
                            'learning_rate': lr,
                            'batch_size': batch_size,
                            'accuracy': accuracy,
                            'mae': mae,
                            'mse': mse,
                            'r2': r2
                        })

# Convert results to DataFrame
results_df = pd.DataFrame(results)

# Save results to CSV
results_df.to_csv("mlp_classification_results.csv", index=False)
print("All results have been saved to 'mlp_classification_results.csv'.")
```

Output:

```
All results have been saved to 'mlp_classification_results.csv'.
```

Analisis:

Kode ini membangun model klasifikasi berbasis Multi-Layer Perceptron (MLP) dengan arsitektur model. diatur dengan jumlah lapisan tersembunyi (hidden_layers) dan fungsi aktivasi (activation) yang dapat dikustomisasi, seperti relu, sigmoid, tanh, atau softmax. Model mencakup output layer dengan 2 neuron untuk klasifikasi dua kelas. Hyperparameter seperti jumlah epoch, learning rate, batch size, dan kombinasi hidden layers dieksplorasi untuk tuning model. Setelah pelatihan, model dievaluasi menggunakan metrik seperti akurasi, MAE, MSE, dan R^2.

```python
from tabulate import tabulate
# Display top 20 results
top_20_results = results_df.sort_values(by='accuracy', ascending=False).head(20)
# Display the top results in a tabular format
print(tabulate(top_20_results, headers='keys', tablefmt='pretty'))
```

✓ 0.0s

Output:

	hidden_layers	activation	epochs	learning_rate	batch_size	accuracy	mae	mse	r2
6460	[4, 16, 32]	sigmoid	250	0.1	256	0.821	0.179	0.179	-0.03917532955194347
6215	[4, 16, 32]	relu	100	0.01	512	0.821	0.179	0.179	-0.03917532955194347
7510	[16, 32, 64]	tanh	100	0.01	256	0.8203333333333334	0.17966666666666667	0.17966666666666667	-0.043045628730907778
6246	[4, 16, 32]	relu	250	0.01	16	0.8201666666666667	0.17983333333333335	0.17983333333333335	-0.04401320352564886
6249	[4, 16, 32]	relu	250	0.01	128	0.82	0.18	0.18	-0.044980778320389936
1025	[16]	relu	100	0.1	512	0.82	0.18	0.18	-0.044980778320389936
3182	[64]	tanh	100	0.1	64	0.8198333333333333	0.18016666666666667	0.18016666666666667	-0.04594835311513101
4484	[16, 32]	relu	100	0.01	64	0.8196666666666667	0.18033333333333335	0.18033333333333335	-0.04691592790987231
1498	[16]	tanh	250	0.01	256	0.8195	0.1805	0.1805	-0.04788350270461339
989	[16]	relu	50	0.1	512	0.8195	0.1805	0.1805	-0.04788350270461339
3869	[4, 16]	sigmoid	250	0.1	512	0.8195	0.1805	0.1805	-0.04788350270461339
1052	[16]	relu	250	1.0	64	0.8193333333333334	0.18066666666666667	0.18066666666666667	-0.048851077499354477
4052	[4, 16]	tanh	100	0.01	64	0.8193333333333334	0.18066666666666667	0.18066666666666667	-0.048851077499354477
3868	[4, 16]	sigmoid	250	0.1	256	0.8193333333333334	0.18066666666666667	0.18066666666666667	-0.048851077499354477
2712	[64]	relu	50	0.1	16	0.8191666666666667	0.18083333333333335	0.18083333333333335	-0.049818652294095545
4087	[4, 16]	tanh	250	0.01	32	0.8191666666666667	0.18083333333333335	0.18083333333333335	-0.049818652294095545
3651	[4, 16]	relu	250	0.1	128	0.819	0.181	0.181	-0.05078622708883662
1917	[32]	relu	250	1.0	128	0.819	0.181	0.181	-0.05078622708883662
4693	[16, 32]	sigmoid	100	0.1	32	0.819	0.181	0.181	-0.05078622708883662
1914	[32]	relu	250	1.0	16	0.819	0.181	0.181	-0.05078622708883662

Analisis:

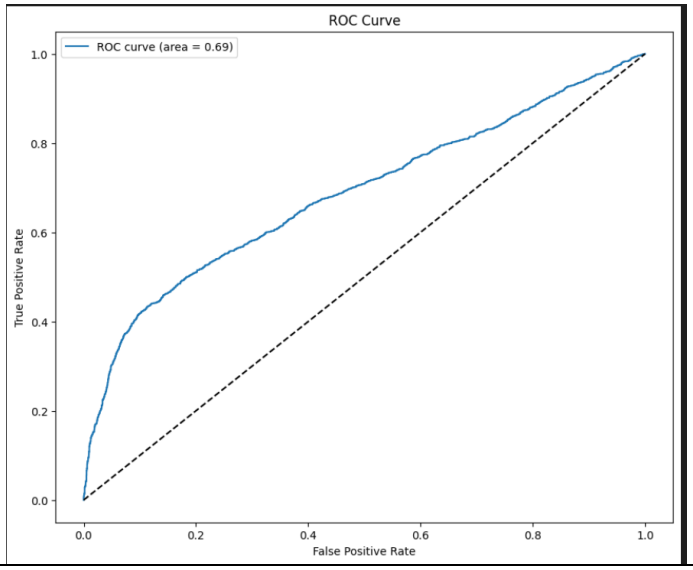
Hasil dari model MLP Classification menunjukkan akurasi yang cukup baik, mencapai 82.1% pada kombinasi hidden_layers = [4, 16, 32], activation = sigmoid, epochs = 250, learning_rate = 0.1, dan batch_size = 256. Namun, nilai R^2 yang negatif mengindikasikan bahwa model belum optimal dalam menjelaskan hubungan antara fitur dan target, kemungkinan akibat ketidakseimbangan kelas dalam dataset atau kurang relevannya beberapa fitur. Fungsi aktivasi seperti sigmoid dan struktur hidden layers yang kompleks berhasil meningkatkan performa model, sementara epoch lebih tinggi dan batch size besar memberikan stabilitas dalam pelatihan.

```python
from sklearn.metrics import roc_curve, auc, confusion_matrix

# 2. ROC Curve
plt.figure(figsize=(10, 8))
fpr, tpr, _ = roc_curve(y_test, y_pred_prob)
roc_auc = auc(fpr, tpr)
plt.plot(fpr, tpr, label=f'ROC curve (area = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], 'k--')
plt.title('ROC Curve')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend()
plt.show()
```

✓ 0.2s

Output :

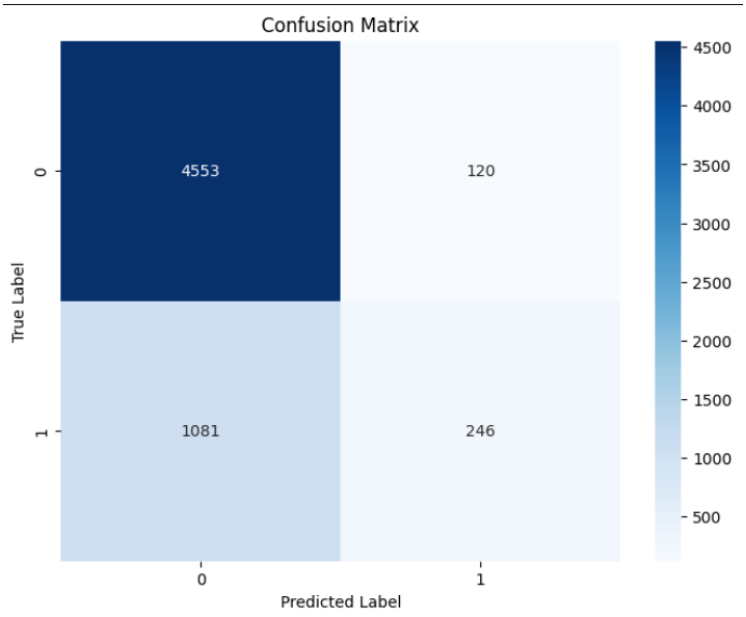


Analisis:
Kurva ROC (Receiver Operating Characteristic) menunjukkan kemampuan model dalam membedakan antara dua kelas berdasarkan probabilitas prediksi, dengan area di bawah kurva (AUC) sebesar 0.69. Nilai ini menunjukkan bahwa model memiliki performa yang lebih baik daripada prediksi acak (AUC = 0.5), tetapi masih jauh dari kategori model yang sangat baik (AUC > 0.8). Bentuk kurva yang relatif dekat dengan garis diagonal mengindikasikan bahwa model mungkin kesulitan dalam menangkap pola yang jelas untuk memisahkan kelas target, kemungkinan disebabkan oleh ketidakseimbangan kelas, kurang relevannya fitur, atau hyperparameter tuning yang belum optimal. Hasil ini juga dapat dipengaruhi oleh data dengan noise tinggi atau kurangnya representasi fitur yang signifikan terhadap target.

```python
# 3. Confusion Matrix
conf_matrix = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues')
plt.title('Confusion Matrix')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.show()
```

✓ 0.2s

Output:

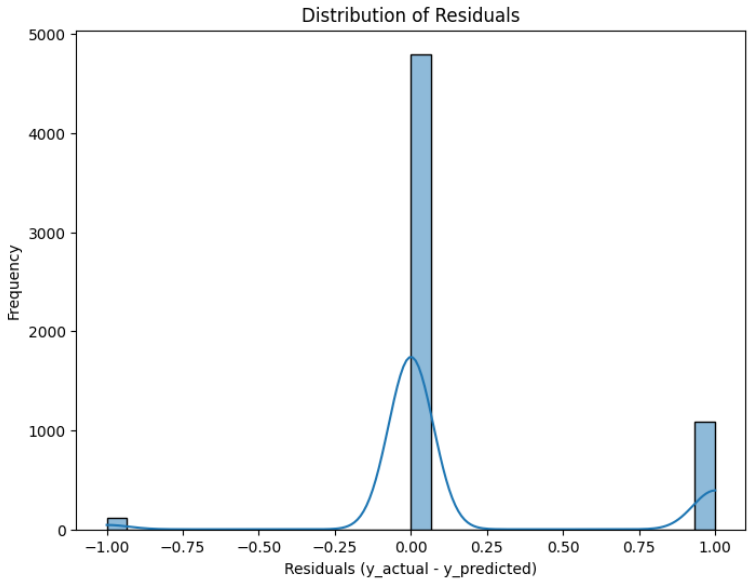


Analisis:
Kode ini menghasilkan confusion matrix yang memberikan rincian tentang performa model klasifikasi dengan membandingkan label sebenarnya (y_test) dengan label prediksi (y_pred). Confusion matrix terdiri dari empat elemen utama: true positive (TP), true negative (TN), false positive (FP), dan false negative (FN). Analisis confusion matrix memungkinkan evaluasi yang lebih mendalam terhadap kesalahan model, seperti ketidakseimbangan prediksi pada kelas tertentu. Jika hasil menunjukkan dominasi nilai diagonal (TP dan TN), maka model memiliki kemampuan klasifikasi yang baik. Namun, jika FN atau FP tinggi, ini mengindikasikan model kesulitan memisahkan kelas target. Hasil tersebut dapat terjadi karena ketidakseimbangan kelas dalam dataset, kurang relevannya fitur yang digunakan, atau model yang terlalu sederhana untuk menangkap pola kompleks.

```python
# Distribution of Errors (Residuals)
plt.figure(figsize=(8, 6))
residuals = y_test - y_pred  # Assuming y_pred is the final predictions from the best model
sns.histplot(residuals, kde=True, bins=30)
plt.title('Distribution of Residuals')
plt.xlabel('Residuals (y_actual - y_predicted)')
plt.ylabel('Frequency')
plt.show()
```

✓ 0.4s

Output:



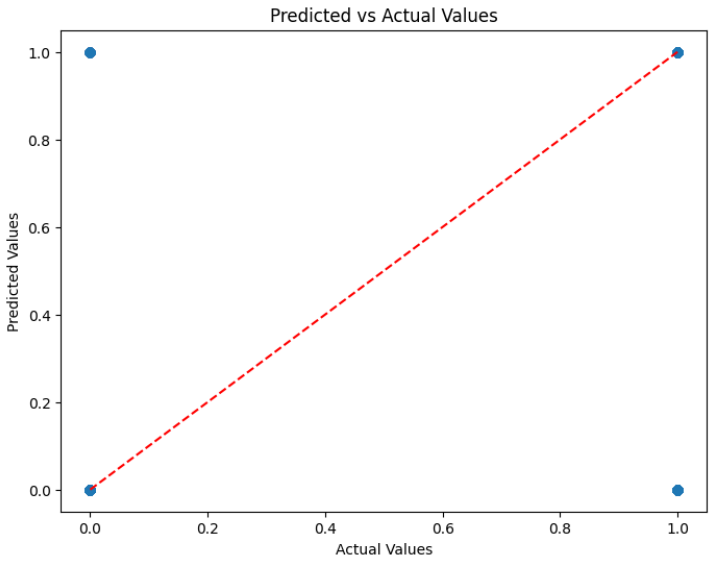
Analisis:

Distribusi residual dari model menunjukkan bahwa sebagian besar error terpusat di sekitar nol, yang berarti model dapat memprediksi dengan cukup baik pada sebagian besar data. Namun, adanya dua puncak tambahan di -1.0 dan 1.0 mengindikasikan kesalahan yang signifikan pada sebagian kecil data. Hal ini kemungkinan disebabkan oleh outlier dalam dataset atau distribusi data yang tidak seimbang, terutama jika terdapat kelas target dengan jumlah yang jauh lebih sedikit (class imbalance). Selain itu, distribusi residual yang tidak sepenuhnya simetris mengindikasikan bahwa model mungkin bias terhadap pola tertentu dalam data.

```python
# Scatter Plot: Predicted vs Actual
plt.figure(figsize=(8, 6))
plt.scatter(y_test, y_pred, alpha=0.5)
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], 'r--')  # Diagonal line
plt.title('Predicted vs Actual Values')
plt.xlabel('Actual Values')
plt.ylabel('Predicted Values')
plt.show()
```

✓ 0.2s

Output :



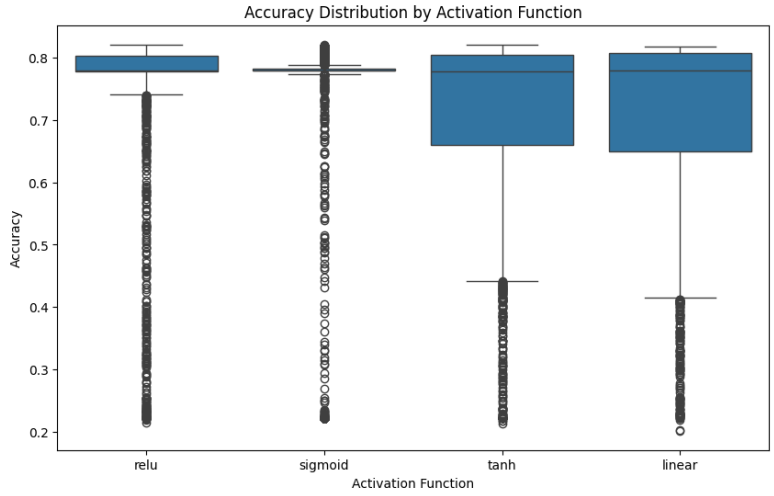
Analisis:

Scatter plot yang menampilkan prediksi (y_pred) versus nilai aktual (y_test) menunjukkan seberapa baik model menangkap pola hubungan antara input dan target. Pada grafik ini, titik-titik data seharusnya terdistribusi sepanjang garis diagonal merah (idealnya prediksi sama dengan nilai aktual). Namun, pola titik terlihat terdistribusi hanya pada dua lokasi utama (0 dan 1), yang menunjukkan bahwa model melakukan klasifikasi biner tanpa probabilitas antara kedua kelas. Pola ini konsisten dengan model klasifikasi yang baik jika kelas benar-benar terpisah secara jelas. Jika ada kesalahan prediksi, titik-titik akan berada jauh dari garis diagonal, mencerminkan kesalahan klasifikasi. Hasil ini mungkin disebabkan oleh dataset yang relatif sederhana atau model yang cukup baik menangkap pola data.

```python
# Boxplot: Accuracy by Activation Function
plt.figure(figsize=(10, 6))
sns.boxplot(x='activation', y='accuracy', data=results_df)
plt.title('Accuracy Distribution by Activation Function')
plt.xlabel('Activation Function')
plt.ylabel('Accuracy')
plt.show()
```

✓ 0.2s

Output:



Analisis:

Boxplot yang menunjukkan distribusi akurasi berdasarkan fungsi aktivasi memberikan wawasan penting tentang kinerja model. Dari hasil tersebut, fungsi aktivasi relu dan tanh memiliki distribusi akurasi yang lebih tinggi dan konsisten dibandingkan sigmoid dan linear. Hal ini dapat dijelaskan oleh karakteristik relu, yang secara efisien menangani masalah vanishing gradient dan memungkinkan konvergensi yang lebih cepat pada deep learning. Fungsi tanh, meskipun memiliki keunggulan dalam mempercepat konvergensi karena outputnya yang terpusat di sekitar nol, menunjukkan variabilitas yang lebih besar pada model tertentu. Sebaliknya, sigmoid memiliki distribusi akurasi yang lebih tersebar karena cenderung mengalami vanishing gradient saat digunakan di jaringan dengan banyak layer. Fungsi linear, yang biasanya lebih cocok untuk regresi, menunjukkan akurasi yang lebih rendah dan distribusi yang kurang stabil dalam tugas klasifikasi.

```
# 4. Classification Report
print("\nClassification Report:\n")
print(classification_report(y_test, y_pred))
```
✓ 0.0s

Output:

```
Classification Report:

              precision    recall  f1-score   support

           0       0.81      0.97      0.88      4673
           1       0.67      0.19      0.29      1327

    accuracy                           0.80      6000
   macro avg       0.74      0.58      0.59      6000
weighted avg       0.78      0.80      0.75      6000
```

Analisis:

Classification report menunjukkan bahwa model memiliki akurasi keseluruhan sebesar 80%, dengan perbedaan performa yang signifikan antara kedua kelas target. Untuk kelas 0 (non-default), precision, recall, dan F1-score cukup tinggi, masing-masing 81%, 97%, dan 88%, menandakan model sangat baik dalam mengidentifikasi data non-default. Namun, untuk kelas 1 (default), precision hanya 67%, recall sangat rendah di 19%, dan F1-score hanya 29%. Hasil ini mengindikasikan bahwa model kesulitan mengidentifikasi data default, yang kemungkinan disebabkan oleh ketidakseimbangan kelas, di mana jumlah data kelas 0 jauh lebih banyak daripada kelas 1. Model cenderung bias terhadap kelas mayoritas (0) karena distribusi data yang tidak seimbang, yang tercermin pada nilai recall rendah untuk kelas 1.