

Nama : Dimas Tri Wicaksono

Kelas : Bravo(Malam)

Pertemuan 52

```
# Mengimpor pustaka yang diperlukan
import tensorflow as tf
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder

file_path = 'C:/Users/Dimas Tri Wicaksono/MSIB2024-2/myenvironment/Week 11/winequality-white.csv'
data = pd.read_csv(file_path, delimiter=';')
```

```
# Membagi dataset menjadi fitur (X) dan label (y)
X = data.drop('quality', axis=1) # Fitur
y = data['quality'] # Target (kualitas anggur)

# Mengencode label (karena kualitas anggur adalah nilai integer, tidak perlu diubah ke one-hot encoding)
label_encoder = LabelEncoder()
y_encoded = label_encoder.fit_transform(y)

# Membagi dataset menjadi data training dan data uji
X_train, X_test, y_train, y_test = train_test_split(X, y_encoded, test_size=0.2, random_state=42, stratify=y_encoded)
```

```
# Standarisasi fitur (normalisasi data)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

```
# Membangun model Deep Learning untuk klasifikasi multi-kelas
model = tf.keras.models.Sequential([
    tf.keras.layers.Dense(128, activation='relu', input_shape=(X_train.shape[1],)), # Input layer
    tf.keras.layers.Dropout(0.3), # Dropout layer untuk mencegah overfitting
    tf.keras.layers.Dense(64, activation='relu'), # Hidden layer 1
    tf.keras.layers.Dropout(0.3), # Dropout layer untuk mencegah overfitting
    tf.keras.layers.Dense(32, activation='relu'), # Hidden layer 2
    tf.keras.layers.Dense(7, activation='softmax') # Output layer untuk 7 kelas (kualitas 0-6)
])

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

# Menggunakan EarlyStopping untuk mencegah overfitting
early_stopping = tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True)


# Melatih model
model.fit(X_train, y_train, epochs=100, batch_size=16, validation_data=(X_test, y_test), callbacks=[early_stopping])
```

```
# Evaluasi model pada data uji
test_loss, test_acc = model.evaluate(X_test, y_test)
print(f'Test Accuracy: {test_acc:.4f}')
```

```
# Prediksi dengan data baru (misal, data baru dengan 11 fitur sesuai format dataset)
def predict_wine_quality(new_data):
    # Melakukan standarisasi data baru sebelum prediksi
    new_data_scaled = scaler.transform([new_data])
    prediction = model.predict(new_data_scaled)
    predicted_class = np.argmax(prediction) # Mengambil kelas dengan probabilitas tertinggi
    return predicted_class
```


```
31/31 0s 921us/step - accuracy: 0.5430 - loss: 1.0543
Test Accuracy: 0.5633
```

```
# Contoh data anggur baru sesuai format dataset Wine Quality (11 fitur)
new_wine = np.array([7.0, 0.27, 0.36, 20.7, 0.045, 45.0, 170.0, 1.0010, 3.00, 0.45, 8.8])
```

```
> 
# Contoh data anggur baru sesuai format dataset Wine Quality (11 fitur)
new_wine = np.array([7.0, 0.27, 0.36, 20.7, 0.045, 45.0, 170.0, 1.0010, 3.00, 0.45, 8.8])

# Memprediksi kualitas untuk anggur baru
predicted_quality = predict_wine_quality(new_wine)
print(f"Prediksi kualitas untuk anggur baru: {predicted_quality}")

[11] ✓ 0.1s Python

... 1/1  0s 64ms/step
Prediksi kualitas untuk anggur baru: 2
```