**LAPORAN PRAKTIKUM**

**PEMBELAJARAN MESIN AP**

**“Supervised Learning”**

Dosen Pengampu:

**Drs. Hari Suparwito, S.J., M.App.IT**



Oleh:

**Samuel Santoso / 235314011**

PROGRAM STUDI INFORMATIKA

FAKULTAS SAINS DAN TEKNOLOGI

UNIVERSITAS SANATA DHARMA

2025

| **Python Code** | **Keterangan** |
| --- | --- |
| # Import required libraries  import pandas as pd  import numpy as np  import matplotlib.pyplot as plt  import sklearn  from sklearn.neural\_network import MLPClassifier  from sklearn.model\_selection import train\_test\_split, GridSearchCV  from sklearn.preprocessing import MinMaxScaler | * import pandas as pd. Mengimpor pustaka Pandas dengan alias pd. * import numpy as np. Mengimpor pustaka NumPy dengan nama np. * import matplotlib.pyplot as plt. Mengimpor modul pyplot dari pustaka Matplotlib dengan nama plt. * import sklearn. Mengimpor pustaka scikit-learn dengan nama sklearn. Menyediakan alat untuk pemodelan prediktif dan analisis data, termasuk algoritma machine learning.\ * from sklearn.neural\_network import MLPClassifier. Mengimpor kelas MLPClassifier dari modul neural\_network di scikit-learn. * from sklearn.model\_selection import train\_test\_split, GridSearchCV  Mengimpor fungsi train\_test\_split dan kelas GridSearchCV dari modul model\_selection di scikit-learn: * train\_test\_split: Digunakan untuk membagi dataset menjadi subset pelatihan dan pengujian secara acak. * GridSearchCV: Digunakan untuk melakukan pencarian grid secara menyeluruh pada kombinasi parameter untuk menemukan konfigurasi terbaik. * from sklearn.preprocessing import MinMaxScaler. Mengimpor kelas MinMaxScaler dari modul preprocessing di scikit-learn. |
| # Load and clean the dataset  df = pd.read\_csv('milknew(rapi).csv')  df = df['pH;Temprature;Taste;Odor;Fat ;Turbidity;Colour;Grade'].str.split(';', expand=True)  df.columns = ['pH', 'Temperature', 'Taste', 'Odor', 'Fat', 'Turbidity', 'Colour', 'Grade']  # Convert relevant columns to numeric types  numeric\_columns = ['pH', 'Temperature', 'Taste', 'Odor', 'Fat', 'Turbidity', 'Colour']  df[numeric\_columns] = df[numeric\_columns].apply(pd.to\_numeric)  print(df.head()) | * df = pd.read\_csv('milknew(rapi).csv'). Membaca file CSV bernama 'milknew(rapi).csv' dan memuat ke dalam DataFrame df. read\_csv mengasumsikan bahwa pemisah antar kolom adalah koma (;). * df = df['pH;Temprature;Taste;Odor;Fat ;Turbidity;Colour;Grade'].str.split(';', expand=True). Mengakses kolom tunggal dengan nama 'pH;Temprature;Taste;Odor;Fat ;Turbidity;Colour;Grade' dan kemudian membaginya menjadi beberapa kolom berdasarkan pemisah titik koma (;). * df.columns = ['pH', 'Temperature', 'Taste', 'Odor', 'Fat', 'Turbidity', 'Colour', 'Grade']. Menetapkan nama-nama kolom untuk DataFrame df. * numeric\_columns = ['pH', 'Temperature', 'Taste', 'Odor', 'Fat', 'Turbidity', 'Colour']. Membuat daftar kolom yang seharusnya berisi nilai numerik. * df[numeric\_columns] = df[numeric\_columns].apply(pd.to\_numeric). Mengonversi kolom-kolom yang ditentukan dalam numeric\_columns menjadi tipe data numerik. * print(df.head()). Menampilkan lima baris pertama dari DataFrame df . |
| # Normalize predictors  target\_column = 'Grade'  predictors = [col for col in df.columns if col != target\_column]  # Apply Min-Max Scaling  scaler = MinMaxScaler()  df[predictors] = scaler.fit\_transform(df[predictors])  # Display normalized summary  summary = df.describe().transpose()  print(summary) | * target\_column = 'Grade'. Menetapkan nama kolom target (label) dalam dataset, yaitu 'Grade'. * predictors = [col for col in df.columns if col != target\_column]. Membuat daftar kolom prediktor dengan mengecualikan kolom target 'Grade'. * scaler = MinMaxScaler(). Membuat objek MinMaxScaler dari pustaka scikit-learn. * df[predictors] = scaler.fit\_transform(df[predictors]). Menerapkan normalisasi Min-Max pada kolom-kolom prediktor: * fit\_transform: Metode ini menyesuaikan scaler dengan data (menghitung nilai minimum dan maksimum untuk setiap fitur), kemudian mentransformasikan data sesuai dengan skala yang ditentukan. * Hasil transformasi menggantikan nilai asli dalam DataFrame df untuk kolom-kolom prediktor. * summary = df.describe().transpose(). Menghasilkan ringkasan statistik deskriptif untuk setiap kolom dalam DataFrame df, lalu mentransposenya agar setiap fitur ditampilkan sebagai baris. * print(summary). Menampilkan ringkasan statistik yang telah dihasilkan. |
| # Step 4 - Creating the Training and Test Datasets  X = df[predictors]  y = df[target\_column]  # Split the dataset  X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  print('Training set size:', X\_train.shape)  print('Test set size:', X\_test.shape) | * X = df[predictors]. Membuat variabel X yang berisi fitur-fitur prediktor dari DataFrame df. * y = df[target\_column]. Membuat variabel y yang berisi kolom target dari DataFrame df. * X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42). Menggunakan fungsi train\_test\_split dari pustaka scikit-learn untuk membagi dataset menjadi subset pelatihan dan pengujian: * test\_size=0.2: Menentukan bahwa 20% dari data akan digunakan sebagai data pengujian, sedangkan 80% sisanya akan digunakan untuk pelatihan. * random\_state=42: Menetapkan seed untuk generator angka acak agar pembagian data dapat direproduksi. * Fungsi ini mengembalikan empat variabel:   + X\_train: Fitur-fitur untuk pelatihan.   + X\_test: Fitur-fitur untuk pengujian.   + y\_train: Label untuk pelatihan.   + y\_test: Label untuk pengujian. * print('Training set size:', X\_train.shape). Menampilkan ukuran dari dataset pelatihan, yaitu jumlah sampel dan fitur yang digunakan. * print('Test set size:', X\_test.shape). Menampilkan ukuran dari dataset pengujian, yaitu jumlah sampel dan fitur yang digunakan. |
| from sklearn.neural\_network import MLPClassifier  mlp = MLPClassifier(hidden\_layer\_sizes=(8,8,8), activation='relu', solver='adam', max\_iter=500)  mlp.fit(X\_train,y\_train)  predict\_train = mlp.predict(X\_train)  predict\_test = mlp.predict(X\_test) | * from sklearn.neural\_network import MLPClassifier. Mengimpor kelas MLPClassifier dari modul neural\_network di pustaka scikit-learn. * mlp = MLPClassifier(hidden\_layer\_sizes=(8,8,8), activation='relu', solver='adam', max\_iter=500). Membuat instance dari MLPClassifier dengan parameter sebagai berikut: * hidden\_layer\_sizes=(8,8,8): Menentukan arsitektur jaringan dengan tiga lapisan tersembunyi, masing-masing terdiri dari 8 neuron. * activation='relu': Menggunakan fungsi aktivasi ReLU (Rectified Linear Unit) untuk neuron-neuron di lapisan tersembunyi. * solver='adam': Menggunakan algoritma optimasi Adam (Adaptive Moment Estimation) untuk memperbarui bobot selama pelatihan. * max\_iter=500: Menetapkan jumlah maksimum iterasi pelatihan menjadi 500. * mlp.fit(X\_train, y\_train). Melatih model mlp menggunakan data pelatihan X\_train dan label y\_train. * predict\_train = mlp.predict(X\_train). Menggunakan model terlatih untuk memprediksi label pada data pelatihan. * predict\_test = mlp.predict(X\_test). Menggunakan model terlatih untuk memprediksi label pada data pengujian. |
| # Hyperparameter tuning using GridSearchCV  from sklearn.exceptions import ConvergenceWarning  import warnings  warnings.filterwarnings('ignore', category=ConvergenceWarning) | * Mengimpor kelas ConvergenceWarning dari sklearn.exceptions dan modul warnings. * Menggunakan warnings.filterwarnings untuk mengabaikan peringatan konvergensi yang mungkin muncul. |
| param\_grid = {  'hidden\_layer\_sizes': [(8,8,8), (16,16,16), (32,32,32)],  'activation': ['relu', 'tanh', 'logistic'],  'solver': ['adam', 'sgd'],  'learning\_rate\_init': [0.001, 0.01, 0.1],  'max\_iter': [500, 1000, 2000]  } | * param\_grid adalah kamus yang mendefinisikan kombinasi hyperparameter yang akan diuji oleh GridSearchCV. |
| grid\_search = GridSearchCV(MLPClassifier(random\_state=42), param\_grid, cv=3, scoring='accuracy', verbose=2) | Membuat objek GridSearchCV dengan parameter:   * MLPClassifier(random\_state=42): Estimator yang akan dituning. * param\_grid: Ruang hyperparameter yang telah didefinisikan sebelumnya. * cv=3: Menggunakan 3-fold cross-validation. * scoring='accuracy': Menggunakan akurasi sebagai metrik evaluasi. * verbose=2: Menampilkan log proses pelatihan secara rinci. |
| # Fit the model and handle potential errors  try:  grid\_search.fit(X\_train, y\_train)  # Best parameters and best score  print('Best Parameters:', grid\_search.best\_params\_)  print('Best Cross-Validation Accuracy:', grid\_search.best\_score\_)  # Train the model with the best parameters  best\_mlp = grid\_search.best\_estimator\_  best\_mlp.fit(X\_train, y\_train)  # Evaluate the model  predict\_test = best\_mlp.predict(X\_test)  test\_accuracy = accuracy\_score(y\_test, predict\_test)  print('Test Accuracy with Best Parameters:', test\_accuracy)  except Exception as e:  print('Error during GridSearchCV:', e) | * grid\_search.fit(X\_train, y\_train): Melatih model untuk setiap kombinasi hyperparameter menggunakan data pelatihan. * grid\_search.best\_params\_: Mengembalikan kombinasi hyperparameter terbaik berdasarkan hasil cross-validation. * grid\_search.best\_score\_: Mengembalikan skor akurasi tertinggi yang dicapai selama cross-validation. * grid\_search.best\_estimator\_: Mengembalikan model yang dilatih dengan kombinasi hyperparameter terbaik. * best\_mlp.fit(X\_train, y\_train): Melatih ulang model terbaik pada seluruh data pelatihan. * best\_mlp.predict(X\_test): Menggunakan model terbaik untuk memprediksi label pada data pengujian. * accuracy\_score(y\_test, predict\_test): Menghitung akurasi prediksi pada data pengujian. * try-except: Menangani potensi kesalahan yang mungkin terjadi selama proses pelatihan dan pencarian hyperparameter. |
| from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix | * accuracy\_score: Menghitung proporsi prediksi yang benar dibandingkan dengan total prediksi. * classification\_report: Menyediakan laporan teks yang merinci metrik klasifikasi utama seperti presisi, recall, dan f1-score untuk setiap kelas. * confusion\_matrix: Membuat matriks yang menunjukkan jumlah prediksi benar dan salah yang diklasifikasikan oleh model.​ |
| # Training accuracy  train\_accuracy = accuracy\_score(y\_train, predict\_train)  print('Training Accuracy:', train\_accuracy) | * accuracy\_score(y\_train, predict\_train): Menghitung akurasi model pada data pelatihan dengan membandingkan label sebenarnya (y\_train) dengan prediksi model (predict\_train). * print: Menampilkan akurasi pelatihan. |
| # Test accuracy  test\_accuracy = accuracy\_score(y\_test, predict\_test)  print('Test Accuracy:', test\_accuracy) | * accuracy\_score(y\_test, predict\_test): Menghitung akurasi model pada data pengujian dengan membandingkan label sebenarnya (y\_test) dengan prediksi model (predict\_test). * print: Menampilkan akurasi pengujian.​ |
| # Classification report  print('Classification Report:\n', classification\_report(y\_test, predict\_test)) | * classification\_report(y\_test, predict\_test): Menghasilkan laporan yang mencakup metrik seperti presisi, recall, dan f1-score untuk setiap kelas. * print: Menampilkan laporan klasifikasi.​ |
| # Confusion matrix  conf\_matrix = confusion\_matrix(y\_test, predict\_test)  print('Confusion Matrix:\n', conf\_matrix) | * confusion\_matrix(y\_test, predict\_test): Menghasilkan matriks kebingungan yang menunjukkan jumlah prediksi benar dan salah untuk setiap kelas. * print: Menampilkan matriks kebingungan.​ |
| # Visualize the confusion matrix  import seaborn as sns | * Mengimpor pustaka Seaborn dengan nama sns |
| plt.figure(figsize=(8, 6)) | * Mengatur ukuran figure (gambar) matplotlib menjadi 8 inci lebar dan 6 inci tinggi. |
| sns.heatmap(conf\_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=mlp.classes\_, yticklabels=mlp.classes\_) | * conf\_matrix: Data confusion matrix yang akan divisualisasikan. * annot=True: Menampilkan nilai numerik dalam setiap sel. * fmt='d': Format angka sebagai bilangan bulat. * cmap='Blues': Menggunakan skema warna biru. * xticklabels=mlp.classes\_, yticklabels=mlp.classes\_: Menetapkan label sumbu x dan y sesuai dengan kelas yang diprediksi oleh model mlp. |
| plt.xlabel('Predicted')  plt.ylabel('Actual')  plt.title('Confusion Matrix')  plt.show() | * Menambahkan label pada sumbu x dan y, memberikan judul pada plot, dan menampilkan plot. |