**LAPORAN PRAKTIKUM**

**PEMBELAJARAN MESIN AP**

**“**[**Supervised Learning (tree algorithm)**](https://belajar.usd.ac.id/course/view.php?id=39859#section-10)**”**

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 Random Forest** | **Keterangan** |
| --- | --- |
| import pandas as pd  import numpy as np  import seaborn as sns  import matplotlib.pyplot as plt  from sklearn.model\_selection import train\_test\_split  from sklearn.ensemble import RandomForestClassifier  from sklearn.metrics import classification\_report, confusion\_matrix | * **pandas**: Digunakan untuk manipulasi dan analisis data. * **numpy**: Digunakan untuk operasi numerik. * **seaborn** dan **matplotlib.pyplot**: Digunakan untuk visualisasi data. * **train\_test\_split**: Membagi dataset menjadi subset pelatihan dan pengujian. * **RandomForestClassifier**: Model klasifikasi berbasis ensemble dari pohon keputusan. * **classification\_report** dan **confusion\_matrix**: Digunakan untuk mengevaluasi performa model klasifikasi.​ |
| dataset = pd.read\_csv(  "D:\\UNTUK KULIAH SEMESTER 4\\Pembelajaran Mesin A\\CODE\\Pertemuan 7\\WEC\_Perth\_49\_1.csv",  delimiter=';'  ) | * Membaca file CSV dengan delimiter ; menggunakan pandas.read\_csv. |
| print(dataset.head())  print(dataset.info()) | * **dataset.head()**: Menampilkan 5 baris pertama dari dataset. * **dataset.info()**: Menampilkan informasi ringkas tentang dataset, termasuk jumlah entri, tipe data, dan jumlah nilai non-null.​ |
| dataset.head()  dataset.info()  dataset.describe()  g = sns.pairplot(dataset, hue='Total\_Power', diag\_kind='hist')  g.fig.suptitle("Scatterplot and histogram of pairs of variables color coded by Total\_Power",  fontsize=14, y=1.05) | * **dataset.describe()**: Memberikan statistik deskriptif dari dataset. * **sns.pairplot()**: Membuat plot pasangan variabel untuk melihat hubungan antar variabel, dengan warna berdasarkan nilai Total\_Power. Parameter diag\_kind='hist' menampilkan histogram pada diagonal. |
| if 'Total\_Power' in dataset.columns:  X = dataset.drop('Total\_Power', axis=1)  y = dataset['Total\_Power']  else:  print("Error: 'Total\_Power' column not found in the dataset.") | * **X**: Fitur (variabel independen) dengan menghapus kolom Total\_Power. * **y**: Target (variabel dependen) yaitu kolom Total\_Power. |
| X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42) | * Membagi dataset menjadi 70% data pelatihan dan 30% data pengujian. Parameter random\_state=42 memastikan hasil yang konsisten. |
| rf\_model = RandomForestClassifier(n\_estimators=100, random\_state=42)  rf\_model.fit(X\_train, y\_train) | * **RandomForestClassifier**: Membuat model Random Forest dengan 100 pohon keputusan. * **fit()**: Melatih model menggunakan data pelatihan. |
| y\_pred = rf\_model.predict(X\_test) | * **predict()**: Menggunakan model terlatih untuk memprediksi label pada data pengujian. |
| scaler = StandardScaler()  scaled\_data = scaler.fit\_transform(numerical\_data) | * StandardScale: Menstandarkan data agar memiliki mean 0 dan standar deviasi 1 * fit\_tranfrom: Menerapkan standarisasi pada data |
| print("Confusion Matrix:")  print(confusion\_matrix(y\_test, y\_pred))  print("\nClassification Report:")  print(classification\_report(y\_test, y\_pred)) | * **confusion\_matrix**: Menampilkan matriks kebingungan yang menunjukkan jumlah prediksi benar dan salah. * **classification\_report**: Memberikan metrik evaluasi seperti precision, recall, f1-score, dan support untuk setiap kelas. |

| **Python Code AdaBoost** | **Keterangan** |
| --- | --- |
| import pandas as pd  from sklearn.model\_selection import train\_test\_split  from sklearn.ensemble import AdaBoostClassifier  from sklearn.metrics import accuracy\_score | * **pandas**: Digunakan untuk manipulasi dan analisis data. * **train\_test\_split**: Membagi dataset menjadi subset pelatihan dan pengujian. * **AdaBoostClassifier**: Model klasifikasi berbasis ensemble yang meningkatkan akurasi dengan menggabungkan beberapa model lemah. * **accuracy\_score**: Menghitung akurasi prediksi model |
| file\_path = r'd:\\UNTUK KULIAH SEMESTER 4\\Pembelajaran Mesin A\\CODE\\Pertemuan 7\\WEC\_Perth\_49\_1.csv'  data = pd.read\_csv(file\_path) | * Membaca file CSV yang berisi dataset menggunakan pandas.read\_csv. |
| if data.empty:  raise ValueError("The dataset is empty. Please check the file path or the dataset content.")  if data.shape[1] < 2:  raise ValueError("The dataset must have at least one feature column and one target column.") | * Memastikan bahwa dataset tidak kosong dan memiliki setidaknya satu fitur dan satu target. |
| X = data.iloc[:, :-1].values  y = data.iloc[:, -1].values | * **X**: Semua kolom kecuali kolom terakhir sebagai fitur. * **y**: Kolom terakhir sebagai target.​ |
| if X.shape[1] == 0:  raise ValueError("No feature columns found in the dataset. Please check the dataset structure.")  if len(y) == 0:  raise ValueError("No target values found in the dataset. Please check the dataset structure.") | * Memastikan bahwa fitur dan target memiliki data yang cukup untuk pelatihan. |
| X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42) | * Membagi dataset menjadi 80% data pelatihan dan 20% data pengujian dengan seed acak untuk reproduktibilitas. |
| model = AdaBoostClassifier() | * Membuat instance dari AdaBoostClassifier dengan parameter default. |
| model.fit(X\_train, y\_train) | * Melatih model AdaBoost menggunakan data pelatihan.​ |
| y\_pred = model.predict(X\_test)  accuracy = accuracy\_score(y\_test, y\_pred)  print('Accuracy: %.3f' % accuracy) | * Memprediksi label untuk data pengujian dan menghitung akurasi prediksi. |
| row = [[10]]  yhat = model.predict(row)  print('Predicted Class: %d' % yhat[0]) | * Melakukan prediksi untuk satu sampel data. |

| **Python Code GBM** | **Keterangan** |
| --- | --- |
| import pandas as pd  from sklearn.compose import ColumnTransformer  from sklearn.ensemble import GradientBoostingClassifier  from sklearn.metrics import classification\_report  from sklearn.model\_selection import cross\_val\_score, train\_test\_split  from sklearn.pipeline import Pipeline  from sklearn.preprocessing import OneHotEncoder, StandardScaler | * **pandas**: Digunakan untuk manipulasi dan analisis data. * **ColumnTransformer**: Menerapkan transformasi yang berbeda pada kolom tertentu dalam dataset. * **GradientBoostingClassifier**: Model klasifikasi berbasis ensemble yang membangun model secara bertahap dengan mengoptimalkan fungsi kerugian. * **classification\_report**: Membuat laporan teks yang menunjukkan metrik utama klasifikasi seperti precision, recall, dan f1-score. * **cross\_val\_score**: Melakukan evaluasi model dengan cross-validation. * **train\_test\_split**: Membagi dataset menjadi subset pelatihan dan pengujian. * **Pipeline**: Menyusun beberapa langkah pemrosesan dan model menjadi satu kesatuan. * **OneHotEncoder**: Mengubah fitur kategorikal menjadi format numerik dengan one-hot encoding. * **StandardScaler**: Menstandarisasi fitur dengan menghapus mean dan menskalakan ke unit variance.​ |
| data = pd.read\_csv("WEC\_Perth\_49\_1.csv") | * Membaca file CSV yang berisi dataset menggunakan pandas.read\_csv. |
| print("Columns in dataset:", data.columns) | * Menampilkan semua nama kolom dalam dataset untuk memastikan struktur data. |
| X = data.drop("Total\_Power", axis=1, errors="ignore")  y = data.get("Total\_Power") | * **X**: Semua kolom kecuali 'Total\_Power' sebagai fitur. * **y**: Kolom 'Total\_Power' sebagai target. |
| if y is None:  raise ValueError("The target column 'Total\_Power' is missing from the dataset.") | * Memastikan bahwa kolom target 'Total\_Power' ada dalam dataset. |
| X\_train, X\_test, y\_train, y\_test = train\_test\_split(  X, y, test\_size=0.2, random\_state=42  ) | * Membagi dataset menjadi 80% data pelatihan dan 20% data pengujian dengan seed acak untuk reproduktibilitas. |
| categorical\_features = X.select\_dtypes(  include=["object"]  ).columns.tolist()  numerical\_features = X.select\_dtypes(  include=["float64", "int64"]  ).columns.tolist() | * **categorical\_features**: Daftar kolom dengan tipe data objek (kategorikal). * **numerical\_features**: Daftar kolom dengan tipe data numerik. |
| preprocessor = ColumnTransformer(  transformers=[  ("cat", OneHotEncoder(), categorical\_features),  ("num", StandardScaler(), numerical\_features),  ]  ) | * Menerapkan OneHotEncoder pada fitur kategorikal dan StandardScaler pada fitur numerik. |
| pipeline = Pipeline(  [  ("preprocessor", preprocessor),  ("classifier", GradientBoostingClassifier(random\_state=42)),  ]  ) | * StandardScale: Menstandarkan data agar memiliki mean 0 dan standar deviasi 1 * Menyusun pipeline yang pertama-tama memproses data menggunakan preprocessor, lalu melatih model GradientBoostingClassifier. |
| cv\_scores = cross\_val\_score(pipeline, X\_train, y\_train, cv=5) | * Melakukan evaluasi model dengan 5-fold cross-validation pada data pelatihan. |
| pipeline.fit(X\_train, y\_train) | * Melatih pipeline (preprocessing + model) pada data pelatihan. |
| y\_pred = pipeline.predict(X\_test) | * Menggunakan model terlatih untuk memprediksi label pada data pengujian. |
| report = classification\_report(y\_test, y\_pred) | * Membuat laporan yang menunjukkan metrik seperti precision, recall, f1-score, dan support untuk setiap kelas. |
| print(f"Mean Cross-Validation Accuracy: {cv\_scores.mean():.4f}")  print("\nClassification Report:")  print(report) | * Menampilkan rata-rata akurasi dari cross-validation dan laporan klasifikasi pada data pengujian. |

| **Python Code XGBoost** | **Keterangan** |
| --- | --- |
| from xgboost import XGBClassifier  from sklearn.model\_selection import train\_test\_split  from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix  import pandas as pd | * **XGBClassifier**: Model klasifikasi dari XGBoost yang menggunakan boosting gradien. * **train\_test\_split**: Fungsi untuk membagi dataset menjadi subset pelatihan dan pengujian. * **accuracy\_score**, **classification\_report**, **confusion\_matrix**: Metrik evaluasi untuk menilai kinerja model. * **pandas**: Library untuk manipulasi dan analisis data. |
| data = pd.read\_csv('WEC\_Perth\_49\_1.csv') # Ganti dengan file dataset Anda | * Membaca file CSV yang berisi dataset ke dalam DataFrame data.; |
| X = data.drop('Total\_Power', axis=1) # Ganti 'target' dengan nama kolom target Anda  y = data['Total\_Power'] | * **X**: Semua kolom kecuali 'Total\_Power', digunakan sebagai fitur. * **y**: Kolom 'Total\_Power', digunakan sebagai target. |
| X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42) | * Membagi data menjadi 80% untuk pelatihan dan 20% untuk pengujian. * **random\_state=42**: Menetapkan seed untuk memastikan hasil yang konsisten. |
| model = XGBClassifier(n\_estimators=100, learning\_rate=0.1, random\_state=42, use\_label\_encoder=False)  model.fit(X\_train, y\_train, eval\_metric='logloss') | * **n\_estimators=100**: Menentukan jumlah pohon yang akan dibangun. * **learning\_rate=0.1**: Menentukan seberapa besar kontribusi setiap pohon terhadap model akhir. * **random\_state=42**: Menetapkan seed untuk reproduktibilitas. * **use\_label\_encoder=False**: Menonaktifkan label encoder bawaan XGBoost yang telah usang. * **eval\_metric='logloss'**: Menggunakan log-loss sebagai metrik evaluasi selama pelatihan |
| y\_pred = model.predict(X\_test)  accuracy = accuracy\_score(y\_test, y\_pred)  print(f"XGBoost Accuracy: {accuracy}") | * **y\_pred**: Prediksi label untuk data pengujian. * **accuracy**: Menghitung akurasi prediksi. * Menampilkan akurasi model. |
| print("Classification Report:")  print(classification\_report(y\_test, y\_pred))  print("Confusion Matrix:")  print(confusion\_matrix(y\_test, y\_pred)) | * **classification\_report**: Menampilkan metrik seperti precision, recall, dan f1-score untuk setiap kelas. * **confusion\_matrix**: Menampilkan matriks kebingungan yang menunjukkan prediksi benar dan salah. |