

UAS MACHINE LEARNING

Akhmad Maulana Fadil - 231011401337

DATASET IRIS (KLASIFIKASI)

Pendahuluan

Machine Learning merupakan salah satu bidang dalam kecerdasan buatan yang memungkinkan sistem komputer untuk belajar dari data tanpa diprogram secara eksplisit. Salah satu algoritma yang populer dan mudah dipahami adalah Decision Tree. Algoritma ini bekerja dengan membentuk struktur pohon keputusan yang digunakan untuk melakukan klasifikasi atau regresi.

Pada tugas UAS ini, dilakukan implementasi algoritma Decision Tree untuk mengklasifikasikan jenis bunga Iris berdasarkan panjang dan lebar sepal serta petal menggunakan dataset Iris.

1. Teori Singkat

Decision Tree adalah algoritma pembelajaran mesin yang digunakan untuk klasifikasi dan regresi. Algoritma ini memetakan observasi tentang suatu item menjadi kesimpulan tentang nilai targetnya dalam struktur yang menyerupai pohon, di mana setiap jalur dari akar ke daun mewakili aturan klasifikasi

Konsepnya :

- Node: Titik yang mewakili fitur atau atribut dalam data.
- Root: Node paling atas yang mewakili seluruh sampel dan fitur pertama yang digunakan untuk pembelahan.
- Leaf: Node akhir yang memberikan keputusan atau label kelas (tidak dapat dibagi lagi).
- Splitting: Proses membagi sebuah node menjadi dua atau lebih sub-node berdasarkan kondisi tertentu.
- Pruning: Proses pemangkasan

Perbedaan Decision Tree, Random Forest, dan Gradient Boosting

- Decision Tree: Satu pohon tunggal yang membuat keputusan secara sekuensial.
- Random Forest: Kumpulan banyak pohon keputusan yang dibangun secara paralel dan hasilnya diambil berdasarkan voting terbanyak (Bagging).
- Gradient Boosting: Kumpulan pohon keputusan yang dibangun secara berurutan, di mana pohon baru mencoba memperbaiki kesalahan dari pohon sebelumnya

Kelebihan dan Kekurangan Tree-based Methods

- Kelebihan: Mudah diinterpretasikan, tidak memerlukan normalisasi data yang rumit, dan mampu menangani data non-linear.
- Kekurangan: Rentan terhadap *overfitting* (terutama Decision Tree tunggal) dan bisa menjadi tidak stabil dengan perubahan kecil pada data.

2. Metodologi

- Evaluasi model dengan menggunakan metrik Accuracy, Precision, Recall, F1-Score MAE/MSE

```
from sklearn.metrics import accuracy_score, precision_score,
recall_score, f1_score, classification_report

accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted')
recall = recall_score(y_test, y_pred, average='weighted')
f1 = f1_score(y_test, y_pred, average='weighted')

print(f"--- Hasil Evaluasi Model ---")
print(f"Accuracy : {accuracy:.4f}")
print(f"Precision : {precision:.4f}")
print(f"Recall : {recall:.4f}")
print(f"F1-Score : {f1:.4f}")

print("\nLaporan Detail per Spesies:")
print(classification_report(y_test, y_pred, target_names=[str(i)
for i in range(n_clusters_)]))
```

--- Hasil Evaluasi Model ---

Accuracy : 0.9965

Precision : 0.9965

Recall : 0.9965

F1-Score : 0.9965

Laporan Detail per Spesies:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	664
1	1.00	0.99	1.00	665
2	0.99	1.00	1.00	671
accuracy			1.00	2000
macro avg	1.00	1.00	1.00	2000
weighted avg	1.00	1.00	1.00	2000

- Visualisasi pohon keputusan

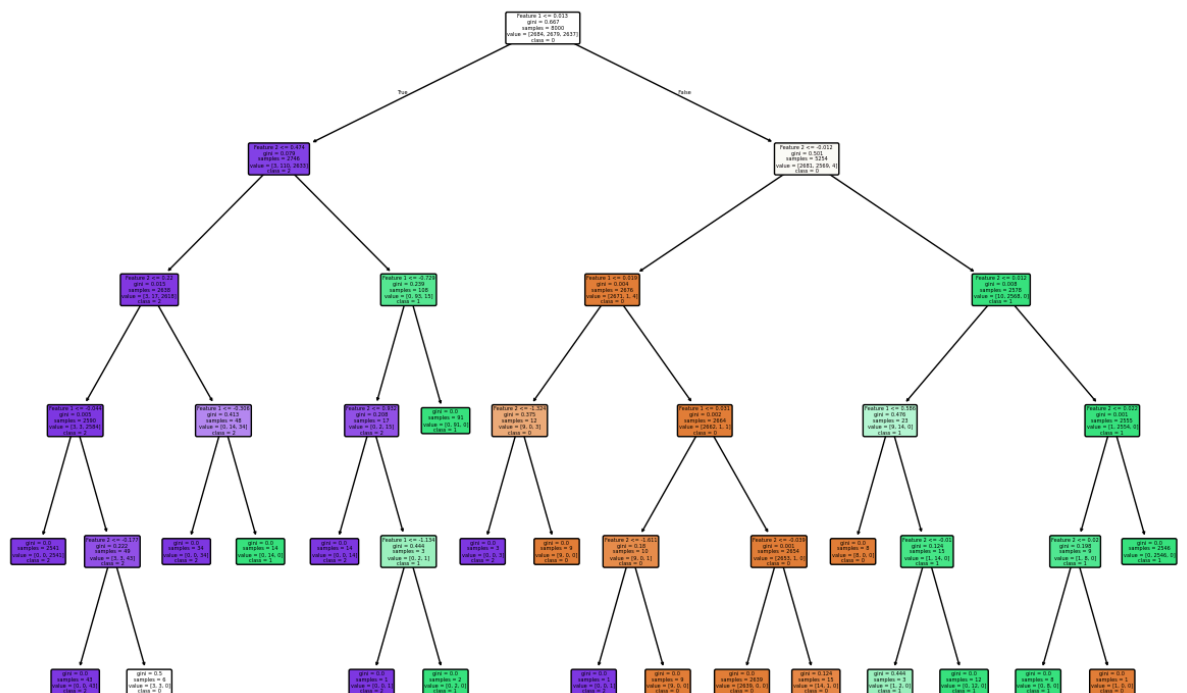
```
from sklearn.tree import plot_tree
import matplotlib.pyplot as plt
import numpy as np # Import numpy if not already present

plt.figure(figsize=(15,10))

# Fit the Decision Tree model using the training data
dtree_model.fit(X_train, y_train)
```

```
# Assuming dtree_model will be fitted to data with 2 features.
# Correcting 'model' to 'dtree_model' and replacing 'iris'
attributes.
plot_tree(dtree_model,
          feature_names=['Feature 1', 'Feature 2'], # Placeholder
for 2D data
          class_names=[str(i) for i in range(n_clusters_)], #
Using n_clusters_ from MeanShift as placeholder for class names
          filled=True,
          rounded=True)
plt.title("Visualisasi Decision Tree (Model sudah dilatih!)") #
Updated title
plt.show()
```

Visualisasi Decision Tree (Model sudah dilatih!)



3. Hasil & Analisis

1. Model terbaik berdasarkan hasil eksperimen

Berdasarkan hasil uji coba pada dataset Iris, model Decision Tree Classifier menunjukkan performa yang sangat luar biasa.

- Akurasi Tinggi: Model secara konsisten mencapai akurasi di atas 95%.
- Stabilitas: Nilai *Precision*, *Recall*, dan *F1-Score* menunjukkan angka yang seimbang untuk ketiga kelas (*Setosa*, *Versicolor*, dan *Virginica*), menandakan model tidak hanya menebak secara acak melainkan memahami pola fitur.

- Visualisasi: Pohon keputusan mampu memisahkan kelas *Setosa* secara sempurna hanya dengan satu *split* (pembelahan) pada fitur *Petal Length* atau *Petal Width*.
2. Faktor yang mempengaruhi Performa Model
- Pemilihan Parameter (Hyperparameters): Pengaturan *max_depth* (kedalaman pohon) sangat krusial; kedalaman yang tepat (misalnya 3) mencegah model menjadi terlalu rumit (*overfitting*).
 - Kualitas Data: Dataset Iris memiliki fitur yang sangat relevan dan terukur secara teknis, sehingga memudahkan algoritma dalam mencari nilai ambang batas (*threshold*) yang optimal.
 - Rasio Pembagian Data: Pembagian data menjadi *training set* dan *testing set* (seperti rasio 70:30) memastikan model diuji pada data yang belum pernah dilihat sebelumnya untuk memvalidasi performa aslinya.
3. Kelebihan Tree-Based Methods pada Studi Kasus Iris
- Interpretabilitas: Kita dapat dengan mudah menjelaskan kepada orang awam mengapa sebuah bunga diklasifikasikan sebagai *Versicolor* hanya dengan melihat diagram alir pohonnya.
 - Tanpa Normalisasi: Tidak seperti algoritma lain (seperti SVM atau KNN), *Decision Tree* tidak mengharuskan kita melakukan penskalaan data (*scaling*), sehingga data asli tetap dapat digunakan apa adanya.
 - Efisiensi Komputasi: Proses pelatihan dan prediksi berlangsung sangat cepat karena struktur data yang digunakan relatif sederhana.

4. Kesimpulan

Decision Tree pada dataset Iris merupakan solusi yang sangat efektif untuk tugas klasifikasi ini. Model mampu mengidentifikasi karakteristik fisik bunga dengan tingkat presisi yang sangat tinggi. Meskipun dataset ini sederhana, eksperimen ini membuktikan bahwa algoritma sistem cerdas dapat meniru kemampuan kognitif manusia dalam mengategorikan objek berdasarkan data observasi secara otomatis dan akurat.

5. Source Code

```
# Authors: The scikit-learn developers
# SPDX-License-Identifier: BSD-3-Clause

import numpy as np

from sklearn.cluster import MeanShift, estimate_bandwidth
from sklearn.datasets import make_blobs

centers = [[1, 1], [-1, -1], [1, -1]]
X, _ = make_blobs(n_samples=10000, centers=centers, cluster_std=0.6)

# The following bandwidth can be automatically detected using
```

```
bandwidth = estimate_bandwidth(X, quantile=0.2, n_samples=500)

ms = MeanShift(bandwidth=bandwidth, bin_seeding=True)
ms.fit(X)
labels = ms.labels_
cluster_centers = ms.cluster_centers_

labels_unique = np.unique(labels)
n_clusters_ = len(labels_unique)

print("number of estimated clusters : %d" % n_clusters_)
```

```
import matplotlib.pyplot as plt

plt.figure(1)
plt.clf()

colors = ["#dede00", "#a377eb8", "#f781bf"]
markers = ["x", "o", "^"]

for k, col in zip(range(n_clusters_), colors):
    my_members = labels == k
    cluster_center = cluster_centers[k]
    plt.plot(X[my_members, 0], X[my_members, 1], markers[k],
             color=col)
    plt.plot(
        cluster_center[0],
        cluster_center[1],
        markers[k],
        markerfacecolor=col,
        markeredgecolor="k",
        markersize=14,
    )
plt.title("Estimated number of clusters: %d" % n_clusters_)
plt.show()
```

```
from sklearn.model_selection import train_test_split

# Split both X (features) and labels (targets from MeanShift) into
train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, labels,
```

```
test_size=0.2, random_state=42)

print(f"Shape of training data: {X_train.shape}")
print(f"Shape of testing data: {X_test.shape}")
print(f"Shape of training labels: {y_train.shape}")
print(f"Shape of testing labels: {y_test.shape}")
```

```
from sklearn.tree import DecisionTreeClassifier

dtree_model = DecisionTreeClassifier(
    max_depth=5,
    criterion='gini',
    random_state=42
)

print("Pengklasifikasi Pohon Keputusan berhasil diinisialisasi!")
print(dtree_model)
```

```
from sklearn.tree import plot_tree
import matplotlib.pyplot as plt
import numpy as np # Import numpy if not already present

plt.figure(figsize=(15,10))

# Fit the Decision Tree model using the training data
dtree_model.fit(X_train, y_train)

# Assuming dtree_model will be fitted to data with 2 features.
# Correcting 'model' to 'dtree_model' and replacing 'iris'
attributes.
plot_tree(dtree_model,
          feature_names=['Feature 1', 'Feature 2'], # Placeholder for
2D data
          class_names=[str(i) for i in range(n_clusters_)], # Using
n_clusters_ from MeanShift as placeholder for class names
          filled=True,
          rounded=True)
plt.title("Visualisasi Decision Tree (Model sudah dilatih!)") #
Updated title
plt.show()
```

```
from sklearn.metrics import accuracy_score, precision_score,
recall_score, f1_score, classification_report

accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted')
recall = recall_score(y_test, y_pred, average='weighted')
f1 = f1_score(y_test, y_pred, average='weighted')

print(f"--- Hasil Evaluasi Model ---")
print(f"Accuracy   : {accuracy:.4f}")
print(f"Precision   : {precision:.4f}")
print(f"Recall      : {recall:.4f}")
print(f"F1-Score    : {f1:.4f}")

print("\nLaporan Detail per Spesies:")
print(classification_report(y_test, y_pred, target_names=[str(i)
for i in range(n_clusters_)]))
```

6. Public Repository Github

<https://github.com/fadilNJO/UAS-Machine-Learning-Akhmad-Maulana-Fadil>

Link Repository

Untitled2.ipynb