# Aplikasi Algoritma CART dalam Klasifikasi Jamur Berdasarkan Kelayakan Makan

Hanif Muhammad Zhafran - 13521157[1]
*Program Studi Teknik Informatika*
*Sekolah Teknik Elektro dan Informatika*
*Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia*
*[1]13521157@std.stei.itb.ac.id*

***Abstract*—Algoritma CART (Classification and Regression Tree) merupakan salah satu algoritma pembelajaran mesin berbasis pohon yang banyak dipakai untuk menyelesaikan permasalahan klasifikasi dan regresi. Penggunaannya sudah meluas ke dalam berbagai bidang. Pada penelitian ini, akan ditinjau penerapan algoritma CART dalam bidang pangan dan mikologi, yaitu klasifikasi jamur berdasarkan kelayakan makan. Algoritma CART yang diterapkan pada penelitian ini menggunakan Gini impurity sebagai metrik ketika splitting pohon keputusan. Performa model pada kasus ini dapat mencapai nilai F1-score sebesar 100%, dengan parameter max_leaf_nodes=22. Penyebab akurasi yang tinggi dimungkinkan oleh bersihnya data, data yang seimbang, dan kardinalitas rendah. Untuk kedepannya, penulis menyarankan untuk mengambil pendekatan dari data bentuk lain seperti data gambar dalam melakukan klasifikasi.**

***Keywords*—Pohon keputusan, Algoritma CART, Jamur, Klasifikasi**

## I. Pendahuluan

Seluk-beluk kehidupan manusia sudah berpindah ke era digital yang hampir semua permasalahan bergantung kepada data. Dari hal tersebut lah berkembang berbagai macam algoritma untuk mengotomasi berbagai macam penyelesaian permasalahan. Algoritma dibawah cakupan Inteligensi Buatan dan Pembelajaran Mesin merupakan salah satu yang sangat berkembang pesat pada zaman sekarang.

Salah satu algoritma yang sederhana namun sangat *powerful* adalah algoritma CART (*Classification and Regression Tree*). Algoritma CART merupakan salah satu algoritma pembelajaran mesin berbasis pohon. Dimulai dari pengembangannya yang sangat sederhana, banyak orang telah berkontribusi dalam algoritma ini sehingga algoritma ini dapat menjadi salah satu algoritma memiliki performa yang cukup baik.

Algoritma CART sudah pernah diaplikasikan ke dalam banyak bidang yang lain, seperti kesehatan dan ekonomi. Pada penelitian ini, penulis akan meninjau penerapan algoritma CART dalam bidang pangan dan mikologi, yaitu klasifikasi jamur berdasarkan kelayakan makan. Jamur merupakan salah satu bahan pangan yang umum dikonsumsi oleh manusia. Jamur yang tersedia kepada umum merupakan jamur yang telah dipastikan layak untuk dimakan (tidak beracun). Proses tersebut pasti diawali dengan adanya jamur tersebut di alam liar, kemudian baru diindustrialisasikan. Penerapan algoritma CART pada klasifikasi jamur berdasarkan kelayakan makan diharapkan dapat membantu dan mempercepat proses awal dalam mengklasifikasi jamur.

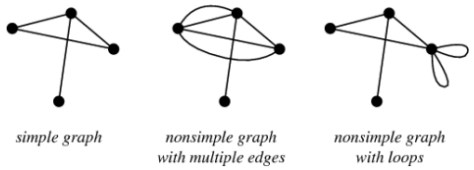
## II. Dasar Teori

### A. Graf

Graf pada matematika diskrit merupakan bentuk representasi hubungan antara objek-objek diskrit. Secara formal, graf G dinyatakan dengan (V, E), dengan V merupakan himpunan tidak-kosong dari simpul-simpul (*vertices*) ($\{v_1, v_2, \dots, v_n\}$) dan E merupakan himpunan sisi (*edges*) yang menghubungkan sepasang simpul ($\{e_1, e_2, \dots, e_n\}$). Setiap simpul dapat dinyatakan sebagai suatu angka (Contoh: $v_1 = 1$). Setiap sisi dapat dinyatakan dengan pasangan simpul yang dihubungkan oleh sisi tersebut (Contoh: $e_1 = (v_1, v_2)$).

Berdasarkan ada atau tidaknya suatu gelang atau sisi ganda pada graf, graf dapat diklasifikasikan seperti berikut:

1. Graf sederhana (*simple graph*)
   Graf yang tidak mengandung sisi ganda maupun gelang.
2. Graf tak-sederhana (*nonsimple graph*)
   Graf yang mengandung sisi ganda ataupun gelang. Graf tak-sederhana dapat dibagi lagi menjadi seperti berikut:
   2.1. Graf ganda (*multi-graph*)
   Graf yang mengandung sisi ganda tetapi tidak memiliki sisi gelang.
   2.2. Graf semu (*pseudo-graph*)
   Graf yang mengandung sisi gelang.

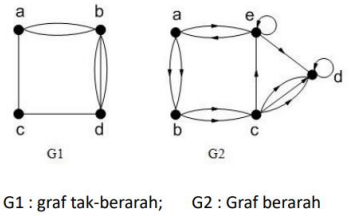



Gambar 2.1: Jenis graf berdasarkan keberadaan sisi gelang dan sisi ganda.
Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir

Berdasarkan orientasi arah pada sisi, graf dapat diklasifikasikan seperti berikut:

1. Graf tak-berarah (*undirected graph*)
   Graf yang sisinya tidak mempunyai orentasi arah.
2. Graf berarah (*directed graph*)

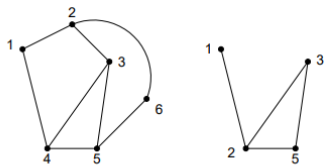
Graf yang setiap sisinya terdapat orientasi arah.




Gambar 2.2: Jenis graf berdasarkan orientasi
Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir

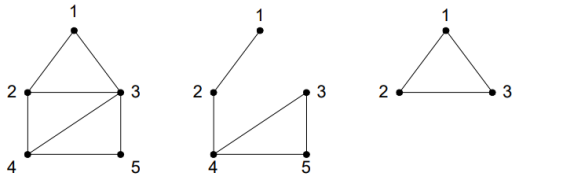
Dalam konsep graf, terdapat sejumlah terminologi yang sering dipakai untuk membantu dalam menjelaskan suatu graf. Beberapa diantaranya ialah:

1. Ketetanggaan (*adjacent*)
   Dua simpul bertetangga satu sama lain ketika keduanya terhubung langsung oleh suatu sisi.
2. Berisisan (*incidency*)
   Suatu sisi $e$ bersisian dengan simpul $v_1$ dan $v_2$ ketika $e = (v_1, v_2)$.
3. Simpul terpencil (*isolated vertex*)
   Suatu simpul dikatakan sebagai simpul terpencil ketika tidak ada sisi yang bersisian dengan simpul itu sendiri.
4. Graf kosong (*null graph/empty graph*)
   Graf kosong merupakan graf yang tidak memiliki sisi ($E = \emptyset$).
5. Derajat (*degree*)
   Derajat suatu simpul diartikan sebagai jumlah sisi yang bersisian dengan simpul itu sendiri.
6. Lintasan (*path*)
   Sekumpulan sisi pada graf yang menghubungkan suatu simpul awal $v_0$ dengan suatu simpul akhir $v_n$ sedemikian sehingga $e_1 = (v_0, v_1), e_2 = (v_1, v_2), \dots, e_n = (v_{n-1}, v_n)$.
7. Sirkuit
   Sirkuit adalah lintasan yang memiliki simpul awal dan simpul akhir yang sama.
8. Keterhubungan (*connected*)
   Sepasang simpul dikatakan terhubung ketika ada lintasan yang menghubungan kedua simpul tersebut. Suatu graf disebut sebagai graf terhubung ketika setiap pasang simpul pada graf merupakan pasangan simpul yang terhubung.
9. Upagraf (*subgraph*)
   Upagraf dari suatu graf $G = (V, E)$ adalah $G' = (V', E')$, dengan $V' \subseteq V$ dan $E' \subseteq E$.




Gambar 2.3: Contoh upagraf dari graf G1
Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir

10. Upagraf merentang (*spanning subgraph*)
    Suatu upagraf disebut sebagai upagraf merentang ketika seluruh simpul dari graf awal termasuk kedalam simpul upagraf.



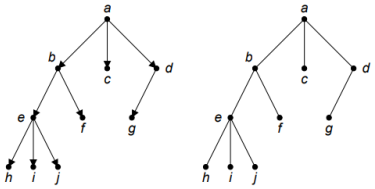

Gambar 2.4: Contoh upagraf merentang
Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir

## *B. Pohon*

Pohon didefinisikan sebagai graf terhubung yang tak-berarah dan tidak mengandung sirkuit. Secara formal, pohon didefinisikan sebagai berikut: Misalkan $G = (V, E)$ adalah graf tak-berarah sederhana dan jumlah simpulnya adalah n. Maka, seluruh pernyataan di bawah ini adalah benar:

1. G juga merupakan sebuah pohon.
2. Setiap pasangan simpul dalam G terhubung dengan lintasan tunggal.
3. G merupakan graf terhubung.
4. G tidak mengandung sirkuit.
5. Penambahan satu sisi pada suatu pohon akan membentuk satu sirkuit.
6. Semua sisinya merupakan suatu jembatan
7. Memiliki sisi sebanyak $m = n - 1$.

Salah satu representasi pohon secara struktural adalah pohon berakar (*rooted tree*). Pohon berakar merupakan suatu pohon yang salah satu simpulnya dijadikan sebagai akar kemudian seluruh sisinya diarahkan keluar dari simpul akar.




Gambar 2.5: Ilustrasi pohon berakar
Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir

Berikut merupakan beberapa terminologi pada pohon berakar:

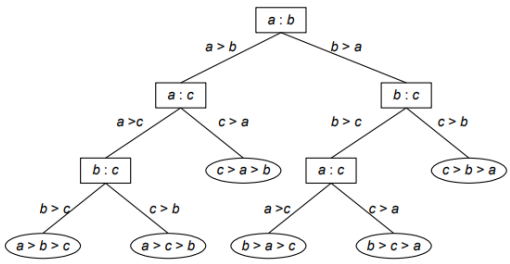
1. Anak (*child*) dan Orangtua (*parent*)
   Pada gambar 2.5, h, i, dan j merupakan anak-anak dari e, dan e adalah orangtua dari h, i, dan j.
2. Lintasan (*path*)
   Sama dengan definisi lintasan pada graf.
3. Saudara kandung (*sibling*)
   Sepasang simpul memiliki hubungan saudara kandung ketika kedua simpul tersebut memiliki orangtua yang sama. Pada gambar 2.5, b dan d merupakan saudara kandung dari c.
4. Upapohon (*subtree*)
   Kumpulan simpul dan sisi yang merupakan suatu pohon yang seluruh simpul dan sisinya merupakan subset dari pohon awal.
5. Derajat (*degree*)

Derajat sebuah simpul adalah jumlah anak yang dimiliki oleh simpul tersebut.

6. Daun (*leaf*)
   Simpul daun merupakan simpul yang tidak mempunyai anak (derajatnya 0).
7. Simpul dalam (*internal nodes*)
   Simpul yang mempunyai anak dan bukan merupakan simpul yang dijadikan sebagai akar disebut sebagai simpul dalam.
8. Aras (*level*) atau Tingkat
   Panjang lintasan terpendek yang ditempuh dari akar sampai ke simpul tertentu.
9. Tinggi (*height*) atau Kedalaman (*depth*)
   Tinggi atau Kedalaman dari suatu pohon adalah tingkat maksimum dari pohon tersebut.

## *C. Pohon Keputusan*

Pohon keputusan merupakan suatu pohon yang berfungsi untuk mengeluarkan suatu keputusan berdasarkan prekondisi yang ada. Setiap simpul pada suatu keputusan melambangkan perbandingan yang akan dilakukan. Kemudian setiap sisi dari orangtua ke tiap-tiap anaknya memiliki prekondisinya masing-masing untuk dilewati. Keputusan yang final akan didapatkan apabila proses telah mencapai simpul daun.




Gambar 2.6: Ilustrasi pohon keputusan yang mengurutkan 4 bilangan
Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir

## *D. Pembelajaran Mesin (Machine Learning)*

Pembelajaran Mesin merupakan suatu bidang yang membahas tentang bagaimana suatu mesin (computer) dapat meniru cara kerja manusia dalam berpikir dan menentukan keputusan. Pembelajaran Mesin termasuk ke dalam salah satu sub-bidang dari Intelegensi Buatan (*Artificial Intelligence*).

Terdapat tiga tujuan umum dari suatu sistem pembelajaran mesin, yaitu deskriptif (menggunakan data untuk menjelaskan suatu hal), prediktif (menggunakan data untuk memprediksi suatu hal di masa depan), dan preskriptif (menggunakan data untuk menghasilkan saran terhadap suatu tindakan).

Tahap pertama dari suatu sistem pembelajaran mesin adalah tahap pengumpulan data. Secara garis besar, ada dua macam data yang dapat digunakan dalam pembelajaran mesin, yaitu:

1. Data terstruktur (*Structured Data*)
   Data yang sudah dikelompokkan, terorgansir, mengikuti format yang mudah dipahami oleh suatu sistem pembelajaran mesin, dan memiliki atribut masing-masing. Biasanya berbentuk tabular.
2. Data tak-terstruktur (*Unstructured Data*)
   Data yang tidak bisa diproses langsung oleh suatu model atau belum mengikuti format yang cocok untuk diproses suatu model. Contohnya adalah data gambar, teks, dan suara.

Data yang terkumpul akan dimasukkan ke dalam suatu model pembelajaran mesin, yang nantinya model akan ‘belajar’ dari data tersebut. Setelah proses pembelajaran, model akan menjalankan fungsi sesuai dengan tugasnya. Manusia sebagai pembuat sistem akan mengoreksi model atau menambahkan data untuk meningkatkan performa suatu model. Seluruh proses tersebut akan terus berjalan secara siklus untuk mencapai suatu model pembelajaran mesin yang optimal.

Terdapat tiga jenis utama dari algoritma pembelajaran mesin, yaitu:

1. *Supervised machine learning*
   Model ini memakai data yang sudah diberi label dalam proses pembelajarannya. Misalkan suatu data berisi tentang ciri-ciri dari suatu spesies jamur, kemudian jamur tersebut dilabeli beracun atau tidak beracun. Model akan belajar dari data tersebut untuk memprediksi apakah suatu sel tumor berbahaya atau tidak berdasarkan data yang telah diberikan.
2. *Unsupervised machine learning*
   Pada *unsupervised machine learning*, model akan mencari pola pada data yang tidak diberi label. Hal ini berguna untuk menemukan *trend* atau pola tertentu dalam suatu data. Salah satu contoh contoh permasalahannya adalah menemukan *trend* perilaku pelanggan dalam suatu data penjualan online.
3. *Reinforcement machine learning*
   Metode ini merupakan metode berbasis *trial and error*. Model akan diberitahu kapan mereka membuat keputusan yang tepat dan kapan mereka membuat keputusan yang salah. Contoh penerapannya adalah dalam *video game* dan kendaraan otonom.

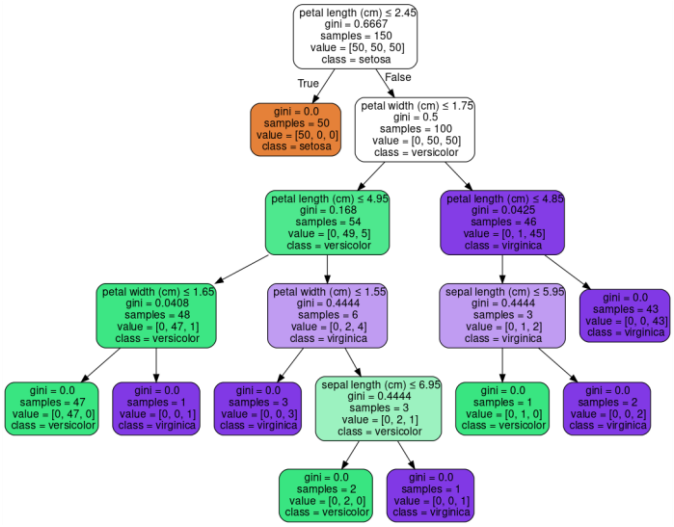
## *E. Algoritma CART (Classification and Regression Tree) dalam Pembelajaran Mesin*

Algoritma CART (*Classification and Regression Tree*) merupakan salah satu algoritma pembelajaran mesin berbasis pohon keputusan. Sesuai dengan namanya, algoritma ini digunakan untuk menyelesaikan masalah klasifikasi dan regresi. Algoritma tersebut akan membangun suatu pohon keputusan berdasarkan data yang diberikan.

Pohon keputusan akan dibangun dengan cara melakukan pembelahan (*split*) secara bertahap. Yang dimaksud dengan proses *splitting* adalah mencari suatu kondisi di simpul tertentu berdasarakan salah satu atribut pada data yang telah diberikan. *Splitting* dilakukan dengan cara meminimalisasi *impurity* dalam suatu simpul. Salah satu metrik pengukuran *impurity* adalah *Gini impurity*, yang didefinisikan sebagai berikut:

$$Gini\ Impurity = 1 - \sum_{i=1}^{n} p_i^2$$

dengan $p_i$ adalah fraksi dari kelas $i$ terhadap seluruh sampel di simpul tersebut. Proses *splitting* akan terus dilakukan sampai tidak didapatkan nilai minimum *Gini impurity* yang lebih kecil dibandingkan simpul orangtuanya.

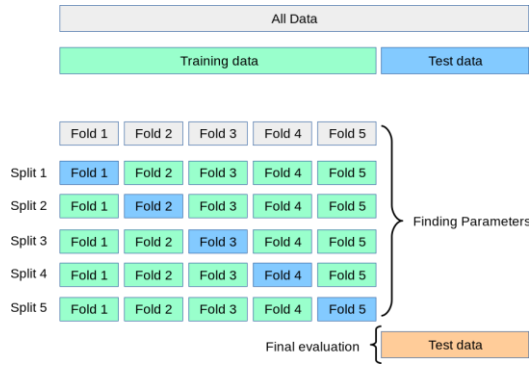



Gambar 2.7: Contoh ilustrasi pohon keputusan hasil algoritma CART
Sumber: https://scikit-learn.org/stable/modules/tree.html

### *F. K-fold Cross-Validation*

*Cross-Validation* merupakan salah satu metode dalam evaluasi model pembelajaran mesin untuk mencegah terjadinya *overfitting*. *Overfitting* merupakan suatu kondisi ketika suatu model memiliki performa yang bagus terhadap data *train* tetapi performanya buruk ketika memprediksi masukan data baru. Hal tersebut bisa saja terjadi ketika model terlalu menyesuaikan dengan data pertama yang diterimanya sehingga tidak bagus dalam menghadapi data yang baru.

*K-fold Cross-Validation* akan memecah data *train* menjadi dua bagian, satu untuk *training* model dan yang lainnya untuk validasi. Kemudian proses ini akan diiterasi sebanyak *K* kali dengan komposisi pemecahan data berbeda dengan iterasi sebelumnya, sehingga pada akhirnya seluruh data pernah termasuk ke bagian *training* maupun validasi. Proses ini akan membuat model mempelajari data secara lebih menyeluruh. Hasil evaluasi dari *Cross-Validation* akan menjadi patokan terhadap pembuat model dalam menentukan parameter dari model tersebut.




Gambar 2.8: Skema *k-fold cross-validation* dengan k = 5
Sumber: https://scikit-learn.org/stable/modules/cross_validation.html

### *G. Metrik Evaluasi F1-Score*

*F1-score* merupakan salah satu metrik evaluasi model pembelajaran mesin dalam permasalahan klasifikasi. *F1-score* memiliki persamaan sebagai berikut:

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

*Precision* dan *recall* didefinisikan sebagai berikut:

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

TP (*True Positive*) adalah jumlah data yang memiliki kelas positif dan hasil prediksi model juga menunjukkan kelas positif. FP (*False Positive*) adalah jumlah data yang memiliki kelas positif namun hasil prediksi model menunjukkan kelas negatif. TN (*True Negative*) merupakan jumlah data yang memiliki kelas negatif dan hasil prediksi model juga menunjukkan kelas negatif. FN (*False Negative*) merupakan jumlah data yang memiliki kelas negatif namun hasil prediksi model menunjukkan kelas positif.

*F1-score* merupakan metrik klasifikasi yang berguna dalam kasus data yang tidak seimbang (*imbalanced*). Data disebut tidak seimbang ketika banyak suatu kelas dalam dataset jauh melampaui kelas lainnya. Metrik *F1-score* akan mempertimbangkan hal tersebut, sehingga akurasi dari model dapat diketahui secara lebih akurat.

## III. METODOLOGI

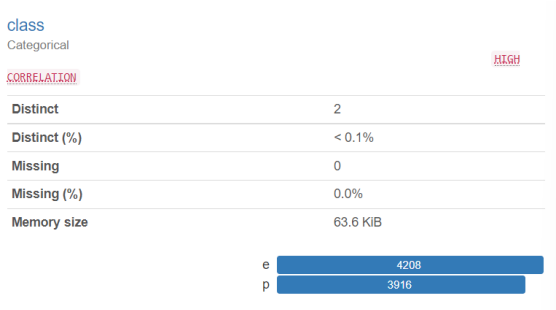
### *A. Kakas*

Pada penelitian ini, digunakan bahasa pemrograman Python, termasuk dengan beberapa *library* terkait, yaitu Pandas, NumPy, scikit-learn, dan pandas-profiling untuk eksplorasi data, pemrosesan data, dan pemodelan. Digunakan juga *library* matplotlib untuk visualisasi data.

### *B. Gambaran Dataset*

Dataset yang digunakan pada penelitian ini adalah data *Mushroom Classification* yang diambil dari platfrom Kaggle (Mushroom Classification | Kaggle). Data tersebut berisi tentang sampel jamur yang termasuk ke dalam 23 spesies *gilled mushroom* dari famili *Agaricus* dan *Lepiota*. Dataset terdiri dari 8124 baris dan 23 kolom. Dari 23 atribut data, 22 diantaranya memiliki tipe data kategorikal, 1 sisanya memliki tipe data boolean.

Atribut yang akan dijadikan target prediksi adalah *class*, yaitu atribut yang menunjukkan apakah jamur tersebut beracun atau aman untuk dimakan. Sebanyak 3916 data jamur diklasifikasikan sebagai jamur beracun dan 4208 data jamur diklasifikasikan sebagai jamur yang aman untuk dimakan (*edible*). Angka tersebut menunjukkan bahwa dataset cukup seimbang (*balanced*).




Gambar 3.1: Keterangan atribut data *class*
Sumber: Dokumen Penulis

```
Data columns (total 23 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   class                     8124 non-null   int64
 1   cap-shape                 8124 non-null   object
 2   cap-surface               8124 non-null   object
 3   cap-color                 8124 non-null   object
 4   bruises                   8124 non-null   object
 5   odor                      8124 non-null   object
 6   gill-attachment           8124 non-null   object
 7   gill-spacing              8124 non-null   object
 8   gill-size                 8124 non-null   object
 9   gill-color                8124 non-null   object
 10  stalk-shape               8124 non-null   object
 11  stalk-root                8124 non-null   object
 12  stalk-surface-above-ring  8124 non-null   object
 13  stalk-surface-below-ring  8124 non-null   object
 14  stalk-color-above-ring    8124 non-null   object
 15  stalk-color-below-ring    8124 non-null   object
 16  veil-type                 8124 non-null   object
 17  veil-color                8124 non-null   object
 18  ring-number               8124 non-null   object
 19  ring-type                 8124 non-null   object
 20  spore-print-color         8124 non-null   object
 21  population                8124 non-null   object
 22  habitat                   8124 non-null   object
dtypes: int64(1), object(22)
memory usage: 1.4+ MB
```

Gambar 3.2: Keterangan umum tiap atribut data
Sumber: Dokumen Penulis

Seluruh atribut data memiliki kardinalitas yang cukup rendah ( <= 10). Pada dataset juga tidak ditemukan adanya data kosong ataupun data duplikat. Terdapat satu atribut yang memiliki nilai yang konstan pada seluruh baris dalam dataset, yaitu *veil-type*.

### *C. Pra-pemrosesan Data*

Pemrosesan data yang dilakukan hanya 2 hal. Yang pertama adalah penghapusan atribut data *veil-type*, karena memiliki nilai yang konstan. Yang kedua adalah mengubah seluruh kolom kategorikal menjadi numerikal menggunakan metode *label encoding*. *Label encoding* akan memetakan setiap kategori dalam suatu atribut data menjadi angka yang dimulai dari 0 (Contoh: [‘apel’, ‘pisang’, ‘mangga’, ‘pisang’] → [0, 1, 2, 1]). *Label encoding* dilakukan dengan memanfaatkan kelas LabelEncoder pada *library* scikit-learn.

```python
def label_encoding(df_inp):
    df_out = df_inp.copy()
    label_encoder = LabelEncoder()
    for col in df_out.columns:
        if df_out[col].dtype == "object":
            label_encoder.fit(df_out[col])
            print(list(label_encoder.classes_))
            df_out[col] = label_encoder.transform(df_out[col])
            df_out = df_out.astype({col: "int64"})

    return df_out
```

Gambar 3.3: Source code fungsi label_encoding
Sumber: Dokumen Penulis

### *D. Pemodelan*

Model dibuat dengan memanfaatkan kelas DecisionTreeClassifier pada *library* scikit-learn. *Loss function* yang akan digunakan adalah *Gini impurity*. Parameter model yang akan di-*tuning* adalah *max_leaf_nodes*, yaitu simpul daun maksimal dari pohon keputusan tersebut.

### *E. Validasi*

Metrik evaluasi yang akan digunakan adalah *F1-score*. Hasil *F1-score* yang akan dijadikan sebagai acuan adalah *F1-score* dari hasil *K-fold cross-validation* dengan $K = 5$. Proses *training* akan dilakukan sebanyak 24 kali, dengan parameter *max_leaf_nodes* awal adalah 2. Tiap iterasi, parameter *max_leaf_nodes* akan ditambah 1. Hasil *F1-score* tiap iterasi akan disimpan di dalam array *y_loss*.

```python
def validate(model, X, y, cv, verbose=1):
    recalls = []
    precisions = []
    specificities = []
    f1s = []

    for i, (train_idx, val_idx) in enumerate(cv.split(X, y)):

        # Split for cross-validation
        X_train, X_val = X.iloc[train_idx], X.iloc[val_idx]
        y_train, y_val = y.iloc[train_idx], y.iloc[val_idx]

        # Training
        fitted = model.fit(X_train, y_train)
        y_pred = fitted.predict(X_val)
        tn, fp, fn, tp = confusion_matrix(y_pred, y_val).ravel()

        # Evaluating
        recall = tp / (tp + fn)
        precision = tp / (fp + tp)
        specificity = tn / (tn + fp)
        f1 = 2 * (precision * recall) / (precision + recall)

        recalls.append(recall)
        precisions.append(precision)
        specificities.append(specificity)
        f1s.append(f1)

        if (verbose == 2):
            print(f'FOLD {i+1} DONE!')

    # Average score
    if (verbose == 1):
        print(f'f1_mean = {(np.mean(f1s)*100):.6f}')
    elif (verbose == 2):
        print('\nMEAN')
        print('')
        print(f'recall = {(np.mean(recalls)*100):.6f}')
        print(f'precision = {(np.mean(precisions)*100):.6f}')
        print(f'specificity = {(np.mean(specificities)*100):.6f}')
        print(f"f1_value_per_fold = {f1s}")
        print(f'f1_mean = {(np.mean(f1s)*100):.6f}')
    return f1s
```

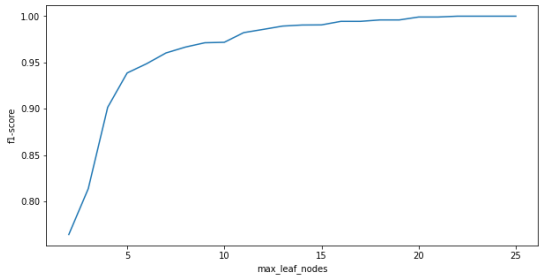
Gambar 3.4: Source code fungsi validate
Sumber: Dokumen Penulis

```
y_loss = []
cv = KFold(n_splits=5, shuffle=True, random_state=1)
for i in range(2, 26):
    model = DecisionTreeClassifier(
        random_state=2,
        max_leaf_nodes=i,
        criterion='gini',
        )
    loss = validate(model, X_labelenc, y_labelenc, cv, 1)
    y_loss.append(sum(loss)/5)
```

Gambar 3.5: Source code object model dan cross-validation
Sumber: Dokumen Penulis

## IV. HASIL DAN ANALISIS

Hasil *cross-validation* dari model pembelajaran mesin yang telah dibuat adalah sebagai berikut:



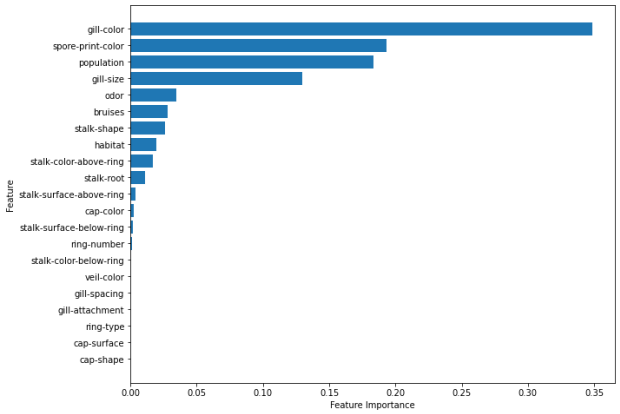

Gambar 3.6: Grafik F1-score terhadap parameter max_leaf_nodes
Sumber: Dokumen Penulis

| | max_leaf_nodes | f1-score (%) |
|---|---|---|
| 12 | 14 | 99.046359 |
| 13 | 15 | 99.059918 |
| 14 | 16 | 99.441847 |
| 15 | 17 | 99.441847 |
| 16 | 18 | 99.594848 |
| 17 | 19 | 99.594848 |
| 18 | 20 | 99.910998 |
| 19 | 21 | 99.910998 |
| 20 | 22 | 100.000000 |
| 21 | 23 | 100.000000 |
| 22 | 24 | 100.000000 |
| 23 | 25 | 100.000000 |

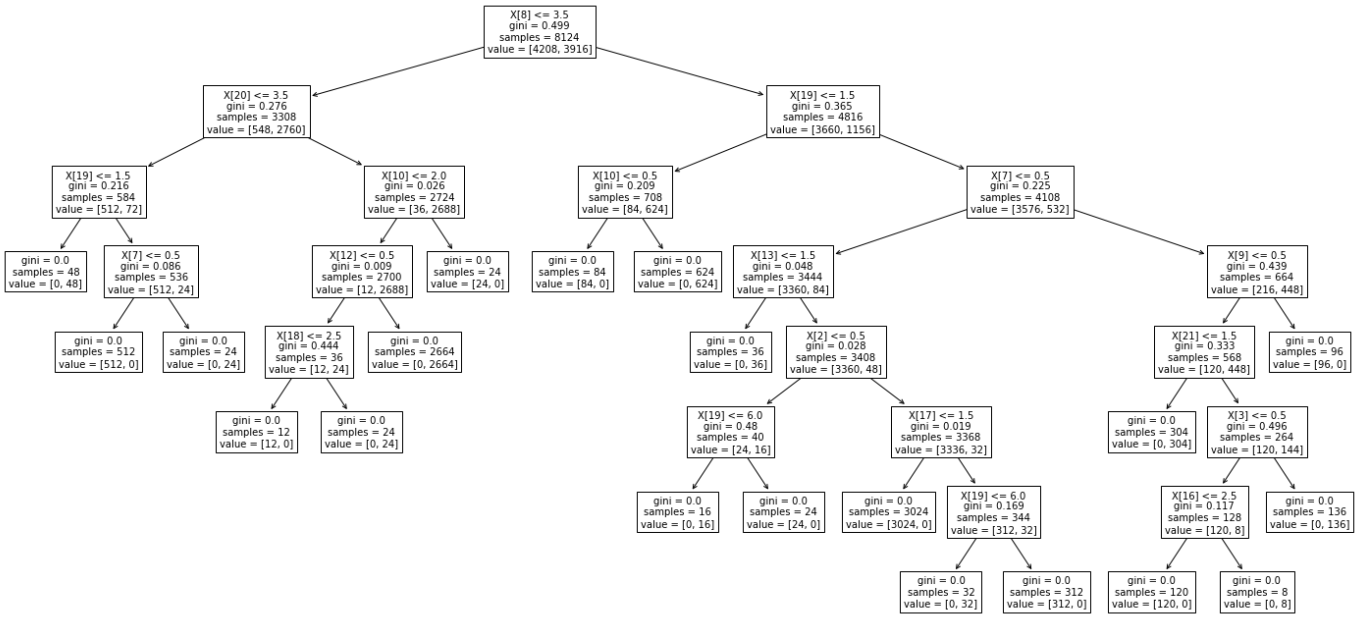
Gambar 3.7: Tabel parameter max_leaf_nodes dari 14-25 beserta f1-scorenya
Sumber: Dokumen Penulis

Model berhasil mencapai *F1-score* sebesar 100% ketika parameter *max_leaf_nodes* menyentuh angka 22. Hal tersebut menunjukkan bahwa model yang dibuat berhasil mengklasifikasikan seluruh data validasi saat proses *cross-validation* secara 100% benar. Beberapa kemungkinan penyebab akurasi yang sangat tinggi adalah kualitas data yang bagus, data yang seimbang (*balanced*), tidak ada *missing data*, dan kardinalitas yang rendah. Beberapa hal tersebut membuat model mudah menangkap pola pada dataset.

Setelah itu, dilakukan juga pengecekan *feature importance* dari masing-masing atribut data. Tiga atribut data yang paling berpengaruh terhadap penyusunan pohon keputusan adalah *gill-color* (warna serabut di bawah tudung jamur), *spore-print-color* (warna spora jamur), dan *population*.




Gambar 3.8: Plot batang horizontal dari *feature importance* setiap atribut data
Sumber: Dokumen Penulis




Gambar 3.9: Ilustrasi pohon keputusan yang dihasilkan
Sumber: Dokumen Penulis

## V. KESIMPULAN DAN SARAN

Model DecisionTreeClassifier yang menerapkan algoritma CART berhasil mencapai nilai *F1-score* sebesar 100% dengan parameter *max_leaf_nodes* sama dengan 22.

Untuk penelitian kedepannya, dapat dilakukan klasifikasi jamur berdasarkan kelayakan makan dengan pendekatan *unstructured data*, seperti menggunakan *Computer Vision* untuk klasifikasi berdasarkan data gambar. Hal tersebut dapat membuat proses klasifikasi jauh lebih cepat jika dibandingkan dengan penelitian ini, karena pada data yang dipakai saat ini, manusia masih harus mendata tiap atribut data satu per satu sehingga membutuhkan waktu yang lebih lama.

## VI. UCAPAN TERIMA KASIH

Puji syukur kepada Allah SWT yang telah melimpahkan rahmat dan karunia-Nya kepada penulis sehingga penulis masih diberi kesempatan untuk menyelesaikan makalah ini secara tepat waktu. Penulis juga ingin menyampaikan rasa terima kasih kepada kedua orang tua, kakak, dan juga teman-teman yang senantiasa mendukung saya dalam pengerjaan makalah ini. Saya juga ingin menyampaikan terima kasih kepada tim dosen pengampu mata kuliah IF2120 Matematika Diskrit, terkhusus Bu Fariska Zakhralativa Ruskanda sebagai pengampu kelas K2 yang telah melimpahkan ilmunya kepada penulis selama semester 3 (tiga) Tahun Ajaran 2022/2023. Penulis memohon maaf jika ada kesalahan dalam penulisan makalah ini. Penulis berharap makalah ini dapat memberikan manfaat kepada masyarakat.

## REFERENSI

[1] Brown, S. (2021, April 21). Machine learning, explained. Retrieved from MIT Management Sloan School: https://mitsloan.mit.edu/ideas-made-to-matter/machine-learning-explained. Diakses pada 12 Desember 2022, 17.22 WIB.

[2] Dobilas, S. (2021, January 31). CART: Classification and Regression Trees for Clean but Powerful Models. Retrieved from Towards Data Science: https://towardsdatascience.com/cart-classification-and-regression-trees-for-clean-but-powerful-models-cc89e60b7a85. Diakses pada 11 Desember 2022, 23.38 WIB

[3] Jayaswal, V. (2020, September 14). Performance Metrics: Confusion matrix, Precision, Recall, and F1 Score. Retrieved from Towards Data Science: https://towardsdatascience.com/performance-metrics-confusion-matrix-precision-recall-and-f1-score-a8fe076a2262J. Diakses pada 11 Desember 23.13 WIB

[4] https://informatika.stei.itb.ac.id/~rinaldi.munir/. Diakses pada 12 Desember 15.45 WIB; 12 Desember 16.35 WIB; 12 Desember 16.51 WIB.

## PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 12 Desember 2022

Hanif Muhammad Zhafran | 13521157