BAB 4
HASIL PENELITIAN

4.1 Ringkasan Hasil

Pada penelitian ini, telah dikembangkan sistem pendeteksi situs *phising* berbasis URL menggunakan algoritma *Decision Tree* (CART). Sistem ini dirancang untuk mendeteksi dan mengklasifikasikan URL ke dalam dua kategori utama, yaitu *phising* dan *legitimate*, berdasarkan karakteristik struktural yang dimiliki oleh masing-masing URL. Proses perancangan sistem dilakukan secara menyeluruh, dimulai dari pengumpulan data, *preprocessing* data, *ekstraksi* fitur, pelatihan model, hingga evaluasi performa model. Semua tahapan tersebut dilakukan menggunakan platform *Google Colab* sebagai lingkungan pemrograman berbasis *cloud*, dengan bantuan berbagai pustaka *Python* seperti *Pandas*, *Matplotlib*, *Scikit-learn*, dan lainnya.

Sistem ini tidak hanya menekankan pada keakuratan prediksi, tetapi juga pada kemampuan generalisasi terhadap data baru yang belum pernah dikenali sebelumnya. Oleh karena itu, pendekatan berbasis pembelajaran mesin menjadi pilihan tepat karena mampu menangkap pola-pola kompleks dari data URL yang ada. Dengan memanfaatkan algoritma CART yang bersifat interpretatif, sistem yang dibangun juga dapat menghasilkan visualisasi dalam bentuk pohon keputusan yang memberikan transparansi terhadap proses klasifikasi.

Perlu diketahui bahwa penelitian ini hanya melakukan klasifikasi URL berdasarkan struktur dan tidak mengidentifikasi secara spesifik jenis serangan seperti *email, phising, smishing, atau vishing.* Hal ini disebabka karena dataset yang diperoleh dari *PhisTank* dan *Tranco* tidak memuat informasi mengenai tipe serangan, melainkan memberikan label *phising* atau *legitimate.* Oleh karena itu, penelitian ini secara fungsional digunakan untuk mendeteksi serangan berbasis *web phising*, yang merupakan jenis *phising* umum dan relevan dengan struktur URL. Hasil pengujian ini menjadi tolak ukur dalam menilai keberhasilan pendekatam yang digunakan, sekaligus menjadi dasar untuk mengembangkan sistem keamanan berbasis URL di masa mendatang.

4.2 Pengumpulan data

Data yang digunakan merupakan data sekunder yang terdiri dari dua jenis yaitu, URL *phising* yang diperoleh dari *PhishTank* dan URL *legitimate* yang diperoleh dari sumber terpercaya seperti daftar URL di *Tranco*. Data dikumpulkan secara *crawling* dan disimpan dalam format CSV. Dataset *phising* diperoleh dari hasil *crawling* pada platform *PhishTank* untuk mendapatkan dataset melalui colab dalam bahasa pemrograman *Python* dapat menggunakan kode berikut:

```
print
phishtank_URL = "http://data.phishtank.com/data/online-valid.csv"
phising_data = []
try:
    response = requests.get(phishtank_URL, timeout=30)
    response.raise_for_status()
    decoded_content = response.content.decode('utf-8')
    lines = decoded_content.strip().split('\n')
    reader = csv.DictReader(lines)
    for i, row in enumerate(reader):
        if i >= 54000:
            break
        phising_data.append({
            "URL": row.get("URL"),
            "label": 1  # phising
        })
    print(f"✅ {len(phising_data)}")
```

Jumlah data yang didapat sebanyak 54.647 *URL*. Dataset *legitimate* diperoleh dari hasil *crawling* top domain dari *Tranco* untuk mendapatkan dataset melalui colab dapat menggunakan kode berikut:

```
TRANCO_URL = "https://tranco-list.eu/top-1m.csv.zip"
NUM_URLS_TO_GET = 54000
response = requests.get(TRANCO_URL)
content = io.BytesIO(response.content)
try:df_tranco = pd.read_csv(content, compression='zip',
header=None, names=['rank', 'domain'])
except:content.seek(0)df_tranco=pd.read_csv(content,header=None,n
ames=['rank’,'domain'])
current_timestamp = datetime.now().strftime('%Y-%m-%d %H:%M:%S')
selected_domains = df_tranco['domain'].head(NUM_URLS_TO_GET)
df_legitimate = pd.DataFrame({
    'URL': ['https://' + domain for domain in selected_domains],
    'label': 1,
```

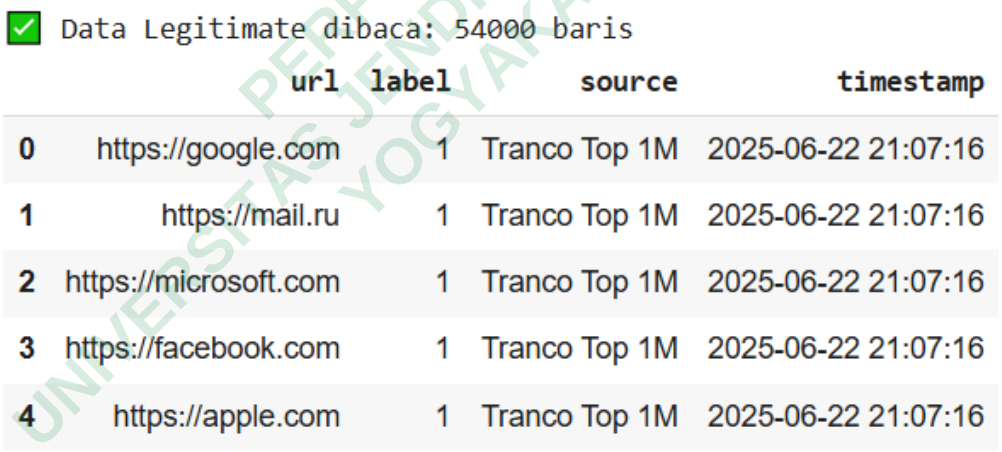
```
    'source': 'Tranco Top 1M',
    'timestamp': current_timestamp})
output_file = "legitimate_URLs_tranco.csv"
df_legitimate.to_csv(output_file, index=False)
files.download(output_file)
```

Jumlah data sebanyak 54.000 URL. Setelah data diperoleh kemudian digabungkan untuk pemrosesan selanjutnya dengan total data yang digabungkan yaitu 108.647 URL, dari data yang diperoleh dapat ditampilkan beberapa sampel data awal pada gambar 4.1 dan 4.2 berikut.

Data Phishing dibaca: 54647 baris

	phish_id	url	phish_detail_url	submission_time	verified	verification_time	online	target	label
0	9117813	https://programme-depot.com/	http://www.phishtank.com/phish_detail.php?phis...	2025-06-03T15:07:08+00:00	yes	2025-06-03T15:12:46+00:00	yes	Other	0
1	9117812	https://programme-depot.com/as.php	http://www.phishtank.com/phish_detail.php?phis...	2025-06-03T15:06:58+00:00	yes	2025-06-03T15:12:46+00:00	yes	Other	0
2	9117803	http://allegro.pl-oferta23451.top	http://www.phishtank.com/phish_detail.php?phis...	2025-06-03T14:58:05+00:00	yes	2025-06-03T15:03:23+00:00	yes	Allegro	0
3	9117795	http://allegrolokalnie.pl-oferta23451.top	http://www.phishtank.com/phish_detail.php?phis...	2025-06-03T14:44:22+00:00	yes	2025-06-03T14:51:44+00:00	yes	Allegro	0
4	9117793	https://sites.google.com/view/verifiedcodecod/...	http://www.phishtank.com/phish_detail.php?phis...	2025-06-03T14:35:31+00:00	yes	2025-06-03T14:41:50+00:00	yes	Other	0

Gambar 4.1 Sampel Dataset *Phising*



Data Legitimate dibaca: 54000 baris

	url	label	source	timestamp
0	https://google.com	1	Tranco Top 1M	2025-06-22 21:07:16
1	https://mail.ru	1	Tranco Top 1M	2025-06-22 21:07:16
2	https://microsoft.com	1	Tranco Top 1M	2025-06-22 21:07:16
3	https://facebook.com	1	Tranco Top 1M	2025-06-22 21:07:16
4	https://apple.com	1	Tranco Top 1M	2025-06-22 21:07:16

Gambar 4.2 Sampel Dataset *Legitimate*

4.3 *Pre-Processing* Data

Pada tahap ini, dilakukan pembersihan awal terhadap data *phising* dan *legitimate* yang diperoleh, Dataset *phising* awal yang dikumpulkan terdiri dari 54.647 baris data. Namun, setelah dilakukan pemeriksaan terhadap nilai kosong (*null*) dan duplikat, ditemukan sebanyak 17 baris data duplikat untuk pemeriksaan

nilai kosong dan duplikat data dalam pemrograman *Python* dapat menggunakan kode berikut:

```
df_phish = df_phish[['URL']].dropna().drop_duplicates()
df_phish['label'] = 0
```

Data tersebut kemudian dihapus, sehingga jumlah data *phising* yang digunakan dalam penelitian menjadi 54.630. Sedangkan untuk data *legitimate*, sebanyak 54.000 URL dikumpulkan, yang semuanya dinyatakan valid tanpa nilai kosong maupun duplikat. Oleh karena itu, total gabungan dataset yang digunakan setelah proses pembersihan adalah sebanyak 108.630 *URL*.

Proses *preprocessing* dilakukan untuk memastikan bahwa data yang akan digunakan dalam pelatihan model bebas dari kesalahan atau duplikasi, yang dapat memengaruhi performa model secara signifikan. Selain itu, pada tahap ini juga dilakukan pemisahan antara fitur (fitur *URL*) dan label (klasifikasi *phising* atau *legitimate*), serta penyamaan struktur dataset agar kedua jenis data dapat digabungkan dalam satu DataFrame. Proses ini penting untuk menjamin kualitas data sebelum masuk ke tahap ekstraksi fitur.

4.4 Ekstraksi Fitur

Tahapan ini bertujuan untuk mengekstraksi karakteristik dari setiap URL yang dapat dijadikan sebagai parameter klasifikasi. Ekstraksi dilakukan menggunakan fungsi *extract_features()*, yang dirancang untuk menghasilkan sejumlah fitur statistik dari struktur *URL*, seperti panjang *URL*, jumlah titik (*dot*), jumlah karakter khusus, jumlah subdomain, keberadaan protokol *HTTPS*, jumlah karakter angka, dan jumlah tanda tanya serta tanda sama dengan dalam *URL*. Fitur-fitur ini secara tidak langsung merepresentasikan pola umum dari URL *phising* yang biasanya cenderung lebih panjang, mengandung banyak simbol dan karakter mencurigakan, serta meniru tampilan domain sah. Berikut fitur fitur yang diekstraksi berdasarkan struktur URL dapat dilihat pada tabel 4.1.

Tabel 4.1 Daftar URL yang Digunakan

No	Nama Fitur	Keterangan
1	*URL_length*	Panjang total karakter dalam URL

No	Nama Fitur	Keterangan
2	*num_dots*	Jumlah tanda titik (.) dalam URL, biasanya menandakan banyaknya subdomain
3	*num_hyphens*	Jumlah tanda strip atau minus (-), sering digunakan dalam URL *phising*
4	*num_at*	Jumlah simbol @, indikasi umum *phising URL*
5	*num_question_marks*	Jumlah tanda tanya (?), biasanya dalam parameter query
6	*num_equals*	Jumlah tanda sama dengan (=), juga umum di query string URL
7	*num_slashes*	Jumlah garis miring (/) menunjukkan tingkat kedalaman URL
8	*num_digits*	Jumlah karakter angka dalam *URL*
9	*num_special_chars*	Jumlah karakter *non-alfanumerik* selain simbol umum
10	*has_https*	Nilai biner (1/0), apakah URL menggunakan protokol HTTPS (aman atau tidak)
11	*num_subdomains*	Jumlah subdomain berdasarkan jumlah titik di *hostname*

Fitur yang digunakan berjumlah 11, dipilih karena banyak penelitian sebelumnya menunjukkan bahwa URL *phising* memiliki pola struktur tertentu, seperti panjang URL berlebihan, banyak tanda khusus, penggunaan simbol "@" atau "=", serta tidak menggunakan *HTTPS*. Dengan mengekstraksi dan menggabungkan fitur-fitur ini, model machine learning dapat mempelajari perbedaan pola yang umum pada *phising* dan *legitimate URL*. Semuanya diekstrak dari struktur string URL tanpa perlu mengakses konten halaman tersebut. Hal ini membuat metode deteksi ini lebih efisien secara waktu dan aman, karena tidak perlu mengunjungi halaman yang berpotensi berbahaya. Seluruh fitur yang diperoleh kemudian disimpan dalam sebuah DataFrame baru yang juga mencakup label asli dari URL tersebut, yaitu 0 untuk *phising* dan 1 untuk *legitimate*. Berdasarkan

analisis terhadap struktur URL dan referensi penelitian terdahulu, beberapa fitur terbukti sangat berpengaruh dalam klasifikasi URL *phising*, yaitu:

1. Panjang URL (*URL_length*), URL yang terlalu panjang seringkali menandakan upaya untuk menyembunyikan domain asli.
2. Jumlah subdomain (*num_subdomains / num_dots*), banyaknya subdomain menambahkan kerumitan URL, sering digunakan dalam *phising.*
3. Simbol "@" (*num_at*), digunakan untuk mengecoh pengguna dengan mengarahkan tampilan awal URL ke domain palsu.
4. Karakter khusus seperti "?", "=", atau karakter non-alfanumerik (*num_special_chars*), umumnya muncul dalam URL *phising* yang mencoba mengelabui sistem keamanan.
5. Ketiadaan protokol HTTPS (*has_https=0*), menjadi indikator umur dari sistus tidak aman atau palsu.
6. Jumlah digit (*num_digits*), banyak digunakan untuk membuat URL terlihat acak atau tidak menyerupai pola sah.
7. Jumlah garis miring (*num_slashes)*, URL dengan banyak folder atau path sering digunakan untuk meyembunyikan konten asli.

4.5 Normalisasi Data

Setelah fitur diekstraksi, dilakukan proses normalisasi terhadap seluruh nilai fitur numerik menggunakan metode *Min-Max Scaler*. Tujuan dari normalisasi adalah untuk memastikan bahwa setiap fitur memiliki skala yang sama, sehingga algoritma pembelajaran tidak bias terhadap fitur tertentu yang memiliki rentang nilai lebih besar. Proses normalisasi ini menghasilkan nilai antara 0 dan 1 untuk setiap fitur, yang kemudian digunakan dalam pelatihan model. Gambar 4.3 menunjukkan hasil dari normlisasi pada setiap fitur.

	url_length	num_dots	num_hyphens	num_at	num_question_marks	num_equals	num_slashes	num_digits	num_special_chars	has_https	num_subdomains
0	0.000627	0.000000	0.02	0.0	0.0	0.0	0.009804	0.000000	0.008547	1.0	0.0
1	0.000862	0.021739	0.02	0.0	0.0	0.0	0.009804	0.000000	0.012821	1.0	0.0
2	0.000823	0.021739	0.02	0.0	0.0	0.0	0.000000	0.001465	0.008547	0.0	0.1
3	0.001137	0.021739	0.02	0.0	0.0	0.0	0.000000	0.001465	0.008547	0.0	0.1
4	0.001490	0.021739	0.00	0.0	0.0	0.0	0.029412	0.000000	0.017094	1.0	0.1

Gambar 4.3 Normalisasi dengan MinMax Scaler

Dengan normalisasi ini, model pembelajaran dapat melakukan konvergensi secara lebih stabil dan efisien, serta meminimalisir terjadinya ketidakseimbangan bobot antar fitur. Hal ini sangat penting, terutama pada model pohon keputusan yang sensitif terhadap nilai skala input. Hasil dari proses ini adalah dataset yang telah siap untuk dibagi ke dalam data latih dan data uji.

4.6 Pelatihan dan Pengujian Model *Decision Tree CART*

Dataset hasil normalisasi kemudian dibagi menjadi dua bagian, yaitu 80% data latih dan 20% data uji, menggunakan metode *stratified split*. Pembagian ini dilakukan untuk menjaga proporsi label antara *phising* dan *legitimate* tetap seimbang dalam kedua subset data. Gambar 4.4 dan 4.5 pembagian dataset menghasilkan 86904 data latih dan 21726 data uji berikut.

Data Latih (Training Set):
Jumlah: 86904 sampel

	url_length	num_dots	num_hyphens	num_at	num_question_marks	num_equals	num_slashes	num_digits	num_special_chars	has_https	num_subdomains	label
40824	0.000627	0.021739	0.02	0.0	0.000000	0.00000	0.009804	0.001465	0.012821	1.0	0.1	0.0
65750	0.000235	0.000000	0.00	0.0	0.000000	0.00000	0.000000	0.000000	0.000000	1.0	0.0	1.0
92938	0.000392	0.000000	0.00	0.0	0.000000	0.00000	0.000000	0.000000	0.000000	1.0	0.0	NaN
44033	0.006389	0.021739	0.02	0.0	0.058824	0.06383	0.049020	0.004395	0.064103	1.0	0.1	1.0
68080	0.000078	0.000000	0.00	0.0	0.000000	0.00000	0.000000	0.000000	0.000000	1.0	0.0	1.0

Gambar 4.4 Data Latih

Data Uji (Testing Set):
Jumlah: 21726 sampel

	url_length	num_dots	num_hyphens	num_at	num_question_marks	num_equals	num_slashes	num_digits	num_special_chars	has_https	num_subdomains	label
30911	0.000823	0.021739	0.00	0.0	0.000000	0.00000	0.009804	0.000000	0.008547	1.0	0.1	NaN
71106	0.000235	0.000000	0.00	0.0	0.000000	0.00000	0.000000	0.000000	0.000000	1.0	0.0	NaN
85057	0.000627	0.000000	0.08	0.0	0.000000	0.00000	0.000000	0.000000	0.017094	1.0	0.0	NaN
46330	0.006311	0.021739	0.08	0.0	0.058824	0.06383	0.049020	0.005567	0.076923	1.0	0.1	NaN
10682	0.000784	0.021739	0.00	0.0	0.000000	0.00000	0.009804	0.000000	0.008547	1.0	0.1	1.0

Gambar 4.5 Data Uji

Model yang digunakan untuk klasifikasi adalah algoritma *Decision Tree* dengan pendekatan CART *(Classification and Regression Tree)*, menggunakan kriteria pemisahan *Gini Impurity* dan kedalaman pohon maksimal 5 untuk mencegah *overfitting*. Implementasi untuk pelatihan model dapat dilihat pada kode berikut.

```
from sklearn.tree import DecisionTreeClassifier

# Inisialisasi model CART dengan kriteria Gini dan kedalaman
maksimum
```

```
model = DecisionTreeClassifier(criterion='gini', max_depth=5,
random_state=42)
model.fit(X_train, y_train)
```

Model ini kemudian digunakan untuk mengklasifikasikan data uji dan mengevaluasi performa klasifikasinya pada tahap berikutnya.

4.7 Evaluasi Model *Decision Tree CART*

Setelah model *Decision Tree* selesai dilatih menggunakan data latih, langkah selanjutnya adalah melakukan evaluasi terhadap performa model dengan data uji. Proses evaluasi dilakukan dengan cara memprediksi label dari data uji dan menyimpannya ke dalam variabel prediksi. Untuk mengetahui evaluasi performa model dapat menggunakan kode berikut.

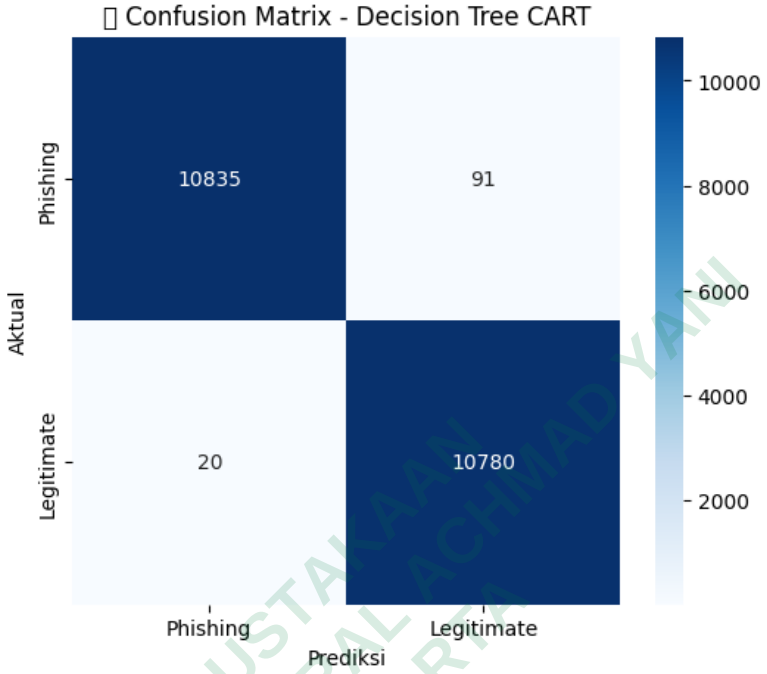
```
y_pred = model.predict(X_test)
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\n Classification Report:\n",
classification_report(y_test, y_pred))
```

Confusion matrix memberikan informasi jumlah prediksi yang benar dan salah untuk masing-masing kelas, baik *phising* maupun *legitimate*. Sementara itu, *classification report* memberikan metrik evaluasi yang lebih detail seperti nilai *precision, recall, F1-score*, dan support untuk masing-masing kelas. Hasil perhitungan dari *Confusion Matrix* dapat dilihat pada tabel 4.2 berikut.

Tabel 4.2 *Confusion Matrix*

	Prediksi *Phising*(0)	**Prediksi *Legitimate*(1)**
Aktual *Phising* (0)	10.835 (*True Negative*)	91 (*False Positive*)
Aktual *Legitimate* (1)	20 (*False Negative*)	10.780 (*True Positive*)

Visualisasi confusion matrix dengan model Decision Tree CART dapat dilihat pada gambar 4.6 berikut.



Gambar 4.6 Visualisasi Confusion Matrix CART

Berdasarkan nilai confusion matrix:

1. True Positive (TP) = 10.780
2. True Negative (TN) = 10.835
3. False Positive (FP) = 91
4. False Negative (FN) = 20

Maka perhitungan metrik dilakukan sebagai berikut:

1. Akurasi

   $$Akurasi = \frac{TP+TN}{TP+TN+FP+FN} = \frac{10.780+10.835}{10.780+10.835+91+20} = \frac{21.615}{21.726} \approx 0.995$$

2. *Precision*

   a. Untuk kelas *phising* (label 0):

   $$Precision_0 = \frac{TN}{TN+FP} = \frac{10.835}{10.835+91} = \frac{10.835}{10.926} \approx 0.9917$$

   b. Untuk kelas *legitimate* (label 1):

$$Precision_1 = \frac{TN}{TP+FN} = \frac{10.780}{10.780+20} = \frac{10.780}{10.800} \approx 0.9981$$

3. *Recall*

   a. Untuk kelas *phising* (label 0):

   $$Recall_0 = \frac{TN}{TN+FN} = \frac{10.835}{10.835+20} = \frac{10.835}{10.855} \approx 0.9981$$

   b. Untuk kelas *legitimate* (label 1):

   $$Recall_1 = \frac{TP}{TP+FP} = \frac{10.780}{10.780+91} = \frac{10.780}{10.871} \approx 0.9917$$

4. *F1-Score*

   a. Untuk kelas *phising* (label 0):

   $$F1 - Score_0 = 2 \times \frac{Recall \times Precision}{Recall + Precision} = \frac{0.9917 \times 0.9981}{0.9917 + 0.9981} \approx 0.9949$$

   b. Untuk kelas *legitimate* (label 1):

   $$F1 - Score_1 = 2 \times \frac{Recall \times Precision}{Recall + Precision} = \frac{0.9981 \times 0.9917}{0.9981 + 0.9917} \approx 0.9949$$

Hasil perhitungan dari *classification report* dengan kedua *class label phising* dan *legitimate* dapat dilihat pad tabel 4.3 berikut.

Tabel 4.3 *Classification Report*

Class Label	***Precision***	***Recall***	***F1-Score***	***Support***
0 (*Phising*)	1.00	0.99	0.99	10.926
1 (*Legitimate*)	0.99	1.00	0.99	10.800
Accuracy			0.99	21.726
Macro AVG	0.99	0.99	0.99	21.726
Weighted AVG	0.99	0.99	0.99	21.726

Berdasarkan hasil di atas, model Decision Tree CART menunjukkan performa klasifikasi yang sangat baik dalam membedakan antara URL phishing dan legitimate. Precision untuk kelas phishing sebesar 1.00 menunjukkan bahwa seluruh URL yang diprediksi sebagai phishing memang benar-benar phishing. Sementara itu, precision untuk kelas legitimate sebesar 0.99 menandakan bahwa sebagian besar prediksi legitimate juga akurat, meskipun terdapat sedikit kesalahan.

Dari sisi recall, model juga menunjukkan performa yang baik. Recall sebesar 0.99 untuk kelas phishing mengindikasikan bahwa 99% dari URL phishing berhasil dikenali dengan benar. Recall 1.00 untuk kelas legitimate menunjukkan bahwa semua URL legitimate berhasil diklasifikasikan tanpa adanya kesalahan.

Nilai F1-score yang tinggi (0.99) pada kedua kelas menunjukkan keseimbangan yang baik antara precision dan recall. Hal ini sangat penting dalam sistem deteksi phishing, di mana kesalahan klasifikasi dapat berdampak pada risiko keamanan.

Akurasi model sebesar 99% menegaskan bahwa model mampu memberikan prediksi yang tepat terhadap hampir seluruh data uji. Nilai macro average dan weighted average yang sama-sama tinggi juga menunjukkan bahwa model memiliki performa yang konsisten, baik untuk kelas yang seimbang maupun tidak seimbang.

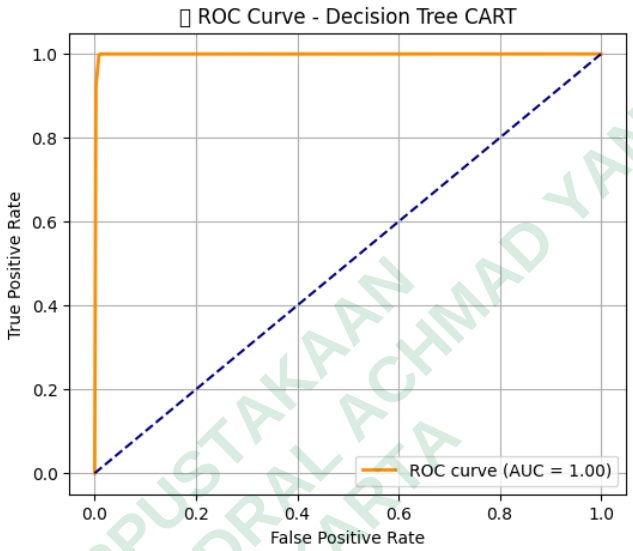
4.7.1 ROC Curve & AUC

Receiver Operating Characteristic (ROC) curve merupakan metode evaluasi yang menggambarkan hubungan antara *True Positive Rate (TPR)* dan *False Positive Rate (FPR)* pada berbagai *threshold*. Untuk membentuk kurva ROC, digunakan probabilitas prediksi dari model terhadap kelas positif (dalam hal ini, *legitimate URL*). Pemrosesan ROC *curve* dan AUC dapat menggunakan kode berikut.

```
y_probs = model.predict_proba(X_test)[:, 1]
fpr, tpr, thresholds = roc_curve(y_test, y_probs)
roc_auc = auc(fpr, tpr)

plt.figure(figsize=(6, 5))
plt.plot(fpr, tpr, color='darkorange', lw=2, label=f"ROC curve (AUC
= {roc_auc:.2f})")
plt.plot([0, 1], [0, 1], color='navy', linestyle='--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('📉 ROC Curve - Decision Tree CART')
plt.legend(loc="lower right")
plt.grid(True)
plt.show()
```

Nilai AUC yang mendekati 1 menunjukkan bahwa model memiliki kemampuan klasifikasi yang sangat baik. Sebaliknya, nilai AUC mendekati 0.5 menunjukkan performa yang setara dengan tebakan acak. Visualisasi hasil ROC Curve dan perhitungan AUC dari model Decision Tree CART ditampilkan pada Gambar 4.7 berikut.



Gambar 4.7 Kurva ROC AUC

4.7.2 *Precision-Recall Curve*

Selain menggunakan kurva ROC, evaluasi performa model juga dilakukan melalui *Precision-Recall* (PR) *Curve*. Evaluasi ini menjadi sangat berguna ketika distribusi data tidak seimbang atau ketika perhatian utama tertuju pada nilai f*alse positive* dan *false negative*, seperti dalam kasus deteksi *phising*.

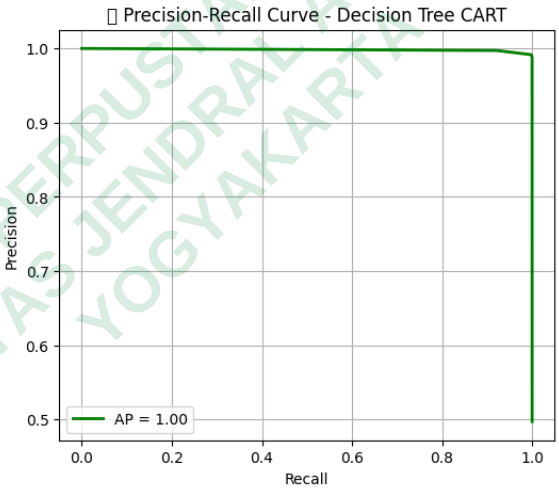
Precision mengukur proporsi prediksi *phising* yang benar dibandingkan dengan seluruh prediksi *phising* yang dilakukan oleh model. *Recall* (*sensitivitas*) mengukur proporsi *phising* yang berhasil dikenali oleh model dari seluruh data *phising* yang sebenarnya.

Precision dan *recall* dihitung dengan *precision_recall_curve*, dan nilai *Average Precision* dihitung dengan *average_precision_score*. Berikut adalah implementasi kode untuk menghitung dan memvisualisasikan kurva *Precision-Recall* menggunakan *Google Colab*.

```
precision, recall, _ = precision_recall_curve(y_test, y_probs)
avg_precision = average_precision_score(y_test, y_probs)

plt.figure(figsize=(6, 5))
plt.plot(recall, precision, color='green', lw=2, label=f"AP = {avg_precision:.2f}")
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.title('📈 Precision-Recall Curve - Decision Tree CART')
plt.legend(loc="lower left")
plt.grid(True)
plt.show()
```

Kurva ini divisualisasikan dengan warna hijau, dan semakin tinggi *average precision*, semakin baik performa model dalam hal presisi dan sensitivitas. Visualisasi hasil *Precision-Recall Curve* untuk model *Decision Tree* CART ditampilkan pada Gambar 4.8.



Gambar 4.8 Kurva *Precision-Recall*

4.8 Cross validation dan Evaluasi Fairness

Untuk mengevaluasi kestabilan model dan menghindari bias akibat pembagian data uji tunggal, dilakukan proses validasi silang *(cross-validation)* sebanyak *5-fold* pada data pelatihan. Teknik ini bertujuan untuk memastikan bahwa performa model tidak bergantung pada subset data tertentu serta dapat digeneralisasikan dengan baik terhadap data baru.

Hasil validasi silang menunjukkan bahwa model menghasilkan nilai akurasi rata-rata sebesar 0.9956 dengan standar *deviasi* sebesar 0.0039. Nilai ini

mengindikasikan bahwa model memiliki performa yang stabil dan konsisten di setiap *fold*, serta tidak mengalami variasi besar terhadap perubahan data pelatihan.

Selain itu, evaluasi *fairness* dilakukan dengan membandingkan metrik performa antar kelas, khususnya antara kelas *phising* dan *legitimate*. Hasil evaluasi menunjukkan bahwa tidak terdapat ketimpangan signifikan dalam performa model terhadap kedua kelas tersebut yang dapat dilihat pada tabel 4.4 dan 4.5 berikut.

Tabel 4.4 Hasil Cross-Validation 5-Fold

Fold	Akurasi
1	0.9961
2	0.9958
3	0.9961
4	0.9959
5	0.9944
Rata-rata Akurasi	**0.9957**
Standar Devisiasi	**0.0007**

Tabel 4.5 Akurasi Model pada Data Latih dan Uji

Dataset	Akurasi
Data Latih	0.9957
Data Uji	0.9949

4.9 Visualisasi Pohon Keputusan

Visualisasi pohon keputusan dihasilkan menggunakan fungsi *plot_tree()* dari pustaka *Scikit-learn*. Hasil visualisasi memperlihatkan bagaimana setiap fitur berkontribusi terhadap proses pengambilan keputusan. Model pohon ini dibangun dengan data yang diproses oleh platform *Google colab*, kode untuk membangun model ini sebagai berikut.

```
plt.figure(figsize=(20, 10))
plot_tree(
    model,
    filled=True,
    feature_names=X.columns,
```

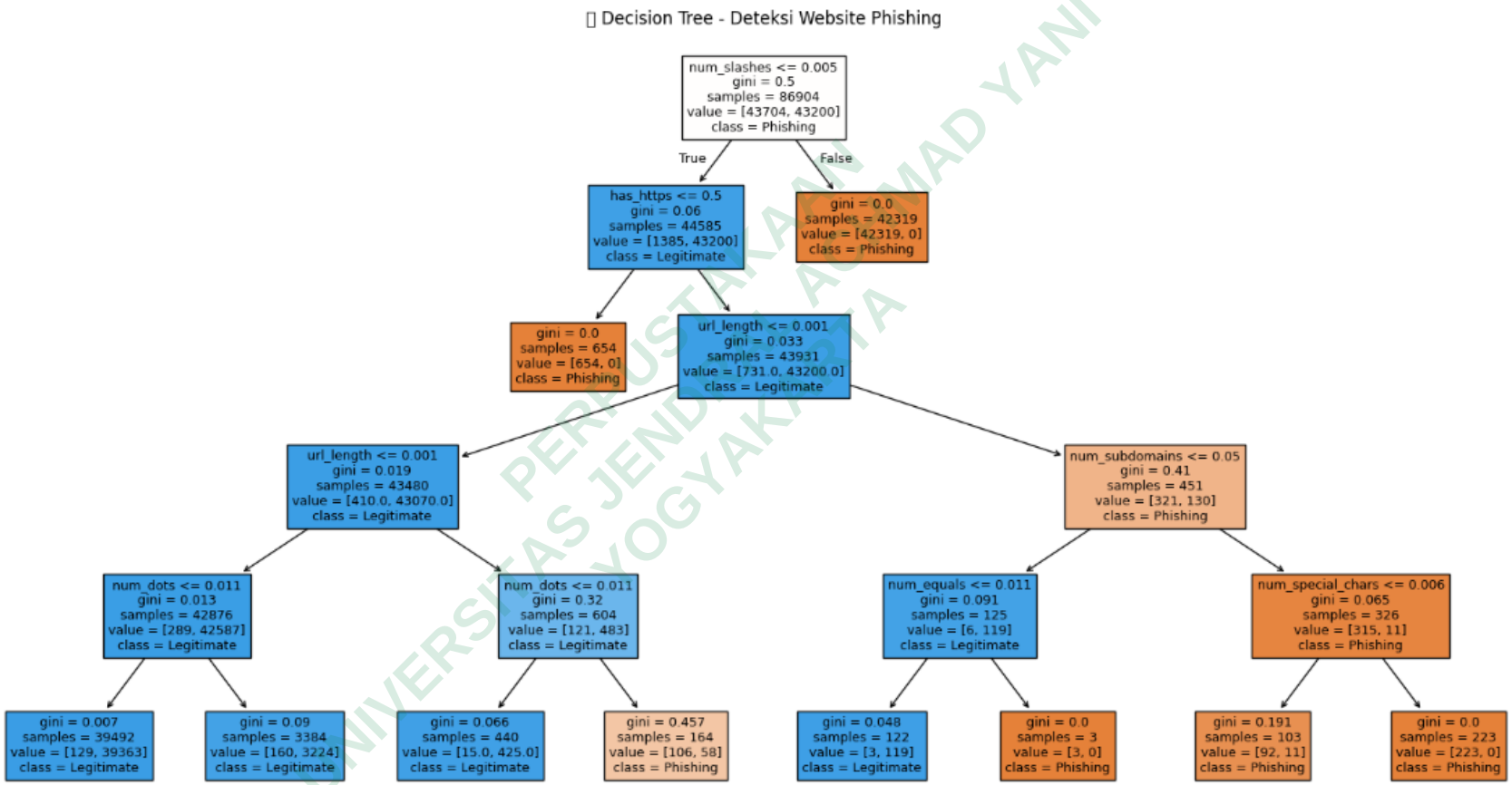
```
    class_names=["Phising", "Legitimate"]  # Sesuai urutan label: 0 → Phising, 1 → Legitimate)
plt.title("🌳 Decision Tree - Deteksi Website Phising")
plt.show()
```

Hasil visualisasi menunjukkan bahwa node akar *(root)* dari pohon menggunakan fitur *num_slashes* (jumlah garis miring dalam URL) sebagai pemisah awal. URL dengan jumlah garis miring yang lebih sedikit dari ambang batas tertentu (≤ 0.005) diarahkan ke cabang kiri, yang umumnya merepresentasikan URL *legitimate*. Sebaliknya, URL dengan jumlah garis miring yang lebih banyak diarahkan ke cabang kanan, yang sebagian besar diklasifikasikan sebagai *phising*. Hal ini terlihat pada node di sisi kanan dengan nilai *Gini* = 0, yang memuat sebanyak 42.319 URL, semuanya diprediksi sebagai *phising*.

Pada kiri dari pohon memperlihatkan pemisahan lebih lanjut berdasarkan keberadaan protokol HTTPS *(has_https)*, yang merupakan fitur krusial dalam mendeteksi keamanan URL. URL yang tidak menggunakan HTTPS (nilai 0) cenderung mengarah ke prediksi *phising*, sedangkan yang menggunakan HTTPS diklasifikasikan lebih lanjut berdasarkan panjang URL *(URL_length)*. URL yang pendek cenderung dianggap *legitimate*, sedangkan URL yang terlalu panjang perlu diuji dengan fitur tambahan seperti jumlah titik *(num_dots)* dan simbol khusus *(num_special_chars)*, yang juga merupakan ciri khas URL *phising*.

Node-node selanjutnya menunjukkan bagaimana model mengevaluasi fitur-fitur seperti jumlah subdomain *(num_subdomains)*, jumlah simbol sama dengan *(num_equals)*, dan jumlah karakter angka *(num_digits)* untuk memperhalus klasifikasi terhadap URL yang memiliki struktur ambigu.

Warna pada setiap node menggambarkan proporsi dari masing-masing kelas biru untuk *legitimate* dan oranye untuk *phising*. Warna yang semakin pekat menunjukkan semakin kuatnya keyakinan model terhadap klasifikasi tersebut, sedangkan node-node dengan nilai gini rendah menunjukkan bahwa data dalam node tersebut relatif homogen. Model pohon keputusan (*Decision Tree)* yang telah dibangun dengan memproses data pada pemrograman Google colab dapat dilihat pada gambar 4.9 berikut.



Gambar 4.9 *Decision Tree*

Secara keseluruhan, visualisasi pohon keputusan ini sangat berguna untuk memahami logika internal dari model klasifikasi yang digunakan. Selain menunjukkan hasil akhir prediksi, visualisasi ini juga menjelaskan jalur keputusan yang diambil oleh model berdasarkan fitur-fitur yang relevan. Hal ini memberikan tingkat transparansi dan interpretabilitas yang tinggi, serta menunjukkan bahwa model mampu menangkap pola-pola struktural khas yang membedakan URL *phising* dari *legitimate* dengan cukup baik.

4.10 Uji Prediksi *Real-Time*

Sebagai bentuk pengujian tambahan, dilakukan pengujian terhadap 10 URL secara *real-time*. URL ini diambil dari *phistank* dan *tranco* secara acak, dengan menggunakan model yang telah dilatih. Uji *real-time* terhadap URL dilakukan pemrosesan oleh Google colab dengan kode sebagai berikut.

```
def predict_multiple_URLs(URL_list):
      feature_dicts = [extract_features_single(URL) for URL in
URL_list]
    df_features = pd.DataFrame(feature_dicts)
    scaled = scaler.transform(df_features)
    preds = model.predict(scaled)
    probs = model.predict_proba(scaled)

    results = []
    for URL, pred, prob in zip(URL_list, preds, probs):
        label = "✅ Legitimate" if pred == 1 else "⚠️ Phising"
        results.append({
            "URL": URL,
            "Prediction": label,
            "Prob_Legitimate": round(prob[1], 2),
            "Prob_Phising": round(prob[0], 2)
        })
    return pd.DataFrame(results)
real_time_URLs = [
    "https://force.com",
    "https://sites.google.com/view/protonmailservice/home",
    "https://wired.com",
    "https://gitlab.com",
    "https://sites.google.com/view/awspage",
    "http://dhanushr24.github.io/clone-Netflix/",
```

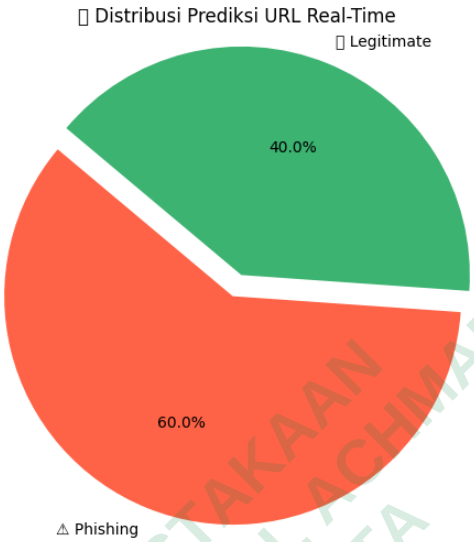
```
    "http://soumya252000.github.io/Netflix/",
    "https://xiaomi.net",
    "http://52.20.22.249:8080/BannerTool/HtmlPages/welcome",
    "https://server-networksolutions.web.app/#adsf@adsf.com"
]
```

Hasilnya divisualisasikan dalam bentuk tabel prediksi, serta grafik pie chart yang menunjukkan proporsi prediksi *phising* dan *legitimate*. Model mampu mendeteksi URL mencurigakan dengan akurasi yang cukup tinggi, meskipun beberapa URL menggunakan domain yang terlihat sah tetapi memiliki struktur yang kompleks. Hasil dari visualisasi kode pemrosesan pada Google colab berupa tabel prediksi dan grafik pie chart dapat dilihat pada tabel 4.6 dan gambar 4.10 sebagai berikut.

Tabel 4.6 Prediksi *Real-Time*

Index	**URL**	***Prediction***	***Prob_Legitimate***	***Prob_Phising***
0	https://force.com	*Legitimate*	1.0	0.0
1	https://sites.google.com/view/protonmailservice/home	*Phising*	0. 0	1.0
2	https://wired.com	*Legitimate*	1.0	0.0
3	https://gitlab.com	*Legitimate*	0.98	0.02
4	https://sites.google.com/view/awspage	*Phising*	0.0	1.0
5	http://dhanushr24.github.io/clone-Netflix/	*Phising*	0.0	1.0
6	http://soumya252000.github.io/Netflix/	*Phising*	0.0	1.0
7	https://xiaomi.net	*Legitimate*	0.98	0.02
8	http://52.20.22.249:8080/BannerTool/HtmlPages/welcome	*Phising*	0.0	1.0
9	https://server-networksolutions.w	*Phising*	0.0	1.0

Index	URL	*Prediction*	*Prob_Legitimate*	*Prob_Phising*
	eb.app/#adsf@adsf.com			



Gambar 4.10 *Pie Chart* Distribusi Prediksi *Real-Time*

Pengujian *real-time* ini membuktikan bahwa model tidak hanya bekerja baik pada data uji, tetapi juga mampu menangani data baru dengan efektif. Hal ini menjadi indikator bahwa model memiliki kemampuan untuk diterapkan secara nyata dalam sistem keamanan URL berbasis deteksi *website phising*.