Faktor Exacta

Vol. 17, No. 1, March 2024, pp. 58~68
eISSN: 2502-339X, pISSN: 1979-276X, DOI: 10.30998/faktorexacta.v17i1.22393

Eksplorasi Teknik *Web Scraping* pada Data Mining: Pendekatan Pencarian Data Berbasis Python

Debora Chrisinta[1], Justin Eduardo Simarmata[2]
[1]Program Studi Teknologi Informasi, Universitas Timor, Indonesia
[2]Program Studi Pendidikan Matematika, Universitas Timor, Indonesia

Article Info

Article history:

Received Feb 16, 2024
Revised Apr 19, 2024
Accepted Apr 23, 2024

Keywords:

Scraping
Data Mining
Hierarchical Clustering
Decision Tree
Python

ABSTRACT

Web scraping was an automated information extraction technique from web pages for data collection and was applied in data mining. Two common algorithms used in data mining are clustering and classification. The data source used originated from the Google Search Engine. The design of the web scraping script using Python was implemented to collect data, process HTML, and extract information from web pages. Data was successfully gathered from the Google Search Engine regarding tourism, with the number of links and processing time measured. Data processing involved cleaning the data and implementing hierarchical clustering algorithms. The evaluation was carried out by selecting the optimal number of clusters using the Dunn index. Subsequently, the data was used to train a decision tree model, and the results were evaluated using accuracy, confusion matrix, and classification reports. The results of this research indicated that the importance of web scraping in data mining could provide a comprehensive understanding of the effectiveness of web scraping techniques and the application of data mining.

Corresponding Author:

Debora Chrisinta,
Program Studi Teknologi Informasi,
Universitas Timor,
Sasi, Kec. Kota Kefamenanu, Kabupaten Timor Tengah Utara, Nusa Tenggara Timur
Email: deborachrisinta@unimor.ac.id

1. PENDAHULUAN

Web scraping adalah proses ekstraksi informasi secara otomatis dari halaman web menggunakan program komputer atau skrip. Teknik ini dapat mengumpulkan data seperti teks, gambar, dan informasi lainnya dari berbagai situs *web*. Keuntungan utama dari web scraping melibatkan pengumpulan data secara otomatis, memungkinkan analisis dan penelitian efisien, serta integrasi data untuk meningkatkan efisiensi operasional [1]. Meskipun demikian, terdapat beberapa kelemahan, seperti pelanggaran hak cipta, ketergantungan pada struktur HTML, dan tantangan etika serta legalitas [2]. Penerapan *web scraping* melibatkan banyak bidang, seperti pemantauan harga produk, analisis sentimen dari media sosial, pembaruan berita otomatis, dan penelitian pasar untuk memahami tren industri dan kebutuhan pelanggan. Meski memberikan keuntungan signifikan, penggunaan *web scraping* perlu memperhatikan aspek-etika dan kepatuhan hukum agar tidak menimbulkan masalah.

Pengumpulan data melalui web scraping memungkinkan data mining untuk dilakukan pada dataset yang lebih besar dan representatif. Dua jenis algoritma yang umum digunakan untuk mengelompokkan dan mengklasifikasikan data dalam data mining yaitu *clustering* dan klasifikasi. *Clustering* pada *web scraping*

dapat digunakan untuk mengelompokkan data yang diperoleh dari berbagai situs *web* ke dalam kelompok-kelompok yang memiliki kemiripan tertentu [3][4][5]. Ini membantu dalam mengidentifikasi pola dan struktur yang mungkin ada dalam data *web*. Misalnya, dengan menggunakan algoritma *clustering* seperti *hierarchical clustering* pada data produk dari beberapa situs *e-commerce*, dapat mengelompokkan produk dengan fitur serupa ke dalam kelompok-kelompok tertentu [6]. Hal ini dapat membantu dalam membuat katalog yang terorganisir atau memahami preferensi konsumen. Sedangkan pada algoritma kasifikasi, *web scraping* dapat membantu dalam memberi label pada data yang diperoleh berdasarkan karakteristik atau fitur tertentu. Ini memungkinkan untuk mengorganisir dan memahami data dengan lebih baik. Sebagai contoh, dengan menggunakan algoritma klasifikasi seperti *Decision Trees*, dapat memberikan label pada ulasan produk yang diambil dari situs *review* untuk menentukan apakah suatu ulasan bersifat positif, negatif, atau netral [7]. Ini memudahkan dalam analisis sentimen terhadap produk atau merek.

Pentingnya *web scraping* dalam data mining untuk algoritma klasifikasi dan *clustering* juga tercermin dalam kemampuannya untuk menyediakan data *real-time* atau berkala. Hal ini memungkinkan model klasifikasi dan *clustering* untuk tetap relevan dengan perubahan yang terjadi dalam data secara terus-menerus, memberikan pemahaman yang lebih dinamis tentang lingkungan atau *trend* yang sedang diamati. Beberapa penelitian terkini telah berhasil menerapkan pengumpulan data melalui *web scraping* dalam proses data mining. Seperti penelitian oleh [8] yang melakukan analisis produk di *marketplace* secara otomatis. Hasil penelitian ini menunjukkan bahwa data yang dikumpulkan melalui *web scraping* sesuai dengan kebutuhan yang diinginkan, menandakan keberhasilan dalam mengoptimalkan proses pengumpulan data. Selain itu, Priadana dan Murdiyanto juga melibatkan *web scraping* dalam penelitiannya, fokus pada pemantauan *trend hashtag* di Instagram [9]. Dengan menggunakan teknik *web scraping*, penelitian ini berhasil menampilkan posting dengan jumlah *like* dan jumlah *comment* terbanyak dari suatu *hashtag* pada periode waktu tertentu. Hasilnya mencerminkan efektivitas pengumpulan data melalui *web scraping* untuk memahami *trend* dan interaksi pengguna di *platform* media sosial.

Sebagai tambahan, penelitian oleh [10] menonjolkan penerapan *web scraping* untuk analisis sentimen pemilik dompet digital. Dalam penelitian ini, teknik *web scraping* diintegrasikan dengan algoritma *machine learning* untuk menghasilkan hasil analisis sentimen dengan tingkat akurasi di atas 80%. Selanjutnya, Fahrudin dkk melakukan penelitain dalam mengimplementasikan *web scraping* pada *google search engine* dalam rangka pengumpulan data *list* 2D terstruktur [11]. Hal ini menegaskan bahwa *web scraping* bukan hanya efektif untuk mengumpulkan data, tetapi juga berperan penting dalam memberikan dasar data yang berkualitas untuk analisis lanjutan menggunakan algoritma *machine learning*.

Secara keseluruhan, penelitian di atas ini mencerminkan keberhasilan dan fleksibilitas *web scraping* dalam mendukung proses data mining di berbagai domain, dari analisis produk *online* hingga pemantauan *trend* dan analisis sentimen, memberikan kontribusi signifikan pada pemahaman dan pemanfaatan data dalam berbagai konteks. Oleh karena itu pada penelitian ini, bertujuan untuk mendalami dan mengeksplorasi berbagai teknik *web scraping* yang dapat diterapkan dalam konteks data mining dengan pendekatan berbasis Python. Selain itu, penelitian ini bertujuan untuk meningkatkan integrasi antara web scraping dan data mining, sehingga mendukung efektivitas analisis data.

2. METODE

2.1 Studi Pustaka

2.1.1 *Web Scraping*

Web scraping adalah teknik ekstraksi data otomatis dari halaman-halaman web. Proses ini melibatkan penggunaan program komputer atau skrip untuk mengambil dan mengumpulkan informasi dari berbagai situs *web*. Tujuan utama dari *web scraping* adalah untuk mendapatkan data yang terstruktur dari *web* dan menggunakannya untuk berbagai keperluan seperti analisis, penelitian pasar, pemantauan pesaing, atau otomatisasi tugas tertentu. Beberapa komponen utama dari *web scraping* meliputi [12]:

a) HTTP *requests* yaitu pengambilan data dimulai dengan pengiriman permintaan HTTP ke server *web*. Permintaan ini dapat berupa GET untuk mengambil informasi dari *server* atau POST untuk mengirimkan data ke *server*.
b) HTML parsing yaitu mengurai (*parsing*) HTML.
c) Seleksi data yaitu memilih data yang terkandung dalam HTML tertentu.

2.1.2 Data Mining

Data mining adalah proses menggunakan alat analisis data canggih untuk menemukan pola dan hubungan yang sebelumnya tidak diketahui dalam kumpulan data besar. Ini melibatkan penggunaan model statistik, algoritma matematika, dan metode pembelajaran mesin seperti jaringan saraf atau pohon keputusan. Data mining tidak hanya terbatas pada pengumpulan dan pengelolaan data, tetapi juga mencakup analisis dan

prediksi. Proses ini dapat dilakukan pada data kuantitatif, tekstual, atau multimedia, dan menggunakan berbagai parameter untuk memeriksa data. Beberapa algoritma dalam data mining terdiri dari asosiasi (mengidentifikasi pola di mana satu peristiwa terkait dengan peristiwa lain), analisis urutan atau jalur (mencari pola di mana satu peristiwa mengarah ke peristiwa lain), klasifikasi (identifikasi pola baru), *clustering* (menemukan dan memvisualisasikan kelompok fakta yang sebelumnya tidak diketahui), dan peramalan (mengidentifikasi pola untuk membuat prediksi tentang aktivitas di masa depan) [13].

2.1.3 Python

Python adalah bahasa pemrograman yang diinterpretasikan, interaktif, dan berorientasi objek. Bahasa ini menyediakan struktur data tingkat tinggi seperti daftar dan *array* asosiatif (yang disebut kamus), tiping dinamis, pengikatan dinamis, modul, kelas, manajemen memori otomatis, dan lainnya. Didesain pada tahun 1990 oleh Guido van Rossum, Python memiliki sintaks yang sederhana dan elegan namun merupakan bahasa pemrograman yang kuat dan serbaguna. Seperti banyak bahasa skrip lainnya, Python gratis bahkan untuk keperluan komersial dan dapat dijalankan di hampir semua komputer modern. Program Python dikompilasi otomatis oleh interpreter menjadi bytecode yang independen dari *platform*, yang kemudian diinterpretasikan. Komponen yang ditulis dalam Python dapat dijalankan tanpa modifikasi di bawah berbagai sistem operasi seperti Linux, Windows NT, 98, 95, IRIX, SunOS, dan OSF. Python bersifat modular. Kernelnya sangat kecil dan dapat diperluas dengan mengimpor modul ekstensi. Distribusi Python mencakup perpustakaan ekstensi standar yang beragam (beberapa ditulis dalam Python, yang lain dalam C atau C++) untuk operasi mulai dari manipulasi *string*, hingga pembuat antarmuka pengguna grafis (GUI) dan utilitas terkait *web*, layanan sistem operasi, alat *debugging*, dan *profiling*, dll. Modul ekstensi baru dapat dibuat untuk memperluas bahasa dengan kode baru atau yang sudah ada [14].

2.2 Sumber Data

Sumber data yang digunakan pada penelitian adalah data dari *google search engine*. Penerapan metode data mining, seperti *clustering* dan klasifikasi, pada sumber data dari *google search engine* dapat memberikan wawasan yang berharga dari hasil pencarian dan pola-pola yang mungkin tersembunyi dalam data tersebut. Pada algoritma clustering yang digunakan adalah *hierarchical clustering* dalam mengelompokkan hasil pencarian ke dalam kelompok-kelompok yang memiliki kesamaan. Pada algoritma klasifikasi menerapkan *decision trees* dalam menentukan kategori untuk memahami distribusi topik tertentu dari hasil pencarian.

2.3 Perancangan Skrip Web Scraping

Berikut adalah perancangan skrip program Python *web scraping* untuk melakukan pengumpulan data dengan atribut jumlah tautan dan waktu pemrosesan pada input kueri tertentu:

a) Membuat fungsi untuk *parsing html*, dengan *source code* berikut:

```
from bs4 import BeautifulSoup
def extract_links(html_content):
 # Membuat objek BeautifulSoup
 soup = BeautifulSoup(html_content, 'html.parser')
 # Mencari tautan (URL) dalam elemen 'a'
 links = [a['href'] for a in soup.find_all('a', href=True)]
 return links
```

b) Menggunakan fungsi *requests* untuk mengambil halaman *web*, dengan *source code* berikut:

```
import requests
def get_google_search_links(query, num_results=10):
 # Membuat URL pencarian Google
 search_url = f"https://www.google.com/search?q={query}&num={num_results}"
 # Mengambil konten halaman web menggunakan requests
 response = requests.get(search_url)
 html_content = response.content
 # Menggunakan fungsi extract_links untuk mendapatkan tautan
 links = extract_links(html_content)
 return links
```

c) Mengekstrak informasi atribut yang diinginkan, dengan *source code* berikut:

```
import time
```

```
def extract_page_info(url):
 # Mengambil konten halaman web menggunakan requests
 response = requests.get(url)
 html_content = response.content
 # Menggunakan BeautifulSoup untuk mem-parsing HTML
 soup = BeautifulSoup(html_content, 'html.parser')
 # Mengumpulkan data yang dibutuhkan
 jumlah_halaman = len(soup.find_all('a'))
 jumlah_tautan = len(soup.find_all('a'))
 waktu_pemrosesan = time.process_time()
 return {
 'url': url,
 'jumlah_tautan': jumlah_tautan,
 'waktu_pemrosesan': waktu_pemrosesan
 }
```

d) Menjalankan proses, dengan *source code* berikut:

```
if __name__ == "__main__":
 # Ganti query dengan kata kunci pencarian
 search_query = " "
 # Mendapatkan tautan dari hasil pencarian Google
 search_links = get_google_search_links(search_query, num_results=10)
 # Mengekstrak informasi dari setiap halaman web
 for link in search_links:
 page_info = extract_page_info(link)
 print(page_info)
```

3. HASIL DAN PEMBAHASAN

Data mining adalah proses ekstraksi pola atau pengetahuan yang berguna dari suatu set data yang besar dan kompleks. Pada pengambilan informasi dari internet, teknik *web scraping* menjadi salah satu metode yang populer. Salah satu sumber data yang kaya informasi adalah mesin pencari Google. Penggunaan teknik *web scraping* pada *google search engine* memungkinkan pengumpulan data yang signifikan untuk analisis lebih lanjut. Penerapan data mining pada hasil *web scraping google search engine* membuka potensi untuk mendapatkan wawasan berharga tentang tren pencarian, preferensi pengguna, dan informasi terkait lainnya. Namun, seperti halnya dalam setiap proses pengambilan data, penting untuk memahami dan mengatasi tantangan yang mungkin muncul. Seperti perlu diperhatikan bahwa Google memiliki kebijakan penggunaan yang perlu diikuti oleh pengembang atau peneliti yang menggunakan layanan mereka. Penggunaan *web scraping* harus mematuhi aturan dan tidak melanggar kebijakan privasi atau hak cipta.

Tahapan setelah pengumpulan data dan penerapan pada algoritma data mining yaitu pada penelitian ini *clustering* dan klasifikasi, perlu dilakukan evaluasi. Evaluasi merupakan langkah kritis dalam proses data mining, terutama ketika menggunakan teknik *web scraping*. Evaluasi dilakukan untuk memastikan bahwa data yang diambil sesuai dengan kebutuhan analisis, dan bahwa proses pengambilan data berjalan secara efektif. Pengukuran seperti keakuratan, kelengkapan, dan konsistensi data dapat menjadi fokus evaluasi. Validasi adalah tahap selanjutnya yang diperlukan untuk memastikan bahwa hasil analisis atau model yang dibangun berlaku dengan baik pada data yang baru. Pada konteks *web scraping*, validasi dapat melibatkan pengujian ulang teknik scraping pada periode waktu yang berbeda atau menggunakan parameter pencarian yang berbeda untuk memastikan konsistensi hasil.

Pada bagian pembahasan ini akan dijelaskan lebih rinci mengenai penerapan data mining pada hasil *web scraping google search engine*, beserta evaluasi dan validasi yang diterapkan. Tantangan dan solusi yang ditemui selama proses pengambilan data juga akan dibahas untuk memberikan pemahaman yang komprehensif tentang metode ini. Dengan demikian, pembaca akan mendapatkan gambaran yang jelas tentang keefektifan dan validitas informasi yang diperoleh melalui teknik web scraping pada mesin pencari Google.

Data yang berhasil dikumpulkan menggunakan teknik web scraping dengan kueri pencarian "Pariwisata" adalah sebanyak 27 URL. Adanya keterbatasan dalam mengakses URL disebabkan karena adanya aspek-etika yang dibatasi dalam pengumpulan data pada *google search engine*. Berikut hasil pengumpulan data diberikan pada Tabel 1.

Table 1. Data Tautan (URL) dari Hasil Web Scraping

No	URL	Jumlah_Tautan	Waktu_Pemrosesan (detik)
1	https://id.wikipedia.org/wiki/Pariwisata	339	10.34375
2	https://id.wikipedia.org/wiki/Pariwisata_di_In…	966	10.390625
3	https://kemenparekraf.go.id/rumah-difabel/Meng…	60	10.5
4	https://disbudpar.jatimprov.go.id/	51	10.625
5	https://pariwisata.jogjakota.go.id/	103	10.765625
6	https://dpmptsp.karimunkab.go.id/sektor-pariwi…	119	10.90625
7	https://stipram.ac.id/64-pariwisata-yang-berta…	140	10.96875
8	https://dinpar.kulonprogokab.go.id/	66	11.0625
9	https://www.detik.com/tag/pariwisata	126	11.125
10	https://disparbud.kebumenkab.go.id/	167	11.15625
11	https://www.bisnis.com/topic/355/pariwisata	172	11.28125
12	https://pariwisata.kepulauanselayarkab.go.id/	176	11.359375
13	https://www.instagram.com/kemenparekraf.ri/?hl=en	0	11.484375
14	https://pariwisata.bantulkab.go.id/hal/visi-da…	82	11.59375
15	https://pariamankota.go.id/pemerintahan/dinas?...	79	11.609375
16	https://www.pnb.ac.id/jurusan/pariwisata	0	11.671875
17	https://indonesiabaik.id/infografis/pariwisata…	73	11.75
18	https://btp.ac.id/jenis-jenis-pariwisata/	0	11.8125
19	https://mediakeuangan.kemenkeu.go.id/article/s…	120	11.90625
20	https://www.sindonews.com/topic/1075/pariwisata	0	11.96875
21	https://pariwisata.bantulkab.go.id/hal/visi-da…	82	12.15625
22	https://pariamankota.go.id/pemerintahan/dinas?...	79	12.296875
23	https://www.pnb.ac.id/jurusan/pariwisata	0	12.375
24	https://indonesiabaik.id/infografis/pariwisata…	73	12.46875
25	https://btp.ac.id/jenis-jenis-pariwisata/	0	12.578125
26	https://mediakeuangan.kemenkeu.go.id/article/s…	120	12.6875
27	https://www.sindonews.com/topic/1075/pariwisata	0	12.734375

Pada Tabel 1 kolom URL merupakan alamat situs *web* terkait pariwisata, kolom Jumlah_Tautan merupakan jumlah tautan yang terdapat di situs *web* tersebut, sedangkan Waktu_Pemrosesan (detik) merupakan waktu yang dibutuhkan untuk memproses situs *web* dalam detik. Ada beberapa situs *web* dengan jumlah tautan yang nol, yang mungkin menunjukkan masalah atau kesalahan dalam pengumpulan data. Waktu pemrosesan situs *web* bervariasi, dengan beberapa yang membutuhkan waktu lebih lama daripada yang lain. Jumlah tautan maksimum diperoleh pada *website* pariwisata Kabupaten Kepulauan Selayar dan waktu pemrosesan maksimum ditemukan pada *website* sindonews yaitu 12.734375 detik. *Website* dengan jumlah tautan 0 dan akan dilakukan pembersihan data sebelum diterapakan dalam algoritma *clustering* dan klasifikasi. Berikut skrip perintah dan output hasil pembersihan data:

```python
# Membersihkan data yang mengandung nilai 0 pada kolom 'Jumlah_Tautan'
df_cleaned = df[df['jumlah_tautan'] != 0]

# Menampilkan data frame setelah pembersihan
print(df_cleaned)
```

```
                                                  url  jumlah_tautan  \
0                https://id.wikipedia.org/wiki/Pariwisata            339
1   https://id.wikipedia.org/wiki/Pariwisata_di_In...            966
2   https://kemenparekraf.go.id/rumah-difabel/Meng...             60
3                https://disbudpar.jatimprov.go.id/               51
4                https://pariwisata.jogjakota.go.id/             103
5   https://dpmptsp.karimunkab.go.id/sektor-pariwi...            119
6   https://stipram.ac.id/64-pariwisata-yang-berta...            140
7                https://dinpar.kulonprogokab.go.id/              66
8               https://www.detik.com/tag/pariwisata             126
9                https://disparbud.kebumenkab.go.id/             167
10       https://www.bisnis.com/topic/355/pariwisata             172
11     https://pariwisata.kepulauanselayarkab.go.id/             176
13  https://pariwisata.bantulkab.go.id/hal/visi-da...             82
14  https://pariamankota.go.id/pemerintahan/dinas?...             79
16  https://indonesiabaik.id/infografis/pariwisata...             73
18  https://mediakeuangan.kemenkeu.go.id/article/s...            120
20  https://pariwisata.bantulkab.go.id/hal/visi-da...             82
21  https://pariamankota.go.id/pemerintahan/dinas?...             79
23  https://indonesiabaik.id/infografis/pariwisata...             73
25  https://mediakeuangan.kemenkeu.go.id/article/s...            120

    waktu_pemrosesan
0          10.343750
1          10.390625
2          10.500000
3          10.625000
4          10.765625
5          10.906250
6          10.968750
7          11.062500
8          11.125000
9          11.156250
10         11.281250
11         11.359375
13         11.593750
14         11.609375
16         11.750000
18         11.906250
20         12.156250
21         12.296875
23         12.468750
25         12.687500
```

Gambar 1. Pembersihan Data Baris Pada Kolom Jumlah Tautan

Berikut skrip implementasi algoritma *hierarchical clustering* dengan pemilihan jumlah *cluster* menggunakan Dunn Index pada data yang telah dibersihkan:

 DOI: 10.30998/faktorexacta.v17i1.22393

```python
from scipy.cluster.hierarchy import dendrogram, linkage, fcluster
from sklearn.cluster import AgglomerativeClustering
from sklearn.metrics import pairwise_distances
import numpy as np
import matplotlib.pyplot as plt

# Pemilihan fitur untuk clustering
features = df_cleaned[[ "jumlah_tautan", "waktu_pemrosesan"]]

# Normalisasi data
features = (features - features.mean()) / features.std()

# Melakukan hierarchical clustering
linked = linkage(features, 'ward')

# Menampilkan dendrogram
dendrogram(linked, orientation='top', distance_sort='descending', show_leaf_counts=True)
plt.title('Dendrogram Hierarchical Clustering')

# Menghitung Dunn Index untuk setiap jumlah cluster
def calculate_dunn_index(data, labels):
    cluster_centers = [np.mean(data[labels == i], axis=0) for i in np.unique(labels)]
    intra_cluster_distances = np.array([np.mean(pairwise_distances(data[labels == i])) for i in np.unique(labels)])
    inter_cluster_distances = pairwise_distances(cluster_centers)

    min_inter_cluster_distance = np.min(inter_cluster_distances[inter_cluster_distances > 0])
    max_intra_cluster_distance = np.max(intra_cluster_distances)

    dunn_index = min_inter_cluster_distance / max_intra_cluster_distance
    return dunn_index

# Coba beberapa nilai jumlah cluster
dunn_values = []
for k in range(2, 6):
    # Menggunakan AgglomerativeClustering dengan jumlah cluster k
    cluster_model = AgglomerativeClustering(n_clusters=k, affinity='euclidean', linkage='ward')
    clusters = cluster_model.fit_predict(features)

    # Menghitung Dunn Index
    dunn_index = calculate_dunn_index(features, clusters)
    dunn_values.append((k, dunn_index))

    print(f"Number of clusters: {k}, Dunn Index: {dunn_index}")

# Menentukan cut-off point berdasarkan Dunn Index maksimum
optimal_k, optimal_dunn = max(dunn_values, key=lambda x: x[1])

# Menambahkan garis cut-off pada dendrogram
plt.axhline(y=optimal_dunn, color='r', linestyle='--', label=f'Optimal Cut-off (k={optimal_k})')
plt.legend()
plt.show()

```

Gambar 2. Skrip Perintah *Hierarchical Clustering*

Pada struktur data terdapat kolom yang tidak digunakan pada analisis *hierarchical clustering*, sehingga perlu dilakukan pada tahap awal adalah pemilihan fitur yang digunakan untuk *clustering* dipilih dari data frame yang telah dibersihkan, yaitu "jumlah_tautan" dan "waktu_pemrosesan" menggunakan perintah "*features = df_cleaned[["jumlah_tautan", "waktu_pemrosesan"]]*". Data dinormalisasi untuk memastikan semua fitur memiliki skala yang seragam, dengan perintah "*features = (features - features.mean()) / features.std()*". Selanjutnya melakukan *hierarchical clustering* menggunakan metode Ward dan hasil *clustering* disimpan dalam variabel *linked* dengan perintah: "*linked = linkage(features, 'ward')*". Menampilkan dendrogram hasil *clustering* dengan perintah "*dendrogram(linked, orientation='top', distance_sort='descending', show_leaf_counts=True)* dan *plt.title('Dendrogram Hierarchical Clustering')*".

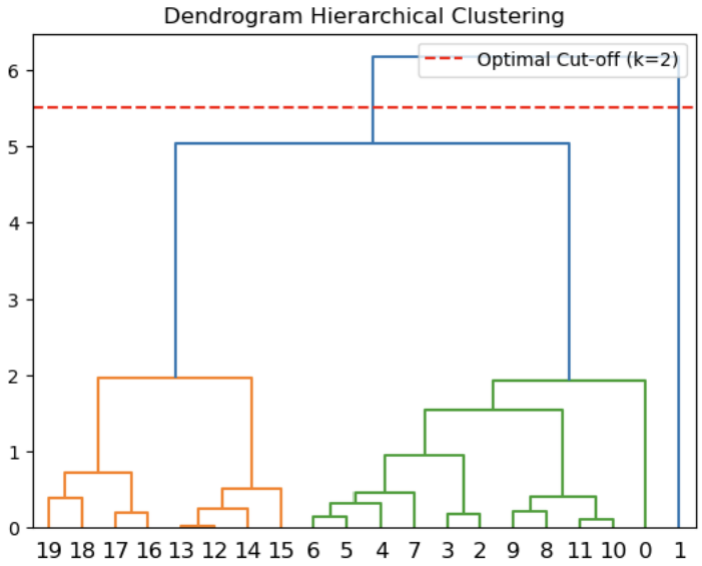
Pemilihan cluster optimal sebagai akhir visualisasi dendogram dilakukan setelah diperoleh nilai Dunn Index terbesar dari *cluster* yang dihasilkan. Perintah perhitungan Dunn Index yang terdapat dalam skrip adalah sebagai berikut:

```
dunn_values = []
for k in range(2, 6):
 cluster_model = AgglomerativeClustering(n_clusters=k, affinity='euclidean', linkage='ward')
 clusters = cluster_model.fit_predict(features)
 dunn_index = calculate_dunn_index(features, clusters)
 dunn_values.append((k, dunn_index))
 print(f"Number of clusters: {k}, Dunn Index: {dunn_index}")
```

Iterasi dilakukan dari cluster 2 hingga 5 untuk mencoba beberapa nilai jumlah cluster. Setiap kali jumlah cluster dicoba, Dunn Index dihitung menggunakan fungsi calculate_dunn_index dan hasilnya disimpan dalam dunn_values. Berikutnya menentukan cut-off point dan menambahkan garis pada dendrogram dengan perintah:

```
optimal_k, optimal_dunn = max(dunn_values, key=lambda x: x[1])
plt.axhline(y=optimal_dunn, color='r', linestyle='--', label=f'Optimal Cut-off (k={optimal_k})')
plt.legend()
plt.show()
```

Memilih jumlah cluster optimal berdasarkan Dunn Index maksimum dan menambahkan garis cut-off pada dendrogram untuk merepresentasikan hasil pemilihan jumlah cluster optimal. Dunn Index digunakan sebagai kriteria untuk mengevaluasi keberhasilan pemilihan jumlah cluster. Semakin tinggi Dunn Index, semakin baik pemisahan antar cluster. Pemilihan jumlah cluster optimal dilakukan dengan memilih nilai k yang memberikan Dunn Index maksimum. Berikut output yang diperoleh dari skrip perintah diatas:



Gambar 3. Dendogram *Hierarchical Clustering*

Cluster optimal yang diperoleh adalah sebanyak 2 kelompok, dengan anggota kelompok 1 terdiri dari 19 URL dan kelompok 2 terdiri dari 1 URL. Kelompok 1 ditandai dengan wara orange dan hijau, sedangkan kelompok 2 ditandai dengan biru yang ditunjukkan pada dendogram Gambar 3. Pembentukan kelompok tersebut berdasarkan jumlah tautan yang dibagi menjadi kelas rendah dan tinggi. Hasil pembentukan *cluster* ditambahkan dalam data frame dan akan dianalisis menggunakan algoritma *Decision Tree*. Namun, *cluster* yang dipilih adalah sebanyak 3 *cluster*, berikut tahapan pembuatan skrip dari pemrosesan data hingga evaluasi dan visualisasi model *Decision Tree* dan implementasi Python pada Gambar 4:

a) Mengimpor pustaka yang diperlukan, seperti pandas untuk manipulasi data, train_test_split untuk membagi data menjadi *set* pelatihan dan pengujian, *Decision Tree Classifier* untuk membuat model *Decision Tree*, dan fungsi-fungsi lain untuk evaluasi model dan visualisasi *Decision Tree*.
b) Data dimuat ke dalam DataFrame menggunakan pd.DataFrame. Fitur (*features*) dan target (*label*) dipisahkan.
c) Data dibagi menjadi *set* pelatihan dan pengujian menggunakan train_test_split. 80% data digunakan untuk pelatihan (X_train, y_train), dan 20% digunakan untuk pengujian (X_test, y_test).
d) Model *Decision Tree* diinisialisasi dan dilatih menggunakan data pelatihan.
e) Model diterapkan pada data pengujian untuk membuat prediksi kelas.
f) DataFrame dibuat untuk menampilkan kelas aktual dan prediksi yang dihasilkan oleh model.
g) Model dievaluasi menggunakan beberapa metrik, seperti akurasi, *confusion matrix*, dan *classification report*.
h) DataFrame prediksi ditampilkan untuk melihat kelas aktual dan prediksi.
i) Metrik evaluasi model, seperti akurasi, confusion matrix, dan *classification report*, ditampilkan.
j) *Decision Tree* yang telah dilatih dapat diplot untuk visualisasi.

 DOI: 10.30998/faktorexacta.v17i1.22393

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.tree import DecisionTreeClassifier, export_text
from sklearn import tree
import graphviz


# Pisahkan fitur (features) dan target (label)
X = df[['jumlah_tautan', 'waktu_pemrosesan']]
y = df['kelas']

# Bagi data menjadi set pelatihan (training set) dan set pengujian (test set)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Inisialisasi model Decision Tree Classifier
model = DecisionTreeClassifier()

# Latih model menggunakan set pelatihan
model.fit(X_train, y_train)

# Prediksi kelas untuk set pengujian
y_pred = model.predict(X_test)
# Buat DataFrame prediksi
df_prediksi = pd.DataFrame({'Aktual': y_test, 'Prediksi': y_pred})

# Evaluasi model
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
classification_rep = classification_report(y_test, y_pred)

# Tampilkan DataFrame prediksi
print(f'Accuracy:\n{df_prediksi}')

# Menampilkan hasil evaluasi
print(f'Confusion Matrix:\n{conf_matrix}')
print(f'Accuracy: {accuracy}')
print(f'Classification Report:\n{classification_rep}')
# Plot decision tree
plt.figure(figsize=(15, 8))
plot_tree(model, feature_names=X.columns.tolist(), class_names=['1', '2', '3'], filled=True, rounded=True, fontsize=10)
plt.show()
```

Gambar 4. Skrip Perintah *Decision Tree*

Hasil *output* pertama mencetak DataFrame yang berisi kolom "Aktual" (kelas sebenarnya) dan "Prediksi" (kelas yang diprediksi) untuk data pengujian. Bagian ini membantu melihat seberapa baik model melakukan prediksi pada data pengujian.

```
Tabel Perbandingan Prediksi dan Aktual:
    Aktual  Prediksi
0        2         2
17       1         1
15       1         1
1        3         2
```

Gambar 5. *Output* Tabel Prediksi

Pada Gambar 5 diperoleh menggunakan perintah "*print(f'Accuracy:\n{df_prediksi}')*", serta hasil pembagian data *test* yang digunakan sebagai perbandingan nilai prediksi berasal dari URL pada baris 0,17,15 dan 1. Terdapat 1 kesalahan prediksi yaitu pada baris 1, dimana nilai aktual menunjukkan kelas 3 tetapi pada kelas prediksi menghasilkan kelas 2. Apabila ditampilkan menggunakian tabel *confusion matrix* menggunakan perintah "*print(f'Confusion Matrix:\n{conf_matrix}')*" , *output* yang dihasilkan adalah seperti berikut ini:

```
Confusion Matrix:
[[2 0 0]
 [0 1 0]
 [0 1 0]]
```

Gambar 6. *Output* Tabel *Confusion Matrix*

Confusion matrix memberikan informasi tentang jumlah *true positive* (TP), *true negative* (TN), *false positive* (FP), dan *false negative* (FN). *Output* pada Gambar 6 memberikan gambaran visual tentang seberapa baik model bekerja pada setiap kelas. Sesuai dengan Tabel prediksi terdapat kesalahan prediksi pada satu URL, sehingga untuk menentukan akurasi dari model *Decision Tree* dilakukan perhitungan nilai akurasi. Akurasi adalah rasio antara prediksi yang benar (TP + TN) dengan total data. Output ini mencetak nilai akurasi dengan perintah yang digunakan adalah "*print(f'Accuracy: {accuracy}')*", berikut nilai akurasi yang dihasilkan:

```
Accuracy: 0.75
```

Gambar 7. *Output* Nilai Akurasi

Nilai akurasi yang diperoleh adalah sebesar 0.75, nilai ini memberikan informasi tentang seberapa akurat model dalam memprediksi kelas. Apabila dipresentasekan artinya sebesar 75 persen keakuratan model yang diperoleh. Selanjutnya, untuk menampilkan laporan metrik evaluasi yang lebih rinci, termasuk presisi, *recall*, dan f1-*score* untuk setiap kelas dilakukan dengan perintah "*print(f'Classification Report:\n{classification_rep}')*". Hasil yang diberikan seperti pada Gamabr 8 berikut:

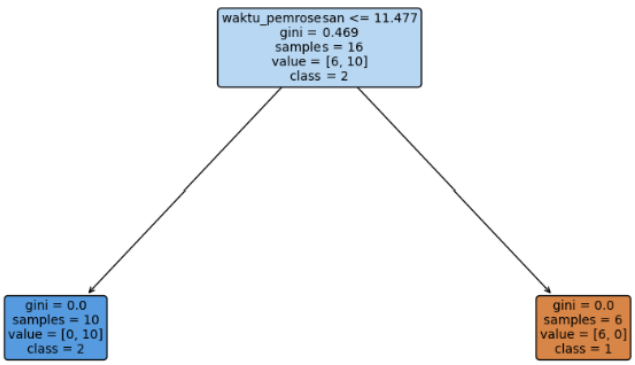
```
Classification Report:
              precision    recall  f1-score   support

           1       1.00      1.00      1.00         2
           2       0.50      1.00      0.67         1
           3       0.00      0.00      0.00         1

    accuracy                           0.75         4
   macro avg       0.50      0.67      0.56         4
weighted avg       0.62      0.75      0.67         4
```

Gambar 8. *Output Classification Report*

Berdasarkan Gambar 8 diperoleh nilai presisi (*precision*) yaitu 1 dan 0.5, *recall* (sensitivitas) yaitu 1, serta f1-*score* yaitu 1 dan 0.67. Presisi tinggi menunjukkan bahwa ketika model memprediksi suatu kelas, itu cenderung benar. *Recall* tinggi menunjukkan bahwa model dapat mendeteksi sebagian besar kelas positif. F1-*score* digunakan ketika ingin mencapai keseimbangan antara presisi dan *recall*[15]. Tahapan terakhir adalah membuat visualisasi *Decision Tree* yang memberikan visualisasi pohon keputusan yang dapat membantu interpretasi model secara keseluruhan. Perintah yang digunakan adalah sebagai berikut:

```
plt.figure(figsize=(15, 8))
plot_tree(model, feature_names=X.columns.tolist(), class_names=['1', '2', '3'], filled=True,
    rounded=True, fontsize=10)
plt.show()
```

Hasil *output* yang diberikan adalah sebagai berikut:



Gambar 9. Visualisasi *Decision Tree*

Berdasarkan Gambar 9 diperoleh bahwa model menghasilkan 2 kelas, dimana pada kelas pertama merupakan kelas berdasarkan atribut waktu pemrosesan lebih besar dari 11.477, sebaliknya untuk kelas kedua. Banyaknya URL yang masuk dalam kelas 1 yaitu 6, sedangkan kelas 2 sebanyak 10 URL.

Berdasarkan, contoh sederhana implementasi data mining algoritma clustering dan klasifikasi diatas diperoleh bahwa eksplorasi terhadap metode data mining, dengan fokus pada pengambilan data dari *google search engine* menggunakan teknik *web scraping* dilakukan untuk memahami *trend* pencarian dan preferensi pengguna terkait pariwisata. Tahapan awal mencakup pembersihan data, di mana 27 URL terkait pariwisata dikumpulkan dengan waktu pemrosesan yang bervariasi. Proses *hierarchical clustering* diterapkan dengan memilih fitur jumlah tautan dan waktu pemrosesan. Pemilihan jumlah *cluster* optimal dilakukan menggunakan Dunn Index, dan hasilnya visualisasi dalam dendrogram. Selanjutnya, data yang sudah bersih digunakan untuk melatih model *Decision Tree* dengan evaluasi menggunakan metrik akurasi, *confusion matrix*, dan *classification report*. Model *Decision Tree* menghasilkan akurasi sebesar 75%, dengan visualisasi pohon keputusan yang memberikan wawasan tentang aturan klasifikasi. Penjelasan tersebut mencakup juga pemahaman tentang keterbatasan dan etika dalam pengambilan data dari *google search engine*. Keseluruhan,

penelitian ini memberikan pandangan komprehensif tentang efektivitas teknik *web scraping* dan aplikasi data mining dalam mendapatkan wawasan dari mesin pencari Google terkait topik pariwisata.

4. PENUTUP

Penggunaan *web scraping* dalam konteks data mining, khususnya untuk algoritma klasifikasi dan clustering, memberikan kontribusi signifikan pada pemahaman dan pemanfaatan data dalam berbagai konteks. *Web scraping* memungkinkan pengumpulan data secara otomatis dari berbagai situs *web*, yang dapat digunakan untuk analisis sentimen, pembaruan berita otomatis, pemantauan harga produk, dan penelitian pasar. Pentingnya *web scraping* dalam data mining tercermin dalam kemampuannya untuk menyediakan data *real-time* atau berkala, memungkinkan model klasifikasi dan *clustering* tetap relevan dengan perubahan dalam data secara terus-menerus. Beberapa penelitian terkini juga telah berhasil menerapkan *web scraping* dalam proses data mining, seperti analisis produk di *marketplace*, pemantauan *trend hashtag* di Instagram, dan analisis sentimen pemilik dompet digital. Penelitian ini memiliki fokus pada eksplorasi berbagai teknik *web scraping* yang dapat diterapkan dalam konteks data mining dengan pendekatan berbasis Python. Selain itu, penelitian ini bertujuan untuk meningkatkan integrasi antara *web scraping* dan data mining, sehingga mendukung efektivitas analisis data. Keseluruhan, penelitian ini mencerminkan keberhasilan dan fleksibilitas *web scraping* dalam mendukung proses data mining yang dapat diakses di berbagai domain.

DAFTAR PUSTAKA

[1] S. Munzert, C. Rubba, P. Meißner, and D. Nyhuis, *Automated data collection with R: A practical guide to web scraping and text mining*, no. 1. John Wiley & Sons, 2014.

[2] V. Krotov, L. Johnson, and L. Silva, "Tutorial: Legality and Ethics of Web Scraping," *Communications of the Association for Information Systems*, vol. 47, 2020, doi: 10.17705/1CAIS.04724.

[3] V. Franzoni and A. Milani, "A Semantic Comparison of Clustering Algorithms for the Evaluation of Web-Based Similarity Measures. In: Gervasi, O., et al. Computational Science and Its Applications – ICCSA 2016. ICCSA 2016," in *Lecture Notes in Computer Science(), vol 9790*, Springer, Cham, 2016. doi: 10.1007/978-3-319-42092-9_34.

[4] D. Chrisinta, I. M. Sumertajaya, and I. Indahwati, "Evaluasi Kinerja Metode Cluster Ensemble Dan Latent Class Clustering Pada Peubah Campuran," *Indonesian Journal of Statistics and Its Applications*, vol. 4, no. 3, pp. 448–461, 2020, doi: 10.29244/ijsa.v4i3.630.

[5] D. Chrisinta, "Identifikasi Karakteristik Desa di Provinsi Bengkulu Tahun 2018 Berdasarkan Latent Class Cluster (LCC)," in *Seminar Nasional Official Statistics*, 2022, pp. 927–936. doi: 10.34123/semnasoffstat.v2022i1.1287.

[6] A. Mabrouk, R. P. D. Redondo, and M. Kayed, "Seopinion: summarization and exploration of opinion from e-commerce websites," *Sensors*, vol. 21, no. 2, p. 636, 2021, doi: 10.3390/s21020636.

[7] Z. Zuo, "Sentiment analysis of steam review datasets using naive bayes and decision tree classifier," no. 3, 2018. doi: 10.30598/barekengvol14iss3pp343-356.

[8] I. N. Husada, E. H. Fernando, H. Sagala, A. E. Budiman, and H. Toba, "Ekstraksi dan Analisis Produk di Marketplace Secara Otomatis dengan Memanfaatkan Teknologi Web Crawling," *Jurnal Teknik Informatika dan Sistem Informasi*, vol. 5, no. 3, pp. 2443–2229, 2019, doi: 10.28932/jutisi.v5i3.1977.

[9] A. Priadana and A. W. Murdiyanto, "Analisis Waktu Terbaik untuk Menerbitkan Konten di Instagram untuk Menjangkau Audiens," *Jurnal Penelitian Pers dan Komunikasi Pembangunan*, vol. 24, no. 1, pp. 59–70, 2020, doi: 10.46426/jp2kp.v24i1.118.

[10] E. Yuniar, D. Safiroh, D. Wahyuningsih, S. Informasi, S. Ppkia, and P. Paramita, "Implementasi Scrapping Data Untuk Sentiment Analysis Pengguna Dompet Digital dengan Menggunakan Algoritma Machine Learning," *janitra.orgE Yuniar, DS Utsalinah, D WahyuningsihJurnal Janitra Informatika dan Sistem Informasi, 2022•janitra.org*, vol. 2, no. 1, pp. 35–42, 2022, doi: 10.25008/janitra.v2i1.145.

[11] T. M. Fahrudin, P. A. Riyantoko, and K. M. Hindrayani, "Implementation of Web Scraping on Google Search Engine for Text Collection Into Structured 2D List," *Telematika: Jurnal Informatika dan Teknologi Informasi*, vol. 20, no. 2, pp. 139–152, 2023, doi: 10.31315/telematika.v20i2.9575.

[12] SCM de S Sirisuriya, "A comparative study on web scraping," in *Proceedings of 8th International Research Conference*, 2015, pp. 135–140.

[13] J. W. Seifert, *Data mining: An overview*. National security issues, 2004.

[14] M. F. Sanner, "Python: a programming language for software integration and development," *J Mol Graph Model*, vol. 17, no. 1, pp. 57–61, 1999, doi: 10.1016/j.str.2005.01.010.

[15] D. Chrisinta and J. E. Simarmata, "Analisis Sentimen Penilaian Masyarakat Terhadap Pejabat Publik Menggunakan Algoritma Naïve Bayes Classifier," *Komputika: Jurnal Sistem Komputer*, vol. 12, no. 1, pp. 93–101, 2023, doi: 10.34010/KOMPUTIKA.V12I1.9638.