

Laporan Tugas Besar 2 IF3170 Inteligensi Artifisial

Implementasi Algoritma Pembelajaran Mesin



Kelompok AIKamuMah:

| | |
|-----------------------------|----------|
| Andi Farhan Hidayat | 13523128 |
| Ahmad Syafiq | 13523135 |
| Andri Nurdianto | 13523145 |
| Rafael Marchel D. W. | 13523146 |
| Muhammad Kinan Arkansyaddad | 13523152 |

Teknik Informatika
Institut Teknologi Bandung

Daftar Isi

| | |
|--|-----------|
| 1. Implementasi Decision Tree Learning..... | 3 |
| 2. Implementasi Logistic Regression..... | 5 |
| 3. Implementasi SVM..... | 8 |
| 4. Cleaning dan Preprocessing..... | 9 |
| 5. Perbandingan Hasil..... | 15 |
| 5.1. Decision Tree Learning..... | 15 |
| 5.2. Logistic Regression..... | 15 |
| 5.3. SVM..... | 15 |
| 6. Kontribusi..... | 15 |
| 7. Referensi..... | 15 |

1. Implementasi Decision Tree Learning

Decision Tree adalah model supervised learning yang memprediksi label kelas dengan membangun struktur pohon keputusan berdasarkan fitur-fitur input. Setiap node internal merepresentasikan tes pada sebuah fitur, cabang merepresentasikan hasil dari tes tersebut, dan setiap leaf node merepresentasikan label kelas prediksi. Decision tree bekerja dengan membagi data secara rekursif berdasarkan fitur yang memberikan pemisahan terbaik.

Implementasi Decision Tree Learning mendukung tiga algoritma splitting yang dapat dipilih melalui parameter *algorithm*:

A. ID3 (Information Gain)

Algoritma ini menggunakan Entropy sebagai ukuran ketidakmurnian (impurity). Entropy mengukur tingkat ketidakteraturan dalam kumpulan data:

$$H(S) = - \sum_{i=1}^c p_i \log_2(p_i)$$

Dengan p_i adalah proporsi sampel kelas i dalam dataset S , dan c adalah jumlah kelas.

Information Gain mengukur pengurangan entropy setelah melakukan split:

$$IG(S, A) = H(S) - \sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} H(S_v)$$

Fitur dengan Information Gain tertinggi dipilih untuk split.

B. C4.5 (Gain Ratio)

Algoritma C4.5 menggunakan Gain Ratio untuk mengatasi bias Information Gain terhadap fitur dengan banyak nilai unik. Gain Ratio menormalisasi Information Gain dengan Split Information:

$$\text{SplitInfo}(S, A) = - \sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} \log_2 \left(\frac{|S_v|}{|S|} \right)$$

$$\text{GainRatio}(S, A) = \frac{IG(S, A)}{\text{SplitInfo}(S, A)}$$

Pendekatan ini memberikan preferensi yang lebih seimbang terhadap fitur dengan jumlah kategori yang moderat.

C. CART (Gini Index)

CART menggunakan Gini Impurity sebagai ukuran ketidakmurnian:

$$\text{Gini}(S) = 1 - \sum_{i=1}^c p_i^2$$

Gini Gain dihitung sebagai pengurangan Gini impurity setelah split:

$$\text{GiniGain}(S, A) = \text{Gini}(S) - \sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} \text{Gini}(S_v)$$

Gini Index cenderung lebih efisien secara komputasi karena tidak menggunakan operasi logaritma.

Implementasi DTL menangani missing values dengan strategi majority branch assignment. Saat training:

- Hanya sampel dengan nilai fitur yang diketahui yang digunakan untuk menghitung gain.
- Sampel dengan missing values diarahkan ke cabang yang memiliki sampel lebih banyak (majority branch).

Saat prediksi, jika value fitur missing, observasi diarahkan ke cabang yang sama dengan yang digunakan saat training (left_branch_majority).

Untuk mencegah overfitting, kami mengimplementasikan Cost-Complexity Pruning (CCP), juga dikenal sebagai weakest link pruning. Metode ini menambahkan penalti kompleksitas ke fungsi cost:

$$R_\alpha(T) = R(T) + \alpha|T|$$

Pendekatan ini memastikan tree yang dihasilkan memiliki keseimbangan optimal antara akurasi dan generalisasi sesuai dengan nilai `ccp_alpha` yang ditentukan.

2. Implementasi Logistic Regression

Logistic Regression adalah model statistik yang digunakan untuk memprediksi probabilitas kejadian suatu kelas biner. Berbeda dengan regresi linear yang memprediksi nilai kontinu, Logistic Regression memetakan keluaran linear ke dalam rentang probabilitas $[0,1]$ menggunakan fungsi Sigmoid (atau Logistik).

Secara matematis, untuk sebuah sampel data x dengan bobot w dan bias b , skor linear z dihitung sebagai:

$$z = w^T x + b$$

Skor ini kemudian dipetakan menjadi probabilitas μ menggunakan fungsi Sigmoid:

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

Jika nilai $\mu \geq 0.5$, sampel diklasifikasikan ke dalam kelas positif (1), dan sebaliknya ke kelas negatif (0).

Dataset yang digunakan memiliki variabel target dengan tiga kategori (Dropout, Enrolled, Graduate). Karena Logistic Regression standar hanya bekerja pada klasifikasi biner, kami mengimplementasikan dua strategi untuk menangani masalah multikelas ini, yaitu One-vs-Rest (OvR) dan Multinomial (Softmax) Regression.

Strategi One-vs-Rest atau One-vs-All memecah masalah klasifikasi K kelas menjadi K masalah klasifikasi biner yang independen. Pendekatan ini dipilih berdasarkan studi oleh Rifkin & Klautau (2004) yang menyatakan bahwa skema ini memiliki akurasi yang setara dengan pendekatan multikelas lainnya jika pengklasifikasi biner dasarnya dioptimasi dengan baik. Rifkin & Klautau (2004) juga menyimpulkan bahwa One-vs-Rest sebaiknya menjadi pendekatan untuk klasifikasi

multikelas karena kesederhanaan konseptual dan kemudahan implementasinya dibandingkan metode yang lebih kompleks. Dalam implementasi kami pada metode `_fit_ovr`:

1. Model melatih $K = 3$ klasifier biner secara terpisah
2. Untuk setiap kelas target (misalnya Dropout), label data diubah menjadi biner: 1 untuk Dropout dan 0 untuk sisanya (Enrolled dan Graduate).
3. Setiap model memiliki parameter bobot w dan bias b masing-masing.

Untuk melatih model, kami menggunakan algoritma Stochastic Gradient Ascent (SGA) sesuai yang diajarkan di perkuliahan. Tujuan optimasi adalah memaksimalkan Log-Likelihood (setara meminimalkan Log-Loss).

Langkah algoritma:

1. Inisialisasi bobot w dan bias b dengan 0.
2. Lakukan iterasi sebanyak T epoch (iterasi).
3. Pada setiap epoch, urutan data diacak untuk menjamin sifat stokastik.
4. Pembaruan bobot setiap iterasi data:

$$w \leftarrow w + \eta \cdot (y^{(i)} - p^{(i)}) \cdot x^{(i)}$$

Untuk melakukan prediksi, input x dimasukkan ke dalam ketiga model tersebut. Kelas prediksi diambil berdasarkan model yang memberikan probabilitas tertinggi (*argmax*):

$$\hat{y} = \arg \max_{k \in \{1, \dots, K\}} (\sigma(w_k^T x + b_k))$$

Sebagai alternatif, kami juga mengimplementasikan strategi Multinomial (Softmax) pada metode `_fit_softmax`. Berbeda dengan OvR yang melatih model secara terpisah, strategi ini melatih satu model gabungan yang memprediksi distribusi probabilitas untuk seluruh kelas sekaligus.

Fungsi Sigmoid digantikan oleh fungsi Softmax, yang menjamin bahwa total probabilitas seluruh kelas berjumlah 1. Fungsi Softmax adalah generalisasi dari fungsi Sigmoid untuk kasus multikelas target, yang

memetakan vektor skor linear z menjadi distribusi probabilitas yang valid (total probabilitas = 1).

$$P(y = k|x) = \frac{e^{z_k}}{\sum_{j=1}^K e^{z_j}}$$

Implementasi kami menerapkan teknik shift-invariance

$$\text{Softmax}(z)_k = \frac{e^{z_k - \max(z)}}{\sum_{j=1}^K e^{z_j - \max(z)}}$$

untuk mencegah numerical overflow saat menghitung eksponensial dari nilai fitur yang besar. Pembuktian bahwa fungsi Softmax dengan shift-invariance ekuivalen dengan fungsi Softmax dapat dilihat di sini: <https://blester125.com/blog/softmax.html>

Secara algoritmik, proses pelatihan ini juga dilakukan menggunakan Stochastic Gradient Ascent dengan langkah-langkah sebagai berikut:

1. Pada setiap iterasi, satu sampel (x,y) diambil secara acak dari dataset untuk menjamin sifat stokastik.
2. Label kategori y dikonversi menjadi vektor one-hot encoding (y_{onehot}) untuk memfasilitasi perhitungan error terhadap distribusi probabilitas.
3. Model menghitung prediksi probabilitas p menggunakan fungsi Softmax.
4. Gradien error dihitung sebagai selisih antara label one-hot asli dan probabilitas prediksi ($\text{error} = y_{\text{onehot}} - p$).
5. Matriks bobot W diperbarui menggunakan hasil outer product antara vektor input x dan vektor error, dikalikan dengan learning rate η .

Untuk mencegah overfitting, kami menerapkan regularisasi L1 dan L2 yang menambahkan term penalti pada fungsi objektif untuk membatasi kompleksitas model. L2 Regularization (Ridge) menambahkan penalti berupa kuadrat magnitudo bobot ke fungsi loss utama. Hal ini mendorong bobot agar tetap kecil tetapi tidak nol.

$$w \leftarrow w + \eta (\text{Error} \cdot x - \alpha w)$$

L1 Regularization menambahkan penalti berupa jumlah nilai absolut bobot ke fungsi loss. Berbeda dengan L2, L1 cenderung menghasilkan

solusi yang sparse, banyak bobot didorong menjadi tepat nol. Hal ini sangat berguna sebagai metode *feature selection* karena model akan "mematikan" fitur yang tidak relevan.

$$w \leftarrow w + \eta (\text{Error} \cdot x - \alpha \cdot \text{sign}(w))$$

3. Implementasi SVM

Support Vector Machine (SVM) adalah salah satu algoritma *machine learning*. SVM memanfaatkan *hyperplane* sebagai metode klasifikasinya. Titik-titik data dari kelas dipisahkan oleh *hyperplane* pada ruang dimensi N . *Hyperplane* dicari dengan memaksimalkan margin dari 2 titik terdekat dengan kelas yang berbeda. Dua titik terdekat yang memiliki kelas berbeda disebut sebagai *support vectors* (vektor pendukung).

Secara matematis, bidang pemisah ditulis sebagai:

$$w \cdot x + b = 0 \text{ (hyperplane optimal)}$$

$$w \cdot x + b = 1 \text{ (batas margin untuk kelas 1)}$$

$$w \cdot x + b = -1 \text{ (batas margin untuk kelas -1)}$$

Lalu, untuk menentukan kelas dari titik baru dituliskan sebagai berikut:

$$f(x) = w \cdot x + b$$

$$f(x) > 0 \text{ (titik masuk ke kelas 1)}$$

$$f(x) < 0 \text{ (titik masuk ke kelas -1)}$$

Dataset yang digunakan memiliki variabel target dengan tiga kategori (Dropout, Enrolled, Graduate). Dalam hal ini, strategi *one-vs-all* digunakan karena proses perbandingan hanya dilakukan sebanyak n kali sesuai dengan jumlah kelas yang ada sesuai yang ditulis oleh.

Untuk implementasi, kami terlebih dahulu hal-hal sebagai berikut dalam fungsi fit:

1. Model melatih $K=3$ *classifier* berbeda
2. Untuk setiap kelas target (misal Dropout), label bernilai 1 jika label yang dicek sesuai dengan kelas saat ini dan menjadi -1 jika label tidak sesuai dengan kelas saat ini (selain Dropout)
3. Model memiliki parameter bobot w dan bias b

Langkah-langkah penyelesaian algoritma SVM dengan menggunakan *one-vs-all* dijabarkan sebagai berikut:

1. Inisiasi bobot w dan b dengan nilai 0
2. Lakukan iterasi sebanyak n_iters epoch (iterasi).
3. Pengecekan fitur menggunakan $f(x) = w \cdot x + b$ dan memeriksa margin hasil perhitungan.
4. Pembaruan bobot. Jika kondisi margin terpenuhi, w diubah. Jika kondisi margin tidak terpenuhi, w dan b diubah.

4. Cleaning dan Preprocessing

Untuk mendapatkan hasil prediksi yang optimal, dataset perlu disiapkan dengan melakukan *data cleaning* dan *preprocessing*.

4.1. Data Cleaning

Data cleaning atau pembersihan data adalah proses penting dalam siklus analisis data untuk memastikan data yang digunakan akurat, konsisten, dan relevan. Data yang belum dibersihkan dapat menyebabkan kesalahan dalam analisis dan modeling, mengurangi keakuratan hasil, dan berdampak buruk pada pengambilan keputusan. Proses ini mencakup identifikasi, penanganan, dan penghapusan data yang tidak valid, duplikat, tidak lengkap, atau tidak relevan.

Pertama, dilakukan identifikasi *missing data*, *duplicate data*, dan *outlier*. Berdasarkan hasil yang didapat, ditemukan bahwa tidak ada *missing* ataupun *duplicate data*, namun terdapat *data outlier* dan beberapa kolom memiliki *skewness* yang tinggi

```
df.isna().sum()
✓ 0.0s

Mother's qualification      0
Father's qualification     0
Mother's occupation        0
Father's occupation        0
Admission grade            0
Displaced                  0
Educational special needs  0
Debtor                     0
Tuition fees up to date    0
Gender                     0
Scholarship holder         0
Age at enrollment          0
International               0
Curricular units 1st sem (credited)  0
Curricular units 1st sem (enrolled)  0
Curricular units 1st sem (evaluations)  0
Curricular units 1st sem (approved)  0
Curricular units 1st sem (grade)      0
Curricular units 1st sem (without evaluations)  0
Curricular units 2nd sem (credited)    0
Curricular units 2nd sem (enrolled)    0
Curricular units 2nd sem (evaluations)  0
Curricular units 2nd sem (approved)    0
Curricular units 2nd sem (grade)       0
Curricular units 2nd sem (without evaluations)  0
Unemployment rate          0
Inflation rate              0
GDP                         0
Target                      0
dtype: int64
```

Gambar 4.1. Pengecekan *missing data*

```
# Tidak ada data duplikat
df.duplicated().sum()
✓ 0.0s

np.int64(0)
```

Gambar 4.2. Pengecekan *duplicate data*

```
skewness_scores = df[numerical_cols].skew().sort_values(ascending=False)

print("Koefisien Kemiringan (Skewness):")
print(skewness_scores)
```

✓ 0.0s

Koefisien Kemiringan (Skewness):

| | |
|--|-----------|
| Nationality | 11.388826 |
| Educational special needs | 8.986992 |
| Curricular units 2nd sem (without evaluations) | 7.767834 |
| Curricular units 1st sem (without evaluations) | 7.748450 |
| International | 6.329504 |
| Curricular units 2nd sem (credited) | 4.647403 |
| Curricular units 1st sem (credited) | 4.188695 |
| Age at enrollment | 2.081474 |
| Curricular units 1st sem (enrolled) | 1.588180 |
| Curricular units 1st sem (evaluations) | 0.886651 |
| Curricular units 1st sem (approved) | 0.772061 |
| Curricular units 2nd sem (enrolled) | 0.713280 |
| Admission grade | 0.529363 |
| Curricular units 2nd sem (evaluations) | 0.334332 |
| Previous qualification (grade) | 0.299831 |
| Curricular units 2nd sem (approved) | 0.299546 |
| Inflation rate | 0.273135 |
| Unemployment rate | 0.212522 |
| Curricular units 2nd sem (grade) | -1.283620 |
| Curricular units 1st sem (grade) | -1.548017 |

dtype: float64

Gambar 4.3. Skewness tiap kolom

Selanjutnya, dilakukan *feature engineering* dengan menambahkan fitur-fitur yang informatif dan relevan. Berikut adalah fitur yang ditambahkan beserta alasannya

| Nama Fitur | Deskripsi | Alasan |
|------------------------|--|--|
| Success_Rate_1st_Sem | Rasio sks lulus dibanding jumlah ujian | Semakin tinggi maka semakin pintar siswa dalam mengerjakan ujian (karena banyak lulusnya) |
| Load_vs_Credit_1st_Sem | Rasio sks kredit dibanding jumlah sks yang diambil | Semakin tinggi maka sks yang diambil kebanyakan merupakan sks kredit sehingga mengurangi beban siswa |

| | | |
|-------------------|--|---|
| Grade_Improvement | Selisih nilai semester 2 dengan semester 1 | Menunjukkan peningkatan/penurunan nilai yang akan condong ke graduate/dropout |
|-------------------|--|---|

Setelah itu, dimulai tahap preprocessing yang dilakukan dengan kode berikut

```

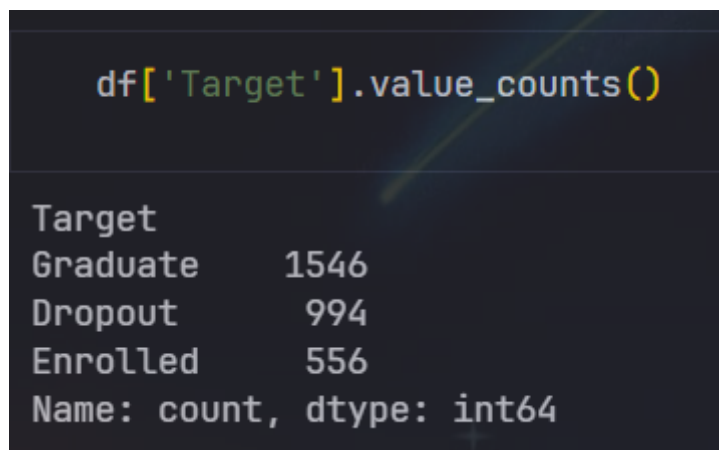
1 cols_to_drop = [
2     'Educational special needs',
3     'International',
4     'Nationality',
5 ]
6
7 numerical_cols = [
8     'Previous qualification (grade)',
9     'Admission grade',
10    'Curricular units 1st sem (grade)',
11    'Curricular units 2nd sem (grade)',
12    'Age at enrollment',
13    'Curricular units 1st sem (credited)',
14    'Curricular units 1st sem (enrolled)',
15    'Curricular units 1st sem (evaluations)',
16    'Curricular units 1st sem (approved)',
17    'Curricular units 1st sem (without evaluations)',
18    'Curricular units 2nd sem (credited)',
19    'Curricular units 2nd sem (enrolled)',
20    'Curricular units 2nd sem (evaluations)',
21    'Curricular units 2nd sem (approved)',
22    'Curricular units 2nd sem (without evaluations)',
23    'Unemployment rate',
24    'Inflation rate',
25    'Success_Rate_1st_Sem',
26    'Load_vs_Credit_1st_Sem',
27    'Grade_Improvement',
28 ]
29
30 log_cols = [
31     'Curricular units 2nd sem (without evaluations)',
32     'Curricular units 1st sem (without evaluations)',
33     'Curricular units 2nd sem (credited)',
34     'Curricular units 1st sem (credited)',
35 ]
36
37 std_cols = [col for col in numerical_cols if col not in log_cols]
38
39 categorical_cols = [
40     'Marital status',
41     'Application mode',
42     'Application order',
43     'Course',
44     'Mother\'s qualification',
45     'Father\'s qualification',
46     'Mother\'s occupation',
47     'Father\'s occupation',
48     'Previous qualification',
49     '# Kolom Biner',
50     'Daytime/evening attendance\t',
51     'Displaced',
52     'Debtor', 'Tuition fees up to date',
53     'Gender',
54     'Scholarship holder',
55 ]
56
57 # Numerical pipeline
58 numerical_pipeline = Pipeline([
59     ('scaler', StandardScaler())
60 ])
61
62 # Categorical pipeline
63 categorical_pipeline = Pipeline([
64     ('encoder', OneHotEncoder(drop='first', handle_unknown='ignore', sparse_output=False))
65 ])
66
67 # Combine Pipeline
68 preprocessor = ColumnTransformer([
69     ('log', log_scaler, log_cols),
70     ('num', numerical_pipeline, numerical_cols),
71     ('cat', categorical_pipeline, categorical_cols),
72 ], remainder='drop', sparse_threshold=0) # Drop column
73
74 final_pipeline = Pipeline([
75     ('preprocessor', preprocessor),
76 ])
77

```

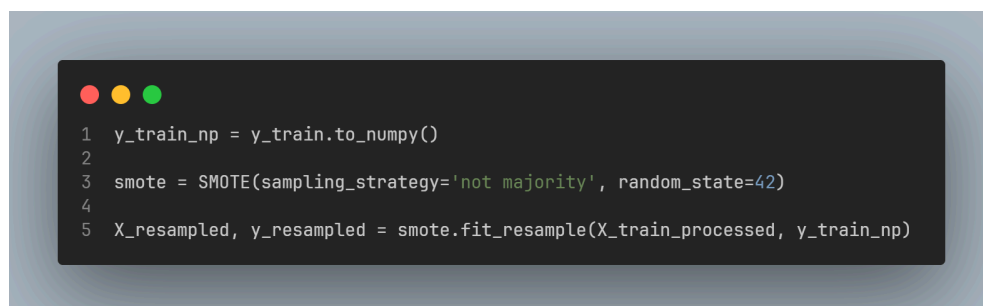
Gambar 4.4. *Preprocessing pipeline*

Tahap preprocessing dilakukan dengan menggunakan pipeline yang berbeda untuk kolom numerik dan kolom kategorikal. Pada kolom kategorikal, dilakukan *one-hot encoding* untuk menghilangkan bias ordinal terhadap data yang seharusnya nominal. Sedangkan, dilakukan *standard scaling* untuk memastikan tiap kolom memiliki rata-rata nol dan deviasi standar satu. Selain itu, ditemukan bahwa beberapa kolom memiliki kemiringan (*skewness*) yang sangat tinggi, sehingga dilakukan log-transformation untuk mengurangi kemiringan ekstrem data, menjadikannya lebih mendekati distribusi normal sebelum distandarisasi.

Selanjutnya, dilakukan teknik oversampling karena proporsi dataset tidak seimbang, di mana jumlah data yang berlabel Graduate lebih tinggi dibandingkan dengan jumlah data yang berlabel Enrolled dan Dropout. Teknik oversampling dilakukan dengan menggunakan library imblearn SMOTE



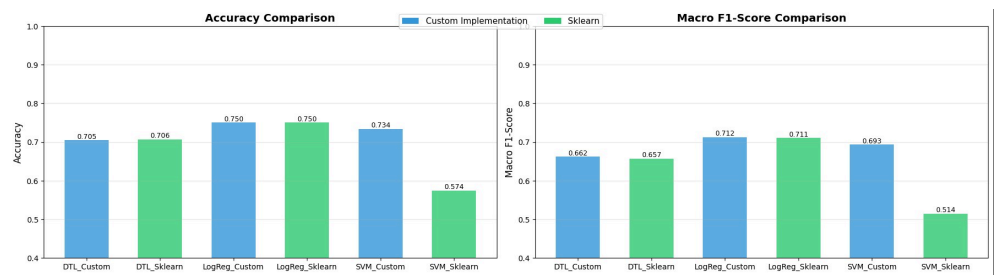
Gambar 4.5. Proporsi label data



Gambar 4.6. Teknik Oversampling dengan SMOTE

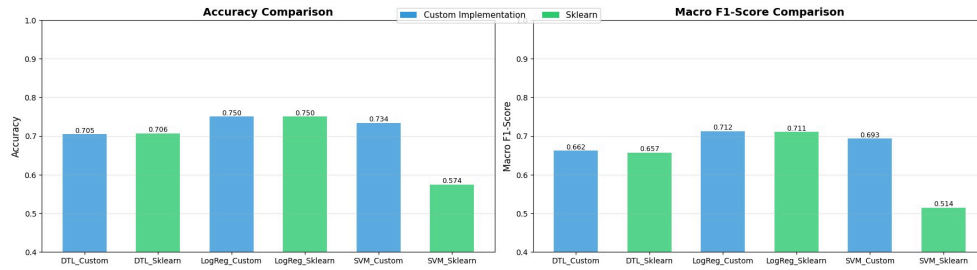
5. Perbandingan Hasil

5.1. Decision Tree Learning



| | | |
|--|---|--|
| <div>Sklearn Accuracy: 0.6935 Macro F1: 0.6501</div> | <div>Custom Implementaton ID3 Accuracy: 0.7048 Macro F1: 0.6620</div> | |
| <div>C4.5 Custom Implementation (C4.5): Accuracy: 0.7419 Macro F1: 0.6598</div> <div>Sklearn Implementation (criterion='entropy', ccp_alpha=0.00): Accuracy: 0.6935 Macro F1: 0.6501</div> | | |
| | | |
| | | |
| | | |

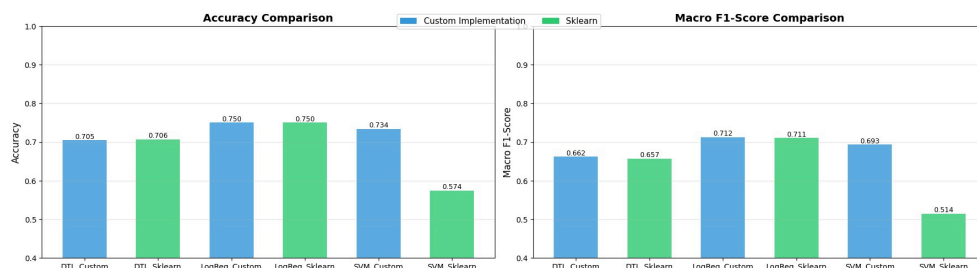
5.2. Logistic Regression



| Skor | Custom Implementation | Sklearn Implementation |
|------------|-----------------------|------------------------|
| Accuracy: | 0.7500 | 0.7500 |
| Macro F1: | 0.7121 | 0.7105 |
| Precision: | 0.7135 | 0.7110 |
| Recall: | 0.7219 | 0.7203 |

Perbandingan hasil menunjukkan bahwa implementasi logistic regression from scratch dan versi sklearn memberikan performa yang hampir identik pada data validasi. Akurasi kedua model sama, sementara perbedaan pada metrik F1, precision, dan recall sangat kecil dan tidak signifikan secara praktis. Hal ini mengindikasikan bahwa algoritma yang dibuat dari scratch telah berhasil mereplikasi perilaku model sklearn dengan baik, baik dalam proses pembelajaran maupun kemampuan generalisasi. Dengan demikian, implementasi kustom dapat dianggap valid dan konsisten terhadap standar library yang sudah teruji.

5.3. SVM



| | Custom Implementation | Sklearn Implementation |
|------------|-----------------------|------------------------|
| Accuracy: | 0.7339 | 0.5742 |
| Macro F1: | 0.6931 | 0.5144 |
| Precision: | 0.4014 | 0.5177 |
| Recall: | 0.6998 | 0.5161 |

SVM custom menambahkan perhitungan dengan weight dan bias menggunakan gradient descent ketika terdapat margin <1 . SVM pada sklearn menggunakan one-vs-one sedangkan pada SVM custom menggunakan one-vs-all sehingga SVM custom dapat membuat hyperplane dengan pola dominan pada data tersebut. Namun, hal itu memiliki kelemahan yaitu kurang stabil.

6. Kontribusi

| NIM | Nama | Kontribusi |
|----------|-----------------------|---------------------------------|
| 13523128 | Andi Farhan Hidayat | Model Optimization & Comparison |
| 13523135 | Ahmad Syafiq | Data Preprocessing & Cleaning |
| 13523145 | Andri Nurdianto | SVM |
| 13523146 | Rafael Marchel D. W. | DTL |
| 13523152 | M. Kinan Arkansyaddad | Logistic Regression |

7. Referensi

Jurafsky, D., & Martin, J. H. (2025). Speech and Language Processing (3rd ed. draft). Chapter 4: Logistic Regression and Text Classification. Diakses dari <https://web.stanford.edu/~jurafsky/slp3/4.pdf>

Rifkin, R., & Klautau, A. (2004). In defense of one-vs-all classification. Journal of Machine Learning Research, 5(Jan). Diakses dari <https://www.jmlr.org/papers/volume5/rifkin04a/rifkin04a.pdf>

<https://blester125.com/blog/softmax.html>

Martin, Eric (2025) . Multiclass Classification Using Support Vector Machines. Diakses dari <https://www.baeldung.com/cs/svm-multiclass-classification>

Deliverables:

- Folder **src**, digunakan untuk menyimpan source code
- Folder **doc**, digunakan untuk menyimpan laporan dalam bentuk **.pdf** yang terdiri atas komponen berikut:
 - Cover
 - Penjelasan singkat implementasi Decision Tree Learning.
 - Penjelasan singkat implementasi Logistic Regression.
 - Penjelasan singkat implementasi SVM.
 - Penjelasan tahap cleaning dan preprocessing yang dilakukan beserta dengan alasannya.
 - Perbandingan hasil prediksi dari algoritma yang diimplementasikan dengan hasil yang didapatkan dengan menggunakan pustaka. Jelaskan insight yang kalian dapatkan dari perbandingan tersebut.
 - Perbandingan hasil dapat menggunakan metrics yang sesuai dengan permasalahan yang ada.

5 of 6

- Kontribusi setiap anggota dalam kelompok.
- Referensi
- **README.md**, yang berisi deskripsi singkat repository, cara setup dan run program, dan pembagian tugas tiap anggota kelompok.

