Tugas Besar 2 IF3170 Inteligensi Artifisial Implementasi Algoritma Pembelajaran Mesin

Laporan Tugas Besar 2

IF3170 Inteligensi Artifisial

Disusun Oleh:

Kelompok 2 – Senku Intelligence

13523131 Ahmad Wafi Idzharulhaqq

13523143 Amira Izani

13523147 Frederiko Eldad Mugiyono

13523157 Natalia Desiany Nursimin

13523160 I Made Wiweka Putera

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2025

Daftar Isi

Implementasi Decision Tree Learning

1. Struktur Kelas dan Inisialisasi

Kelas utama DecisionTreeScratch dirancang dengan parameter berikut:

a. criterion: Untuk mendukung Gini Impurity (default) dan Entropy (Information Gain).
b. max_depth: Membatasi kedalaman maksimum pohon untuk mencegah overfitting.
c. min_samples_split & min_samples_leaf: Mengatur batasan jumlah sampel minimum untuk melakukan pemecahan node dan pada daun.
d. class_weight: Menangani ketidakseimbangan kelas (imbalanced data) dengan memberikan bobot lebih pada kelas minoritas.
e. min_impurity_decrease: Ambang batas minimum gain yang diperlukan untuk melakukan split.
f. max_features: Mengontrol jumlah fitur yang dipertimbangkan saat mencari best split.
g. random_state: Mengatur pengacakan seed agar hasil eksperimen tetap sama saat dijalankan ulang.

2. Kriteria Pemecahan (Splitting Criterion)

Algoritma memilih pemecahan terbaik dengan meminimalkan impurity (ketidakmurnian) pada child nodes.

a. Gini Impurity: Digunakan untuk mengukur seberapa sering elemen yang dipilih secara acak akan salah label jika dilabeli secara acak sesuai distribusi label di set tersebut.
b. Entropy: Mengukur ketidakpastian informasi dalam node. Nilai entropy tinggi menunjukkan ketidakpastian tinggi, sedangkan entropy 0 menunjukkan node murni (pure).
c. Penanganan Class Weights: Implementasi kami memperhitungkan sample_weights dalam perhitungan Gini dan Entropy. Hal tersebut krusial untuk dataset yang tidak seimbang, di mana frekuensi kelas minoritas

dikalikan dengan bobotnya sehingga berkontribusi setara dengan kelas mayoritas.

3. Proses Pembangunan Pohon (Tree Building)

Metode _build_tree membangun pohon secara rekursif:

a. Pengecekan Kondisi Berhenti (Stopping Conditions):
   i. Kedalaman maksimum tercapai.
   ii. Simpul murni (hanya 1 kelas n_classes == 1).
   iii. Jumlah sampel kurang dari min_samples_split.
   iv. Jumlah sampel kurang dari 2 x min_samples_leaf.
b. Pencarian Split Terbaik (_best_split):
   i. Iterasi melalui subset fitur (dapat diatur via max_features untuk variasi seperti Random Forest).
   ii. Untuk fitur kontinu, kami menggunakan optimasi Midpoint Thresholds: alih-alih mengecek setiap nilai unik, kami mengecek nilai tengah antar nilai unik yang terurut, atau menggunakan persentil jika nilai unik terlalu banyak (>30) untuk efisiensi komputasi.
   iii. Gain dihitung sebagai selisih antara impurity induk dan rata-rata tertimbang impurity anak.
c. Pembentukan Node:
   i. Jika gain terbaik memenuhi min_impurity_decrease, node internal dibuat, dan data dipecah menjadi kiri (<= threshold) dan kanan (> threshold).
   ii. Jika tidak, node tersebut menjadi Leaf Node (daun) yang menyimpan prediksi kelas mayoritas.

4. Penanganan Tipe Fitur

a. Fitur Biner: Diperlakukan dengan threshold 0.5.
b. Fitur Kontinu: Menggunakan logika thresholding dinamis seperti dijelaskan di atas.
c. Missing Values: Jika nilai fitur hilang (NaN) saat prediksi, node akan mengembalikan majority class dari node tersebut sebagai fallback.

5. **Pasca-Pemangkasan (Post-Pruning)**

   Untuk mengatasi overfitting, kami mengimplementasikan Reduced Error Pruning (REP) dalam metode _prune_tree.

   a. Proses: Pohon dibangun penuh terlebih dahulu.
   b. Evaluasi: Algoritma menelusuri pohon dari bawah ke atas (bottom-up).
   c. Keputusan: Untuk setiap sub-pohon, algoritma membandingkan akurasi validasi antara:
      i. Membiarkan sub-pohon tetap ada.
      ii. Mengganti sub-pohon dengan satu leaf node (berisi kelas mayoritas).
   d. Jika penggantian menjadi leaf tidak menurunkan akurasi pada validation set, maka pemangkasan dilakukan.
   e. Hasil: Model yang lebih sederhana dengan generalisasi yang lebih baik.

6. **Mekanisme Prediksi**

   a. predict: Menelusuri pohon dari akar hingga daun untuk setiap sampel. Pada setiap node internal, fitur sampel dibandingkan dengan threshold. Arah penelusuran (kiri/kanan) ditentukan oleh hasil perbandingan.
   b. predict_proba: Mengembalikan probabilitas kelas berdasarkan distribusi kelas (class_counts) yang tersimpan di daun tempat sampel berakhir.

7. **Fitur Tambahan**

   a. Feature Importance: Dihitung berdasarkan total penurunan impurity (gain) yang dikontribusikan oleh setiap fitur di seluruh node pohon, dinormalisasi hingga totalnya 1.0. Hal ini bermanfaat untuk menunjukkan tingkat pengaruh setiap fitur terhadap keputusan model. Semakin besar nilai feature importance, semakin penting peran fitur tersebut, sedangkan fitur dengan nilai kecil dapat dihilangkan untuk menyederhanakan model dan mengurangi overfitting.
   b. Visualisasi Tree: Pembuatan visualisasi tree dilakukan melalui dua fungsi, yaitu visualize_tree() dan visualize_top_branches(). Fungsi visualize_tree() digunakan untuk menampilkan struktur pohon hingga kedalaman tertentu dalam bentuk diagram hierarkis, sedangkan visualize_top_branches()

digunakan untuk menampilkan daftar pemisahan (split) paling penting berdasarkan nilai gain tertinggi. Pada visualize_tree(), proses visualisasi dilakukan dengan memotong pohon hingga max_depth, kemudian setiap node digambarkan sebagai kotak menggunakan matplotlib dan setiap hubungan parent–child akan dihubungkan dengan garis. Sedangkan, visualize_top_branches() bekerja dengan menelusuri seluruh node pohon, mengumpulkan node non-leaf, kemudian mengurutkannya berdasarkan nilai gain. Seluruh informasi akan kemudian ditampilkan dalam bentuk tabel yang berisi peringkat, fitur, nilai threshold, gain, jumlah sampel dan kedalaman node. Hasilnya akan disimpan dalam bentuk PNG.

Implementasi Logistic Regression

1. **Struktur Kelas dan Inisialisasi**

   Kelas ini mengimplementasikan regresi logistik standar dengan optimasi Gradient Descent dan regularisasi L2. Parameter-parameter kelas ini sebagai berikut:

   a. learning_rate: Laju pembelajaran awal (eta_0).
   b. n_iterations: Jumlah epoch pelatihan.
   c. batch_size: Ukuran mini-batch untuk Stochastic Gradient Descent (SGD).
   d. lambda_reg: Kekuatan regularisasi L2 (lambda) untuk mencegah overfitting.
   e. decay_rate: Parameter untuk mengurangi learning rate seiring waktu.

2. **Landasan Matematika**

   a. Fungsi Aktivasi (Sigmoid): Kami menggunakan fungsi matematika bernama Sigmoid yang bentuknya seperti huruf 'S'. Fungsi ini bertugas mengubah skor mentah (berapapun nilainya) menjadi sebuah probabilitas antara 0% sampai 100%.
      i. Jika skor sangat positif, probabilitas mendekati 100%.
      ii. Jika skor sangat negatif, probabilitas mendekati 0%.
   b. Fungsi Kerugian (Loss Function): Kami menggunakan metode Binary Cross-Entropy:
      i. Jika model yakin 90% itu "Dropout", dan ternyata benar, kesalahannya kecil.
      ii. Jika model yakin 90% itu "Dropout", tapi ternyata "Lulus", kesalahannya sangat besar.
      iii. Tujuan pelatihan adalah membuat nilai kesalahan ini sekecil mungkin.
   c. Cara Model Belajar (Gradient Descent): Proses belajar dilakukan dengan cara memperbarui "bobot" (pengetahuan model) sedikit demi sedikit.
      i. Model menebak jawaban.
      ii. Dihitung seberapa jauh tebakannya meleset.
      iii. Bobot model digeser ke arah yang mengurangi kesalahan tersebut.
         1. Jika tebakan meleset jauh, pergeserannya besar.

2. Jika tebakan hampir benar, pergeserannya kecil.

3. Proses Pelatihan (fit)

Kami menggunakan teknik Mini-Batch Gradient Descent:

a. Inisialisasi Cerdas: Sebelum mulai belajar, bobot awal diatur secara acak namun terukur (Metode Xavier) agar proses belajar tidak macet di awal.
b. Belajar Bertahap: Data dipelajari dalam kelompok-kelompok kecil (batch), bukan satu per satu atau semuanya sekaligus. Ini membuat proses lebih cepat dan stabil.
c. Makin Lama Makin Teliti: Kecepatan belajar (learning_rate) dibuat makin lambat seiring berjalannya waktu, supaya saat model hampir pintar, ia tidak mengubah pemahamannya terlalu drastis.

4. Strategi Multi-Kelas: One-vs-All (OneVsAll)

Karena Logistic Regression pada dasarnya adalah algoritma klasifikasi biner (hanya 2 kelas), kami menggunakan strategi One-vs-All untuk menangani 3 kelas target (Dropout, Enrolled, Graduate).

Proses Pelatihan

Sistem memecah masalah klasifikasi multi-kelas menjadi 3 sub-masalah biner yang terpisah:

a. Model 1: Melatih pemisahan antara kelas Dropout (1) melawan Sisa/Lainnya (0).
b. Model 2: Melatih pemisahan antara kelas Enrolled (1) melawan Sisa/Lainnya (0).
c. Model 3: Melatih pemisahan antara kelas Graduate (1) melawan Sisa/Lainnya (0).

Setiap model dilatih secara independen untuk menjadi spesialis dalam mengenali satu kelas tertentu.

Proses Prediksi

Untuk memprediksi label data baru:

a. Ketiga model masing-masing menghitung probabilitas bahwa data tersebut masuk ke kelas target mereka.
b. Sistem membandingkan nilai probabilitas dari ketiga model tersebut.

c. Model dengan probabilitas tertinggi dipilih sebagai hasil prediksi akhir.

Implementasi SVM

1. **Struktur Kelas dan Inisialisasi**

   Kelas utama SVMScratch bertindak sebagai orkestrator untuk klasifikasi multi-kelas, sementara SVMNode menangani klasifikasi biner inti. Parameter-parameter utama pada kelas:

   a. learning_rate: Besar langkah awal yang diambil saat memperbarui bobot.
   b. reg_strength: Parameter regularisasi (lambda) untuk mengontrol trade-off antara margin yang lebar dan kesalahan klasifikasi (Soft Margin SVM).
   c. batch_size: Jumlah sampel yang digunakan dalam satu langkah pembaruan gradien (Mini-Batch).
   d. lr_decay: Fitur untuk mengurangi learning rate seiring bertambahnya epoch.
   e. early_stopping: Mekanisme untuk menghentikan pelatihan jika loss tidak membaik setelah sejumlah epoch tertentu (patience).

2. **Landasan Logika: Linear SVM (Binary)**

   a. Konsep Dasar: Tujuan utama SVM adalah mencari hyperplane pemisah optimal yang memaksimalkan margin antara dua kelas. Keputusan prediksi dibuat berdasarkan tanda dari fungsi linear: Skor = (Bobot * Input) + Bias
      i. Jika Skor > 0, maka diprediksi sebagai Kelas Positif (+1).
      ii. Jika Skor < 0, maka diprediksi sebagai Kelas Negatif (-1).
   b. Fungsi Kerugian: Hinge Loss. Kami menggunakan Hinge Loss dengan regularisasi L2. Fungsi ini menghitung "biaya" kesalahan model.
      i. Sistem menghitung skor margin untuk setiap sampel data.
      ii. Jika sampel sudah diklasifikasikan dengan benar dan berada di posisi aman (jauh dari garis pemisah), maka biaya (loss) adalah nol.
      iii. Jika sampel salah diklasifikasikan atau berada terlalu dekat dengan garis pemisah (di dalam margin), maka biaya akan bertambah sesuai seberapa jauh kesalahannya.
      iv. Total Loss ditambahkan dengan penalti regularisasi (untuk mencegah bobot menjadi terlalu besar).

c. Optimasi: Sub-Gradient Descent. Kami menggunakan teknik penurunan gradien (Gradient Descent) untuk memperbaiki bobot model secara bertahap. Aturan Pembaruan Bobot: Proses ini membedakan dua kondisi:
    i. Kondisi Aman (Klasifikasi Benar & Margin Cukup):
        1. Bobot hanya diperbarui sedikit untuk tujuan regularisasi (mencegah overfitting).
    ii. Kondisi Salah/Melanggar Margin:
        1. Bobot diperbarui secara signifikan ke arah yang memperbaiki kesalahan tersebut (menggeser garis pemisah agar sampel tersebut menjadi benar).
        2. Bias (nilai konstanta) juga disesuaikan.

3. Strategi Multi-Kelas: One-vs-All (OvA)

Karena SVM dasarnya adalah klasifikasi biner, kami menggunakan strategi One-vs-All untuk menangani 3 kelas target (Dropout, Enrolled, Graduate).

Proses Pelatihan

a. Sistem memecah masalah menjadi 3 sub-masalah biner:
    i. Model 1: Dropout (+1) vs Sisa (-1)
    ii. Model 2: Enrolled (+1) vs Sisa (-1)
    iii. Model 3: Graduate (+1) vs Sisa (-1)
b. Setiap model SVMNode dilatih secara independen.

Proses Prediksi

a. Ketiga model masing-masing menghitung skor kepercayaannya.
b. Sistem membandingkan skor dari ketiga model tersebut.
c. Model dengan skor tertinggi (paling yakin positif) dipilih sebagai hasil prediksi akhir.

4. Fitur Lanjutan dan Optimasi

a. Mini-Batch Training: Kami menerapkan teknik Mini-Batch, yaitu memperbarui model menggunakan sekelompok kecil data sekaligus, bukan satu per satu atau semuanya sekaligus. Cara ini membuat proses belajar menjadi lebih cepat namun tetap stabil. Selain itu, urutan data diacak ulang setiap kali satu putaran belajar (epoch) dimulai agar model tidak menghafal urutan data.

b. Learning Rate Decay: Learning rate (laju pembelajaran) dikurangi secara bertahap seiring berjalannya waktu.
    i. Pada awal pelatihan, langkah pembaruan besar agar cepat mendekati solusi.
    ii. Pada akhir pelatihan, langkah diperkecil agar model dapat "mendarat" dengan presisi di titik optimal.
c. Early Stopping: Untuk mencegah overfitting dan menghemat waktu, pelatihan dihentikan secara otomatis jika Loss model tidak lagi menurun secara signifikan selama beberapa periode (epoch) berturut-turut.

Tahap Cleaning dan Preprocessing

Cleaning and Preprocessing dilakukan untuk memastikan kualitas data, mengekstrak pola yang bermakna, dan mempersiapkan dataset agar optimal saat digunakan dalam pelatihan model machine learning. Pada Tugas Besar ini, berikut adalah langkah-langkah sistematis yang kami lakukan:

1. **Pemuatan dan Pembersihan Awal Data**
    a. Pemuatan Data: Sistem memuat dataset pelatihan (train.csv) dan pengujian (test.csv) yang masih mentah.
    b. Penhapusan Duplikat: Baris data yang duplikat diidentifikasi dan dihapus dari data pelatihan untuk mencegah bias pada model dan kebocoran data.
    c. Manajemen ID: Kolom Student_ID diekstrak dari dataset pengujian untuk keperluan submisi akhir, kemudian dihapus dari dataset pelatihan dan pengujian karena tidak memiliki nilai prediktif terhadap target.
2. **Feature Engineering**

    Dilakukan penambahan 23 fitur turunan (derived features) untuk menangkap pola perilaku dan kondisi mahasiswa yang lebih mendalam, antara lain:

    a. Tren Akademik: Grade_Trend (perubahan nilai semester 2 vs 1), Approval_Trend.
    b. Rasio Kinerja: Approval_Ratio (SKS lulus / SKS diambil), Credit_Efficiency.
    c. Indikator Risiko: Total_Failed_Credits, Sem1_Failure_Rate, Sem2_Failure_Rate.
    d. Faktor Ekonomi & Sosial: Econ_Stress_Index (Pengangguran + Inflasi), Financial_Risk (Kombinasi status utang dan beasiswa), First_Gen_Risk (Kualifikasi orang tua).
    e. Interaksi: Fitur interaksi seperti Scholarship_Grade_Interaction dan Age_Performance_Interaction.
3. **Categorical Encoding**
    a. Enkoding Target: Variabel target dipetakan ke nilai numerik:
        i. Dropout menjadi 0
        ii. Enrolled menjadi 1
        iii. Graduate menjadi 2

b. One-Hot Encoding

4. **Data Splitting**

Data latih yang telah diproses dibagi menjadi Set Pelatihan (80%) dan Set Validasi (20%). Hal ini memungkinkan evaluasi kinerja model pada data yang belum pernah dilihat (*unseen data*) sebelum menghasilkan prediksi akhir.

5. **Imputasi Data dan Penanganan Outlier**
    a. Imputasi Nilai Hilang: Nilai yang hilang (*missing value*s) diganti dengan rata-rata (*mean*) dari kolom masing-masing (dihitung berdasarkan training set) untuk memastikan kelengkapan dataset.
    b. Outlier Clipping: Untuk mengurangi dampak anomali esktrem, fitur kontinu dipangkas pada persentil ke-1 dan ke-99.
6. **Feature Scaling**
    a. Metode: Standarisasi (Normalisasi Z-Score)
    b. Proses: Semua fitur ditransformasikan agar memiliki rata-rata 0 dan standar deviasi 1. Langkah ini bertujuan agar data dapat digunakan oleh algoritma seperti SVM dan Logistic Regression yang sensitif terhadap skala data input.
7. **Seleksi Fitur**

Dilakukan penyaringan fitur untuk mengurangi dimensi dan *noise* pada data:

    a. Hapus Kolom Konstan: Fitur yang memiliki nilai sama untuk semua baris dihapus.
    b. Hapus Korelasi Tinggi: Fitur yang memiliki korelasi absolut > 0.95 dengan fitur lain dihapus untuk mengurangi redundansi (multikolinearitas).
8. **Penanganan Ketidakseimbangan Kelas (Saat Pelatihan)**
    a. Metode: SMOTE (*Synthetic Minority Over-sampling Technique*).
    b. Proses: Pada tahap pelatihan (training loop), data latih yang tidak seimbang (jumlah kelas Dropout/Enrolled/Graduate berbeda jauh) ditangani dengan membangkitkan sampel sintetis untuk kelas minoritas. Hal ini memastikan model tidak bias terhadap kelas mayoritas. SMOTE diterapkan hanya pada data training setelah splitting, tidak pada data validasi atau testing.

Perbandingan Algoritma dengan Library

1. Metrik Performa

Tabel berikut merangkum akurasi validasi yang dicapai oleh kedua implementasi pada dataset yang sama yang telah diproses sebelumnya (dengan penyeimbangan SMOTE).

Model	Akurasi Scratch	Akurasi Sklearn	Selisih Absolut
Decision Tree	85.26%	88.85%	3.59%
Logistic Regression	78.39%	79.08%	0.69%
SVM	78.27%	79.60%	1.33%

2. Analisis

a. DTL
   i. Perbedaan Utama: Metode pemangkasan (pruning). Scikit-Learn menggunakan Cost-Complexity Pruning yang canggih, sedangkan implementasi scratch menggunakan strategi Post-Pruning sederhana berbasis validasi akurasi.
   ii. Dampak: Akurasi scratch lebih rendah (~3.6%) karena generalisasi yang kurang optimal dibanding optimasi heuristik kompleks milik Scikit-Learn.

b. Logistic Regression
   i. Perbedaan Utama: Metode optimasi (solver). Scikit-Learn menggunakan solver tingkat lanjut (seperti L-BFGS atau Newton-CG), sedangkan scratch menggunakan Mini-Batch Gradient Descent.
   ii. Dampak: Akurasi hampir identik (selisih ~0.7%). Ini membuktikan bahwa Gradient Descent, meskipun lebih sederhana, mampu mencapai minimum global yang sama pada masalah konveks.

c. Support Vector Machine (SVM)
   i. Perbedaan Utama: Pendekatan matematika. Scikit-Learn (libsvm) menggunakan Quadratic Programming (SMO) untuk menemukan

hyperplane eksak. Scratch menggunakan pendekatan Gradient Descent (Pegasos) untuk mengaproksimasi Hinge Loss.

ii. Dampak: Akurasi sangat kompetitif (selisih ~1.3%). Pendekatan Gradient Descent terbukti menjadi alternatif yang efisien dan cepat.

3. Kesimpulan

Implementasi scratch kami telah terbukti efektif, mencapai akurasi dalam rentang 0.7% hingga 3.6% dari *library*.

Pembagian Tugas

NIM	Nama	Pembagian Tugas
13523131	Ahmad Wafi	• SVM • Laporan
13523143	Amira Izani	• Logistic Regression • Laporan
13523147	Frederiko Eldad Mugiyono	• Integration • Tuning • Submission • Laporan
13523157	Natalia Desiany	• Decision Tree Learning • Laporan
13523160	I Made Wiweka Putera	• Data Cleaning and Preprocessing • Laporan

Referensi

[1] GeeksforGeeks, “CART (Classification and Regression Tree) in Machine Learning,” [Online]. Available: https://www.geeksforgeeks.org/machine-learning/cart-classification-and-regression-tree-in-machine-learning/. [Accessed: Nov. 2025].

[2} GeeksforGeeks, “Decision Tree Algorithms,” [Online]. Available: https://www.geeksforgeeks.org/machine-learning/decision-tree-algorithms/. [Accessed: Nov. 2025].

[3] GeeksforGeeks, “Support Vector Machine Algorithm,” [Online]. Available: https://www.geeksforgeeks.org/machine-learning/support-vector-machine-algorithm/. [Accessed: Nov. 2025].

[4] GeeksforGeeks, “Understanding Logistic Regression,” [Online]. Available: https://www.geeksforgeeks.org/machine-learning/understanding-logistic-regression/. [Accessed: Nov. 2025]

[5] GeeksforGeeks, “Introduction to Tree Data Structure,” [Online]. Available: https://www.geeksforgeeks.org/dsa/introduction-to-tree-data-structure/. [Accessed: Nov. 2025].