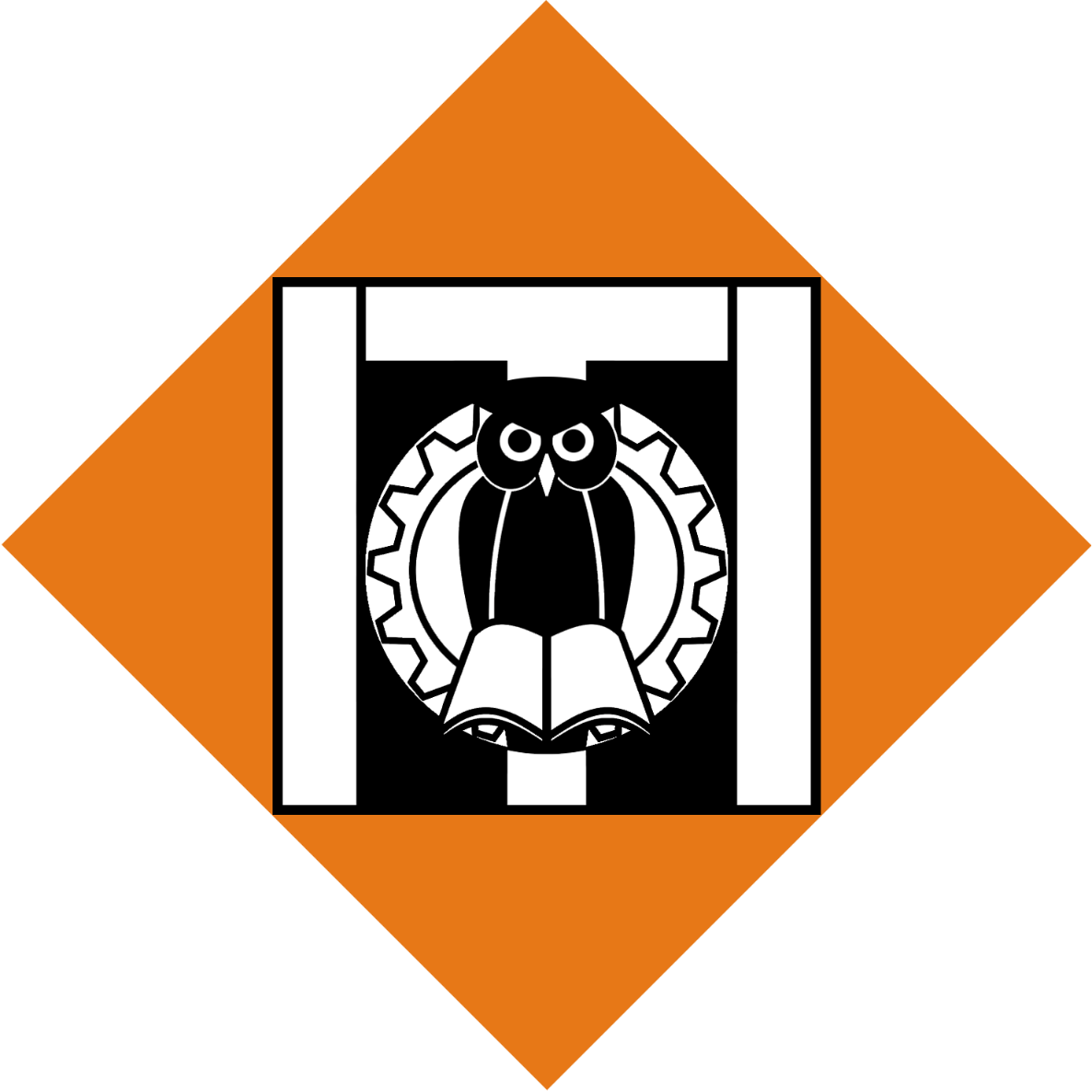
LAPORAN KECERDASAN BUATAN

“NAÏVE BAYES DAN LAPLACIAN CORRECTION WITH PYTHON”

Ditulis oleh:

IKHWAN EL FARIS (1152200009)

JONATHAN NATANNAEL ZEFANYA (1152200024)

DAFFA DANINDRA (1152200028)

DETHALISA AURA KIRANA (1152200037)

# **BAB I**

**PENDAHULUAN**

## **1.1 Masalah**

1. Bagaimana cara mengimplementasikan Naïve Bayes dan Laplacian Correction secara manual tanpa menggunakan library eksternal dalam Python?
2. Seberapa akurat dan efisien implementasi Naïve Bayes dan Laplacian Correction manual dalam mengklasifikasikan dataset Pendapatan Orang Dewasa dibandingkan dengan menggunakan library yang sudah ada?
3. Bagaimana menyusun kode Python untuk Naïve Bayes dan Laplacian Correction yang mudah dipahami, dioptimalkan, dan tetap memberikan hasil yang akurat dalam menghitung probabilitas dataset Pendapatan Orang Dewasa?
4. Bagaimana cara mengolah dan menyiapkan dataset Pendapatan Orang Dewasa agar dapat digunakan dalam Naïve Bayes dan Laplacian Correction?

## **1.2 Tujuan**

* 1. Mengimplementasikan Naïve Bayes dan Laplacian Correction secara manual dalam Python tanpa menggunakan library eksternal.
  2. Membandingkan akurasi dan efisiensi antara implementasi Naïve Bayes dan Laplacian Correction manual dan implementasi Naïve Bayes dan Laplacian Correction menggunakan library **eksternal**.
  3. Membuat kode Python untuk Naïve Bayes dan Laplacian Correction agar memberikan hasil yang akurat dalam menghitung probabilitas dataset Pendapatan Orang Dewasa.
  4. Mengolah dan menyiapkan dataset Pendapatan Orang Dewasa agar dapat digunakan dalam perhitungan Naïve Bayes dan Laplacian Correction manual dengan benar.

# **BAB II**

**DASAR TEORI**

## **2.1 Naïve Bayes**

Naïve Bayes adalah salah satu algoritma klasifikasi yang menggunakan prinsip probabilitas. Algoritma ini didasarkan pada Teorema Bayes yang mengkalkulasi probabilitas suatu kelas berdasarkan fitur-fitur yang ada, dengan asumsi bahwa setiap fitur bersifat independen satu sama lain. Asumsi independensi inilah yang membuat algoritma ini disebut "naïve". Probabilitas adalah konsep dasar dalam statistika yang mengukur kemungkinan terjadinya suatu kejadian. Dalam konteks machine learning, probabilitas sering digunakan untuk memperkirakan kelas atau label suatu data berdasarkan fitur-fitur yang dimilikinya.

Naïve Bayes bekerja dengan menghitung probabilitas prior dari setiap kelas, serta probabilitas likelihood dari fitur-fitur yang diberikan kelas tertentu. Probabilitas posterior, yang merupakan probabilitas suatu kelas diberikan fitur-fitur tertentu, kemudian dihitung dan digunakan untuk memprediksi kelas dengan probabilitas tertinggi. Meskipun asumsi independensi antara fitur seringkali tidak realistis dalam kasus dunia nyata, Naïve Bayes tetap menjadi algoritma yang efektif dan efisien, terutama untuk dataset besar dan situasi dimana kecepatan prediksi menjadi faktor penting.

1. Keuntungan
   1. Mudah diimplementasikan.
   2. Akurasi bagus dalam banyak kasus.
2. Kerugian:
   1. Asumsi bahwa : Masing-masing ciri tidak saling tergantung (independent) satu sama lain, sehingga tidak cocok untuk kasus kasus tertentu yang membutuhkan kebergantungan antar ciri dan kelas.

Diberikan data testing X, maka probabilitas (posteriori probability) dari sebuah hipotesa H, P(H|X) adalah:

P ( H | X ) = P ( X | H ) P ( H ) / P ( X ) = P ( X | H ) × P ( H ) / P ( X )

Secara informal dapat ditulis:

posteriori = likelihood x prior/evidence

X adalah data (“evidence”) yang klasifikasinya tidak diketahui.

H adalah hipotesa bahwa X masuk klasifikasi C.

Klasifikasi adalah menentukan P(H|X) atau yang disebut dengan posteriori probability yaitu probabilitas dari sebuah hipotesa diberikan data X.

P(H) (prior probability), adalah probabilitas awal.

P(X): Probabilitas dari data yang akan dicari klasifikasinya.

P(X|H) (likelihood), the probabilitas data X jika diberikan given that the hipotesa H.

## **2.2 Laplacian Correction**

Laplacian correction, atau dikenal juga sebagai Laplace smoothing, adalah teknik yang digunakan dalam statistika dan pemodelan probabilistik untuk menangani masalah zero probability. Dalam konteks klasifikasi Naive Bayes, hal ini terjadi ketika suatu fitur dalam data pengujian tidak muncul dalam data pelatihan untuk kelas tertentu, yang mengakibatkan probabilitas nol dan membuat prediksi menjadi tidak akurat.

Untuk mengatasi masalah ini, Laplacian correction menambahkan nilai konstan (biasanya 1) ke setiap hitungan kategori. Dengan demikian, tidak ada probabilitas yang menjadi nol, sehingga memperbaiki estimasi probabilistik. Metode ini efektif karena memungkinkan semua kemungkinan nilai untuk setiap fitur memiliki peluang non-nol, meskipun tidak pernah muncul dalam data pelatihan. Hal ini memastikan model lebih robust dan dapat menangani data baru yang mungkin mengandung fitur yang tidak terlihat selama pelatihan. Implementasi manual dari Laplacian correction dalam Python biasanya melibatkan penambahan nilai smoothing pada hitungan frekuensi kategori sebelum menghitung probabilitas kondisional.

Mengganti nilai 0 dengan nilai kecil. 0 ditambahkan dengan nilai yang kecil, misalnya nilai 1, begitu juga dengan semua pembilang pada perhitungan probabilitas pada ciri dan klasifikasi yang bersangkutan. Kita sebut α ==1. Penyebutnya (penyebut pada ciri dan klasifikasi yang bersangkutan) tentu juga harus ditambahkan dengan sebuah nilai yaitu: k\*α. Mengapa dikalikan dengan k? Karena penyebut harus besar, agar nilai kecil. Nilai k adalah jumlah nilai pada ciri (feature) yang mengalami zero probability tersebut.

# **BAB III**

**DATA DAN METODOLOGI**

## **3.1 Data**

### 3.1.1 Sumber Data

Kumpulan data berisi Pendapatan tahunan seseorang dihasilkan dari berbagai faktor. Secara intuitif dipengaruhi oleh tingkat pendidikan individu, usia, jenis kelamin, pekerjaan, dan lain-lain, Yang dibuat oleh “Wenruliu”.

Berikut link dari data yang digunakan:

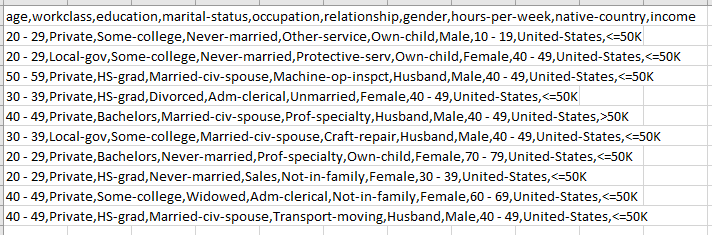
https://www.kaggle.com/datasets/wenruliu/adult-income-dataset

### 3.1.2 Pre-processing Data

- Data Sebelum Dirubah

**Gambar 3.1 Data Sebelum Dirubah**

- Data Testing:

****

**Gambar 3.2 Data Testing**

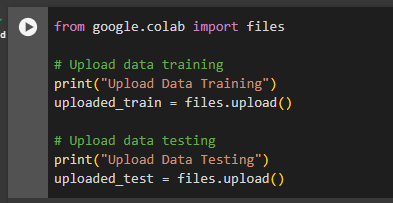
* Data Training:

**Gambar 3.3 Data Training**

## **3.2 Metodologi**

### 3.2.1 Mengupload Data

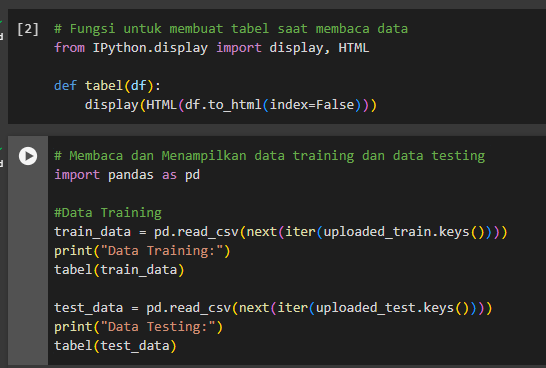
Kode tersebut merupakan bagian dari program Python yang digunakan untuk mengunggah file data ke lingkungan Google Colab. Pertama, modul `files` dari `google.colab` diimpor untuk memungkinkan interaksi dengan sistem file Colab. Setelah itu, kode mencetak pesan "Upload Data Training" untuk memberi tahu pengguna bahwa mereka perlu mengunggah file data pelatihan. Fungsi `files.upload()` kemudian dipanggil untuk membuka dialog pengunggahan file, memungkinkan pengguna untuk memilih dan mengunggah file data pelatihan mereka. Proses ini diulangi untuk file data pengujian, dengan mencetak pesan "Upload Data Testing" dan memanggil `files.upload()` lagi untuk mengunggah file data pengujian. File yang diunggah akan disimpan dalam variabel `uploaded\_train` dan `uploaded\_test` sebagai dictionary yang berisi nama file dan konten file tersebut.



**Gambar 3.4 Mengupload Data**

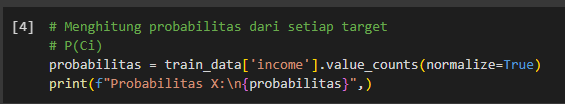
### 3.2.2 Membuat table dan membaca,menampilkan data.

Kode tersebut merupakan bagian dari program Python yang digunakan untuk membaca dan menampilkan data dalam bentuk tabel di lingkungan Jupyter Notebook atau Google Colab. Pertama, modul `display` dan `HTML` dari `IPython.display` diimpor untuk menampilkan tabel HTML. Fungsi `tabel` didefinisikan untuk mengkonversi DataFrame pandas menjadi HTML dan menampilkannya tanpa indeks. Selanjutnya, modul `pandas` diimpor sebagai `pd` untuk memudahkan manipulasi data. Data pelatihan dibaca dari file CSV yang telah diunggah sebelumnya menggunakan `pd.read\_csv` dan disimpan dalam `train\_data`. Data ini kemudian ditampilkan dengan mencetak "Data Training:" dan memanggil fungsi `tabel` untuk menampilkan DataFrame. Proses serupa dilakukan untuk data pengujian, yang dibaca dan ditampilkan dengan cara yang sama.



**Gambar 3.5 membuat table dan membaca, menampilkan data.**

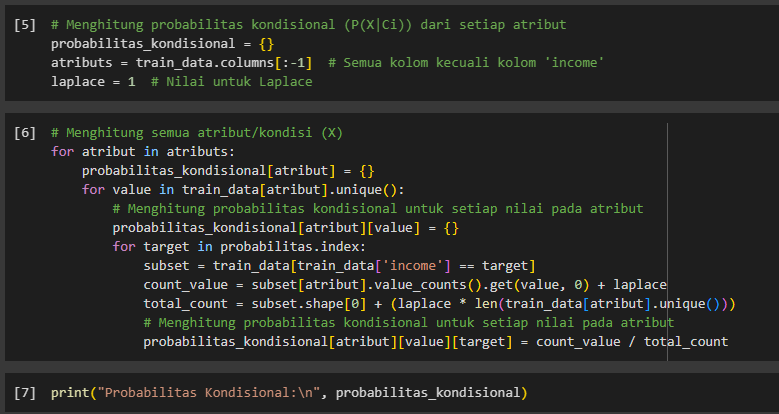
### 3.2.3 Menghitung probabilitas dari setiap target

 Kode tersebut menghitung probabilitas prior dari setiap kategori dalam kolom `income` pada data pelatihan `train\_data`. Dengan menggunakan metode `value\_counts` dari pandas dan mengatur parameter `normalize=True`, kode ini mengkalkulasi proporsi atau probabilitas dari setiap nilai unik dalam kolom `income`. Hasilnya adalah seri pandas yang menyimpan probabilitas masing-masing kategori. Kemudian, hasil ini dicetak dengan judul "Probabilitas Prior:" diikuti oleh nilai-nilai probabilitas yang telah dihitung, memberikan gambaran tentang distribusi awal kelas-kelas dalam data pelatihan.

**Gambar 3.6 Menghitung probabilitas dari setiap target.**

### 3.2.4 Perhitungan Probabilitas Kondisional

Kode tersebut menghitung probabilitas kondisional (P(X|Ci)) untuk setiap atribut dalam dataset pelatihan, di mana Ci adalah kelas target (misalnya, "income"). Pertama, sebuah dictionary `probabilitas\_kondisional` disiapkan untuk menyimpan probabilitas ini. Semua atribut, kecuali kolom target "income", diambil dari dataset `train\_data`. Untuk mengatasi kemungkinan nilai nol dalam perhitungan probabilitas, digunakan Laplace smoothing dengan nilai 1.

Kemudian, untuk setiap atribut dalam dataset, kode ini menghitung probabilitas kondisional untuk setiap nilai unik dari atribut tersebut dalam konteks setiap kelas target. Proses ini melibatkan iterasi melalui setiap nilai unik dari atribut dan menghitung jumlah kejadian nilai tersebut dalam subset data yang memiliki kelas target tertentu. Jumlah ini ditambah dengan nilai Laplace untuk smoothing. Total kejadian untuk kelas target tersebut juga disesuaikan dengan smoothing. Probabilitas kondisional akhirnya dihitung dengan membagi jumlah kejadian yang telah di-smoothed dengan total kejadian yang juga telah di-smoothed, dan hasilnya disimpan dalam dictionary `probabilitas\_kondisional`. Terakhir, dictionary ini dicetak untuk menunjukkan probabilitas kondisional yang telah dihitung.

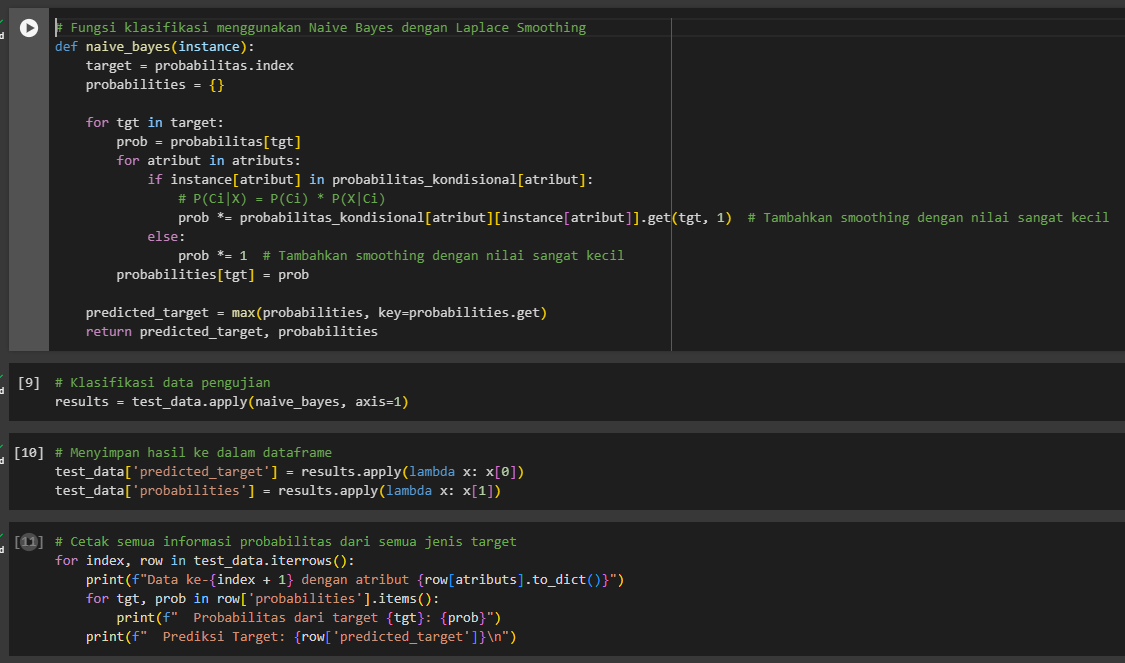
**Gambar 3.7 Perhitungan Probabilitas Kondisional**

### 3.2.5 Klasifikasi Data dan Prediksi Target

Kode tersebut mendefinisikan fungsi `naive\_bayes` untuk melakukan klasifikasi menggunakan algoritma Naive Bayes dengan Laplace Smoothing. Fungsi ini menerima sebuah instance (baris data) dan menghitung probabilitas setiap kelas target berdasarkan probabilitas a priori dari kelas dan probabilitas kondisional dari atribut. Untuk setiap kelas target, probabilitas awal dikalikan dengan probabilitas kondisional dari setiap atribut dalam instance. Jika nilai atribut tidak ditemukan dalam probabilitas kondisional, probabilitas dikalikan dengan 1 (dengan asumsi smoothing).

Setelah menghitung probabilitas untuk semua kelas target, kelas dengan probabilitas tertinggi dipilih sebagai prediksi. Fungsi mengembalikan prediksi dan probabilitas yang dihitung.

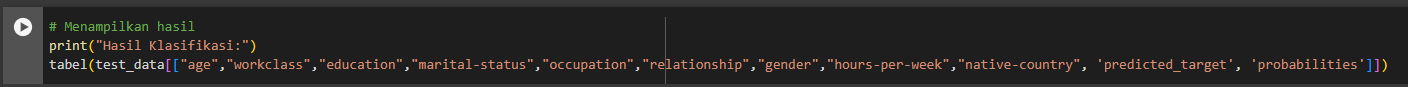
Selanjutnya, fungsi `naive\_bayes` diterapkan pada setiap baris data pengujian menggunakan metode `apply`. Hasil prediksi dan probabilitas disimpan dalam kolom baru 'predicted\_target' dan 'probabilities' pada dataframe `test\_data`. Terakhir, kode ini mencetak informasi tentang setiap instance dalam data pengujian, termasuk atributnya, probabilitas untuk setiap kelas target, dan prediksi target.



**Gambar 3.8 Klasifikasi Data dan Prediksi Target.**

### 3.2.6 Hasil Klasifikasi

Kode tersebut digunakan untuk menampilkan hasil klasifikasi dalam bentuk tabel dengan kolom-kolom tertentu. Baris `print("Hasil Klasifikasi:")` digunakan untuk mencetak teks "Hasil Klasifikasi" sebagai judul atau pengantar hasil yang akan ditampilkan. Kemudian, fungsi `tabel()` digunakan untuk menampilkan data yang ada pada variabel `test\_data` dengan hanya menampilkan kolom-kolom yang terpilih, yaitu "age" (usia), "workclass" (kelas pekerjaan), "education" (pendidikan), "marital-status" (status perkawinan), "occupation" (pekerjaan), "relationship" (hubungan), "gender" (jenis kelamin), "hours-per-week" (jam kerja per minggu), "native-country" (negara asal), "predicted\_target" (target yang diprediksi), dan "probabilities" (probabilitas). Kolom-kolom ini mencakup informasi demografis, hasil prediksi klasifikasi, dan probabilitas terkait dari model yang digunakan.



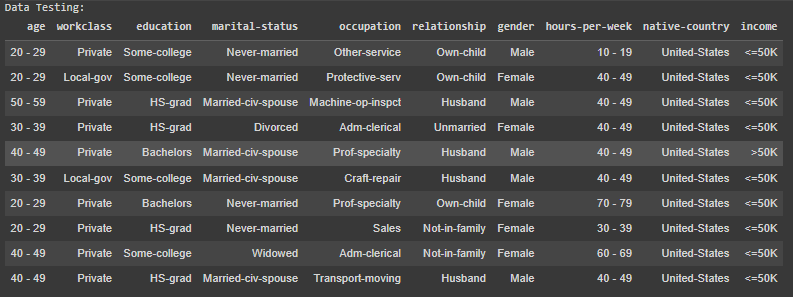
**Gambar 3.9 Hasil Klasifikasi.**

# **BAB IV**

**HASIL DAN ANALISA**

## **4.1 Hasil**

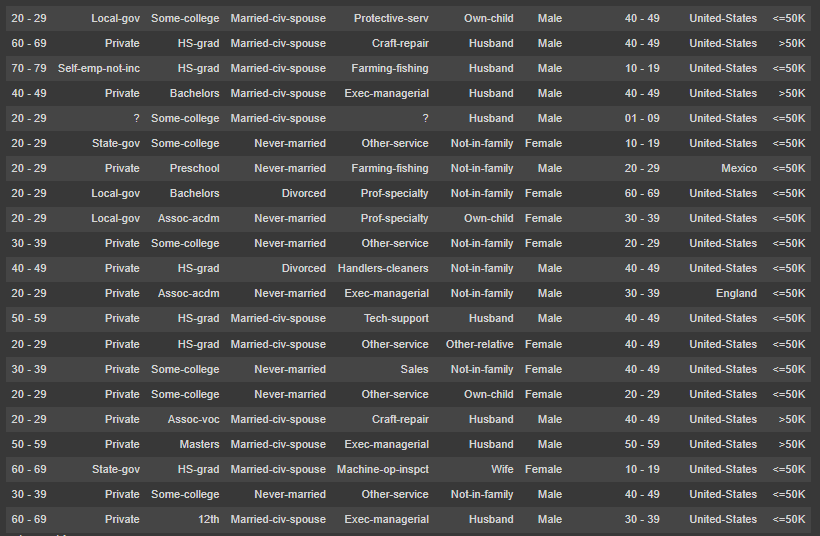
### 4.1.1 Hasil Menampilkan Data Testing



**Gambar 4.1 Hasil Menampilkan Data Testing**

### 4.1.2 Hasil Menampilkan Data Training

**Gambar 4.2 Hasil Menampilkan Data Training.**

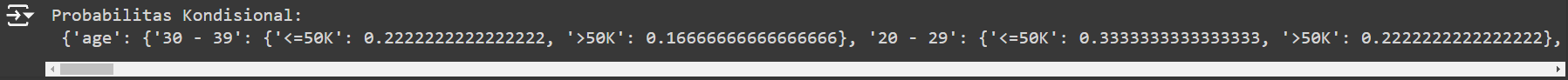
****

**Gambar 4.3 Hasil Menampilkan Data Training.**

### 4.1.3 Hasil Perhitungan Probabilitas X

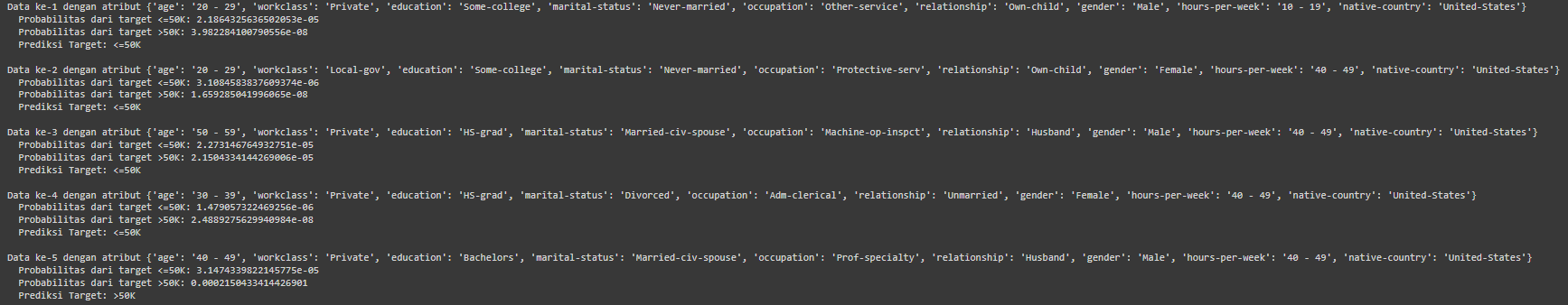
**Gambar 4.4 Hasil Perhitungan Probabilitas X**

### 4.1.4 Hasil Perhitungan Probabilitas Kondisional

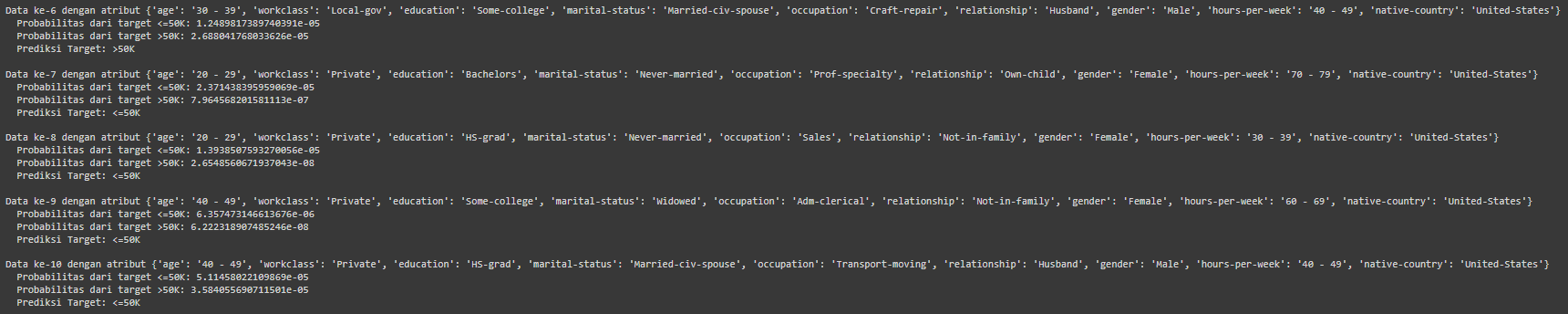
Pada perhitungan ini menampilkan hasil yang cukup Panjang.

**Gambar 4.5 Hasil Perhitungan Probabilitas Kondisional.**

### 4.1.5 Hasil Probabilitas dan Prediksi Target

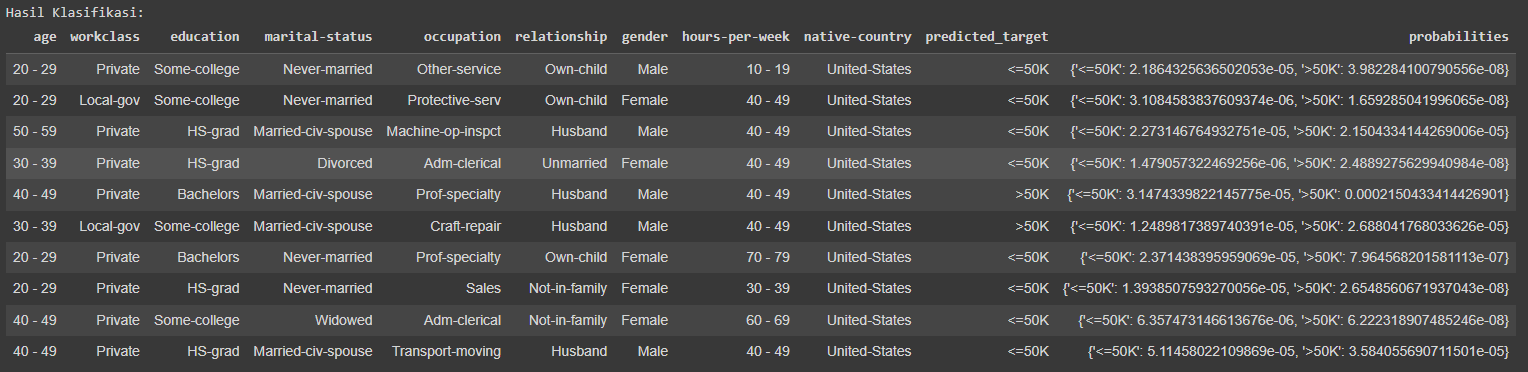


**Gambar 4.6 Hasil Probabilitas dan Prediksi Target.**

****

**Gambar 4.7 Hasil Probabilitas dan Prediksi Target.**

### 4.1.6 Hasil Klasifikasi



**Gambar 4.8 Hasil Klasifikasi.**

# **BAB V**

**KESIMPULAN**

## **5.1 Kesimpulan**

Kesimpulan adalah bahwa algoritma Naïve Bayes dengan Laplacian Correction telah berhasil diimplementasikan secara manual dalam Python tanpa menggunakan library eksternal, menunjukkan bahwa pemahaman yang mendalam tentang teori dasar dapat menghasilkan kode yang efisien dan akurat. Implementasi ini tidak hanya membuktikan kemampuan algoritma Naïve Bayes dalam mengklasifikasikan dataset Pendapatan Orang Dewasa secara efektif, tetapi juga menunjukkan bahwa meskipun asumsi independensi antara fitur tidak selalu realistis, algoritma ini tetap memberikan kinerja yang baik. Penggunaan Laplacian Correction terbukti esensial dalam mengatasi masalah probabilitas nol, yang meningkatkan estimasi probabilistik dan akurasi prediksi secara keseluruhan. Proses pre-processing data yang meliputi pengunggahan, pembacaan, dan pengolahan data, dilakukan dengan cermat untuk memastikan bahwa dataset siap digunakan dalam perhitungan Naïve Bayes dan Laplacian Correction. Hasil klasifikasi menunjukkan bahwa model yang dibangun mampu mengklasifikasikan data dengan baik, dan perhitungan probabilitas kondisional serta prediksi target mengonfirmasi akurasi model. Keseluruhan eksperimen ini menegaskan bahwa dengan teknik yang tepat, algoritma Naïve Bayes dapat dioptimalkan untuk berbagai aplikasi praktis dalam kecerdasan buatan, sekaligus memberikan dasar yang kuat untuk pengembangan lebih lanjut dan penerapan algoritma probabilistik dalam berbagai domain.

# **Referensi**

* Wenruliu, Adult income dataset. Kaggle. Retrived June 19, 2024, from

<https://www.kaggle.com/datasets/wenruliu/adult-income-dataset>

* Naive Bayes Classification dan Contoh Perhitungannya (Materi Naïve Bayes pada matakuliah Kecerdasan Buatan).
* Laplacian Correction\_v.Newest (Materi Laplacian Correction pada matakuliah Kecerdasan Buatan).
* Wasserman, Larry. "All of Statistics: A Concise Course in Statistical Inference." Springer Science & Business Media, 2013.
* Murphy, Kevin P. "Machine Learning: A Probabilistic Perspective." MIT press, 2012.
* Bishop, Christopher M. "Pattern Recognition and Machine Learning." Springer, 2006.
* Mitchell, Tom M. "Machine Learning." McGraw-Hill, 1997.
* Domingos, Pedro, and Michael Pazzani. "On the optimality of the simple Bayesian classifier under zero-one loss." Machine learning 29.2 (1997)
* Jurafsky, D., & Martin, J. H. (2021). "Speech and Language Processing" (3rd Edition).
* Wikipedia - "Additive smoothing".

# **SIAPA MENGERJAKAN APA:**

1. Jonathan Natannael Zefanya (1152200024) – Mengerjakan Program
2. Ikhwan El Faris (1152200009) – Mencari Dan Memberikan Algoritma Program
3. Daffa Danindra (1152200028) – Mengambil Dan Membuat Seluruh Data Yang Dibutuhkan, Mengerjakan Laporan
4. Dethalisa Aura Kirana (1152200037) – Mengerjakan Laporan