Implementasi Algoritma *Supervised Learning*

Diajukan sebagai pemenuhan tugas besar 2

IF3170 - Intelegensi Buatan.

Oleh:

Kelompok 12 - NJ Kuadrat

Alexander Jason	13521100
Juan Christopher Santoso	13521116
Nathania Calista Djunaedi	13521139
Antonio Natthan Krishna	13521162

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2023

Daftar Isi

Daftar Tabel

Daftar Gambar

Bab I
Pendahuluan

Berdasarkan *exploratory data analysis* (EDA) yang telah disusun pada tugas kecil 2, penulis diminta untuk melakukan implementasi terhadap dua algoritma pembelajaran mesin yaitu KNN (*K-nearest neighbor*) dan Naive-Bayes. Data yang digunakan pada pelaksanaan implementasi algoritma dalam hal ini adalah data yang sama yang digunakan pada pengerjaan tugas kecil 2 dengan rincian setiap kolom pada data sebagai berikut:

Tabel 1.1 Rincian nama dan deskripsi kolom pada data

No	Nama Kolom	Tipe Data	Deskripsi Kolom
1	*battery_power*	*integer*	Total energi baterai dalam satu waktu diukur dalam mAh
2	*blue*	*integer*	Memiliki *bluetooth* atau tidak
3	*clock_speed*	*float*	Kecepatan *microprocessor* menjalankan instruksi
4	*dual_sim*	*integer*	Memiliki dukungan dual sim atau tidak
5	*fc*	*integer*	Resolusi kamera depan dalam *megapixel*
6	*four_g*	*integer*	Memiliki 4G atau tidak
7	*int_memory*	*integer*	Memori internal dalam *gigabyte*
8	*m_dep*	*float*	Ketebalan ponsel dalam *centimeter*
9	*mobile_wt*	*integer*	Berat ponsel
10	*n_cores*	*integer*	Jumlah *core processor*
11	*pc*	*integer*	Resolusi kamera utama dalam *megapixel*
12	*px_height*	*integer*	Tinggi resolusi *pixel*
13	*px_width*	*integer*	Lebar resolusi *pixel*
14	*ram*	*integer*	Ukuran RAM dalam *megabyte*

15	*sc_h*	*integer*	Tinggi layar ponsel dalam *centimeter*
16	*sc_w*	*integer*	Lebar layar ponsel dalam *centimeter*
17	*talk_time*	*integer*	Waktu telepon maksimum dalam satu kali pengisian baterai
18	*three_g*	*integer*	Memiliki 3G atau tidak
19	*touch_screen*	*integer*	Memiliki layar sentuh atau tidak
20	*wifi*	*integer*	Memiliki *wifi* atau tidak
21	*price_range* (***target***)	*integer*	Rentang harga dengan nilai: 0 adalah biaya rendah, 1 adalah biaya sedang, 2 adalah biaya tinggi, dan 3 adalah biaya sangat tinggi.

Bab II
Algoritma KNN

2.1. Dasar Teori

K-Nearest Neighbor adalah algoritma pembelajaran mesin yang menggunakan metode *supervised learning* dan digunakan untuk mengklasifikasi data yang dimasukkan oleh pengguna. Algoritma ini termasuk algoritma yang sederhana dan sering kali digunakan karena tidak memiliki *training phase* dan tidak perlu dibentuk model. Algoritma *K-Nearest Neighbor* melakukan klasifikasi terhadap objek *input* dengan cara menghitung jarak antara objek dengan data *training.* Perhitungan jarak antara objek dengan data *training* dapat menggunakan *euclidean distance* dengan rumus sebagai berikut

$$Dxy = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

Gambar 2.1 *Euclidean Distance*
(Sumber:E Journal UNIB)

Pada rumus di atas, Dxy menggambarkan jarak antara data x dengan data y. X adalah data *training* dan y adalah data *testing*. Variabel n pada rumus di atas menggambarkan jumlah atribut atau kolom individu.

Setelah mendapatkan jarak antara data *training* dengan data *testing,* algoritma ini akan memilih K data dengan *euclidean distance* terkecil. Dari seluruh K data tersebut, program akan melihat mayoritas label yang dimiliki oleh K data tersebut. Selanjutnya, program akan melabeli data *input* dengan mayoritas label yang sudah dipilih tadi.

2.2. Implementasi Algoritma tanpa penggunaan *library*

Dalam penerapan implementasi algoritma K-Nearest Neighbor dari *scratch,* langkah - langkah yang penulis lakukan adalah sebagai berikut :

1. Membaca *dataset training* yang berada dalam *file csv*
2. Membaca *dataset valid* yang berada dalam file *csv* dan akan digunakan untuk diuji terhadap *dataset training*.
3. Melakukan iterasi sebanyak jumlah data dalam *dataset valid*

4. *Assign* data ke - i pada iterasi i ke dalam sebuah variabel, misalnya variabel bernama *current*
5. Cari selisih antara variabel *current* dengan seluruh data pada *dataset training* dan masukkan selisih ke dalam sebuah *array* bernama *distances*
6. Setelah selesai menghitung selisih antara variabel *current* dengan seluruh data pada *dataset training,* masukkan *distances* sebagai kolom baru di *dataset training*.
7. Urutkan *dataset training* berdasarkan dari nilai *distances* terkecil sampai terbesar
8. Ambil k *data* dari *dataset training* teratas
9. Dari sejumlah k data yang diambil, temukan mayoritas nilai *price range.*
10. Setelah menemukan *price range,* nilai ini akan dianggap sebagai prediksi yang dibuat oleh program dan data ke-i akan diberi label dengan nilai prediksi tersebut.
11. Setelah seluruh data dalam *dataset valid* mendapatkan label, program akan menghitung akurasi dari program dengan cara membandingkan nilai prediksi dengan nilai sesungguhnya.

2.3. Implementasi Algoritma menggunakan *library*

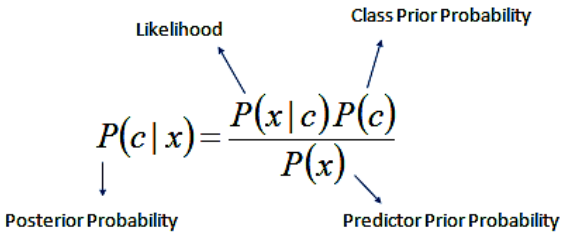
Dalam penerapan implementasi algoritma K-Nearest Neighbor menggunakan *library,* langkah - langkah yang penulis lakukan adalah sebagai berikut :

1. Melakukan *import library* KNeighborClassifier dan SelectKBest dari Scikit-Learn
2. Membaca *dataset training* yang berada dalam *file csv*
3. Membaca *dataset valid* yang berada dalam file *csv* dan akan digunakan untuk diuji terhadap *dataset training*.
4. Memisahkan *dataset training* yang hanya berisi kolom target (“price_range”), yang diberi nama ‘X_test’; dengan kolom lain, yang diberi nama ‘X_train’.
5. Memisahkan *dataset valid* yang hanya berisi kolom target (“price_range”), yang diberi nama ‘y_test’; dengan kolom lain, yang diberi nama ‘y_train’.
6. Menentukan jumlah *neighbors* k
7. Membuat model KNN dari ‘X_train’ sebagai data latih dan ‘X_test’ sebagai data uji.
8. Memeriksa skor akurasi dari hasil model yang sudah dihasilkan.

Bab III
Algoritma Naive-Bayes

3.1. Dasar Teori

Algoritma Naive-Bayes didasari oleh penghitungan probabilitas menggunakan teorema Bayes dalam yaitu:



Gambar 3.1 Teorema Bayes
(Sumber: UC Business Analytics R Programming Guide)

Dalam hal ini, variabel c, pada gambar 3.1, dapat diasumsikan sebagai variabel target yang ingin dicapai menggunakan algoritma Naive Bayes. Di sisi lain, variabel x merepresentasikan variabel yang mempengaruhi kolom target. Dengan kata lain, semua variabel, selain variabel target, akan digunakan menjadi variabel x untuk menentukan probabilitas akhir dari kolom target.

Rumusan teorema Bayes pada gambar 3.1 lantas dapat diturunkan sehingga dihasilkan rumusan sebagai berikut:

$$P(c \mid \mathbf{X}) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$$

Gambar 3.2 Penurunan *Naive Bayes Classifier*
(Sumber: UC Business Analytics R Programming Guide)

Berdasarkan gambar 3.2, variabel X besar menunjukkan seluruh variabel selain kolom target. Dengan kata lain, probabilitas yang ditentukan oleh rumusan tersebut adalah probabilitas untuk variabel target (variabel c) yang dipengaruhi oleh seluruh variabel selain variabel target (variabel X). Nilai probabilitas tersebut lantas dapat diturunkan yaitu dengan mengalikan seluruh probabilitas variabel x untuk nilai variabel target (variabel c) tertentu.

Penghitungan nilai probabilitas dari $p(xi|c)$ dibedakan berdasarkan jenis nilai yang dikandung dalam sebuah kolom. Berdasarkan data yang digunakan dalam hal ini, terdapat jenis nilai yang dikandung, yaitu nilai nominal, untuk data yang bernilai 1 atau 0 (selanjutnya akan disebut dengan data *boolean*), dan nilai ratio, untuk data yang memiliki nilai angka bahkan desimal (selanjutnya akan disebut dengan data numerik). Pemrosesan untuk masing-masing jenis data akan dijelaskan sebagai berikut:

3.1.1. Pemrosesan data *boolean*

Untuk membantu memahami cara penghitungan nilai probabilitas untuk data *boolean*, diperlukan pemahaman mengenai bagaimana nilai *likelihood* bekerja. Nilai *likelihood* dapat diartikan sebagai kecenderungan terjadinya suatu kejadian *dependent* bila diketahui bahwa diketahui nilai kejadian *independent*. *Likelihood*, sering dinotasikan sebagai $p(x|c)$, dalam hal ini variabel x adalah variabel selain target, berkebalikan dengan variabel c yang merupakan variabel target. Untuk penjelasan yang dapat lebih mudah dimengerti, akan digunakan gambar 3.3 di bawah ini:

outlook			temperature			humidity			windy			play	
	yes	no		yes	no		yes	no		yes	no	yes	no
sunny	2	3	hot	2	2	high	3	4	false	6	2	9	5
overcast	4	0	mild	4	2	normal	6	1	true	3	3		
rainy	3	2	cool	3	1								

Gambar 3.3 Tabel Data Probabilitas Target dan Nilai *Likelihood*
(Sumber: PPT Materi Pembelajaran IF3170)

Pada tabel tersebut, baris paling atas yang terdapat pada tabel adalah nama kolom dengan kolom paling kanan yaitu *play* sebagai variabel target. Di bagian bawah nama kolom, terdapat daftar kemungkinan nilai untuk variabel *play* yaitu nilai *yes* dan *no*. Di bawah nilai-nilai tersebut, didaftarkan seluruh kemungkinan nilai yang mungkin pada kolom yang bersangkutan. Contohnya, pada kolom *outlook*, seluruh daftar nilai yang mungkin adalah *sunny*, *overcast*, dan *rainy*.

Untuk setiap baris pada nilai tersebut, diberikan sebuah nilai yang disesuaikan dengan nilai target yang ada. Contohnya untuk baris nilai *sunny* pada kolom *yes* bernilai 2 dan pada kolom *no* bernilai 3. Nilai-nilai tersebut adalah nilai frekuensi untuk kolom *sunny* pada variabel target tertentu. Dengan

kata lain, frekuensi kemunculan *sunny* untuk nilai target *yes,* $n(sunny|yes)$, adalah 2, sedangkan frekuensi kemunculan *sunny* ketika nilai target *no,* $n(sunny|no)$, adalah 3. Untuk menghitung nilai *likelihood* nilai frekuensi tersebut dibagi dengan nilai frekuensi dari kemunculan target yang bersangkutan. Pada gambar 3.3, dapat dilihat bahwa frekuensi nilai *yes* adalah 9 dan frekuensi nilai *no* adalah 5. Lantas, nilai *likelihood* untuk *sunny* ketika *yes*, $p(sunny|yes)$, adalah 2/9 dan nilai *likelihood* untuk *sunny* ketika *no*, $p(sunny|no)$, adalah 3/5. Nilai *likelihood* inilah yang digunakan sebagai probabilitas dalam perhitungan nilai $p(xi|c)$.

3.1.2. Pemrosesan data numerik

Untuk pemrosesan data numerik, penghitungan nilai *likelihood* tidak bisa dilakukan. Hal ini dikarenakan kemungkinan tak terbatas yang dapat terjadi pada nilai data numerik. Maka dari itu, penghitungan nilai probabilitas untuk data numerik dilakukan dengan menggunakan rumus distribusi normal.

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2}$$

Gambar 3.4 Rumus Probabilitas untuk Data Distribusi Normal
(Sumber: Investopedia)

Dalam menggunakan rumus tersebut dibutuhkan nilai rata-rata ($\boldsymbol{\mu}$), nilai standar deviasi (σ), dan nilai x itu nilai data sendiri. Untuk setiap nilai rata-rata dan standar deviasi yang dibutuhkan adalah nilai rata-rata dan standar deviasi yang dikelompokkan untuk setiap nilai target. Pada data yang diberikan, dapat diketahui bahwa domain dari nilai target hanyalah nilai 0, 1, 2, dan 3. Maka dari itu pada setiap kolom yang ada, terdapat masing-masing 4 nilai rata-rata dan standar deviasi untuk setiap kelompok nilai target.

Ketika digunakan untuk memprediksi (atau berarti ditemukan sebuah data baru), data baru tersebut akan dimasukkan sebagai nilai x. Nilai x pada kolom tertentu akan dihitung dengan setiap nilai rata-rata dan standar deviasi yang telah disimpan untuk seluruh klasifikasi nilai target. Hasil perhitungan yang dihasilkan dari rumus tersebut merepresentasikan nilai *likelihood* $p(xi|c)$.

3.2. Implementasi Algoritma tanpa penggunaan *library*

Sebelum masuk lebih dalam mengenai penjelasan implementasi algoritma, diasumsikan bahwa data telah melewati tahap pra-pemrosesan atau *pre-processing*. Data akan dipecah (*split*) menjadi beberapa kelompok yang kelak akan disebut sebagai *train* (data latih), *valid* (data validasi), dan *test* (data pengujian). Dalam mengimplementasikan algoritma Naive Bayes tanpa menggunakan *library*, berikut adalah langkah-langkah yang diambil oleh penulis:

1. Untuk setiap kelompok data yang telah dijelaskan pada bagian dasar teori, nilai-nilai tersebut akan disimpan ke dalam tipe data *dictionary*. Proses penyimpanan dan penghitungan nilai pada *dictionary* ini merupakan tahap *training* untuk algoritma Naive Bayes. *Dictionary* yang dibentuk antara lain:
    a. *Dictionary* frekuensi kemunculan untuk setiap klasifikasi nilai *target*,
    b. *Dictionary* frekuensi kemunculan untuk setiap nilai kolom data *boolean* terhadap klasifikasi nilai *target*. *Dictionary* inilah yang kelak digunakan sebagai titik acu dari penghitungan nilai *likelihood*, dan
    c. *Dictionary* yang berisikan nilai rata-rata dan standar deviasi untuk setiap kolom data numerik dan setiap klasifikasi nilai target. *Dictionary* inilah yang kelak digunakan untuk menghitung nilai probabilitas menggunakan persamaan distribusi normal.
2. Untuk melakukan proses validasi, data hasil *training* tersebut digunakan untuk melakukan prediksi terhadap data pada dataset *valid*.
3. Proses dimulai dengan melakukan iterasi terhadap setiap baris pada dataset *valid*. Untuk setiap baris data pada *valid*, pisahkan kolom *target* dan kolom *feature* lainnya selain kolom target.
4. Simpan nilai awal untuk masing-masing klasifikasi nilai target yaitu sebesar 1. Nilai mula-mula inilah yang lantas akan digunakan sebagai nilai probabilitas yang digunakan untuk menentukan hasil akhir nilai target.
5. Untuk setiap nilai kolom pada baris data tersebut dan klasifikasi nilai target tertentu, lakukan iterasi perkalian nilai mula-mula pada poin 4 dengan nilai *likelihood*nya. Setelah iterasi selesai dilakukan, lakukan perkalian nilai hasil iterasi perkalian tersebut dengan probabilitas kemunculan nilai target. Untuk lebih lengkapnya, silakan perhatikan kembali gambar 3.2.
6. Nilai *likelihood* yang digunakan disesuaikan dengan kolom yang sedang dikalikan. Nilai *likelihood* dengan rumus probabilitas Bayes digunakan untuk data

boolean dan nilai probabilitas dengan rumus distribusi normal digunakan untuk data numerik guna merepresentasikan nilai *likelihood*.

7. Lakukan poin 5 untuk setiap nilai mula-mula yang ada pada poin 4. Mengingat terdapat empat klasifikasi nilai target (0, 1, 2, 3), maka di akhir poin 7 akan terdapat empat nilai hasil iterasi perkalian.
8. Tentukan nilai maksimum dari keempat nilai hasil iterasi perkalian tersebut. Klasifikasi target yang memiliki hasil iterasi perkalian maksimum lah yang akan menjadi nilai akhir dari prediksi baris tersebut.
9. Hitung banyak ketidaksamaan nilai target pada hasil prediksi dan nilai target pada *dataset valid* untuk menghitung nilai *error* pada prediksi.
10. Untuk melakukan prediksi dengan *dataset test*, ulangi proses pada poin 3 hingga poin 8, tetapi alih-alih menggunakan *dataset valid*, *dataset test* lah yang dimasukkan ke dalam proses.

3.3. Implementasi Algoritma menggunakan *library*

Dalam penerapan implementasi algoritma Naive Bayes menggunakan library yang sudah tersedia*,* langkah - langkah yang penulis lakukan adalah sebagai berikut :

1. Melakukan *import library* GaussianNB dari Scikit-Learn
2. Membaca *dataset training* yang berada dalam *file csv*
3. Membaca *dataset valid* yang berada dalam file *csv* dan akan digunakan untuk diuji terhadap *dataset training*.
4. Memisahkan *dataset training* yang hanya berisi kolom target (“price_range”), yang diberi nama ‘X_test’; dengan kolom lain, yang diberi nama ‘X_train’.
5. Memisahkan *dataset valid* yang hanya berisi kolom target (“price_range”), yang diberi nama ‘y_test’; dengan kolom lain, yang diberi nama ‘y_train’.
6. Membuat model Naive Bayes dari ‘X_train’ sebagai data latih dan ‘X_test’ sebagai data uji.
7. Memeriksa skor akurasi dari hasil model yang sudah dihasilkan.

Bab IV
Hasil Implementasi

4.1. Pengukuran Performansi Model

Pengukuran performansi model dilakukan pada dataset validation dengan menggunakan bantuan library classification report dari Scikit Learn Metrics. Hasil yang diprediksi pada setiap model akan diuji dengan label validasi yang sudah tersedia.

a. KNN tanpa menggunakan library

	precision	recall	f1-score	support
0	0.96	0.97	0.97	142
1	0.91	0.93	0.92	144
2	0.91	0.92	0.91	155
3	0.99	0.93	0.96	159
accuracy			0.94	600
macro avg	0.94	0.94	0.94	600
weighted avg	0.94	0.94	0.94	600

b. KNN dengan menggunakan library

	precision	recall	f1-score	support
0	0.96	0.97	0.97	142
1	0.91	0.93	0.92	144
2	0.91	0.92	0.91	155
3	0.99	0.93	0.96	159
accuracy			0.94	600
macro avg	0.94	0.94	0.94	600
weighted avg	0.94	0.94	0.94	600

c. Naive Bayes tanpa menggunakan library

	precision	recall	f1-score	support
0	0.87	0.87	0.87	142
1	0.66	0.64	0.65	144
2	0.68	0.71	0.70	155
3	0.90	0.89	0.90	159
accuracy			0.78	600
macro avg	0.78	0.78	0.78	600
weighted avg	0.78	0.78	0.78	600

d. Naive Bayes dengan menggunakan library

	precision	recall	f1-score	support
0	0.87	0.87	0.87	142
1	0.66	0.64	0.65	144
2	0.68	0.71	0.70	155
3	0.90	0.89	0.90	159
accuracy			0.78	600
macro avg	0.78	0.78	0.78	600
weighted avg	0.78	0.78	0.78	600

4.2. Analisis Perbandingan Algoritma

Seperti yang tertera pada bagian 4.1, penulis sudah berhasil mengimplementasikan model pembelajaran KNN dan Naive Bayes yang memiliki performa yang sama dengan model bersesuaian yang sudah tersedia pada Scikit Learn. Performa yang sama ini dapat menunjukkan bahwa model yang ada pada

Scikit Learn memiliki bentuk yang mirip dengan model yang kami bangun secara *scratch*.

Untuk perbandingan 2 model pembelajaran, yaitu antara KNN dan Naive Bayes, kita dapat melihat bahwa KNN memiliki performansi yang lebih tinggi dibandingkan dengan Naive Bayes. Hal ini ditunjukkan oleh hasil metrik pengukuran performansi yang ditampilkan pada bagian sebelumnya. Hal ini tentunya disebabkan oleh perbedaan strategi pembelajaran yang dilakukan antara Naive Bayes dan KNN. KNN mencari data terdekat yang paling mirip dengan data *test*. Naive Bayes melakukan pembelajaran dari probabilitas kemunculannya pada data *train*. Secara umum, tentunya KNN lebih dapat melakukan prediksi dengan lebih baik karena hal ini disebabkan karena KNN dapat melakukan prediksi data baru dengan hanya melihat jarak terdekat, sebaliknya Naive Bayes akan mengalami kesulitan ketika bertemu data baru yang tidak tersedia pada data *train* (probabilitas kemunculannya akan langsung bernilai 0 sehingga model ini langsung mengeluarkan pilihan terkait).

Namun dalam segi waktu pemrosesan, Naive Bayes memerlukan waktu yang lebih sedikit untuk memprediksi data. Hal ini disebabkan karena bentuk model Naive Bayes yang berbentuk probabilitas. Hal ini memungkinkan proses *learning* bisa dilakukan hanya satu kali. Setiap *request* prediksi hanya akan menggunakan hasil learning yang sudah ada. Sebaliknya, KNN memerlukan waktu yang relatif lebih lama karena model KNN termasuk *lazy learner*. Model yang disimpan hanyalah data yang menjadi data *train* dari model KNN tersebut. Model akan melakukan komparasi secara sekuensial ke seluruh data yang ada pada data *train* setiap kali adanya *request* prediksi. Hal ini tentunya membutuhkan waktu, apalagi jika data *train* memiliki jumlah *record* yang cukup besar.

4.3. Submisi Kaggle

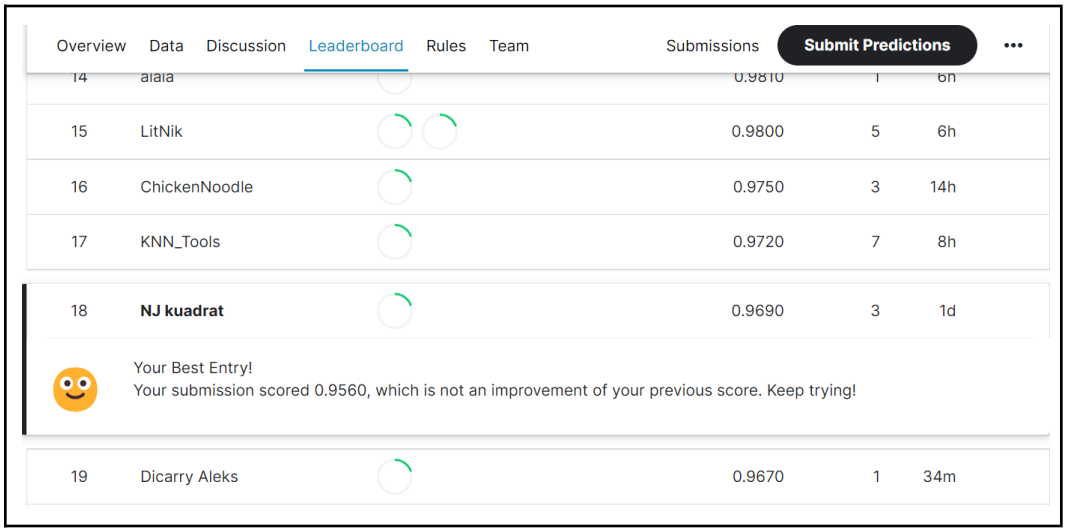
a. Tahapan Submisi

Dalam memenuhi spesifikasi bonus, kami menggunakan algoritma KNN yang kami buat tanpa menggunakan *library* untuk dikumpulkan ke Kaggle. Berikut merupakan tahapan yang kami lakukan.

1. Membaca *dataset training* yang berada dalam *file csv* ‘data_training.csv’
2. Membaca *dataset valid* yang berada dalam file *csv* ‘data_validation.csv’
3. Mengunduh *dataset test* yang berada pada *website* Kaggle bernama ‘test.csv’. Memisahkan kolom ‘id’ dengan kolom lainnya.

4. Menggabungkan *dataset training* dan *valid* sebagai data latih, serta *dataset test* sebagai data uji
5. Melakukan pemrosesan data dengan algoritma KNN tanpa *library*
6. Hasil yang didapatkan lalu digabungkan dengan kolom 'id' pada *dataset test* lalu dituliskan ke dalam *file csv* 'result.csv'
7. *File* 'result.csv' dikumpulkan ke Kaggle

b. Hasil



Bab V
Penutup

5.1. Kesimpulan

Dari program yang sudah dibuat oleh penulis, terdapat beberapa kesimpulan yang dapat ditarik, yaitu :

1. Model yang dibangun dengan algoritma K *nearest neighbor* memiliki tingkat akurasi yang lebih tinggi jika dibandingkan dengan model yang dibangun dengan algoritma *Naive Bayes*
2. Model yang dibangun dengan algoritma K *nearest neighbor* memiliki waktu pengujian yang lebih lama jika dibandingkan dengan model yang dibangun dengan algoritma *Naive Bayes*

5.2. Saran

Setelah melakukan analisis dan pengujian terkait algoritma *K nearest neighbor* dan algoritma *Naive Bayes,* terdapat beberapa saran yang dapat diberikan oleh penulis, yaitu :

1. Hanya menggunakan *dataset training* memiliki kualitas lebih baik agar prediksi yang dibuat lebih akurat. Contoh *dataset* yang berkualitas adalah *dataset* yang memiliki data yang beragam dan memiliki sedikit *missing values*.
2. Melakukan penelitian lebih lanjut terkait pembobotan kolom - kolom pada dataset agar menghasilkan prediksi yang lebih akurat untuk *K nearest neighbor.*

Daftar Pustaka

Air Force Institute of Technology. (n.d.). *Naïve Bayes Classifier*. Retrieved November 26, 2023, from https://afit-r.github.io/naive_bayes

Asahar, J. T., Yanosma, D., & Anggriani, K. (2016). *IMPLEMENTASI METODE K-NEAREST NEIGHBOR (KNN) DAN SIMPLE ADDITIVE WEIGHTING (SAW) DALAM PENGAMBILAN KEPUTUSAN SELEKSI PENERIMAAN ANGGOTA PASKIBRAKA*. https://ejournal.unib.ac.id/index.php/pseudocode/article/view/1039

Chen, J. (2023, March 30). *Normal Distribution: What It Is, Properties, Uses, and Formula*. https://www.investopedia.com/terms/n/normaldistribution.asp

KK IF Teknik Informatika STEI ITB. (n.d.). *Modul: Supervised Learning Naive Bayes*. Retrieved November 27, 2023, from https://cdn-edunex.itb.ac.id/53145-Artificial-Intelligence-Parallel-Class/210071-Supervised-Learning/90133-Supervised-Learning/1699250380758_IF3170_Materi09_Seg02_AI-NaiveBayes.pdf

Malhotra, K. R. (2016). *Feature Selection and KNN Classification*. https://www.kaggle.com/code/dom12345/feature-selection-and-knn-classification

Pateljay731. (2021). *Prediction with the KNN Classifier*. https://www.kaggle.com/code/pateljay731/prediction-with-the-knn-classifier

University of Cincinnati. (n.d.). *Naïve Bayes Classifier*. Retrieved November 29, 2023, from https://uc-r.github.io/naive_bayes

Kontribusi Anggota

NIM	Nama	Kontribusi
13521100	Alexander Jason	Perbaikan Performansi KNN KNN *ScikitLearn* (*Library)* Submisi Kaggle Laporan
13521116	Juan Christopher Santoso	Naive Bayes *Scratch* Laporan
13521139	Nathania Calista Djunaedi	KNN *Scratch* Laporan
13521162	Antonio Natthan Krishna	Naive Bayes *ScikitLearn* (*Library)* Refactor model *Scratch* *Save Load* model Laporan

Lampiran

Github: https://github.com/AJason36/IF3170_Tubes2
Jupyter: Tubes2_AI.ipynb