

**Tugas Besar II IF3170 Inteligensi Buatan**

**Implementasi Algoritma Pembelajaran Mesin**



Oleh:

Fazel Ginanda 13521098

Akhmad Setiawan 13521164

Irgiansyah Mondo 13521167

Satria Octavianus Nababan 13521168

**Program Studi Teknik Informatika**

**Sekolah Teknik Elektro dan Informatika**

**Institut Teknologi Bandung**

**2023**

## 1. Penjelasan Implementasi KNN

Langkah pertama adalah membaca data latih dari *file* dan menyimpannya ke dalam variabel. Data latih tersebut dipisah menjadi fitur dan label kemudian disimpan dalam variabel yang menggunakan struktur data *list*. Variabel tersebut diberi nama `train_features` dan `train_label`. Sebelum disimpan, setiap nilai dari fitur data latih diubah tipe datanya menjadi *integer* dan *float*. Kemudian, hal yang sama juga dilakukan terhadap data validasi. Variabel yang menyimpan data validasi tersebut adalah `test_features` dan `test_label`.

Langkah selanjutnya adalah menghitung jarak antara setiap *instance* dari data validasi dengan seluruh *instance* pada data latih. Besaran jarak ditentukan dengan *euclidean distance*. Kemudian, dibentuk sebuah list yang setiap elemennya terdiri dari pasangan *instance* dari data latih beserta nilai jarak yang sudah dihitung sebelumnya. List tersebut diurutkan menaik berdasarkan jarak. Kemudian, dari list tersebut diambil sebanyak *k* elemen pertama yang merupakan *k* tetangga terdekat dari sebuah *instance* data validasi.

Setelah *k* tetangga terdekat dari sebuah *instance* data latih diperoleh, berikutnya ditentukan label yang sesuai untuk *instance* tersebut. Penentuan label berdasarkan label yang paling banyak muncul dari *k* tetangga terdekat *instance* tersebut. Setiap prediksi label dari data validasi disimpan dalam *list predictions*. Terakhir, dihitung akurasi dari algoritma dengan cara membandingkan setiap elemen dari *predictions* dengan elemen yang bersesuaian di *validation\_label*. Nilai akurasi yang didapatkan tersebut ditampilkan sebagai keluaran dari algoritma ini.

## 2. Penjelasan Implementasi Naive-Bayes

Implementasi Naive Bayes dilakukan dengan menggunakan fungsi-fungsi berikut.

### 1. `load_data(file_path):`

Fungsi ini digunakan untuk membaca data dari file CSV menggunakan `csv_reader`. Setelah di-*load*, dilakukan juga pemisahan antara kolom-kolom fitur dan label target.

**2. `separate_by_class(features, labels)`:**

Fungsi ini digunakan untuk mengelompokkan data dari kolom-fitur berdasarkan kelas target (`price_range`) menggunakan *dictionary*, sehingga terdapat 4 key (0, 1, 2, 3) dengan masing-masing memiliki *values* berupa baris dari data-data yang berkaitan.

**3. `summarize_data(data)`:**

Fungsi ini digunakan untuk menghitung statistik untuk fitur numerik (*mean* dan *standard deviation*) berdasarkan kelompok kelas targetnya untuk nanti digunakan saat mencari probabilitas  $P(X_{\text{input}} | \text{class\_target})$ . Selain itu pada data non-numerik (kategorikal), dicari probabilitas kemunculan masing-masing nilai 1 dan 0 (digunakan untuk mencari  $P(1 | \text{class\_target})$  dan  $P(0 | \text{class\_target})$  untuk keempat *class\_target*).

**4. `train_naive_bayes(train_file_path)`:**

Fungsi ini digunakan untuk memanggil data latih yakni `data_train.csv` (`load_data`), memisahkan data latih berdasarkan kelas target (`separate_by_class`), menghitung statistik untuk setiap kelompok kelas target (`summarize_data`), lalu menyimpannya sebagai model (statistik) dalam *file pickle*.

**5. `calculate_numerical_probability(x, mean, stdev)`:**

Fungsi ini digunakan untuk menghitung probabilitas Gaussian untuk fitur-fitur numerik.

$$P(x_i | y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp \left( -\frac{(x_i - \mu_y)^2}{2\sigma_y^2} \right)$$

**Gambar 1** Probabilitas Gaussian

Sumber: [What is Naive Bayes Classifier? \[Explained With Example\] | upGrad blog](#)

**6. calculate\_class\_probabilities(summaries, input):**

Menghitung probabilitas kemungkinan pengkategorian kelas untuk sebuah *input instance* baru berdasarkan perhitungan perkalian probabilitas *Gaussian* dan kategorikal sesuai dengan kelasnya masing-masing. Setiap kelas akan memperoleh masing-masing probabilitas seberapa mungkin sebuah *instance* akan masuk ke kategori kelas tersebut.

**7. predict(summaries, input):**

Memprediksi kelas untuk *input instance* baru berdasarkan probabilitas kelas yang telah dikalkulasi pada kelas sebelumnya. Prediksi dilakukan dengan membandingkan probabilitas setiap kelas dan dipilih probabilitas yang tertinggi.

**8. load\_naive\_bayes\_model(model\_file\_path):**

Fungsi ini digunakan untuk memuat model Naive Bayes dari file pickle yang ditentukan.

**9. evaluate\_model(model, validation\_data, target\_labels):**

Mengevaluasi model dengan memprediksi kelas untuk data *data\_validation* berdasarkan model yang telah dibuat sebelumnya, serta menghitung akurasi prediksinya.

Adapun urutan eksekusi utamanya:

- Memuat data latih dari file CSV.
- Melatih model Naive Bayes.
- Memuat data validasi.
- Memuat model yang telah dilatih.
- Mengevaluasi model dan mencetak akurasi.

### 3. Perbandingan Hasil Prediksi dari Implementasi Algoritma dengan Hasil yang Didapatkan dari Penggunaan Pustaka

Perbandingan hasil prediksi pada implementasi manual dengan implementasi menggunakan *library scikit-learn* dihitung berdasarkan tingkat akurasi.

#### 1. KNN-Algorithm

Pada implementasi Algoritma KNN, akurasi yang diperoleh dari implementasi *from scratch* tidak jauh berbeda dengan implementasi menggunakan pustaka *scikit-learn*, yakni didapatkan akurasi berada pada rentang 91%-93% (tergantung nilai  $k$ ). Berdasarkan percobaan yang kami lakukan, nilai  $k$  yang menghasilkan akurasi paling baik adalah 21 dengan akurasi sebesar 93,33% sebagaimana yang diperlihatkan di bawah ini.

```
PS C:\Users\ASUS\Documents\Python Notebook\Tubes2_AI\src> python -u "c:\Users\ASUS\Documents\Python Notebook\Tubes2_AI\src\knn.py"
Predictions: [1 2 3 0 2 1 2 0 3 2 3 2 3 0 3 0 2 1 0 2 3 2 0 1 2 0 3 1 0 3 1 3 3 0 2 3 1
3 2 1 1 3 0 2 2 1 1 2 2 3 1 2 3 0 1 3 2 3 3 2 2 3 3 1 3 2 3 2 3 3 2 3 1 0
1 2 0 3 1 0 3 3 0 2 3 1 3 3 0 2 1 1 1 2 2 1 3 2 0 3 3 3 1 2 3 1 3 3 3 3
3 2 1 3 1 0 2 1 3 1 2 3 3 2 0 2 2 1 3 3 1 0 0 3 0 0 2 3 0 1 3 3 1 2 3 1 2
1 2 3 0 0 2 1 1 2 0 0 3 3 3 0 2 3 0 0 1 0 2 2 1 0 1 0 1 3 1 2 0 3 1 1 2 0
2 0 3 2 0 3 2 1 0 2 0 1 3 3 1 1 2 2 3 2 3 3 3 0 2 1 2 3 1 1 2 2 0 0 2 2 1
2 3 1 2 0 3 0 2 2 2 2 3 0 1 3 3 3 0 0 0 1 3 3 1 1 0 2 2 1 3 3 0 2 2 0 0
1 0 2 2 3 0 1 3 3 0 0 2 1 1 1 1 3 2 2 2 0 1 0 2 1 1 0 1 3 1 3 0 1 0 1 3
3 3 0 3 1 0 0 1 2 2 0 3 1 1 0 2 3 1 3 2 3 1 2 3 0 0 3 1 2 1 0 1 1 0 3 2 3
2 2 2 3 0 1 0 1 1 0 2 2 2 2 2 1 1 3 0 2 2 2 2 1 3 2 0 3 0 0 2 2 0 3 2 1 2
0 0 2 2 1 3 0 0 2 2 0 2 1 1 2 2 3 3 1 0 0 3 1 0 3 1 0 2 0 3 1 0 1 0 1 2 1
1 1 2 3 1 2 2 3 1 1 2 2 2 0 1 3 0 0 2 0 0 0 3 0 1 1 0 0 2 1 0 3 2 1 0 1 1
2 1 1 1 2 0 1 2 0 2 0 3 0 0 1 2 2 1 3 1 1 3 3 0 2 2 0 1 2 2 0 1 2 0 2 0 2
3 0 0 3 0 1 3 3 3 1 2 3 0 1 2 1 2 1 0 3 1 3 3 2 1 3 0 3 3 1 3 0 3 0 2 2 0
2 3 1 3 0 2 3 3 0 2 1 2 3 2 3 1 2 0 2 3 3 1 1 3 1 0 0 0 3 0 1 2 1 3 2 1 2
3 0 3 0 0 3 3 2 0 1 2 0 2 0 1 0 2 0 1 0 0 1 3 0 0 1 1 0 0 2 0 1 3 0 2 1 1
0 2 3 1 3 0 2 2]
Accuracy: 93.33333333333333%
```

**Gambar 2** Hasil Prediksi dan Akurasi KNN Implementasi Manual  
( $k=21$ )

```

PS C:\Users\ASUS\Documents\Python Notebook\Tubes2_AI> python -u "c:\Users\ASUS\Documents\Python Notebook\Tubes2_AI\src\knn_sklearn.py"
The predictions are:
['1' '2' '3' '0' '2' '1' '2' '0' '3' '2' '3' '2' '3' '0' '3' '0' '2' '1'
'0' '2' '3' '2' '0' '1' '2' '0' '3' '1' '0' '3' '1' '3' '3' '0' '2' '3'
'1' '3' '2' '1' '1' '3' '0' '2' '2' '1' '1' '2' '2' '3' '1' '2' '3' '0'
'1' '3' '2' '3' '3' '2' '2' '3' '3' '1' '3' '2' '3' '2' '3' '3' '2' '3'
'1' '0' '1' '2' '0' '3' '1' '0' '3' '3' '0' '2' '3' '1' '3' '3' '0' '2'
'1' '1' '1' '2' '2' '1' '3' '2' '0' '3' '3' '3' '1' '2' '3' '1' '3' '3'
'3' '3' '3' '3' '2' '1' '3' '1' '0' '2' '1' '3' '1' '2' '3' '3' '2' '0'
'2' '2' '1' '3' '3' '1' '0' '0' '3' '0' '0' '2' '3' '0' '1' '3' '3' '1'
'2' '3' '1' '2' '1' '2' '3' '0' '0' '2' '1' '1' '2' '0' '0' '3' '3' '3'
'0' '2' '3' '0' '0' '1' '0' '2' '2' '1' '0' '1' '0' '1' '3' '1' '2' '0'
'3' '1' '1' '2' '0' '2' '0' '3' '2' '0' '3' '2' '1' '0' '2' '0' '1' '3'
'3' '1' '1' '2' '2' '3' '2' '3' '3' '3' '0' '2' '1' '2' '3' '1' '1' '2'
'2' '0' '0' '2' '2' '1' '2' '3' '1' '2' '0' '3' '0' '2' '2' '2' '2' '2'
'3' '0' '1' '3' '3' '3' '0' '0' '0' '1' '3' '3' '1' '1' '0' '2' '2' '1'
'3' '3' '0' '2' '2' '0' '0' '1' '0' '2' '2' '3' '3' '0' '1' '3' '3' '0'
'0' '2' '1' '1' '1' '1' '3' '2' '2' '2' '0' '1' '0' '2' '1' '1' '0' '1'
'3' '1' '3' '0' '1' '0' '1' '3' '3' '3' '0' '3' '1' '0' '0' '1' '2' '2'
'0' '3' '1' '1' '0' '2' '3' '1' '3' '2' '3' '1' '2' '3' '0' '0' '3' '1'
'2' '1' '0' '1' '1' '0' '3' '2' '3' '2' '2' '2' '3' '0' '1' '0' '1' '1'
'0' '2' '2' '2' '2' '2' '1' '1' '3' '0' '2' '2' '2' '2' '1' '3' '2' '0'
'3' '0' '0' '2' '2' '0' '3' '2' '1' '2' '0' '0' '2' '2' '1' '3' '0' '0'
'2' '2' '0' '2' '1' '1' '2' '2' '3' '3' '1' '0' '0' '3' '1' '0' '3' '1'
'0' '2' '0' '3' '1' '0' '1' '0' '1' '2' '1' '1' '1' '2' '3' '1' '2' '2'
'3' '1' '1' '2' '2' '2' '0' '1' '3' '0' '0' '2' '0' '0' '0' '3' '0' '1'
'1' '0' '0' '2' '1' '0' '3' '2' '1' '0' '1' '1' '2' '1' '1' '1' '2' '0'
'1' '2' '0' '2' '0' '3' '0' '0' '1' '2' '2' '1' '3' '1' '1' '3' '3' '0'
'2' '2' '0' '1' '2' '2' '0' '1' '2' '0' '2' '0' '2' '3' '0' '0' '3' '0'
'1' '3' '3' '3' '1' '2' '3' '0' '1' '2' '1' '2' '1' '0' '3' '1' '3' '3'
'2' '1' '3' '0' '3' '3' '1' '3' '0' '3' '0' '2' '2' '0' '2' '3' '1' '3'
'0' '2' '3' '3' '0' '2' '1' '2' '3' '2' '3' '1' '2' '0' '2' '3' '3' '1'
'1' '3' '1' '0' '0' '0' '3' '0' '1' '2' '1' '3' '2' '1' '2' '3' '0' '3'
'0' '0' '3' '3' '2' '0' '1' '2' '0' '2' '0' '1' '0' '2' '0' '1' '0' '0'
'1' '3' '0' '0' '1' '1' '0' '0' '2' '0' '1' '3' '0' '2' '1' '1' '0' '2'
'3' '1' '3' '0' '2' '2']
Accuracy of KNN with k=21 is 93.33333333333333

```

**Gambar 3** Hasil Prediksi dan Akurasi KNN Implementasi Pustaka sklearn (k=21)

*Insight* yang diperoleh, implementasi dari algoritma KNN secara manual telah mendekati baik, karena mampu menyamai implementasi dengan menggunakan bantuan pustaka *scikit-learn*. Selain itu, dari tingkat akurasi, implementasi algoritma ini cukup baik diterapkan karena memiliki akurasi yang lebih tinggi dibanding dengan Naive Bayes di bawah.

## 2. Naive Bayes-Algorithm

Pada implementasi Naive Bayes-Algorithm, akurasi yang diperoleh antara implementasi manual dengan implementasi menggunakan *library* cenderung tidak jauh berbeda, yakni berada di angka 78% - 79%

```

PS C:\Users\Lenovo\Documents\AI\Tubes2_AI\src> python3 naive_bayes.py
[2, 2, 3, 0, 3, 1, 3, 0, 3, 1, 3, 2, 3, 0, 3, 0, 2, 1, 0, 2, 3, 1, 0, 1, 1, 1, 2, 2, 0, 2, 2, 3, 3, 1, 2, 3,
1, 3, 1, 1, 1, 3, 0, 2, 2, 1, 1, 2, 1, 3, 1, 2, 3, 0, 1, 3, 2, 3, 3, 2, 2, 3, 3, 1, 3, 2, 3, 2, 2, 3, 2, 3,
1, 0, 1, 2, 0, 3, 1, 0, 3, 3, 0, 2, 3, 1, 3, 3, 0, 3, 1, 1, 1, 2, 1, 1, 3, 2, 1, 3, 3, 3, 1, 2, 3, 2, 3, 3,
3, 3, 3, 3, 2, 2, 3, 2, 0, 3, 1, 3, 1, 2, 2, 3, 3, 1, 2, 2, 1, 3, 3, 1, 0, 0, 3, 0, 0, 1, 3, 0, 1, 3, 3, 1,
2, 3, 1, 2, 1, 2, 3, 1, 0, 2, 1, 0, 3, 0, 0, 3, 3, 3, 0, 2, 3, 0, 0, 1, 1, 2, 3, 1, 0, 1, 1, 1, 3, 0, 2, 0,
3, 1, 1, 2, 0, 2, 0, 3, 2, 0, 3, 2, 0, 1, 2, 0, 1, 3, 3, 1, 2, 2, 2, 3, 2, 3, 3, 0, 2, 0, 2, 3, 1, 2, 3,
2, 0, 0, 2, 2, 2, 3, 3, 0, 2, 0, 3, 0, 2, 1, 2, 2, 1, 3, 0, 1, 3, 3, 3, 0, 0, 1, 1, 3, 3, 1, 2, 0, 2, 1, 2,
3, 3, 0, 2, 1, 1, 1, 2, 0, 3, 2, 3, 3, 0, 1, 3, 3, 0, 0, 2, 1, 1, 1, 0, 3, 2, 2, 2, 0, 1, 0, 1, 1, 1, 0, 1,
3, 0, 3, 0, 1, 0, 1, 3, 2, 2, 0, 2, 1, 0, 0, 1, 2, 2, 0, 3, 1, 1, 0, 2, 3, 1, 3, 1, 3, 1, 2, 2, 0, 0, 3, 1,
2, 1, 0, 1, 1, 0, 3, 2, 3, 2, 2, 3, 3, 0, 1, 0, 1, 2, 0, 2, 2, 2, 2, 3, 1, 2, 3, 0, 2, 1, 3, 3, 2, 3, 2, 0,
3, 0, 0, 3, 3, 0, 3, 2, 1, 2, 1, 0, 1, 2, 1, 3, 0, 0, 2, 1, 0, 2, 1, 0, 3, 2, 3, 3, 1, 0, 0, 3, 1, 0, 3, 0,
0, 2, 0, 3, 1, 0, 1, 1, 2, 0, 1, 1, 3, 3, 1, 2, 2, 3, 1, 1, 2, 2, 2, 0, 1, 3, 0, 0, 1, 0, 0, 0, 3, 1, 1,
2, 0, 0, 2, 1, 0, 3, 2, 1, 0, 2, 0, 2, 1, 1, 1, 2, 0, 1, 2, 0, 2, 0, 3, 0, 0, 1, 2, 1, 1, 3, 2, 1, 2, 3, 0,
2, 2, 0, 2, 2, 0, 1, 2, 0, 2, 0, 1, 3, 0, 0, 3, 0, 2, 3, 3, 1, 3, 3, 0, 1, 1, 1, 3, 1, 0, 3, 1, 3, 3,
1, 2, 3, 0, 3, 3, 1, 2, 0, 3, 0, 2, 2, 0, 2, 3, 0, 3, 0, 2, 3, 3, 0, 3, 1, 2, 3, 3, 3, 1, 2, 0, 3, 3, 3, 2,
2, 3, 1, 1, 0, 3, 0, 2, 1, 1, 3, 2, 1, 1, 3, 0, 3, 0, 0, 3, 3, 3, 1, 2, 2, 0, 2, 0, 1, 0, 2, 0, 2, 0, 0,
2, 3, 0, 1, 2, 1, 0, 1, 2, 0, 1, 3, 0, 2, 1, 1, 0, 1, 3, 1, 3, 0, 1, 3]
Accuracy: 79.33333333333333%

```

**Gambar 4** Hasil Prediksi dan Akurasi Naive Bayes Implementasi

Manual

```

PS C:\Users\Lenovo\Documents\AI\Tubes2_AI\src> python3 naive_bayes_sklearn.py
['2' '2' '3' '0' '3' '1' '3' '0' '3' '1' '3' '2' '3' '0' '3' '0' '2' '1'
'0' '2' '3' '1' '0' '1' '1' '1' '2' '2' '0' '2' '2' '3' '3' '0' '2' '3'
'2' '3' '1' '1' '0' '3' '0' '2' '2' '1' '1' '2' '1' '3' '1' '2' '3' '0'
'1' '3' '2' '3' '3' '2' '2' '3' '3' '1' '3' '2' '2' '2' '2' '3' '2' '3'
'1' '0' '1' '2' '0' '3' '1' '0' '3' '3' '0' '2' '3' '1' '3' '3' '0' '2'
'1' '1' '1' '2' '1' '1' '3' '2' '1' '3' '3' '3' '1' '2' '3' '2' '3' '2'
'3' '3' '3' '3' '2' '2' '3' '2' '0' '3' '1' '3' '1' '2' '2' '3' '3' '1'
'2' '2' '1' '3' '3' '1' '0' '3' '0' '0' '1' '3' '0' '1' '3' '0' '3' '1'
'2' '3' '1' '2' '1' '2' '3' '1' '0' '2' '1' '0' '3' '0' '0' '3' '3' '3'
'0' '2' '3' '0' '1' '1' '0' '2' '3' '1' '0' '1' '1' '1' '3' '0' '2' '0'
'3' '1' '1' '2' '0' '2' '0' '3' '2' '0' '3' '2' '0' '1' '2' '0' '1' '3'
'3' '1' '2' '2' '2' '3' '2' '3' '3' '3' '0' '2' '0' '2' '3' '1' '2' '3'
'2' '0' '0' '2' '2' '2' '3' '3' '0' '2' '0' '3' '0' '2' '2' '2' '2' '1'
'3' '0' '1' '3' '3' '0' '0' '1' '1' '3' '3' '2' '2' '0' '2' '1' '2'
'2' '3' '0' '2' '2' '1' '1' '2' '0' '3' '2' '3' '3' '3' '0' '1' '3' '0'
'0' '2' '1' '1' '1' '0' '3' '2' '2' '2' '0' '1' '0' '1' '1' '1' '0' '0'
'3' '0' '3' '0' '1' '0' '1' '3' '2' '2' '0' '2' '1' '0' '0' '1' '2' '2'
'0' '3' '1' '1' '0' '2' '3' '1' '3' '1' '3' '1' '2' '2' '0' '0' '2' '1'
'2' '1' '0' '1' '1' '0' '2' '2' '3' '2' '2' '3' '3' '0' '1' '0' '1' '2'
'0' '2' '2' '2' '2' '3' '1' '2' '3' '0' '2' '1' '3' '3' '2' '3' '2' '0'
'3' '0' '0' '3' '2' '0' '3' '2' '1' '2' '1' '0' '1' '2' '1' '3' '0' '0'
'2' '1' '0' '2' '1' '0' '3' '2' '3' '3' '1' '0' '0' '3' '1' '0' '3' '0'
'0' '2' '0' '3' '1' '0' '1' '1' '1' '2' '0' '1' '1' '3' '3' '1' '2' '2'
'3' '1' '1' '2' '2' '2' '0' '1' '3' '0' '0' '1' '0' '0' '0' '3' '1' '1'
'2' '0' '0' '2' '1' '0' '3' '2' '1' '0' '2' '0' '2' '1' '2' '1' '2' '0'
'1' '2' '0' '2' '0' '3' '0' '0' '1' '2' '1' '1' '3' '2' '1' '2' '3' '0'
'2' '2' '0' '2' '2' '2' '0' '1' '2' '0' '2' '0' '1' '3' '0' '0' '3' '0'
'2' '2' '3' '3' '1' '3' '3' '0' '1' '2' '1' '3' '1' '0' '3' '1' '3' '3'
'1' '2' '3' '0' '3' '3' '1' '2' '0' '3' '0' '2' '2' '0' '2' '3' '0' '3'
'0' '2' '3' '3' '0' '2' '1' '2' '3' '3' '3' '1' '2' '0' '3' '2' '3' '2'
'2' '3' '1' '1' '0' '0' '3' '0' '2' '1' '1' '3' '2' '1' '1' '3' '0' '3'
'0' '0' '3' '3' '3' '1' '2' '2' '0' '2' '0' '1' '0' '2' '0' '2' '0' '0'
'2' '3' '0' '1' '2' '1' '0' '1' '2' '0' '1' '3' '0' '2' '1' '1' '0' '1'
'3' '1' '3' '0' '1' '3']
Accuracy: 78.16666666666666%

```

**Gambar 5** Hasil Prediksi dan Akurasi Naive Bayes Implementasi Pustaka

sklearn

Insight yang diperoleh, berarti implementasi algoritma Naive Bayes yang dibuat secara manual sudah cukup baik, karena dibandingkan dengan menggunakan pustaka *scikit-learn* tidak menghasilkan akurasi yang jauh berbeda. Namun demikian, tingkat akurasinya tidak sebagus algoritma KNN, namun masih cukup baik karena mendekati 80%.

#### 4. Pembagian Tugas

NIM	Nama	Tugas
13521098	Fazel Ginanda	KNN-Algorithm
13521164	Akhmad Setiawan	Naive Bayes-Algorithm
13521167	Irgiansyah Mondo	Naive Bayes-Algorithm
13521168	Satria Octavianus Nababan	KNN-Algorithm

#### 5. Repository Github

[https://github.com/fazelginanda/Tubes2\\_AI](https://github.com/fazelginanda/Tubes2_AI)