

**LAPORAN TUGAS BESAR II
IF3170 INTELEGENSI BUATAN**

**IMPLEMENTASI ALGORITMA
KNN dan Naive-Bayes**



DISUSUN OLEH :

13521044 RACHEL GABRIELA CHEN

13521046 JEFFREY CHOW

13521094 ANGELA LIVIA ARUMSARI

13521134 RINALDY ADIN

**PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG**

2023

DAFTAR ISI

DAFTAR ISI.....	2
DAFTAR TABEL.....	3
IMPLEMENTASI KNN.....	4
IMPLEMENTASI NAIVE-BAYES.....	6
PERBANDINGAN HASIL IMPLEMENTASI DAN PUSTAKA.....	9
SUBMISI KAGGLE.....	11
KONTRIBUSI KELOMPOK.....	13
REPOSITORY.....	14

DAFTAR TABEL

Tabel 1. Atribut Kelas KNeighborsClassifier.....	4
Tabel 2. Method Kelas KNeighborsClassifier.....	4
Tabel 3. Atribut Kelas NaiveBayesGaussian.....	6
Tabel 4. Method Atribut Kelas NaiveBayesGaussian.....	6
Tabel 5. Atribut Kelas NaiveBayesCategorical.....	7
Tabel 6. Method Kelas NaiveBayesCategorical.....	8
Tabel 7. Performance Score KNN.....	9
Tabel 8. Performance Score Naive Bayes.....	9

IMPLEMENTASI KNN

Algoritma K-Nearest Neighbors (KNN) diimplementasikan dalam kelas `KNeighborsClassifier`. Daftar atribut dari `class` ini adalah sebagai berikut.

Tabel 1. Atribut Kelas `KNeighborsClassifier`

Nama Atribut	Keterangan
<code>n_neighbors</code>	Jumlah <i>nearest neighbor</i> yang diambil dalam melakukan prediksi
<code>metric</code>	Pilihan <i>metric</i> yang digunakan untuk menghitung <i>distance</i> antara 2 titik: <ul style="list-style-type: none"> - 'manhattan': Perhitungan jarak dengan <i>manhattan distance</i> - 'euclidean': Perhitungan jarak dengan <i>euclidean distance</i>
<code>X_train</code>	List yang berisi nilai-nilai X yang digunakan untuk <i>training</i>
<code>y_train</code>	List yang berisi nilai-nilai target(y) yang digunakan untuk <i>training</i>
<code>weight</code>	Pilihan <i>weight</i> yang digunakan untuk melakukan prediksi: <ul style="list-style-type: none"> - 'distance': <i>weight</i> dari setiap titik adalah invers dari jaraknya ke titik yang akan diprediksi - 'uniform': setiap titik memiliki <i>weight</i> yang sama

Daftar *method* dari `class KNeighborsClassifier` adalah sebagai berikut.

Tabel 2. *Method* Kelas `KNeighborsClassifier`

Nama Method	Keterangan
<code>KNeighborsClassifier(n_neighbors, metric, weight)</code>	Konstruktor kelas <code>KNeighborsClassifier</code> dengan argumen <code>n_neighbors</code> , <code>metric</code> , dan <code>weight</code> sebagai parameter yang digunakan untuk prediksi
<code>fit(X_train, y_train)</code>	Fungsi yang digunakan untuk set <code>X_train</code> dan <code>y_train</code> yang digunakan untuk <i>training</i>
<code>predict(X_test)</code>	<p>Fungsi yang digunakan untuk melakukan prediksi nilai y untuk setiap elemen pada <code>X_test</code>.</p> <p>Fungsi ini akan memanggil <code>get_nearest_neighbors()</code> untuk mendapatkan daftar <i>neighbor</i> terdekat. Kemudian, fungsi ini akan menghitung target dengan memilih target dengan <i>weight</i> terbesar. Jika parameter <i>weight</i> pada kelas adalah 'uniform', maka akan dipilih <i>target</i> dengan count terbanyak pada list <i>nearest neighbors</i>. Jika parameter <i>weight</i> pada kelas adalah 'distance', maka akan dipilih <i>target</i> dengan <i>sum</i> invers jarak terbesar pada list <i>nearest neighbors</i>.</p>

score(X_test, y_test)	Fungsi ini adalah fungsi untuk menghitung akurasi hasil fungsi predict(X_test) terhadap y_test
__get_nearest_neighbors(X_test)	Fungsi ini merupakan <i>helper function</i> untuk mendapatkan <i>nearest neighbors</i> dari setiap X pada X_test. Fungsi ini menggunakan <i>helper function</i> calculate_test_to_train_distance(X_test) untuk mendapatkan <i>distance</i> titik test terhadap setiap titik train. Kemudian hasilnya di-sort secara ascending. Fungsi ini kemudian mengembalikan n_neighbors pertama pada list tersebut.
__calculate_test_to_train_distance(X_test)	Fungsi ini merupakan fungsi untuk menghitung jarak setiap X pada X_test terhadap setiap titik X pada X_train. Jika <i>metric</i> adalah 'manhattan', fungsi ini akan memanggil <i>helper function</i> calculate_manhattan_distance, jika 'euclidean', akan memanggil <i>helper function</i> calculate_euclidean_distance

IMPLEMENTASI NAIVE-BAYES

Algoritma Naive-Bayes adalah sebuah algoritma klasifikasi probabilistik yang didasari oleh *Bayes' Theorem*. Algoritma ini memiliki asumsi bahwa semua *features* yang digunakan untuk klasifikasi independen satu sama lain. Algoritma ini membedakan pemrosesan fitur kategorikal dan numerik sehingga diimplementasikan dalam dua kelas, yaitu `NaiveBayesGaussian` dan `NaiveBayesCategorical`.

Kelas `NaiveBayesGaussian` menangani fitur numerik dengan daftar atribut sebagai berikut.

Tabel 3. Atribut Kelas `NaiveBayesGaussian`

Nama Atribut	Keterangan
<code>numerical_columns</code>	List yang berisi nilai-nilai seluruh kolom x numerik yang digunakan untuk <i>training</i>
<code>y_train</code>	List yang berisi nilai-nilai target (y) yang digunakan untuk <i>training</i>
<code>target_probs</code>	Dictionary dengan pasangan “key: value” dengan key merupakan nilai unik dari kolom target (y) dan value bernilai $P(key)$
<code>target_values</code>	List ini berisi nilai-nilai unik dari kolom target (y)

Kelas `NaiveBayesGaussian` dilengkapi dengan *method* sebagai berikut.

Tabel 4. *Method* Atribut Kelas `NaiveBayesGaussian`

Nama Method	Keterangan
<code>NaiveBayesGaussian()</code>	Konstruktor dari kelas <code>NaiveBayesGaussian</code> yang melakukan inisialisasi pada seluruh atribut kelas.
<code>fit(numerical_columns, y_train)</code>	<p>Fungsi akan melakukan <i>set</i> pada <code>numerical_columns</code> dan <code>y_train</code> pada atribut kelas sesuai dengan parameter yang diberikan.</p> <p>Fungsi ini juga akan melakukan kalkulasi untuk mengisi atribut <code>target_probs</code> dan <code>conditional_probs</code>. <code>target_probs</code> akan dikalkulasi dengan menghitung jumlah tiap nilai unik dan membaginya dengan banyak data pada kolom target.</p>
<code>predict_proba(x_test_numerical)</code>	Fungsi ini akan menginisialisasi perhitungan probabilitas setiap nilai unik target untuk setiap baris data numerik yang diberikan pada <code>x_test_numerical</code> , yaitu baris-baris data yang

	ingin diprediksi nilai y nya. Perhitungan probabilitas dilakukan dengan memanggil fungsi <code>__gaussian_proba</code> terhadap masing-masing baris data.
<code>__gaussian_proba(x_test_numerical)</code>	<p>Fungsi ini bertujuan untuk menghitung probabilitas dari setiap nilai unik target untuk <code>x_test_numeric</code> yang berupa baris data masukan yang ingin diprediksi. Probabilitas dari setiap nilai unik dikembalikan dalam bentuk <i>list</i>.</p> <p>Perhitungan dilakukan dengan mencari nilai $P(x y)$ untuk setiap kolom untuk setiap nilai target y yang mungkin. Nilai x merupakan nilai dari suatu kolom pada <code>x_test_numeric</code>. Nilai $P(x y)$ dihitung dengan,</p> $P(x y) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$ <p>Untuk menghindari perkalian nilai $P(x y)$ yang terlalu kecil, dicari nilai logaritma dari nilai probabilitas,</p> $\log(P(x y)) = -0.5 \log(2\pi\sigma^2) - \frac{(x-\mu)^2}{2\sigma^2}$ <p>Dengan σ^2 bernilai dari varians dari semua nilai pada kolom yang dicari pada <code>x_train</code> yang memiliki nilai target y, serta μ merupakan nilai rata-rata dari nilai pada kolom yang dicari pada <code>x_train</code> yang memiliki nilai target y.</p> <p>Probabilitas dari nilai target y untuk <code>x_test_numeric</code> dapat dihitung dengan,</p> $P(y x_1, x_2, x_3...) = P(y) \prod P(x_i y)$ <p>Dengan dihitung secara logaritma,</p> $\log(P(y x_1, x_2,...)) = \log(P(y)) + \sum \log(P(x_i y))$ <p>dengan nilai $P(y)$ ditemukan dari atribut <code>target_probs</code> pada kelas.</p> <p>Nilai probabilitas dari setiap nilai y dalam bentuk logaritma akan dihitung eksponennya untuk dikembalikan.</p>

Kelas `NaiveBayesCategorical` menangani fitur kategorikal dengan daftar atribut sebagai berikut.

Tabel 5. Atribut Kelas `NaiveBayesCategorical`

Nama Atribut	Keterangan
<code>categorical_columns</code>	List yang berisi nilai-nilai seluruh kolom x kategorikal yang digunakan untuk <i>training</i>

y_train	List yang berisi nilai-nilai target (y) yang digunakan untuk <i>training</i>
target_probs	Dictionary dengan pasangan “key: value” dengan key merupakan nilai unik dari kolom target (y) dan value bernilai $P(key)$
conditional_probs	List yang berisi <i>dictionary</i> dengan pasangan “a-b: value” dengan a merupakan nilai unik dari kolom pada categorical_columns, b merupakan nilai unik dari kolom target (y), dan value merupakan nilai dari $P(a b)$

Kelas `NaiveBayesCategorical` dilengkapi dengan *method* sebagai berikut.

Tabel 6. *Method* Kelas `NaiveBayesCategorical`

Nama Method	Keterangan
<code>NaiveBayesCategorical()</code>	Konstruktor dari kelas <code>NaiveBayesCategorical</code> yang melakukan inisialisasi pada seluruh atribut kelas.
<code>fit(categorical_columns, y_train)</code>	<p>Fungsi akan melakukan <i>set</i> pada categorical_columns dan y_train pada atribut kelas sesuai dengan parameter yang diberikan.</p> <p>Fungsi ini juga akan melakukan kalkulasi untuk mengisi atribut target_probs dan conditional_probs. Target_probs akan dikalkulasi dengan menghitung jumlah tiap nilai unik dan membaginya dengan banyak data pada kolom target. Conditional_probs berupa “a-b: value” akan dikalkulasi dengan menghitung banyak data pada salah satu kolom categorical_columns yang bernilai a ketika kolom target bernilai b dibagi dengan banyaknya data pada kolom target yang bernilai b. Pada perhitungan conditional_probs akan dilakukan normalisasi.</p>
<code>predict_proba(x_test)</code>	<p>Fungsi ini akan melakukan perhitungan probabilitas masing-masing baris data terhadap klasifikasi kolom target. Nilai probabilitas masing-masing klasifikasi kolom target dilakukan dengan rumus berikut,</p> $P(y x_1, x_2, x_3...) = P(y) \prod P(x_i y)$ <p>Hal tersebut diimplementasi dengan mendapatkan nilai $P(x_i y)$ dari conditional_probs, serta nilai $P(y)$ yang didapatkan dari target_probs. Untuk hasil yang lebih optimal, probabilitas akan dinormalisasi dengan pembagian terhadap jumlah nilai probabilitas pada baris data tersebut.</p>

PERBANDINGAN HASIL IMPLEMENTASI DAN PUSTAKA

Untuk melakukan perbandingan *performance* antara hasil implementasi dan pustaka, digunakan beberapa metrik evaluasi kinerja klasifikasi. Metrik-metrik ini termasuk *Accuracy Average*, *F1 Macro Average*, *F1 Micro Average*, *Precision Macro Average*, *Precision Micro Average*, *Recall Macro Average*, dan *Recall Micro Average*.

Berikut adalah *performance score* dari algoritma KNN yang dihasilkan oleh hasil implementasi dan pustaka:

Tabel 7. *Performance Score KNN*

Metrik	Performance Score	
	Implementasi	Pustaka
<i>Accuracy Average</i>	0.94	0.94
<i>F1 Macro Average</i>	0.94	0.94
<i>F1 Micro Average</i>	0.94	0.94
<i>Precision Macro Average</i>	0.94	0.94
<i>Precision Micro Average</i>	0.94	0.94
<i>Recall Macro Average</i>	0.94	0.94
<i>Recall Micro Average</i>	0.94	0.94

Dari tabel *performance score* terhadap algoritma KNN diatas, *performance score* untuk algoritma KNN hasil implementasi dan pustaka sama yang menunjukkan bahwa hasil implementasi sudah baik dan benar. Keduanya menunjukkan *performance score* di angka 0.94 yang menunjukkan *process tuning* dan *training* model sudah baik.

Berikut adalah *performance score* dari algoritma Naive Bayes yang dihasilkan oleh hasil implementasi dan pustaka:

Tabel 8. *Performance Score Naive Bayes*

Metrik	Performance Score	
	Implementasi	Pustaka
<i>Accuracy Average</i>	0.7817	0.7817
<i>F1 Macro Average</i>	0.7809	0.7809
<i>F1 Micro Average</i>	0.7817	0.7817
<i>Precision Macro Average</i>	0.7814	0.7814

<i>Precision Micro Average</i>	0.7817	0.7817
<i>Recall Macro Average</i>	0.7806	0.7806
<i>Recall Micro Average</i>	0.7817	0.7817

Dari tabel *performance score* algoritma Naive-Bayes di atas, algoritma yang diimplementasikan memiliki *performance score* yang sama. Hal ini membuktikan bahwa implementasi dari Naive Bayes yang diterapkan sudah baik dan benar. Hal ini juga dapat dilihat dari prediksi kedua implementasi yang menghasilkan hasil yang serupa. Akurasi yang bernilai 0.78 menandakan bahwa pada kasus ini algoritma Naive-Bayes tidak sebaik algoritma KNN dalam melakukan prediksi klasifikasi. Hal ini dapat disebabkan oleh asumsi pada Naive Bayes yang menganggap seluruh *features* independen.

SUBMISI KAGGLE

Submisi Kaggle dilakukan dengan memanfaatkan implementasi algoritma KNN yang memiliki akursi lebih tinggi. Untuk melakukan submisi dilakukan pemrosesan menggunakan parameter yang telah diperoleh melalui eksperimen menggunakan scikit-learn library dan eksperimen lebih lanjut. Berikut merupakan pemrosesan menggunakan parameter yang didapatkan melalui eksperimen menggunakan scikit-learn library.

```
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import f_classif
from lib.knn import KNeighborsClassifier

df_train = pd.concat([pd.read_csv('data/data_train.csv'), pd.read_csv('data/data_validation.csv')])
df_test = pd.read_csv('data/data_test.csv')
X_train = df_train.drop('price_range',axis=1)
y_train = df_train['price_range'].values
X_test = df_test

X_train_selected, X_test_selected, selected_features = get_x_test_best_features(N_FEATURES_SELECTED,
X_train, y_train, X_test)
X_test_selected = X_test_selected.values
KNN = KNeighborsClassifier(n_neighbors=BEST_N_NEIGHBORS, metric=BEST_METRIC, weight=BEST_WEIGHT)
KNN.fit(X_train_selected,y_train)
predictions = df_test['price_range'] = KNN.predict(X_test_selected)
df_test[['id','price_range']].to_csv("submission.csv", index=False)
```

Berikut merupakan pemrosesan menggunakan parameter yang didapatkan melalui eksperimen lebih lanjut.

```
from lib.knn import KNeighborsClassifier

df_train = pd.concat([pd.read_csv('data/data_train.csv'), pd.read_csv('data/data_validation.csv')])
df_test = pd.read_csv('data/data_test.csv')
X_train = df_train.drop('price_range',axis=1)
y_train = df_train['price_range'].values
X_test = df_test
X_train_selected, X_test_selected, _ = get_x_test_best_features(4, X_train, y_train, X_test)
X_test_selected = X_test_selected.values
WeightedKNN = KNeighborsClassifier(n_neighbors=2, metric='euclidean', weight='distance')
WeightedKNN.fit(X_train_selected, y_train)
weighted_predictions = df_test['price_range'] = WeightedKNN.predict(X_test_selected)
df_test[['id','price_range']].to_csv("submission3.csv", index=False)
```

Pemrosesan untuk melakukan submisi Kaggle secara lebih jelas adalah sebagai berikut.

1. Mendefinisikan df_train yang berupa data dari data_train.csv dan data_validation.csv
2. Mendefinisikan df_test yang berupa data dari data_test.csv

3. Mendefinisikan `X_train` yang berupa `df_train` tanpa kolom target yaitu kolom "price_range"
4. Mendefinisikan `y_train` kolom target "price_range" pada `df_train`
5. Mencari `X_train_selected`, `X_test_selected`, dan `selected_features` melalui fungsi `get_x_test_best_features` untuk mencari fitur paling baik yang akan digunakan untuk melakukan *training*
6. Membuat model KNN dengan `KNeighborsClassifier` dengan parameter jumlah neighbor, metric, dan weight
7. Melakukan prediksi menggunakan model KNN dengan `X_test_selected` yang telah didapatkan
8. Melakukan *export* kolom id dan price_range ke csv untuk melakukan submisi

Berdasarkan kedua submisi dan eksperimen yang dilakukan, didapatkan parameter yang paling optimal yaitu fitur terbaik yang dipilih empat, jumlah *neighbor* dua, *metric manhattan*, dan *weight distance*.

KONTRIBUSI KELOMPOK

Fitur	Kontributor
KNN	13521044, 13521046
Naive-Bayes	13521094, 13521134

REPOSITORY

Seluruh implementasi lengkap dapat diakses melalui repository berikut
<https://github.com/liviaarumsari/Tubes2-whoosh>