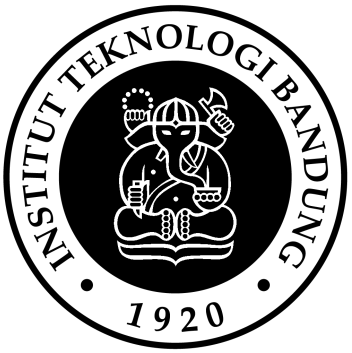
IF3170 Inteligensi Buatan

IMPLEMENTASI ALGORITMA K-NEAREST NEIGHBOURS DAN NAIVE BAYES UNTUK MODEL PREDIKSI PRICE_RANCE

Laporan Tugas Besar II

Disusun untuk memenuhi tugas mata kuliah Inteligensi Buatan
pada Semester I (satu) Tahun Akademik 2023/2024.



Oleh

Fakhri Muhammad Mahendra	**13521045**
Fatih Nararya R.I.	**13521060**
Akbar Maulana Ridho	**13521093**
Alisha Listya Wardhani	**13521171**

PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
BANDUNG
2023

Daftar Isi

Daftar Tabel

BAB I
PENDAHULUAN

Dataset yang diberikan pada Tugas Besar ini merupakan dataset spesifikasi *handphone*. Jenis data pada dataset dapat berupa *binary* yang menandakan true/false ataupun nilai *continuous*. Pada dataset tersebut terdapat beberapa deskripsi kolom, yaitu:

1. battery_power: Total energi baterai dalam satu waktu diukur dalam mAh
2. blue: Memiliki bluetooth atau tidak
3. clock_speed: Kecepatan mikroprosesor menjalankan instruksi
4. dual_sim: Memiliki dukungan dual sim atau tidak
5. fc: Resolusi kamera depan dalam megapiksel
6. four_g: Memiliki 4G atau tidak
7. int_memory: Memori internal dalam gigabyte
8. m_dep: Ketebalan ponsel dalam cm
9. mobile_wt: Berat ponsel
10. n_cores: Jumlah core prosesor
11. pc: Resolusi kamera utama dalam megapiksel
12. px_height: Tinggi resolusi piksel
13. px_width: Lebar resolusi piksel
14. ram: Ukuran RAM dalam megabyte
15. sc_h: Tinggi layar ponsel dalam cm
16. sc_w: Lebar layar ponsel dalam cm
17. talk_time: Waktu telepon maksimum dalam satu kali pengisian baterai
18. three_g: Memiliki 3G atau tidak
19. touch_screen: Memiliki layar sentuh atau tidak
20. wifi: Memiliki wifi atau tidak
21. price_range (target): Rentang harga dengan nilai 0 (biaya rendah), 1 (biaya sedang), 2 (biaya tinggi) atau 3 (biaya sangat tinggi).

Dalam Tugas Besar II Intelegensi Buatan ini, dibuat algoritma untuk memprediksi nilai price_range (target). Algoritma yang digunakan adalah Algoritma K-Nearest Neighbors dan Algoritma Naive Bayes. Penulis memanfaatkan algoritma Naive Bayes dengan menggunakan distribusi gaussian dan bernoulli.

BAB II
RANCANGAN DAN IMPLEMENTASI ALGORITMA

2.1. Rancangan Pemilihan Fitur

Tujuan dari pemilihan fitur adalah menemukan himpunan fitur terbaik yang memungkinkan untuk membangun model yang optimal untuk fenomena tersebut. Berikut merupakan metode pemilihan fitur yang digunakan dalam Tugas Besar II Intelegensi Buatan.

2.2.1. Information Gain

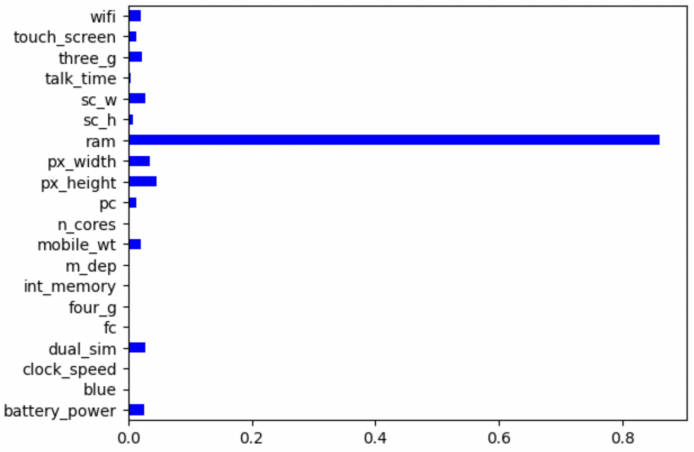
Information Gain menghitung reduksi entropi dari sebuah transformasi dataset. Metode ini dapat mengevaluasi masing-masing informasi yang didapatkan dari setiap kolom (variabel) berdasarkan variabel target. Setelah itu, akan diambil n fitur dengan nilai paling besar.

Tabel 2.2.1.1. Information Gain Method

```
from sklearn.feature_selection import mutual_info_classif
import matplotlib.pyplot as plt

X = df_train.drop('price_range', axis=1)
Y = df_train['price_range']

%matplotlib inline
importances = mutual_info_classif(X,Y)
feat_importances = pd.Series(importances, df_train.columns[0:len(df_train.columns) - 1])
feat_importances.plot(kind='barh', color='blue')
plt.show()
```



2.2.2. Correlation Map

Metode korelasi merupakan pengukuran linear antara 2 atau lebih variabel. Semakin besar korelasinya dengan target, maka fitur tersebut mempunyai hubungan yang kuat. Hal ini berimplikasi pada fitur tersebut dapat menjadi acuan dalam memprediksi target. Setelah itu, akan diambil n fitur dengan nilai korelasi paling besar.

Tabel 2.2.2.1. Correlation Map Method

```
from scipy import stats

for colName in CATEGORICAL_VARIABLES:
   corr = stats.pointbiserialr(df[colName], df['price_range'])
   print("Korelasi dari %s terhadap price_range: \n %s \n" %(colName, corr))
```

```
Korelasi dari blue terhadap price_range:
PointbiserialrResult(correlation=0.041946888607414495, pvalue=0.11669514987098381)
Korelasi dari dual_sim terhadap price_range:
PointbiserialrResult(correlation=-0.010756163737641003, pvalue=0.6876023812586639)
Korelasi dari four_g terhadap price_range:
PointbiserialrResult(correlation=0.0005508484718003217, pvalue=0.9835707600444511)
Korelasi dari three_g terhadap price_range:
PointbiserialrResult(correlation=0.02709762831217497, pvalue=0.3109738038894839)
Korelasi dari touch_screen terhadap price_range:
PointbiserialrResult(correlation=-0.029842480103712092, pvalue=0.26448442018462326)
Korelasi dari wifi terhadap price_range:
PointbiserialrResult(correlation=0.034329357621172944, pvalue=0.19924065090605791)
```

```
for colName in CONTINUOUS_VARIABLES:
   corr = df[colName].corr(df['price_range'])
   print("Korelasi data %s dengan price_range: %s" %(colName, corr))
```

```
Korelasi data battery_power dengan price_range: 0.184800924495531
Korelasi data clock_speed dengan price_range: 0.014031254818008095
Korelasi data fc dengan price_range: -0.0038420102981917303
Korelasi data int_memory dengan price_range: 0.026175706877841626
Korelasi data m_dep dengan price_range: 0.0012049180209846376
Korelasi data mobile_wt dengan price_range: -0.0747687504832367
Korelasi data pc dengan price_range: -0.005214430491653005
Korelasi data px_height dengan price_range: 0.15883273548307988
Korelasi data px_width dengan price_range: 0.17871269011026591
Korelasi data ram dengan price_range: 0.9183192307843847
Korelasi data sc_h dengan price_range: 0.012148883173074995
Korelasi data sc_w dengan price_range: 0.019911698810365034
Korelasi data talk_time dengan price_range: 0.01111273175475493
```

2.2. Implementasi Algoritma KNN

Algoritma KNN yang dibuat oleh tugas besar ini terenkapsulasi dalam satu kelas saja, yaitu kelas `KNNClassifier`. Instans kelas ini dibuat dengan memberikan tiga parameter, `standardize` (apakah ingin dilakukan normalisasi pada data), `n_neighbours` (jumlah *neighbours* yang digunakan saat algoritma berjalan), dan `multiply_correlations` (apakah korelasi tiap kolom dengan target diperhitungkan dalam proses prediksi). Terdapat dua metode publik yang digunakan untuk menjalankan prediksi pada kelas tersebut, yaitu `fit` (menerima data latihan sebagai parameter) untuk melatih model dan `predict` (menerima data validasi) untuk melakukan prediksi.

Desain dari kelas `KNNClassifier` pada tugas besar ini terinspirasi dari kelas `KNeighborsClassifier` pada pustaka `scikit-learn` yang juga digunakan pada tugas besar ini sebagai pembanding. Berikut diberikan seluruh metode yang ada pada kelas ini beserta deskripsinya.

Tabel 2.2.1. Daftar metode pada kelas `KNNClassifier`

```
def __init__(self, standardize : bool = False, n_neighbors=5, multiply_correlations : bool
= True):
'''
Parameters:
    -   standardize (bool) : Apakah data latihan akan dinormalisasi
    -   n_neighbors (int) : Jumlah tetangga yang digunkan saat prediksi
    -   multiply_correlations (bool) : Apakah besar korelasi dari sebuah kolom
        diperhitungkan pada kolom target
'''
```

```
def fit(self, x: DataFrame, y: Series, categorical_column: List[str] = []):
'''
Melatih model yang diinstansiasi dengan data latihan yang diberikan dan konfigurasi
Parameters:
    -   x (Dataframe) : Data latihan tanpa label
    -   y (Series) : Label dari data latihan (target)
    -   categorical_column (List[str]) : Kolom dari x yang bertipe kategorikal
'''
```

```
def save_model(self, feature_path = 'features.csv', target_path = 'target.csv', conf_path =
'conf.json'):
'''
Save the model to the given files
'''
```

```
def load_model(self,feature_path = 'features.csv', target_path = 'target.csv', conf_path =
'conf.json'):
'''
Load the model from the given files
```

```
'''
```

```
def _get_correlation(self):
'''
Mendapatkan korelasi dari setiap kolom yang ada pada data latihan dengan label, kemudian
menyimpannya ke atribut column_correlations dari instansi
'''
```

```
def _standardize_data(self, df: DataFrame, columns: List[str]):
'''
Melakukan standarisasi dari data yang diberikan pada kolom yang diberikan dari data
tersebut, lalu mengembalikan hasilnya
Parameters:
    -   df (Dataframe) : Data yang ingin dinormalisasi
    -   columns (List[str]) : Kolom dari df yang ingin dinormalisasi
Returns:
    -   df_standardized (Dataframe): salinan dari df yang telah dinormalisasi pada kolom
        yang diberikan
'''
```

```
def _calculate_distance(self, row: Series) -> np.ndarray:
'''
Menghitung jarak dari baris yang diberikan
Data diasumsikan data sudah ternormalisasi sehingga kolom numerik dan kategorikal bisa
dianggap sama perhitungannya
Jika multiply_correlations dari instansi adalah True, maka korelasi dari kolom yang ada
dengan target akan diperhitungkan dengan dikalikan pada jarak
Parameters:
    -   y (Series) : Label dari data latihan (target)
Returns:
    -   result (int): Jarak dari baris yang diberikan
'''
```

```
def _predict_row(self, row: Series):
'''
Memprediksi nilai label dari baris yang diberikan lalu mengembalikan hasilnya
Parameters:
    -   row (Series) : Baris data yang ingin diprediksi nilai labelnya
Returns:
    -   result (int): Prediksi nilai label untuk baris yang diberikan
'''
```

```
def predict(self, to_predict: DataFrame):
'''
Memprediksi nilai label dari baris yang diberikan lalu mengembalikan hasilnya
Parameters:
    -   to_predict (Dataframe) : Data validasi yang ingin diprediksi nilai labelnya
Returns:
    -   result (array of int): Nilai label hasil prediksi untuk data yang diberikan, terurut
        sesuai tiap baris pada data
'''
```

Kalkulasi jarak pada metode `_calculate_distance` dilakukan dengan cara mengambil RMS dari selisih data latihan dan baris yang diberikan. Hasil prediksi pada metode `_predict_row`

dilakukan dengan mengambil sejumlah `n_neighbours` baris terdekat dari baris yang diberikan, lalu mengambil modus dari nilai label semua tetangga tersebut.

2.3. Penjelasan Algoritma Naive Bayes

Naive-Bayes merupakan algoritma yang menggunakan teorema bayes sebagai pondasinya. Teorema ini menghitung probabilitas posterior $P(Y|X)$ dari fitur Y diberikan X. Pada algoritma ini, diasumsikan semua fitur independen antara satu sama lain. Pada tugas besar ini, dibuat 2 macam pendekatan algoritma karena ada fitur yang bersifat kategorik dan kontinu.

Fitur yang bersifat kategorik memanfaatkan probabilitas yang diajarkan pada materi perkuliahan, sedangkan fitur yang bersifat kontinu memanfaatkan probabilitas dengan distribusi gaussian. Dengan mengasumsikan bahwa model mengikuti distribusi gaussian (normal), dihasilkan rumus gaussian seperti pada gambar 2.3.1. Rumus tersebut mengasumsikan bahwa variabel x mengikuti distribusi normal dengan mean u_k dengan variansi σ_k^2 untuk kelas C_k.

$$p(x = v \mid C_k) = \frac{1}{\sqrt{2\pi\sigma_k^2}} e^{-\frac{(v-\mu_k)^2}{2\sigma_k^2}}$$

Gambar 2.3.1. Rumus Gaussian Naive Bayes

2.4. Implementasi Algoritma Naive Bayes

Algoritma Naive Bayes pada Tugas Besar II ini berada dalam satu kelas. Kelas ini hanya membutuhkan parameter data yang dilatih (train). Pada kelas ini, terdapat dua metode publik yaitu fit (menerima data latihan sebagai parameter) untuk melatih model dan predict (menerima data validasi) untuk melakukan prediksi. Berikut merupakan penjabaran metode.

Tabel 2.4.1. Daftar metode pada kelas `NaiveBayesClassifier`

```
def __init__(self, smoothing : bool = True):
        '''
            Parameters:
                smoothing (bool) : Apakah pelatihan data menggunakan smoothing
        '''
```

```
def _calculate_gaussian_pdf(self, feature, mean, std):
        '''
            Mengkalkulasi nilai gaussian berdasarkan fitur, mean, dan standar deviasi
        '''
```

```
def _calculate_probabilities(self):
        '''
            Mengkalkulasi probabilitas setiap fitur
        '''
```

```
def _convert_keys_to_string(self, dictionary: dict):
        '''
            Fungsi utilitas untuk mengubah key value integer ke string
        '''
```

```
def _print_to_json(self):
        '''
            Menyimpan model yang telah dibuat ke dalam file berbentuk JSON
        '''
```

```
def fit(self, x: DataFrame, y: Series, save : bool):
        '''
        Melatih model yang diinstansiasi dengan data latihan yang diberikan
            Parameters:
                x (Dataframe) : Data latihan tanpa label
        '''
```

```
def fit_from_json(self, means_file: str, variance_file: str, prior_file: str,
result_dict_file: str, nominal_unique_counts_file:str):
        '''
            Mengkonfigurasi model berdasarkan masukan JSON File Path yang diterima
            Parameters:
                    -   means_file (String) : Pathname nilai mean
                    -   variance_file (String) : Pathname nilai string
                    -   prior_file (String) : Pathname probabilitas setiap target class
                    -   result_dict_file (String) : Pathname hasil probabilitas kolom
                    -   nominal_unique_counts (String) : Pathname jumlah setiap kolom nominal
        '''
```

```
def predict(self, to_predict: DataFrame):
        '''
            Memprediksi nilai label dari baris yang diberikan lalu mengembalikan hasilnya
            Parameters:
                    -   to_predit (Dataframe) : Data validasi yang ingin diprediksi
            Returns:
                    -   predictions (array of int) : Nilai label hasil prediksi untuk data
                        yang diberikan, terurut setiap baris pada data
        '''
```

2.5. Implementasi KNN dan Naive Bayes dengan Library Scikit

Pada Tugas Besar II ini, juga di implementasi algoritma KNN dan Naive Bayes dengan Library Scikit. Setelah itu, dihitung akurasi untuk setiap metode pengetesan.

Tabel 2.5.1. Implementasi algoritma dengan Library `Scikit`

Implementasi

`# TEST WITH KNN`
`model = KNeighborsClassifier(n_neighbors=17, weights='distance')`
`model.fit(x_train, y_train)`
`knn_accuracy = accuracy_score(y_validation, model.predict(x_validation))`
`print(f"KNN Accuracy {knn_accuracy}")`

`# TEST WITH KNN : Normalized`
`model = KNeighborsClassifier(n_neighbors=17, weights='distance')`
`model.fit(x_train_standardized, y_train)`
`knn_accuracy = accuracy_score(y_validation, model.predict(standardize_data(x_validation, ratio_columns)))`
`print(f"KNN Normalized Accuracy {knn_accuracy}")`

`# Test with Naive Bayes (Gaussian)`
`model_gaussian = GaussianNB()`
`model_gaussian.fit(x_train, y_train)`
`gaussian_accuracy = accuracy_score(y_validation, model_gaussian.predict(x_validation))`
`print(f"Gaussian accuracy {gaussian_accuracy}")`

`# Test with Naive Bayes (Multinomial)`
`model_multinomial = MultinomialNB()`
`model_multinomial.fit(x_train, y_train)`
`multinomial_acc = accuracy_score(y_validation, model_multinomial.predict(x_validation))`
`print(f"Multinomial accuracy {multinomial_acc}")`

`# Test with Naive Bayes (Multinomial)`
`model_categorical = CategoricalNB()`
`model_categorical.fit(x_train, y_train)`
`categorical_acc = accuracy_score(y_validation, model_categorical.predict(x_validation))`
`print(f"Categorical accuracy {categorical_acc}")`

`# Test with Naive Bayes (Bernoulli)`
`model_bernoulli = BernoulliNB()`
`model_bernoulli.fit(x_train, y_train)`
`bernoulli_acc = accuracy_score(y_validation, model_bernoulli.predict(x_validation))`
`print(f"Bernoulli accuracy {bernoulli_acc}")`

BAB III
PENGUJIAN DAN ANALISIS ALGORITMA

Perbandingan dari `accuracy_score` setiap metode prediksi diberikan pada tabel berikut. Nilai `n_neighbours` yang dipilih bersifat *arbitrary* tanpa alasan khusus. NB menandakan Naive Bayes.

Tabel 4.1. Perbandingan `accuracy_score` dari semua metode prediksi dan implementasi

No	Metode Prediksi	Implementasi	Keterangan	`accuracy_score`
1.	KNN	Sendiri	`multiply_correlations=True, n_neighbours=17, standardize=True`	0.8617
2.	KNN	Sendiri	`multiply_correlations=False, n_neighbours=17, standardize=False`	0.9317
3.	KNN	Sendiri	`multiply_correlations=False, n_neighbours=17, standardize=True`	0.6767
4.	KNN	Sendiri	`multiply_correlations=True, n_neighbours=17, standardize=False`	0.928
5.	KNN	Pustaka	`scikit-learn` KNN; data tidak dinormalisasi; `n_neighbors=10, weights='distance'`	0.9367
6.	KNN	Pustaka	`scikit-learn` KNN; data dinormalisasi; `n_neighbors=10, weights='distance'`	0.685
7.	NB	Sendiri	Fitur dipilih semua, training menggunakan data train_csv	0.7816
8.	NB	Sendiri	Membaca JSON file yang sudah disimpan dari pengujian nomor 7	0.798
9.	NB	Sendiri	Fitur dipilih hanya nilai yang bersifat continuous	0.795
10.	NB	Sendiri	Sepuluh fitur terbaik dipilih berdasarkan hasil information_gain	0.795

11.	NB	Sendiri	Fitur dipilih berdasarkan hasil korelasi	0.7766
12.	NB	Pustaka	`scikit-learn` GaussianNB;	0.7816
13.	NB	Pustaka	`scikit-learn` MultinomialNB;	0.5316
14.	NB	Pustaka	`scikit-learn` CategoricalNB;	0.2966
15.	NB	Pustaka	`scikit-learn` BernoulliNB;	0.2333

4.1. Analisis Nilai KNN

Perbandingan skor antara nilai KNN dengan tidak dinormalisasi hampir sama antara algoritma implementasi sendiri dan pustaka `scikit-learn`, begitupun untuk data yang dinormalisasi. Hal ini menunjukkan bahwa implementasi dari algoritma KNN yang kami lakukan semestinya sudah benar.

Bagian yang lebih menarik pada pengujian ini adalah perbedaan drastis pada normalisasi data. Tampak bahwa baik pada algoritma implementasi sendiri maupun dari pustaka `scikit-learn`, prediksi yang lebih akurat didapatkan dari menggunakan data yang tidak dinormalisasi. Pada permukaannya, hal ini cukup aneh, mengingat tanpa normalisasi, maka nilai jarak yang dihasilkan dan digunakan oleh KNN tidak memperlakukan kolom-kolom yang ada secara setara oleh algoritma ini, tetapi malah didominasi oleh kolom dengan nilai tinggi.

Namun, hal tersebut mudah dijelaskan dengan melihat korelasi dari kolom-kolom yang ada. Seperti yang telah ditemukan pada tugas kecil, hanya terdapat satu buah kolom yang memiliki korelasi kuat (mendekati 1) dengan target, yaitu kolom `ram`. Sementara terdapat tiga buah kolom lain yang memiliki korelasi lebih lemah (di antara 0.2 dan 0.4) : `battery_power`, `px_height`, dan `px_width`. Sementara kolom-kolom lainnya tidak memiliki korelasi dengan target. Kebetulan (atau mungkin memang dibuat asisten sedemikian rupa supaya kami bingung) keempat kolom tersebut memiliki nilai numerik yang jauh lebih tinggi, berada pada jangkauan ribuan, dari kolom-kolom numerik lainnya yang memiliki nilai di bawah seratus.

Memperhitungkan perbedaan jangkauan kolom yang drastis tersebut, data yang tidak dinormalisasi tentu saja memiliki akurasi yang lebih tinggi, karena nilai jarak jauh lebih banyak ditentukan oleh kolom yang memiliki korelasi dengan target. Hipotesis ini juga

didukung dengan fakta bahwa `accuracy_score` dari data yang dinormalisasi langsung meningkat tajam begitu korelasi dari kolom diperhitungkan (dari 0.6767 pada no.3 ke 0.8617 pada no.1).

Perbedaan jangkauan nilai yang drastis tersebut juga membuat penggunaan korelasi kolom hanya memberikan peningkatan `accuracy_score` yang marginal pada data yang tidak dinormalisasi (no.2 dan no.4) karena kolom yang tidak berkorelasi ≪ kolom berkorelasi, sehingga membuat kolom yang tidak berkorelasi menjadi bernilai 0 tidak berpengaruh banyak.

4.2. Analisis Nilai Naive Bayes

Berdasarkan tabel 4.1., didapat bahwa hasil pengujian menggunakan algoritma Naive Bayes menunjukan hasil yang stabil. Skor yang diperoleh dengan memanfaatkan semua fitur mendapatkan `accuracy_score` 0.7816. Skor ini menunjukan bahwa model cukup baik. Namun, hal ini dapat menyebabkan model mengalami *overfitting* jika fitur-fitur tersebut tidak semua relevan. Menariknya, jika hasil dari model tersebut di-*save* dan di-*load* untuk membuat model baru menghasilkan `accuracy_score` 0.798. Hal ini menunjukan bahwa proses penyimpanan dan pembacaan tidak mengubah performa model. Skor yang bertambah sedikit dapat terjadi karena *underflow* dalam menyimpan nilai probabilitas.

Jika fitur yang digunakan dipilih, secara garis besar tidak mengubah performa model. Didapatkan bahwa pemilihan fitur menggunakan information gain adalah metode yang tepat untuk kasus ini karena sedikit meningkatkan `accuracy_score` menjadi 0.795. Skor yang sama juga diperoleh ketika fitur yang dipilih hanya fitur kontinu. Hal ini menunjukkan bahwa fitur kontinu memiliki pengaruh kuat dalam melakukan prediksi yang akurat.

Untuk pengujian menggunakan variasi Naive Bayes seperti GaussianNB, MultinomialNB, CategoricalNB, dan BernoulliNB, perbedaan skor akurasi dapat disebabkan oleh seberapa cocok distribusi data dengan asumsi yang dibuat oleh setiap varian. Misalnya GausianNB mengasumsikan bahwa data pada fitur berdistribusi normal sedangkan BernoulliNB untuk fitur bersifat biner. Jika data tidak sesuai dengan asumsi, dapat terlihat bahwa model tidak akan berperforma baik.

BAB IV
KESIMPULAN

4.1. Simpulan

Pada tugas besar II Inteligensi Buatan ini telah diimplementasikan algoritma KNN (k-Nearest Neighbors) dan Naive Bayes untuk memprediksi kolom target pada dataset yang telah disediakan. Berdasarkan tabel pengujian, diperoleh bahwa algoritma yang performanya paling baik adalah algoritma KNN dengan data tidak distandarisasi dan jumlah neighbor 17.

Dari pengujian yang dilakukan, algoritma KNN menunjukkan performa yang beragam tergantung pada parameter yang digunakan. Penggunaan normalisasi pada data, jumlah tetangga (neighbors), dan metode pembobotan (weighting) terbukti memiliki dampak signifikan terhadap akurasi. Secara khusus, algoritma KNN tanpa normalisasi data dan jumlah neighbor yang tidak terlalu besar cenderung memberikan hasil prediksi yang lebih akurat. Hal ini menunjukkan pentingnya pemilihan parameter yang tepat dalam meningkatkan performa model KNN.

Model Naive Bayes menunjukkan hasil yang stabil dan cukup baik dalam prediksi. Penggunaan semua fitur yang ada dalam model mampu memberikan skor akurasi yang memadai. Namun, terdapat risiko overfitting jika fitur yang tidak relevan ikut dimasukkan dalam model. Pemilihan fitur berdasarkan information gain dan penggunaan fitur kontinu secara spesifik dapat sedikit meningkatkan akurasi model, mengindikasikan pentingnya seleksi fitur dalam membangun model prediktif yang baik.

4.2. Saran

Tugas Besar II Semester I Tahun 2023/2024 menjadi salah satu tugas yang memberikan pelajaran baru bagi penulis. Berdasarkan pengalaman penulis mengerjakan tugas ini, berikut merupakan saran untuk pembaca yang ingin melakukan atau mengerjakan hal yang serupa. Keefektifan dalam kerja sama tim merupakan hal yang penting dalam mengerjakan tugas ini. Selain pembagian tugas yang merata, penulis terbantu oleh beberapa kali kerja kelompok dan pemakaian *real-time collaboration app*, seperti Datalore. Aplikasi tersebut membantu dalam mengolah data secara realtime.

LAMPIRAN

Kaggle dan screenshot

Submisi Kaggle dilakukan dengan semua fitur lengkap, tanpa normalisasi, dan menggunakan algoritma KNN yang dibuat sendiri.

Repository Github

https://github.com/alishalistyaa/IF3170-Tubes-2-AI.git

Lembar Kontribusi

NIM	Nama	Kontribusi
13521045	Fakhri Muhammad Mahendra	Algoritma Naive-Bayes
13521060	Fatih Nararya R.I.	Laporan KNN dan algoritma KNN (minor)
13521093	Akbar Maulana Ridho	Algoritma KNN
13521171	Alisha Listya Wardhani	Algoritma Naive-Bayes, laporan Naive Bayes, dan merapikan laporan

REFERENSI

1. T. Cover and P. Hart, "Nearest neighbor pattern classification," in *IEEE Transactions on Information Theory*, vol. 13, no. 1, pp. 21-27, January 1967, doi: 10.1109/TIT.1967.1053964.
2. Takahashi, K. (2016, January 17). Naive Bayes from Scratch in Python. Kenzo's Blog. https://kenzotakahashi.github.io/naive-bayes-from-scratch-in-python.html;
3. John, George H.; Langley, Pat (1995). Estimating Continuous Distributions in Bayesian Classifiers. Proc. Eleventh Conf. on Uncertainty in Artificial Intelligence. Morgan Kaufmann. pp. 338–345. arXiv:1302.4964
4. Hand, D. J.; Yu, K. (2001). "Idiot's Bayes — not so stupid after all?". International Statistical Review. 69(3): 385–399. doi:10.2307/1403452. ISSN 0306-7734. JSTOR 1403452