

**Tugas Besar II IF3170 Inteligensi Buatan**  
*KNN and Naive-Bayes Algorithm Implementation*  
**Semester I Tahun 2023/2024**



Dibuat oleh:

Muhammad Bangkit Dwi Cahyono	13521055
Farhan Nabil Suryono	13521114
Johanes Lee	13521148
Made Debby Almadea Putri	13521153

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

## **Daftar Isi**

<b>1. K-Nearest Neighbors Algorithm (KNN)</b>	<b>3</b>
<b>2. Naive-Bayes Algorithm</b>	<b>4</b>
<b>3. Perbandingan Hasil Prediksi</b>	<b>5</b>
<b>4. Submisi Kaggle</b>	<b>6</b>
<b>5. Kontribusi Anggota</b>	<b>7</b>

## 1. *K-Nearest Neighbors Algorithm (KNN)*

Algoritma KNN adalah salah satu algoritma dalam *Supervised Learning* yang bersifat *non-parametric* dan *lazy learning*. Konsep *non-parametric* dalam KNN menggambarkan sifat algoritma yang tidak membuat asumsi tertentu tentang distribusi data yang digunakan. Hal ini menandakan algoritma KNN tidak memiliki hipotesis dan berdasarkan pada *Instance-Base Learning*. Dengan algoritma KNN untuk kasus klasifikasi, model menyimpan data latih kemudian membuat prediksi berdasarkan mayoritas kelas dari tetangga terdekat. Penyebutan *lazy learning* untuk algoritma KNN disebabkan model tidak memiliki fase pelatihan yang signifikan melainkan seluruh kalkulasi dilakukan saat menguji atau melakukan klasifikasi data baru.

Algoritma KNN mengasumsikan bahwa objek yang mirip memiliki jarak atau perbedaan yang kecil untuk setiap nilai atribut. Dengan kata lain, data yang memiliki karakteristik serupa akan cenderung terletak berdekatan (perbedaan nilai yang kecil). Dalam hal ini, algoritma KNN menggunakan seluruh data yang tersedia untuk melakukan klasifikasi. Ketika ada data baru yang perlu diklasifikasikan, algoritma akan mengukur tingkat kemiripan atau fungsi jarak antara data baru tersebut dengan data yang sudah ada. Data baru kemudian ditempatkan dalam kelas yang paling banyak dimiliki oleh data tetangga terdekatnya. Dalam proses tersebut, hal-hal berikut penting untuk disesuaikan agar algoritma KNN bekerja dengan baik.

### 1. Menentukan metrik jarak

Untuk melakukan klasifikasi, diperlukan perhitungan jarak antara objek yang ingin diklasifikasikan dengan seluruh data latih. Dalam hal ini, berbagai perhitungan jarak seperti *Euclidean distance*, *Hamming distance*, *Manhattan distance*, dan *Minkowski distance* dapat digunakan. Namun, pada percobaan ini digunakan *Euclidean distance*.

### 2. Tahap *data-preprocessing*

Sebelum data latih digunakan oleh model KNN, perlu dilakukan *data-processing*. Hal ini disebabkan suatu atribut tertentu dapat membuat model menjadi sangat bias akibat rentang nilai yang sangat besar. Atribut yang memiliki *range* yang besar menyebabkan atribut tersebut memiliki *importance* yang besar akibat nilai jarak yang dihasilkan oleh atribut tersebut. Dalam hal ini, dapat dilakukan normalisasi ataupun standarisasi nilai atribut. Pada percobaan ini, digunakan standarisasi untuk atribut-atribut numerik.

Setelah dilakukan standarisasi ataupun normalisasi, *weighting* dapat dilakukan dengan mengalikan masing-masing atribut dengan nilai tertentu. Salah satu tujuannya

adalah membuat atribut yang memiliki korelasi yang besar terhadap *target value* juga memiliki *weight* (dampak terhadap perhitungan jarak) yang lebih besar. Dengan demikian, atribut-atribut dengan korelasi kecil terhadap atribut target tidak terlalu menjadi *noise* dalam perhitungan jarak.

### 3. Mencari nilai K terbaik

Nilai K pada algoritma KNN mendefinisikan banyak tetangga terdekat yang akan diperiksa kelasnya untuk melakukan klasifikasi. Misalnya, jika K bernilai 1, *instance* itu akan diklasifikasikan ke kelas yang sama dengan 1 tetangga terdekatnya. Pemilihan nilai K merupakan langkah penting karena dapat memengaruhi kinerja algoritma. Nilai K yang rendah dapat menyebabkan *overfitting* sehingga menyebabkan model memiliki performa buruk untuk data di luar data latih. Di sisi lain, nilai K yang tinggi dapat menghasilkan model yang terlalu sederhana atau *underfitting*. Pilihan K harus disesuaikan dengan data yang digunakan dan umumnya dipilih nilai K yang ganjil untuk menghindari kebingungan dalam klasifikasi.

Strategi pemilihan nilai K yang digunakan pada percobaan adalah dengan menggunakan strategi *k-fold cross-validation*. *Heuristic rule of thumb* berupa akar kuadrat dari jumlah sampel data latih dijadikan acuan dalam memilih rentang nilai K pada tahap *cross validation*. Berikut merupakan langkah-langkah umum dalam penentuan nilai K untuk algoritma KNN.

#### a. Pembagian Data

Data latih dibagi menjadi K bagian (*fold*) yang sama ukurannya.

#### b. Iterasi Melalui Fold

Setiap iterasi, salah satu dari K bagian dijadikan sebagai data validasi sementara K-1 bagian lainnya digunakan sebagai data latih.

#### c. Pengujian dengan Berbagai Nilai K

Model KNN dilatih menggunakan data latih dan kemudian diuji menggunakan data validasi.

#### d. Penilaian Kinerja Model

Performa model dievaluasi berdasarkan metrik evaluasi yang relevan seperti akurasi untuk setiap nilai K yang diuji.

#### e. Pemilihan K Terbaik

Nilai  $K$  yang memberikan performa terbaik (akurasi tertinggi) diambil sebagai nilai  $K$  yang optimal.

Langkah-langkah ini diulang kembali dengan pengacakan ulang pembagian data hingga seluruh *fold* telah dijadikan sebagai data validasi. Hasil penilaian kinerja model untuk setiap iterasi digabung untuk memberikan estimasi yang lebih baik tentang kinerja model menggunakan nilai  $K$  yang berbeda. Seperti yang telah dibahas, percobaan ini memanfaatkan *heuristik* untuk menentukan rentang nilai  $K$  yang digunakan pada *cross validation*.

Nilai  $K$  optimal yang telah didapatkan kemudian dapat digunakan model untuk melakukan klasifikasi. Seperti yang telah dijelaskan, model KNN tidak memiliki tahap pelatihan. Dengan demikian, setelah nilai  $K$  ditentukan, model dapat langsung digunakan untuk melakukan prediksi terhadap kelas *instance* baru.

## 2. *Naive-Bayes Algorithm*

Algoritma Naive-Bayes adalah salah satu algoritma dalam Supervised Learning karena terdapat feedback berupa data berlabel. Algoritma ini berdasarkan teorema Bayes yang mengasumsikan bahwa setiap fitur dalam dataset independen terhadap satu sama lain. Naive-Bayes termasuk dalam probabilistik classifier karena terdapat perhitungan nilai probabilitas sebuah atribut untuk suatu kelas dari himpunan kelas terbatas  $V$ . Hipotesis atau model yang dihasilkan adalah model probabilitas yang terdiri atas probabilitas setiap kelas  $P(v_j)$  dan probabilitas setiap nilai atribut untuk tiap kelas yang ada  $P(a_i | v_j)$ . Proses klasifikasi atau prediksi data baru adalah dengan menghitung peluang atau probabilitas dari semua kelas yang ada pada himpunan kelas  $V$  jika dimasukkan atribut atau fitur tertentu. Penentuan kelas akhir adalah kelas dengan peluang paling besar  $\max(P(v_j | a_1, a_2, \dots, a_n))$ .

Implementasi algoritma Naive-Bayes menggunakan *binning* untuk melakukan diskritisasi fitur numerik. Hal ini dilakukan karena *dataset* tidak terdistribusi normal sehingga *Gaussian Naive-Bayes* kurang tepat untuk digunakan.

### 1. Pemilihan kolom numerik

Kolom numerik yang digunakan hanyalah atribut *px\_height*, *px\_width*, *battery\_power*, dan *ram*. Hal ini salah satunya untuk mempercepat komputasi dengan menghilangkan atribut yang tidak terlalu berkorelasi dengan kolom target. Atribut yang disebutkan merupakan kolom dengan korelasi lebih dari 0.1.

### 2. Diskritisasi data numerik

Diskritisasi data numerik dilakukan dengan cara membagi data menjadi beberapa *bin* sehingga sebuah *bin* merupakan sebuah kategori. Pemilihan banyak bin untuk tiap atribut numerik berbeda-beda dan dipilih dari hasil *brute force* kombinasi banyak bin untuk setiap atribut.

atribut	<i>px_height</i>	<i>px_width</i>	<i>battery_power</i>	<i>ram</i>
banyak bin	6	5	4	6

### 3. Proses *fitting*

Langkah-langkah dari implementasi proses *fitting* adalah sebagai berikut:

- Melakukan diskritisasi untuk setiap kolom numerik dengan cara yang sudah dijelaskan pada nomor 2.
- Untuk setiap kelas dihitung peluang setiap kelas tersebut. Kelas disini berarti kategori pada kolom target yaitu 0, 1, 2, dan 3.
- Untuk setiap atribut, dihitung peluang kemunculan atribut tersebut untuk setiap kelas yang ada.
- Untuk setiap atribut, ditambahkan sebuah nilai sentinel sehingga tidak ada peluang kemunculan atribut pada suatu kelas yang bernilai 0. Hal ini dilakukan karena sebesar apapun peluang dari kolom lain apabila bertemu satu peluang 0 akan menjadi 0.

### 4. Proses prediksi

Langkah-langkah dari implementasi proses prediksi adalah sebagai berikut:

- Melakukan diskritisasi untuk setiap kolom numerik pada data yang diuji dengan kategori yang sama dengan data *train*

- b. Menghitung peluang dari setiap kelas pada kolom target jika dimasukkan atribut tertentu. Atribut yang digunakan sama dengan atribut saat fitting yaitu *px\_height*, *px\_width*, *battery\_power*, dan *ram*.
- c. Penentuan kelas akhir adalah kelas yang memiliki peluang tertinggi.

### 3. Perbandingan Hasil Prediksi

#### a. Algoritma KNN

Untuk hasil prediksi yang dilakukan oleh algoritma KNN, saat kami melakukan *train* dan mencari nilai K yang optimal, kami mendapatkan hasil bahwa K yang optimal pada data *train* adalah 15 dengan akurasi bernilai 0.9233. Namun, setelah kami lihat di sekitar nilai K tersebut dan mencoba untuk melatih data dengan K di sekitar 15, kami mendapatkan bahwa ternyata K=15 hanya memiliki akurasi 0.9233, dan ternyata K=10, menghasilkan akurasi yang bernilai lebih tinggi, yakni sebesar 0.93.

```
15 1389
K: 1, Test accuracy: 0.9133
K: 2, Test accuracy: 0.9133
K: 3, Test accuracy: 0.9117
K: 4, Test accuracy: 0.9167
K: 5, Test accuracy: 0.9200
K: 6, Test accuracy: 0.9233
K: 7, Test accuracy: 0.9233
K: 8, Test accuracy: 0.9283
K: 9, Test accuracy: 0.9267
K: 10, Test accuracy: 0.9300
K: 11, Test accuracy: 0.9200
K: 12, Test accuracy: 0.9283
K: 13, Test accuracy: 0.9217
K: 14, Test accuracy: 0.9250
K: 15, Test accuracy: 0.9233
K: 16, Test accuracy: 0.9283
K: 17, Test accuracy: 0.9167
K: 18, Test accuracy: 0.9250
K: 19, Test accuracy: 0.9200
Best K for test data: 10, Test accuracy: 0.9300
```

```
1 # Instantiate and Train the Model
2
3 knn_model = KNNModel(n_neighbors=best_test_k)
4
5 knn_model.fit(X_train_final, y_train.values)
6
7 # Evaluate the Model on the Validation Set
8 accuracy_val = knn_model.evaluate(X_val_final, y_val.values)
9
10 print(f"Validation accuracy: {accuracy_val}")
```

Validation accuracy: 0.93

Sebagai perbandingan, data yang sama jika dilakukan oleh *sci-kit learn*, menghasilkan hasil sebagai berikut.



```

1 # SK-LEARN CROSS VALIDATION
2
3 def find_best_k(X, y, num_folds=5):
4     best_k = None
5     best_accuracy = 0.0
6
7     sk_learn_default_k = CHOSEN_K_VALUE_SKLEARN_KNN if SKIP_CROSS_VALIDATION else heuristic_k
8     sk_learn_k_diff = 0 if SKIP_CROSS_VALIDATION else DEFAULT_K_DIFF
9
10    sk_learn_min_k = max(3, sk_learn_default_k - sk_learn_k_diff)
11    sk_learn_max_k = min(len(X_train_final), sk_learn_default_k + sk_learn_k_diff)
12
13    for k in range(sk_learn_min_k, sk_learn_max_k + 1, DEFAULT_K_JUMP):
14        knn_model = KNeighborsClassifier(n_neighbors=k)
15        kf = KFold(n_splits=num_folds, shuffle=True, random_state=42)
16        cross_val_scores = cross_val_score(knn_model, X, y, cv=kf, scoring='accuracy')
17        average_accuracy = cross_val_scores.mean()
18
19        if average_accuracy > best_accuracy:
20            best_accuracy = average_accuracy
21            best_k = k
22
23    return best_k, best_accuracy
24
25 # Assuming you have already preprocessed your data and have X_train_transformed, y_train
26 sk_learn_best_k, sk_learn_best_accuracy = find_best_k(X_combined, y_combined)
27
28 print(f"Best k: {best_k}, Best Accuracy: {sk_learn_best_accuracy:.2%}")

```

Best k: 15, Best Accuracy: 93.77%

```

1 # MODEL VALIDATION
2
3 # Instantiate the KNeighborsClassifier
4 sk_learn_knn_model = KNeighborsClassifier(n_neighbors=sk_learn_best_k)
5
6 # Train the model
7 sk_learn_knn_model.fit(X_train_final, y_train)
8
9 # Make predictions on the validation set
10 y_val_pred = knn_model.predict(X_val_final)
11
12 # Evaluate the accuracy of the model
13 accuracy = accuracy_score(y_val, y_val_pred)
14 print(f"Validation Accuracy: {accuracy:.2%}")

```

Validation Accuracy: 93.00%

Kami rasa, model yang kami buat sudah sangat baik karena akurasi setara dengan menggunakan library *sci-kit learn*, yakni dengan akurasi sebesar **93%**.

## b. Algoritma Naive-Bayes

Untuk hasil prediksi yang dihasilkan oleh algoritma Naive-bayes dengan menggunakan metode diskritisasi diperoleh sebagai berikut:

```

1 # PREDICTION
2 nb_kaggle_model = 0
3 nb_kaggle_model = NBModel(['px_height', 'px_width', 'battery_power', 'ram'], {'px_height': 6, 'px_width': 5, 'battery_
4 nb_kaggle_model.fit(X_train, y_train)
5 nb_predictions = nb_kaggle_model.predict(X_val)
6
7 accuracy = accuracy_score(y_val, nb_predictions)
8 print("Accuracy:", accuracy)

```

Accuracy: 0.81

Sebagai perbandingan, data yang sama jika dilakukan oleh *sci-kit learn* dengan algoritma Gaussian Naive Bayes (GNB), menghasilkan akurasi sebagai berikut:

```
1 nb_classifier = GaussianNB()
2 X_train_drop_col = X_train.copy()
3 nb_classifier.fit(X_train, y_train)
4
5 y_pred = nb_classifier.predict(X_val)
6
7 # Evaluate the accuracy of the model
8 accuracy = accuracy_score(y_val, y_pred)
9 print(f"Validation Accuracy: {accuracy:.2%}")
```



Validation Accuracy: 78.50%

Diperoleh model yang kami buat memiliki akurasi yang lebih tinggi (**81.00%**) dibandingkan dengan library *sci-kit learn* (**78.50%**) pada data yang sama. Hal ini disebabkan karena pada algoritma GNB mengasumsikan setiap atribut memiliki distribusi normal, sedangkan atribut pada data tidak berdistribusi normal. Asumsi ini dapat menyebabkan kesalahan perhitungan probabilitas sehingga akurasi prediksi menurun.


## 4. Submisi Kaggle

Untuk submisi kaggle pada KNN Model kami yang memiliki akurasi 93%, kami mendapatkan nilai kaggle sebagai berikut.

	<b>predictions_knn.csv</b> Complete · bangkitdc · 5d ago	<b>0.964</b>	<input type="checkbox"/>
	<b>predictions_knn_test.csv</b> Complete · Johanes Lee · 6d ago	<b>0.956</b>	<input type="checkbox"/>
	<b>predictions_knn (3).csv</b> Complete · Johanes Lee · 6d ago	<b>0.946</b>	<input type="checkbox"/>

Kami masih merasa akurasi yang kami dapatkan masih dapat ditingkatkan lagi. Pendekatan yang kami lakukan selanjutnya untuk meningkatkan nilai akurasi pada kaggle adalah dengan menambahkan data *validation* ke data *train*. Data *train* berjumlah sekitar 1400 baris dan data *validation* berjumlah sekitar 600 baris. Menurut kami, jika data *train* yang dilakukan ditambah dengan data *validation* akan memperluas data yang ada dan mengurangi bias yang ada pada model.

Oleh karena itu, untuk submisi kaggle selanjutnya, kami menyatukan antara data *train* dan data *validation* sehingga menghasilkan nilai K yang berbeda pula. Karena data yang dicoba untuk akurasi adalah data *validation* juga, maka tentunya nilai K yang didapat akan berkurang. Pertama, kami mencoba-coba untuk nilai K dan ternyata untuk nilai  $K = 1$  mendapatkan hasil di kaggle sebagai berikut.

	<b>predictions_knn (4).csv</b> Complete · bangkitdc · 8h ago	<b>0.997</b>	<input type="checkbox"/>
	<b>predictions_knn (3).csv</b> Complete · bangkitdc · 9h ago	<b>0.983</b>	<input type="checkbox"/>
	<b>predictions_knn (2).csv</b> Complete · bangkitdc · 9h ago	<b>0.978</b>	<input type="checkbox"/>

## 5. Kontribusi Anggota

No	NIM	Nama	Kontribusi
1	13521055	Muhammad Bangkit D. C.	- Implementasi KNN
2	13521114	Farhan Nabil S.	- Implementasi Naive-Bayes
3	13521148	Johanes Lee	- Implementasi KNN
4	13521153	Made Debby Almadea P.	- Implementasi Naive-Bayes