Perbandingan Metode *Naïve Bayes*, *Decision Tree*, dan *K-Nearest Neighbors* dalam Klasifikasi Data "*Fraud Detection at Self-Checkouts in Retail*"

Laporan Tugas Besar

Kelas MK Penambangan Data IF-41-GAB01 [IMD]

1301174688

Imam Nurul Ihsan

Program Studi Sarjana Informatika

Fakultas Informatika

Universitas Telkom

Bandung

2020

DAFTAR ISI

DAFTAR ISI 2
DAFTAR GAMBAR 3
DAFTAR TABEL 4
A. Latar Belakang Masalah 5
B. Tujuan 5
C. Deskripsi Data 6
D. Praproses 8
E. Analisis Pemilihan Algoritma 11
F. Analisis Penentuan Parameter 13
G. Hasil Percobaan 14
H. Ringkasan Model Yang Diperoleh 17
I. Interpretasi Model 18
Daftar Pustaka 22

DAFTAR GAMBAR

Gambar 1 Data Training 6
Gambar 2 Data Testing 6
Gambar 3 Tipe Data dan Jumlah Baris serta Kolom Data 7
Gambar 4 Pengecekkan Duplikasi Data 8
Gambar 5 Pengecekkan Missing Value 9
Gambar 6 Boxplot untuk Variabel trustLevel 9
Gambar 7 Boxplot untuk Variabel lineItemVoidsPerPosition 10
Gambar 8 Penghilangan Outlier Variabel lineItemVoidsPerPosition 1 10
Gambar 9 Penghilangan Outlier Variabel lineItemVoidsPerPosition 2 10
Gambar 10 Plot Korelasi Data 11
Gambar 11 Perhitungan Korelasi Data 11
Gambar 12 Normalisasi Data 11
Gambar 13 Algoritma Naive Bayes 13
Gambar 14 Algoritma Decision Tree 13
Gambar 15 Algoritma KNN 13
Gambar 16 Feature Importance 14
Gambar 17 Train Test Split Data 14
Gambar 18 Akurasi Naive Bayes 15
Gambar 19 Akurasi Decision Tree 16
Gambar 20 Akurasi K-Nearest Neighbors 16
Gambar 21 Prediksi Naive Bayes 1 19
Gambar 22 Prediksi Naive Bayes 2 19
Gambar 23 Prediksi Decision Tree 1 20
Gambar 24 Prediksi Decision Tree 2 20
Gambar 25 Prediksi K-Nearest Neighbors 1 21
Gambar 26 Prediksi K-Nearest Neighbors 2 21

DAFTAR TABEL

Tabel 1 Karakteristik Data 7
Tabel 2 Confussion Matrix 17
Tabel 3 Confussion Matrix Naive Bayes 17
Tabel 4 Confussion Matrix Decision Tree 18
Tabel 5 Confussion Matrix K-Nearest Neighbors 18

A. Latar Belakang Masalah

Data mining merupakan sebuah studi ilmu teknologi yang mempelajari data dalam jumlah yang besar, mulai dari mengumpulkan (*collecting*) data, membersihkan (*cleaning*) data, *processing* data, menganalisis data, sampai dengan mengekstraksi *knowledge* atau pengetahuan yang menarik dan bermanfaat dari sebuah data yang tersimpan dalam basis data. *Data mining* juga merupakan inti dari proses KDD (*Knowledge Discovering in Database*). Jenis data yang digunakan diantaranya bersumber dari *database*, *data warehouse*, ataupun data dari internet. Proses-proses dalam *data mining* yaitu, *data collection, data cleaning, feature selection/extraction,* dan terakhir proses analitik lalu pengimplementasian algoritma. Salah satu *task* yang ada dalam *data mining* yaitu prediksi. *Task* ini menggunakan beberapa atribut untuk mengetahui atau memprediksi nilai yang sebelumnya belum diketahui (*unknown*) atau nilai selanjutnya (*future*) dari atribut lain. Contoh dari *task* ini yaitu klasifikasi, regresi, dan *deviation detection* [1]. Penggunaan *data mining* sangatlah luas. Begitupun dalam mendeteksi penipuan dalam pembayaran secara mandiri di supermarket.

Kita tahu bahwa teknologi berkembang dengan sangat cepat. Begitupun dampaknya dalam pembayaran barang di supermarket. Pembayaran mandiri sudah sangat meningkat penggunaannya di beberapa supermarket. Hal ini sangat menguntungkan pelanggan dimana mereka bisa menghindari antrian yang panjang untuk memindai dan membayar produk mereka secara langsung menggunakan *smartphone.* Dengan begitu, proses pembelian dan pembayaran berlangsung sangat cepat.

Tetapi, hal ini tidak sepenuhnya menguntungkan bagi para penjual. Seringkali banyak pelanggan “nakal” yang menyalahgunakan teknologi ini dengan tidak melakukan pembayaran. Permasalahan ini tentu harus diatasi untuk mencegah kerugian bagi para penjual. Dengan begitu untuk mengatasi hal ini, penulis tertarik untuk membangun system klasifikasi data menggunakan beberapa algoritma seperti *Naïve Bayes*, *Decision Tree*, dan *K-Nearest Neighbors* untuk mendeteksi penipuan dalam pembayaran mandiri di supermarket.

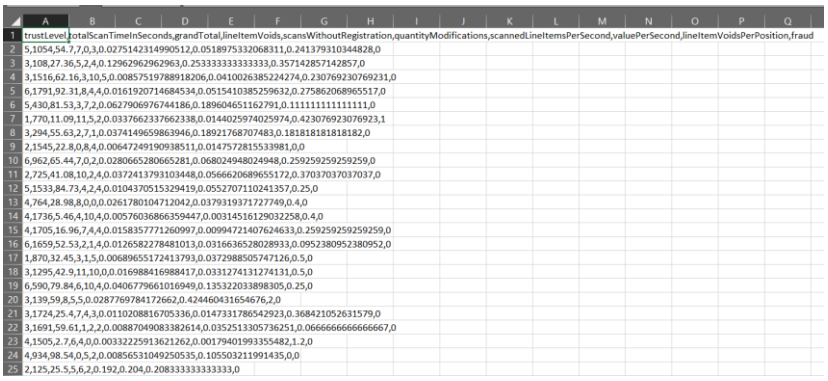
B. Tujuan

Tujuan yang akan dicapai yaitu:

1. Melakukan proses klasifikasi data dengan menggunakan algoritma *Naïve Bayes*, *Decision Tree*, dan *K-Nearest Neighbors*.
2. Membandingkan performansi algortima *Naïve Bayes*, *Decision Tree*, dan *K-Nearest Neighbors* dalam klasifikasi data "Fraud Detection at Self-Checkouts in Retail".
3. Mendapatkan akurasi terbaik dari algoritma yang dipilih.

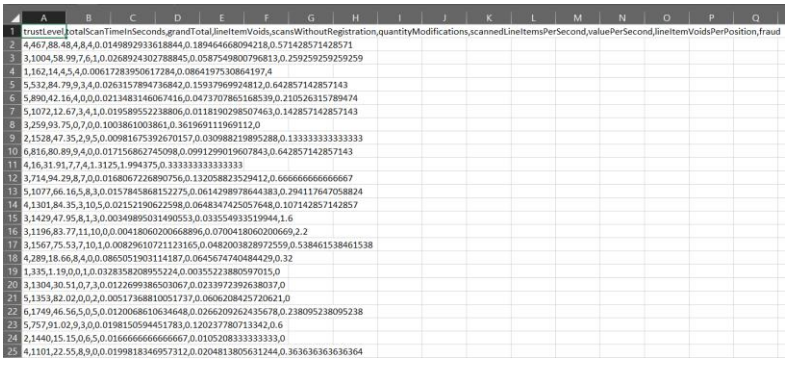
C. Deskripsi Data

Dataset yang digunakan dalam penelitian kali ini bersumber dari https://www.data-mining-cup.com/reviews/dmc-2019/. *Dataset* terbagi menjadi dua, yaitu *data training* dan *data testing*. *Data training* terdiri dari 1879 baris data dan jumlah atribut 10. Untuk *data testing* terdiri dari 498121 baris data dengan jumlah atribut 10.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	trustLevel,totalScanTimeInSeconds,grandTotal,lineItemVoids,scansWithoutRegistration,quantityModifications,scannedLineItemsPerSecond,valuePerSecond,lineItemVoidsPerPosition,fraud																
2	5,1054,54.7,7,0,3,0.0275142314990512,0.0518975332068311,0.2413793103444828,0																
3	3,108,27.36,5,2,4,0.12962962962963,0.2533333333333333,0.357142857142857,0																
4	3,1516,62.16,3,10,5,0.008575197889182 06,0.04100263852242 74,0.2307692307692 31,0																
5	6,1791,92.31,8,4,4,0.016192071468453 4,0.05154103852596 32,0.2758620689655 17,0																
6	5,430,81.53,3,7,2,0.06279069767441 86,0.18960465116279 1,0.1111111111111 11,0																
7	1,770,11.09,11,5,2,0.03376623376623 38,0.01440259740259 74,0.4230769230769 23,1																
8	3,294,55.63,2,7,1,0.03741496598639 46,0.18921768707483,0.181818181818182,0																
9	2,1545,22.8,0,8,4,0.006472491909385 11,0.01475728155339 81,0,0																
10	6,962,65.44,7,0,2,0.02806652806652 81,0.06802494802494 8,0.2592592592592 59,0																
11	2,725,41.08,10,2,4,0.03724137931034 48,0.05666206896551 72,0.3703703703703 7,0																
12	5,1533,84.73,4,2,4,0.01043705153294 19,0.05527071102413 57,0.25,0																
13	4,764,28.98,8,0,0,0.026178010471204 2,0.03793193717277 49,0.4,0																
14	4,1736,5.46,4,10,4,0.005760368663594 47,0.00314516129032258,0.4,0																
15	4,1705,16.96,7,4,4,0.015835771260997,0.009947214076246 33,0.2592592592592 59,0																
16	6,1659,52.53,2,1,4,0.012658227848101 3,0.03166365280289 33,0.0952380952380 952,0																
17	1,870,32.45,3,1,5,0.006896551724137 93,0.03729885057471 26,0.5,0																
18	3,1295,42.9,11,10,0,0.016988416988417,0.03312741312741 31,0.5,0																
19	6,590,79.84,6,10,4,0.040677966101694 9,0.13532033898305,0.25,0																
20	3,139,59,8,5,5,0.028776978417266 2,0.424460431654676,2,0																
21	3,1724,25.4,7,4,3,0.011020881670533 6,0.01473317865429 23,0.3684210526315 79,0																
22	3,1691,59.61,1,2,2,0.008870490833826 14,0.03525133057362 51,0.0666666666666 67,0																
23	4,1505,2.7,6,4,0,0.003322259136212 62,0.00179401993355482,1.2,0																
24	4,934,98.54,0,5,2,0.008565310492505 35,0.10550321199143 5,0,0																
25	2,125,25.5,5,6,2,0.192,0.204,0.2083333333333333,0																

Gambar 1 Data Training



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	trustLevel,totalScanTimeInSeconds,grandTotal,lineItemVoids,scansWithoutRegistration,quantityModifications,scannedLineItemsPerSecond,valuePerSecond,lineItemVoidsPerPosition,fraud																
2	4,467,88.48,4,8,4,0.014989293361844,0.1894646680942 18,0.5714285714285 71																
3	3,1004,58.99,7,6,1,0.026892430278884 5,0.05875498007968 13,0.2592592592592 59																
4	1,162,14,4,5,4,0.006172839506172 84,0.08641975308641 97,4																
5	5,532,84.79,9,3,4,0.026315789473684 2,0.15937969924812,0.642857142857143																
6	5,890,42.16,4,0,0,0.021348314606741 6,0.04737078651685 39,0.2105263157894 74																
7	5,1072,12.67,3,4,1,0.019589552238806,0.01181902985074 63,0.142857142857143																
8	3,259,93.75,0,7,0,0.1003861003861,0.3619691119691 12,0																
9	2,1528,47.35,2,9,5,0.009816753926701 57,0.03098821989528 8,0.1333333333333 33																
10	6,816,80.89,9,4,0,0.017156862745098,0.09912990196078 43,0.642857142857143																
11	4,16,31.91,7,7,4,1.3125,1.994375,0.3333333333333333																
12	3,714,94.29,8,7,0,0.016806722689075 6,0.13205882352941 2,0.6666666666666 67																
13	5,1077,66.16,5,8,3,0.015784586815227 5,0.06142989786443 83,0.2941176470588 24																
14	4,1301,84.35,3,10,5,0.021521906225 98,0.06483474250576 48,0.1071428571428 57																
15	3,1429,47.95,8,1,3,0.003498950314905 53,0.03355493351994 4,1.6																
16	3,1196,83.77,11,10,0,0.004180602006688 96,0.07004180602006 69,2.2																
17	3,1567,75.53,7,10,1,0.008296107211231 65,0.04820038289725 59,0.5384615384615 38																
18	4,289,18.66,8,4,0,0.086505190311418 7,0.06456747404844 29,0.32																
19	1,335,1.19,0,0,1,0.032835820895522 4,0.00355223880597015,0																
20	3,1304,30.51,0,7,3,0.012269938650306 7,0.02339723926380 37,0																
21	5,1353,82.02,0,0,2,0.005173688110051 737,0.06062084257206 21,0																
22	6,1749,46.56,5,0,5,0.012006861063464 8,0.02662092624356 78,0.238095238095238																
23	5,757,91.02,9,3,0,0.019815059445178 3,0.12023778071334 2,0.6																
24	2,1440,15.15,0,6,5,0.016666666666667,0.01052083333333333,0																
25	4,1101,22.55,8,9,0,0.019981834695731 2,0.02048138056312 44,0.3636363636363 64																

Gambar 2 Data Testing

Berikut merupakan penjelasan karakteristik dari *dataset* yang digunakan dalam penelitian ini:

Nama Atribut	Deskripsi	*Range* Nilai
trustLevel	Tingkat kepercayaan individu pelanggan.	{1,2,3,4,5,6}

	*6: Dapat dipercaya tertinggi.	
totalScanTimeInSeconds	Total waktu dalam detik antara produk pertama dan terakhir dipindai.	Bilangan bulat positif.
grandTotal	Total keseluruhan produk yang dipindai.	Angka desimal positif dengan maksimum dua tempat decimal.
ineItemVoids	Jumlah pemindaian yang dibatalkan.	Bilangan bulat positif.
scansWithoutRegistration	Jumlah upaya untuk mengaktifkan pemindai tanpa benar-benar memindai apa pun.	Bilangan bulat positif atau 0.
quantityModification	Jumlah kuantitas yang dimodifikasi untuk salah satu produk yang dipindai.	Bilangan bulat positif atau o.
scannedLineItemsPerSecond	Jumlah rata-rata produk yang dipindai per detik.	Angka desimal positif.
valuePerSecond	Nilai total rata-rata produk yang dipindai per detik.	Angka desimal positif.
lineItemVoidsPerPosition	Jumlah rata-rata kekosongan item per jumlah total semua produk yang dipindai dan tidak dibatalkan.	Angka desimal positif.
fraud	Klasifikasi sebagai *fraud*/penipuan (1) atau bukan *fraud*/penipuan (0)	{0,1}

Tabel 1 Karakteristik Data

Dataset ini merupakan *dataset* yang berkualitas dimana tidak adanya *missing value* dan duplikasi data.

```
# Melihat jumlah baris dan kolom
data_train.shape
```

```
(1879, 10)
```

```
# Tipe data setiap kolom
data_train.dtypes
```

```
trustLevel                    int64
totalScanTimeInSeconds        int64
grandTotal                  float64
lineItemVoids                 int64
scansWithoutRegistration      int64
quantityModifications         int64
scannedLineItemsPerSecond   float64
valuePerSecond              float64
lineItemVoidsPerPosition    float64
fraud                         int64
dtype: object
```

Gambar 3 Tipe Data dan Jumlah Baris serta Kolom Data

D. Praproses

Pre-processing data merupakan suatu strategi dan teknik yang ada dalam *data mining* untuk membuat data menjadi lebih berkualitas. *Pre-processing* merupakan salah satu tahap awal yang terpenting dalam menentukan keberhasilan proses *data mining*. Dengan adanya transformasi data yang sesuai dengan kebutuhan pemrosesan, maka hal tersebut akan berpengaruh pada hasil akhir [2]. Tujuan dari proses ini yaitu meningkatkan hasil analisis *data mining* terkait dengan waktu, biaya, dan kualitas. Salah satu teknik dalam *pre-pocessing* yang akan digunakan dalam penelitian ini adalah *data cleaning*. Teknik *data cleaning* bisa digunakan untuk mengisi *missing value*, identifikasi *outliers*, dan menyelesaikan permasalahan ketidaksesuaian data [2].

Teknik *pre-processing* yang dilakukan dalam penelitian kali ini adalah mengecek duplikasi data, *missing value*, *outiler*, penghitungan korelasi data, dan normalisasi. *Dataset* mungkin saja terdapat objek data yang duplikat. Oleh karena itu harus dilakukan *data* cleaning. Selain itu, ada *missing value yang* menyebabkan suatu informasi tidak terkumpul dengan lengkap. Solusi penangan *missing values* diantaranya mengeliminasi objek data, mengestimasi *missing values*, tidak memperhatikan *missing value* saat analisis, dan menggantikan dengan semua kemungkinan nilai (pembobotan berdasarkan probabilitasnya). Selanjutnya *outlier* merupakan objek data yang berbeda karakteristik dengan objek data kebanyakan. Jenis *outlier* diantaranya adalah *outlier* yang mengganggu analisis dan *outlier* yang penting untuk dianalisis. Korelasi data adalah pengukuran untuk mengetahui hubungan linier antara objek-objek. Normalisasi data dilakukan untuk penskalaan nilai atribut dari data sehingga nilai data berada pada range tertentu. Teknik normalisasi yang dilakukan adalah Minmax. Minmax merupakan metode normalisasi dengan melakukan transformasi linier terhadap data asli. Hal tersebut dilakukan agar *dataset* lebih berkualitas untuk digunakan dalam pengklasifikasian data.

Berikut hasil teknik *pre-processing* dalam penelitian kali ini.

1. Pengecekkan Duplikasi Data

```
cekDuplikat = data_train.drop_duplicates(keep= False, inplace= True)
print(cekDuplikat)
```

```
None
```

Gambar 4 Pengecekkan Duplikasi Data

Bisa dibuktikan bahwa dalam *dataset* ini tidak terdapat duplikasi data.

2. Pengecekkan *Missing Value*

```
data_train.isnull().sum()
```

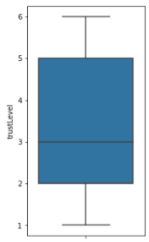
```
trustLevel                  0
totalScanTimeInSeconds      0
grandTotal                  0
lineItemVoids               0
scansWithoutRegistration    0
quantityModifications       0
scannedLineItemsPerSecond   0
valuePerSecond              0
lineItemVoidsPerPosition    0
fraud                       0
dtype: int64
```

Gambar 5 Pengecekkan Missing Value

Bisa dibuktikan bahwa dalam *dataset* ini tidak terdapat adanya *missing value*.

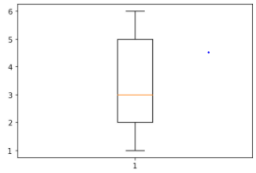
3. Pengecekkan *Outlier*

```
plt.figure(figsize=(3,6))
sns.boxplot(y='trustLevel', data=data_train,showfliers=False);
```




```
plt.boxplot(data_train['trustLevel'])
data_train['trustLevel'].describe()
```

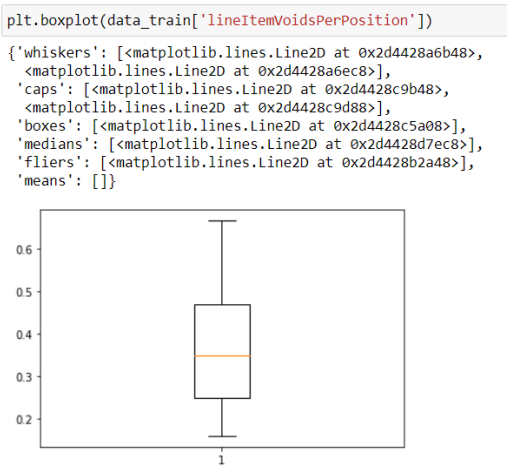
```
count    1879.000000
mean        3.401809
std         1.709404
min         1.000000
25%         2.000000
50%         3.000000
75%         5.000000
max         6.000000
Name: trustLevel, dtype: float64
```



Gambar 6 Boxplot untuk Variabel trustLevel

Gambar 7 Boxplot untuk Variabel lineItemVoidsPerPosition

Gambar 8 Penghilangan Outlier Variabel lineItemVoidsPerPosition 1

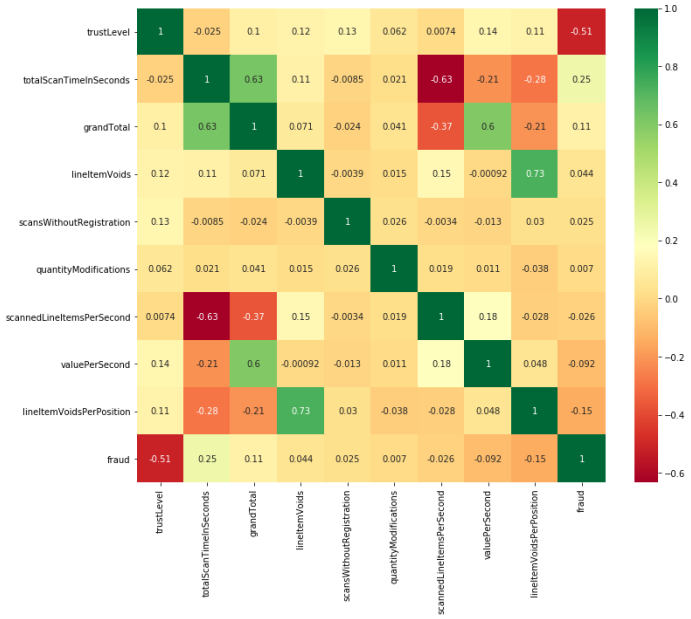


Gambar 9 Penghilangan Outlier Variabel lineItemVoidsPerPosition 2

Bisa dibuktikan bahwa dalam *dataset* ini terdapat data yang termasuk kedalam *outlier*. Contohnya dalam variabel lineItemVoidsPerPosition. Hal ini dilakukan penghapusan *outlier*. Penghapusan *outlier* pada dataset ini dilakukan dengan IQR. Pemilihan IQR dipilih karena ingin menandai poin sebagai "*far outliers*" atau outliers

kesalahan ekstrim agar tidak sembarangan dalam mendrop data *outlier* sehingga tidak mengurangi kualitas dari data tersebut.

4. Penghitungan Korelasi



Gambar 10 Plot Korelasi Data

	trustLevel	totalScanTimeInSeconds	grandTotal	lineItemVoids	scansWithoutRegistration	quantityModifications	scannedLineItemsPe
trustLevel	1.000000	-0.024948	0.103686	0.120705	0.132895	0.061527	0
totalScanTimeInSeconds	-0.024948	1.000000	0.626141	0.107065	-0.008474	0.021292	-0
grandTotal	0.103686	0.626141	1.000000	0.071234	-0.023717	0.041304	-0
lineItemVoids	0.120705	0.107065	0.071234	1.000000	-0.003910	0.014895	0
scansWithoutRegistration	0.132895	-0.008474	-0.023717	-0.003910	1.000000	0.026107	-0
quantityModifications	0.061527	0.021292	0.041304	0.014895	0.026107	1.000000	0
scannedLineItemsPerSecond	0.007441	-0.633287	-0.366228	0.147232	-0.003432	0.019430	
valuePerSecond	0.136289	-0.213530	0.599572	-0.000919	-0.012548	0.010825	0
lineItemVoidsPerPosition	0.113809	-0.282647	-0.207748	0.726872	0.030141	-0.037809	-0
fraud	-0.514081	0.251497	0.113001	0.043623	0.024787	0.006964	-0

Gambar 11 Perhitungan Korelasi Data

5. Normalisasi

```
from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
X =  pd.DataFrame(sc_X.fit_transform(data_train.drop(["fraud"],axis = 1),),
        columns=['trustLevel', 'scannedLineItemsPerSecond','totalScanTimeInSeconds',"quantityModifications","scansWithoutRegistra
```

Gambar 12 Normalisasi Data

E. Analisis Pemilihan Algoritma

Salah satu *task* yang ada dalam *data mining* yaitu prediksi. *Task* ini menggunakan beberapa atribut untuk mengetahui atau memprediksi nilai yang sebelumnya belum diketahui (*unknown*) atau nilai selanjutnya (*future*) dari atribut lain. Contoh dari *task* ini yaitu klasifikasi, regresi, dan *deviation detection*.

Pada penelitan kali ini, akan dilakukan klasifikasi data. Dimana klasifikasi dibagi menjadi 3, yaitu klasifikasi menggunakan algoritma *naïve bayes, decision tree,* dan *k-nearest neighbors*

Naïve Bayes adalah salah satu algoritma yang digunakan untuk klasifikasi probabilistik di dalam *data mining*. *Naïve Bayes* dikenalkan oleh ilmuwan asal Inggris bernama Thomas Bayes. Algoritma *Naive Bayes* adalah pengklasifikasi probabilistik sederhana yang menghitung sekumpulan probabilitas dengan menghitung frekuensi dan kombinasi nilai dalam kumpulan data tertentu. *Naïve Bayes* mendefinisikan seluruh atribut independen atau tidak saling berkaitan satu sama lain. Metode ini dianggap sebagai metode yang bagus dan efisien untuk klasifikasi. Teorema *Naïve Bayes* sering digunakan untuk melakukan klasifikasi sederhana maupun kompleks. Keuntungan menggunakan metode *Naïve Bayes* adalah model ini mudah dibuat tanpa estimasi parameter berulang yang rumit yang membuatnya sangat berfungsi untuk kumpulan data yang sangat besar.

Decision tree adalah salah satu metode klasifikasi yang paling populer, karena mudah untuk diinterpretasi oleh manusia. *Decision tree* adalah model prediksi menggunakan struktur pohon atau struktur berhirarki. Konsep dari pohon keputusan adalah mengubah data menjadi *decision tree* dan aturan-aturan keputusan. Manfaat utama dari penggunaan *decision tree* adalah kemampuannya untuk mem-*break down* proses pengambilan keputusan yang kompleks menjadi lebih *simple*, sehingga pengambil keputusan akan lebih menginterpretasikan solusi dari permasalahan.

k-Nearest Neighbor adalah salah satu algoritma yang termasuk pada *supervised learning*. *Supervised learning* merupakan pendekatan dimana sudah terdapat atribut yang ditargetkan sehingga tujuan dari pendekatan ini adalah mengkelompokan data ke data yang sudah ada. Algoritma KNN akan mengklasifikasikan data berdasarkan *data train* yang diambil dari tetangga terdekatnya. *K-Nearest Neighbor* merupakan salah satu metode yang paling banyak digunakan dalam *data mining*. Hal ini didasarkan pada sebuah gagasan bahwa “objek yang berdekatan satu sama lain memiliki karakterisktik yang sama”.

```
from sklearn.naive_bayes import GaussianNB
gaussian = GaussianNB()
gaussian.fit(X_train, y_train)
y_pred = gaussian.predict(X_test)
y_pred
```

```
array([0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 0], dtype=int64)
```

Gambar 13 Algoritma Naive Bayes

```
from sklearn.tree import DecisionTreeClassifier
decision_tree = DecisionTreeClassifier()
decision_tree.fit(X_train, y_train)
y_pred = decision_tree.predict(X_test)
y_pred
```

```
array([0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1,
       1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 0], dtype=int64)
```

Gambar 14 Algoritma Decision Tree

```
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors=4)
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
y_pred
```

```
array([0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0], dtype=int64)
```

Gambar 15 Algoritma KNN

F. Analisis Penentuan Parameter

Feature selection merupakan tahap yang biasanya dilakukan sebelum menerapkan algoritma yang akan digunakan. Contoh dari *feature selection* misalnya adalah atribut jenis kelamin dalam *dataset* kurang relevan untuk memprediksi penyakit diabetes dibandingkan dengan atribut umur. Apabila menggunakan fitur yang kurang relevan, maka akan memungkin untuk merusak keakuratan model klasifikasi yang akan dibuat. Oleh karena itu, dilakukan *feature selection* yang bertujuan untuk mengesampingkan fitur/atribut yang tidak relevan dan memilih beberapa fitur/atribut yang berpengaruh untuk digunakan dalam proses klasifikasi data. Tiga jenis metode atau pendekatan *feature selection* adalah pendekatan *wrapper*, *filter*, dan *embedded.* Pada penelitian kali ini, penulis akan menggunakan pendekata *embedded*. Pendekatan *embedded* akan dilakukan sebagai bagian dari algoritma *data mining* dan akan memberikan rekomendasi atribut berpengaruh. Pendekatan *embedded* akan menangani

setiap iterasi dari *data train* dan mengekstraksi fitur-fitur yang akan berkontribusi paling besar untuk diterapkan pada proses klasifikasi. Embedded *model* memanfaatkan suatu *learning machine* diantaranya adalah *Decision Tree* dan *Random Forest*.

```
from sklearn.ensemble import RandomForestClassifier
model= RandomForestClassifier(n_estimators=100,random_state=0)
X=data_train[data_train.columns[:9]]
Y=data_train['fraud']
model.fit(X,Y)
pd.Series(model.feature_importances_,index=X.columns).sort_values(ascending=False)
```

```
trustLevel                  0.349842
totalScanTimeInSeconds      0.145916
grandTotal                  0.124942
valuePerSecond              0.101868
scannedLineItemsPerSecond   0.089816
lineItemVoidsPerPosition    0.057838
scansWithoutRegistration    0.047446
lineItemVoids               0.044251
quantityModifications       0.038080
dtype: float64
```

Gambar 16 Feature Importance

Bisa dilihat bahwa dengan melakukan *feature selection* dengan pendekatan *embedded* didapatkan beberapa fitur yang relevan. Fitur *importance* tertinggi yaitu trustLevel diikuti dengan totalScanTimeInSeconds, grandTotal, dan valuePerSecond.

```
#importing train_test_split
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.20,random_state=0, stratify=y)
```

Gambar 17 Train Test Split Data

Untuk proses *train test split data* ditentukan bahwa x adalah variabel yang dipilih sesuai dengan *selection feature* dan y adalah variabel “fraud”, Ditentukan juga bahwa *test size* nya adalah 0.20 dengan *random state* 0.

G. Hasil Percobaan

Berikut merupakan hasil penelitian dari 3 algoritma berbeda yaitu, *decision tree, naïve bayes,* dan *k-nearest neigbhors* dengan berdasarkan tujuan penelitian dan penentuan sesuai dengan poin B dan G.

Penelitian ini menggunakan beberapa pengukuran evaluasi performasi seperti *precision, recall (sensitivity)* dan *f1-score*.

1. Precision

Presisi adalah tingkat ketepatan antara informasi yang diminta oleh pengguna dengan jawaban yang diberikan oleh sistem. Persamaan presisi adalah sebagai berikut:

$$Precision = \frac{TP}{FP+TP} \times 100\% \quad (1)$$

2. Recall (Sensitivity)

Recall adalah tingkat keberhasilan sistem dalam menemukan kembali sebuah informasi. Persamaan *recall* adalah sebagai berikut:

$$Recall = \frac{TP}{FN+TP} \times 100\% \quad (2)$$

3. Akurasi

Akurasi didefinisikan sebagai tingkat kedekatan antara nilai prediksi dengan nilai aktual. Semakin besar nilai akurasi, maka performansi sistem klasifikasi semakin baik. Persamaan akurasi adalah sebagai berikut:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \times 100\% \quad (3)$$

4. *F1-Score*

F1 *Score* adalah perbandingan rata-rata antara *precission* dan *recall*. *Score* ini akan memperhitungkan *false positive* dan *false negative.* Persamaan F1 *Score* adalah sebagai berikut:

$$F1\ Score = 2\ \times \frac{Recall \times Precission}{Recall+Precission} \quad (4)$$

Berikut merupakan hasil dari penelitian menggunakan evaluasi performansi:

1. *Naïve Bayes*

```
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn import metrics
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
[[38  3]
 [ 4  2]]
              precision    recall  f1-score   support

           0       0.90      0.93      0.92        41
           1       0.40      0.33      0.36         6

    accuracy                           0.85        47
   macro avg       0.65      0.63      0.64        47
weighted avg       0.84      0.85      0.85        47

Accuracy: 0.851063829787234
```

Gambar 18 Akurasi Naive Bayes

Bisa dilihat bahwa dalam penelitian yang menggunakan algoritma *naïve bayes* didapatkan akurasi sebesar 85.1% dengan *precision, recall,* dan *f1-score* masing-masing sebesar 90%, 93%, dan 92 %.

2. *Decision Tree*

```
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn import metrics
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
[[38  3]
 [ 3  3]]
              precision    recall  f1-score   support

           0       0.93      0.93      0.93        41
           1       0.50      0.50      0.50         6

    accuracy                           0.87        47
   macro avg       0.71      0.71      0.71        47
weighted avg       0.87      0.87      0.87        47

Accuracy: 0.8723404255319149
```

Gambar 19 Akurasi Decision Tree

Bisa dilihat bahwa dalam penelitian yang menggunakan algoritma *decision tree* didapatkan akurasi sebesar 87.2 % dengan *precision*, *recall*, dan *f1-score* masing-masing sebesar 93%.

3. *K-Nearest Neighbors*

```
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn import metrics
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
[[41  0]
 [ 5  1]]
              precision    recall  f1-score   support

           0       0.89      1.00      0.94        41
           1       1.00      0.17      0.29         6

    accuracy                           0.89        47
   macro avg       0.95      0.58      0.61        47
weighted avg       0.91      0.89      0.86        47

Accuracy: 0.8936170212765957
```

Gambar 20 Akurasi K-Nearest Neighbors

Bisa dilihat bahwa dalam penelitian yang menggunakan algoritma *k-nearest neighbors* didapatkan akurasi sebesar 89.3% dengan *precision*, *recall*, dan *f1-score* masing-masing sebesar 89%, 100%, dan 94 %.

Dapat disimpulkan bahwa algoritma yang paling baik dalam penelitian ini adalah algoritma *k-nearest neighbors* diikuti dengan algoritma *decision tree* dan *naïve bayes.* Akurasi terbesar dalam penelitian ini dengan algoritma *k-nearest neighbors* yaitu sebesar 89.3 %.

H. Ringkasan Model Yang Diperoleh

Pada penelitian kali ini dilakukan penghitungan akurasi, *precision, recal,* dan *f1-score*. Selain perhitungan evaluasi performasi, pada penelitian ini juga dilakukan evaluasi performansi menggunakan tabel *confusion matrix.* Istilah dan tabel dalam *confusion matrix* adalah sebagai berikut:

Kategori	**Kelas Aktual**		
Kelas Prediksi		**Positif**	**Negatif**
	Positif	TP	FP (*Type I Error*)
	Negatif	FN (*Type II Error*)	TN

Tabel 2 Confussion Matrix

- *True Positive* (TP): ketika prediksi *yes* dan faktanya *yes* (hasilnya benar).
- *True Negative* (TN): ketika prediksi *no* dan faktanya *no* (tidak ada hasil yang benar).
- *False Positive* (FP): ketika prediksi *yes* dan faktanya *no* (hasilnya tidak diharapkan).
- *False Negative* (FN): ketika prediksi *no* dan faktanya *yes* (hasilnya meleset).

Berikut merupakan hasil tabel *confusion matrix* untuk ketiga algoritma yang telah ditentukan:

1. *Naïve Bayes*

Kategori	**Kelas Aktual**		
Kelas Prediksi		**0 (no fraud)**	**1 (fraud)**
	0 (no fraud)	38	3
	1 (fraud)	4	2

Tabel 3 Confussion Matrix Naive Bayes

- *True Positive* (TP): 38.
- *True Negative* (TN): 2.
- *False Positive* (FP): 3.
- *False Negative* (FN): 4.

2. *Decision Tree*

Kategori	Kelas Aktual		
Kelas Prediksi		**0 (no fraud)**	**1 (fraud)**
	0 (no fraud)	38	3
	1 (fraud)	3	3

Tabel 4 Confussion Matrix Decision Tree

- *True Positive* (TP): 38.
- *True Negative* (TN): 3.
- *False Positive* (FP): 3.
- *False Negative* (FN): 3.

3. *K-Nearest Neighbors*

Kategori	Kelas Aktual		
Kelas Prediksi		**0 (no fraud)**	**1 (fraud)**
	0 (no fraud)	41	0
	1 (fraud)	5	1

Tabel 5 Confussion Matrix K-Nearest Neighbors

- *True Positive* (TP): 41.
- *True Negative* (TN): 1.
- *False Positive* (FP): 0.
- *False Negative* (FN): 5.

I. Interpretasi Model

Berdasarkan tujuan yang telah ditentukan pada poin B menurut penulis, penelitian ini sudah tercapai. Tujuan itu diantaranya adalah:

1. Melakukan proses klasifikasi data dengan menggunakan algoritma *Naïve Bayes*, *Decision Tree*, dan *K-Nearest Neighbors*.
2. Membandingkan performansi algortima *Naïve Bayes*, *Decision Tree*, dan *K-Nearest Neighbors* dalam klasifikasi data "Fraud Detection at Self-Checkouts in Retail".
3. Mendapatkan akurasi terbaik dari algoritma yang dipilih.
4. Dari ketika tujuan tersebut, alasan dimana penelitian ini tercapai diantaranya

adalah: penulis sudah melakukan proses klasifikasi data dengan menggunakan algoritma *Naïve Bayes*, *Decision Tree*, dan *K-Nearest Neighbors*, penulis juga telah melakukan proses membandingkan performansi algortima *Naïve Bayes*, *Decision Tree*, dan *K-Nearest Neighbors* dalam klasifikasi data "Fraud Detection at Self-Checkouts in Retail", dan yang terakhir adalah mendapatkan akurasi terbaik dari algoritma yang dipilih. Dan pada kasus ini, algoritma terbaik adalah algoritma *k-nearest neighbors* dengan akurasi sebesar 89.2%. Berikut merupakan ringkasan interpretasi model dan hasil prediksi untuk kelas "fraud" menggunakan algoritma *Naïve Bayes*, *Decision Tree*, dan *K-Nearest Neighbors.*

1. *Naïve Bayes*

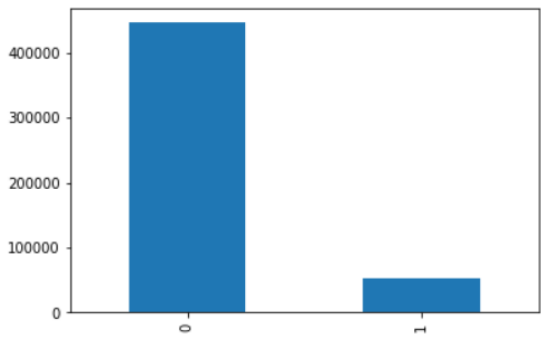
```
data_hasil_from_csv = pd.read_csv('HasilPrediksiNB.csv')
data_hasil_from_csv.drop(columns=['Unnamed: 0'])
```

econds	grandTotal	lineItemVoids	scansWithoutRegistration	quantityModifications	scannedLineItemsPerSecond	valuePerSecond	lineItemVoidsPerPosition	fraud
467	88.48	4	8	4	0.014989	0.189465	0.571429	0
1004	58.99	7	6	1	0.026892	0.058755	0.259259	0
162	14.00	4	5	4	0.006173	0.086420	4.000000	0
532	84.79	9	3	4	0.026316	0.159380	0.642857	0
890	42.16	4	0	0	0.021348	0.047371	0.210526	0
...	...	...	...	...	...	...	...	...
783	59.10	2	2	0	0.012771	0.075479	0.200000	0
278	98.90	9	5	4	0.050360	0.355755	0.642857	1
300	5.41	6	6	4	0.030000	0.018033	0.666667	0
1524	33.97	2	5	3	0.005906	0.022290	0.222222	0
1456	56.97	11	7	2	0.019231	0.039128	0.392857	0

Gambar 21 Prediksi Naive Bayes 1

```
## checking the balance of the data by plotting the count of outcomes by their value
color_wheel = {1: "#0392cf",
               2: "#7bc043"}
colors = data_hasil_from_csv["fraud"].map(lambda x: color_wheel.get(x + 1))
print(data_hasil_from_csv.fraud.value_counts())
p=data_hasil_from_csv.fraud.value_counts().plot(kind="bar")
```

```
0    446491
1     51630
Name: fraud, dtype: int64
```



Gambar 22 Prediksi Naive Bayes 2

Pada penelitian yang menggunakan algoritma *Naïve Bayes* didapatkan bahwa prediksi untuk kelas *fraud* yaitu untuk variabel 0 atau *no fraud* sebesar 446.491

dan variabel 1 atau *fraud* sebesar 51.630.

2. *Decision Tree*

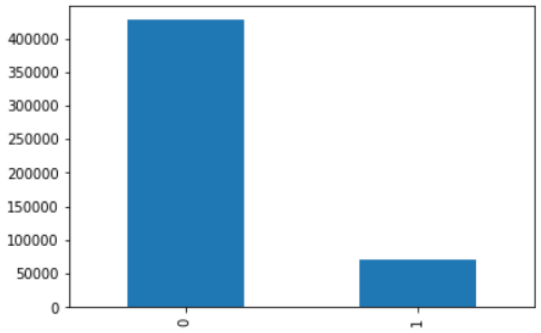
```
data_hasil_from_csv = pd.read_csv('HasilPrediksiDT.csv')
data_hasil_from_csv.drop(columns=['Unnamed: 0'])
```

seconds	grandTotal	lineItemVoids	scansWithoutRegistration	quantityModifications	scannedLineItemsPerSecond	valuePerSecond	lineItemVoidsPerPosition	fraud
467	88.48	4	8	4	0.014989	0.189465	0.571429	0
1004	58.99	7	6	1	0.026892	0.058755	0.259259	0
162	14.00	4	5	4	0.006173	0.086420	4.000000	0
532	84.79	9	3	4	0.026316	0.159380	0.642857	0
890	42.16	4	0	0	0.021348	0.047371	0.210526	0
...	...	...	...	...	...	...	...	...
783	59.10	2	2	0	0.012771	0.075479	0.200000	0
278	98.90	9	5	4	0.050360	0.355755	0.642857	1
300	5.41	6	6	4	0.030000	0.018033	0.666667	0
1524	33.97	2	5	3	0.005906	0.022290	0.222222	1
1456	56.97	11	7	2	0.019231	0.039128	0.392857	0

Gambar 23 Prediksi Decision Tree 1

```
## checking the balance of the data by plotting the count of outcomes by their value
color_wheel = {1: "#0392cf",
               2: "#7bc043"}
colors = data_hasil_from_csv["fraud"].map(lambda x: color_wheel.get(x + 1))
print(data_hasil_from_csv.fraud.value_counts())
p=data_hasil_from_csv.fraud.value_counts().plot(kind="bar")
```

```
0    427233
1     70888
Name: fraud, dtype: int64
```



Gambar 24 Prediksi Decision Tree 2

Pada penelitian yang menggunakan algoritma *Decision Tree* didapatkan bahwa prediksi untuk kelas *fraud* yaitu untuk variabel 0 atau *no fraud* sebesar 427.233 dan variabel 1 atau *fraud* sebesar 70.888.

3. *K-Nearest Neighbors*

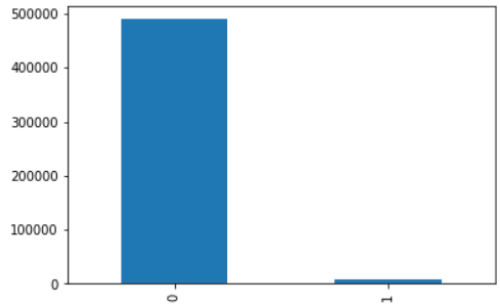
```
data_hasil_from_csv = pd.read_csv('HasilPrediksiKNN.csv')
data_hasil_from_csv.drop(columns=['Unnamed: 0'])
```

seconds	grandTotal	lineItemVoids	scansWithoutRegistration	quantityModifications	scannedLineItemsPerSecond	valuePerSecond	lineItemVoidsPerPosition	fraud
467	88.48	4	8	4	0.014989	0.189465	0.571429	0
1004	58.99	7	6	1	0.026892	0.058755	0.259259	0
162	14.00	4	5	4	0.006173	0.086420	4.000000	0
532	84.79	9	3	4	0.026316	0.159380	0.642857	0
890	42.16	4	0	0	0.021348	0.047371	0.210526	0
...	...	...	...	...	...	...	...	...
783	59.10	2	2	0	0.012771	0.075479	0.200000	0
278	98.90	9	5	4	0.050360	0.355755	0.642857	0
300	5.41	6	6	4	0.030000	0.018033	0.666667	0
1524	33.97	2	5	3	0.005906	0.022290	0.222222	0
1456	56.97	11	7	2	0.019231	0.039128	0.392857	0

Gambar 25 Prediksi K-Nearest Neighbors 1

```
## checking the balance of the data by plotting the count of outcomes by their value
color_wheel = {1: "#0392cf",
               2: "#7bc043"}
colors = data_hasil_from_csv["fraud"].map(lambda x: color_wheel.get(x + 1))
print(data_hasil_from_csv.fraud.value_counts())
p=data_hasil_from_csv.fraud.value_counts().plot(kind="bar")
```

```
0    489663
1      8458
Name: fraud, dtype: int64
```



Gambar 26 Prediksi K-Nearest Neighbors 2

Pada penelitian yang menggunakan algoritma *K-Nearest Neighbors* didapatkan bahwa prediksi untuk kelas *fraud* yaitu untuk variabel 0 atau *no fraud* sebesar 489.663 dan variabel 1 atau *fraud* sebesar 8.458

Daftar Pustaka

[1] C. C. Aggarwal and C. C. Aggarwal, *Data Classification*. 2015.

[2] E. Acuna, “International Encyclopedia of Statistical Science,” *Int. Encycl. Stat. Sci.*, no. January 2011, 2011, doi: 10.1007/978-3-642-04898-2.