INFORMATION SYSTEM DEVELOPMENT
VOLUME X NO.X (20XX) | p34-41
e-ISSN 2528-5114 | p-ISSN 2477-863X
http://dx.doi.org/10.19166/xxxx

KOMPARASI TINGKAT AKURASI RANDOM FOREST DAN KNN UNTUK MENDIAGNOSIS PENYAKIT KANKER PAYUDARA

Vincent Angkasa[1], Jefri Junifer Pangaribuan[2]

[1]Informatika, Fakultas Ilmu Komputer, Universitas Pelita Harapan
[2]Sistem Informasi, Fakultas Ilmu Komputer, Universitas Pelita Harapan
E-mail: [1]va80064@student.uph.edu, [2]jefri.pangaribuan@uph.edu

Abstract – Breast cancer is a cancer that is formed on breast cells. According to Observatory, breast cancer contributed 30.8% for death in all-ages woman that is caused by cancer in 2020. This research uses breast cancer data set to increase awareness and knowledge about breast cancer because the awareness of breast cancer should be public knowledge. KNN is often used for classification. Random Forest is versatile and can be used without tuning to give good result. Previous research indicates SVM has 96.47% of accuracy, Neural Network 97.06%, Naive Bayes 91.18%. The data set is from Kaggle. With the diagnosis of 'M' for malignant and 'B' for benign. The data set consists of 569 data and 33 columns in which 31 columns are used. Seventy-five percent of the data is the training data. This research concludes that KNN achieves a ROC score of 0.9959 while Random Forest produces 0.9951..

***Keywords**: Breast Cancer, KNN, Random Forest*

Abstrak – Kanker payudara adalah kanker yang terbentuk di sel-sel bagian payudara. Menurut data dari *Observatory* kanker payudara berkontribusi sebanyak 30,8% untuk kematian penyakit kanker pada wanita untuk semua usia pada tahun 2020. Penelitian ini memakai data set kanker payudara untuk menambah kesadaran, karena, kesadaran akan kanker payudara itu penting dan seharusnya menjadi ilmu pengetahuan umum. Algoritma KNN sering digunakan untuk kasus klasifikasi dan *Random Forest* memiliki sifat versatile dan tanpa di-tune dapat memberikan akurasi yang bagus dalam klasifikasi. Dari penelitian sebelumnya, SVM memiliki 96.47*% accuracy, Neural Network* sebanyak 97,06%, dan Naive Bayes 91,18% *accuracy*. Ppenelitian ini peneliti memiliki ketertarikan untuk membandingkan kedua algoritma dengan ROC curve. Sumber data berasal dari Kaggle. Diagnosis 'M' (malignant) dan 'B' (benign). Terdiri dari 569 data dan 33 kolom. Data *training* sebesar 75% dan memakai 31 kolom. Dari penelitian ini dapat disimpulkan nilai ROC yang dimiliki oleh KNN adalah sebesar 0.9959 dan *Random Forest* sebesar 0.9951.

Kata Kunci: Kanker Payudara, KNN, *Random Forest*

PENDAHULUAN]

Kanker payudara adalah kanker yang terbentuk di sel-sel bagian payudara. Kanker payudara merupakan kanker yang paling sering terjadi pada wanita dan penyebab utama kematian akibat kanker di kalangan wanita di seluruh dunia [1].

Data yang bersumber dari Observatory

menunjukkan kanker payudara berkontribusi sebanyak 30,8% untuk kematian penyakit kanker pada wanita untuk semua usia pada tahun 2020. Penyakit kanker banyak di-*record* agar pasien bisa dianalisis dan diantisipasi supaya pencegahan bisa dilakukan [2].

Dengan memakai data set kanker payudara untuk penelitian ini diharapkan bisa meningkatkan kesadaran dan juga menambah wawasan terhadap kanker payudara. Karena, kesadaran akan kanker payudara itu penting dan seharusnya menjadi ilmu pengetahuan umum [3].

Data mining adalah alat yang populer untuk melakukan analisis yang sudah dibuktikan di bidang obat, keuangan,

marketing dan social science. Contohnya, memakai teknik *machine learning* untuk melihat tingkah laku tumor pada pasien kanker payudara [4].

Machine learning adalah cabang dari kecerdasan buatan dan juga komputer sains yang memiliki fokus dalam pemakaian data dan juga algoritma-algoritma agar bisa meniru cara manusia belajar untuk meningkatkan accuracy machine learning [5].

Pemakaian mesin belajar ini untuk Kesehatan juga mengalami perkembangan di mana Google membuat algoritma untuk mesin belajar agar bisa mengidentifikasi tumor kanker dengan melihat *mammogram* [6].

Machine learning ini memerlukan algoritma agar bisa dipakai, dan untuk penelitian ini, peneliti akan memakai algoritma dengan *supervised learning. Supervised learning* cocok untuk dipakai karena masalah yang diatasi oleh *supervised learning* adalah klasifikasi dan regresi [7].

KNN atau *K-Nearest Neighbor* adalah contoh algoritma *supervised learning*. KNN merupakan algoritma dasar untuk *machine learning* yang sering dipakai untuk klasifikasi [8].

Algoritma *Random Forest*, yang memiliki kelebihan sebagai *versatile*, algoritma ini biasanya menghasilkan *accuracy* yang bagus tanpa di-*tune*, dengan *Random Forest, overfitting* bisa diatasi [9].

Dari penelitian sebelumnya, ada peneliti yang memakai SVM memiliki *accuracy* sebanyak 96,47%, *Neural Network* sebanyak 97,06%, dan memakai Naive Bayes dengan *accuracy* 91,18% [10]. Ada juga para peneliti yang meneliti untuk diagnosis kanker payudara dengan Naive Bayes dan menghasilkan akurasi sebesar 96,9% [11].

Peneliti lain juga ada melakukan penelitian diagnosis kanker payudara dengan Naive Bayes, *Decision Tree*, dan SVM, dengan akurasi 65,71%, 60%, dan 74,29% [12].

Oleh karena itu, untuk penelitian ini peneliti memiliki ketertarikan untuk memakai algoritma *Random Forest* dan juga KNN dan melakukan perbandingan antara kedua algoritma tersebut dengan melihat tingkat akurasi algoritma tersebut untuk kasus diagnosis kanker payudara serta ingin mengetahui apakah kedua algoritma tersebut bisa dipakai untuk melakukan prediksi diagnosis kanker payudara. Untuk *dataset* kanker payudaranya bersumber dari Kaggle.

METODOLOGI PENELITIAN

Penelitian ini memakai metode yang terdiri dari pengumpulan data, penyelesaian, melakukan prediksi dengan *Random Forest* dan KNN, serta melakukan analisis dengan ROC *curve*.

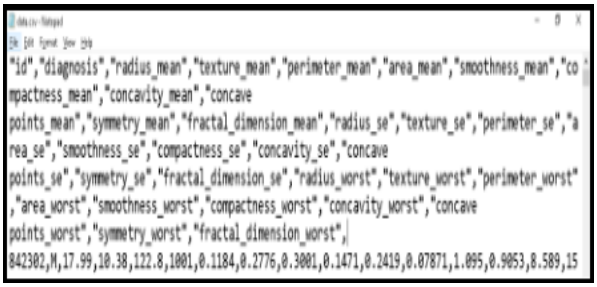
Metode Pengumpulan Data

Data set yang diteliti, data tersebut diperoleh dari *repository data set* yang ada di *website Kaggle* [13].

Kaggle adalah sebuah website yang dipakai oleh perusahaan-perusahaan untuk memberikan tantangan-tantangan di bidang *data science*. *Kaggle* memberikan *data set* dan *guidance* yang berisi tentang apa yang perlu dilakukan terhadap data tersebut [14].

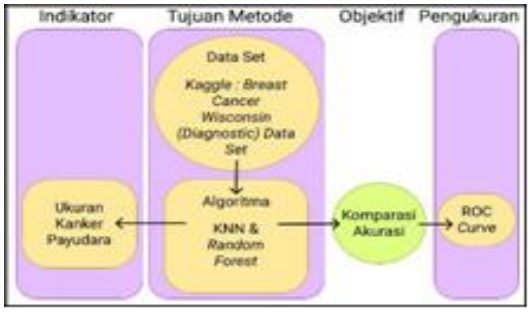
"*Breast Cancer Wisconsin (Diagnostic) Data Set*" adalah nama *data set* yang dipakai. *Data set* tersebut diperoleh dari *UCI Machine Learning Repository* dan ditaruh di *Kaggle*. Peneliti-peneliti *data mining* pernah memakai dan meneliti *data set* tersebut. *Data set* ini diakses serta di *download* melalui *platform Kaggle* pada tanggal 16 Juni 2021. Data set ini berupa *file* CSV.

File CSV tersebut di bagian awal berisikan *string*. Di mana *string* tersebut akan memberikan nama kolom. Dari gambar 1. bisa kelihatan *string value* yang ditaruh di dalam *file* CSV. Berdasarkan *file* CSV tersebut ada 32 kolom. Jika file CSV tersebut dibuka dengan *Microsoft Excel* memberikan kolom sejumlah 32 buah. Dan di *Kaggle* juga memberikan jumlah kolom sebanyak 32 kolom. Akan tetapi ketika di *import* ke *Python* menjadi *dataframe*, kolom tersebut menjadi 33 kolom. Hal ini dikarenakan di *file* CSV pada *string value* untuk kolom terakhi yaitu"*fractal_dimension_worst*", setelah *string value* tersebut muncul karakter koma. Yang membuat *Python* menganggap koma tersebut menandakan kolom yang baru tetapi tidak ada nama. Oleh sebab itu *Python* memberikan nama kolom tersebut menjadi "*Unnamed*: 32". *Unnamed* berarti tidak ada nama. Angka 32 datang dari indeks kolom yang dimulai dari 0.



Gambar 1. *String Value* di *File* CSV

Setelah data set di *import* maka *data set* kanker payudara ini memiliki 33 *attribute*. Akan tetapi *attribute* pertama adalah *ID number* dan *attribute* terakhir bernama "*Unnamed*: 32" yang tidak membantu dalam klasifikasi, sehingga *attribute* tersebut tidak akan dipakai yang menjadikan 31 *attribute* dipakai pada penelitian ini. *Attribute* untuk mengetahui ada kanker payudara atau tidak itu terletak di kolom *diagnosis* dengan *value* "B" yang artinya tidak ada kanker payudara.



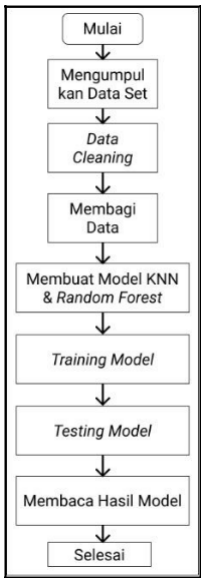
Gambar 2. Kerangka Pikir

Diterima <06122021> | Revisi <20122021> | Diterima untuk publikasi <28012022>

Jika *value diagnosis* adalah "M" memiliki arti ada kanker payudara. Berikut kerangka pikir (Gambar 2) pada penelitian ini.

Metode Penyelesaian

Model *machine learning* dengan algoritma KNN dan *Random Forest* dipakai dalam penelitian ini untuk menyelesaikan masalahnya. Berikut merupakan tahapan-tahapan pemakaian *machine learning* tersebut:



Gambar 3. *Flowchart* Penyelesaian Masalah

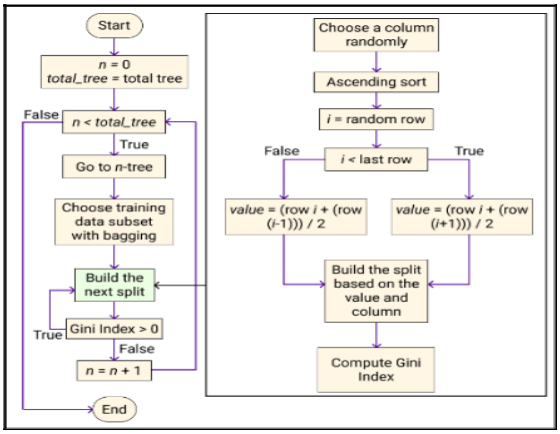
Gambar 3. memberikan tahapan-tahapan yang terjadi. Tahapan-tahapan tersebut menjadi metode penyelesaian pada penelitian ini. Yang diawali dengan mulai. Tahap kedua mencari data set. Data set ini didapatkan melalui *Kaggle*. Melakukan proses *data cleaning* dan membagi data menjadi *training data set* dan *testing data set*. Setelah proses tersebut berakhir maka selanjutnya membuat model KNN dan *Random Forest* untuk dilatih dan dicoba. Kemudian ROC *curve* dipakai untuk membaca hasil kedua model.

Melakukan Prediksi dengan *Random Forest*

Prediksi kanker payudara dilakukan dengan bantuan aplikasi *Google Colab* memanfaatkan *Python* dan *library scikit-learn*. Model *Random Forest* dilatih dengan data sebanyak 75% dan diuji dengan data sebanyak 25%.

Random Forest memiliki beberapa pohon, dan *attribute* dipilih secara *random* dengan memakai *bagging. Random Forest* memakai sekumpulan pohon di mana setiap pohon tersebut memiliki nilai vekor *arbitrary* yang di sampel secara terpisah untuk semua pohon di *Random Forest*. Fase algoritma *Random Forest* dibagi menjadi 3 fase, yang pertama adalah pohon sebanyak *i* dibuat. Fase kedua menggabungkan semua pohon menjadi *Random Forest*. Yang terakhir adalah memberikan *output*. *Output* dari *Random Forest* dihasilkan melalui hasil voting dari setiap pohon. Gambar 3 merupakan cara kerja algoritma *Random Forest* dalam bentuk *flowchart*.

Flowchart yang terdapat di gambar 3 dimulai dengan "*start*". Langkah selanjutnya memberikan *value* kepada *n* dan *total_tree*. Kedua *variable* tersebut berfungsi untuk mengatur banyaknya *looping* yang terjadi. Dengan kata lain for looping. Di sini cara kerja *for each looping* juga digunakan yang bisa kelihatan dari "*Go to n-tree*". Random Forest pada penelitian ini juga menggunakan metode *bagging* dalam membangun setiap pohon. Untuk *node* tiap *tree* diciptakan dengan memilih *random column* dari *bagging*. Setelah *column* dipilih, *column* tersebut dilakukan *ascending sorting*. *Random row* dipilih dan *row* di bawahnya dipilih untuk diambil *value* dari *row* tersebut dan dibagi 2. Jika *random row* yang dipilih merupakan *row* terakhir maka yang dipilih *row* kedua adalah *row* di atasnya. Atas kedua *value* tersebut maka *node* dibangun dan *split* bisa terjadi. Setelah *node* tersebut dibangun maka di cari *gini index*. Jika *gini index* sudah 0 maka *split* tersebut akan stop. Tetapi jika belum 0 atau pure maka *split* akan terus terjadi hingga menghasilkan *pure leaf*. Setelah itu jika sudah selesai looping untuk tiap tree maka algoritma Random Forest menjadi berhenti dan selesai. Hal ini diwakili pada gambar 4 dengan kata "*End*".



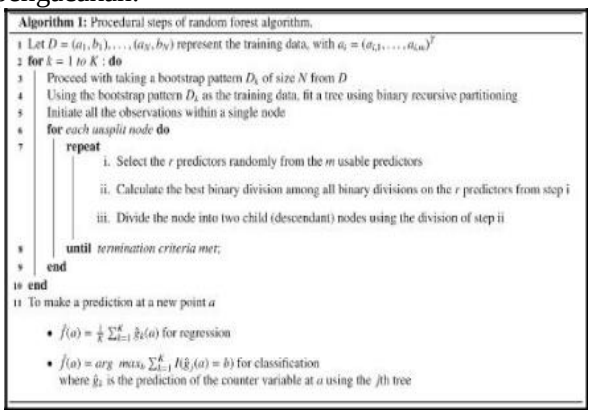
Gambar 4. *Flowchart* Algoritma *Random Forest*

Random Forest menggunakan pohon $g_k(A, \theta_k)$ di mana k^{th} sebagai *base learners*. adalah tumpukan variabel acak yang bersifat independen untuk $k = 1,..., K$. Untuk *training data* $D = (a_1, b_1), \ldots, (a_N, y_N)$, di mana $a_I = (a_I, 1,\ldots, a_I, p)^T$ mewakili *m predictor* dan b_1 sebagai respons dan sebuah realisasi spesifik θ_k dari Θ_k.

Diterima <06122021> | Revisi <20122021> | Diterima untuk publikasi <28012022>

Pohon yang sudah *fit* diwakili sebagai $\hat{g}k\,(a, \theta_k, D)$. Rumus tersebut dipakai untuk menghasilkan acak pada 2 fase. Pada fase pertama termasuk *bagging*, sebuah pola *bootstrap* independen yang diambil dari data *original* dan di-*fit* ke masing-masing pohon. Sampling *bootstrap* memanfaatkan fungsi acak yang memberikan satu bagian dari θ_k.

Pada tahap kedua, ketika pemilihan acak, *split* terbaik ditemukan dari *subset* variabel *r predictor* secara terpisah pada tiap-tiap *node* ketika melakukan *splitting*. *Predictor* pengambilan sampel memberikan bagian θ_k yang tersisa dengan pengacakan.


Algorithm 1: Procedural steps of random forest algorithm.

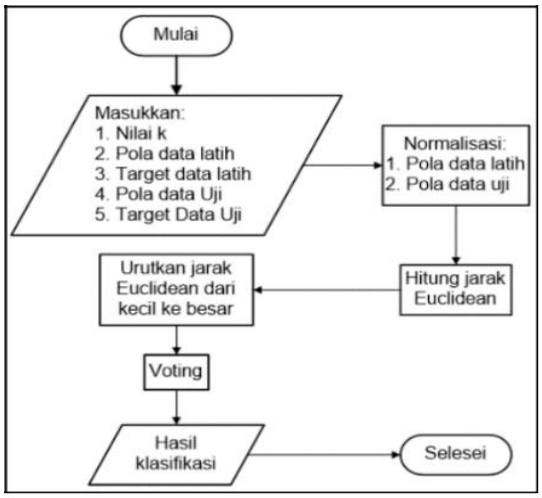
1 Let $D = (a_1, b_1), \ldots, (a_N, b_N)$ represent the training data, with $a_i = (a_{i,1}, \ldots, a_{i,M})^T$
2 **for** $k = 1$ *to* K : **do**
3 Proceed with taking a bootstrap pattern D_k of size N from D
4 Using the bootstrap pattern D_k as the training data, fit a tree using binary recursive partitioning
5 Initiate all the observations within a single node
6 **for** *each unsplit node* **do**
7 **repeat**
i. Select the r predictors randomly from the m usable predictors
ii. Calculate the best binary division among all binary divisions on the r predictors from step i
iii. Divide the node into two child (descendant) nodes using the division of step ii
8 **until** *termination criteria met;*
9 **end**
10 **end**
11 To make a prediction at a new point a
- $\hat{f}(a) = \frac{1}{K}\sum_{k=1}^{K} \hat{g}_k(a)$ for regression
- $\hat{f}(a) = arg\ max_b \sum_{k=1}^{K} I(\hat{g}_k(a) = b)$ for classification
where $\hat{g}_k$ is the prediction of the counter variable at a using the kth tree

Gambar 5. Pseudo-Code Random Forest [15]

Gambar 5 merupakan *pseudo-code* dari algoritma *Random Forest*. Algoritma ini memiliki beberapa pohon dan tiap pohon dibuat dengan langkah-langkah berikut [15]:

1. Asumsikan kasus *training* adalah *X* dan *classifier* tersebut terdiri dari jumlah variabel = *Y*.
2. Untuk penentuan keputusan pada *node* pohon, terdapat jumlah variabel *input y*, dan $y < Y$.
3. Pada pohon ini, pemilihan set pelatihan dilakukan dengan mengambil sampel *bootstrap*, yaitu memilih *x* kali dengan restorasi dari semua *X* kasus pelatihan yang tersedia. Estimasi kesalahan pohon dilakukan dengan menggunakan sisa kasus.
4. Secara acak, variabel y dipilih untuk setiap node pohon untuk melakukan keputusan pada node tersebut. Pemisahan terbaik dihitung oleh variabel y ini dalam set pelatihan.
5. Setiap pohon benar-benar dewasa dan tidak dipotong. Sampel baru diprediksi dengan memanfaatkan pohon tersebut. Label diberikan ke sampel pelatihan di akhir terminal node. Iterasi berlanjut untuk semua pohon, dan prediksi Random Forest dinyatakan dengan perhitungan suara rata-rata semua pohon.

Gambar 6. merupakan gambaran flowchart cara kerja algoritma KNN. Berikut adalah tahapan-tahapan algoritma KNN [16]:



Gambar 6. Flowchart Algoritma KNN [16]

Melakukan Prediksi dengan KNN.

Model KNN pada penelitian ini juga di bangun dengan *training data* sebanyak 75% dan *testing data* sebanyak 25%. KNN adalah salah satu metode klasifikasi dalam *data mining* dan termasuk bagian dari supervised learning. KNN menggunakan hasil yang datang dari query instance yang akan dikategorikan berdasarkan suara terbanyak dari kategori di KNN. Cara kerja KNN adalah dengan melakukan voting dari beberapa jawaban sesuai dengan nilai k objek.

1. Langkah pertama yaitu masukkan nilai k dengan minimal nilai 1 dan maksimalnya sebanyak jumlah data set yang di gunakan untuk latihan.
2. Langkah kedua melakukan normalisasi terhadap training data atau testing data supaya range nilai antara 0 hingga 1. Rumus normalisasi yang dipakai adalah rumus *min-max*. Persamaan min-max adalah:

$$Normalisasi = \frac{data_x - data_{min}}{data_{max} - data_{min}}$$

Keterangan:
$data_x$ = data yang bakal dicari nilai normalisasinya menurut kolom datanya
$data_{min}$ = nilai terkecil terletak pada kolom data yang akan dinormalisasi
$data_{max}$ = nilai terbesar dan 1 kolom dengan data yang ingin dinormalisasi

3. Langkah ketiga mencari jarak Euclidean. Persamaan untuk menghitung jarak Euclidean adalah:

$$d_{Euclidean}(x, y) = \sqrt{\sum_i (x_i - y_i)^2}$$

Keterangan:
i = banyak set data
x = jumlah *testing data*
y = jumlah *training data*

Diterima <06122021> | Revisi <20122021> | Diterima untuk publikasi <28012022>

4. Langkah keempat yaitu melaksanakan *voting* atau memilih panjang *euclidean* yang paling kecil yang berada di urutan nilai k.
5. Yang terakhir yaitu menentukan *output* klasifikasi dari langkah keempat yang terbanyak.

Metode Analisis ROC Curve

ROC *curve* (*receiver operating characteristic curve*) adalah sebuah graf yang menampilkan performa sebuah klasifikasi model pada semua *classification thresholds*. ROC *curve* menghasilkan graf dengan 2 parameter yaitu true positive rate (TPR) dan *false positive rate* (FPR). Rumus TPR dan FPR yaitu:

$$TPR = \frac{TP}{TP + FN}$$

$$FPR = \frac{FP}{FP + TN}$$

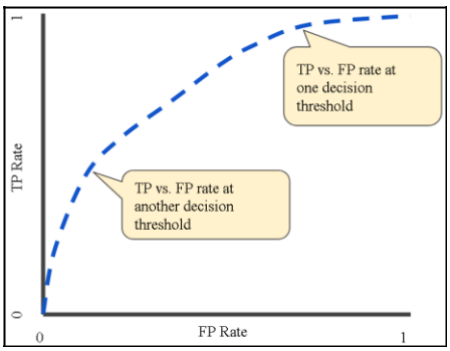
Keterangan [17]:

TP = *True positive* / jumlah kasus yang hasil prediksi positif dan memang positif FN = *False negative* / jumlah kasus yang hasil prediksi negatif tapi harusnya positif

FP = *False positive* / jumlah kasus yang hasil prediksi positif tapi harusnya negatif

TN = *True negative* / jumlah kasus yang hasil prediksi negatif dan memang negatif

ROC *curve* menggambarkan kurva TPR dengan FPR pada *classification thresholds* yang berbeda. Menurunkan *classification thresholds* menghasilkan klasifikasi positif yang lebih banyak, sehingga meningkatkan FP dan TP. Gambar 3.8 adalah contoh visualisasi ROC *curve* [18].



Gambar 7 Visualisasi ROC Curve [18]

HASIL DAN PEMBAHASAN

Hasil Pengumpulan Data

Setelah mengambil data dari *platform Kaggle*, maka didapatkan *data set* kanker payudara dengan sifat klasifikasi. *Data set* ini memiliki 569 pasien. *Data set* tersebut disediakan dalam bentuk *file* CSV (*Comma Separated Values*).

Gambar 4 merupakan informasi *data set* kanker payudara yang diperoleh dari *function data.info*(). Gambar 8. memiliki 4 kolom. Berikut penjelasan terhadap kolom di gambar 8:

```
data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 33 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   id                       569 non-null    int64
 1   diagnosis                569 non-null    object
 2   radius_mean              569 non-null    float64
 3   texture_mean             569 non-null    float64
 4   perimeter_mean           569 non-null    float64
 5   area_mean                569 non-null    float64
 6   smoothness_mean          569 non-null    float64
 7   compactness_mean         569 non-null    float64
 8   concavity_mean           569 non-null    float64
 9   concave points_mean      569 non-null    float64
 10  symmetry_mean            569 non-null    float64
 11  fractal_dimension_mean   569 non-null    float64
 12  radius_se                569 non-null    float64
 13  texture_se               569 non-null    float64
 14  perimeter_se             569 non-null    float64
 15  area_se                  569 non-null    float64
 16  smoothness_se            569 non-null    float64
 17  compactness_se           569 non-null    float64
 18  concavity_se             569 non-null    float64
 19  concave points_se        569 non-null    float64
 20  symmetry_se              569 non-null    float64
 21  fractal_dimension_se     569 non-null    float64
 22  radius_worst             569 non-null    float64
 23  texture_worst            569 non-null    float64
 24  perimeter_worst          569 non-null    float64
 25  area_worst               569 non-null    float64
 26  smoothness_worst         569 non-null    float64
 27  compactness_worst        569 non-null    float64
 28  concavity_worst          569 non-null    float64
 29  concave points_worst     569 non-null    float64
 30  symmetry_worst           569 non-null    float64
 31  fractal_dimension_worst  569 non-null    float64
 32  Unnamed: 32              0 non-null      float64
dtypes: float64(31), int64(1), object(1)
memory usage: 146.8+ KB
```

Gambar 8. *Data Info Breast Cancer*

	0	1	2	3	4
id	842302	842517	84300903	84348301	84358402
diagnosis	M	M	M	M	M
radius_mean	17.99	20.57	19.69	11.42	20.29
texture_mean	10.38	17.77	21.25	20.38	14.34
perimeter_mean	122.8	132.9	130	77.58	135.1
area_mean	1001	1326	1203	386.1	1297
smoothness_mean	0.1184	0.08474	0.1096	0.1425	0.1003
compactness_mean	0.2776	0.07864	0.1599	0.2839	0.1328
concavity_mean	0.3001	0.0869	0.1974	0.2414	0.198
concave points_mean	0.1471	0.07017	0.1279	0.1052	0.1043
symmetry_mean	0.2419	0.1812	0.2069	0.2597	0.1809
fractal_dimension_mean	0.07871	0.05667	0.05999	0.09744	0.05883
radius_se	1.095	0.5435	0.7456	0.4956	0.7572
texture_se	0.9053	0.7339	0.7869	1.156	0.7813
perimeter_se	8.589	3.398	4.585	3.445	5.438
area_se	153.4	74.08	94.03	27.23	94.44
smoothness_se	0.006399	0.005225	0.00615	0.00911	0.01149
compactness_se	0.04904	0.01308	0.04006	0.07458	0.02461
concavity_se	0.05373	0.0186	0.03832	0.05661	0.05688
concave points_se	0.01587	0.0134	0.02058	0.01867	0.01885
symmetry_se	0.03003	0.01389	0.0225	0.05963	0.01756
fractal_dimension_se	0.006193	0.003532	0.004571	0.009208	0.005115
radius_worst	25.38	24.99	23.57	14.91	22.54
texture_worst	17.33	23.41	25.53	26.5	16.67
perimeter_worst	184.6	158.8	152.5	98.87	152.2
area_worst	2019	1956	1709	567.7	1575
smoothness_worst	0.1622	0.1238	0.1444	0.2098	0.1374
compactness_worst	0.6656	0.1866	0.4245	0.8663	0.205
concavity_worst	0.7119	0.2416	0.4504	0.6869	0.4
concave points_worst	0.2654	0.186	0.243	0.2575	0.1625
symmetry_worst	0.4601	0.275	0.3613	0.6638	0.2364
fractal_dimension_worst	0.1189	0.08902	0.08758	0.173	0.07678
Unnamed: 32	NaN	NaN	NaN	NaN	NaN

Gambar 9. Isi Data Pertama Hingga Kelima

1. Di kolom pertama (Gambar 8) "#" adalah kolom untuk menyatakan nilai *index* pada *dataframe*. *Index* di *dataframe* ini dimulai dari 0 sehingga kolom pertama memiliki *index* 0.
2. Kolom kedua bernama "*Column*". "*Column*" ini berisikan nama kolom yang menjadi *attribute* pada *data set*.
3. Kolom ketiga adalah "*Non-Null Count*" di mana semua kolom kecuali kolom terakhir memiliki nilai yaitu 569 *non-null*. Dari kolom tersebut dapat disimpulkan bahwa sebanyak 569 data pada kolom tersebut memiliki nilai dengan kata lain tidak kosong. Karena ada data set yang memiliki nilai kosong seperti kolom terakhir kolom

Diterima <06122021> | Revisi <20122021> | Diterima untuk publikasi <28012022>

"*Unnamed*: 32". Kolom "*Unnamed*: 32" tidak memiliki nilai sama sekali makanya nilainya 0.

4. Kolom keempat adalah "*Dtype*" yang memberikan informasi tentang tipe data pada *dataframe*. *Int*64 memiliki arti integer. Integer dipakai untuk kolom "*id*" karena ID tidak memiliki nilai koma. Untuk *object* sendiri, memiliki arti agar data tersebut diperlakukan menjadi *string*. *Float*64 berarti kolom tersebut dapat memiliki nilai koma.

Gambar 9 adalah gambar yang menunjukkan 5 data pertama di *data set* kanker payudara. Untuk gambar 9, nama kolom terletak pada sebelah kiri dan yang atas menyatakan data pertama, kedua, ketiga, dan seterusnya.

Hasil Penelitian

Sebelum melatih model KNN dan *Random Forest* dengan *Scikit-learn*, *data set* kanker payudara perlu dibersihkan terlebih dahulu, supaya *data set* bisa dipakai tanpa *error* ketika *coding* dan juga membuang *attribute* yang tidak berguna dalam

***Data Cleaning* dan Pembagian Data**

1. Data set ini memiliki 33 attributes dan bisa terlihat dari gambar 4. Untuk data cleaning hanya perlu menghapus kolom "id" dan "Unnamed: 32". Serta melakukan transformasi data kolom diagnosis. Gambar 4 menunjukkan kolom data setelah data cleaning Setelah data cleaning, data set tersebut akan dilakukan split function dari Scikit-learn. train_test_split adalah function untuk melakukan split. Data set akan di bagi menjadi training data sebesar 75% dan test data sebesar 25%, dengan random state sebesar 42. Random state dipakai supaya pembagian datanya akan tetap sama sehingga, training data dan test data menjadi konsisten jika function tersebut dijalankan lagi.
2. Membangun Model KNN dan Random Forest. Model KNN dan Random Forest dibangun dengan library Scikit-*learn*. Karena *Random Forest* memanfaatkan *random*, agar hasilnya tetap konsisten, *function* tersebut diberi *random state* sebesar. Setelah kedua model dibangun, model tersebut diberi *data training* dan *data testing* dipakai untuk melihat akurasi kedua model tersebut.
3. Implementasi di *Google Colab* Pemakaian *platform Google Colab* untuk penelitian ini memiliki tujuan untuk membangun model *machine learning* lalu model tersebut melakukan klasifikasi. Berikut tahapan memakai *Google Colab*:

    1. Melakukan *import* untuk *library* yang diperlukan. Pada penelitian ini *library Scikit-learn*, *Pandas*, dan *Matplotlib* dipakai untuk keperluan *machine learning*, *data set*, dan *graph*. Ketika melakukan *import library*, walaupun *library* tersebut tidak terpakai, tidak masalah. Gambar 11. merupakan cara implementasi *import library* pada *Google Colab*.

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import sklearn
import json
import keras
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense
from keras.models import Sequential
from keras.layers import Dense, Dropout
from sklearn.model_selection import train_test_split
from sklearn.base import clone
from sklearn import tree, metrics
from sklearn.preprocessing import OneHotEncoder
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
from sklearn.ensemble import (RandomForestClassifier, GradientBoostingClassifier)
from sklearn.datasets import make_classification
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn import preprocessing
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import recall_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import f1_score
from sklearn.metrics import precision_score
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn import datasets
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_curve, roc_auc_score
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
```

Gambar 11. Cara Implementasi *Import Library*

    2. Menaruh *data set* ke dalam *dataframe* dengan *library Pandas*. Gambar 12. adalah*code* yang dipakai untuk pindah *data set* ke *dataframe*.

```
data = pd.read_csv(link)
df = data.copy()
```

Gambar 12. *Code Data Set* ke *Dataframe*

    3. Melakukan *data cleaning*. Penelitian ini melakukan penghapusan kolom dan transformasi kolom untuk proses *data cleaning*. Gambar 13 adalah *code* untuk melakukan penghapusan kolom dan transformasi kolom diagnosis.

```
del df['Unnamed: 32']
del df['id']

#--------------


le = preprocessing.LabelEncoder()
df['diagnosis'] = le.fit_transform(df.diagnosis.values)
```

Gambar 13. Penghapusan Kolom dan Transformasi

    4. Membagi *dataframe* menjadi training data set dan testing data set. Gambar 14. menunjukkan cara untuk memakai function train_test_split dari library Scikit-

```
X_train, X_test, y_train, y_test = train_test_split(
                    df.iloc[:, 1:], df['diagnosis'],
                    test_size=0.25, random_state=42)
```

Gambar 14. Implementasi *train_test_sp*

learn untuk membagi data. Yang berisi parameter ukuran testing data sebesar 25%, *random state* 42, dan *index data frame* untuk dibagi.

Diterima <06122021> | Revisi <20122021> | Diterima untuk publikasi <28012022>

5. Membangun model *machine learning* dengan Scikit-learn. Gambar 15. ialah cara untuk membuat model *Random Forest* dan KNN. Setelah dibuat model tersebut dilatih dengan *training data*.

```
randomforest = RandomForestClassifier(random_state=0)
randomforest.fit(X_train, y_train)

#y_pred = randomforest.predict(X_test)
#accuracy_score(y_test, y_pred)

#knn
knn = KNeighborsClassifier()
knn.fit(X_train, y_train)
```

Gambar 15. Membangun Model di *Google Colab*

6. Setelah dilatih, model siap dipakai untuk di tes dengan *testing data*. Untuk melakukan prediksi tinggal memakai *function .predict*(). Karena penelitian ini menggunakan ROC *curve*, maka hasil akurasi diagnosis akan dinilai dengan nilai ROC *curve*. Gambar 16 memberikan cara implementasi *code ROC curve* di *Google Colab*.

```
y_pred1 = randomforest.predict_proba(X_test)[:,1]
y_pred2 = knn.predict_proba(X_test)[:,1]
false_positive_rate1, true_positive_rate1, threshold1 = roc_curve(y_test, y_pred1)
false_positive_rate2, true_positive_rate2, threshold2 = roc_curve(y_test, y_pred2)

auc1 = roc_auc_score(y_test, y_pred1)
auc2 = roc_auc_score(y_test, y_pred2)
print('roc_auc_score for Random Forest: ', roc_auc_score(y_test, y_pred1))
print('roc_auc_score for KNN: ', roc_auc_score(y_test, y_pred2))
```

Gambar 16. *Implementasi* ROC *Curve* di Google Colab

7. Hasil ROC *curve* bisa juga untuk ditampilkan dalam bentuk *graph*. Dari gambar 17. bisa terlihat *code* yang dipakai untuk menghasilkan *graph* dari ROC *curve*.

```
plt.subplots(1, figsize=(10,10))
plt.title('ROC Curve Random Forest & KNN')
plt.plot(false_positive_rate1, true_positive_rate1, label='Random Forest, auc='+str(auc1))
plt.plot(false_positive_rate2, true_positive_rate2, label='KNN, auc='+str(auc2))
#plt.plot(1,label="data 1, auc="+str(auc1))
plt.plot([0, 1], ls="--")
#plt.plot([0, 0], [1, 0] , c=".7"), plt.plot([1, 1] , c=".7")
#plt.plot(false_positive_rate1,true_positive_rate1,label="data 1, auc="+str(auc1))
#plt.plot(false_positive_rate2,true_positive_rate2,label="data 2, auc="+str(auc2))
plt.legend(loc=4)
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```

Gambar 17. Implementasi *Graph* ROC *Curve*

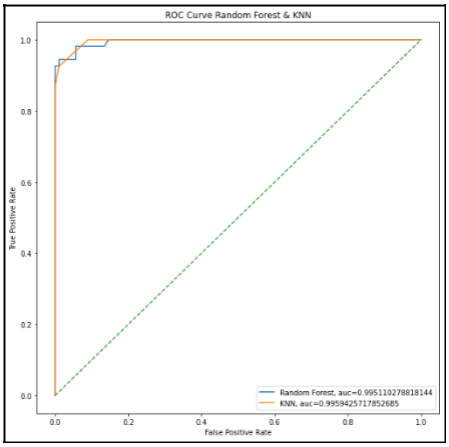
8. Gambar 18. memberikan *code* untuk melihat hasil akurasi (*accuracy score*)dari kedua model tersebut

```
y_pred = randomforest.predict(X_test)
print("Accuracy Random Forest:\n", accuracy_score(y_test, y_pred))
y_pred = knn.predict(X_test)
print("Accuracy KNN:\n", accuracy_score(y_test, y_pred))
```

Gambar 18. Implementasi *Accuracy Score*

Pembahasan

Berikut merupakan hasil graf ROC *curve* model KNN dan *Random Forest*. Gambar 19. menunjukkan graf ROC *curve* dari model KNN dan *Random Forest*. Di mana garis yang berwarna biru adalah milik *Random Forest* dan warna *orange* dari KNN. Untuk nilai ROC sendiri memiliki nilai sebesar 0.9951 untuk Random Forest. Untuk KNN memiliki nilai ROC sebesar 0.9959. Dari kedua nilai tersebut dapat disimpulkan bahwa pada kasus ini, untuk melakukan klasifikasi pada *data set* kanker payudara, yang unggul adalah model KNN. Dari grafnya juga dapat disimpulkan bahwa memang KNN lebih unggul jika dibandingkan dengan *Random Forest* pada kasus ini.



Gambar 19. ROC Curve KNN dan Random Forest

Garis *orange* tersebut lebih cepat mencapai titik tertinggi dibandingkan garis biru. Yang berarti KNN memiliki *False Positive Rate* (FPR) yang rendah memberikan dampak semakin sedikit kesalahan ketika klasifikasi. Misalnya, semakin rendah FPR, semakin sedikit pula pasien yang didiagnosis kena kanker payudara, tapi ternyata tidak kena Serta untuk akurasi dari KNN dan *Random Forest*, akurasi dari kedua model tersebut sebesar 96.5% dan 97.2% yang di tes dengan data set testing. Dengan kata lain, ketika KNN melakukan klasifikasi dari 100 kasus ada 96 kasus yang benar dan 4 kasus yang salah. Dan untuk *Random Forest* dari 100 kasus ada 97 kasus yang benar dan 3 kasus yang salah.

KESIMPULAN

Kesimpulan yang dapat diambil setelah melakukan penelitian ini dan menganalisis hasil yang diperoleh sebagai berikut:

1. Algoritma yang cara kerjanya sederhana belum tentu akan kalah dengan algoritma yang memiliki proses yang lebih kompleks, hal ini terbukti dari hasil nilai ROC yang dimiliki oleh KNN sebesar 0.9959 dan *Random Forest* sebesar 0.9951. Di mana nilai ROC memiliki maksimum nilai sebesar 1 dan nilai minimum sebesar 0. Sehingga, semakin nilai ROC menuju nilai 1 maka akurasi diagnosis menjadi lebih akurat.
2. Pada kasus ini, *machine learning* algoritma KNN dan *Random Forest* dengan *library Scikit-learn* yang dilatih dengan *Breast Cancer Wisconsin* (*Diagnostic*) *Data Set*, KNN memiliki *False Positive Rate* yang lebih rendah dari *Random Forest*. Hal ini bisa dibuktikan melalui graf ROC *curve* untuk kedua model tersebut.
3. Nilai ROC yang dihasilkan oleh kedua *machine learning* tersebut memiliki nilai yang sangat tinggi pada kasus ini sehingga, *library Scikit-learn* terbukti dapat dimanfaatkan untuk membuat model *machine learning* dengan *tuning*

Diterima <06122021> | Revisi <20122021> | Diterima untuk publikasi <28012022>

yang minimum untuk kasus prediksi klasifikasi lainnya.

4. Implementasi *code* untuk membangun model *machine learning* dengan *library Scikit-learn* memiliki *code* yang pendek dan mudah untuk dipahami.
5. Dari hasil graf ROC *curve* dapat disimpulkan juga bahwasannya algoritma *Random Forest struggling* (susah) untuk mencapai *True Positive Rate* sebesar 1 dibandingkan dengan algoritma KNN yang memiliki garis ROC *curve* yang mulus.
6. *Starting point* yang dihasilkan dari ROC *curve* untuk algoritma *Random Forest* yang lebih tinggi dari KNN, tidak menjamin bahwa kalibrasi *Random Forest* berikutnya menghasilkan *True Positive Rate* dan *False Positive Rate* lebih bagus dari KNN.

DAFTAR PUSTAKA

[1] M. M. Pavani Chalasani, "Breast Cancer: Practice Essentials, Background, Anatomy," Oct. 16, 2021. https://emedicine.medscape.com /article/1947145-overview (accessed Nov. 30, 2021).

[2] M. Faizal Kurniawan,"Komparasi Algoritma Data Mining Untuk Klasifikasi Penyakit Kanker Payudara," 2017. [Online]. Available: http://jurnal.stmik-wp.ac.id.

[3] admin, "Why Breast Cancer Awareness Is Important - PGOMG," Oct. 14, 2019. https://www.pacgyn.com/breast-cancer-awareness-important/ (accessed Nov. 30, 2021).

[4] S. A. Mohammed, S. Darrab, S. A. Noaman, and G. Saake, "Analysis of breast cancer detection using different machine learning techniques," in Communications in Computer and Information Science , 2020, vol. 1234 CCIS, pp. 108–117, doi: 10.1007/978-981-15-7205-0_10.

[5] IBM Cloud Education, "What is Machine Learning? | IBM," Jul. 15, 2020. https://www.ibm.com/cloud/learn/machine-learning (accessed Nov. 30, 2021).

[6] M. Ed Corbett, "The Real-World Benefits of Machine Learning in Healthcare," Apr. 25, 2017. https://www.healthcatalyst.com/clinical-applications-of- machine-learning-in-healthcare (accessed Feb. 10, 2021).

[7] Jason Brownlee, "A Tour of Machine Learning Algorithms," Aug. 12, 2019. https://machinelearningmastery.com/a-tour-of-machine-learning-algorithms/ (accessed Nov. 30, 2021).

[8] Aditya Kumar, "KNN Algorithm: When? Why? How? KNN: K Nearest Neighbour is one of the by Aditya Kumar; Towards Data Science," May 25, 2020. https://towardsdatascience.com/knn-algorithm-what-when-why-how-41405c16c36f (accessed Nov. 30, 2021).

[9] Niklas Donges, "Random ForestAlgorithms: A Complete Guide Built In," Sep. 16, 2021. https://builtin.com/data-Science/random-forest-algorithm (accessed Nov. 30, 2021).

[10] D. A. Omondiagbe, S. Veeramani, and A. S. Sidhu, "Machine Learning Classification Techniques for Breast Cancer Diagnosis," in IOP Conference Series: Materials Science and Engineering , 2019, vol. 495, no. 1, doi: 10.1088/1757-899X/495/1/012033.

[11] H. Oktavianto and R. P. Handri, "Analisis Klasifikasi Kanker Payudara Menggunakan Algoritma Naive Bayes," 2019. [Online]. Available: https://archive.ics.uci.edu/ml/.

[12] L. Indah Prahartiwi, W. Dari, and S. Nusa Mandiri, "Komparasi Algoritma Naive Bayes, Decision Tree dan Support Vector Machine untuk Prediksi Penyakit Kanker Payudara," J. Tek. Komput. AMIK BSI , vol. 7, no. 1, 2021, doi: 10.31294/jtk.v4i2.

[13] UCI Machine Learning, "Breast Cancer Wisconsin (Diagnostic) Data Set | Kaggle," Sep. 25, 2016. https://www.kaggle.com/uciml/breast-cancer-wisconsin-data (accessed Nov. 30, 2021).

[14] Eleanor Bennett, "What Is Kaggle? | Logit.io," Oct. 07, 2021. https://logit.io/blog/post/what-is-kaggle (accessed Nov. 30, 2021).

[15] N. Kunhare, R. Tiwari, and J. Dhar, "Particle swarm optimization and feature selection for intrusion detection system," Sadhana - Acad. Proc. Eng. Sci. , vol. 45, no. 1, Dec. 2020, doi: 10.1007/s12046-020-1308-5.

[16] Abdul Muiz Khalimi, "Tahapan Dasar Algoritma kNN (k-Nearest Neighbor) – Sistemku Infomasimu," Jun. 23, 2020. https://www.pengalaman-edukasi.com/2020/06/tahapan- dan-cara-kerja-yang-benar.html (accessed Nov. 30, 2021).

[17] S. K. . M. K. Dr. Maria Susan Anggreany, "Confusion Matrix," Nov. 01, 2020. https://socs.binus.ac.id/2020/11/01/confusion-matrix/ (accessed Nov. 30, 2021).

[18] Google, "Classification: ROC Curve and AUC | Machine Learning Crash Course," Feb. 10, 2020. https://developers.google.com/machine-learning/crash- course/classification/roc-and-auc (accessed Nov. 30, 2021).

Diterima <06122021> | Revisi <20122021> | Diterima untuk publikasi <28012022>