e-ISSN 2962-6129
Seminar Nasional Mahasiswa Ilmu Komputer dan Aplikasinya (SENAMIKA)
Jakarta-Indonesia, 20 Agustus 2022

Klasifikasi Diagnosis Penyakit *Stroke* Dengan Menggunakan Metode *Random Forest*

Nur Aliffiyanti Iskandar [1], Iin Ernawati [2], Yuni Widiastiwi[3]
Program Studi S1 Informatika / Fakultas Ilmu Komputer
Universitas Pembangunan Nasional Veteran Jakarta
Jl. RS. Fatmawati Raya, Pd. Labu, Kec. Cilandak, Kota Depok, Jawa Barat 12450
nurai@upnvj.ac.id[1], iin_ernawati@yahoo.com[2], widiastiwi@yahoo.com[3]

Abstrak. Penyakit *stroke* menyebabkan kematian kedua dan kecacatan ketiga di dunia, dimana 70% penderita penyakit *stroke* terjadi pada negara berpenghasilan rendah dan menengah. Sementara itu, kematian dan kecacatan yang disebabkan oleh penyakit *stroke* menyumbang 87%. Penyakit *stroke* dan TIA *(Transient Ischemic Attack)* termasuk ke dalam kasus emergensi. Namun gejala dini penyakit *stroke* sulit untuk diketahui. Data mining dapat dimanfaatkan untuk mendiagnosis penyakit. Tujuan penelitian adalah mendapatkan model terbaik dengan *Random Forest* dari penggunaan beberapa pohon berbeda untuk penyakit *stroke*. Model dengan penggunaan jumlah pohon 90 menghasilkan nilai yang optimal, dimana nilai *accuracy* yang dihasilkan sebesar 95.2%, *sensitivity* sebesar 4.1%, *specificity* sebesar 99.8%, *precision* sebesar 66.7%, dan *F-measure* sebesar 7.6%. *Serta ROC Curve* sebesar 0.8048 yang menandakan bahwa model termasuk ke dalam *Good Classification.*

Kata Kunci: Klasifikasi, *Random Forest, Stroke*

1 Pendahuluan

Penyakit *stroke* termasuk salah satu penyakit tidak menular (PTM) yang dapat dialami secara tiba-tiba dikarenakan adanya gangguan dalam mengantarkan darah ke otak selama 24 jam atau lebih dan di negara maju maupun negara berkembang, penyakit *stroke* menjadi masalah kesehatan yang cukup serius[1]. Penyakit *stroke* menyebabkan kematian kedua dan kecacatan ketiga di dunia, dimana 70% penderita penyakit *stroke* terjadi pada negara berpenghasilan rendah dan menengah. 87% kematian dan kecacatan disebabkan oleh penyakit *stroke*[2].

Penyakit *stroke* dan TIA *(Transient Ischemic Attack*) merupakan kasus emergensi, yang mana jika tidak diberikan penanganan sesegera mungkin dapat menimbulkan dampak yang lebih parah, tetapi gejala dini dari penyakit *stroke* sulit untuk diketahui[3]. Penderita juga tidak menyadari atau merasakan adanya gejala dini penyakit *stroke*[4]. Sehingga ketika penderita telah mengalami gejala yang cukup serius, barulah si penderita dievaluasi ke rumah sakit untuk diberikan penanganan yang lebih baik. Padahal jika gejala dini dari penderita sudah diketahui, dampak buruk yang ditimbulkan oleh penyakit *stroke* dapat diminimalisir.

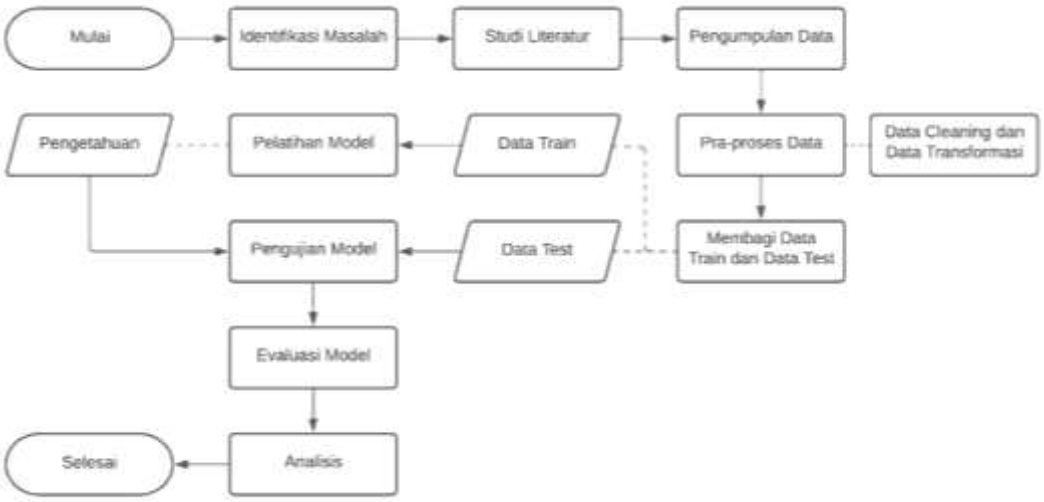
Data mining dapat dimanfaatkan untuk mendiagnosis penyakit[5]. Dari metode klasifikasi untuk prediksi, *Random Forest* menjadi metode yang menghasilkan nilai akurasi yang baik[6]. Penelitian terkait melakukan klasifikasi untuk memprediksi penyakit jantung dengan metode *Random Forest*, yang memperoleh nilai akurasi sebesar 85.3%[5]. Sedangkan penelitian kedua melakukan klasifikasi untuk memprediksi pasien menderita penyakit ginjal kronik dengan menggunakan metode *Naïve Bayes*, *Naïve Bayes* dengan *Adaboost*, dan *Random Forest* yang menghasilkan nilai akurasi 95.4% untuk *Naïve Bayes*, *Naïve Bayes* dengan *Adaboost* menghasilkan nilai akurasi 98.6%, dan *Random Forest* menghasilkan nilai akurasi 99.3%[7].

Oleh karena itu, penelitian ini akan menggunakan metode *Random Forest* untuk klasifikasi diagnosis penyakit *stroke*. Tujuan penelitian adalah mengetahui performa yang dihasilkan oleh metode *Random Forest* melalui tahapan evaluasi dengan menggunakan beberapa pohon yang berbeda. Luaran akhir yang akan diperoleh berupa model terbaik untuk mendiagnosis penyakit

stroke dengan menggunakan *Random Forest.*

2 Metodologi Penelitian

Di bawah ini adalah skema alur penelitian yang dilakukan.



Gambar. 1. Alur penelitian terdiri dari beberapa tahapan, yang dimulai dari tahapan identifikasi masalah hingga tahapan analisis.

2.1 Identifikasi Masalah

Tahapan pertama yang dilakukan dalam melakukan penelitian adalah mengidentifikasi masalah atas topik yang dipilih sebelumnya. Identifikasi masalah ini berguna untuk mengetahui rincian masalah yang akan dilakukan dan diselesaikan melalui prosedur atau tahapan penelitian.

2.2 Studi Literatur

Kemudian melakukan studi literatur dengan mencari bahan referensi melalui artikel, jurnal, buku, atau e-Book yang bersumber dari internet dan perpustakaan. Bahan referensi yang digunakan meliputi teori *Stroke* dan metode *Random Forest.*

2.3 Pengumpulan Data

Dataset stroke didapatkan dari *website* Kaggle (https://www.kaggle.com/fedesoriano/stroke-prediction-dataset), yang terdiri dari 11 atribut dan 1 *class*. *Dataset stroke* ini memiliki 5110 *records*. Berikut ini adalah deskripsi dari atribut *dataset stroke*.

Tabel 1. Deskripsi Atribut *Dataset Stroke*

.	Atribut	Deskripsi	Tipe Data
		al unik	Numerik
		elamin	Kategorik
			Numerik
	nsion	n tidak hipertensi n dengan hipertensi	Numerik
	Disease	n tidak memiliki penyakit jantung n memiliki penyakit jantung	Numerik
	Married	au “Tidak”	Kategorik

	ype	anak", "Pekerjaan Pemerintah", "Tidak Bekerja", "Swasta", atau "Wiraswasta"	Kategorik
	ce Type	an" atau "Perkotaan"	Kategorik
	ucose Level	rata-rata glukosa pada darah	Numerik
.		massa tubuh	Numerik
.	*g_Status*	mnya Merokok", "Tidak Pernah Merokok", "Merokok", atau "Tidak Diketahui"	Kategorik
.		n tidak *stroke* n mengalami *stroke*	Numerik

2.4 Pra-proses Data

Dari 12 atribut, terdapat satu atribut yang tidak digunakan dalam pengolahan yaitu atribut *id* sehingga pada penelitian ini hanya menggunakan 11 atribut saja. Dalam tahap pra-proses data, dilakukan *data cleaning* dan data transformasi. Pada *data cleaning* membersihkan data dari *missing value* dan *outlier*. Pada penelitian ini penanganan terhadap *missing value* data numerik adalah dengan memasukkan nilai tengah *(median)* dan untuk *missing value* data kategorik adalah dengan memasukkan nilai sebelum dari *missing value*[8]. Sedangkan penanganan untuk data *outlier* adalah dengan metode IQR *(Interquartile Range)* yaitu pembagian data dari Q1 (kuartil pertama), Q2 *(median)*, hingga Q3 (kuartil ketiga)[8]. Data transformasi dilakukan dengan merubah tipe data ke dalam format yang sesuai untuk memudahkan dalam melakukan klasifikasi. Data dengan tipe kategorik akan diubah menjadi tipe numerik yaitu 0 atau 1.

2.5 Membagi Data *Train* dan Data *Test*

Pembagian *dataset* menggunakan *splitting* data 80/20[9]. Dimana 80% untuk data *train* dalam melatih model dan 20% untuk data *test* dalam menguji model.

2.6 Pelatihan Model

Metode *Random Forest* adalah metode berbasis *Decision Tree* (pohon keputusan), dimana saat pelatihan *Random Forest* akan membuat beberapa pohon keputusan sehingga dari sampel-sampel set pelatihan tersebut akan menghasilkan sejumlah pohon[10]. Untuk membangun *Random Forest* diperlukan untuk mencari nilai *Gini Index* yang berfungsi dalam menentukan split untuk dijadikan *root/node*[10].

$$Gini\ (K) = 1 - \sum_{i=1}^{n} P_i^2 \quad \textbf{(1)}$$

Keterangan:
n = Jumlah kelas
P_i^2 = Kemungkinan pengamatan pada K termasuk suatu kelas

Gini Index mengasumsikan *binary split* pada keseluruhan atribut dalam S, seperti T1 dan T2. *Gini Index* dari K bisa dihitung dengan rumus[10]:

$$Gini_s\ (K) = \frac{T_1}{T} Gini\ (T_1) - \frac{T_2}{T} Gini\ (T_2) \quad \textbf{(2)}$$

Keterangan:
T_1, T_2 = Jumlah record pada kelas 1, 2, …
T = Jumlah record pada node p

Sedangkan pengurangan terhadap ketidakmurnian dihitung dengan rumus[10]:

$$Gini\ (K) - Gini_s\ (K) \quad \textbf{(3)}$$

Langkah-langkah yang dilakukan dalam pelatihan model dengan menggunakan metode *Random Forest*, sebagai berikut:

a. Membaca data *train* yang didapatkan dari proses pembagian data.

b. Jumlah pohon *(tree)* yang digunakan adalah 10, 20, 30, 40, 50, 60, 70, 80, 90, dan 100. Tujuannya adalah untuk mendapatkan hasil yang optimal.
c. Data *train* diambil secara acak untuk dijadikan sampel data. Selanjutnya data tersebut diterapkan metode *Random Forest* dengan melakukan percobaan dengan jumlah pohon yang sudah ditentukan sebelumnya.
d. Pelatihan data *train* dilakukan dengan membandingkan keseluruhan kolom dengan kolom target.

2.7 Pengujian Model

Pada tahap pengujian model menggunakan data *test*. Tahapan ini memprediksikan nilai target pada tiap-tiap baris dari data *test*. Klasifikasi terhadap data *test* dilakukan dengan cara memasukkan dan membandingkan nilai pada kolom terhadap pohon-pohon yang telah terbentuk sehingga menghasilkan keputusan akhir dari nilai yang paling banyak muncul *(majority voting).*

2.8 Evaluasi Model

Tahap selanjutnya adalah mengevaluasi kinerja dari model yang dihasilkan untuk mendapatkan dan mengetahui performa dari model tersebut. Evaluasi yang digunakan adalah *Confusion Matrix* dan ROC. Berikut ini adalah ilustrasi dari *Confusion Matrix*.

		Kelas hasil prediksi		
		Ya	Tidak	Jumlah
Kelas aktual	Ya	*TP*	*FN*	*P*
	Tidak	*FP*	*TN*	*N*
	Jumlah	*P'*	*N'*	*P + N*

Gambar. 2. *Confusion Matrix* terdiri dari dua label kelas, label pertama adalah kelas aktual atau kelas yang ada pada data *test* dan label kedua adalah kelas hasil prediksi atau kelas yang dihasilkan oleh model klasifikasi. TP *(True Positive)* adalah total tuple aktual yang berlabel positif yang berhasil dilabeli secara benar oleh model. TN *(True Negative)* adalah total tuple aktual yang berlabel negatif yang berhasil dilabeli secara benar oleh model. FP *(False Positive)* adalah total tupel aktual yang berlabel negatif, tetapi dilabeli kelas positif oleh model. FN *(False Negative)* adalah total tupel aktual yang berlabel positif, tetapi dilabeli kelas negatif oleh model [11].

Pada penelitian ini ukuran ketepatan yang digunakan yaitu:
a. *Accuracy*, persentase dari keseluruhan tuple data *test* yang berhasil diklasifikasikan secara benar oleh model klasifikasi[11].

$$Accuracy = (TP + TN) / (P + N) . \quad \textbf{(1)}$$

b. *Sensitivity,* tingkat pengenalan pada tuple positif dimana bagian tuple positif diklasifikasikan secara benar[11].

$$Sensitivity = (TP / P) . \quad (\textbf{2})$$

c. *Specificity,* tingkat pengenalan pada tuple negatif dimana bagian tuple negatif diklasifikasikan secara benar[11].

$$Specificity = (TN / N) . \quad (\textbf{3})$$

d. *Precision,* berapa persentase tuple yang dilabel positif yang kenyataannya benar positif[11].

$$Precision = (TP / (TP + FP)) . \quad \textbf{(4)}$$

e. *F-measure*, rata-rata harmonik dari *precision* dan *recall* yang memberikan masing-masing bobot yang sama pada *precision* dan *recall*[11].

$$F\text{-}measure = ((2 \times precision \times recall) / (precision + recall)) . \quad \textbf{(5)}$$

ROC *Curve* digunakan untuk merepresentasikan *Confusion Matrix* sehingga hasil perhitungan yang dihasilkan dapat dibuat grafiknya atau visualisasinya[12]. ROC *Curve* memiliki beberapa tingkatan dalam menilai diagnosa yang dihasilkan, diantaranya:

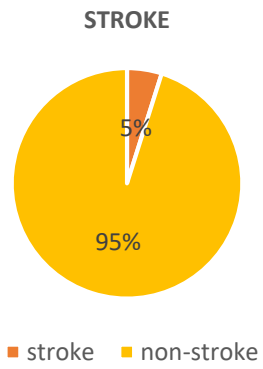
a. Nilai akurasi sebesar 0.90 – 1.00 menandakan *Excellent Classification*.
b. Nilai akurasi sebesar 0.80 – 0.90 menandakan *Good Classification.*
c. Nilai akurasi sebesar 0.70 – 0.80 menandakan *Fair Classification.*
d. Nilai akurasi sebesar 0.60 – 0.70 menandakan *Poor Classification.*
e. Nilai akurasi sebesar 0.50 – 0.60 menandakan *Failure*.

2.9 Analisis

Analisis ini melakukan perbandingan terhadap nilai *accuracy*, *sensitivity*, *specificity*, *precision*, dan *F-measure* dari masing-masing model yang dihasilkan dengan penggunaan jumlah pohon yang berbeda. Analisis dilakukan untuk mengetahui penggunaan jumlah pohon manakah yang menghasilkan nilai optimal. Lalu mencari nilai ROC dari model tersebut untuk mengetahui model yang dibangun menghasilkan nilai akurasi diagnosa yang baik atau sebaliknya.

3 Hasil dan Pembahasan

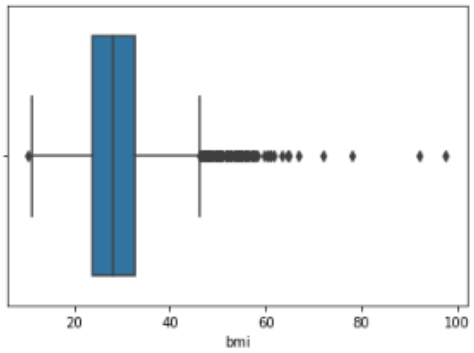
Hasil eksplorasi dari *dataset stroke*, diketahui bahwa data pasien *stroke* hanya berjumlah 249 sedangkan data pasien tidak *stroke* berjumlah 4861. Sehingga pasien tidak *stroke* lebih banyak dibandingkan data pasien *stroke*. Hal ini menandakan bahwa atribut *stroke* pada dataset *stroke* memiliki kelas data yang tidak seimbang.



Gambar. 3. Diagram *pie* di atas adalah ilustrasi dari pasien *stroke* dan pasien tidak *stroke*. Pasien *stroke* hanya menampilkan persentase sebesar 5% yang ditunjukkan oleh warna jingga dan pasien tidak *stroke* menampilkan persentase sebesar 95% yang ditunjukkan oleh warna kuning.

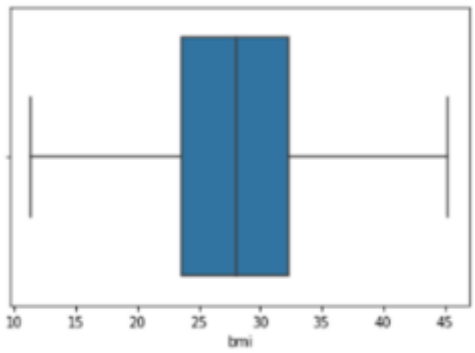
Sebelum melakukan penanganan terhadap *missing value*, dilakukan pemeriksaan menyeluruh pada keseluruhan atribut *dataset*. Didapatkan bahwa atribut *gender*, bmi, dan *smoking_status* memiliki *missing value*. Pada atribut *gender* terdapat 1 *missing value* dengan nilai NaN, atribut bmi terdapat 201 *missing value* dengan nilai NaN, dan atribut *smoking_status* terdapat 1543 *missing value* dengan nilai NaN. Nilai NaN tersebut menunjukkan bahwa informasi mengenai pasien tidak ada.

Untuk atribut *gender* dan *smoking_status* dilakukan penanganan *missing value* dengan mengisi nilai sebelum dari baris *missing value*, penanganan ini dilakukan karena kedua atribut tersebut bertipe kategorik. Sedangkan untuk atribut bmi yang bertipe numerik, *missing value* diisi dengan menggunakan nilai *median*. Selanjutnya memeriksa apakah terdapat data *outlier* dengan menggunakan *boxplot.*



Gambar. 4. Dari gambar *boxplot* di atas diketahui bahwa atribut bmi mengandung data *outlier*.

Penanganan *outlier* pada atribut bmi dilakukan dengan menggunakan metode IQR *(Interquartile Range)*, yang membuat nilai max atribut bmi yang sebelumnya adalah 97.6 menjadi 45.3 sehingga menghapus 147 baris dari 5110 baris. Maka 4963 baris digunakan pada tahap berikutnya. Berikut ini adalah *boxplot* setelah dilakukan penanganan *outlier*.



Gambar. 5. Setelah diperiksa kembali didapatkan bahwa atribut bmi telah bersih dari data *outlier*.

Dataset stroke memiliki tipe data numerik dan kategorik, dimana tipe kategorik harus diubah menjadi tipe numerik dengan angka 0, 1, 2, dan seterusnya. Atribut dengan tipe kategorik diantaranya atribut *gender, ever_married, work_type, residence_type*, dan *smoking_status*. Perubahan ini dapat dilakukan dengan menggunakan *LabelEncoder* dari *library* Python *sklearn.preprocessing.* Proses *LabelEncoder* menghasilkan tipe data *object* pada keseluruhan atribut. Karena komputer tidak dapat memproses tipe data *object*, dilakukan perubahan tipe data ke dalam format yang sesuai seperti tabel ini.

Tabel 2. Tipe Data Pada *Dataset Stroke*

Atribut	Sebelum Transformasi	Setelah Transformasi
Gender	*object*	int64
Age	*object*	int64
Hypertension	*object*	int64
Heart_Disease	*object*	int64
Ever_Married	*object*	int64
Work_Type	*object*	int64
Residence_Type	*object*	int64

Avg_Glucose_Level	*object*	float64
BMI	*object*	float64
Smoking_Status	*object*	int64
Stroke	*object*	int64

Data *train* dan data *test* dibagi dengan rasio 80/20, 80% untuk data *train* dan 20% untuk data *test*. Data *train* digunakan untuk melatih model dan data *test* digunakan untuk menguji model dalam mengetahui model tersebut menghasilkan ukuran ketepatan yang baik atau tidak pada saat klasifikasi data dilakukan. Dari 5110 baris data pada *dataset* penyakit *stroke* diperoleh 4963 baris data dari pra-proses data. Berikut ini adalah pembagian datanya.

Tabel 3. Pembagian Data *Train* dan Data *Test*

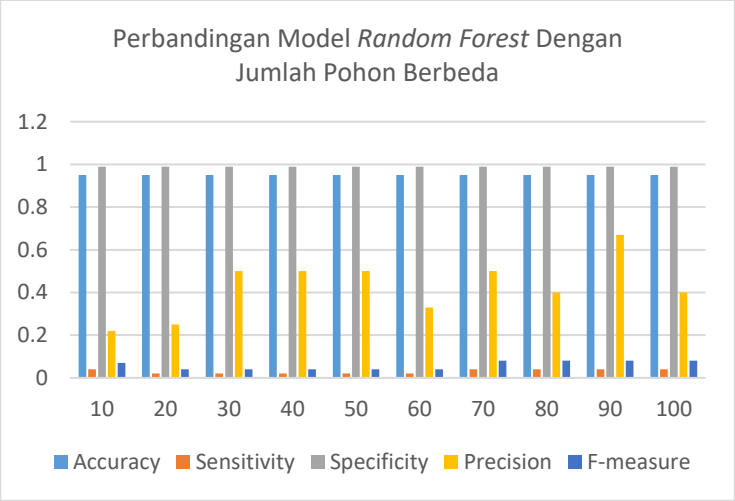
Data *Train*	**Data *Test***	**Total**
3970	993	4963

Pada penelitian ini jumlah pohon yang akan dicobakan adalah 10, 20, 30, 40, 50, 60, 70, 80, 90, dan 100. Penggunaan beberapa jumlah pohon yang berbeda tersebut bertujuan untuk memperoleh model yang menghasilkan nilai optimal. Karena *dataset stroke* memiliki kelas data yang tidak seimbang, maka dibutuhkan ukuran nilai ketepatan yang lain selain *accuracy*. Pada penelitian ini digunakan juga *sensitivity, specificity, precision,* dan *F-measure.* Berikut ini adalah hasil dari model *Random Forest* dengan menggunakan beberapa jumlah pohon yang berbeda.

Tabel 4. Perbandingan Model *Random Forest* Dengan Jumlah Pohon Berbeda

n_estimators	***Accuracy***	***Sensitivity***	***Specificity***	***Precision***	***F-measure***
10	0.946	0.04	0.99	0.22	0.07
20	0.949	0.02	0.99	0.25	0.04
30	0.951	0.02	0.99	0.50	0.04
40	0.951	0.02	0.99	0.50	0.04
50	0.951	0.02	0.99	0.50	0.04
60	0.950	0.02	0.99	0.33	0.04
70	0.951	0.04	0.99	0.50	0.08
80	0.950	0.04	0.99	0.40	0.08
90	0.952	0.04	0.99	0.67	0.08
100	0.950	0.04	0.99	0.40	0.08

Dari tabel di atas dapat terlihat bahwa jumlah pohon 90 menghasilkan nilai yang lebih optimal dibandingkan jumlah pohon lainnya, dengan menghasilkan *accuracy* 0.952, *sensitivity* 0.04, *specificity* 0.99, *precision* 0.67, dan *F-measure* 0.08. Dari tabel di atas juga diketahui bahwa penggunaan jumlah pohon yang banyak pada penelitian ini tidak membuat nilai *accuracy* meningkat cukup banyak dan semakin banyak jumlah pohon yang digunakan mengakibatkan proses komputasi menjadi lebih lama. Untuk melihat perbedaan dari setiap model dapat dilihat pada *Bar Chart* berikut.



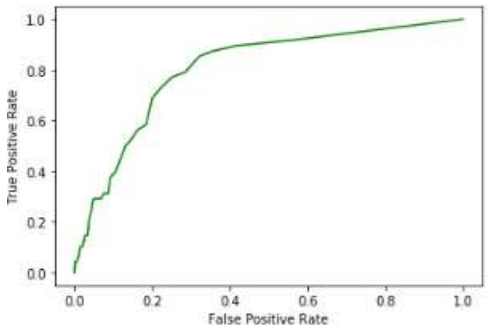
Gambar. 6. *Bar chart* model *Random Forest* dengan jumlah pohon berbeda. Warna biru muda mewakili *accuracy*, warna orange mewakili *sensitivity*, warna abu mewakili *specificity*, warna kuning mewakili *precision*, dan warna biru tua mewakili *F-measure.*

Berikut ini adalah *Confusion Matrix* dari model *Random Forest* dengan jumlah pohon 90 yang disajikan dalam bentuk tabel.

Tabel 5. *Confusion Matrix*

		Predicted		
		Stroke	**Sehat**	**Total**
Actual	***Stroke***	2	46	48
	Sehat	1	944	945
	Total	3	990	993

Untuk melihat grafik dari probabilitas prediksi dapat menggunakan ROC *Curve* yang ditampilkan dengan *library* Python *matplotlib.pyplot* sehingga menampilkan grafik seperti gambar berikut.



Gambar. 7. Grafik probabilitas model *Random Forest* dengan jumlah pohon 90.

Grafik dengan ROC *Curve* memperlihatkan bahwa kurva yang dihasilkan tidak membentuk garis diagonal sehingga model yang dibangun cukup bagus, karena apabila kurva mendekati garis diagonal maka menandakan model klasifikasi tersebut tidak akurat. Luas area di bawah ROC *Curve* adalah ukuran dari akurasi model klasifikasi. Model yang dibangun semakin semakin bagus apabila mendekati angka 1. Sehingga untuk luas area di bawah ROC *Curve* pada model ini menghasilkan nilai 0.8048, yang menunjukkan model tersebut masuk ke dalam *Good Classification.*

Namun walaupun termasuk ke dalam *Good Classification*, model ini tidak direkomendasikan untuk digunakan dalam perancangan aplikasi prediksi penyakit *stroke*. Karena mengingat bahwa *dataset stroke* yang digunakan ini memiliki kelas data tidak seimbang dan nilai dari *sensitivity* model memperoleh nilai yang sangat rendah. Padahal *sensitivity* memiliki peran yang penting dalam dataset tidak seimbang, dimana kelas minoritas bisa diklasifikasikan dengan baik. Maka, penggunaan *sensitivity* sangat penting pada dataset tidak seimbang. Untuk membangun aplikasi prediksi dengan *dataset* yang digunakan memiliki kelas data tidak seimbang, sebaiknya model yang digunakan adalah model yang menghasilkan nilai *sensitivity* yang tinggi.

4 Kesimpulan dan Saran

4.1 Kesimpulan

Metode *Random Forest* dapat digunakan untuk mendiagnosis penyakit *stroke*, pada penelitian ini dihasilkan nilai *accuracy* dengan selisih yang sedikit pada setiap penggunaan jumlah pohon yang berbeda, serta menghasilkan nilai *precision, sensitivity* dan *f-measure* yang cukup berbeda. Penggunaan jumlah pohon yang banyak pada penelitian ini tidak membuat nilai *accuracy* meningkat cukup banyak dan mengakibatkan proses komputasi menjadi lebih lama. Model dengan penggunaan jumlah pohon 90 menghasilkan nilai yang optimal, dengan memperoleh nilai *accuracy* 95.2%, nilai *sensitivity* 4.1%, nilai *specificity* 99.8%, nilai *precision* 66.7%, dan nilai *F-measure* 7.6%. Serta nilai ROC *Curve* 0.8048 yang menunjukkan bahwa model masuk ke dalam *Good Classification.*

4.2 Saran

Agar penelitian ini dapat dikembangkan menjadi lebih baik lagi, adapun saran-saran yang dapat diterapkan pada penelitian selanjutnya:

a. Menyeimbangkan *dataset stroke* dengan menerapkan metode *resampling* yaitu *Oversampling*.
b. Menggunakan metode klasifikasi yang lain, seperti *Logistic Regression*, *Support Vector Machine* (SVM), dan metode lainnya.
c. Menerapkan seleksi atribut dengan *Particle Swarm Optimization* ataupun metode seleksi atribut yang lainnya.

Referensi

[1] Khariri, & Saraswati, R. D. (2021). Transisi Epidemiologi Stroke sebagai Penyebab Kematian pada Semua Kelompok Usia di Indonesia. Seminar Nasional Riset Kedokteran (Sensorik II), 81–86.

[2] Singh, Poonam Khetrapal. (2021). World Stroke Day. Diakses pada 6 Januari 2022, dari https://www.who.int/southeastasia/news/detail/28-10-2021-world-stroke-day

[3] Gofir, A. (2021). Tatalaksana Stroke dan Penyakit Vaskuler Lain. Yogyakarta: UGM PRESS.

[4] Ridwan, M. (2017). Mengenal, Mencegah, dan Mengatasi Silent Killer, "Stroke". Yogyakarta: Hikam Pustaka.

[5] Harahap, A. H., Malik, I. M. B. A., Imam, M. I. N., Bilhaq, M. T. S., Nur, A. A., & Agustini, S. L. D. (2021). Klasifikasi Diagnosa Penyakit Jantung menggunakan Algoritma Random Forest. Teknik Informatika UIN Sunan Gunung Djati Bandung, 3, 1–51.

[6] Apriliah, W., Kurniawan, I., Baydhowi, M., & Haryati, T. (2021). Prediksi Kemungkinan Diabetes pada Tahap Awal Menggunakan Algoritma Klasifikasi Random Forest. Sistemasi, 10(1), 163. https://doi.org/10.32520/stmsi.v10i1.1129

[7] Tyasnurita, R., & Hapsari, S. W. (2020). Identification Of Chronic Kidney Disease Using Naive Bayes, Adaboost, And Random Forest Learning Methods. JITK (Jurnal Ilmu Pengetahuan Dan Teknologi Komputer), 6(1), 115–120. https://doi.org/10.33480/jitk.v6i1.1403

[8] Saputri, N. D. (2021). Komparasi Penerapan Metode Bagging Dan Adaboost Pada Algoritma C4.5 Untuk Prediksi Penyakit Stroke. Universitas Islam Negeri Sunan Ampel.

[9] Suliztia, M. L. (2020). Penerapan Analisis Random Forest pada Prototype Sistem Prediksi Harga Kamera Bekas Menggunakan Flask. In Fakultas Matematika Dan Ilmu Pengetahuan Alam. Universitas Islam Indonesia.

[10] Pramana, S., Yuniarto, B., Mariyah, S., Santoso, I., & Nooraeni, R. (2018). Data Mining dengan R Konsep Serta Implementasi. Bogor: IN MEDIA.

[11] Suyanto. (2017). Data Mining untuk Klasifikasi dan Klasterisasi Data. Bandung: Informatika Bandung.

[12] Mutiara, E. (2020). Algoritma Klasifikasi Naive Bayes Berbasis Particle Swarm Optimization Untuk Prediksi Penyakit Tuberculosis (TB). Jurnal Swabumi, 8(1), 46–58.