# BAB IV
# HASIL DAN PEMBAHASAN

Pada bab 4 akan menjelaskan mengenai tahapan *data understanding* (Pemahaman data), *data preparation* (pengolahan data), *modelling* (pelatihan model), dan *evaluation* (pengujian model dan analisis hasil) sebagaimana yang telah diuraikan pada bab 3 sebelumnya. Tahap awal adalah *data understanding* merupakan tahap pemahaman data terhadap dataset penyakit stroke. Setelah itu, dilanjutkan dengan tahap *data preparation* mencakup proses pengolahan data. Langkah berikutnya adalah pemodelan, di mana model akan dilatih menggunakan enam skenario klasifikasi yang telah ditentukan sebelumnya. Tahap akhir adalah *evaluation/*evaluasi, yang mencakup pengujian model dan analisis hasil untuk setiap skenario klasifikasi. Keseluruhan proses penelitian ini dilakukan menggunakan *tools* RapidMiner.

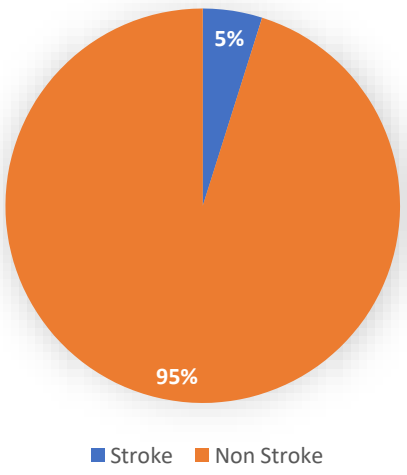
## *4.1 Data Understanding*

Pada tahap ini merupakan fase pemahaman data. Data yang digunakan yaitu *dataset stroke prediction* yang diunduh dari situs kaggle.com dengan link sebagai berikut: (https://www.kaggle.com/datasets/fedesoriano/stroke-prediction-dataset). Jumlah data awal sebelum proses preprocessing adalah sebanyak 5110 data. Dataset ini terdiri dari 12 atribut, dimana 11 di antaranya berfungsi sebagai variabel prediktor, meliputi *id, gender, age, hypertension, heart_disease, ever_married, work_type, residence_type, avg_glucose_level,* bmi (*body mass index*), dan *smoking_status*. Sementara itu, satu atribut lainnya berperan sebagai variabel target atau label, yaitu stroke. Variabel target ini memiliki dua nilai output, yaitu 0 dan 1. Nilai 0 menunjukkan bahwa pasien tidak terkena stroke, sementara nilai 1 menandakan bahwa pasien terkena stroke. Berikut Gambar 4.1 menunjukan dataset awal atau mentah yang belum dilakukan proses preprocessing data.

| id | gender | age | hypertension | heart_disease | ever_married | work_type | Residence_type | avg_glucose_level | bmi | smoking_status | stroke |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 9046 | Male | 67 | 0 | 1 | Yes | Private | Urban | 22869 | 36.6 | formerly smoked | 1 |
| 51676 | Female | 61 | 0 | 0 | Yes | Self-employed | Rural | 20221 | N/A | never smoked | 1 |
| 31112 | Male | 80 | 0 | 1 | Yes | Private | Rural | 10592 | 32.5 | never smoked | 1 |
| 60182 | Female | 49 | 0 | 0 | Yes | Private | Urban | 17123 | 34.4 | smokes | 1 |
| 1665 | Female | 79 | 1 | 0 | Yes | Self-employed | Rural | 17412 | 24 | never smoked | 1 |
| 56669 | Male | 81 | 0 | 0 | Yes | Private | Urban | 18621 | 29 | formerly smoked | 1 |
| 53882 | Male | 74 | 1 | 1 | Yes | Private | Rural | 7009 | 27.4 | never smoked | 1 |
| 10434 | Female | 69 | 0 | 0 | No | Private | Urban | 9439 | 22.8 | never smoked | 1 |
| 27419 | Female | 59 | 0 | 0 | Yes | Private | Rural | 7615 | N/A | Unknown | 1 |
| 60491 | Female | 78 | 0 | 0 | Yes | Private | Urban | 5857 | 24.2 | Unknown | 1 |
| 12109 | Female | 81 | 1 | 0 | Yes | Private | Rural | 8043 | 29.7 | never smoked | 1 |
| 12095 | Female | 61 | 0 | 1 | Yes | Govt_job | Rural | 12046 | 36.8 | smokes | 1 |
| 12175 | Female | 54 | 0 | 0 | Yes | Private | Urban | 10451 | 27.3 | smokes | 1 |
| 8213 | Male | 78 | 0 | 1 | Yes | Private | Urban | 21984 | N/A | Unknown | 1 |
| 5317 | Female | 79 | 0 | 1 | Yes | Private | Urban | 21409 | 28.2 | never smoked | 1 |
| 58202 | Female | 50 | 1 | 0 | Yes | Self-employed | Rural | 16741 | 30.9 | never smoked | 1 |
| 56112 | Male | 64 | 0 | 1 | Yes | Private | Urban | 19161 | 37.5 | smokes | 1 |
| 34120 | Male | 75 | 1 | 0 | Yes | Private | Urban | 22129 | 25.8 | smokes | 1 |
| 27458 | Female | 60 | 0 | 0 | No | Private | Urban | 8922 | 37.8 | never smoked | 1 |
| 25226 | Male | 57 | 0 | 1 | No | Govt_job | Urban | 21708 | N/A | Unknown | 1 |
| 70630 | Female | 71 | 0 | 0 | Yes | Govt_job | Rural | 19394 | 22.4 | smokes | 1 |
| 13861 | Female | 52 | 1 | 0 | Yes | Self-employed | Urban | 23329 | 48.9 | never smoked | 1 |
| 68794 | Female | 79 | 0 | 0 | Yes | Self-employed | Urban | 2287 | 26.6 | never smoked | 1 |

Gambar 4.1 Dataset yang digunakan

Berdasarkan Gambar 4.1 diatas, pada atribut stroke memiliki jumlah data pasien tidak stroke sebanyak 4861 data. Sedangkan jumlah data pasien stroke sebanyak 249 data. Oleh karena itu, dapat disimpulkan jumlah pasien tidak stroke lebih banyak daripada pasien yang mengalami stroke. Hal ini menunjukkan bahwa atribut stroke memiliki kelas data yang tidak seimbang. Gambar 4.2 dibawah ini mengilustrasikan perbandingan jumlah data pasien stroke dan pasien non stroke. Persentase pasien non stroke sebanyak 95% sedangkan persentase pasien yang terkena stroke hanya 5%.




Gambar 4.2 Persentase Pasien Stroke

## 4.2 *Data Preparation* (Pengolahan Data)

Pada *dataset stroke prediction* yang berjumlah 5110 data didalamnya terdapat beberapa data *noise*. *Noise* adalah data yang berisi nilai-nilai yang salah atau anomali, yang biasanya disebut juga *outlier*. [31] Data *noise* merupakan data yang mengalami kerusakan, seperti data duplikat, data tidak konsisten, data yang hilang/*missing value*, data tidak beraturan dan data outlier yang dapat mempengaruhi nilai akurasi pada proses klasifikasi.

### *4.2.1 Data Cleaning*

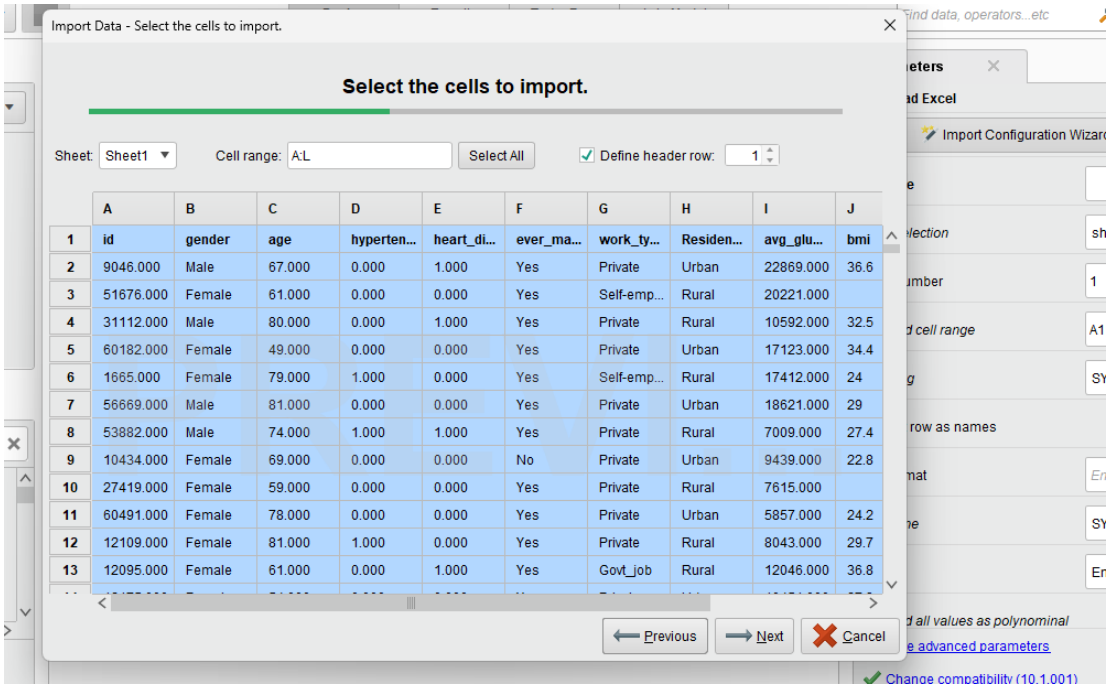
*Data Cleaning* merupakan serangkaian tindakan yang melibatkan identifikasi, penanganan, dan perbaikan masalah dalam dataset. Tujuannya adalah menghasilkan dataset yang lebih bersih, konsisten, dan sesuai untuk analisis atau pemodelan. Proses pembersihan data dilakukan pada dataset stroke prediction yang memiliki nilai tidak konsisten N/A dan *missing value*. Pada dataset ini ada atribut yang memiliki nilai N/A sebanyak 201 data yaitu atribut bmi. Data tersebut merupakan data *noise*, sehingga sebelum masuk ke tahap pelatihan model harus dihapus atau diganti agar tidak mempengaruhi nilai akurasi. Data yang tidak konsisten berupa nilai N/A akan diubah menjadi nilai rata-rata dari atribut BMI. Untuk menghandling atribut bmi yang memiliki nilai N/A, ada 2 *tools* yang digunakan pada penelitian ini yaitu Microsoft excel dan aplikasi RapidMiner. Berikut Langkah-langkah yang dilakukan:

**1. Mengubah Nilai N/A menjadi nilai kosong/*missing value***

Tahap ini merupakan tahapan yang dilakukan untuk mengubah nilai N/A menjadi nilai *missing value*/kosong menggunakan program Ms. Excel. Hal ini dilakukan supaya pada saat proses *preprocessing data* pada aplikasi RapidMiner menghasilkan nilai imputasi. Karena jika data memiliki nilai N/A tidak bisa dilakukan imputasi missing value.

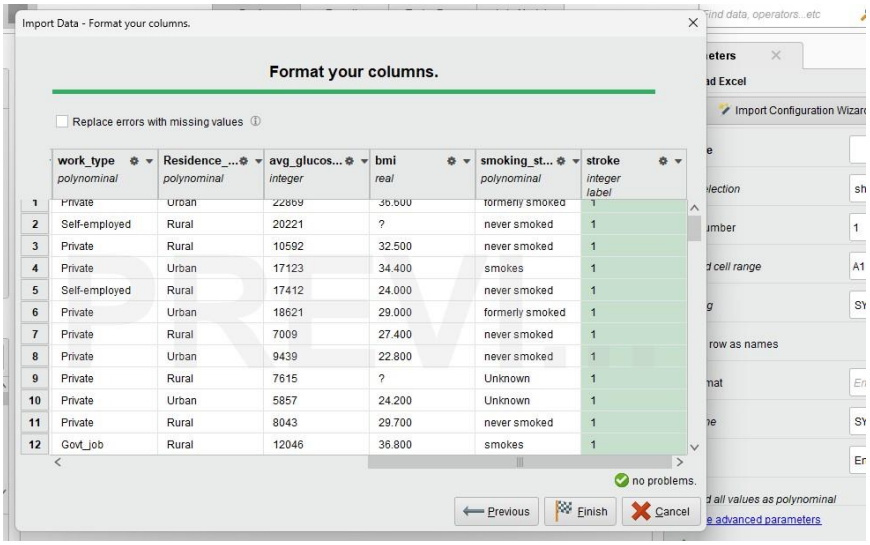
## 2. Import Data Mentah Yang Memiliki Nilai *Missing Value*

Setelah tahap pertama selesai, dilanjutkan tahap kedua yaitu *import data* yang memiliki nilai *missing value* untuk dilakukan proses *preprocessing data.*

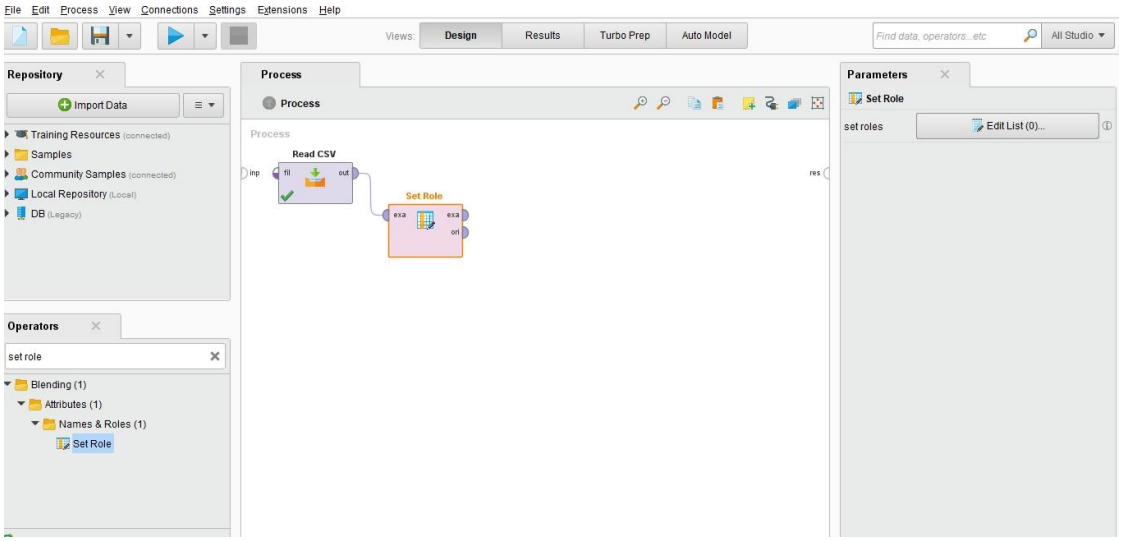



Gambar 4.3 Import Data

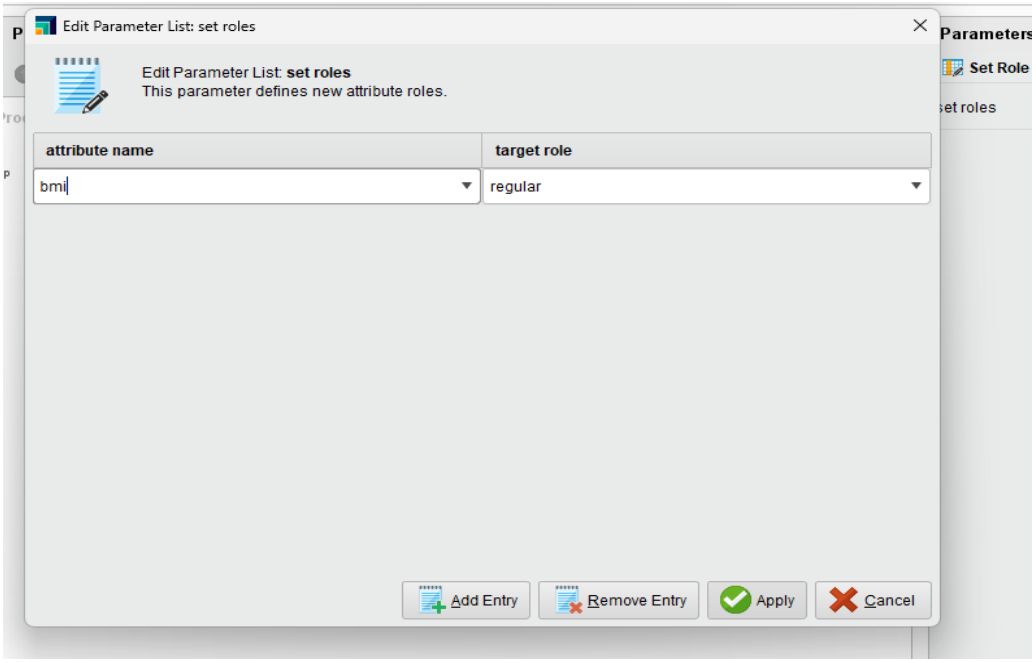
data yang memiliki *missing value* ditandai dengan tanda tanya “?”. Atribut stroke dijadikan sebagai label.




Gambar 4.4 Atribut Stroke Dijadikan Sebagai Label

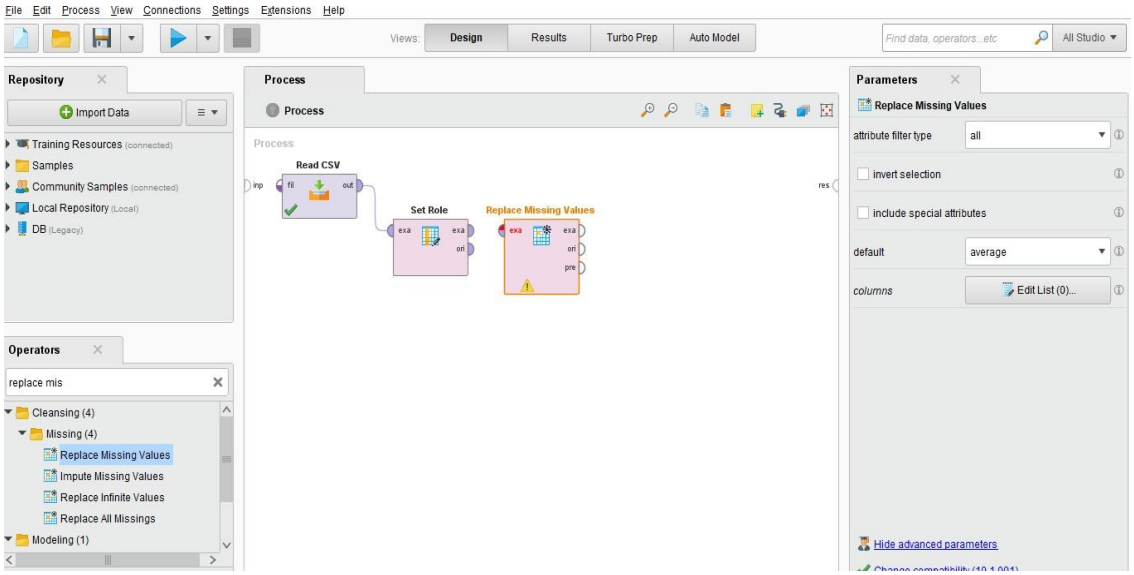



Edit parameter *set roles* dilakukan untuk mengisi atribut mana yang memiliki *missing value*. Pada dataset ini yaitu atribut bmi sebanyak 201 data.



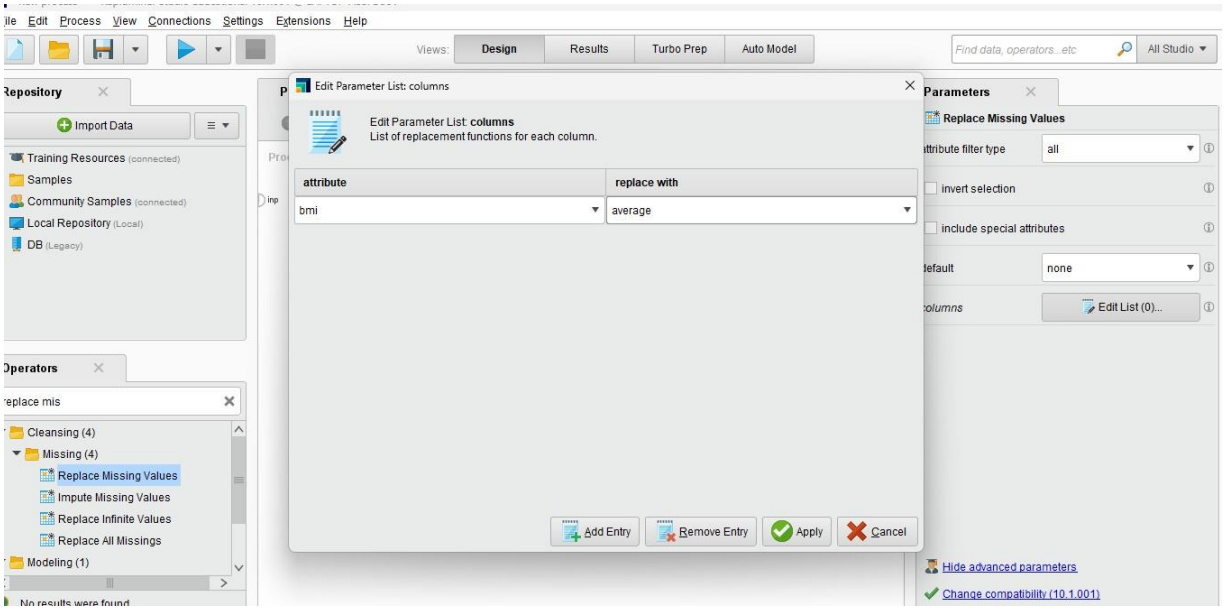

Gambar 4.5 Edit Parameter *Set Roles*

### 3. Mengganti Nilai yang *missing*/ *Replace Missing Value*




Gambar 4.6 Tampilan Menu *Replace Missing Value*

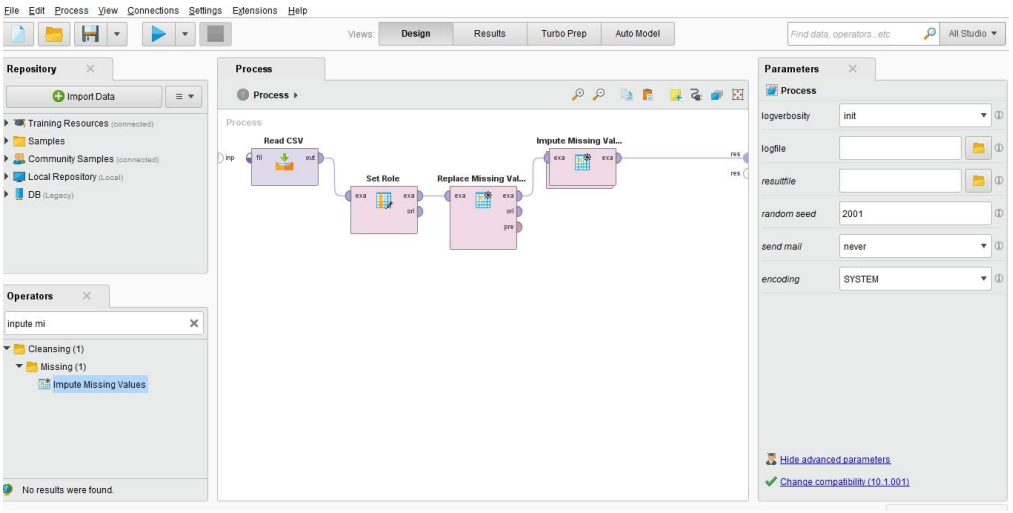
Edit list parameter pada operator *replace missing value*. Isikan atribut bmi yang memiliki nilai *missing value* dan pada *column replace with* isi dengan pilihan *average*. Karena akan di *replace* dengan nilai rata-rata.




Gambar 4.7 Setting Nilai *Missing Value* Dengan Nilai Rata-Rata

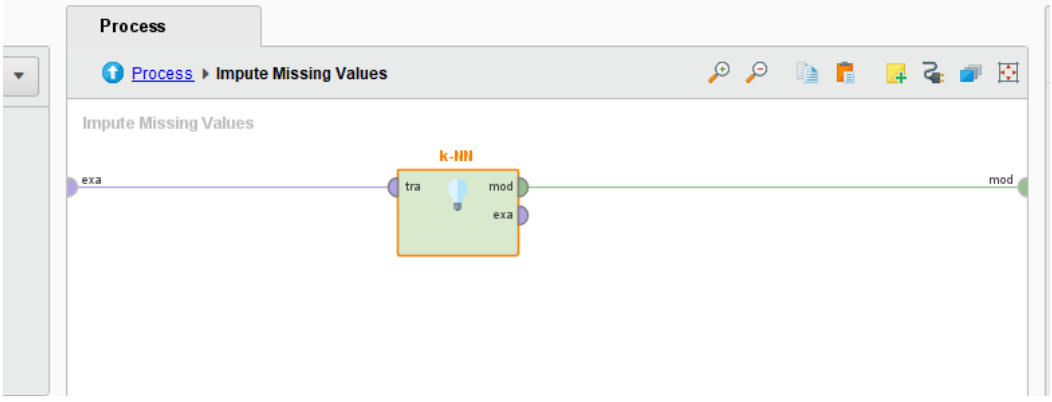
### 4. Imputasi *Missing Value*

Pada tahap ini dilakukan imputasi *missing value* pada data yang kosong/missing yaitu dengan menambahkan operator impute missing value.



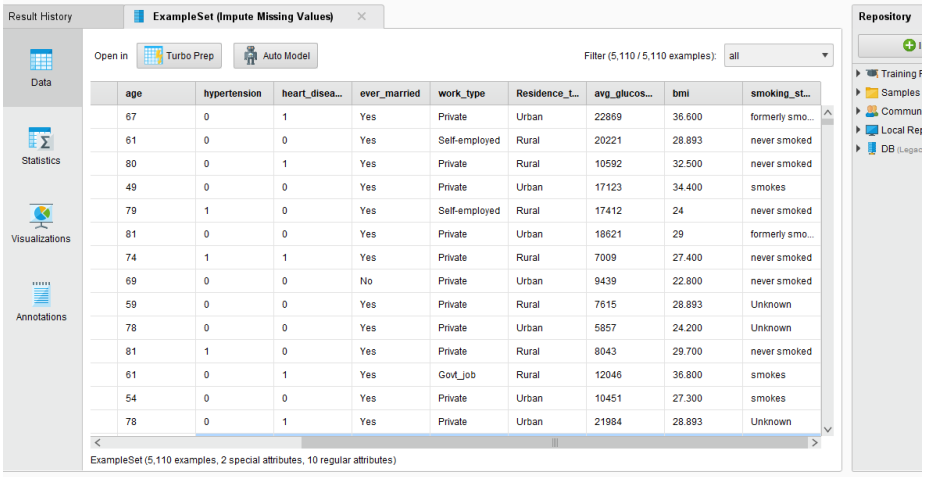

Gambar 4.8 Imputasi *Missing Value*

Di dalam proses *impute missing values* ditambahkan algoritma *machine learning* yaitu K-NN, karena algoritma tersebut menghitung nilai berdasarkan jarak tetangga terdekat.



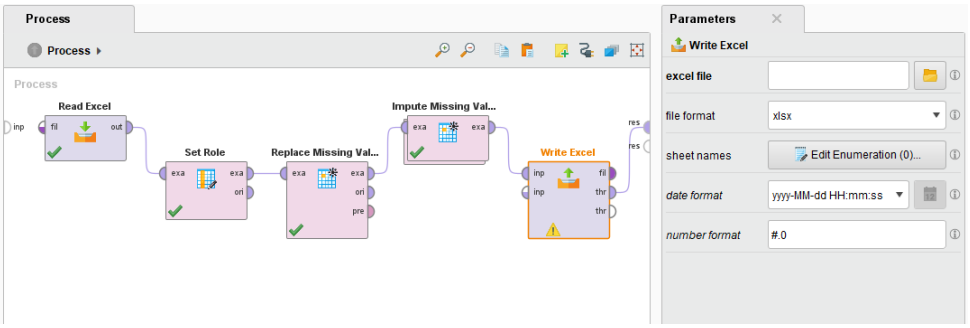

Gambar 4.9 Imputasi Missing Value dengan Algoritma K-NN

Proses imputasi *missing value* berhasil. Data yang tadinya *missing*/kosong sekarang sudah terisi dengan nilai *mean*/rata-rata. Berikut hasil *preprocessing data*:



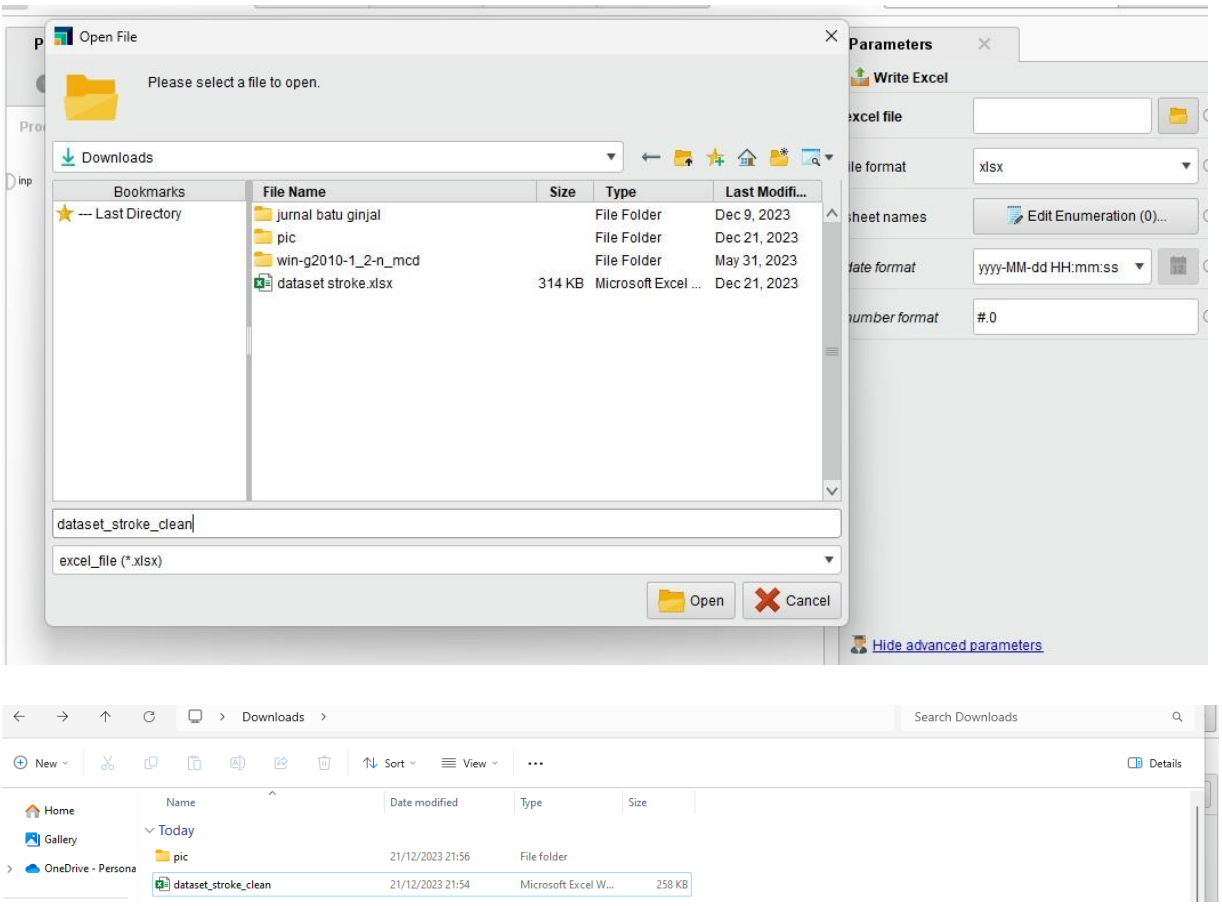

Gambar 4.10 Hasil *Preprocessing Data*

Setelah didapatkan dataset yang sudah tidak ada nilai *missing* dan siap diolah, langkah selanjutnya yaitu download file. File yang didownload berupa file excel maka operator yang digunakan yaitu *write excel*.




Gambar 4.11 Download File Excel

Setelah itu di "*Run*". Tunggu beberapa saat maka file excel yang tadi didownload akan tersimpan di *storage* yang kita pilih. Berikut tampilannya:



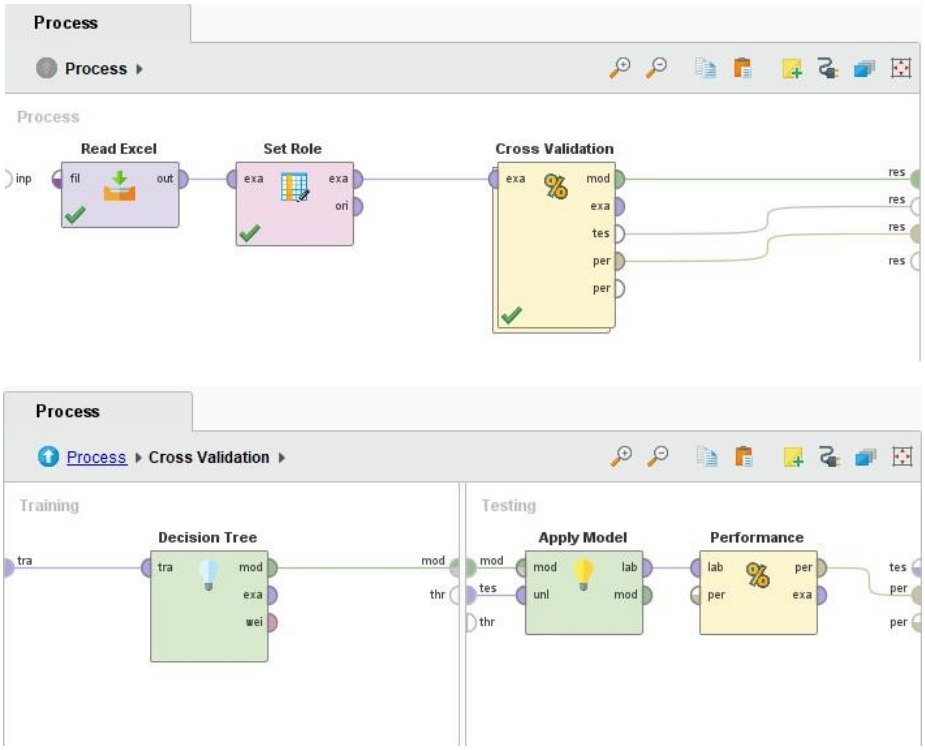

Gambar 4.12 Tampilan File Excel Yang Berhasil di Download

Setelah tahapan *preprocessing data* selesai dan data siap diolah, selanjutnya masuk ke tahap pelatihan model untuk mengetahui tingkat akurasi algoritma yang digunakan. Pada penelitian ini algoritma inti yang digunakan yaitu decision tree dan naïve bayes. Sedangkan teknik *ensemble learning* yang digunakan yaitu adaboost dan bagging. Pada pengujian 2 algoritma inti yang dikombinasi dengan teknik *ensemble* tersebut maka dibuat 6 skenario pengujian, diantaranya yaitu pengujian algoritma decision tree, naïve bayes, decision tree + bagging, decision tree + adaboost, naïve bayes + bagging dan naïve bayes + adaboost. Sehingga akan didapatkan 6 hasil akurasi.

## 4.3 *Modelling* (Pemodelan)

### 4.3.1 Klasifikasi Algoritma Decision Tree

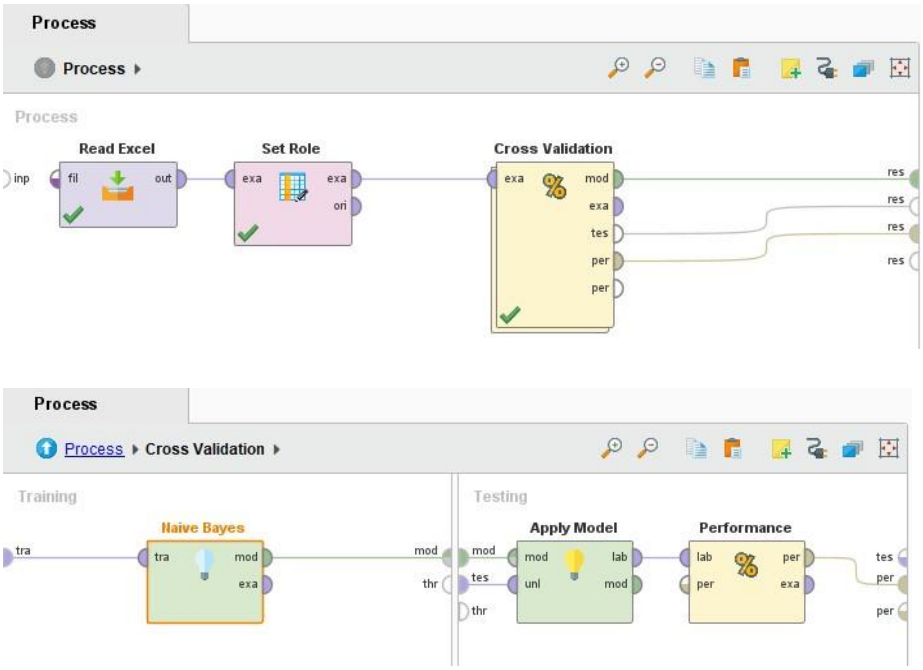
Setelah melalui proses pengolahan data atau *preprocessing data* pada *dataset stroke prediction*, selanjutnya yaitu masuk ke tahap pelatihan model. Berdasarkan skenario klasifikasi pada Tabel 3.2 pelatihan model yang pertama yaitu menggunakan algoritma decision tree. Proses klasifikasi dilakukan menggunakan *tool* RapidMiner. Terdapat 2 tahapan pada proses klasifikasi yaitu *training* dan *testing*. Dari proses klasifikasi tersebut maka dihasilkan output salah satunya berupa nilai akurasi dari algoritma decision tree. Proses klasifikasi algoritma decision tree pada aplikasi RapidMiner ditunjukkan pada Gambar 4.13 berikut.




Gambar 4.13 Klasifikasi Algoritma Decision Tree

### 4.3.2 Klasifikasi Algoritma Naïve Bayes

Pelatihan model kedua yaitu menggunakan algoritma naïve bayes. Algoritma ini adalah metode klasifikasi berbasis probabilitas yang menggunakan teorema Bayes. Langkah-langkah yang dilakukan diantaranya adalah pemahaman data, pembagian data, ekstraksi fitur, perhitungan probabilitas prior, perhitungan Probabilitas Likelihood, Perhitungan Probabilitas Posterior, Pemilihan Kelas, Evaluasi Model dan Implementasi. Proses klasifikasi menggunakan aplikasi rapid miner. Dari proses klasifikasi tersebut dihasilkan output salah satunya berupa nilai akurasi dari algoritma naïve bayes. Proses klasifikasi algoritma naïve bayes pada aplikasi rapid miner ditunjukkan pada Gambar 4.14 berikut.

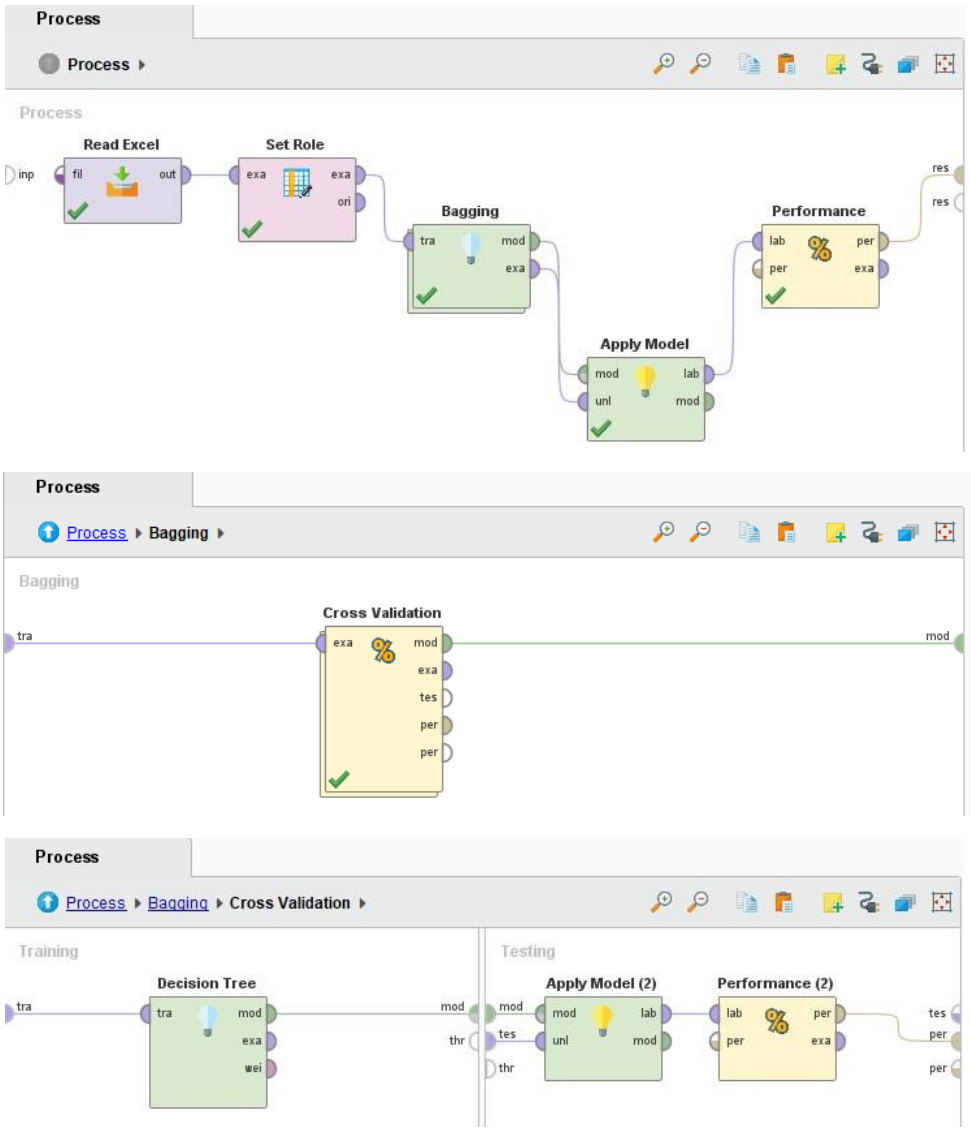



Gambar 4.14 Klasifikasi Algoritma Naïve Bayes

### 4.3.3 Klasifikasi Algoritma Decision Tree + Bagging

Pelatihan model ketiga menerapkan metode bagging pada algoritma decision tree. Bagging merupakan salah satu teknik ensemble learning,

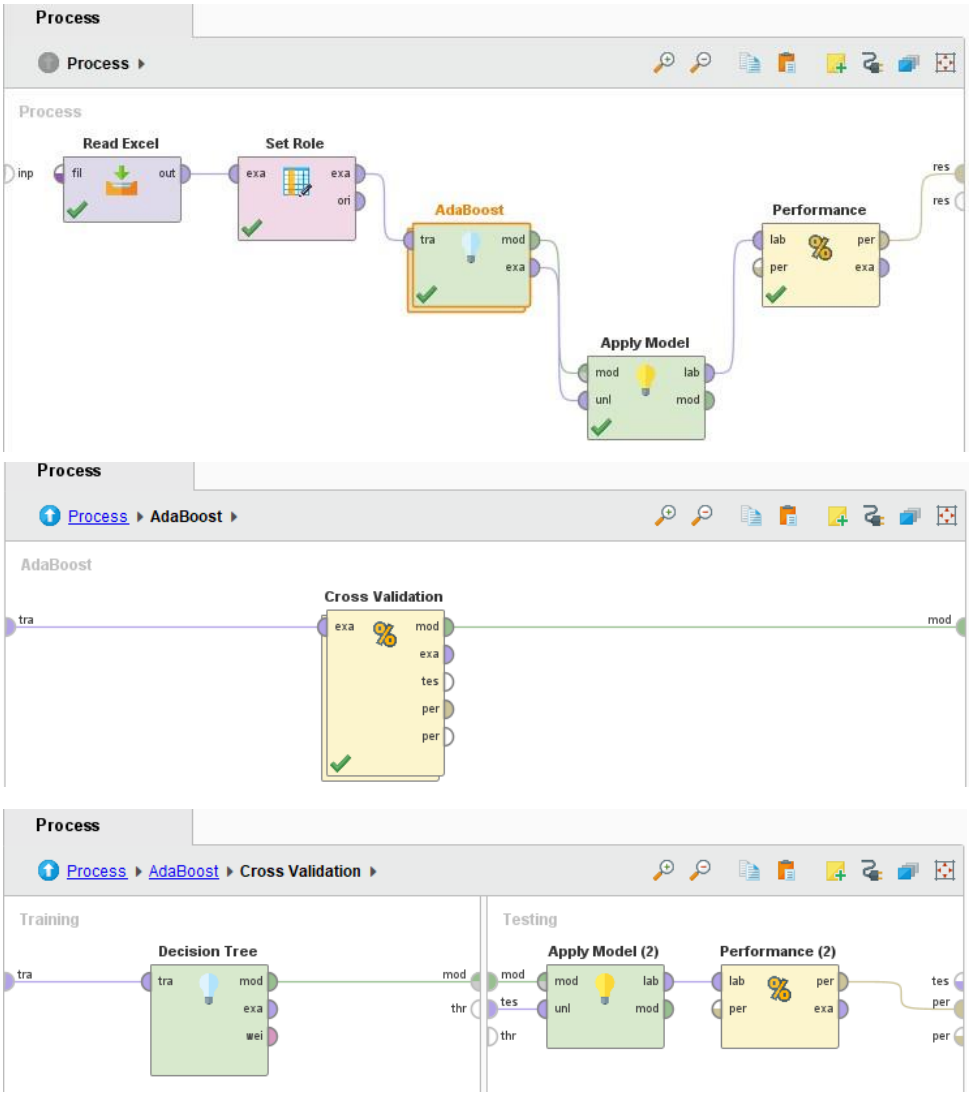
bertujuan untuk meningkatkan performa algoritma decision tree. Sehingga menghasilkan nilai akurasi yang lebih tinggi setelah dioptimalkan menggunakan metode tersebut. Proses klasifikasi algoritma decision tree yang dikombinasikan dengan metode bagging dapat dilihat pada Gambar 4.15 berikut ini.




Gambar 4.15 Klasifikasi Algoritma Decision Tree + Bagging

### 4.3.4 Klasifikasi Algoritma Decision Tree + Adaboost

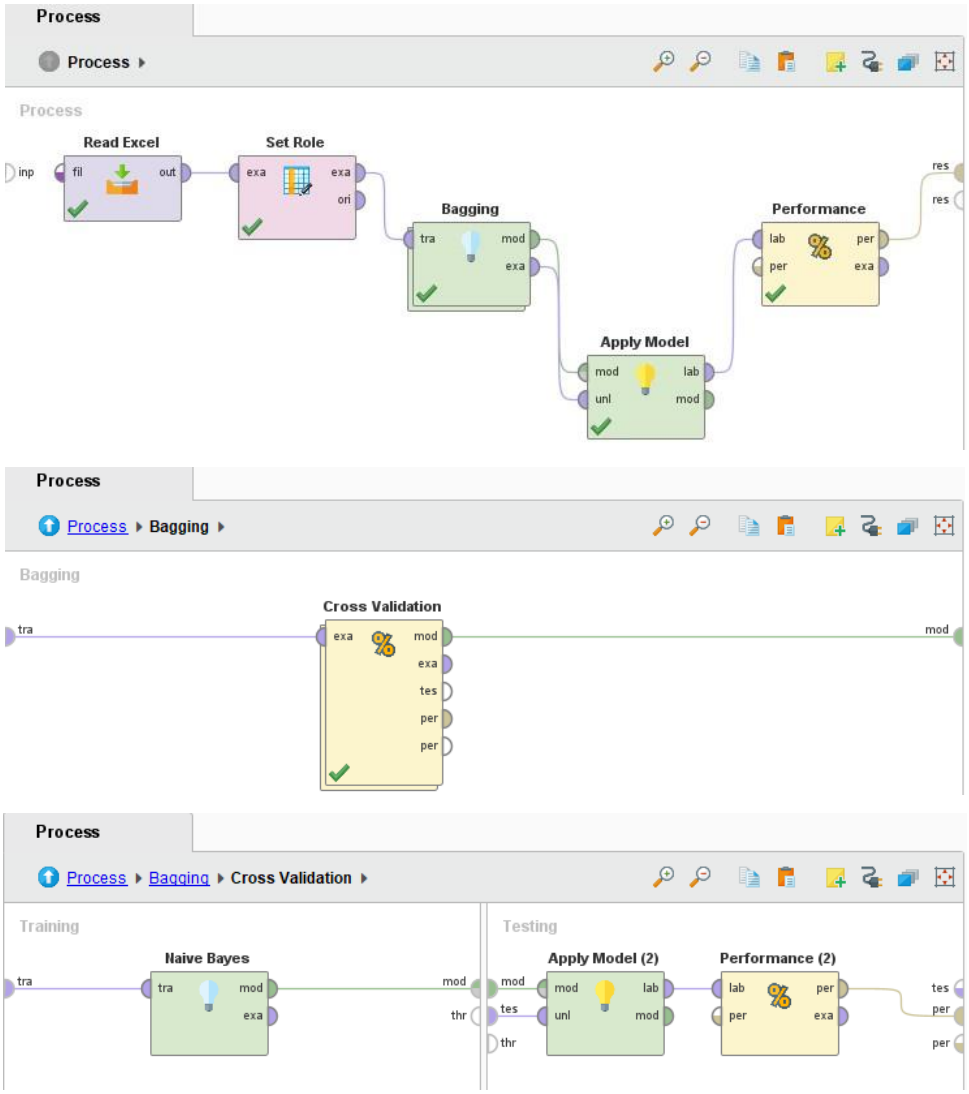
Pelatihan model keempat yaitu menerapkan metode adaboost pada algoritma decision tree. Adaboost merupakan salah satu teknik ensemble yang berfungsi sebagai boosting pada algoritma decision tree. Penerapan adaboost pada decision tree bertujuan untuk meningkatkan kinerja algoritma decision tree sehingga menghasilkan nilai akurasi yang lebih tinggi. Proses klasifikasi algoritma decision tree yang dikombinasikan dengan metode adaboost dapat dilihat pada Gambar 4.16 berikut.




Gambar 4.16 Klasifikasi Algoritma Decision Tree + Adaboost

### 4.3.5 Klasifikasi Algoritma Naïve Bayes + Bagging

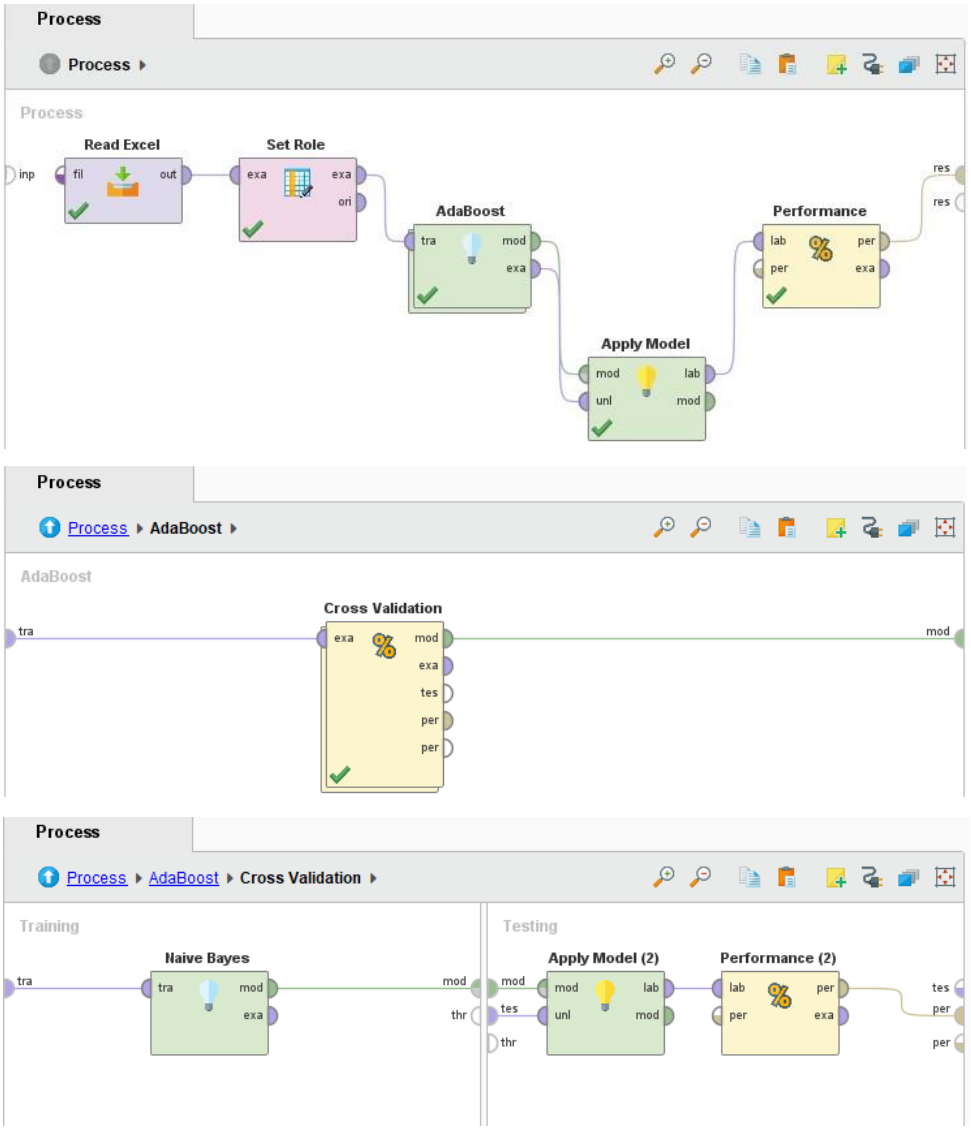
Pelatihan model kelima yaitu menerapkan metode bagging pada algoritma naïve bayes. Pada proses ini algoritma naïve bayes dikombinasikan dengan metode bagging dengan tujuan untuk meningkatkan kinerja dari algoritma naïve bayes. Sehingga diperoleh nilai akurasi yang lebih tinggi setelah dioptimalkan dengan metode bagging. Proses klasifikasi algoritma naïve bayes yang di kombinasikan dengan metode bagging dapat dilihat pada Gambar 4.17 berikut.




Gambar 4.17 Klasifikasi Algoritma Naïve Bayes + Bagging

### 4.3.6 Klasifikasi Algoritma Naïve Bayes + Adaboost

Pelatihan model yang keenam yaitu penerapan metode adaboost pada algoritma naïve bayes. Adaboost sebagai teknik ensemble berfungsi untuk meningkatkan kinerja dari algoritma naïve bayes. Proses klasifikasi algoritma naïve bayes yang dikombinasikan dengan metode adaboost dapat dilihat pada Gambar 4.18 sebagai berikut.




Gambar 4.18 Klasifikasi Algoritma Naïve Bayes + Adaboost

## 4.4 *Evaluation* (Evaluasi)

### 4.4.1 Pengujian Model

Pengujian model pada *dataset stroke prediction* dilakukan untuk mengevaluasi kinerja model dalam memprediksi kemungkinan terjadinya stroke berdasarkan berbagai atribut yang ada dalam dataset. Proses pengujian model pada dataset *stroke prediction* melibatkan beberapa langkah, diantaranya:

1. Pemisahan Data: Dataset dibagi menjadi dua bagian utama, yaitu data latih (*training data*) dan data uji (*testing data*). Data latih digunakan untuk melatih model, sedangkan data uji digunakan untuk menguji performa model. Pada penelitian ini menggunakan *cross validation* sebagai metode pengujian model untuk mengevaluasi kinerja model dengan membagi data menjadi subset yang berbeda. Operator *cross validation* yang digunakan yaitu *K-Fold Cross-Validation*. Data dibagi menjadi k subset yang sama besar. Model dilatih pada k-1 subset dan diuji pada subset yang tersisa. Proses ini diulangi k kali, sehingga setiap subset digunakan untuk pengujian. Hasil akhir adalah rata-rata atau agregasi dari k hasil pengujian.
2. Pemilihan Model: model yang digunakan pada penelitian ini yaitu algoritma klasifikasi decision tree dan naïve bayes sebagai algoritma inti.
3. Pembuatan Model: model dipelajari menggunakan data latih. Langkah ini melibatkan proses pembelajaran di mana model belajar hubungan antara atribut-atribut dalam data dengan variabel target, yaitu prediksi kemungkinan terjadinya stroke.
4. Validasi Model: model yang telah dibuat dievaluasi menggunakan data uji yang terpisah. Pada penelitian ini metrik evaluasi yang digunakan seperti akurasi, presisi, *recall*, dan kurva (AUC-ROC). Penggunaan metrik yang sesuai tergantung pada kebutuhan spesifik dalam konteks prediksi stroke.

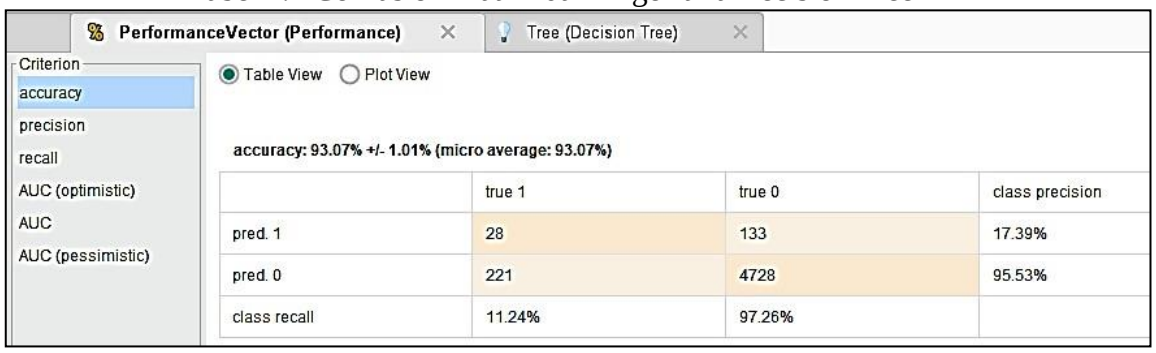
5. Optimasi: pada penelitian ini menerapkan dua teknik *ensemble* yaitu bagging dan adaboost. Dengan penerapan teknik *ensemble* bertujuan untuk meningkatkan kinerja dari algoritma inti yaitu decision tree dan naïve bayes. Sehingga dihasilkan akurasi yang jauh lebih baik setelah diterapkan teknik *ensemble* pada kedua algoritma tersebut.
6. Analisis Hasil: Hasil pengujian model dievaluasi dan dianalisis untuk memahami seberapa baik model dalam memprediksi kemungkinan terjadinya stroke. Jika ditemukan kekurangan atau masalah, langkah-langkah perbaikan atau pembaruan model dapat dilakukan.
7. Interpretasi Model: Setelah model dievaluasi, interpretasi hasil dapat membantu dalam memahami faktor-faktor apa yang berkontribusi terhadap kemungkinan terjadinya stroke. Ini dapat memberikan wawasan yang berharga untuk pemahaman lebih lanjut tentang prediksi stroke dan upaya pencegahan yang mungkin diperlukan.

Proses ini membantu dalam membangun model yang dapat membantu dalam identifikasi risiko stroke dan mengarah pada perbaikan kebijakan kesehatan serta pemberian perawatan yang lebih baik kepada individu yang berisiko.

#### 4.4.1.1 Pengujian Algoritma Decision Tree

Pada pengujian model ini, digunakan metode *confusion matrix* untuk mengidentifikasi nilai kelas yang sesungguhnya dan nilai kelas yang diprediksi dari *dataset stroke prediction*. Tabel 4.1 menampilkan hasil pengujian model *confusion matrix* menggunakan algoritma decision tree.

Tabel 4.1 Confusion Matrix dari Algoritma Decision Tree




| | true 1 | true 0 | class precision |
|---|---|---|---|
| pred. 1 | 28 | 133 | 17.39% |
| pred. 0 | 221 | 4728 | 95.53% |
| class recall | 11.24% | 97.26% | |

Diketahui :

TP (True Positive) = 28

TN (True Negative) = 4728

FP (False Positive) = 133

FN (False Negative) = 221

Dilihat dari Tabel 4.1 dapat disimpulkan bahwa pasien yang dinyatakan terkena stroke sebanyak 28, pasien dinyatakan tidak terkena stroke atau non stroke sebanyak 4728, pasien stroke yang diidentifikasi menjadi non stroke sebanyak 221 termasuk ke dalam "Type II Error", sedangkan pasien non stroke yang diidentifikasi menjadi stroke sebanyak 133 termasuk ke dalam "Type I Error". Maksud dari Type I Error atau Kesalahan tipe I adalah Pasien diprediksi stroke namun pada kenyataannya itu salah, pasien tidak stroke. Sedangkan Type II Error atau Kesalahan tipe II adalah pasien diprediksi non stroke namun pada kenyataannya pasien tersebut terkena stroke. Kesalahan tipe II ini sangat berbahaya karena memprediksi seseorang tidak terkena stroke tetapi sebenarnya positive stroke.

Selanjutnya yaitu menghitung nilai akurasi, presisi, *recall*, dan spesifisitas. Berikut rumus yang digunakan:

*a. Accuracy*

Akurasi adalah ukuran seberapa baik suatu tes atau model

mampu memberikan hasil yang benar atau tepat dalam mengklasifikasikan atau memprediksi data. Ini adalah ukuran kesesuaian antara hasil tes atau prediksi dengan kebenaran yang sebenarnya. Dalam konteks klasifikasi, akurasi mengukur persentase prediksi yang benar dari keseluruhan prediksi yang dilakukan oleh suatu model atau tes.

*Accuracy* = (TP+TN) / (TP+FP+FN+TN)

= (28+4728) / (28+133+221+4728)

= 4756/5110

= 0,9307

= 0,9307*100% = 93,07 %

*b. Precision*

Presisi adalah ukuran dari seberapa baik suatu model atau tes mampu mengidentifikasi kasus positif secara akurat dari keseluruhan kasus yang diprediksi sebagai positif. Dalam konteks klasifikasi, presisi mengukur persentase prediksi positif yang benar dari keseluruhan prediksi positif yang dilakukan oleh suatu model atau tes.

*Precision* = (TP) / (TP + FP)

= (28) / (28+133)

= 0,1739

= 0,1739 * 100% = 17,39%

*c. Recall* atau *sensitivity*

*Recall* atau *sensitivity*: ukuran dari seberapa baik suatu tes atau model mampu mengidentifikasi kasus positif yang sebenarnya ada. Dalam konteks diagnostik medis atau evaluasi model klasifikasi, sensitivitas mengukur persentase kasus positif yang benar-benar diidentifikasi sebagai positif dari keseluruhan kasus positif yang ada.

*Recall* = TP / (TP + FN)

= (28) / (28+221)

= 0, 1124

= 0, 1124 * 100%

= 11,24%

d. *Specificity*/Spesifisitas

Spesifisitas adalah ukuran dari seberapa baik suatu tes atau model mampu mengidentifikasi kasus negatif yang sebenarnya ada. Dalam konteks diagnostik medis atau evaluasi model klasifikasi, spesifisitas mengukur persentase kasus negatif yang benar-benar diidentifikasi sebagai negatif dari keseluruhan kasus negatif yang ada.

Spesifisitas = TN / (TN+FP)

= 4728 / (4728+133)

= 4728 / 4861

= 0,9726

= 0,9726*100%

= 97,26%

Berdasarkan perhitungan manual sesuai rumus baik akurasi, presisi, *recall* dan spesifisitas nilai yang dihasilkan sesuai dengan perhitungan pada aplikasi RapidMiner.

Dimana:

Akurasi : 93,07%

Presisi : 17,39%

Recall : 11,24%

Spesifisitas : 97,26%

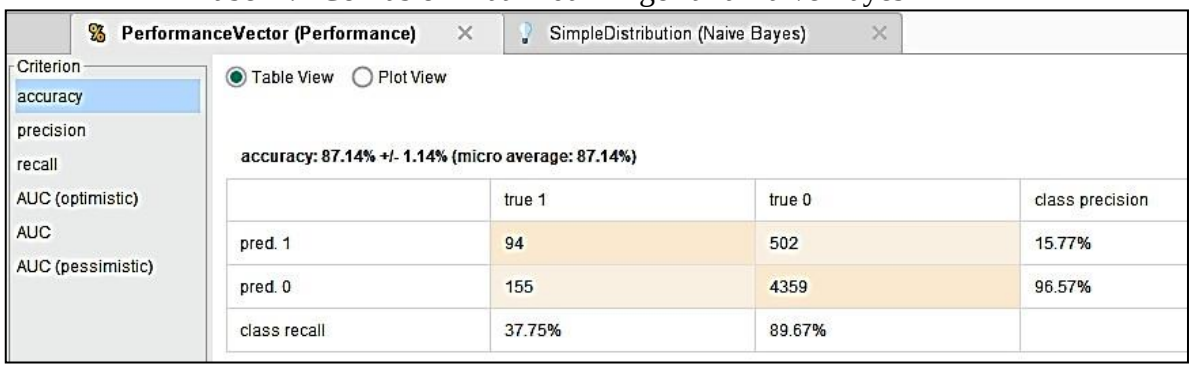
Jadi, dapat disimpulkan bahwa hasil pengujian algoritma decision tree menggunakan aplikasi RapidMiner adalah **Valid**.

4.4.1.2 Pengujian Algoritma Naïve Bayes

Pengujian model kedua menggunakan algoritma naïve bayes dengan menggunakan metode *confusion matrix* untuk mengetahui

nilai kelas yang sebenarnya dan nilai kelas yang diprediksi dari *dataset stroke prediction*. Pada Tabel 4.2 menampilkan hasil pengujian model dari *confusion matrix* menggunakan algoritma naïve bayes.

Tabel 4.2 Confusion Matrix dari Algoritma Naïve Bayes




| | true 1 | true 0 | class precision |
|---|---|---|---|
| pred. 1 | 94 | 502 | 15.77% |
| pred. 0 | 155 | 4359 | 96.57% |
| class recall | 37.75% | 89.67% | |

Diketahui :
TP (True Positive) = 94
TN (True Negative) = 4359
FP (False Positive) = 502
FN (False Negative) = 155

Berdasarkan Tabel 4.2 dapat disimpulkan bahwa pasien yang dinyatakan terkena stroke sebanyak 94, pasien dinyatakan tidak terkena stroke atau non stroke sebanyak 4359, pasien stroke yang diidentifikasi menjadi non stroke sebanyak 155 termasuk ke dalam "Type II Error", sedangkan pasien non stroke yang diidentifikasi menjadi stroke sebanyak 502 termasuk ke dalam "Type I Error". Maksud dari Type I Error atau Kesalahan tipe I adalah Pasien diprediksi stroke namun pada kenyataannya prediksi itu salah, pasien tidak stroke. Sedangkan Type II Error atau Kesalahan tipe II adalah pasien diprediksi non stroke namun pada kenyataannya pasien tersebut terkena stroke. Kesalahan tipe II ini sangat berbahaya karena memprediksi seseorang tidak terkena stroke tetapi sebenarnya stroke.

Selanjutnya yaitu menghitung nilai akurasi, presisi *recall* dan spesifisitas.

Berikut rumus perhitungannya:

*a. Accuracy* = (TP+TN) / (TP+FP+FN+TN)
= (94+4359) / (94+502+155+4359)
= 4453/5110
= 0,8714
= 0,8714*100% = 87,14 %

*b. Precision* = (TP) / (TP + FP)
= (94) / (94+502)
= 0,1577
= 0,1577 * 100% = 15,77%

*c. Recall* = TP / (TP + FN)
= (94) / (94+155)
= 0, 3775
= 0, 3775 * 100%
= 37,75%

*d.* Spesifisitas= TN / (TN+FP)
= (4359) / (4359+502)
= 0, 8967
= 0, 8967 * 100%
= 89,67%

Berdasarkan perhitungan manual sesuai rumus baik akurasi, presisi, *recall* dan spesifisitas nilai yang dihasilkan sesuai dengan perhitungan pada aplikasi RapidMiner.

Dimana:

Akurasi : 87,14%
Presisi : 15,77%
Recall : 37,75%
Spesifisitas : 89,67%

Jadi, dapat disimpulkan bahwa hasil pengujian algoritma naïve

bayes menggunakan aplikasi RapidMiner adalah **Valid**.

#### 4.4.1.3 Pengujian Algoritma Decision Tree + Bagging

Pada pengujian model ketiga, dilakukan kombinasi antara algoritma decision tree dan bagging dengan menerapkan metode *confusion matrix* untuk mengidentifikasi nilai kelas sebenarnya dan nilai kelas prediksi dari dataset prediksi stroke. Tabel 4.3 menampilkan hasil pengujian model dari confusion matrix menggunakan algoritma decision tree dan bagging.

Tabel 4.3 Confusion Matrix dari Algoritma Decision Tree dan Bagging

PerformanceVector (Performance)

Criterion
- accuracy
- precision
- recall
- AUC (optimistic)
- AUC
- AUC (pessimistic)

Table View | Plot View

**accuracy: 96.91%**

| | true 1 | true 0 | class precision |
|---|---|---|---|
| pred. 1 | 92 | 1 | 98.92% |
| pred. 0 | 157 | 4860 | 96.87% |
| class recall | 36.95% | 99.98% | |

Diketahui :

TP (True Positive) = 92

TN (True Negative) = 4860

FP (False Positive) = 1

FN (False Negative) = 157

Berdasarkan Tabel 4.3 dapat disimpulkan bahwa pasien yang dinyatakan terkena stroke sebanyak 92, pasien dinyatakan tidak terkena stroke atau non stroke sebanyak 4860, pasien stroke yang diidentifikasi menjadi non stroke sebanyak 157 termasuk ke dalam "Type II Error", sedangkan pasien non stroke yang diidentifikasi menjadi stroke sebanyak 1 termasuk ke dalam "Type I Error". Selanjutnya yaitu menghitung nilai akurasi, presisi, *recall* dan spesifisitas. Berikut rumus yang digunakan:

a. *Accuracy* = (TP+TN) / (TP+FP+FN+TN)
= (92+4860) / (92+1+157+4860)
= 4952/5110
= 0,9691
= 0,9691*100% = 96,91%

b. *Precision* = (TP) / (TP + FP)
= (92) / (92+1)
= 0,9892
= 0,9892 * 100% = 98,92%

c. *Recall* = TP / (TP + FN)
= (92) / (92+157)
= 0,3695
= 0,3695 * 100%
= 36,95%

d. Spesifisitas= TN / (TN+FP)
= (4860) / (4860+1)
= 0,9998
= 0,9998 *100%
= 99,98%

Berdasarkan perhitungan manual sesuai rumus matematis baik akurasi, presisi, *recall* dan spesifisitas nilai yang dihasilkan sesuai dengan perhitungan pada aplikasi RapidMiner.

Dimana:

Akurasi : 96,91%
Presisi : 98,92%
Recall : 36,95%
Spesifisitas : 99,98%

Jadi, dapat disimpulkan bahwa hasil pengujian algoritma decision tree yang dikombinasikan dengan bagging menggunakan aplikasi RapidMiner adalah **Valid**.

#### 4.4.1.4 Pengujian Algoritma Decision Tree + Adaboost

Pengujian model keempat yaitu kombinasi algoritma decision tree dan adaboost menerapkan metode *confusion matrix* untuk mengetahui nilai kelas sebenarnya dan nilai kelas prediksi dari *dataset stroke prediction*. Pada Tabel 4.4 menampilkan hasil pengujian model dari *confusion matrix* menggunakan algoritma decision tree dan adaboost.

Tabel 4.4 Confusion Matrix dari Algoritma Decision Tree dan Adaboost

PerformanceVector (Performance)

Criterion
- accuracy
- precision
- recall
- AUC (optimistic)
- AUC
- AUC (pessimistic)

Table View | Plot View

accuracy: 96.58%

| | true 1 | true 0 | class precision |
|---|---|---|---|
| pred. 1 | 102 | 28 | 78.46% |
| pred. 0 | 147 | 4833 | 97.05% |
| class recall | 40.96% | 99.42% | |

Diketahui :
TP (True Positive) = 102
TN (True Negative) = 4833
FP (False Positive) = 28
FN (False Negative) = 147

Dari Tabel 4.4, dapat diketahui bahwa terdapat 102 pasien yang terdiagnosis menderita stroke, 4833 pasien dinyatakan tidak menderita stroke, dan 147 pasien stroke yang salah diidentifikasi sebagai non-stroke, yang termasuk dalam kategori "Type II Error". Selain itu, 28 pasien non-stroke yang salah diprediksi sebagai stroke, yang masuk dalam kategori "Type I Error". Selanjutnya yaitu menghitung nilai akurasi, presisi, *recall* dan spesifisitas. Berikut rumus yang digunakan:

*a. Accuracy* = (TP+TN) / (TP+FP+FN+TN)
= (102+4833) / (102+28+147+4833)

= 4935/5110

= 0,9658

= 0,9658*100% = 96,58%

*b. Precision* = (TP) / (TP + FP)

= (102) / (102+28)

= 0,7846

= 0,7846 * 100% = 78,46%

*c. Recall* = TP / (TP + FN)

= (102) / (102+147)

= 0,4096

= 0,4096 * 100%

= 40,96%

e. Spesifisitas= TN / (TN+FP)

= (4833) / (4833+28)

= 0,9942

= 0,9942*100%

= 99,42%

Berdasarkan perhitungan manual sesuai rumus matematis baik akurasi, presisi, *recall*, dan spesifisitas untuk nilai yang dihasilkan sesuai dengan perhitungan pada aplikasi RapidMiner.

Dimana:

Akurasi : 96,58%

Presisi : 78,46%

Recall : 40,96%

Spesifisitas : 99,42%

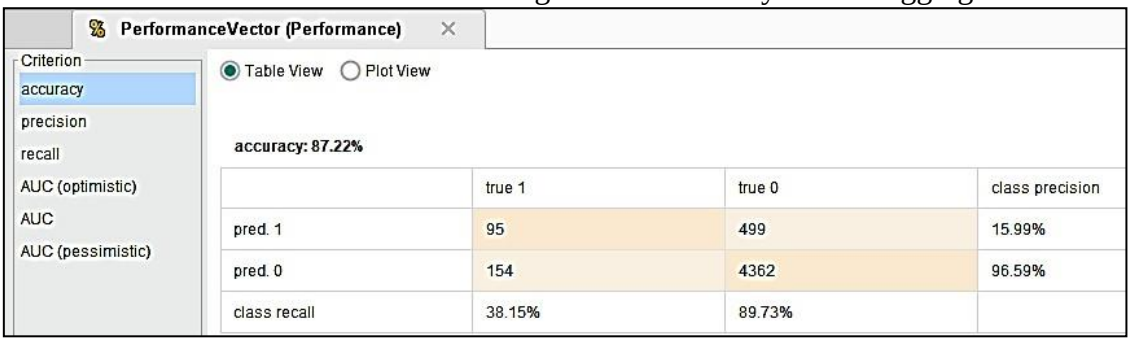
Jadi, dapat disimpulkan bahwa hasil pengujian algoritma decision tree dan adaboost menggunakan aplikasi RapidMiner adalah **Valid**.

#### 4.4.1.5 Pengujian Algoritma Naïve Bayes + Bagging

Pada pengujian model kelima, dilakukan pengujian algoritma naïve bayes yang telah dikombinasikan dengan bagging

dan menerapkan metode *confusion matrix* untuk mengidentifikasi nilai kelas sebenarnya dan nilai kelas prediksi dari *dataset stroke prediction*. Tabel 4.5 menampilkan hasil pengujian model dari *confusion matrix* menggunakan algoritma naïve bayes dan teknik bagging.

Tabel 4.5 Confusion Matrix dari Algoritma Naïve Bayes dan Bagging




Diketahui :

TP (True Positive) = 95

TN (True Negative) = 4362

FP (False Positive) = 499

FN (False Negative) = 154

Dari Tabel 4.5 dapat disimpulkan bahwa terdapat 95 pasien yang terdiagnosis menderita stroke, 4362 pasien dinyatakan tidak menderita stroke, dan 154 pasien stroke yang salah diprediksi sebagai non-stroke, yang termasuk dalam kategori "Type II Error". Selain itu, 499 pasien non-stroke salah diprediksi sebagai stroke, yang masuk dalam kategori "Type I Error". Selanjutnya yaitu menghitung nilai akurasi, presisi, *recall* dan spesifisitas. Berikut rumus yang digunakan:

*a. Accuracy* = (TP+TN) / (TP+FP+FN+TN)

= (95+4362) / (95+499+154+4362)

= 4457/5110

= 0,8722

= 0,8722*100% = 87,22%

*b.* *Precision* = (TP) / (TP + FP)

= (95) / (95+499)

= 0,1599

= 0,1599*100% = 15,99%

*c.* *Recall* atau *sensitivity*

*Recall* = TP / (TP + FN)

= (95) / (95+154)

= 0,3815

= 0,3815*100%

= 38,15%

*d.* Specificity/Spesifisitas = TN / (TN+FP)

= (4362) / (4362+499)

= 0,8973

= 0,8973*100%

= 89,73%

Berdasarkan perhitungan manual diatas, sesuai rumus yang digunakan baik akurasi, presisi, *recall* dan spesifisitas nilai yang dihasilkan sesuai dengan perhitungan pada aplikasi RapidMiner.

Dimana:

Akurasi : 87,22%

Presisi : 15,99%

Recall : 38,15%

Spesifisitas : 89,73%

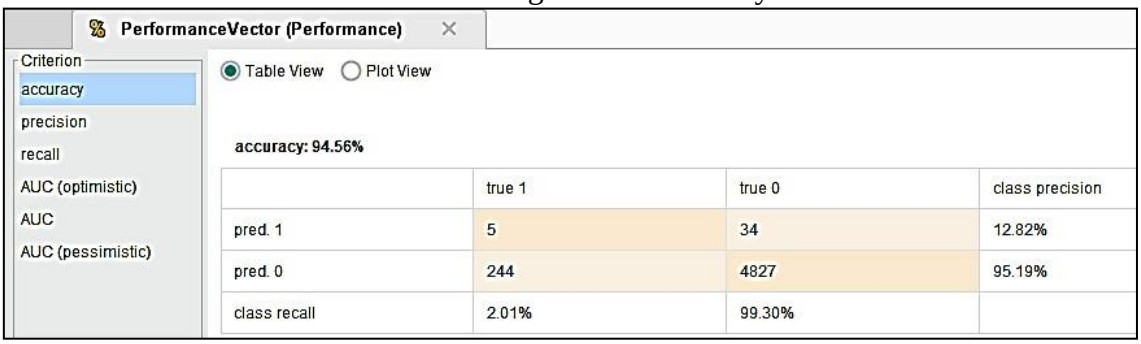
Jadi, dapat disimpulkan bahwa hasil pengujian algoritma naïve bayes yang dikombinasikan dengan teknik bagging menggunakan aplikasi RapidMiner adalah **Valid**.

#### 4.4.1.6 Pengujian Algoritma Naïve Bayes + Adaboost

Pengujian model keenam yaitu pengujian algoritma naïve

bayes yang dikombinasikan dengan adaboost menerapkan metode *confusion matrix*. Pada Tabel 4.6 menampilkan hasil pengujian model menggunakan algoritma naïve bayes dan adaboost.

Tabel 4.6 Confusion Matrix dari Algoritma Naïve Bayes dan Adaboost




| | true 1 | true 0 | class precision |
|---|---|---|---|
| pred. 1 | 5 | 34 | 12.82% |
| pred. 0 | 244 | 4827 | 95.19% |
| class recall | 2.01% | 99.30% | |

Diketahui:

TP (True Positive) = 5

TN (True Negative) = 4827

FP (False Positive) = 34

FN (False Negative) = 244

Berdasarkan Tabel 4.6 dapat ditarik kesimpulan bahwa terdapat 28 pasien yang terdiagnosis menderita stroke, 4827 pasien dinyatakan tidak menderita stroke, dan 244 pasien stroke salah diprediksi sebagai non-stroke, yang masuk dalam kategori "Type II Error". Selain itu, 34 pasien non-stroke salah diprediksi sebagai stroke, yang termasuk dalam kategori "Type I Error". Maksud dari Type I Error atau Kesalahan tipe I adalah Pasien diprediksi stroke namun pada kenyataannya itu salah, pasien tidak stroke. Sedangkan Type II Error atau Kesalahan tipe II adalah pasien diprediksi non stroke namun pada kenyataannya pasien tersebut terkena stroke. Kesalahan tipe II ini sangat berbahaya karena memprediksi seseorang tidak terkena stroke tetapi sebenarnya positive stroke.

Selanjutnya yaitu menghitung nilai akurasi, presisi, *recall* dan spesifisitas. Berikut rumus yang digunakan:

a. Akurasi = (TP+TN) / (TP+FP+FN+TN)
= (5+4827) / (5+34+244+4827)
= 4832/5110
= 0,9456
= 0,9456*100% = 94,56%

b. *Precision*/Presisi = (TP) / (TP + FP)
= (5) / (5+34)
= 0,1282
= 0,1282*100% = 12,82%

c. *Recall* atau *sensitivity*
*Recall* = TP / (TP + FN)
= (5) / (5+244)
= 0,0201
= 0,0201*100%
= 2,01%

d. Specificity/Spesifisitas = TN / (TN+FP)
= 4827 / (4827+34)
= 0,9930
= 0,9930*100%
= 99,30%

Berdasarkan perhitungan manual sesuai rumus baik akurasi, presisi, *recall* dan spesifisitas nilai yang dihasilkan sesuai dengan perhitungan pada aplikasi rapid miner.

Dimana:

Akurasi : 94,56%
Presisi : 12,82%
Recall : 2,01%
Spesifisitas : 99,30%

Jadi, dapat disimpulkan bahwa hasil pengujian algoritma naïve bayes dan adaboost menggunakan aplikasi RapidMiner adalah **Valid**.

### 4.4.2 Analisis Hasil

Setelah melakukan pengujian model dari skenario pertama hingga keenam pada *dataset stroke prediction*, langkah berikutnya adalah menganalisis hasil dengan memperbandingkan kinerja klasifikasi. Bagian ini akan membahas akurasi, presisi, *recall*, dan spesifisitas dari setiap skenario pengujian. Selanjutnya, nilai-nilai tersebut akan dibandingkan untuk mencari hasil optimal. Perbandingan hasil pengujian ini diharapkan dapat memberikan solusi terhadap permasalahan yang telah disampaikan sebelumnya. Skenario pertama yaitu klasifikasi *dataset stroke prediction* menggunakan algoritma decision tree. Model algoritma decision tree ketika diterapkan ke dalam aplikasi RapidMiner menggunakan tingkat kedalaman pohon atau *max_depth* sebanyak 10 tingkat dan pada pilihan criterion menggunakan *information_gain* bertujuan untuk menghasilkan model keputusan yang seimbang antara kompleksitas, kinerja, dan interpretabilitas yang optimal. Karena pemilihan kriteria menggunakan *information gain* sebagai kriteria untuk pemisahan node dalam pohon keputusan bertujuan untuk memaksimalkan penurunan ketidakpastian atau meningkatkan informasi saat memilih fitur untuk membagi data. Ini membantu dalam membangun pohon keputusan yang lebih informatif dan efektif dalam klasifikasi. Pengujian model algoritma decision tree menghasilkan nilai akurasi 93,07%, presisi sebesar 17,39%, recall/sensitivitas sebesar 11,24% dan spesifisitas sebesar 97,26%. Hasil pengujian skenario pertama menggunakan algoritma decision tree dapat dilihat pada Tabel 4.7 berikut.

Tabel 4.7 Hasil Pengujian Skenario Pertama

| **Uji Validasi** | **Algoritma Decision Tree** |
|---|---|
| Akurasi | 93,07% |
| Presisi | 17,39% |
| Recall/ Sensitivitas | 11,24% |
| Spesifisitas | 97,26% |

Skenario kedua yaitu pengujian model menggunakan algoritma klasifikasi naïve bayes. Nilai performa yang dihasilkan dari algoritma naïve bayes yaitu nilai akurasi sebesar 87,14%, presisi 15,77%, recall/sensitivitas 37,75% dan spesifisitas sebesar 89,67%. Detail hasil pengujian dari skenario kedua dapat dilihat pada Tabel 4.8 berikut ini.

Tabel 4.8 Hasil Pengujian Skenario Kedua

| **Uji Validasi** | **Algoritma Naïve Bayes** |
|---|---|
| Akurasi | 87,14% |
| Presisi | 15,77% |
| Recall/ Sensitivitas | 37,75% |
| Spesifisitas | 89,67% |

Skenario ketiga yaitu klasifikasi algoritma decision tree yang dikombinasi dengan metode bagging. Model dalam skenario ini merupakan gabungan antara algoritma decision tree dan metode bagging dengan tujuan meningkatkan kinerja dari pengklasifikasi yang lemah, yaitu algoritma decision tree. Gabungan model ini menghasilkan performa dengan nilai akurasi mencapai 96,91%, presisi sebesar 98,92%, recall/sensitivitas sebesar 36,95%, dan spesifisitas sebesar 99,98%. Detail hasil pengujian dari skenario ketiga terdapat dalam Tabel 4.9 di bawah ini.

Tabel 4.9 Hasil Pengujian Skenario Ketiga

| **Uji Validasi** | **Algoritma Decision Tree + Bagging** |
|---|---|
| Akurasi | 96,91% |
| Presisi | 98,92% |
| Recall/ Sensitivitas | 36,95% |
| Spesifisitas | 99,98% |

Skenario keempat melibatkan pengujian model dengan

menggunakan algoritma klasifikasi decision tree dan Adaboost. Model pada skenario ini merupakan gabungan antara algoritma decision tree dan metode Adaboost, dengan tujuan yang sama seperti skenario sebelumnya, yaitu meningkatkan kinerja algoritma decision tree. Pengujian model algoritma decision tree yang dipadukan dengan metode Adaboost menghasilkan nilai akurasi sebesar 96,58%, presisi 78,46%, recall/sensitivitas sebesar 40,96%, dan spesifisitas sebesar 99,42%. Detail hasil pengujian model dari skenario keempat terdokumentasi pada Tabel 4.10 di bawah ini.

Tabel 4.10 Hasil Pengujian Skenario Keempat

| **Uji Validasi** | **Algoritma Decision Tree + Adaboost** |
|---|---|
| Akurasi | 96,91% |
| Presisi | 98,92% |
| Recall/ Sensitivitas | 36,95% |
| Spesifisitas | 99,98% |

Skenario kelima yaitu pengujian model menggunakan algoritma naïve bayes dan teknik bagging. Model skenario ini merupakan model kombinasi dari algoritma naïve bayes dan teknik bagging yang bertujuan untuk meningkatkan kinerja dari algoritma naïve bayes. Pengujian model algoritma naïve bayes yang dikombinasikan dengan metode bagging menghasilkan nilai akurasi sebesar 87,22%, presisi 15,99%, recall/sensitivitas 38,15% dan spesifisitas sebesar 89,73%. Tabel 4.11 di bawah ini merupakan detail hasil pengujian model dari skenario kelima.

Tabel 4.11 Hasil Pengujian Skenario Kelima

| **Uji Validasi** | **Naïve Bayes + Bagging** |
|---|---|
| Akurasi | 87,22% |
| Presisi | 15,99% |
| Recall/ Sensitivitas | 38,15% |
| Spesifisitas | 89,73% |

Skenario keenam yaitu pengujian menggunakan algoritma naïve bayes dan adaboost. Pada model skenario ini dilakukan kombinasi algoritma naïve bayes dengan metode adaboost yang bertujuan untuk meningkatkan kinerja algoritma naïve bayes. Pengujian model algoritma naïve bayes yang dioptimasi dengan metode adaboost menghasilkan nilai akurasi sebesar 94,56%, presisi 12,82%, recall/sensitivitas 2,01% dan spesifisitas sebesar 99,30%. Tabel 4.12 di bawah ini menunjukkan detail hasil pengujian model dari skenario keenam menggunakan algoritma naïve bayes dan adaboost.

Tabel 4.12 Hasil Pengujian Skenario Keenam

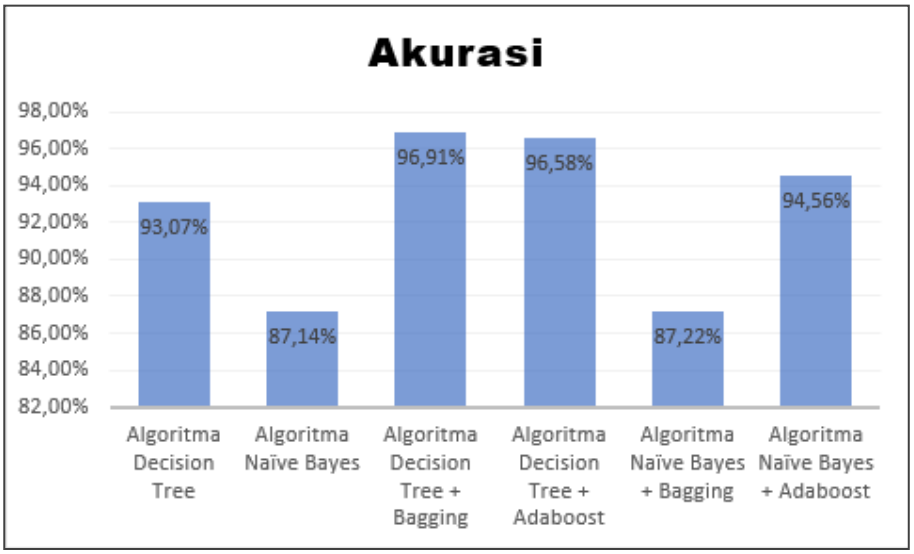
| **Uji Validasi** | **Naïve Bayes + Adaboost** |
|---|---|
| Akurasi | 94,56% |
| *Precision*/Presisi | 12,82% |
| Recall/ Sensitivitas | 2,01% |
| Spesifisitas | 99,30% |

Berdasarkan Tabel 4.9, 4.10, 4.11 dan 4.12 terlihat bahwa penggunaan teknik bagging dan adaboost memiliki dampak positif terhadap peningkatan nilai akurasi dari algoritma Decision Tree dan Naïve Bayes. Kombinasi algoritma dengan metode bagging dan adaboost untuk klasifikasi *Dataset Stroke Prediction* secara keseluruhan menghasilkan kinerja yang lebih baik daripada hanya menerapkan satu metode saja. Tabel 4.13 berikut merupakan detail perbandingan hasil pengujian untuk keseluruhan skenario klasifikasi.

Tabel 4.13 Perbandingan Hasil Pengujian 6 Skenario

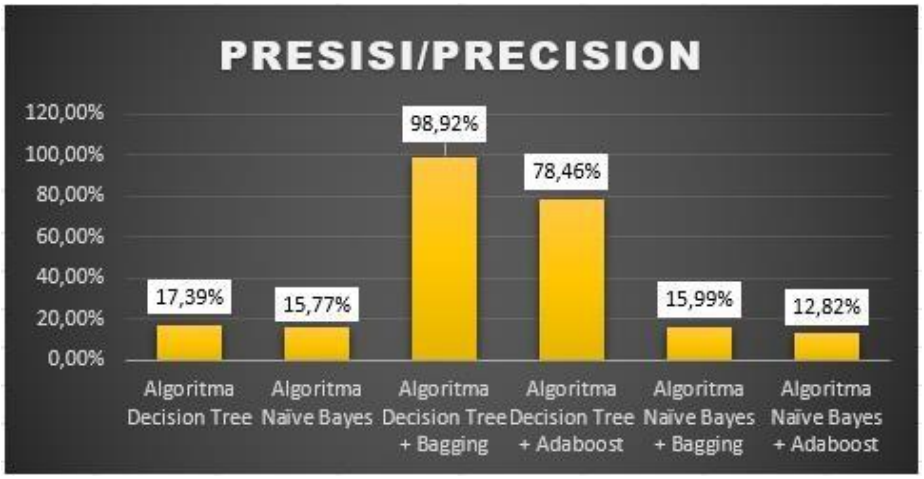
| Uji Validasi | Algoritma Decision Tree | Algoritma Naïve Bayes | Algoritma Decision Tree + Bagging | Algoritma Decision Tree + Adaboost | Algoritma Naïve Bayes + Bagging | Algoritma Naïve Bayes + Adaboost |
|---|---|---|---|---|---|---|
| Akurasi | 93,07% | 87,14% | 96,91% | 96,58% | 87,22% | 94,56% |
| Presisi | 17,39% | 15,77% | 98,92% | 78,46% | 15,99% | 12,82% |
| Recall/ Sensitivitas | 11,24% | 37,75% | 36,95% | 40,96% | 38,15% | 2,01% |
| Spesifisitas | 97,26% | 89,67% | 99,98% | 99,42% | 89,73% | 99,30% |

Berdasarkan nilai akurasi, presisi, *recall*/sensitivitas, dan spesifisitas pada Tabel 4.13 maka dapat digambarkan menggunakan grafik pada Gambar 4.19 – 4.21.



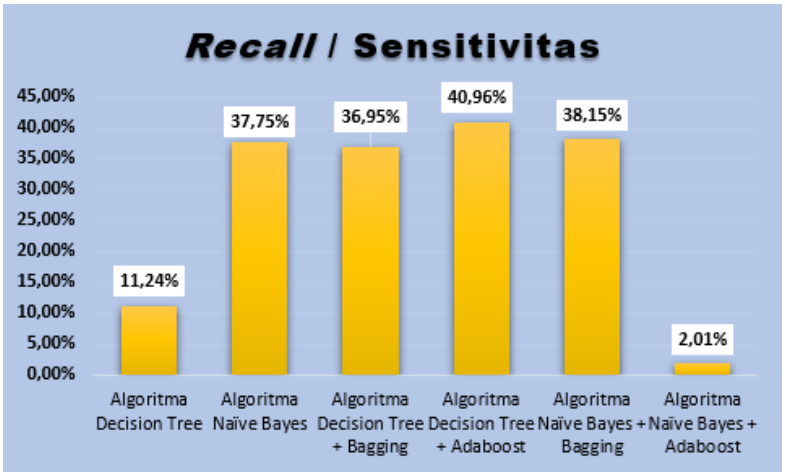
Gambar 4.19 Perbandingan Nilai Akurasi

Berdasarkan Gambar 4.19 diatas untuk nilai akurasi tertinggi yaitu algoritma Decision Tree + Bagging dengan nilai 96,91%. Sedangkan nilai akurasi terendah yaitu algoritma Naïve Bayes dengan nilai 87,14%. Selanjutnya, perbandingan nilai presisi/*precision* dapat dilihat pada Gambar 4.20 berikut.




Gambar 4.20 Perbandingan Nilai Presisi/Precision

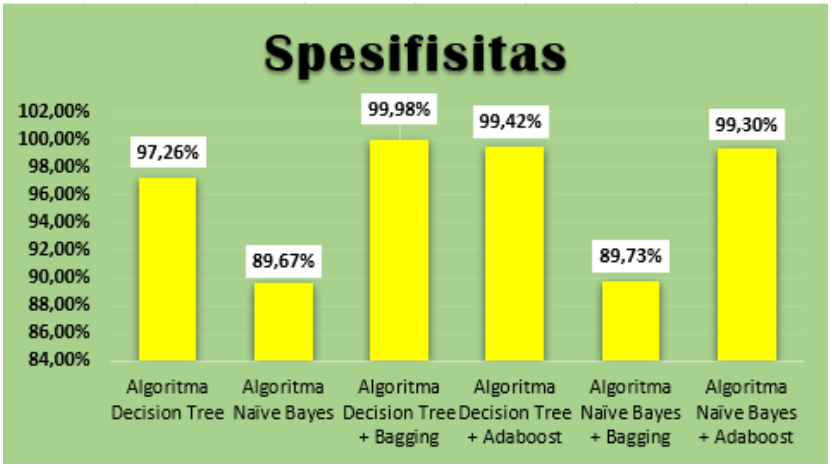
Dilihat dari gambar 4.20 diatas maka nilai presisi tertinggi dimiliki oleh algoritma Decision Tree yang dikombinasikan dengan teknik bagging yaitu sebesar 98,92%. Sedangkan nilai presisi terendah yaitu algoritma Naïve Bayes dan Adaboost dengan nilai 12,82%. Berikutnya yaitu perbandingan nilai *recall*/sensitivitas yang digambarkan pada grafik Gambar 4.21 berikut.




Gambar 4.21 Perbandingan Nilai Recall / Sensitivitas

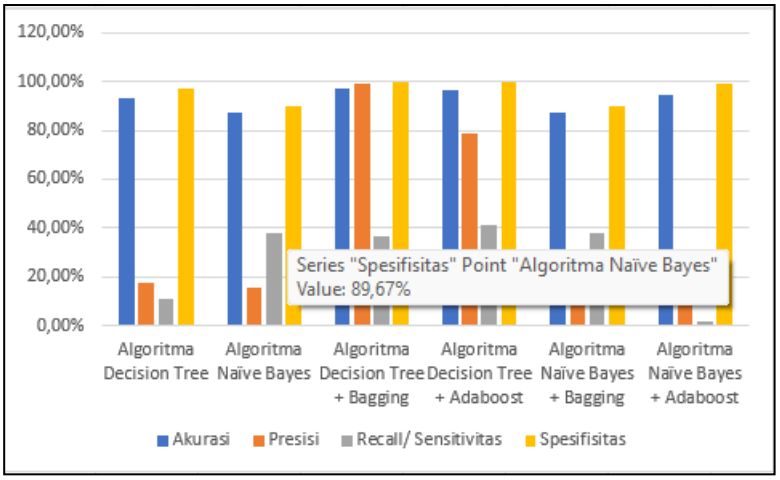
Berdasarkan Gambar 4.21 diatas dapat disimpulkan bahwa nilai *recall*/sensitivitas tertinggi adalah algoritma Decision Tree yang dikombinasikan dengan Adaboost sebesar 40,96%. Sedangkan nilai *recall*

terendah yaitu algoritma Naïve Bayes + Adaboost hanya 2,01%. Terakhir yaitu perbandingan nilai spesifisitas dapat dilihat pada grafik Gambar 4.22 berikut.

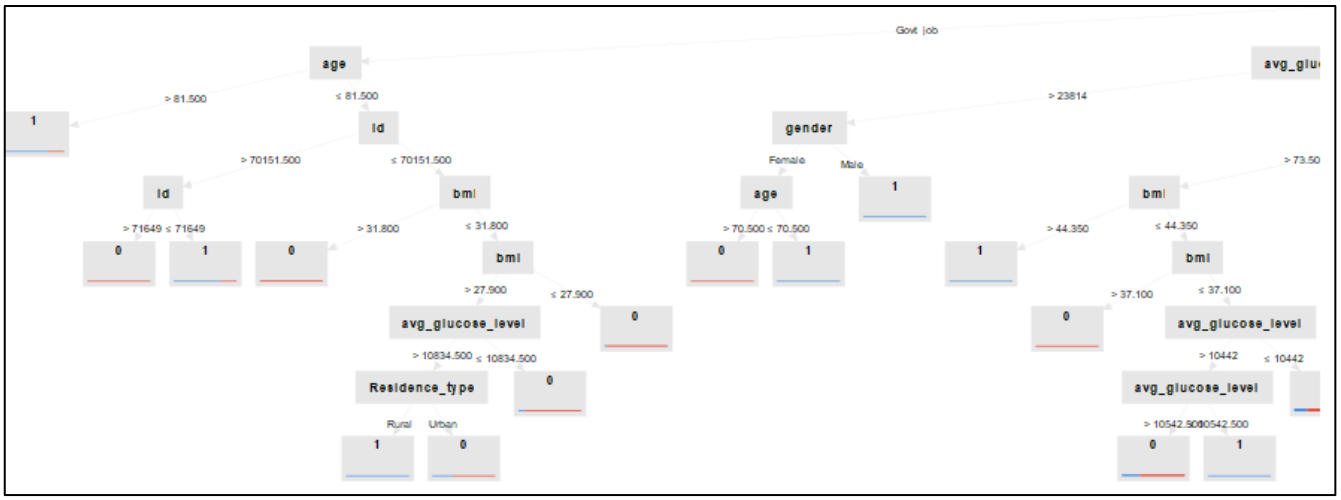


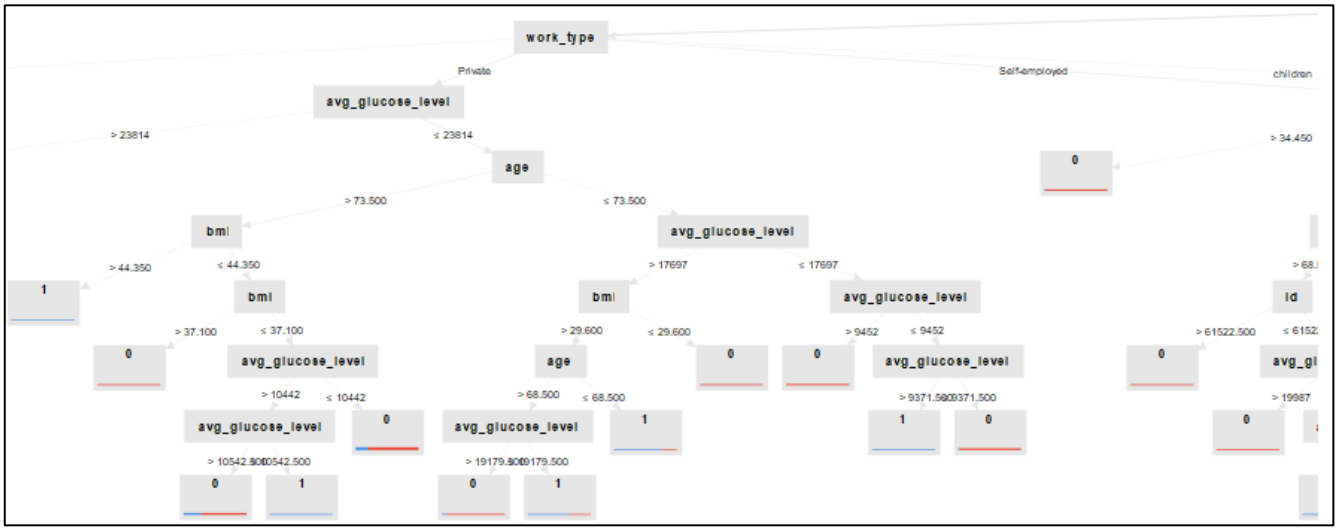

Gambar 4.22 Perbandingan Nilai Spesifisitas

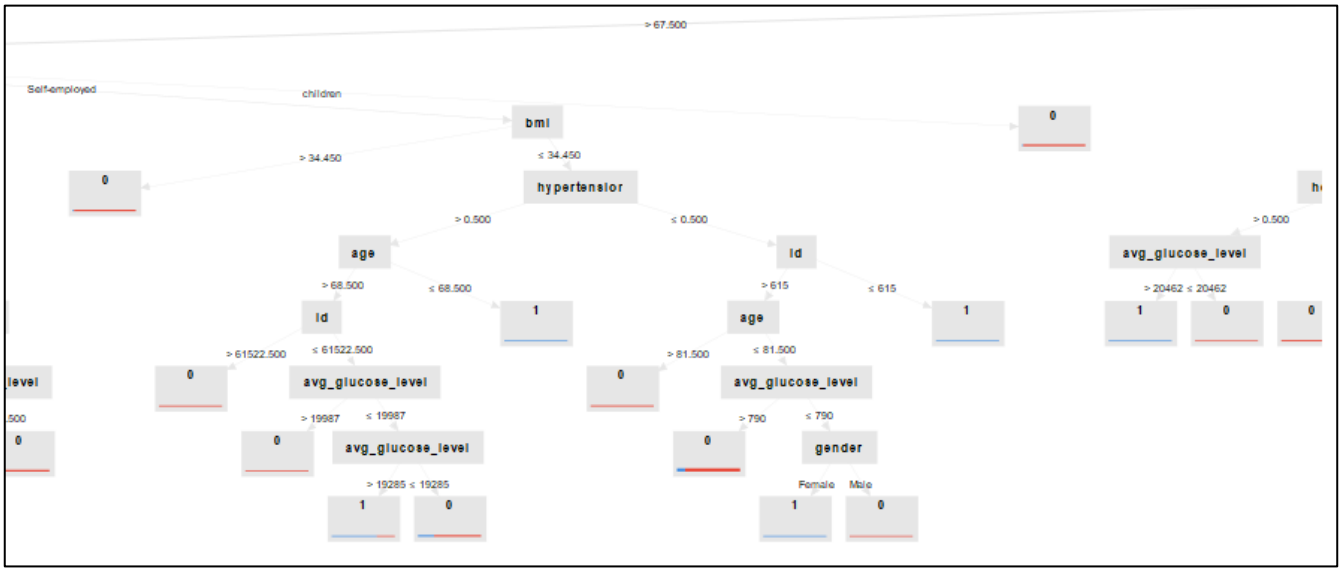
Berdasarkan Gambar 4.22 diatas dapat dilihat untuk nilai spesifisitas tertinggi diraih oleh algoritma Decision Tree + Bagging mencapai 99,98% dan nilai terendah yaitu algoritma Naïve Bayes sebesar 89,67%. Untuk perbandingan nilai keseluruhan dari masing-masing skenario pengujian algoritma dapat dilihat pada grafik Gambar 4.23 dibawah ini.




Gambar 4.23 Grafik Perbandingan Akurasi, Presisi, *Recall*/Sensitivitas dan Spesifisitas dari Keenam Skenario Pengujian Algoritma




Gambar 4.24 Contoh Tree Hasil Model Terbaik (Decision Tree + Bagging)

Berdasarkan Tabel 4.13 dan Gambar 4.23 diatas, hasil perbandingan pengujian pada *Dataset Stroke Prediction* menggunakan algoritma decision tree mendapatkan nilai akurasi, presisi dan spesifisitas yang tinggi dibanding dengan algoritma naïve bayes. Namun, untuk nilai sensitivitas lebih tinggi naïve bayes daripada decision tree. Kemudian pada skenario ketiga yaitu kombinasi algoritma decision tree dan teknik bagging menghasilkan nilai akurasi sebesar 96,91% dimana terdapat peningkatan sebesar 3,84% dari pengujian algoritma decision tree sebelum dikombinasi dengan metode bagging. Selain itu terjadi peningkatan yang sangat signifikan sebesar 81,53% pada nilai presisi. Untuk nilai sensitivitas juga mengalami peningkatan sebesar 25,71% begitupun dengan spesifisitas yang meningkat 2,72%. Pada skenario keempat yaitu kombinasi algoritma decision tree dengan metode adaboost menghasilkan nilai akurasi yang lebih tinggi daripada hanya menggunakan algoritma decision tree saja. Kenaikan akurasi sebesar 3,51% dari 93,07% menjadi 96,58%.

Namun, jika dibandingkan dengan skenario ketiga yaitu kombinasi decision tree dan bagging maka nilai akurasi, presisi dan spesifisitas lebih kecil tetapi nilai sensitivitasnya lebih besar. Dapat disimpulkan bahwa teknik bagging memberikan dampak yang lebih besar dalam meningkatkan kinerja algoritma decision tree jika dibandingkan metode adaboost. Selanjutnya, pada skenario kelima yaitu kombinasi algoritma naïve bayes dengan teknik bagging menghasilkan nilai akurasi, presisi, sensitivitas dan spesifisitas yang lebih tinggi daripada hanya menggunakan algoritma naïve bayes saja. Namun, kenaikannya tidak begitu signifikan, rata – rata kenaikannya dibawah 1%. Sehingga dapat disimpulkan teknik bagging tidak terlalu signifikan jika di kombinasi dengan algoritma naïve bayes. Sementara pada skenario keenam yaitu kombinasi algoritma naïve bayes dengan metode adaboost menghasilkan nilai akurasi dan spesifisitas yang tinggi daripada hanya menggunakan algoritma naïve bayes saja. Tetapi, dalam hal presisi dan sensitivitas, algoritma naïve bayes menunjukkan peningkatan nilai yang lebih tinggi setelah dikombinasikan dengan metode

adaboost. Namun, jika dibandingkan dengan skenario kelima yaitu kombinasi naïve bayes dan teknik bagging, maka nilai akurasi dan spesifisitas lebih tinggi menggunakan metode adaboost. Tetapi, untuk nilai presisi dan sensitivitas lebih tinggi menggunakan teknik bagging daripada adaboost.

Meskipun keduanya sama-sama memberikan pengaruh terhadap kinerja algoritma naïve bayes, namun peningkatan yang diperoleh dari metode bagging masih kurang signifikan dibandingkan menggunakan metode adaboost. Dari hasil pengujian dan analisis secara keseluruhan, dapat disimpulkan bahwa teknik bagging dan adaboost memiliki dampak yang cukup besar terhadap peningkatan kinerja algoritma decision tree, yakni melebihi 3,5%. Sedangkan pada algoritma naïve bayes hanya metode adaboost yang memberikan pengaruh yang cukup besar terhadap peningkatan kinerja algoritma naïve bayes yaitu mencapai 7,42%. Sementara itu, teknik bagging tidak memberikan peningkatan yang signifikan terhadap kinerja algoritma Naïve Bayes, dengan peningkatan yang kurang dari 1%. Oleh karena itu, dilihat dari keenam skenario pengujian tersebut penerapan teknik bagging pada algoritma decision tree terbukti lebih unggul dibandingkan metode adaboost. Tetapi, pada algoritma naïve bayes penerapan metode adaboost terbukti lebih unggul dibandingkan metode bagging.