# Prediksi Penyakit Jantung Dengan Menggunakan Algoritma XgBoost dan Randomized Search Optimizer

**TUGAS AKHIR**

Sebagai syarat untuk memperoleh gelar sarjana S-1 di Program Studi Informatika, Jurusan Teknik Informatika, Fakultas Teknik Industri, Universitas Pembangunan Nasional "Veteran" Yogyakarta


Disusun oleh:

Reo Sahobby

123170067



**PROGRAM STUDI INFORMATIKA**

**JURUSAN TEKNIK INFORMATIKA**

**FAKULTAS TEKNIK INDUSTRI**

**UNIVERSITAS PEMBANGUNAN NASIONAL "VETERAN"**

**YOGYAKARTA**


**2021**

# BAB I
# PENDAHULUAN

## 1.1.Latar Belakang

Dunia kesehatan akhir-akhir ini sedang ramai dibicarakan karena kemunculan virus baru di tahun 2019 yang bernama *corona virus* atau sering disebut dengan istilah covid 19. Virus tersebut menyerang pada bagian pernafasan atau paru-paru manusia. Nanum, selain virus yang sedang *booming* tersebut, kita juga harus memperdulikan tentang kesehatan jantung yang tidak kalah pentingnya. Jantung merupakan organ dalam manusia yang fungsinya sangatlah penting yaitu untuk mengedarkan darah yang berisi oksigen dan nutrisi ke seluruh tubuh manusia dan untuk mengangkut sisa hasil metobolisme tubuh, sehingga tubuh dapat bekerja dengan optimal. Akan sangat fatal apabila di dalam organ jantung terdapat gangguan, seperti penyumbatan pembuluh darah dan lain-lain. Sehingga menyebabkan jantung tidak dapat bekerja dan dapat menyebabkan kematian.

Berdasarkan data dari WHO terdapat sebanyak 7,3 juta penduduk di seluruh dunia meninggal karena penyakit jantung. Penyakit jantung adalah penyakit yang menyerang pada organ jantung yang berkaitan dengan pembuluh darah, contohnya adalah pembuluh darah di organ jantung yang tersumbat. Penyakit ini menyerang pada pembuluh darah arteri karena tejadi proses *arterosklerosis* pada dinding arteri yang menyebabkan penyempitan (Marleni & Alhabib, 2017). Penyakit jantung juga bisa disebut dengan istilah *suddent death* (Widiastuti et al., 2014). Karena penyakit jantung tersebut sering kali tidak menimbulkan gejala, namun tiba-tiba pembuluh darah di jantung yang tersumbat tidak dapat memompa darah dan menyalurkannya ke seluruh tubuh, sehingga dapat menyebabkan kematian.

Penyebab penyakit jantung dapat berupa beberapa faktor kebiasaan hidup, berbagai penelitian sudah banyak dilakukan untuk menemukan penyebab dari peyankit jantung tersebut. Diantaranya adalah kebiasaan merokok, pola makan yang tidak sehat, jarang berolahraga, dan lain-lain (Marleni & Alhabib, 2017). Dan tentunya kebiasaan dan faktor penyebab penyakit jantung tersebut sangat bergantung sesuai dengan umur, jenis kelamin, kondisi geografis atau tempat tinggal, tingkat kolesterol, obesitas dan kecenderungan *stress* seseorang tersebut (Anwar, 2004). Sedangkan menurut (Zulaekah et al., 2009), faktor-faktor penyebab terjadinya penyakit jantung juga dipengaruhi oleh kondisi tubuh dan nutrisi seseorang, seperti asupan lemak yang tinggi dan kurangnya tubuh melakukan aktivitas fisik seperti olahraga, sehingga jantung tidak terbiasa dengan aktivitas tubuh yang berat. Kadar kolesterol darah yang tinggi dipengaruhi oleh kebiasaan mengonsumsi makanan yang berlemak, semakin banyak mengonsumsi makanan yang berlemak, peluang untuk menaikkan kadar kolesterol di dalam darah akan semakin tinggi dan akan menurunkan kadar *high density lipoprotein*. Kandungan HDL yang rendah di dalam darah akan mempengaruhi rasio total kolesterol darah dan HDL, semakin tinggi angka rasio total kolesterol dan HDL, maka akan semakin tinggi pula risiko terjadinya penyakit jantung (Zulaekah et al., 2009).

Gejala penyakit jantung yang sering ditemui adalah penderita terkadang merasa sesak napas, kondisi fisik penderita yang mudah lelah, penderita mungkin saja

mengalami ganguan seksual, dan penderita sering merasakan nyeri dada (Nuraeni, 2016). Selain itu seseorang yang menderita penyakit jantung juga memiliki gejala non fisik seperti sering merasa cemas, ketakutan berlebihan, dan sering merasakan depresi. Namun, selain penderita penyakit jantung yang dapat merasakan gejala dari penyakit yang dialaminya, terdapat juga penderita penyakit jantung yang tidak merasakan gejala apa-apa. Selama 50 tahun terakhir, semakin banyak penderita penyakit jantung coroner yang penderitanya tidak merasakan gejala apa-apa, baik gejala fisik maupun gejala non fisik (Zahrawardani et al., 2013). Kondisi tersebut dibilang lebih berbahaya daripada kondisi penderita penyakit jantung yang dapat merasakan gejala, karena penderita tidak akan menyadari kondisi tubuh mereka dan tidak melakukan pencegahan penyakit jantung yang sedang dialami. Karena menurut jurnal (Indrawati, 2014), terdapat hubungan antara kesadaran dan pengetahuan tentang penyakit jantung dan kondisi diri sendiri untuk dapat melakukan upaya pencegahan penyakit jantung.

Di Indonesia sendiri, penyakit jantung sering kali tidak dihiraukan oleh masyarakat, masyarakat masih belum terlalu sadar untuk mengubah pola hidup mereka menjadi pola hidup yang lebih sehat. Padahal jika masyarakat tahu memiliki pengetahuan yang cukup tentang penyakit jantung koroner dan faktor risikonya, maka akan mudah untuk melakukan pencegahan penyakit tersebut (Zahrawardani et al., 2013). Angka kematian karena penyakit jantung di Indonesia meningkat, dari yang tadinya sebesar 41,7% pada tahun 1995, menjadi sebesar 59,5% pada tahun 2007 (Depkes RI, 2009). Di rumah sakit Siti Khadijah di Palembang, tercatat pada tahun 2015 jumlah pasien di poli jantung mengalami peningkatan pasien mencapai 354 pasien, dan pada tahun 2016 terdapat 274 pasien penderita penyakit jantung di rumah sakit tersebut (Marleni & Alhabib, 2017). Di Jawa Tengah, dari laporan puskesmas di daerah tersebut terdapat 26,38 kasus penyakit jantung dari 1.000 penduduk di daerah tersebut (Zahrawardani et al., 2013). Oleh karena itu, di Indonesia penyakit jantung juga harus tetap diwaspadai.

Proses pendeteksian apakah seseorang tersebut terkana penyakit jantung atau tidak dapat dilakukan dengan cara melakukan konsultasi kepada dokter spesialis jantung yang nantinya akan dilakukan pemeriksaan laboratorium dan dikonsultasikan oleh dokter spesialis jantung (Wibisono & Fahrurozi, 2019). Namun cara tersebut tidaklah efektif, selain memakan waktu yang lama karena proses pemeriksaan, menunggu hasil pemeriksaan, dan konsultasi tentunya memakan waktu yang lama, juga karena memakan biaya yang cukup tinggi. Oleh karena itu perlu dilakukan pendeteksian penyakit jantung secara digital supaya dapat meningkatkan efektifitas kerja. Banyak penelitian yang sudah menciptakan pendeteksian penyakit jantung secara digital, yaitu dengan menggunakan data-data hasil rekam jantung yang ada, yang nantinya dipelajari pola-pola datanya dan akan menghasilkan prediksi, berdasarkan data tersebut apakah seseorang ini berpotensi menderita penyakit jantung atau tidak. Teknik yang digunakan dalam melakukan prediksi tersebut dinamakan teknik klasifikasi. Klasifikasi adalah jenis analisis data yang digunakan untuk memprediksi label kelas dari data tersebut (Annisa, 2019).

Dalam klasifikasi terdapat beberapa teknik atau algoritma yang dapat dilakukan untuk mengerjakan klasifikasi, diantaranya adalah dapat menggunakan algoritma KNN, algoritma *Naïve Bayes*, algoritma *Support Vector Machine*, algoritma *Decision Tree*,

algoritma *Random Forest*, dan lain-lain. Dalam kasus prediksi penyakit jantung ini, penelitian-penelitian sebelumnya telah banyak dilakukan dengan menggunakan berbagai algoritma klasifikasi yang ada. Diantaranya adalah penelitian yang dilakukan oleh Retnasari dan Rahmawati, yang melakukan penelitian dengan menggunakan algoritma *Naïve Bayes* dan algoritma C4,5. Penelitian tersebut dilakukan dengan menggunakan 270 data yang bersumber dari UCI *Machine Learning Repository* dengan jumlah *features* yaitu 13, penelitian tersebut dilakukan dengan menggunakan *rapid mider* dan *confusion matrix* untuk menghitung akurasi masing-masing algoritma. Hasil dari penelitian yang dilakukan tersebut menunjukkan bahwa algoritma *Naïve Bayes* lebih baik dengan mendapatkan nilai akurasi sebesar 86,67% dan algoritma C4.5 mendapat akurasi sebesar 83,70% (Retnasari & Rahmawati, 2017). Penelitian selanjutnya yang dilakukan oleh Ardea dan Achmad, penelitian tersebut dilakukan untuk mencari algoritma terbaik dengan cara membandingkan masing-masing hasil dari algoritma tersebut. Algoritma yang dibandingkan di dalam penelitian tersebut adalah algoritma *Naïve Bayes*, algoritma *Random Forest*, algoritma *Decision Tree*, dan algoritma *K-Nearest Neighbohr*. Hasil dari penelitian tersebut untuk masing-masing algoritma dihitung dengan menggunakan *confusion matrix* dan didapat hasil akurasi untuk masing-masing algoritma sebagai berikut. Algoritma *Random Forest* memiliki nilai akurasi tertinggi dengan 85,67%, kemudian algoritma *Naïve Bayes* dan algoritma *Decision Tree* memiliki nilai akurasi yang sama dengan nilai akurasi 80,33%, dan algoritma *K-Nearest Neighbor* memiliki nilai akurasi paling rendah yaitu 69,67%. Dengan hasil tersebut, algoritma yang terbaik adalah algoritma Random Forest (Wibisono & Fahrurozi, 2019). Selanjutnya penelitian yang dilakukan oleh Erwin Prasetyo dan Budi Prasetiyo, penelitian tersebut dilakukan dengan menerapkan teknik *bagging* pada algoritma C4.5 untuk melihat apakah teknik *bagging* dapat meningkatkan akurasi dari model klasifikasi yang dibuat. Data yang digunakan dalam penelitian tersebut adalah data *Heart Disease* yang diambil dari UCI *Machine Learning* sejumlah 300 data. Hasil dari penelitian tersebut membuktikan bahwa penerapan teknik *bagging* pada algoritma C4.5 dapat meningkatkan akurasi model yang dibuat dengan kenaikan yaitu 8,86% dengan hasil akurasi algoritma C4.5 sebesar 72,98% dan akurasi algoritma C4.5 yang dikombinasikan dengan teknik *bagging* adalah 81,84% (Prasetyo & Prasetiyo, 2020)

Dari berbagai macam algoritma yang sudah digunakan dalam penelitian-penelitian sebelumnya, tentunya masing-masing algoritma memiliki kelebihan dan kelemahan. Sebenarnya beberapa metode yang sudah digunakan dalam penelitian sebelumnya sudah menghasilkan nilai akurasi model yang baik, namun seringkali apabila model memiliki nilai akurasi yang terlalu tinggi, maka model akan terlalu fokus mempelajari data *training* sehingga nilai akurasi model sangatlah tinggi, namun pada saat dilakukan prediksi menggunakan data real yang belum pernah ditemui oleh model, hasil prediksi seringkali tidak tepat, kondisi tersebut dinamakan *overfitting*. Untuk mencegah terjadinya *overfitting* pada model yang sudah dibuat, perlu diterapkan teknik *regularization* untuk mengurangi *overfitting* pada model. Pada penelitian ini algoritma yang dipilih adalah menggunakan algoritma XgBoost. Algoritma XgBoost adalah algoritma *gradien boosting* yang dibuat dengan *tree base* yang dapat membuat *boosted*

*tree* secara efisien dan dapat dikerjakan secara paralel (Karo, 2020). Algoritma tersebut sudah memiliki operasi *regularization*, sehingga algoritma tersebut dapat mencegah terjadinya *overfitting*. Tujuan dari penelitian ini adalah untuk mengetahui apakah pendeteksian penyakit jantung dapat dilakukan dengan menggunakan algoritma XgBoost, dan bagaimana hasil akurasi dari model yang dibuat dengan algoritma tersebut.

## 1.2. Rumusan Masalah

Sesuai dengan uraian latar belakang yang sudah dijelaskan di atas, rumusan masalah dalam penelitian ini adalah sebagai berikut:

a. Penerapan algoritma klasifikasi XgBoost pada kasus prediksi penyakit jantung.
b. performa model yang dibuat dengan algoritma XgBoost dalam menyelesaikan permasalahan prediksi penyakit jantung.

## 1.3. Batasan Masalah

Batasan masalah yang ada di dalam penelitian ini adalah sebagai berikut:

a. Data yang digunakan dalam penelitian ini adalah data *Heart Disease* yang diambil dari UCI *Machine Learning*.
b. Metode algoritma klasifikasi yang digunakan dalam penelitian ini adalah menggunakan algoritma XgBoost.

## 1.4. Tujuan Penelitian

Tujuan yang ingin dicapai dari penelitian ini adalah sebagai berikut:

a. Menerapkan algoritma klasifikasi XgBoost dalam kasus prediksi penyakit jantung.
b. Mengetahui performa model yang dibuat dengan algoritma XgBoost dalam menyelesaikan permasalahan prediksi penyakit jantung.

## 1.5. Manfaat Penelitian

Manfaat yang ingin dicapai dari penelitian yang ingin dilakukan ini adalah sebagai berikut:

a. Manfaat bagi peneliti, peneliti dapat menerapkan ilmu yang didapat selama perkuliahan, dapat mengimplementasikan algoritma XgBoost untuk menyelesaikan permasalahan prediksi penyakit jantung.
b. Manfaat penelitian ini bagi industri kesehatan adalah, untuk kedepannya diharapkan mampu membantu dalam proses pendeteksian penyakit jantung supaya lebih efektif.

## 1.6. Tahapan Penelitian

Pada penelitian yang akan dilakukan ini, terdapat beberapa tahapan yang akan dilakukan yaitu sebagai berikut:

a. Study Literatur

   Tahap pertama yang dilakukan dalam penelitian ini adalah melakukan *study literature*. *Study literature* dilakukan untuk mencari referensi, penelitian sebelumnya, data yang akan digunakan, dan lain-lain. Study literature dapat dicari dari jurnal-jurnal yang membahas penelitian serupa.

b. Pengumpulan Data

   Tahap selanjutnya adalah melakukan pengumpulan data, data yang akan digunakan dalam penelitian ini adalah data sekunder, yaitu data *Heart Disease* yang bersumber dari UCI *Machine Learning Repository*.

c. Analisis Sistem
   Selanjutnya adalah melakukan analisis kebutuhan perangkat lunak yang ada dibuat di dalam penelitian ini.
d. Pembuatan Model *Machine Learning*
   Tahap selanjutnya adalah pembuatan model *machine learning*. Pada tahap ini dilakukan pembuatan model prediksi yang menggunakan algoritma dn teknik yang sudah dipilih.
e. Pengujian dan Evaluasi Model
   Setelah model prediksi *machine learning* sudah dibuat, tahap selanjutnya adalah memastikan model yang dibuat memiliki performa yang baik dalam menangani data. Apabila model dirasa belum maksimal, dapat dilakukan pembuatan model ulang dengan *hyper parameter* yang berbeda dan dilakukan pengujian lagi, diharapkan mendapat peningkatan performa.
f. Implementasi Perangkat Lunak
   Selanjutnya, setelah model yang dibuat dirasa memiliki performa yang bagus, model tersebut diimplementasikan dalam bentuk perangkat lunak yang bisa digunakan oleh pengguna. Dalam pembuatan perangkat lunak ini, menggunakan metodogi *waterfall*.
g. Pengujian dan Evaluasi Perangkat Lunak
   Setelah perangkat lunak selesai dibuat, dilakukan pengujian perangkat lunak untuk memastikan perangkat lunak yang dibuat berjalan normal tanpa ada kendala.
h. Kesimpulan dan Saran
   Setelah semua tahap dilakukan, didapatkan kesimpulan dari penelitian yang sudah dilakukan tentang bagaimana performa algoritma XgBoost dalam menangani kasus permasalahan yang dipilih.

## 1.7. Sistematika Penulisan

Penelitian ini disusun berdasarkan sistematika penulisan yang terdiri dari 5 bab yang terdiri dari:

**BAB 1 PENDAHULUAN**

Pada BAB I ini, membahas latar belakang penelitian ini dilakukan, rumusan masalah yang ada di dalam penelitian ini, batasan masalah, tujuan, dan manfaat penelitian ini dilakukan, serta sistematika penulisan laporan mengenai penelitian yang dilakukan.

**BAB II TINJAUAN PUSTAKA**

Dalam BAB II ini, berisi landasan teori mengenai obyek penelitian dan metode yang akan dilakukan di dalam penelitian ini, kemudian juga membahas penelitian-penelitian serupa yang sudah dilakukan sehingga menjadi referensi penulis dalam mengadakan melakukan penelitian ini.

**BAB III METODE PENELITIAN**

Pada BAB III ini berisi penjelasan tentang metode yang akan digunakan oleh penulis di dalam melakukan penelitian ini. Metode-metode yang dipilih nantinya akan digunakan untuk menyelesaikan permasalahan pada kasus yang sedang diteliti, yaitu prediksi penyakit jantung.

**BAB IV HASIL DAN PEMBAHASAN**

Pada BAB IV ini, berisi pemaparan dan penjelasan hasil dari tahapan demi tahapan penelitian yang sudah dilakukan oleh penulis dengan menggunakan metode yang sudah dijelaskan pada bab sebelumnya. Penjelasan hasil penelitian akan berisi evaluasi performa model yang sudah dibuat dengan menggunakan algoritma yang dipilih.

**BAB V PENUTUP**

Bab ini akan berisi kesimpulan hasil dari penelitian yang sudah dilakukan oleh penulis. Kemudian penulis juga menambahkan kekurangan dari penelitian yang sudah dilakukan ditambahkan dengan saran yang bisa dilakukan pada penelitian yang akan datang, dapat berupa saran perbaikan data ataupun saran mengenai perbaikan metode supaya penelitian yang akan datang dapat menghasilkan hasil yang lebih maksimal.

# BAB II
# KAJIAN LITERATUR

## 2.1.Tinjauan Studi

Penyakit jantung menjadi tantangan tersendiri di dunia bagi indsutri pelayanan kesehatan saat ini (Prasetyo & Prasetiyo, 2020). Dalam satu dekade terkahir, penyakit ini merupakan penyakit yang paling utama menjadi penyebab kematian di seluruh dunia (Jothikumar & Siva Balan, 2016). Maka, diperlukan sistem yang dapat menangani pendeteksian penyakit jantung pada penderita secara akurat dan dengan biaya yang terjangkau (Wibisono & Fahrurozi, 2019). Oleh karena itu, penelitian tentang penyakit jantung telah banyak dilakukan, penelitian tersebut dilakukan dengan menggunakan beberapa teknik dan algoritma untuk mendapatkan hasil prediksi yang semaksimal mungkin. Salah satu penelitian yang sudah dilakukan adalah penelitian yang dilakukan oleh Erwin Prasetyo dan Budi Prasetyo(Prasetyo & Prasetiyo, 2020), penelitian tersebut dilakukan menggunakan datasets *heart disease* dengan data sebanyak 303, dengan jumlah kolom yang digunakan 13 kolom. Penelitian tersebut bertujuan untuk membandingkan performa algoritma C4.5 yang dikombinasikan dengan teknik *bagging*, dan algoritma C4.5 murni tanpa penambahan teknik apapun. Penelitian tersebut dilakukan dengan menggunakan bahasa pemrograman python dengan bantuan beberapa perpustakaan seperti *numpy*, *sklearn*, dan *pandas*. Untuk mengukur performa algoritma, penelitian tersebut menggunakan *confusion matrix*. Untuk melakukan validasi hasil performa algoritma yang diuji dalam penelitian tersebut menggunakan *k-fold cross validation* dengan nilai *k*=10 .Hasil dari penelitian yang dilakukan ditampilkan menggunakan tabel *confusion matrix*, dengan hasil akurasi dari algoritma C4.5 adalah 72,98% dan akurasi algoritma C4.5 yang dikombinasikan dengan teknik *bagging* adalah 81,84%. Dengan hasil tersebut, didapatkan kesimpulan bahwa teknik *bagging* yang dilakukan dengan algoritma C4.5 dapat meningkatkan akurasi model, dalam kasus tersebut meningkatkan sebanyak 8,86%.

Selanjutnya, penelitian yang dilakukan oleh Pandito Dewa Putra dan Dian Palupi Rini pada tahun 2019 (Putra & Rini, 2019). Penelitian tersebut dilakukan untuk mengetahui perbandingkan beberapa algoritma klasifikasi seperti *naïve bayes*, *support vector machine*, C4.5, *logistic regression*, dan *back propagation* dalam melakukan klasifikasi untuk melakukan prediksi penyakit jantung. Datasets yang digunakan dalam penelitian tersebut adalah *Statelog Heart Disease Datasets* yang berasal dari UCI *machine learning* yang dapat diunduh secara online. Datasets tersebut berjumlah 270 data dengan kolom yang digunakan adalah 13 kolom seperti *age*, *sex*, *chest pain*, dan lain-lain. Untuk melakukan validasi, dalam penelitian tersebut menggunakan *cross validation* dan nilai yang akan dihitung untuk mengukur performa algoritma adalah akurasi, presisi, dan *recall*. Tahapan yang dilakukan di dalam penelitian tersebut meliputi input datasets, melakukan *preprocessing*, pembuatan model klasifikasi, proses validasi menggunakan *cross validation*, dan pengukuran performa algoritma. Hasil dari penelitian tersebut adalah akurasi algoritma *naïve bayes* mendapatkan nilai tertinggi yaitu 84,07%. Kemudian algoritma dengan presisi tertinggi adalah algoritma *naïve bayes* dengan nilai presisi adalah 86.16%. Selanjutnya untuk pengukuran *recall*, algoritma

yang memiliki *recall* tertinggi adalah *support vector machine* dengan nilai *recall* mencapai 94,67%. Dari hasil penelitian yang dilakukan tersebut didapatkan kesimpulan bahwa algoritma *naïve bayes* tercatat memiliki performa yang lebih baik dari algoritma lainnya baik dari segi akurasi dan presisi.

Selanjutnya penelitian yang dilakukan oleh Ardea Bagas Wibisono dan Achmad Fahrurozi pada tahun 2019 (Wibisono & Fahrurozi, 2019). Penelitian tersebut dilakukan untuk mengimplemantasikan beberapa metode klasifikasi seperti *Naive Bayes*, *K-Nearest Neighbor*, *Decision Tree*, *Random Forest*, dan *Support Vector Machine* untuk kasus pengenalan penyakit jantung koroner. Dengan hasil dari pengukuran yang didapat adalah akurasi, *recall*, dan presisi. Datasets yang digunakan di dalam penelitian tersebut adalah *Cleveland Heart Disease* yang berisi data rekam jantung sejumlah 300 data, dengan 14 kolom. Parameter yang ada dalam datasets tersebut berjumlah 13, 1 kolom sebagai data target dari hasil klasifikasi. Parameter yang ada di dalam datasets tersebut seperti *age*, *sex*, *chol*, *fbs*, *restecg*, *thalach*, dan lain-lain. Dalam penelitian tersebut dilakukan beberapa tahapan diantaranya adalah proses *import* datasest yang digunakan, kemudian melakukan pembagian data menjadi *data training* dan *data testing*, dengan perbandingan *data training* dan *data testing* adalah 80 banding 20. Tahap selanjutnya adalah pembuatan model klasifikasi dengan melakukan *training* pada *data training* menggunakan algoritma yang sudah ditentukan. Selanjutnya, melakukan pengujian dan validasi model klasifikasi yang sudah dibuat. Proses pengujian dilakukan menggunakan *data testing*, dan untuk melakukan validasi terhadap performa model klasifikasi dilakukan dengan menggunakan *Cross Validation* dengan nilai *k*=5. Kemudian tahap terakhir adalah melakukan perhitungan untuk mengukur performa algoritma. Dari tahapan yang dilakukan dalam penelitian tersebut, dari hasil pengujian yang dilakukan terhadap algoritma yang diuji mendapatkan hasil bahwa akurasi tertinggi didapatkan pada algoritma *Random Forest* dengan nilai akurasi mencapai 85,67%. Algoritma *Naïve Bayes* dan algoritma *Decision Tree* mendapat hasil akurasi yang sama, yaitu 80,33%. Algoritma *K-Nearest Neighbor* mendapat akurasi sebesar 69,67%. Dari penelitian tersebut dapat disimpulkan bahwa algoritma *Random Forest* memiliki performa yang paling baik dalam melakukan klasifikasi dengan menggunakan data pasien jantung coroner.

Kemudian, penelitian serupa juga dilakukan oleh Riski Annisa pada tahun 2019 (Annisa, 2019). Penelitian tersebut dilakukan untuk mencari algoritma terbaik dengan cara membandingkan algoritma *Decision Tree*, *Naïve Bayes*, *K-Nearest Neighbor*, *Random Forest*, dan *Decision Stump* menggunakan uji parametrik dengan *t-test*. Datasets yang digunakan untuk melakukan penelitian tersebut adalah datasets laki-laki penderita penyakit jantung yang terdiri dari 8 atribut, datasets tersebut tersedia secara online pada UCI *machine learning*. Dalam penelitian tersebut dilakukan validasi dengan menggunakan *Cross Validation,* dan menggunakan nilai *k=10 fold cross validation* yang berarti *data training* akan dipecah menjadi 10 bagian, nantinya masing-masing bagian akan menjadi *data testing*. Untuk mengukur kinerja algoritma yang dibandingkan, dalam penelitian tersebut pengujian dilakukan dengan menggunakan uji t (*t-test*). Tahapan yang dilakukan di dalam pengujian tersebut adalah pembagian datasets menjadi dua bagian,

yaitu *data training* dan *data testing*. Kemudian diterapkan evaluasi menggunakan AUC atau *Area under Curve*. Sedangkan untuk hasil dari akurasi algoritma yang didapatkan dapat dilihat menggunakan kurva *Receiver Operating Characteristic* (ROC) dan dalam bentuk *confusion matrix*. Kemudian, penelitian tersebut juga dilakukan uji t atau *t-test* yaitu untuk membandingkan hubungan antara dua variabel, variabel respon dan variabel *predictor* yang digunakan. Hasil penelitian tersebut nilai akurasi tertinggi didapatkan pada algoritma *Random Forest* dengan akurasi sebesar 80,38%. Berdasarkan pengukuran menggunakan AUC, algoritma *Random Forest* juga tergolong ke dalam *good classification*. Sedangkan untuk algoritma *K-Nearest Neighbor*, C4.5, dan algoritma *Decision Stump* tergolong ke dalam *fair classification*. Sedangkan untuk hasil *t-test* diketahui algoritma C4.5 dan *Naïve Bayes* tidak menunjukan perbedaan yang signifikan, algoritma *Naïve Bayes* tidak menunjukan perbedaan yang signifikan dengan algoritma *Random Forest* dan *Decision Stump*. Algoritma C4.5 memiliki perbedaan yang signifikan dengan algoritma *Random Forest* dan *Decision Stump*. Algoritma *K-Nearest Neighbor* berbeda secara signifikan dengan algoritma lain yang berarti algoritma K-NN kurang baik diimplementasikan dalam kasus dan datasets tersebut.

Selanjutnya, penelitian yang dilakukan oleh Mei Lestari pada tahun 2014 (Lestari, 2014). Penelitian tersebut dilakukan menggunakan algoritma *K-Nearest Neighbor* (K-NN) yang diterapkan untuk mendeteksi penyakit jantung. Dalam penelitian tersebut datasest yang digunakan adalah data yang bersumber dari UCI yang berjumlah 110 data dengan jumlah atribut sebanyak 13 atribut yang mewakili setiap parameter data tersebut seperti *age, sex, tresting blood pressure, old peak*, dan lain-lain. Tahapan yang dilakukan di dalam penelitian tersebut adalah melakukan *import* datasets yang akan digunakan yaitu berjumlah 110 data. Selanjutnya, membagi datasets menjadi dua bagian yaitu *data training* dan *data testing*, *data training* yang digunakan berjumlah 100 data, dan *data testing* yang digunakan berjumlah 10 data. Kemudian dilanjutkan proses membuat model klasifikasi menggunakan algoritma K-NN dan menentukan nilai *k* di dalam algoritma K-NN tersebut dengan nilai *k*=9. Kemudian, proses training dan pengujian dilakukan menggunakan *confusion matrix* dan kurva ROC untuk mengukur performa algoritma. Hasil dari penelitian tersebut adalah algoritma K-NN yang digunakan menggunakan nilai *k*=9, maka proses yang di dalam algoritma tersebut akan mengecek 9 buah tetangga terdekat untuk masing-masing data, nantinya akan digunakan untuk menentukan klasifikasi. Berdasarkan hasil pengujian menggunakan *data testing* yang berjumlah 10 data, akurasi yang didapatkan dari algoritma K-NN sebesar 70%. Sedangkan metode pengukuran lainnya dilakukan menggunakan kurva ROC dan AUC. Dari pengujian yang dilakukan, nilai AUC yang didapatkan dari algoritma tersebut adalah 0.875 yang berarti dapat dikatakan algoritma KNN tergolong algoritma yang baik untuk melakukan klasifikasi deteksi penyakit jantung.

Penelitian selanjutnya yang masih membahas tentang penyakit jantung adalah penelitian yang dilakukan oleh Abdul Rohman, Vincent Suhartono dan Catur Supriyadi pada tahun 2017 (Rohman et al., 2017). Penelitian tersebut membahas tentang prediksi penyakit jantung yang dilakukan menggunakan algoritma *Decision Tree* atau C4.5 namun dikombinasikan dengan metode *Adaboost*. Tujuan penelitian tersebut adalah

melakukan kombinasi algoritma dan metode untuk mengoptimalkan atribut-atribut yang dimiliki oleh datasets, dan diharapkan dengan menerapkan metode *Adaboost* dapat meningkatkan performa algoritma dalam melakukan klasifikasi. Datasets yang digunakan di dalam penelitian tersebut adalah data gabungan dari datasets Cleveland yang berjumlah 303 data, datasets Statlog yang berjumlah 270 data, dan datasets Hungaria yang berjumlah 294 data. Sehingga jumlah kseseluruhan data yang digunakan berjumlah 867 data, dengan rincian 364 data masuk ke dalam klasifikasi sakit, dan 503 data masuk ke dalam klasifikasi sehat. Semua datasets yang digunakan di dalam penelitian tersebut bersumber dari UCI *machine learning*. Tahapan yang dilakukan di dalam penelitian tersebut sama seperti tahapan yang dilakukan pada penelitian sebelumnya. Mulai dari import data, *preprocessing* data, pembuatan model, pengujian dan pengukuran performa algoritma yang digunakan. Pada tahap *preprocessing*, untuk mendapatkan data yang berkualitas dilakukan seleksi data terlebih dahulu. Proses *preprocessing* data mendapatkan hasil yaitu data yang sudah berkualitas sebanyak 567 data, dengan data yang tergolong dalam klasifikasi sakit 257 data, dan sebanyak 310 data tergolong ke dalam klasifikasi sehat. Dalam pembuatan model algoritma, dilakukan validasi dengan menggunakan *K-Fold Cross Validation* dengan nilai *k*=10. Dan dalam pengujian model, pengujian dilakukan menggunakan kurva *Area Under Curve* (AUC). Hasil penelitian tersebut algoritma C4.5 mendapatkan nilai akurasi sebesar 86,59% dengan nilai AUC adalah 0,957. Sedangkan hasil pengujian algoritma C4.5 yang dikombinasikan dengan metode *Adaboost* mendapat akurasi sebesar 92,24% dan nilai AUC sebesar 0.982. Dengan demikian, dapat disimpulkan bahwa metode *Adaboost* yang dikombinasikan dengan algoritma C4.5 dapat meningkatkan performa algoritma yang signifikan dibandingkan algoritma C4.5 murni tanpa penambahan metode apapun.

Penelitian selanjutnya adalah penelitian yang dilakukan oleh Nur Aeni Widiastuti, Stefanus Santosa, dan Catur Supriyanto pada tahun 2014 (Widiastuti et al., 2014). Penelitian tersebut membahas tentang klasifikasi penyakit jantung yang dilakukan dengan algoritma *Naïve Bayes* berbasis *Particle Swarm Optimization*. PSO atau *Particle Swarm Optimization* dipilih dalam penelitian tersebut karena mudah diterapkan dan terdapat beberapa parameter untuk menyesuaikan. Pada penelitian tersebut datasest yang digunakan berjumlah 300 data, dengan pembagian *data training* sebanyak 75% dan sebanyak 25% untuk *data testing*. Tahapan yang dilakukan dalam penelitian tersebut secara umum dibagi menjadi dua, yang pertama adalah tahapan penelitian pembuatan model klasifikasi menggunakan algoritma *Naïve Bayes* tanpa dikombinasikan dengan metode lainnya. Dan yang kedua adalah tahapan penelitian menggunakan algoritma *Naïve Bayes* yang dikombinasikan dengan metode *Particle Swarm Optimization*. Kedua tahapan tersebut nantinya akan dibandingkan hasil performa algoritmanya, dari kedua perbandingan tersebut nantinya diketahui apakah penggunaan PSO yang dikombinasikan dengan algoritma *Naïve Bayes* dapat meningkatkan performa model yang dibuat, dan apakah hasil pengukuran model yang dibuat dengan algoritma *Naïve Bayes* yang dikombinasikan dengan PSO akan lebih tinggi performanya. Hasil pengujian dari penelitian tersebut algoritma *Naïve bayes* murni tanpa dikombinasikan dengan apapun memiliki akurasi sebesar 82,14% dan nilai AUC sebesar 0,686. Sedangkan hasil

pengukuran performa model algoritma *Naïve Bayes* yang dikombinasikan dengan PSO mendapat akurasi sebesar 92,86% dengan nilai AUC adalah 0,839. Berdasarkan hasil dari penelitian, dapat disimpulkan bahwa penambahan metode *Particle Swarm Optimization* dapat meningkatkan performa model klasifikasi yang dibuat, dan dalam kasus tersebut algoritma yang dikombinasikan adalah algoritma *Naïve Bayes*.

Selanjutnya, terdapat penelitian yang dilakukan oleh Dito Putro Utomo dan Mesran pada tahun 2020 (Utomo & Mesran, 2020). Penelitian tersebut dilakukan untuk membandingkan performa algoritma C5.0 dan *Naïve Bayes*, kedua algoritma tersebut dikobimbinasikan dengan metode *Principal Component Analysis* (PCA) untuk mereduksi jumlah atribut sehingga yang tersisa hanya atribut yang memiliki bobot paling tinggi dan dirasa paling berpengaruh. Tahapan yang dilakukan adalah mencari datasest terlebih dahulu. Datasest yang digunakan adalah dataset yang bersumber dari UCI *machine learning*, datasets tersebut memiliki atribut sebanyak 57 dan memiliki 2 kelas target yaitu *CAD* dan Normal. Penelitian ini menggunakan PCA untuk mengurangi atribut karena dirasa terlalu banyak dan bisa mengurangi efektifitas model dalam melakukan proses *training* dan klasifikasi, hasil dari PCA menyisakan 10 atribut yang memiliki bobot paling tinggi. Kemudian, tahapan selanjutnya adalah melakukan *preprocessing* data untuk menghasilkan data yang berkualitas. Selanjutnya dilakukan proses pembuatan model menggunakan algoritma C5.0 dan *Naïve Bayes*. Kemudian data yang digunakan akan dilakukan proses PCA untuk mereduksi atribut sehingga dapat mengurangi atribut. Tahap terakhir adalah melakukan klasifikasi ulang pada data yang sudah direduksi menggunakan kedua algoritma yang sama dan melakukan pengujian serta pengukuran algoritma, kemudian membandingkan hasil dari masing-masing pengujian dan menentukan kesimpulan berdasarkan hasil dari penelitian yang dilakukan. Hasil dari penelitian tersebut adalah pengujian algoritma C5.0 tidak merubah akurasi yaitu sebesar 95,38% baik menggunakan data yang sudah direduksi ataupun data yang belum direduksi. Sedangkan algoritma *Naïve Bayes*, pengujian menggunakan data yang belum direduksi menghasilkan akurasi sebesar 99,01% dan pengujian menggunakan data yang sudah direduksi mendapatkan akurasi sebesar 98,53%. Dengan demikian dapat disimpulkan bahwa algoritma *Naïve Bayes* memiliki performa yang lebih baik, bahkan setelah data direduksi dan hanya menyisakan 10 atribut algoritma tersebut masih dapat melakukan klasifikasi dan mendapatkan nilai akurasi yang tinggi. Hal ini juga didukung bahwa algoritma *Naïve Bayes* tidak memerlukan *rule* seperti algoritma C5,0. Oleh karena itu, algoritma *Naïve Bayes* dapat melakukan klasifikasi lebih baik.

Kemudian penelitian yang dilakukan oleh Dwi Normawati dan Sri Winiarti pada tahun 2017 (Normawati & Winarti, 2017). Penelitian tersebut membahas tentang diagnosis pada data penyakit jantung menggunakan teknik seleksi fitur bernama *Variable Precision Rough Set* (VPRS) yang merupakan pengembahan dari metode *Rough Set*. Dalam penelitian tersebut, juga dilakukan penggabungan metode yaitu metode VPRS dan metode seleksi fitur berbasis medis atau *Motivated Feature Selection* (MFS) agar menghindari reduksi atribut yang dianggap penting oleh medis apabila hanya menggunakan metode VPRS. Penggabungan dua metode tersebut diharapkan dapat meningkatkan performa model klasifikasi. Data yang digunakan dalam penelitian

tersebut adalah datasets *Cleveland Heart Disease* yang berjumlah 303 data yang bersumber dari UCI *machine learning*. Datasets tersebut memiliki 7 data yang rusak, oleh karena itu 7 data tersebut dihapus supaya tidak mempengaruhi hasil klasifikasi. Metodologi yang dilakukan dalam penelitian tersebut adalah pengumpulan datasets, *preprocessing* data yang dilakukan dengan membersihkan data yang rusak atau *missing value*, merubah kelas data menjadi *binary class* dengan asumsi label 0 yang berarti data tersebut masuk ke dalam klasifikasi sehat, dan label 1 yang berarti data tersebut masuk ke dalam klasifikasi sakit. Kemudian, tahap selanjutnya adalah diskritasi data, yaitu merubah data numerik menjadi diskrit. Selanjutnya dilakukan proses seleksi fitur, dalam penelitian ini seleksi fitur yang dilakukan menggunakan dua acara, yaitu teknik VPRS dan teknik MFS. Kemudian tahapan selanjutnya adalah pembuatan *rule* yang berisi aturan yang akan dijadikan untuk proses klasifikasi menggunakan data tersebut. Langkah terakhir adalah pengujian dan evaluasi model klasifikasi. Evaluasi performa model tersebut dilakukan menggunakan *confusion matrix*. Hasil dari evaluasi program tersebut mendapatkan nilai akurasi, presisi, dan *recall*. Metode VPRS yang digunakan untuk klasifikasi mendapatkan akurasi sebesar 84,84% metode MFS yang digunakan untuk klasifikasi menghasilkan akurasi sebesar 86,86%. Sedangkan kombinasi dari kedua metode VPRS dan metode MFS menghasilkan model yang memiliki akurasi sebesar 84,84%. Dari hasil penelitian tersebut dapat disimpulkan bahwa, penggunakan metode seleksi fitur VPRS dinilai lebih baik karena menghasilkan model yang memiliki performa yang lebih baik daripada proses klasifikasi yang dilakukan tanpa menggunakan seleksi fitur. Sedangkan menggunakan metode VPRS yang dikombinasikan dengan metode MFS akan menghasilkan *rule* yang lebih sedikit, namun tetap mendapatkan akurasi yang baik sebesar 84,84%.

Selanjutnya ada penelitian yang dilakukan oleh Vincentius Adbi Gunawan, Leonardus Sandy Ade Putra, dan Ignitia Imelda Fitriana pada tahun 2020 (Gunawan et al., 2020). Penelitian tersebut dilakukan untuk membuat sistem yang dapat melakukan diagnosis penyakit jantung dengan menggunakan citra mata, kususnya pada bagian iris. Menurut penelitian tersebut, apabila seseorang terkena penyakit jantung maka akan terjadi penyempitan pembuluh darah dan akan sangat berkaitan dengan aliran darah yang terjadi di bagian mata. Oleh karena itu penelitian ini dilakukan pendeteksian penyakit jantung dengan input berupa citra foto bola mata. Metode yang digunakan untuk mendeteksi organ tubuh menggunakan iris mata adalah Iridologi. Setiap organ yang ada di tubuh kita dapat dicerminkan melalui iris, iris mata kanan akan mencerminkan kondisi organ tubuh yang berada di kanan, begitu juga dengan iris mata kiri yang akan mencerminkan kondisi organ tubuh di bagian kiri. Data yang digunakan dalam penelitian tersebut adalah citra mata yang berjumlah 70 foto, terdiri dari 35 foto dengan klasifikasi normal, dan 35 foto lainnya tergolong dalam klasifikasi abnormal. Tahapan yang dilakukan adalah pengolahan citra digital dan pembagian data menjadi *data training* sebanyak 30 foto dan *data testing* sebanyak 40 foto. Kemudian dilakukan proses ekstraksi citra menggunakan GLCM dan klasifikasi dengan menggunakan algoritma SVM, kemudian dilakukan pengujian untuk mengukur performa algoritma. Hasil pengujian model menggunakan algoritma SVM mendapatkan akurasi sebesar 87,15%.

Dari penelitian tersebut, didapatkan kesimpulan bahwa hubungan penyakit jantung dengan iris mata adalah, saat seseorang menderita penyakit jantung maka akan terjadi masalah pada syaraf matanya. Sedangkan untuk orang normal yang tidak memiliki penyakit jantung, maka tidak akan ada masalah pada syaraf mata.

Selanjutnya masih ada penelitian yang membahas tentang penyakit jantung namun menggunakan *deep learning*, yaitu penelitian yang dilakukan oleh Majzoob K. Omer pada tahun 2018 (Omer et al., 2018). Penelitian tersebut dilakukan untuk membuat sistem pendeteksian penyakit jantung, karena dirasa penyakit jantung menjadi penyakit yang perlu diwaspadai namun untuk melakukan pengecekan sebelumnya harus melakukan konsultasi ke dokter. Proses tersebut tentunya tidak efektif, ditambah tidak semua rumah sakit memiliki dokter jantung yang berkompeten untuk dapat melakukan pengecekan penyakit jantung. Oleh karena itu penelitian tersebut dilakukan untuk membuat sistem yang dapat digunakan secara universal untuk mendeteksi penyakit jantung menggunakan metode *deep neural network*. Datasets yang digunakan di dalam penelitian tersebut adalah data pasien yang terkait dengan IHD yang bersumber dari sebuah rumah sakit di Jakarta. Datasets yang digunakan berjumlah 305 data yang memiliki 10 atribut. Data tersebut dibagi menjadi dua bagian, yaitu *data training* sebanyak 250 data, dan *data testing* yang berjumlah 55 data. Metode yang digunakan untuk menyelesaikan permasalahan dalam penelitian ini menggunakan *deep neural network* dengan konfigurasi 152 neuron masukan, 52 neuron keluaran, dan memiliki 4 *hidden layer*. Hasil dari pengujian yang dilakukan sebanyak 5 rangkaian percobaan pada 55 data adalah, dari 5 kali percobaan pengujian yang dilakukan menggunakan metode *neural network* konnvensional mendapatkan rata-rata akurasi sebesar 99,055% sedangkan metode *deep neural network* mendapatkan rata-rata akurasi sebesar 99,787%. Berdasarkan penelitian tersebut didapatkan kesimpulan bahwa melakukan klasifikasi menggunakan *deep neural network* dapat meningkatkan performa yang signifikan yaitu sebesar 0,7322% apabila dibandingkan dengan menggunakan teknik *neural network* konvensional.

Penelitian selanjutnya adalah penelitian yang dilakukan oleh Wiharto, Esti Suryani, dan Vicka Cahyawati pada tahun 2019 (Wiharto et al., 2019). Prediksi penyakit jantung yang dilakukan untuk sepuluh tahun kedepan dirasa masih bisa dilakukan, oleh karena itu data-data sumber daya harus digunakan secara maksimal khususnya adalah data pasien jantung koroner yang dapat digunakan untuk membuat sistem yang dapat memprediksi apakah seseorang tersebut menderita penyakit jantung atau tidak. Penelitian tersebut dilakukan menggunakan metode jaringan syaraf tiruan *multi layer preceptron* (MLP-ANN) dan *Duo Output Ensemble Artificial Neural Network* (DOANNE). Data yang digunakan adalah data yang bersumber dari RSUD Dr.Moewardi, Surakarta. Datasest tersebut memiliki 12 atribut, dengan jumlah data adalah 72 data, 36 pasien terdiagnosis penyakit jantung coroner positif, dan sisa data lainnya tergolong dalam klasifikasi negative atau sehat. Sebanyak 24 data digunakan sebagai *data testing* yang nantinya akan digunakan untuk pengujian. Tahapan yang ada dalam penelitian tesebut adalah pengumpulan data, *preprocessing* untuk membersihkan data, kemudian melakukan *training*. Dari algoritma *neural network* tersebut akan

mengklasifikasikan data sesuai dengan klasifikasi yang diprediksi oleh model yang dibuat. Hasil dari penelitian tersebut menjelaskan bahwa melakukan klasifikasi menggunakan teknik DOANNE-LM dapat meningkatkan performa model dan dapat mencegah *overfitting* pada model. Dibuktikan dengan klasifikasi yang dilakukan dengan teknik DOANNE-LM dapat mendapat akurasi sebesar 86,875%. Model klasifikasi yang dibangun dengan menggunakan DOANNE-LM mampu menekan *overfitting* sebesar 49,09% dibandingkan dengan klasifikasi yang dilakukan dengan menggunakan JST-LM.

Selanjutnya terdapat penelitian yang dilakukan oleh Pareza Alam Jusia pada tahun 2018 (Pareza Alam Jusia, 2018). Penelitian tersebut dilakukan untuk *improve classification accuracy* atau *ensemble methods technique* dengan cara mengkombinasikan algoritma *Decision Tree* dan teknik *Particle Swarm Optimization* (PSO) yang ditambahkan dengan metode *Adaboost* untuk melakukan prediksi penyakit jantung pada seseorang menggunakan data yang sudah ada. Datasets yang digunakan di dalam penelitian tersebut adalah data sekunder yang bersumber dari UCI *machine learning* sejumlah 270 data dengan rincian 120 data tergolong dalam klasifikasi negative dan sebanyak 150 data tergolong ke dalam klasifikasi dengan label positif terkena penyakit jantung. Dalam penelitian tersebut terdapat beberapa tahapan, tahapan tersebut antara lain proses *import* datasets, proses preprocessing data, pembuatan model menggunakan algoritma C4.5 dan algoritma C4.5 yang dikombinasikan dengan PSO dan teknik *Adaboost*. Kemudian dilakukan pengujian dan pengukuran performa algoritma. Dalam penelitian tersebut pengujian dan pengukuran dilakukan menggunakan *Confusion Matrix* dan AUC untuk mengukur akurasi, presisi, dan recall. Hasil dari penelitian tersebut adalah pembuatan model yang dibuat menggunakan algoritma C4.5 murni mendapatkan akurasi sebesar 79,26%. Model klasifikasi yang dibuat menggunakan algoritma C4.5 yang dikombinasikan dengan PSO mendapatkan akurasi paling tinggi, sebesar 82,59%. Sedangkan model yang dibuat dengan menggunakan algoritma C4.5 yang dikombinasikan dengan teknik *Adaboost* mendapatkan akurasi yang sama dengan C4.5 murni, yaitu sebesar 79,26%.

Selanjutnya terdapat penelitian yang dilakukan oleh Tri Retnasari dan Eva Rahmawati yang dilakukan pada tahun 2017 (Retnasari & Rahmawati, 2017). Dalam penelitian tersebut dijelaskan pentingnya menciptakan sistem yang mampu mendiagnosis seseorang apakah terkena penyakit jantung atau tidak. Beberapa penelitian sudah dilakukan menggunakan beberapa algoritma untuk melakukan prediksi penyakit jantung. Dalam penelitian tersebut penulis melakukan penelitian dengan menggunakan algoritma *Naïve Bayes* dan C4.5 yang kemudian akan dibandingkan untuk mendapatkan algoritma yang lebih baik dalam menangani klasifikasi penyakit jantung. Tahapan yang dilakukan dalam penelitian tersebut adalah melakukan pengumpulan data, data yang digunakan di dalam penelitian tersebut adalah datasets sekunder yang bersumber dari UCI *machine learning*. Setelah melakukan pengumpulan data, tahapan selanjutnya adalah *preprocessing* data, *preprocessing* dilakukan untuk memilih data yang baik, membersihkan data, ataupun mentransformasikan data ke dalam bentuk yang diinginkan sebelum dilakukan pemodelan. Kemduian, dilakukan proses pembuatan model dengan menggunakan algoritma *Naïve Bayes* dan C4,5. Kemudian dilakukan pengujian dan

pengukuran untuk mengevaluasi model klasifikasi yang dibuat dengan kedua algoritma tersebut, dan menentukan algoritma terbaik untuk menangani klasifikasi penyakit jantung. Untuk mengukur performa algoritma menggunakan *confusion matrix* dan kurva AUC. Berdasarkan penelitian yang sudah dilakukan algoritma *Naïve Bayes* mendapatkan akurasi sebesar 86,67% dan AUC 0,090. Sedangkan algoritma C4,5 mendapatkan akurasi sebesar 83,70% dan AUC sebesar 0,834. Dari penelitian tersebut dapat disimpulkan bahwa algoritma *Naïve Bayes* memiliki performa yang lebih baik daripada algoritma C4,5 walaupun hanya memiliki selisih yang tidak terlalu signifikan.

Kemudian, terdapat penelitian tentang penyakit jantung yang dilakukan oleh Syafitri Hidayatul Annur Aini, Yuita Arum Sari, dan Achmad Arwan pada tahun 2018 (Aini et al., 2018). Penelitian tersebut dilakukan untuk mengurangi dimensi data menggunakan *information gain*, yang nantinya hanya menyisakan atribut yang penting atau memiliki bobot paling tinggi. Kemudian setelah melakukan *information gain*, dilakukan klasifikasi menggunakan algoritma *Naïve Bayes* dan *K-Nearest Neighbor*. Datasets yang digunakan adalah data sekunder yang bersumber dari UCI *machine learning* dengan data yang berjumlah 270 dan memiliki atribut sebanyak 13, dengan memiliki dua kelas target yaitu terkena penyakit jantung (TPJ) dan tidak terkena penyakit jantung (TTPJ). Tahapan yang dilakukan di dalam penelitian tersebut adalah, pertama data dikonversikan dahulu dari yang semula bersifat numerik menjadi data yang bersifat kategoris. Kemudian data yang sudah dikonversikan dilakukan proses reduksi atribut menggunakan *information gain*. Kemudian pada data yang belum dikonversi, dilakukan proses klasifikasi menggunakan algoritma KNN pada data numerik, dan dilanjutkan dengan perhitungan data yang bersifat kategoris menggunakan algoritma *Naïve Bayes*. Pengujian yang dilakukan terbagi menjadi dua macam, yaitu pengujian dengan data latih dengan kelas seimbang, dan pengujian dengan data latih dengan kelas tidak seimbang. Untuk jumlah atribut yang digunakan pada masing-masing pengujian adalah menggunakan 6 atribut dan 4 akribut, sedangkan nilai *k* yang digunakan berkelipatan 10 mulai dari *k*=5 sampai dengan *k*=95. Berdasarkan hasil penelitian dapat disimpulkan bahwa saat pengujian dengan menggunakan data latih kelas seimbang mendapatkan akurasi tertinggi yaitu 92,31% dengan atribut sejumlah 6 dan nilai *k*=25. Sedangkan pengujian yang dilakukan pada data latih tidak seimbang, nilai akurasi tertinggi yaitu 92,31% dengan menggunakan 4 atribut dan nilai *k*=35.

Selanjutnya terdapat penelitian yang dilakukan dengan menggunakan metode XgBoost. Penelitian tersebut dilakukan oleh Ichwanul Muslim Karo pada tahun 2020 (Karo, 2020). Penelitian tersebut bertujuan untuk melakukan klasifikasi titik api penyebab kebakaran hutan pada data yang bersumber dari *Global Forest Watch* (GFW) dengan menggunakan algoritma XgBoost. Data yang digunakan berjumlah 300 dengan 12 atribut, dengan label kelas yang menjadi target klasifikasi adalah empat kelas label. Tahapan yang dilakukan di dalam penelitian tersebut antara lain adalah tahap *preprocessing* yang dilakukan untuk memilih data yang baik, melakukan normalisasi data, dan melakukan *feature important* untuk mengurangi atribut dan hanya menyisakan atribut yang penting. Kemudian pembuatan model klasifikasi dilakukan menggunakan algoritma XgBoost dengan beberapa parameter seperti *max depth*=5, *seed*=7, *test*

*size*=0,35, *feature*=1-9, dan *learning rate*=0,05. Setelah pembuatan model dilakukan, kemudian dilakukan validasi model untuk mengukur performa model klasifikasi, pengukuran model klasifikasi dilakukan dengan menggunakan *confusion matrix* untuk mengetahui nilai akurasi, presisi, dan *recall*. Berdasarkan penelitian yang dilakukan, hasil yang didapatkan adalah dengan menggunakan *feature important* dapat mengurangi jumlah atribut yang tadinya 12 atribut menjadi 9 atribut yang dirasa penting. Namun saat dilakukan validasi menggunakan algoritma XgBoost, ternyata akan lebih optimal dan memiliki performa yang baik apabila dibuat klasifikasi dilakukan dengan menggunakan 6 atau 7 atribut yang paling berpengaruh. Hasil dari pengukuran performa, model algoritma XgBoost yang dilakukan untuk klasifikasi mendapatkan nilai SE sebesar 91,32% nilai SP sebesar 93,16%, dan nilai MCC sebesar 92,75%.

Selanjutnya terdapat penelitian yang dilakukan oleh Sherla Yualinda, Dr. Dedy Rahman Wijawa, S.T., M.T., dan Elis Hernawati, S.T., M.Kom. yang dilakukan pada tahun 2020 (Yualinda et al., 2020). Penelitian tersebut dilakukan untuk membuat sistem prediksi kemiskinan menggunakan *machine learning* dengan metode algoritma *Naïve Bayes* dan XgBoost. Tahapan yang dilakukan untuk membuat model prediksi adalah melakukan import data, melakukan *preprocessing* data seperti normalisasi data, kemudian dilakukan proses dengan menggunakan teknik *Similarity Based Feature Selection* untuk mengurangi dimensi atribut dan menyisakan atribut yang penting. Kemudian dilakukan proses pemodelan klasifikasi dengan menggunakan algoritma *Naïve Bayes* dan algoritma XgBoost. Setelah itu, dilakukan proses evalusai dengan menghitung nilai RMSE dan R. Hasil dari penelitian tersebut daapt diciptakan sistem prediksi kemiskinan yang ada di Indonesia dengan hasil tampilan akan menghasilkan grafik prediksi kemiskinan yang ada di Indonesia.

Selanjutnya terdapat penelitian yang dilakukan pada tahun 2020 oleh Muhamad Syukron, Rukun Santoso, dan Tatik Widiharih (Syukron et al., 2020). Penelitian tersebut dilakukan untuk membuat model klasifikasi yang digunakan untuk mengklasifikasikan tingkat penyakit hepatitis C. Data yang digunakan memiliki jenis *Imbalance Data*, yang berarti data yang ada memiliki kelas yang tidak seimbang jumlahnya. Apabila dipaksakan membuat model dengan menggunakan data yang tidak seimbang, model akan cenderung memprediksi dengan hasil kelas yang lebih banyak jumlahnya. Oleh karena itu, peneliti menggunakan teknik SMOTE untuk menangani ketidak seimbangan kelas data tersebut. Dalam penelitian ini peneliti juga membandingkan performa model yang dihasilkan menggunakan algoritma *Random Forest*, dan XgBoost untuk mendapatkan kesimpulan algoritma yang memiliki performa terbaik. Tahapan yang digunakan antara lain *preprocessing* data, data yang akan digunakan tentunya belum baik atau memiliki beberapa data yang harus dibuang. Dalam kasus ini, data yang digunakan ternyata memiliki *outlier*, oleh karena itu perlu dihapus supaya model yang dibuat dapat lebih optimal saat melakukan proses *training*. Kemudian dibuat model klasifikasi dengan menggunakan algoritma *Random Forest* dan XgBoost serta menggunakan *Random Search* untuk menentukan parameter terbaik. Setelah itu dilakukan pengujian sekaligus pengukuran algoritma menggunakan *confusion matrix* untuk mengetahui akurasi, presisi, dan *recall*. Hasil dari penelitian tersebut dapat disimpulkan bahwa algoritma

*Random Forest* yang dikombinasikan dengan teknik SMOTE ternyata memiliki akurasi paling tinggi yaitu 80,97% dibandingkan algoritma XgBoost dan SMOTE yang mendapatkan akurasi sebesar 78,63%.

Kemudian, terdapat penelitian yang dilakukan oleh Ngakan Nyoman Pandika Pinata, I Made Sukarsa, Ni Kadek Dwi Rusjayanthi pada tahun 2020 (Pinata et al., 2020). Penelitian tersebut membahas tentang prediksi kecelakaan lalu lintas yang ada di Bali. Sistem klasifikasi tersebut dibuat dengan menggunakan algoritma XgBoost menggunakan bahasa pemrograman python. Data yang digunakan di dalam penelitian tersebut adalah data jumlah kecelakaan lalu lintas yang ada di Bali dari tahun 1996 sampai dengan tahun 2019 sejumlah 24 data sesuai dengan tahun yang ada pada data tersebut. Data yang digunakan dalam penelitian tersebut dibagi menjadi dua bagian yaitu *data training* berjumlah 20 data, dan *data testing* berjumlah 4 data. Pengukuran yang digunakan di dalam penelitian tersebut menggunakan RMSE untuk mengukur kesalahan dari model klasifikasi yang dibuat. Berdasarkan penelitian yang dilakukan, model klasifikasi yang dibuat menggunakan algoritma XgBoost dapat menghasilkan performa yang baik dibuktikan dengan nilai RMSE yang cukup rendah. Nilai *error* pada kategori jumlah kejadian adalah 21,69. Nilai RMSE pada kategori jumlah orang meninggal dunia adalah 4,92. Nilai RMSE pada kategori luka berat adalah 4,11. Dan pada kategori luka ringan mendapatkan nilai RMSE sebesar 77,24.

Selanjutnya terdapat penelitian yang dilakukan oleh Ahmedbahaldin Ibrahem Ahmed Osman, Ali Najah Ahmed pada tahun 2020 di Malaysia (Ibrahem Ahmed Osman et al., 2020). Penelitian tersebut dilakukan untuk membuat sistem yang dapat memprediksi ketinggian air di daerah Selangor negara Malaysia. Data yang digunakan di dalam penelitian tersebut adalah data dari tanggal 20 Oktober 2017 hingga 24 Juli 2018, data tersebut memiliki atribut antara lain adalah curah hujan, suhu, evaporasi, tinggi air. Data yang digunakan tersebut terlebih dahulu dibagi menajdi dua bagian, yaitu *data training* dan *data testing* dengan perbandingan 70% untuk *data training* dan 30% untuk *data testing*. Metode yang digunakan menggunakan beberapa algoritma seperti XgBoost, JST, dan SVR dimana ketiga algoritma tersebut akan dibandingkan hasil performanya untuk mengetahui algoritma mana yang memiliki performa terbaik. Proses pengukuran performa algoritma pada penelitian tersebut menggunakan *R Square*. Hasil dari penelitian tersebut membuktikan bahwa algoritma XgBoost dinilai memiliki performa yang lebih baik daripada kedua algoritma JST dan SVR. Hal tersebut dibuktikan dalam pengukuran kinerja, algoritma XgBoost mendapatkan nilai MAE sebesar 0,086 algoritma JST yang mendapatkan nilai MAE sebesar 0,254 dan algoritma SVR dengan nilai sebesar 0,111.

**Tabel 1.0** *Tabel State of the art*

| No. | Penulis | Judul | Tahun | Metode | Hasil |
|---|---|---|---|---|---|
| 1. | Erwin Prasetyo & Budi Prasetyo | Peningkatan Akurasi Klasifikasi Algoritma C4,5 Menggunakan Teknik Bagging Pada | 2020 | Algoritma C4,5 dan *Bagging* | Algoritma C4,5 murni akurasi 72,98% Algoritma C4,5 digabungkan |

| | | | | | |
|---|---|---|---|---|---|
| | | Diagnosis Penyakit Jantung | | | *Bagging* akurasi 81,84% |
| 2. | Pandito Dewa Putra & Dian Palupi Rini | Prediksi Penyakit Jantung Dengan Algoritma Klasifikasi | 2019 | Algoritma *Naïve Bayes*, SVM, C4.5, *Logistic Regression*, *Back Propagation* | Algoritma *Naïve Bayes* akurasi tertinggi dengan nilai 84,07% |
| 3. | Ardea Bagas Wibisono & Achmad Fahrurozi | Perbandingan Algoritma Klasifikasi Dalam Pengklasifikasian Data Penyakit Jantung Koroner | 2019 | Algoritma *Naïve Bayes*, KNN, *Decision Tree*, *Random Forest*, SVM | Algoritma *Random Forest* paling baik dengan akurasi mencapai 85,67% |
| 4. | Riski Annisa | Analisis Komparasi Algoritma Data Mining Untuk Prediksi Penderita Penyakit Jantung | 2019 | Algoritma *Decision tree*, *Naïve Bayes*, KNN, *Random Forest*, *Decision Stump* | Algoritma *Random Forest* memiliki akruasi tertinggi dengan nilai 80,38% |
| 5. | Mei Lestari | Penerapan Algoritma Klasifikasi *Nearest Neighbor* (K-NN) Untuk Mendeteksi Penyakit Jantung | 2014 | Algoritma K-*Nearest Neighbor* | Akurasi 70%, nilai AUC 0,875 |
| 6. | Abdul Rohman, Vincentius Suharto, Catur Supriyanto | Penerapan Algoritma C4,5 Berbasis *Adaboost* Untuk Prediksi Penyakit Jantung | 2017 | Algoritma C4,5 dan *Adaboost* | Algoritma C4,5 murni akurasi 86,59% Algoritma C4,5 digabungkan *Adaboost* akurasi 92,24% |
| 7. | Nur Aeni Widiastuti, Stefanus Santosa, Catur Supriyanto | Algoritma Klasifikasi Data Mining *Naïve Bayes* Berbasis *Particle Swarm Optimization* Untuk Deteksi Penyakit jantung | 2014 | Algoritma *Naïve Bayes* dan *Particle Swarm Optimization* (PSO) | Algoritma *Naïve Bayes* murni akurasi 82,14% *Naïve Bayes* dan PSO akurasi 92,86% |
| 8. | Dito Putro utomo & Mesran | Analisis Komparasi Metode Klasifikasi Data Mining dan Reduksi Atribut Pada Data Set Penyakit Jantung | 2020 | Algoritma C5.0, *Naïve Bayes*, PCA | Algoritma *Naïve Bayes* terbaik dengan akurasi 99,01% sebelum direduksi, dan 98,53% sesuah direduksi |
| 9. | Dwi Normawati & Sri Winiarti | Seleksi Fitur Menggunakan Penambangan Data Berbasis *Variable* | 2017 | Algoritma *Variabel Precision Rough* | Akurasi kombinasi VPRS dan MFS |

| | | | | | |
|---|---|---|---|---|---|
| | | *Precision Rough Set* (VPRS) Untuk Diagnosis Penyakit Jantung Koroner | | *Set* (VPRS) dan MFS | mencapai 84,44% |
| 10. | Vincentius Abdi Gunawan, Leonardus Sandy A. P., Ignitia Imelda F | Sistem Diagnosis Otomatis Identifikasi Penyakit Jantung Koroner Menggunakan Ciri GLCM dan Klasifikasi SVM | 2020 | Algoritma *Support Vercor Machine* (SVM) | Akurasi algoritma SVM sebesar 87,15% |
| 11. | Majzoob K. Omer, Osama E. Shate, Mohamed S. Adrees, Deris Stiawan, Munawar A. Riyadi, Rahmat Budiarto | *Deep Neural Network for Heart Disease Medical Prescription Expert System* | 2018 | Algoritma *Deep Neural Network* | Akurasi algoritma *Deep Neural Network* mencapai 99,787% |
| 12. | Wiharto Wiharto, Esti Suryani, Vicka Cahyawati | *The Methods of Duo Output Neural Network Ensemble for Prediction of Coronary Heart Disease* | 2019 | Algoritma *Neural Network with Ensemble Learning* | Mendapatkan nilai akurasi sebesar 86,875% dan menekan *overfitting* |
| 13. | Pareza Alam Jusia | Analisis Komparasi Pemodelan Algoritma *Decision Tree* Menggunakan Metode *Particle Swarm Optimization* Dan metode *Adaboost* Untuk Prediksi Awal Penyakit Jantung | 2018 | Algoritma *Decision Tree*, *Particle Swarm Optimizaton*, dan *Adaboost* | Algoritma C4,5 yang dikombinasikan PSO tertinggi dengan akurasi 82,59% |
| 14. | Tri Retnasari & Eva Rahmawati | Diagnosa Prediksi Penyakit Jantung Dengan Model Algoritma *Naïve Bayes* dan Algoritma C4,5 | 2017 | Algoritma *Naïve Bayes* dan Algoritma C4.5 | Algoritma *Naïve Bayes* lebih baik dengan akurasi 86,67% |
| 15. | Syafitri Hidayatul Annur Aini, Yuita Arum Sari, Achmad Arwan | Seleksi Fitur *Information Gain* Untuk Klasifikasi Penyakit Jantung Menggunakan Kombinasi Metode *K-Nearest Neighbor* dan *Naive Bayes* | 2018 | Algoritma *Information Gain*, Algoritma KNN dan *Naïve Bayes* | Kelas data seimbang akurasi tertinggi 92,31% Kelas tidak seimbang akurasi tertinggi 92,31% |
| 16. | Ichwanul Muslim Karo Karo | Implementasi Metode XgBoost dan Feature Important Untuk Klasifikasi Pada | 2020 | Algoritma XgBoost | Nilai SE sebesar 91,32% SP sebesar 93,16% |

| | | | | | |
|---|---|---|---|---|---|
| | | Kebakaran Hutan dan Lahan | | | Nilai MCC sebesar 92,75% |
| 17. | Sherla Yualinda, Dr. Dedy Rahma Wijaya, Elis Hernawati | *Application Based On E-Commerce Dataset For Poverty Prediction Using Naive Bayes Algorithm, XgBoost, And Simillarity Based Feature Selection* | 2020 | Algoritma *Naïve Bayes*, XgBoost, dan *Simillarity Based Feature Selection* | Dibuat sistem yang menampilkan prediksi kemiskinan dengan grafik |
| 18. | Muhamad Syukron, Rukun Santoso, Tatik Widiharih | Perbandingan Metode Smote *Random Forest* dan Smote XgBoost Untuk Klasifikasi Tingkat Penyakit Hepatitis C Pada *Imbalance Class Data* | 2020 | Algoritma *Random Forest*, XgBoost, dan teknik SMOTE | Algoritma *Random Forest* digabungkan SMOTE memiliki akurasi tertinggi dengan nilai 80,97% |
| 19. | Ngakan Nyoman Pandika Pinata, I Made Sukarsa, Ni Kadek Dwi Rusjayanthi | Prediksi Kecelakaan Lalu Lintas di Bali dengan XgBoost Pada Python | 2020 | Algoritma XgBoost | RMSE jumlah kejadian 21,69 RMSE jumlah orang meninggal 4,92 RMSE luka berat 4,11 RMSE luka ringan 77,24 |
| 20. | Ahmedbahaaaldin Ibrahem Ahmed Osman, Ali Najah Ahmed, Ming Fai Chow, Yuk Feng Huang, Ahmed El-Shafie | *Extreme Gradient Boosting (XgBoost) model to predict the groundwater level in* Selangor Malaysia | 2020 | Algoritma XgBoost, JST, dan SVR | Algoritma XgBoost terbaik dengan nilai MAE sebesar 0,086 |

## 2.2.Tinjauan Pustaka

### 2.2.1. Jantung

Dalam penelitian yang akan saya lakukan dengan judul Prediksi Penyakit Jantung Dengan Menggunakan Algoritma XgBoost dan Randomized Search Optimizer ini adalah datasets jantung yang dapat diunduh di UCI *machine learning*. Jantung adalah organ tubuh manusia yang memiliki fungsi yang sangat penting yaitu untuk mengedarkan darah ke seluruh tubuh. Penyakit jantung adalah keadaan jantung yang tidak dapat melaksanakan fungsinya dengan baik, sehingga fungsi kerja jantung yang bekerja untuk memompa dan mengedarkan darah ke seluruh tubuh menjadi terganggu (Anies, 2015). Penyakit jantung umumnya disebabkan karena otot jantung yang melemah sehingga terdapat celah antara serambi kiri dan serambi kanan yang mengakibatkan tercampurnya darah bersih dan darah kotor.

### 2.2.2. Deteksi Penyakit Jantung

Cara deteksi penyakit jantung dapat dilakukan secara manual, yaitu dengan melakukan konsultasi kepada dokter spesialis jantung dan nantinya akan dilakukan pemeriksaan laboratiorium (Wibisono & Fahrurozi, 2019). Namun, cara tersebut tidak terlalu efektif selain itu juga memerlukan waktu dan biaya yang tidak sedikit. Oleh karena itu perlu adanya cara deteksi penyakit jantung secara digital, yaitu dengan menggunakan data-data yang sudah ada, yang nantinya akan digunakan komputer untuk mempelajari data dan memprediksi. Parameter yang digunakan dalam pendeteksian penyakit jantung diantaranya adalah sebagai berikut.

**Tabel 1.1** *Parameter data*

| Atribut | Deskripsi | Keterangan |
|---|---|---|
| *Age* | Umur pasien | *Numerik* |
| *Sex* | Jenis kelamin pasien | 0: Wanita, 1: Pria |
| *Cp* | *Chest pain type* | 1:*typical angina*, 2:*atypical angina*, 3:*non-angina pain*, 4:*asympotomatic* |
| *Trestbps* | *Resting blood presure* | *Numerik* |
| *Chol* | Serum kolesterol | *Numerik* |
| *Fbs* | *Fasting blood sugar*>120 mg/dl | 0:*false*, 1:*true* |
| *Restecg* | Hasil ECG selama istirahat | 0:normal, 1:abnormal (memiliki kelainan gelombang ST-T), 2:*hipertrofil ventrikel* |
| *Thalac* | Detak jantung maksimal yang dicapai | *Numerik* |
| *Exang* | Ukuran *boolean* yang menunjukan apakah latihan angina industri terjadi | 0:*No*, 1:*Yes* |
| *Oldpeak* | *Segment* ST yang diperoleh dari latihan relatif terhadap istirahat | *Numerik* |
| *Slope* | Kemiringan segmen ST untuk latihan maksimal (puncak) | 1:*upsloping*, 2:*flat*, 3:*downsloping* |
| *Ca* | Jumlah *vessel* utama yang diwarnai oleh *fluroskopi* | 0, 1, 2, dan 3 |
| *Thal* | *Thal* | 3:*normal*, 6:cacat tetap, 7:cacat *reversible* |

### 2.2.3. Machine Learning

Pada awalnya komputer diciptakan untuk memudahkan manusia dalam membantu pekerjaan manusia, pekerjaan yang dapat diselesaikan dengan computer antara lain ada perhitungan yang dirasa tidak dapat diselesaikan oleh manusia secara manual, atau dapat dikerjakan oleh manusia namun memerlukan waktu yang sangat lama. Dengan adanya komputer pekerjaan tersebut dapat dikerjaan dengan komputer secara cepat sehingga membuat pekerjaan tersebut efektif dan lebih efisian daripada dikerjakan oleh manusia secara manual. Kemudian, pekerjaan yang diselesaikan oleh computer lebih dapat menghindari dari adanya kesalahan, apabila suatu pekerjaan diselesaikan oleh manusia masih memiliki kemungkinan terjadinya kesalahan dari seseorang yang mengerjakan yang dinamakan *human*

*error*. Namun, seiring berjalannya waktu koomputer menjadi semakin pandai dan terciptalan *machine learning*. Awal mulanya *machine learning* diperkenalkan oleh Arthur Samuel di tahun 1959 melalui jurnal dengan judul "Some Studies in Machine Learning Using the Game of Chekers". Jurnal tersebut dipublikasikan oleh IBM Journal of Research and Development pada bulan Juli tahun 1959. Pengertian *machine learning* adalah cabang dari kecerdasan buatan (AI) yang merupakan disiplin ilmu yang mencakup seperti perancangan algoritma computer dengan tujuan computer dapat mengembangkan perilaku yang didasarkan terhadap data yang dipelajari oleh computer (Purnamasari et al., 2013). Fokus dari *machine learning* adalah bagaimana computer dapat otomatis mengenali pola yang terjadi di dalam data yang sudah dipelajari oleh computer. Di dalam *machine learning* terdapat beberapa jenis, diantaranya adalah *supervised learning*, *unsupervised learning*, dan *reinforcement learning*.

*Supervised learning* adalah jenis *machine learning* yang bertujuan untuk memetakan atau mengelompokkan suatu data berdasarkan atribut-atribut yang sudah dipelajari oleh computer, dimana *output* yang akan dipetakan oleh computer sudah tertera pada data yang digunakan (Alpaydin, 2010). Dengan kata lain, computer sudah mempelajari hasil pemetaan yang diharapkan oleh data, sehingga computer akan menebak *output* yang ada sesuai dengan kategori label kelas dari data. Pada *supervised learning*, data yang digunakan memiliki fitur yang merupakan ciri-ciri yang ada pada masing-masing kelas *output*. Masing-masing kelas *output* memiliki jumlah fitur atau atribut yang sama yang nantinya akan dipelajari oleh computer dan akan digunakan untuk memetakan target *output* dari atribut dan kelas yang sudah dipelajari, salah satu pekerjaan yang termasuk ke dalam jenis *supervised learning* adalah klasifikasi. Permasalah yang ada pada *machine learning* dengan jenis *supervised learning* adalah bagaimana computer dapat memetakan target *output* dengan tepat dan akurat menggunakan atribut yang sudah dipelajari. Berbeda dengan *supervised learning* dimana data yang digunakan sudah memiliki kelas target dan computer akan memetakan data berdasarkan atribut yang dimiliki data sesuai dengan target kelasnya, sedangkan *Unsupervised learning* adalah salah satu jenis *machine learning* yang digunakan untuk menarik kesimpulan dari data yang ada (Nurhayati et al., 2019). Di dalam *unsupervised learning*, data yang digunakan juga memiliki fitur, namun data tersebut tidak memiliki *output* kelas data. Oleh karena itu, di dalam *unsupervised learning* computer akan menentukan sendiri kelas yang kemungkinan akan dimiliki di dalam data tersebut. metode yang umum digunakan dalam *unsupervised learning* adalah klustering, dimana dengan algoritma yang dimiliki, computer akan menentukan sendiri kluster-kluster yang ada di dalam data tersebut. Di dalam klustering, algoritma akan mengelompokkan data-data yang sekiranya memiliki atribut atau ciri-ciri yang mirip, jumlah kluster biasanya ditentukan sendiri, namun tidak menutup kemungkinan juga dapat menggunakan algoritma klustering yang baik untuk menentukan berapa jumlah kluster, sehingga nantinya jumlah kluster yang ada dapat lebih optimal sesuai dengan apa yang dikerjakan oleh algoritma

klustering. Selanjutnya *reinforcement learning* adalah teknik di dalam *machine learning* yang menggunakan konsep *trial and error*, teknik tersebut berinteraksi dengan lingkungan yang ada yang nantinya digunakan untuk memperbarui pengetahuannya (Arulkumaran et al., 2017). Dalam mempelajari lingkungan yang ada, teknik *reinforcement learning* akan mempelajari perubahan lingkungan yang terjadi secara dinamis, sehingga bisa mendapatkan pengetahuan dan mencapai tujuan yang diinginkan dalam menyelesaikan pekerjaan (Andreanus & Kurniawan, 2018). Di dalam menyelesaikan pekerjaan yang membutuhkan reinforcement learning, teknik ini memiliki dua strategi. Yang pertama adalah menemukan ruang dari tingkah laku lingkungan yang bertujuan untuk menentukan performa yang baik di dalam lingkungan yang ada. Dan yang kedua adalah menggunakan teknik statistik yang ada pada algoritma untuk melengkapi pengambilan keputusan yang ada di kondisi nyata.

### 2.2.4. Klasifikasi

Klasifikasi adalah salah satu contoh teknik dalam *machine learning* yang tergolong dalam kategori *supervised learning*. Klasifikasi adalah pekerjaan dalam machine learning yang bertujuan untuk untuk memperkirakan kelas dari suatu data yang memiliki atribut dan ciri-ciri (Indriyono et al., 2015). Dalam proses klasifikas terdapat dua tahapan yang harus dikerjakan. Yang pertama adalah tahap *learning*, dalam tahap tersebut algoritma klasifikasi akan mempelajari data yang digunakan. Data yang digunakan dalam klasifikasi memiliki atribut dan *output* kelas target. Atribut tersebut berisi ciri-ciri yang dimilik oleh data yang terdapat dalam *output* kelas yang dimiliki. Semua data yang digunakan untu klasifikasi memiliki jumlah atribut yang sama pada semua data. Atribut-atribut yang dimiliki masing-masing data akan dipelajari oleh computer menggunakan algoritma yang ada. Dengan mempelajari atribut tersebut, algoritima akan mengetahui ciri-ciri dan kelas yang ada, yang nantinya akan digunakan untuk memprediksi atau menentukan kelas data baru yang belum dikenali *output* kelasnya, namun memiliki ciri-ciri yang sama seperti apa yang dipelajari oleh algoritma. Sebelum algoritma tersebut mempelajari atribut-atribut, sebaiknya data yang digunakan dilakukan tahap *preprocessing* terlebih dahulu untuk membersihkan data, dan menghapus atau memodifikasi data yang memiliki *missing value* pada atribut tertentu. Kemudian, tahap selanjtunya setelah dilakukan proses *learning* adalah dilakukan proses *testing*. Dalam proses *testing*, dilakukan pengujian dari model klasifikasi yang sudah dibentuk pada proses pelatihan data, model yang sudah dibuat dilakukan pengujian dengan menggunakan *data testing*. *Data testing* adalah data yang emang digunakan untuk menguji performa model klasifikasi yang sudah dibuat. Biasanya saat melakukan import data, data yang digunakan akan dibangi menjadi dua bagian, terdiri dari *data training* yang nantinya akan digunakan untuk *proses learning*, dan *data testing* yang nantinya akan digunakan pada proses pengujian untuk menguji performa algoritma.

Beberapa algoritma yang digunakan dalam klasifikasi antara lain adalah algoritma *K-Nearest Neighbor*, algoritma *Decision Tree*, algoritma SVM,

algoritma *Random Forest*, dan algoritma XgBoost. Algoritma *K-Nearest Neighbor* adalah algoritma klasifikasi *machine learning* yang melakukan pengklasifikasian data yang digunakan berdasarkan jarak terdekat yang dimiliki oleh masing-masing data (Dewi, 2016). Dalam algoritma *K-Nearest Neighbor*, ketepatan algoritma tersebut dipengaruhi oleh ada tidaknya fitur yang dirasa relevan, atau jika bobot yang dimiliki fitur tersebut dirasa tidak setara dengan relevansi data yang lain terhadap klasifikasinya. Kemudian algoritma selanjutnya yang juga digunakan dalam klasifikasi *machine learning* adalah algoritma *Decision Tree*, algoritma ini adalah algoritma yang dibuat seperti pohon keputusan, dimana setiap cabang yang dimiliki menunjukan pilihan yang ada diantara sejumlah alternatif pilihan yang ada (Setiawati et al., 2016). Algoritma *Decision Tree* ini biasanya digunakan untuk pengambilan keputusan, algoritma ini dapat mengubah permasalahan yang tadinya bersifat kompleks, dengan menggunakan algoritma *Decision Tree* ini pengambilan keputusan dapat diubah menjadi lebih simple dan spesifik. Algoritma *Decision Tree* ini bekerja dengan menentukan sebuat *root node* atau titik awal yang nantinya akan dijadikan cabang pertama dalam pengambilan keputusan. Dari titik awal tersebut, nantinya akan dibuat beberapa cabang lagi berdasarkan nilai pembobotan cabang, yang akan berisi kemungkinan keputusan yang ada berdasarkan data. Salah satu contoh algoritmanya adalah algoritma C4.5, algoritma tersebutdibangun dengan cara membagi data secara rekursif hingga dari data tersebut terdiri dari beberapa bagian yang terdiri dari kelas yang sama. Algoritma C4.5 akan menentukan *root node* kemudian akan membagi kasus atau kemungkinan hasil keputusan yang ada ke dalam cabang, proses tersebut akan diulangin sampai semua kemungkinan keputusan yang ada pada cabang memiliki kelas yang sama (Ginting et al., 2014).

Kemudian terdapat algoritma SVM atau *Support Vector Machine*. Algoritma SVM adalah algoritma yang menggunakan ruang hipotesis yang berupa fungsi-fungsi linear yang ada di dalam sebuah fitur yang memiliki dimensi tinggi dan dilakukan pembelajaran dengan menggunakan teori optimasi (Puspitasari et al., 2018). Di dalam algoritm SVM ini, kinerja akurasi model yang dihasilkan dengan menggunakan algoritma SVM sangat bergantung dengan fungsi kernel apa yang dipakai dan parameter yang digunakan. Algoritma SVM dibagi menjadi dua, yang pertama adalah algoritma SVM Linear, dan algoritma SVM Non-Linear. Algoritma SVM Linear merupakan algoritma SVM yang dapat memisahkan kedua class label target pada *hyperplane* dengan menggunakan *soft margin*. Sedangkan algoritma SVM Non-Linear adalah algoritma SVM yang menerapkan fungsi linear dari kernel trick terhadap ruang yang memiliki dimensi tinggi. Selanjutnya adalah algoritma *Random Forest*, algoritma tersebut adalah hasil pengembangan dari algoritma *Classification and Regression Tree* (CART) yang dikombinasikan dengan teknik *bootstrap aggregating* (bagging) dan teknik *random feature selection* (Ghani & Subekti, 2018). Algoritma tersebut cocok digunakan untuk klasifikasi pada data yang berjumlah besar, dan pada algoritma *Random Forest* tidak terdapat proses *pruning* atau pemangkasan variabel seperti yang terjadi pada

algoritma *Decision tree*. Jadi, pada algoritma *Random Forest* dalam membuat pohon keputusan tetap menggunakan semua atribut yang dimiliki, atau atribut yang memang digunakan dan sudah ditentukan pada saat proses *preprocessing* data. Pembentukan pohon keputusan yang terdapat di algoritma ini adalah dengan cara melakukan *training* sampel data. Klasifkasi akan dijalankan setelah semua pohon keputusan terbentu dan selanjutnya akan dilakukan proses *vote* untuk menentukan pohon terbaik. Pohon keputusan terbaik adalah pohon keputusan yang memenangkan proses *vote* tersebut. Selain itu, dalam klasifikasi *machine learning* terdapat algoritma XgBoost yang akan digunakan di dalam penelitian ini.

### 2.2.5. Algoritma XgBoost

Algoritma XgBoost adalah algoritma klasifikasi yang merupakan kelanjutan dari algoritma *gradient boosting*. Algoritma XgBoost menggunakan prinsip yang beranama *ensemble*, yang bekerja dengan cara menggabungkan beberapa pohon keputusan yang awalnya memiliki performa lemah yang akan dikuatkan untuk menjadi model klasifikasi yang kuat (Muslim, 2019). Beberapa kelebihan yang dimiliki oleh algoritma XgBoost diantaranya adalah, algoritma ini dapat melakukan pemrosesan secara *parallel* yang dapat mempercepat proses kerja komputasi, kemudian algoritma tersebut juga memiliki fleksibilitas yang tinggi dalam hal pengaturan objektif, selain itu algoritma XgBoost juga memiliki kelebihan dapat mengatasi *split* saat algoritma tersebut menemukan *negative loss*. Dengan kelebihan yang dimiliki oleh algoritma tersebut, algoritma XgBoost dinilai cocok untuk melakukan pekerjaan klasifikasi. Algoritma XgBoost bekerja dengan cara membuat pohon keputusan sebagai cara untuk melakukan klasifikasi pada *data train*, sehingga dengan cara tersebut dapat diperoleh target yang diinginkan.

Algoritma xgboost menggunakan *tree* atau pohon untuk membuat model klasifikasinya. Dalam membuat *tree* dalam xgboost dapat dilakukan dengan cara menghitung *simillarity score* dan menentukan *gain*. Nilai *gain* terbesar yang akan dijadikan *root node* di dalam *tree* tersebut. Rumus untuk menghitung *similarity socre* dan *gain* adalah sebagai berikut.

$$similarity = \frac{\sum(Residual_i)^2}{\sum[prev\ probability_i \times (1 - prev\ probability_i)] + \lambda}$$

Keterangan:

$Residual = actual\ value - propability$

$prev\ probability =$ *initial prediction* atau hasil prediksi saat ini untuk iterasi selanjutnya

selanjutnya, setelah menghitung nilai *similarity score*, kemudian dilakukan dengan menghitung nilai *gain*. Kemudian *node* yang memiliki nilai *gain* terbesar akan dijadikan sebagai *root node*. Begitu juga untuk menentukan *child node* maka kemungkinan *node-node* yang tersisa akan dihitung nilai *gain*nya, kemudian *node* yang memiliki nilai *gain* terbesarlah yang dipilih. Rumus untuk menentukan nilai gain adalah sebagai berikut.

$gain = left\ similarity + right\ similarity - root\ similarity$
keterangan:
$left\ similarity = leaf$ sebelah kiri atau pilihan *yes*
$right\ similarity = leaf$ sebelah kanan atau pilihan *no*
$root\ similarity = node$

Kemudian setelah menentukan nilai *gain* untuk masing-masing *node* dan membuat *tree* yang akan digunakan untuk memprediksi, kemudian *tree* yang sudah dibuat dapat dipangkas bila memungkinkan. Cara memangkas *leaf* adalah dengan menghitung nilai *cover* dari masing-masing *leaf*, kemudian ditentukan syarat minimal dari nilai *cover* tersebut, apabila nilai *cover* lebih kecil dari nilai syarat tersebut maka *leaf* tersebut akan dipangkas atau dihapus. Nilai *default cover* adalah 1, sehingga apabila terdapat *leaf* yang memiliki nilai *cover* kurang dari 1 maka *leaf* tersebut dapat dipangkas. Sedangkan untuk memangkas *node*, kita dapat menentukan nilai *gamma*($\gamma$), apabila nilai ($gain - \gamma$) adalah negatif maka *node* tersebut dapat dipangkas atau dihapus. Rumus untuk menghitung *cover* adalah sebagai berikut.

$cover = \sum[prev\ probability_i \times (1 - prev\ probability_i)]$
Keterangan:
$prev\ probability =$ *initial prediction* atau hasil prediksi saat ini untuk iterasi selanjutnya

setelah menghitung nilai *cover* dan memangkas *leaf* apabila diperlukan, dan sudah membuat *tree* yang akan digunakan untuk melakukan prediksi. Selanjutnya adalah menghitung nilai *output value* untuk masing-masing *leaf*. *Output value* ini yang nantinya akan digunakan untuk perhitungan menentukan prediksi. Rumus untuk menghitung output value adalah sebagai berikut.

$$O_{value} = \frac{\sum(Residual_i)}{\sum[prev\ probability_i \times (prev\ probability_i)] + \lambda}$$
keterangan:
$Residual = actual\ value - probability$
$prev\ probability =$ *initial prediction* atau hasil prediksi saat ini untuk iterasi selanjutnya

kemudian, setelah menghitung *output value* untuk masing-masing *leaf*. Model klasifikasi dengan 1 *tree* sudah dibuat dan dapat dilakukan untuk menghitung prediksi dan melakukan iterasi selanjutnya untuk membuat *tree* baru sesuai dengan nilai *probability* dan *redisual* yang baru. Rumus untuk menghitung prediksi adalah sebagai berikut.

$$p = initial\ probability$$

$$odds = \frac{p}{1-p}$$

$$\log(odds) = \log\left(\frac{p}{1-p}\right)$$

$$\log(odds)\ probability = \log\left(\frac{p}{1-p}\right) + \sum[\varepsilon \times O_{value}]_i$$

$$probability = \frac{e^{\log(odds)probability}}{1 + e^{\log(odds)probability}}$$

Keterangan:

$\varepsilon = learning\ rate$, secara deafult nilainya adalah 0.3

$initial\ probability =$ prediksi awal, biasanya 0.5

Kemudian setelah melakukan prediksi sesuai dengan dataset yang ada, kemudian didapatkan nilai *probability* baru dan nilai *residual* baru yang nantinya dapat digunakan untuk membuat *tree* untuk iterasi berikutnya. Dan pada iterasi berikutnya nilai dari *prev probability* bisa saja berbeda untuk masing-masing data.

### 2.2.6. Parameter Algoritma XgBoost

Hasil klasifikasi yang dilakukan dengan menggunakan XgBoost sangat bergantung dengan nilai dari parameter-parameter yang digunakan. Terdapat beberapa parameter di dalam algoritma XgBoost, diantaranya adalah sebagai berikut.

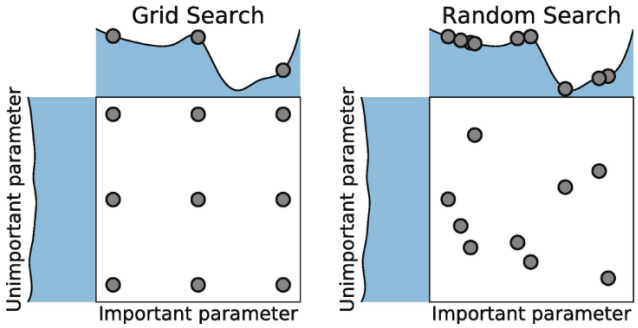
| Parameter | Keterangan |
|---|---|
| Eta | *Learning rate* pada proses *training* |
| Gamma | Nilai parameter *penalty* pada *regularization* |
| Max_depth | Kedalaman pohon yang digunakan untuk klasifikasi |
| Min_child_weight | Nilai bobot minimal (*cover*) yang dibutuhkan *child node* |
| Subsample | Jumlah persenan sebagian data yang digunakan untuk *training* |
| Colsample_bytree | Jumlah sample kolom untuk membuat *tree* baru |

**Tabel 1.2** *Parameter XgBoost*

### 2.2.7. Randomized Search Optimizer

Algoritma XgBoost memiliki beberapa parameter yang harus diatur supaya menghasilkan model klasifikasi yang memiliki performa maksimal. Cara untuk menentukan parameter terbaik dapat menggunakan *grid search optimizer* ataupun *randomized search optimizer*. Algoritma *grid search* akan mencoba semua kombinasi nilai parameter yang dituliskan dalam *hyperparameter tuning* tersebut, sedangkan algoritma *random search* hanya akan mencoba *range* dari nilai parameter yang dituliskan, sejumlah dengan banyaknya kombinasi yang sudah kita tentukan (Syukron et al., 2020). Kemudian *random search* akan mencoba nilai parameter tersebut dan akan menentukan nilai yang menghasilkan akurasi model terbaik yang nantinya akan dijadikan nilai parameter dalam model klasifikasi. Algoritma *randomized search* dinilai lebih baik daripada algoritma *grid search* dalam menentukan parameter terbaik, karena lebih efektif dapat menemukan nilai parameter lebih cepat tanpa harus mencoba seluruh pilihan yang diberikan. Untuk

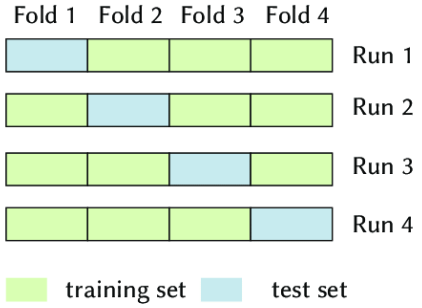
lebih jelas, berikut terdapat ilustrasi perbandingan *grid search* dan *randomized search optimizer* dalam menentukan nilai parameter.




**Gambar 2.0** *grid search dan random search*

### 2.2.8. Cross Validation

Untuk memastikan model klasifikasi yang sudah dibuat, perlu dilakukan validasi dengan cara *data training* dipecah menjadi beberapa bagian untuk dilakukan pengujian pada masing-masing bagian, metode tersebut dinamakan *cross validation*. Metode ini akan membagi data ke dalam beberapa bagian dan memastikan semua bagian akan mendapatkan kesempatan untuk menjadi data *training* dan data *testing*. K-*fold cross validation* adalah *cross validation* yang akan membagi data menjadi sejumlah *k* bagian atau *fold* di mana setiap *fold* akan digunakan sebagai data pengujian (Peryanto et al., 2020). Sebagai contoh seperti gambar dibawah ini. Apabila kita atur nilai *k*=4 maka data akan dibagi menjadi 4 bagian secara acak dan sama jumlahnya untuk masing-masing bagian. Kemudian masing-masing bagian akan menjadi *data test* untuk dilakukan pengujian untuk memvalidasi performa model yang dibuat, jumlah pengujian sesuai dengan jumlah banyaknya data. Oleh karena itu jumlah iterasi pengujian juga dipengaruhi oleh jumlah *k* yang diatur, semakin banyak jumlah *k* maka iterasi pengujian juga semakin banyak, dan memerlukan waktu yang lebih lama.




**Gambar 2.1** *K-fold cross validation*

### 2.2.9. Confusion Matrix

*Confusion Matrix* adalah alat evaluasi secara visual yang biasanya digunakan pada *supervised learning* (Novandya, 2017). *Confusion matrix* merupakan sebuah metode evaluasi yang biasanya digunakan untuk menghitung akurasi pada data mining dan klasifikasi (Dewi, 2016). Sedangkan pengertian dari akurasi sendiri adalah keakuratan keseluruhan prediksi (Syukron et al., 2020). *Confusion matrix*

berbentuk tabel, memiliki kolom yang merepresentasikan prediksi dan baris yang merepresentasikan fakta. Dari *confusion matrix* dapat dihasilkan beberapa nilai evaluasi seperti *accuracy*, *recall*, *precison*, dan *F1 score*.

**Tabel 1.3** Tabel *confusion matrix*

| | | **Prediksi** | |
|---|---|---|---|
| **Fakta** | | **0** | **1** |
| | **0** | TN | FP |
| | **1** | FN | TP |

a. TN: adalah sejumlah data dengan fakta **salah** yang diprediksi **salah.**
b. FP: adalah sejumlah data dengan fakta **salah** yang diprediksi **benar.**
c. FN: adalah sejumlah data dengan fakta **benar** yang diprediksi **salah.**
d. TP: adalah sejumlah data dengan fakta **benar** yang diprediksi **benar.**

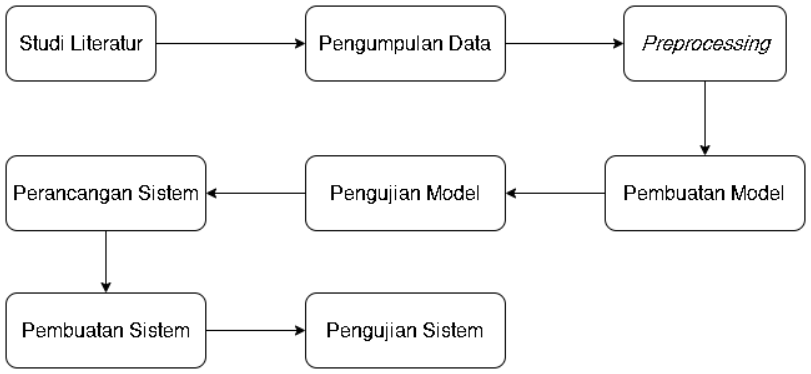
Dari tabel *confusion matrix* tersebut dapat dihasilkan beberapa perhitungan nilai evaluasi, diantaranya adalah sebagai berikut.

1. *Accuracy* = $\frac{TP+TN}{TP+FP+FN+TN}$
2. *Recall* = $\frac{TP}{TP+FN}$
3. *Precision* = $\frac{TP}{TP+FP}$
4. *F1 score* = $2 * \frac{recall*precision}{recall+precision}$

# BAB III
# METODOLOGI PENELITIAN

## 3.1.Metode Penelitian

Metode penelitian yang digunakan penulis di dalam penelitian ini adalah penelitian kuantitatif. Sedangkan jenis penelitian yang akan dilakukan penulis dalam penelitian ini adalah penelitian implementatif, penelitian ini memuat perancangan dan pengembangan. Perancangan yang dilakukan adalah membuat sistem prediksi penyakit jantung yang dibuat menggunakan bahasa pemrograman python yang akan ditampilkan dalam bentuk aplikasi berbasis web yang dapat dibuka menggunakan *browser*, sedangkan pengembangan yang dilakukan dalam penelitian ini adalah pembuatan model *machine learning* yang akan digunakan untuk melakukan klasifikasi pada prediksi penyakit jantung. Model tersebut dibuat menggunakan algoritma XgBoost dan menggunakan *randomized search optimizer* untuk membantu menentukan parameter terbaik yang diperlukan oleh model supaya nantinya menghasilkan model klasifikasi terbaik yang memiliki performa yang maksimal. Pemilihan algoritma XgBoost untuk membuat model klasifikasi karena algoritma tesebut dinilai lebih baik dan lebih cepat dalam hal komputasi dan dapat menghindari *overfitting* karena pada algoritma tersebut juga memiliki *regularization*.




**Gambar 3.1** *Tahapan Penelitian*

## 3.2.Studi Literatur

Studi literatur adalah kegiatan yang dilakukan dengan melakukan pencarian informasi mengenai topik yang bersangkutan lewat sumbersumber referensi seperti jurnal, buku, dan sumber-sumber lain yang dapat dipercaya. Dalam tahapan ini, peneliti mencari dan mengumpulkan informasi mengenai penelitian-penelitian sebelumnya yang berkaitan dengan prediksi penyakit jantung dan penelitian sebelumnya yang dilakukan menggunakan algoritma XgBoost. Pengumpulan informasi tersebut dilakukan untuk mengetahhui perkembangan topik prediksi penyakit jantung seperti algoritma apa saja yang sudah sering digunakan untuk menyelesaikan kasus prediksi tersebut, dan juga untuk mengetahui kelebihan dan kekurangan algoritma lain dalam meyelesaikan kasus prediksi penyakit jantung. Yang nantinya dijadikan bahan evalusai dan pertimbangan penelitian ini dilakukan dengan algoritma yang dipilih, yaitu algoritma XgBoost.

## 3.3.Pengumpulan Data

Data yang akan digunakan dalam penelitian ini adalah data dengan jenis data sekunder. Data sekunder adalah data yang bisa didapatkan secara tidak langsung, misalnya didapatkan lewat orang lain ,lewat sumber dokumen yang sudah memuat data tersebut, melalui website, dan sumber data lainnya (Haqie et al., 2020). Data yang akan digunakan di dalam penelitian ini adalah data *Clevaland Heart Disease* yang didapatkan dengan cara mengunduh data pada *repository* resminya yaitu bersumber dari UCI *machine learning* ataupun dapat juga diunduh dari situs kaggle. Data yang akan digunakan berjumlah sebanyak 303 data dengan rincian 138 data tergolong dalam kelas tidak memiliki penyakit jantung atau sehat, dan sebanyak 165 data tergolong ke dalam memiliki penyakit jantung. Data tersebut memiliki 13 kolom yang memuat atribut data yang akan digunakan untuk klasifikasi. Untuk lebih jelasnya, penjelasan mengenai kolom yang ada pada datasets terdapat pada tabel di bawah ini.

**Tabel 3.1** *Kolom pada datasets*

| Atribut | Deskripsi | Keterangan |
|---|---|---|
| *Age* | Umur pasien | *Numerik* |
| *Sex* | Jenis kelamin pasien | 0: Wanita, 1: Pria |
| *Cp* | *Chest pain type* | 1:*typical angina*, 2:*atypical angina*, 3:*non-angina pain*, 4:*asympotomatic* |
| *Trestbps* | *Resting blood presure* | *Numerik* |
| *Chol* | Serum kolesterol | *Numerik* |
| *Fbs* | *Fasting blood sugar*>120 mg/dl | 0:*false*, 1:*true* |
| *Restecg* | Hasil ECG selama istirahat | 0:normal, 1:abnormal (memiliki kelainan gelombang ST-T), 2:*hipertrofil ventrikel* |
| *Thalac* | Detak jantung maksimal yang dicapai | *Numerik* |
| *Exang* | Ukuran *boolean* yang menunjukan apakah latihan angina industri terjadi | 0:*No*, 1:*Yes* |
| *Oldpeak* | *Segment* ST yang diperoleh dari latihan relatif terhadap istirahat | *Numerik* |
| *Slope* | Kemiringan segmen ST untuk latihan maksimal (puncak) | 1:*upsloping*, 2:*flat*, 3:*downsloping* |
| *Ca* | Jumlah *vessel* utama yang diwarnai oleh *fluroskopi* | 0, 1, 2, dan 3 |
| *Thal* | *Thal* | 3:*normal*, 6:cacat tetap, 7:cacat *reversible* |

## 3.4.Pengolahan Data Awal

Dataset yang akan digunakan setelah diunduh sebelum dilakukan proses *training* pembuatan model dilakukan proses *preprocessing*. Preprocessing data adalah sebuah proses yang dilakukan dengan mengubah data ke dalam format data yang lebih sederhana, lebih efektif, sesuai dengan kebutuhan yang ingin digunakan pengguna (Saifullah et al., 2017). *Preprocessing* yang akan dilakukan pada data yang digunakan antara lain adalah membersihkan *outlier* data, apabila data yang digunakan memiliki

nilai terlalu kecil atau terlalu besar dari dominan rentang datanya, maka data *outlier* tersebut dapat dihilangan supaya data lebih baik dan menghasilkan model yang lebih baik pula. Selanjutnya adalah dilakukan normalisasi pada data *numerik*, apabila nilai-nilai yang ada pada data dengan tipe data *numerik* tidak rata maka diperlukan proses normalisasi sehingga nilai-nilai tersebut memiliki rentang yang sesuai dengan data lainnya. Kemudian pada data kategorikal dilakukan proses *one hot encoding*, yaitu membuat data kategorikal menjadi kolom baru supaya dapat dipelajari oleh komputer. Kemudian setelah data tersebut dilakukan proses *preprocessing*, maka data dapat masuk ke *pipeline* program yang kemudian digunakan untuk pembuatan model klasifikasi pada tahap selanjutnya.

## 3.5.Analisis kebutuhan

Analisis kebutuhan dapat didefinisikan sebuah proses yang dilakukan untuk mencari dan munculkan perbedaan antara tujuan ideal yang ada dan ekspektasi tujuan yang kita harapkan (Briggs, 1991). Definisi lain dari analisis kebutuhan adalah sebuah proses pengumpulan informasi mengenai ketidakseimbangan yang ada dan menentukan prioritas untuk dipecahkan (Sanjaya, 2008). Dalam subbab ini menjelaskan tentang apa aja yang dibutuhkan dan dilakukan oleh sistem yang dibuat. Analisis kebutuhan ini dilakukan untuk mengumpulkan data dan digunakan untuk pengambilan keputusan proses apa saja yang akan terlibat dan menentukan tujuan dari sistem yang dibuat. Terdapat dua jenis kebutuhan, yaitu kebutuhan fungsional dan kebutuhan non fungsional.

### 3.5.1. Kebutuhan Fungsional

Kebutuhan fungsional adalah informasi mengenai apa saja yang harus ada atau yang akan dilakukan sistem tersebut. Beberapa kebutuhan fungsional yang ada di dalam sistem prediksi ini antara lain adalah:

a. Sistem dapat menerima input data sesuai dengan parameter yang ada.
b. Sistem dapat melakukan prediksi berdasarkan data parameter yang diinput oleh user.
c. Sistem dapat menampilkan hasil prediksi.

### 3.5.2. Kebutuhan Non Fungsional

Kebutuhan non fungsional adalah kebutuhan yang melibatkan perilaku sistem. Dalam subbab ini akan menjelaskan spefisikasi perangkat yang digunakan untuk membuat sistem, seperti spefisikasi perangkat keras (*hardware*) ataupun perangkat lunak (*software*) yang digunakan.

a. Analisis perangkat keras (*hardware*)

Perangkat keras atau *hardware* yang digunakan untuk membuat sistem prediksi. Spesifikasi *hardware* dapat dilihat pada tabel dibawah ini.

**Tabel 3.2** *Spesifikasi perangkat keras*

| No | Perangkat Keras | Keterangan |
|---|---|---|
| 1. | Processor | Intel i7-7700HQ |
| 2. | RAM | 8GB DDR5 |
| 3. | Storage | 1TB |
| 4. | Graphic Card | Nvidia GeForce GTX1050 4GB |
| 5. | Perangkat input dan output | Keyboard, mouse, monitor |

| | | |
|---|---|---|
| 6. | Koneksi Internet | Wifi |

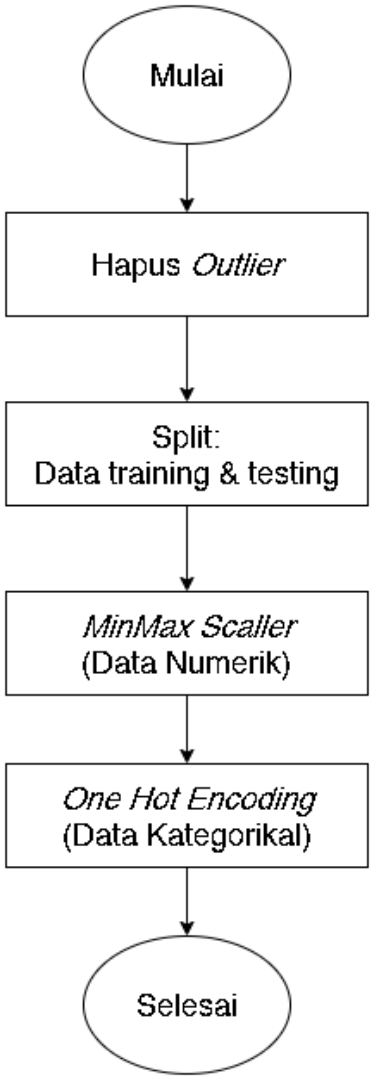
b. Analisis kebutuhan perangkat lunak (*software*)
Perangkat lunak atau software yang digunakan untuk membuat sistem prediksi ini.

### 3.6.Pembuatan Model Klasifikasi

#### 3.6.1. Preprocessing

Dalam preprocessing dilakukan beberapa tahapan diantaranya adalah:

1. Menghapus *outlier* atau pencilan.
2. Pembagian data menjadi *data training* dan *data testing*.
3. Normalisasi pada data numerik.
4. Dilakukan *one hot encoding* pada data kategoris.




**Gambar 3.2** *Flowchart Preprocessing*

Tahap pertama yang dilakukan adalah menghapus *outlier* data atau pencilan, outlier adalah data ekstrem yang mempunyai nilai maksimal atau minimal yang berbeda dengan dominan data lainnya. *Outlier* dinilai menggangu karena dapat membuat data

menjadi tidak stabil, oleh karena itu perlu dihapus. Perhitungan untuk menghapus outlier dapat dilihat dibawah ini.

IQR = Q3 – Q1
Batas *outlier* bawah = Q1 – (1,5 * IQR)
Batas *outlier* atas = Q3 + (1,5 * IQR)
Keterangan:
Q1 = $X\,\frac{1*(n+1)}{4}$
Q3 = $X\,\frac{3*(n+1)}{4}$

Contoh data yang akan dihitung adalah seperti dibawah ini.

**Tabel 3.3** *Contoh Menghapus Outlier*

| No | Data Asli (trestbps) | Data Setelah Diurutkan |
|---|---|---|
| 1. | 145 | 94 |
| 2. | 130 | 94 |
| 3. | 130 | 100 |
| 4. | 120 | 100 |
| 5. | 120 | 100 |
| 6. | 140 | 100 |
| 7. | 140 | 101 |
| .. | ... | ... |
| 297. | 124 | 178 |
| 298. | 164 | 178 |
| 299. | 140 | 180 |
| 300. | 110 | 180 |
| 301. | 144 | 180 |
| 302. | 130 | 192 |
| 303. | 130 | 200 |

Dari tabel diatas dapat dilakukan perhitungan menghapus outlier berikut:
Q1 = $X\,\frac{1*(303+1)}{4}$ = $X\,\frac{304}{4}$ = X ke-76 = 120
Q3 = $X\,\frac{3*(303+1)}{4}$ = $X\,\frac{912}{4}$ = X ke 228 = 140
IQR = Q3 – Q1 = 140 – 120 = 20
Batas *outlier* bawah = Q1 – (1,5 * IQR) = 120 – (1,5*20) = 90
Batas *outlier* atas = Q3 + (1,5 * IQR) = 140 + (1,5*IQR) = 170
Sesuai dengan perhitungan diatas, kemudian dapat dilakukan *filter* untuk data yang memiliki nilai >= 90 dan data yang memiliki nilai <= 170.

Pembagian datasets dilakukan untuk membagi data menjadi *data training* dan *data testing*. Pembagian data tersebut menggunakan komposisi 80:20 yaitu 80% akan menjadi *data training* dan 20% akan menjadi *data testing*.
Kemudian setelah dilakukan proses *splitting* atau pembagian data, selanjutnya dilakukan proses normalisasi pada data numerik. Proses normalisasi dilakukan supaya proses *training* menjadi lebih cepat, karena dapat memudahkan model dalam

memahami data (Hanifa et al., 2017). Normalisasi dalam proses ini dilakukan dengan menggunakan *MinMax*, yaitu merubah data ke dalam range 0 untuk data terkecil dan 1 untuk data terbesar. Rumus perhitungan metode normalisasi *MinMax* dapat dilihat dibawah ini.

$$X_{new} = \frac{X - X_{min}}{X_{max} - X_{min}} \text{.....}$$

Keterangan:

$X_{new}$ = nilai baru

$X$ = nilai lama

$X_{min}$ = nilai minimal / terendah

$X_{max}$ = nilai maksimal / paling tinggi

Contoh perhitungan normalisasi menggunakan data dibawah ini.

**Tabel 3.4** *Perhitungan normalisasi MinMax*

| No. | Data lama | Data Baru |
|---|---|---|
| 1. | 145 | 0.67 |
| 2. | 130 | 0.47 |
| 3. | 130 | 0.47 |
| 4. | 120 | 0.34 |
| 5. | 120 | 0.34 |
| 6. | 140 | 0.6 |
| 7. | 140 | 0.6 |
| 8. | 120 | 0.34 |
| 9. | 150 | 0.73 |
| 10. | 140 | 0.6 |
| ... | ... | |
| 287. | 140 | 0.6 |
| 288. | 124 | 0.39 |
| 289. | 164 | 0.92 |
| 290. | 140 | 0.6 |
| 291. | 110 | 0.21 |
| 292. | 144 | 0.65 |
| 293. | 130 | 0.47 |
| 294. | 130 | 0.47 |

Keterangan:

$X_{min}$ = 94

$X_{max}$ = 170

Contoh perhitungan pada data ke-1:

$$X_{new} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

$$X_{new} = \frac{145 - 94}{170 - 94}$$

$$X_{new} = \frac{51}{76}$$

$$X_{new} = 0{,}67$$

Kemudian setelah dilakukan normalisasi menggunakan *MixMax* supaya nilai numerik dalam kolom tersebut memiliki nilai 0-1, kemudian dilakukan proses *one hot encoding* pada kolom dengan data kategorikal. *One hot encoding* adalah sebuah proses *encoding* yang digunakan untuk merepresentasikan nilai dalam data kategorikal menjadi data *binary* yang terdiri dari nilai 0 dan 1. Cara kerja dari metode *one hot encoding* adalah membuat kolom baru sebanyak jumlah kategori yang ada, kemudian mengisi dengan nilai 1 apabila data tersebut sesuai dengan kolom yang dimaksud. Contoh metode *one hot encoding* dapat dilihat pada tabel dibawah ini.

**Tabel 3.5** *Contoh data one hot encoding*

| **No.** | **Slope** |
|---|---|
| 1. | 0 |
| 2. | 0 |
| 3. | 2 |
| 4. | 2 |
| 5. | 2 |
| 6. | 1 |

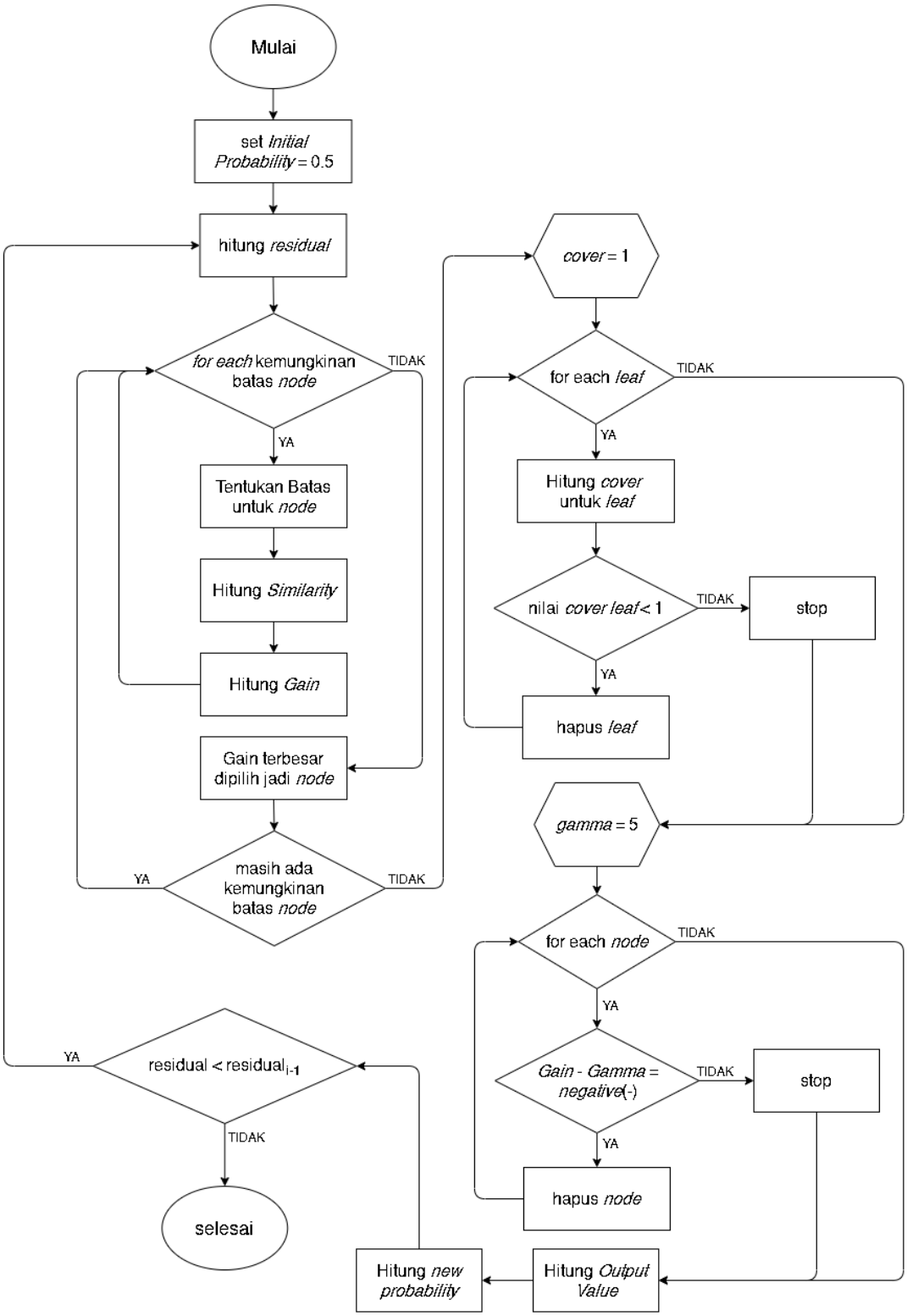
**Tabel 3.6** *Data hasil one hot encoding*

| No. | Slope_0 | Slope_1 | Slope_2 |
|---|---|---|---|
| 1. | 1 | 0 | 0 |
| 2. | 1 | 0 | 0 |
| 3. | 0 | 0 | 1 |
| 4. | 0 | 0 | 1 |
| 5. | 0 | 0 | 1 |
| 6. | 0 | 1 | 0 |

Data diatas adalah contoh dari metode *one hot encoding* pada kolom *slope*, dimana kolom yang baru akan bertambah sejumlah kategori yang ada dan kemudian nilai dari masing-masing kolom hanyalah 1 atau 0 sesuai dengan kategori pada masing-masing data menyesuaikan kolom yang ada.

### 3.6.2. Training

Proses training yang dilakukan oleh algoritma xgboost adalah sebagai berikut.




**Gambar 3.3** *Flowchart umum xgboost*

Sesuai dengan gambar *flowchart* diatas, algoritma xgboost akan menentukan *initial prediction* atau *initial probability* sebagai prediksi awal dengan nilai yaitu 0,5.

Kemudian dilakukan perhitungan untuk mengetahui nilai *residual* dari masing-masing data dengan rumus berikut.

$$residual = actual\ value - probability$$

Setelah mengetahui nilai *residual* dari masing-masing data, kemudian dilakukan proses pembuatan *tree*. Tahap pertama adalah menentukan *root node* untuk *tree* tersebut, untuk menentukan *root note* dilakukan dengan membuat *tree* sebanyak kemungkinan *node* yang ada, kemudian dilakukan perhitungan untuk menentukan nilai *similarity* masing-masing *leaf*. Dari nilai *similarity* tersebut, kemudian dapat diketahui nilai *gain* dari *node* tersebut. node yang memiliki nilai gain terbesarlah yang akan dipilih untuk menjadi root node. Rumus untuk menghitung similarity dan gain adalah sebagai berikut.

$$similarity = \frac{\sum(residual_i)^2}{\sum[prev\ probability_i\ \times (1 - prev\ probability_i)] + \lambda}$$

$$gain = left\ similarity + right\ similarity - root\ similarity$$

Kemudian, setelah satu *tree* terbentuk dapat dilakukan pengecekan untuk memastikan apakah *tree* tersebut perlu dipangkas atau tidak dengan menggunakan *cover* dan $gamma(\gamma)$. Apabila nilai c*over leaf* tersebut lebih kecil dari nilai *cover* yang sudah ditentukan, maka *leaf* tersebut dapat dipangkas. Begitu juga dengan *node*, apabila nilai dari $(gain - \gamma)$ bernilai negative(-), maka *node* tersebut dapat dipangkas. Rumus untuk menentukan nilai *cover* untuk *leaf* adalah sebagai berikut.

$$cover = \sum[prev\ probability_i\ \times (1 - prev\ probability_i)]$$

Selanjutnya, setelah *tree* terbuat dan *tree* juga sudah dilakukan pengecekan apakah memungkinkan untuk dipangkas atau tidak, satu tree sudah dibuat. Tahap selanjutnya dapat dilakukan dengan menghitung *output value* untuk masing-masing *leaf* dengan rumus berikut.

$$O_{value} = \frac{\sum(residual_i)}{\sum[prev\ probability_i\ \times (1 -\ prev\ probability_i)] +\ \lambda}$$

Setelah menentukan $O_{value}$, kemudian dilakukan perhitungan untuk menentukan *new probability* yang nantinya akan digunakan untuk menghitung *residual* baru dan membuat *tree* pada iterasi selanjutnya. Rumus untuk menentukan *probability* adalah sebagai berikut.

$$\log(odds) = \log\left(\frac{p}{1-p}\right) + \sum[\varepsilon\ \times\ O_{value}]_i$$

$$probability = \frac{e^{\log(odds)}}{1 +\ e^{\log(odds)}}$$

Contoh dataset yang akan digunakan untuk proses training adalah sebagai berikut.

**Tabel 3.7** *contoh datasest untuk training*

| Cp | Thalac | Target |
|---|---|---|
| 3 | 150 | 1 |
| 3 | 190 | 1 |
| 0 | 96 | 0 |
| 1 | 174 | 0 |
| 1 | 202 | 1 |

Pertama, buat *initial probability* dengan nilai 0,5. Kemduian dapat dihitung nilai *residual* masing-masing data dengan rumus $actual\ value - probability$. Kemudian buat tabel untuk memperbarui data nilai *residual* yang sudah dihitung. Tabel yang menampilkan nilai *residual* ditampilkan pada tabel di bawah ini.

**Tabel 3.8** *tabel dengan nilai residual*

| Cp | Thalac | Target | Residual |
|---|---|---|---|
| 3 | 150 | 1 | 0.5 |
| 3 | 190 | 1 | 0.5 |
| 0 | 96 | 0 | -0.5 |
| 1 | 174 | 0 | -0.5 |
| 1 | 202 | 1 | 0.5 |

Kemudian nilai *residual* tersebut, diasumsikan sebagai leaf dan dihitung *similarity score*nya yang nantinya akan dijadikan *root similarity* pada saat perhitungan nilai *gain* untuk masing-masing kemunkinan *node* yang ada. Perhitungan *similarity* adalah sebagai berikut. Untuk contoh di atas, kita asumsikan nilai lambda($\lambda$) adalah 0.

$$Root_{similarity} = \frac{(0.5 + 0.5 - 0.5 - 0.5 + 0.5)^2}{\Big((0.5 \times (1 - 0.5)) \times 5\Big) + 0}$$

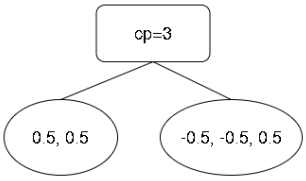
$$Root_{similarity} = \frac{(0.5)^2}{0.25 \times 5}$$

$$Root_{similarity} = \frac{0.25}{1.25}$$

$$Root_{similarity} = 0.2$$

Kemudian, kita lakukan perhitungan untuk mengetahui nilai gain pada masing-masing kemungkinan *node*, nilai *gain* yang paling tinggi yang nantinya akan dijadikan *node*. Beberapa kemungkinan *node* adalah sebagai berikut.

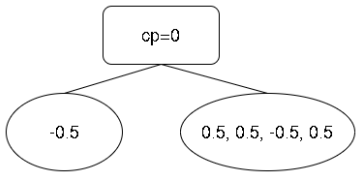
1. Cp = 3




**Gambar 3.4** *tree untuk node cp=3*

$$Left_{similarity} = \frac{(0.5 + 0.5)^2}{((0.5 \times (1 - 0.5)) \times 2) + 0}$$
$$Left_{similarity} = \frac{1^2}{(0.25 \times 2) + 0}$$
$$Left_{similarity} = \frac{1}{0.5}$$
$$Left_{similarity} = 2$$
$$Right_{similarity} = \frac{(-0.5 - 0.5 + 0.5)^2}{\left(\left(0.5 \times (1 - 0.5)\right) \times 3\right) + 0}$$
$$Right_{similarity} = \frac{(-0.5)^2}{(0.25 \times 3)}$$
$$Right_{similarity} = \frac{0.25}{0.75}$$
$$Right_{similarity} = 0.33$$
$$gain = Left_{similarity} + Right_{similarity} - Root_{similarity}$$
$$gain = 2 + 0.33 - 0.2$$
$$gain = 2.13$$

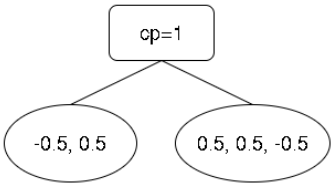
2. Cp = 0




**Gambar 3.5** *tree untuk node cp=0*

$$Left_{similarity} = \frac{(-0.5)^2}{\left(0.5 \times (1 - 0.5)\right) + 0}$$
$$Left_{similarity} = \frac{0.25}{0.25}$$
$$Left_{similarity} = 1$$
$$Right_{similarity} = \frac{(0.5 + 0.5 - 0.5 + 0.5)^2}{\left(\left(0.5 \times (1 - 0.5)\right) \times 4\right) + 0}$$
$$Right_{similarity} = \frac{(0.5 + 0.5)^2}{(0.25 \times 4) + 0}$$
$$Right_{similarity} = \frac{1}{1}$$
$$Right_{similarity} = 1$$
$$gain = Left_{similarity} + Right_{similarity} - Root_{similarity}$$
$$gain = 1 + 1 - 0.2$$
$$gain = 1.8$$

3. Cp = 1

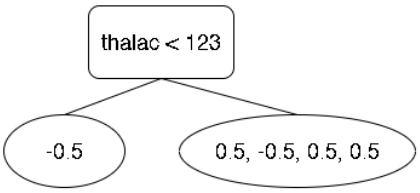



**Gambar 3.6** *tree untuk node cp=1*

$$Left_{similarity} = \frac{(-0.5 + 0.5)^2}{\left(\left(0.5 \times (1 - 0.5)\right) \times 2\right) + 0}$$
$$Left_{similarity} = \frac{0^2}{0.25 \times 2}$$
$$Left_{similarity} = 0$$
$$Right_{similarity} = \frac{(0.5 + 0.5 - 0.5)^2}{\left(\left(0.5 \times (1 - 0.5)\right) \times 3\right) + 0}$$
$$Right_{similarity} = \frac{0.5^2}{0.25 \times 3}$$
$$Right_{similarity} = \frac{0.25}{0.75}$$
$$Right_{similarity} = 0.33$$
$$gain = Left_{similarity} + Right_{similarity} - Root_{similarity}$$
$$gain = 0 + 0.33 - 0.2$$
$$gain = 0.13$$

4. Thalac < 123

   Untuk menentukan batas = $96 + (\frac{150-96}{2})$



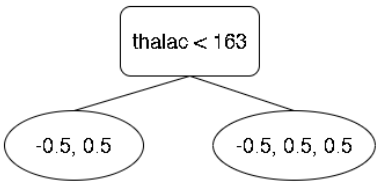

**Gambar 3.7** *tree untuk node thalac<123*

$$Left_{similarity} = \frac{(-0.5)^2}{\left(0.5 \times (1 - 0.5)\right) + 0}$$
$$Left_{similarity} = \frac{0.25}{0.25}$$
$$Left_{similarity} = 1$$
$$Right_{similarity} = \frac{(0.5 - 0.5 + 0.5 + 0.5)^2}{\left(\left(0.5 \times (1 - 0.5)\right) \times 4\right) + 0}$$
$$Right_{similarity} = \frac{(0.5 + 0.5)^2}{0.25 \times 4}$$

$$Right_{similarity} = \frac{1}{1}$$
$$Right_{similarity} = 1$$
$$gain = Left_{similarity} + Right_{similarity} - Root_{similarity}$$
$$gain = 1 + 1 - 0.2$$
$$gain = 1.8$$

5. Thalac < 163

   Untuk menentukan batas = $150 + (\frac{174-150}{2})$

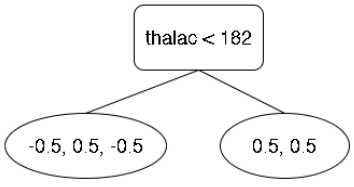



**Gambar 3.8** *tree untuk node thalac<163*

$$Left_{similarity} = \frac{(-0.5 + 0.5)^2}{\Big(\big(0.5 \times (1 - 0.5)\big) \times 2\Big) + 0}$$
$$Left_{similarity} = \frac{0^2}{0.25 \times 2}$$
$$Left_{similarity} = 0$$
$$Right_{similarity} = \frac{(-0.5 + 0.5 + 0.5)^2}{\Big(\big(0.5 \times (1 - 0.5)\big) \times 3\Big) + 0}$$
$$Right_{similarity} = \frac{0.5^2}{0.25 \times 3}$$
$$Right_{similarity} = \frac{0.25}{0.75}$$
$$Right_{similarity} = 0.33$$
$$gain = Left_{similarity} + Right_{similarity} - Root_{similarity}$$
$$gain = 0 + 0.33 - 0.2$$
$$gain = 0.13$$

6. Thalac < 182

   Untuk menentukan batas = $174 + (\frac{190-174}{2})$



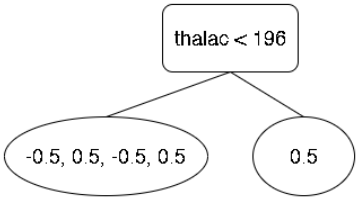

**Gambar 3.9** *tree untuk node thalac<182*

$$Left_{similarity} = \frac{(-0.5 + 0.5 - 0.5)^2}{\Big(\big(0.5 \times (1 - 0.5)\big) \times 3\Big) + 0}$$

$$Left_{similarity} = \frac{-0.5^2}{0.25 \times 3}$$
$$Left_{similarity} = \frac{0.25}{0.75}$$
$$Left_{similarity} = 0.33$$
$$Right_{similarity} = \frac{(0.5 + 0.5)^2}{\left(\left(0.5 \times (1 - 0.5)\right) \times 2\right) + 0}$$
$$Right_{similarity} = \frac{1^2}{0.25 \times 2}$$
$$Right_{similarity} = \frac{1}{0.5}$$
$$Right_{similarity} = 2$$
$$gain = Left_{similarity} + Right_{similarity} - Root_{similarity}$$
$$gain = 0.33 + 2 - 0.2$$
$$gain = 2.13$$

7. Thalac < 196

   Untuk menentukan batas = $190 + (\frac{202-190}{2})$

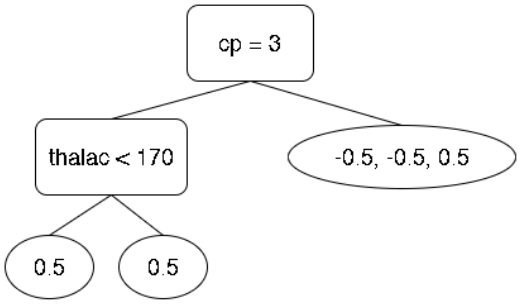



**Gambar 3.10** *tree untuk node thalac<196*

$$Left_{similarity} = \frac{(-0.5 + 0.5 - 0.5 + 0.5)^2}{\left(\left(0.5 \times (1 - 0.5)\right) \times 4\right) + 0}$$
$$Left_{similarity} = \frac{0^2}{1}$$
$$Left_{similarity} = 0$$
$$Right_{similarity} = \frac{0.5^2}{\left(0.5 \times (1 - 0.5)\right) + 0}$$
$$Right_{similarity} = \frac{0.25}{0.25}$$
$$Right_{similarity} = 1$$
$$gain = Left_{similarity} + Right_{similarity} - Right_{similarity}$$
$$gain = 0 + 1 - 0.2$$
$$gain = 0.8$$

Setelah semua kemungkinan *node* dihitung nilai *gain*, maka node yang memiliki nilai *gain* yang paling besar yang akan dijadikan *root node*. Sesuai dengan perhitungan di

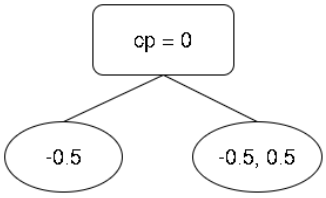
atas, maka yang akan menjadi *root node* adalah cp=3. Untuk *tree* sementara adalah sebagai berikut.




**Gambar 3.11** *tree sementara yang terbentuk*

Kemudian, kita masih melakukan hal yang sama untuk perhitungan kemungkinan *node* yang tersisa pada *leaf* yang ada. Perhitungan dan kemungkinan *node* yang ada adalah sebagai berikut.

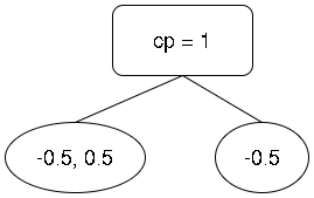
1. Cp=0




**Gambar 3.12** *tree untuk node cp=0*

$$Left_{similarity} = \frac{-0.5^2}{\left(0.5 \times (1-0.5)\right) + 0}$$
$$Left_{similarity} = \frac{0.25}{0.25}$$
$$Left_{similarity} = 1$$
$$Right_{similarity} = \frac{(-0.5+0.5)^2}{\left(\left(0.5 \times (1-0.5)\right) \times 2\right) + 0}$$
$$Right_{similarity} = \frac{0^2}{0.25 \times 2}$$
$$Right_{similarity} = 0$$
$$gain = Left_{similarity} + Right_{similarity} - Root_{similarity}$$
$$gain = 1 + 0 - 0.33$$
$$gain = 0.67$$

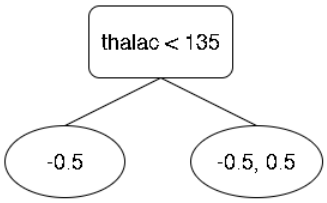
2. Cp=1




**Gambar 3.13** *tree untuk node cp=1*

$$Left_{similarity} = \frac{(-0.5 + 0.5)^2}{\left(\left(0.5 \times (1 - 0.35)\right) \times 2\right) + 0}$$
$$Left_{similarity} = \frac{0^2}{0.25 \times 2}$$
$$Left_{similarity} = 0$$
$$Right_{similarity} = \frac{-0.5^2}{\left(0.5 \times (1 - 0.5)\right) + 0}$$
$$Right_{similarity} = \frac{0.25}{0.25}$$
$$Right_{similarity} = 1$$
$$gain = Left_{similarity} + Right_{similarity} - Root_{similarity}$$
$$gain = 0 + 1 - 0.33$$
$$gain = 0.67$$

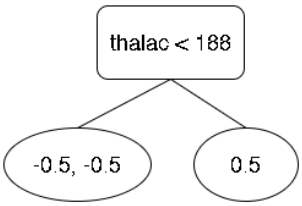
3. Thalac<135




**Gambar 3.14** *tree untuk node thalac<135*

$$Left_{similarity} = \frac{-0.5^2}{\left(0.5 \times (1 - 0.5)\right) + 0}$$
$$Left_{similarity} = \frac{0.25}{0.25}$$
$$Left_{similarity} = 1$$
$$Right_{similarity} = \frac{(-0.5 + 0.5)^2}{\left(\left(0.5 \times (1 - 0.5)\right) \times 2\right) + 0}$$
$$Right_{similarity} = \frac{0^2}{0.25 \times 2}$$
$$Right_{similarity} = 0$$
$$gain = Left_{similarity} + Right_{similarity} - Root_{similarity}$$
$$gain = 1 + 0 - 0.33$$
$$gain = 0.67$$

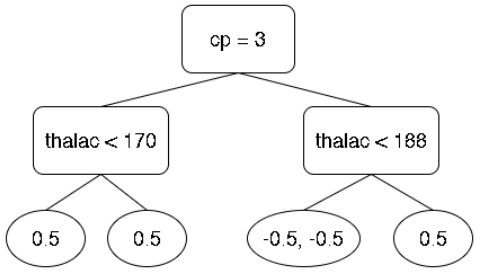
4. Thalac<188




**Gambar 3.15** *tree untuk node thalac<188*

$$Left_{similarity} = \frac{(-0.5 - 0.5)^2}{\left(0.5 \times \left((1 - 0.5)\right) \times 2\right) + 0}$$
$$Left_{similarity} = \frac{-1^2}{0.25 \times 2}$$
$$Left_{similarity} = \frac{1}{0.5}$$
$$Left_{similarity} = 2$$
$$Right_{similarity} = \frac{0.5^2}{(0.5 \times (1 - 0.5))}$$
$$Right_{similarity} = \frac{0.25}{0.25}$$
$$Right_{similarity} = 1$$
$$gain = Left_{similarity} + Right_{similarity} - Root_{similarity}$$
$$gain = 2 + 1 - 0.33$$
$$gain = 2.67$$

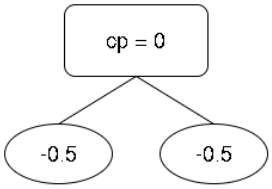
Setelah mengetahui nilai *gain* untuk semua kemungkinan *node* yang ada. *Node* yang memiliki nilai *gain* paling besar akan dijadikan *node*. Sesuai dengan perhitungan nilai *gain* di atas, maka *node* yang akan dipilih adalah *thalac*<188. Maka *tree* sementara yang dibuat adalah sebagai berikut.




**Gambar 3.16** *tree sementara yang sudah dibuat*

Kemudian kita masih menghitung lagi, kemungkinan *node* yang ada untuk kedalaman selanjutnya. Beberapa kemungkinan *node* yang ada dan perhitungannya adalah sebagai berikut.

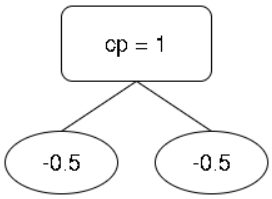
1. Cp=0




**Gambar 3.17** *tree untuk node cp=0*

$$Left_{similarity} = \frac{(-0.5)^2}{\left(0.5 \times (1-0.5)\right) + 0}$$
$$Left_{similarity} = \frac{0.25}{0.25}$$
$$Left_{similarity} = 1$$
$$Right_{similarity} = \frac{(-0.5)^2}{\left(0.5 \times (1-0.5)\right) + 0}$$
$$Right_{similarity} = \frac{0.25}{0.25}$$
$$Right_{similarity} = 1$$
$$gain = Left_{similarity} + Right_{similarity} - Root_{similarity}$$
$$gain = 1 + 1 - 2$$
$$gain = 0$$

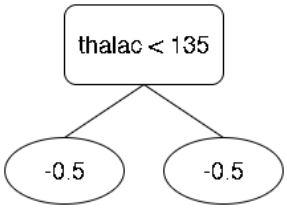
2. Cp=1




**Gambar 3.18** *tree untuk node cp=1*

$$Left_{similairty} = \frac{(-0.5)^2}{\left(0.5 \times (1-0.5)\right) + 0}$$
$$Left_{similarity} = \frac{0.25}{0.25}$$
$$Left_{similarity} = 1$$
$$Right_{similarity} = \frac{(-0.5)^2}{\left(0.5 \times (1-0.5)\right) + 0}$$
$$Right_{similarity} = \frac{0.25}{0.25}$$
$$Right_{similarity} = 1$$
$$gain = Left_{similarity} + Right_{similarity} - Root_{similarity}$$
$$gain = 1 + 1 - 2$$
$$gain = 0$$

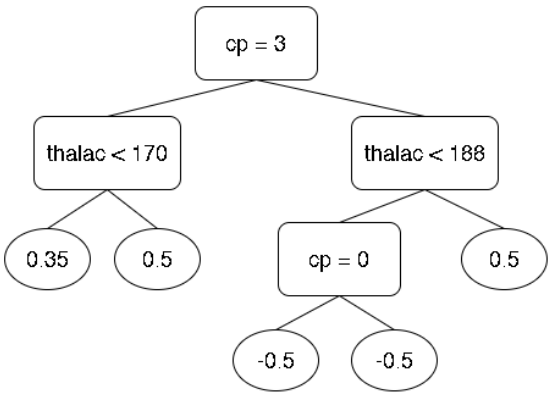
3. Thalac<135




**Gambar 3.19** *tree dengan node thalac<135*

$$Left_{similarity} = \frac{(-0.5)^2}{\big(0.5 \times (1-0.5)\big) + 0}$$
$$Left_{similarity} = \frac{0.25}{0.25}$$
$$Left_{similarity} = 1$$
$$Right_{similarity} = \frac{(-0.5)^2}{\big(0.5 \times (1-0.5)\big) + 0}$$
$$Right_{similarity} = \frac{0.25}{0.25}$$
$$Right_{similarity} = 1$$
$$gain = Left_{similarity} + Right_{similarity} - Root_{similarity}$$
$$gain = 1 + 1 - 2$$
$$gain = 0$$

Setelah mengetahui masing-masing nilai *gain* dari kemungkinan *node* yang ada, kemudian dipilih *node* dengan nilai gain yang paling tinggi. Namun karena semua kemungkinan *node* yang ada memberikan nilai *gain* yang sama, yaitu 0. Maka *node* yang dipilih adalah cp=0. Untuk hasil *tree* yang sudah dibuat adalah sebagai berikut.



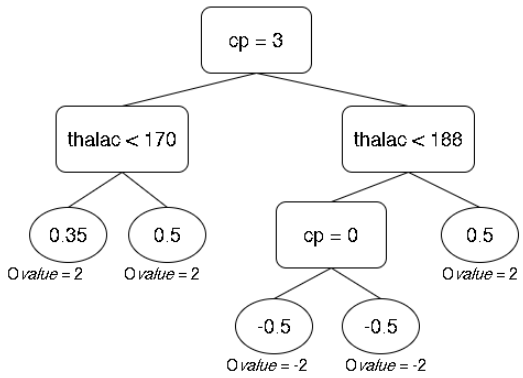

**Gambar 3.20** *hasil tree pertama yang dibuat*

Setelah kita berhasil membuat tree pertama. Langkah selanjutnya adalah melakukan pengecekan, sesuai dengan *flowchart* yang ada. Namun, dalam contoh proses training menggunakan algoritma xgboost ini, nilai *cover* dan *gamma*($\gamma$) kita asumsikan dengan nilai 0. Sehingga tidak ada *leaf* ataupun *node* yang dipangkas.

kemudian adalah menghitung $O_{value}$ untuk masing-masing leaf yang ada. Perhitungan $O_{value}$ untuk masing-masing *leaf* adalah sebagai berikut.

$$O_{value} = \frac{\sum(residual_i)}{\sum[prev\ probability_i \times (1 - prev\ probability_i)] + \lambda}$$

1. $O_{value} = \frac{0.5}{(0.5 \times (1-0.5))+0} = \frac{0.5}{0.25} = 2$
2. $O_{value} = \frac{0.5}{(0.5 \times (1-0.5))+0} = \frac{0.5}{0.25} = 2$
3. $O_{value} = \frac{-0.5}{(0.5 \times (1-0.5))+0} = \frac{-0.5}{0.25} = -2$
4. $O_{value} = \frac{-0.5}{(0.5 \times (1-0.5))+0} = \frac{-0.5}{0.25} = -2$
5. $O_{value} = \frac{0.5}{(0.5 \times (1-0.5))+0} = \frac{0.5}{0.25} = 2$




**Gambar 3.21** *tree dan* $O_{value}$

Setelah mengetahui nilai $O_{value}$ dari masing-masing *leaf*. Maka kita dapat menghitung nilai *probability* baru yang nantinya dapat kita gunakan untuk membuat tree kedua dengan menghitung nilai *residual* yang baru. Perhitungan *probability* adalah sebagai berikut dengan nilai *learning rate*($\varepsilon$) adalah 0.3.

$$\log(odds) = \log\left(\frac{p}{1-p}\right) + \sum[\varepsilon \times O_{valie}]_i$$

$$probability = \frac{e^{\log(odds)}}{1 + e^{\log(odds)}}$$

Keterangan:

$p = 0.5\ (initial\ probability)$

$$\log\left(\frac{p}{1-p}\right) = \log\left(\frac{0.5}{1-0.5}\right)$$

$$\log\left(\frac{p}{1-p}\right) = \log(1)$$

$$\log\left(\frac{p}{1-p}\right) = 0$$

1. $\log(odds) = 0 + (0.3 \times 2) = 0.6$

   $probability = \frac{e^{0.6}}{1 + e^{0.6}}$

   $probability = \frac{2.71828^{0.6}}{1 + 2.71828^{0.6}}$

   $probability = \frac{1.822}{2.822}$

   $probability = 0.65$
2. $\log(odds) = 0 + (0.3 \times 2) = 0.6$

   $probability = \frac{e^{0.6}}{1 + e^{0.6}}$

   $probability = 0.65$
3. $\log(odds) = 0 + (0.3 \times -2) = -0.6$

   $probability = \frac{e^{-0.6}}{1 + e^{-0.6}}$

   $probability = \frac{2.71828^{-0.6}}{1 + 2.71828^{-0.6}}$

   $probability = \frac{0.549}{1.549}$

   $probability = 0.35$
4. $\log(odds) = 0 + (0.3 \times -2) = -0.6$

   $probability = \frac{e^{-0.6}}{1 + e^{-0.6}}$

   $probability = 0.35$
5. $\log(odds) = 0 + (0.3 \times 2) = 0.6$

   $probability = \frac{e^{0.6}}{1 + e^{0.6}}$

   $probability = 0.65$

Setelah menentukan nilai *probability* untuk masing-masing data, maka kita dapat menghitung *residual* yang baru. Dan nantinya kita dapat membuat *tree* lagi untuk iterasi berikutnya. Data yang memuat nilai *residual* baru adalah sebagai berikut.

**Tabel 3.9** *tabel data terbaru dan residual*

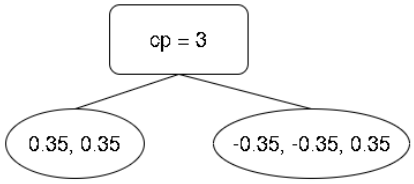
| Cp | Thalac | Traget | Probability | Residual |
|---|---|---|---|---|
| 3 | 150 | 1 | 0.65 | 0.35 |
| 3 | 190 | 1 | 0.65 | 0.35 |
| 0 | 96 | 0 | 0.35 | -0.35 |
| 1 | 174 | 0 | 0.35 | -0.35 |
| 1 | 202 | 1 | 0.65 | 0.35 |

Kemudian kita dapat melakukan pembuatan *tree* untuk iterasi kedua, proses yang dilakukan adalah sama seperti pembuatan *tree* pertama. Yang pertama dilakukan adalah mengasumsikan semua data *residual* sebagai satu *leaf* yang akan dihitung similaritynya sebagai $Root_{similarity}$. Proses perhitungan adalah sebagai berikut.

$$Root_{similarity}$$
$$= \frac{(0.35 + 0.35 - 0.35 - 0.35 + 0.35)^2}{\Big(\big(0.65 \times (1 - 0.65)\big) \times 3\Big) + \Big(\big(0.35 \times (1 - 0.35)\big) \times 2\Big) + 0}$$
$$= \frac{0.35^2}{0.6825 + 0.455}$$
$$= \frac{0.1225}{1.1375}$$
$$= 0.1076$$

Kemudian kita menghitung nilai *gain* dari kemungkinan *node* yang ada. *Node* yang memiliki nilai *gain* paling besar akan dijadikan sebagai *root node*. Perhitungan nilai *gain* terhadap kemungkinan *node* adalah sebagai berikut.

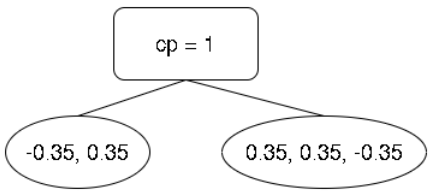
1. Cp=3




**Gambar 3.22** *tree kedua untuk node cp=3*

$$Left_{similaroty} = \frac{(0.35 + 0.35)^2}{\Big(\big(0.65 \times (1 - 0.65)\big) \times 2\Big) + 0}$$
$$Left_{similarity} = \frac{0.7^2}{0.2275 \times 2} = \frac{0.49}{0.455}$$
$$Left_{similarity} = 1.0769$$
$$Right_{similarity} = \frac{(-0.35 - 0.35 + 0.35)^2}{\big(0.65 \times (1 - 0.65)\big) + \Big(\big(0.35 \times (1 - 0.35)\big) \times 2\Big) + 0}$$
$$Right_{similarity} = \frac{-0.35^2}{0.2275 + 0.455} = \frac{0.1225}{0.6825}$$
$$Right_{similarity} = 0.1794$$
$$gain = Left_{similarity} + Right_{similarity} - Root_{similarity}$$
$$gain = 1.0769 + 0.1794 - 0.1076$$
$$gain = 1.1487$$

2. Cp=1




**Gambar 3.23** *tree kedua untuk node cp=1*

$$Left_{similarity} = \frac{(-0.35 + 0.35)^2}{\big(0.35 \times (1 - 0.35)\big) + \big(0.65 \times (1 - 0.65)\big) + 0}$$

$$Left_{similarity} = \frac{0^2}{0.2275 + 0.2275}$$

$$Left_{similarity} = 0$$

$$Right_{similarity} = \frac{(0.35 + 0.35 - 0.35)^2}{\Big(\big(0.65 \times (1 - 0.65)\big) \times 2\Big) + \big(0.35 \times (1 - 0.35)\big) + 0}$$

$$Right_{similarity} = \frac{0.35^2}{0.455 + 0.2275} = \frac{0.1225}{0.6825}$$

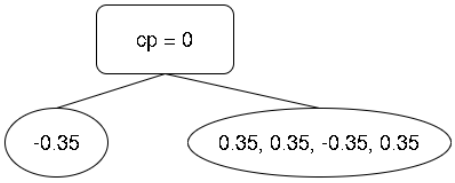
$$Right_{similarity} = 0.1794$$

$$gain = Left_{similarity} + Right_{similarity} - Root_{similarity}$$

$$gain = 0 + 0.1794 - 0.1076$$

$$gain = 0.0718$$

3. Cp=0




**Gambar 3.24** *tree kedua untuk node cp=0*

$$Left_{similarity} = \frac{(-0.35)^2}{\big(0.35 \times (1 - 0.35)\big) + 0} = \frac{0.1225}{0.2275}$$

$$Left_{similarity} = 0.5384$$

$$Right_{similarity} = \frac{(0.35 + 0.35 - 0.35 + 0.35)^2}{\Big(\big(0.65 \times (1 - 0.65)\big) \times 3\Big) + \big(0.35 \times (1 - 0.35)\big) + 0}$$

$$Right_{similarity} = \frac{(0.35 + 0.35)^2}{0.6825 + 0.2275} = \frac{0.7^2}{0.91} = \frac{0.49}{0.91}$$

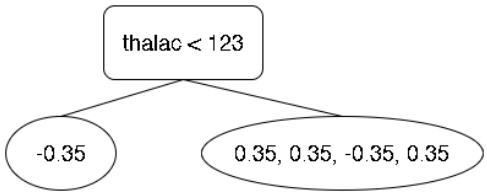
$$Right_{similarity} = 0.5384$$

$$gain = Left_{similarity} + Right_{similarity} - Root_{similarity}$$

$$gain = 0.5384 + 0.5384 - 0.1076$$

$$gain = 0.9692$$

4. Thalac<123




**Gambar 3.25** *tree kedua untuk node thalac<123*

$$Left_{similarity} = \frac{(-0.35)^2}{\big(0.35 \times (1 - 0.35)\big) + 0} = \frac{0.1225}{0.2275}$$

$$Left_{similarity} = 0.5384$$

$$Right_{similarity} = \frac{(0.35 + 0.35 - 0.35 + 0.35)^2}{\Big(\big((0.65 \times (1 - 0.65)\big) \times 3\Big) + \big(0.35 \times (1 - 0.35)\big) + 0}$$

$$Right_{similarity} = \frac{(0.35 + 0.35)^2}{0.6825 + 0.2275} = \frac{0.7^2}{0.91} = \frac{0.49}{0.91}$$

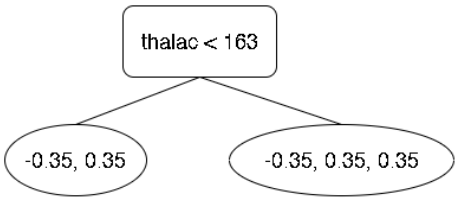
$$Right_{similarity} = 0.5384$$

$$gain = Left_{similarity} + Right_{similarity} - Root_{similarity}$$

$$gain = 0.5384 + 0.5384 - 0.1076$$

$$gain = 0.9692$$

5. Thalac<163




**Gambar 3.26** *tree kedua untuk node thalac<163*

$$Left_{similarity} = \frac{(-0.35 + 0.35)^2}{\big(0.35 \times (1 - 0.35)\big) + \big(0.65 \times (1 - 0.65)\big) + 0}$$

$$Left_{similarity} = 0$$

$$Right_{similarity} = \frac{(-0.35 + 0.35 + 0.35)^2}{\big(0.35 \times (1 - 0.35)\big) + \Big(\big(0.65 \times (1 - 0.65)\big) \times 2\Big) + 0}$$

$$Right_{similarity} = \frac{0.35^2}{0.2275 + 0.455} = \frac{0.1225}{0.6825}$$

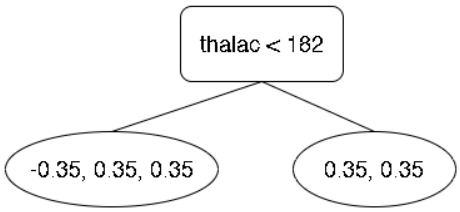
$$Right_{similarity} = 0.1794$$

$$gain = Left_{similarity} + Right_{similarity} - Root_{similarity}$$

$$gain = 0 + 0.1794 - 0.1076$$

$$gain = 0.0718$$

6. Thalac<182




**Gambar 3.27** *tree kedua untuk node thalac<182*

$$Left_{similarity} = \frac{(-0.35 + 0.35 + 0.35)^2}{\big(0.35 \times (1 - 0.35)\big) + \Big(\big(0.65 \times (1 - 0.65)\big) \times 2\Big) + 0}$$

$$Left_{similarity} = \frac{0.35^2}{0.2275 + 0.455} = \frac{0.1225}{0.6825}$$

$$Left_{similarity} = 0.1794$$

$$Right_{similarity} = \frac{(0.35 + 0.35)^2}{\Big(\big(0.65 \times (1 - 0.65)\big) \times 2\Big) + 0}$$

$$Right_{similarity} = \frac{0.7^2}{0.455} = \frac{0.49}{0.455}$$

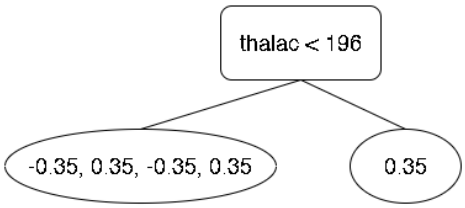
$$Right_{similarity} = 1.0769$$

$$gain = Left_{similarity} + Right_{similarity} - Root_{similarity}$$

$$gain = 0.1794 + 1.0769 - 0.1076$$

$$gain = 1.1487$$

7. Thalac<196




**Gambar 3.28** *tree kedua untuk node thalac<196*

$$Left_{similarity}$$

$$= \frac{(-0.35 - 0.35 + 0.35 + 0.35)^2}{\Big(\big(0.35 \times (1 - 0.35)\big) \times 2\Big) + \Big(\big(0.65 \times (1 - 0.65)\big) \times 2\Big) + 0}$$

$$= 0$$

$$Right_{similarity} = \frac{(0.35)^2}{\big(0.65 \times (1 - 0.65)\big) + 0}$$

$$Right_{similarity} = \frac{0.1225}{0.2275}$$

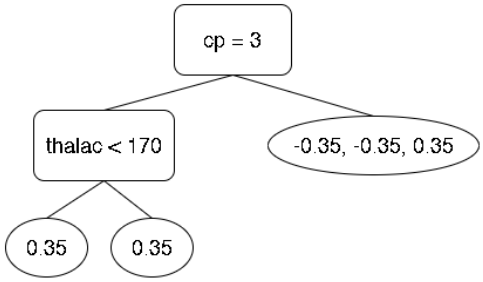
$$Right_{similarity} = 0.5384$$

$$gain = Left_{similarity} + Right_{similarity} - Root_{similarity}$$

$$gain = 0 + 0.5384 - 0.1076$$

$$gain = 0.4308$$

Setelah semua kemungkinan *node* yang ada kita hitung nilai gainnya. *Node* yang memiliki nilai *gain* paling tinggi akan dijadikan sebagai *root node*. Sehingga, sesuai dengan perhitungan nilai *gain* diatas, yang akan dijadikan *root node* adaalah node cp=3. Sehingga *tree* kedua sementara yang kita dapatkan adalah sebagai berikut.




**Gambar 3.29** *hasil tree kedua sementara yang terbentuk*

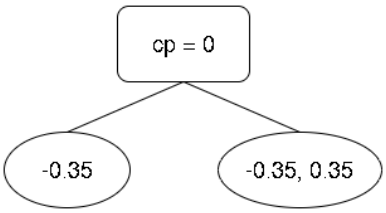
Kemudian, setelah kita menentukan *root node* seperti yang kita lakukan di atas. Selanjutnya kita melakukan hal yang sama untuk menentukan *node* pada kedalaman selanjutnya. Yaitu melakukan perhitungan nilai *gain* untuk kemungkinan *node* yang ada. Perhitungan dan kemungkinan *node* yang ada adalah sebagai berikut.

$$Root_{similarity} = \frac{(-0.35 - 0.35 + 0.35)^2}{\Big(\big(0.35 \times (1 - 0.35)\big) \times 2\Big) + \big(0.65 \times (1 - 0.65)\big) + 0}$$

$$Root_{similarity} = \frac{-0.35^2}{0.455 + 0.2275} = \frac{0.1225}{0.6825}$$

$$Root_{similarity} = 0.1794$$

1. Cp=0




**Gambar 3.30** *tree kedua untuk node cp=0*

$$Left_{similarity} = \frac{-0.35^2}{\big(0.35 \times (1 - 0.35)\big) + 0} = \frac{0.1225}{0.2275}$$

$$Left_{similarity} = 0.5384$$

$$Right_{similarity} = \frac{(-0.35 + 0.35)^2}{\big(0.35 \times (1 - 0.35)\big) + \big(0.65 \times (1 - 0.65)\big) + 0}$$

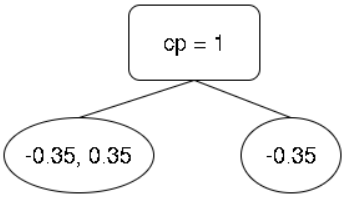
$$Right_{similarity} = 0$$

$$gain = Left_{similarity} + Right_{similarity} - Root_{similarity}$$

$$gain = 0.5384 + 0 - 0.1794$$

$$gain = 0.359$$

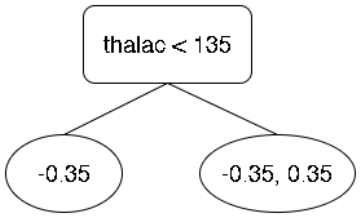
2. Cp=1




**Gambar 3.31** *tree kedua untuk node cp=1*

$$Left_{similarity} = \frac{(-0.35 + 0.35)^2}{\big(0.35 \times (1 - 0.35)\big) + \big(0.65 \times (1 - 0.65)\big) + 0}$$
$$Left_{similarity} = 0$$
$$Right_{similarity} = \frac{-0.35^2}{\big(0.35 \times (1 - 0.35)\big) + 0} = \frac{0.1225}{0.2275}$$
$$Right_{similarity} = 0.5384$$
$$gain = Left_{similarity} + Right_{similarity} - Root_{similarity}$$
$$gain = 0 + 0.5384 - 0.1794$$
$$gain = 0.359$$

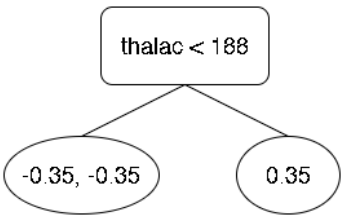
3. Thalac<135




**Gambar3.32** *tree kedua untuk node thalac<135*

$$Left_{similarity} = \frac{-0.35^2}{\big(0.35 \times (1 - 0.35)\big) + 0} = \frac{0.1225}{0.2275}$$
$$Left_{similarity} = 0.5384$$
$$Right_{similarity} = \frac{(-0.35 + 0.35)^2}{\big(0.35 \times (1 - 0.35)\big) + \big(0.65 \times (1 - 0.65)\big) + 0}$$
$$Right_{similarity} = 0$$
$$gain = Left_{similarity} + Right_{similarity} - Root_{similarity}$$
$$gain = 0.5384 + 0 - 0.1794$$
$$gain = 0.359$$

4. Thalac<188




**Gambar 3.33** *tree kedua untuk node thalac<188*

$$Left_{similarity} = \frac{(-0.35 - 0.35)^2}{\big((0.35 \times (1 - 0.35)) \times 2\big) + 0} = \frac{0.49}{0.455}$$

$$Left_{similarity} = 1.0769$$

$$Right_{similarity} = \frac{0.35^2}{\big(0.65 \times (1 - 0.65)\big) + 0} = \frac{0.1225}{0.2275}$$

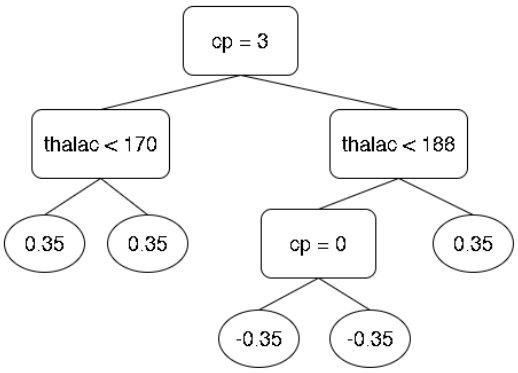
$$Right_{similarity} = 0.5384$$

$$gain = left_{similarity} + Right_{similarity} - Root_{similarity}$$

$$gain = 1.0769 + 0.5384 - 0.1794$$

$$gain = 1.4359$$

Setelah menentukan *node* dengan cara menghitung nilai *gain* masing-masing kemungkinan *node*. Maka *node* yang memiliki *gain* tertinggi adalah thalac<188. Sehingga hasil *tree* kedua yang dibuat adalah sebagai berikut.

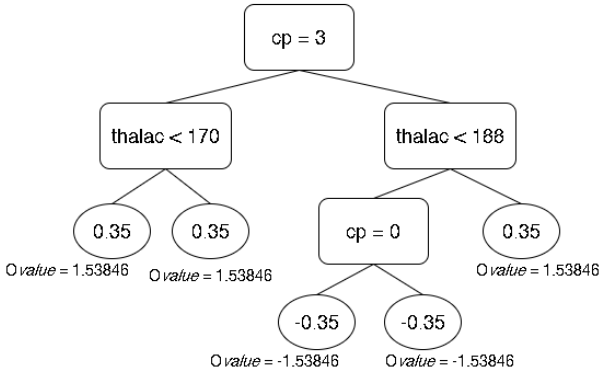



**Gambar 3.34** *hasil tree kedua*

Setelah *tree* kedua terbentuk, kita menentukan $O_{value}$ untuk masing-masing *leaf* yang ada. Rumus untuk menghitung $O_{value}$ dan perhitungan $O_{value}$ untuk masing-masing *leaf* adalah sebagai berikut.

$$O_{value} = \frac{\sum(residual)_i}{\sum[prev\ probability_i \times (1 - prev\ probability_i)] + \lambda}$$

1. $O_{value} = \frac{0.35}{(0.65 \times (1 - 0.65)) + 0} = 1.53846$
2. $O_{value} = \frac{0.35}{(0.65 \times (1 - 0.65)) + 0} = 1.53846$
3. $O_{value} = \frac{-0.35}{(0.35 \times (1 - 0.35)) + 0} = -1.53846$
4. $O_{value} = \frac{-0.35}{(0.35 \times (1 - 0.35)) + 0} = -1.53846$
5. $O_{value} = \frac{0.35}{(0.65 \times (1 - 0.65)) + 0} = 1.53846$




**Gambar 3.35** *tree kedua dan* $O_{value}$

Setelah mendapatkan nilai $O_{value}$ dari masing-masing *leaf* yang ada. Maka kita dapat menghitung *probability* untuk masing-masing data. Dari hasil *probability* tersebut nantinya akan mendapatkan nilai *residual* yang baru yang nantinya akan digunakan untuk membuat *tree* pada iterasi berikutnya. Perhitungan probability adalah sebagai berikut.

1. $\log(odds) = 0 + (0.3 \times 2) + (0.3 \times 1.53846) = 1.0615$
   $probability = \frac{e^{1.0615}}{1 + e^{1.0615}} = \frac{2.891}{3.891}$
   $probability = 0.74299$
2. $\log(odds) = 0 + (0.3 \times 2) + (0.3 \times 1.53846) = 1.0615$
   $probability = \frac{e^{1.0615}}{1 + e^{1.0615}} = \frac{2.891}{3.891}$
   $probability = 0.74299$
3. $\log(odds) = 0 + (0.3 \times -2) + (0.3 \times -1.53846) = -1.0615$
   $probability = \frac{e^{-1.0615}}{1 + e^{-1.0615}} = \frac{0.346}{1.346}$
   $probability = 0.25705$
4. $\log(odds) = 0 + (0.3 \times -2) + (0.3 \times -1.53846) = -1.0615$
   $probability = \frac{e^{-1.0615}}{1 + e^{-1.0615}} = \frac{0.346}{1.346}$
   $probability = 0.25705$
5. $\log(odds) = 0 + (0.3 \times 2) + (0.3 \times 1.53846) = 1.0615$
   $probability = \frac{e^{1.0615}}{1 + e^{1.0615}} = \frac{2.891}{3.891}$
   $probability = 0.74299$

Setelah mengetahui nilai *probability* maka dapat dihitung nilai *residual* yang akan digunakan untuk membuat *tree* pada iterasi berikutnya. Dalam algoritma xgboost, biasanya *tree* yang dibuat 100 *tree* atau lebih. Hasil *probability* baru dan nilai *residual* baru terdapat pada tabel di bawah ini.

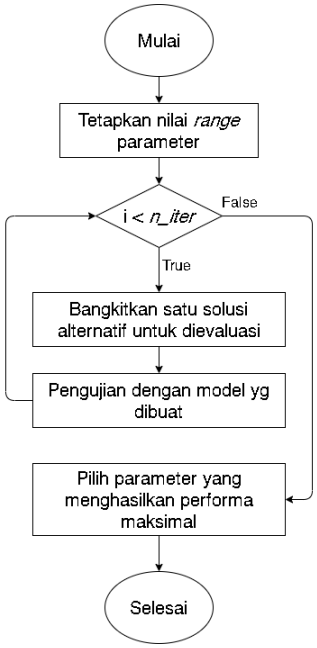
**Tabel 3.10** *tabel data dan residual*

| Cp | Thalac | Target | Probability | Residual |
|---|---|---|---|---|

| | | | | |
|---|---|---|---|---|
| 3 | 150 | 1 | 0.74299 | 0.25701 |
| 3 | 190 | 1 | 0.74299 | 0.25701 |
| 0 | 96 | 0 | 0.25705 | -0.25705 |
| 1 | 174 | 0 | 0.25705 | -0.25705 |
| 1 | 202 | 1 | 0.74299 | 0.25701 |

Sesuai dengan tabel di atas, kemudian akan dilakukan pembuatan *tree* yang ketiga. Dan akan menghasilkan nilai *residual* yang semakin kecil hingga mendekati 0 pada label target 0, dan mendekati 1 untuk label target 1. Banyaknya *tree* yang akan dibuat biasanya kita yang menentukan pada *hyper parameter*. Pada *hyper parameter*, terdapat beberapa parameter yang dapat kita tentukan untuk mendapatkan nilai parameter paling optimal, sehingga model yang dibuat pun memiliki performa yang maksimal dengan akurasi yang tinggi. Untuk menentukan *hyper parameter* yang menghasilkan model yang memiliki akurasi maksimal, maka menggunakan *randomized search optimizer*.

Di dalam *randomized search*, teknik tersebut bekerja dengan cara membangkitkan nilai *random* dari *range* parameter yang sudah kita tentukan dan mencoba nilai parameter tersebut. Teknik *randomized search* akan melakukan pembangkitan nilai *random* dan evaluasi untuk mendapatkan performa dari parameter tersebut secara iteratif sebanyak iterasi yang kita tentukan. Di dalam *randomized search* juga terdapat fitur *cross validation* untuk melakukan validasi terhadap model yang sedang dibuat. Kita dapat menentukan nilai cv, biasanya diatur dengan nilai 3 sampai 5 yang berarti *data training* akan dibagi menjadi sebanyak nilai itu, dan masing-masing bagian akan menjadi *data testing* untuk pengujian model. Apabila iterasi pada *randomized search* berjumlah 100 dan kita mengatur *cross validation* dengan nilai 3, maka jumlah iterasi total adalah 300 iterasi. Flowchart randomized search adalah sebagai berikut.




**Gambar 3.36** *flowchart randomized search*

### 3.6.3. Evaluasi

Untuk melakukan evaluasi dari model klasifikasi xgboost yang dibuat, pada penelitian ini dilakukan dengan menggunakan *confusion matrix*. Dengan menggunakan confusion matrix maka dapat diketahui akurasi, presisi, dan recall dari model klasifikasi yang dibuat. Contoh evaluasi yang dilakukan menggunakan data yang sudah ditraining pada proses diatas adalah sebagai berikut. model yang dibuat dengan menggunakan dua *tree* dan menggunakan learning rate($\varepsilon$) adalah 0,3.

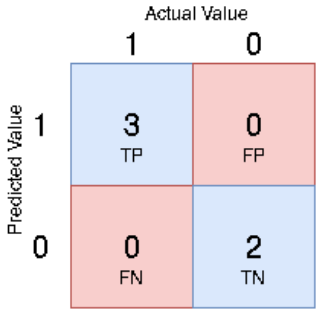
$$\log(odds) = 0 + \sum [0.3 \times O_{value}]_i$$

$$probability = \frac{e^{\log(odds)}}{1 + e^{\log(odds)}}$$

**Tabel 3.11** *hasil prediksi*

| Cp | Thalac | Target | Prediction |
|---|---|---|---|
| 3 | 150 | 1 | 0.74299 (1) |
| 3 | 190 | 1 | 0.74299 (1) |
| 0 | 96 | 0 | 0.25705 (0) |
| 1 | 174 | 0 | 0.25705 (0) |
| 1 | 202 | 1 | 0.74299 (1) |

Karena kita menentukan di awal dengan standar 0.5 yang berarti yang memiliki $probabilti \geq 0.5$ adalah label 1, dan $probability < 0.5$ adalah dengan label 0. Sehingga bentuk *confusion matrix* adalah sebagai berikut.




**Gambar 3.37** *evaluasi dengan confusion matrix*

Sesuai dengan confusion matrix di atas, maka dapat dilakukan perhitungan mengenai akurasi, recall, dan presisi sebagai berikut.

1. $akurasi = \frac{TP+TN}{TP+FP+FN+TN} = \frac{5}{5} = 1.0$
2. $recall = \frac{TP}{TP+FN} = \frac{3}{3} = 1.0$
3. $presisi = \frac{TP}{TP+FP} = \frac{3}{3} = 1.0$

Hasil tersebut hanya berdasarkan sebagian datasets yang digunakan dalam contoh proses training dan menggunakan nilai *lambda*($\lambda$), *gamma*($\gamma$) dan *cover* adalah 0. Apabila menggunakan seluruh *datasets* maka hasil dapat berbeda. Menggunakan

*randomized search* juga akan membantu menemukan parameter terbaik untuk mendapatkan model dengan performa yang baik dan tidak *overfit*.

**3.7.as**

# DAFTAR PUSTAKA


Aini, S. H. A., Sari, Y. A., & Arwan, A. (2018). Seleksi Fitur Information Gain untuk Klasifikasi Penyakit Jantung Menggunakan Kombinasi Metode K-Nearest Neighbor dan Naïve Bayes. *Jurnal Pengembangan Teknologi Informasi Dan Ilmu Komputer*, *2*(9), 2546–2554.

Alpaydin, E. (2010). *Introduction to Machine Learning*. https://books.google.co.id/books?id=tZnSDwAAQBAJ&sitesec=buy&hl=id&source=gbs_vpt_read

Andreanus, J., & Kurniawan, A. (2018). Sejarah , Teori Dasar dan Penerapan Reinforcement Learning : Sebuah Tinjauan Pustaka. *Jurnal Telematika*, *12*(2), 113–118.

Anies. (2015). *Kolesterol dan Penyakit Jantung Koroner*. Ar-Ruzz Media.

Annisa, R. (2019). Analisis Komparasi Algoritma Klasifikasi Data Mining Untuk Prediksi Penderita Penyakit Jantung. *Jurnal Teknik Informatika Kaputama (JTIK)*, *3*(1), 22–28. https://jurnal.kaputama.ac.id/index.php/JTIK/article/view/141/156

Anwar, T. B. (2004). Faktor Risiko Penyakit Jantung Koroner. *E-USU Repository*, *01*(Medan), 1–15. http://repository.usu.ac.id/bitstream/handle/123456789/3472/gizi-bahri4.pdf?sequence=1

Arulkumaran, K., Deisenroth, M., Brundage, M., & Bhatarath, A. (2017). Deep reinforcement learning: A brief survey. *IEEE Signal Process Mag*, *34*, *n*.

Briggs, L. J. (1991). *Instrukticonal Design: Principles and Aplications*. Educational Technology.

Dewi, S. (2016). Komparasi 5 Metode Algoritma Klasifikasi Data Mining Pada Prediksi Keberhasilan Pemasaran Produk Layanan Perbankan. *Techno Nusa Mandiri*, *XIII*(1), 60–66.

Ghani, M. A., & Subekti, A. (2018). Email Spam Filtering Dengan Algoritma Random Forest. *IJCIT (Indonesian Journal on Computer and Information Technology*, *Vol.3*, *No.*(2), 216~221.

Ginting, S. L., Zarman, W., & Hamidah, I. (2014). Analisis dan Penerapan Algoritma C4.5 Dalam Data Mining Untuk Memprediksi Masa Studi Mahasiswa Berdasarkan Data Nilai Akademik. *Snast*, *November*, 159.

Gunawan, V. A., Fitriani, I. I., & Putra, L. S. A. (2020). Sistem Diagnosis Otomatis Identifikasi Penyakit Jantung Coroner Menggunakan Ektraksi Ciri GLCM dan Klasifikasi SVM. *Informatika Mulawarman : Jurnal Ilmiah Ilmu Komputer*, *15*(1), 13. https://doi.org/10.30872/jim.v15i1.2495

Hanifa, T. T., Al-faraby, S., & Adiwijaya. (2017). Analisis Churn Prediction pada Data Pelanggan PT . Telekomunikasi dengan Logistic Regression dan Underbagging. *Universitas Telkom*, *4*(2), 78.

Haqie, Z. A., Nadiah, R. E., & Ariyani, O. P. (2020). Inovasi Pelayanan Publik Suroboyo Bis Di Kota Surabaya. *JPSI (Journal of Public Sector Innovations)*, *5*(1), 23. https://doi.org/10.26740/jpsi.v5n1.p23-30

Ibrahem Ahmed Osman, A., Najah Ahmed, A., Chow, M. F., Feng Huang, Y., & El-Shafie, A. (2020). Extreme gradient boosting (Xgboost) model to predict the groundwater levels in Selangor Malaysia. *Ain Shams Engineering Journal*, *xxxx*. https://doi.org/10.1016/j.asej.2020.11.011

Indrawati, L. (2014). Hubungan Antara Pengetahuan, Sikap, Persepsi, Motivasi, Dukungan Keluarga dan Sumber Informasi Pasien Penyakit Jantung Koroner dengan Tindakan Pencegahan Sekunder Faktor Risiko (Studi Kasus di RSPAD Gatot Soebroto Jakarta). *Jurnal Ilmiah Widya*, *2*(3), 30–36.

Indriyono, B. V., Utami, E., & Sunyoto, A. (2015). *Pemanfaatan Algoritma Porter Stemmer Untuk Bahasa Indonesia Dalam Proses Klasifikasi Jenis Buku*. 301–310.

Jothikumar, R., & Siva Balan, R. (2016). C4.5 classification algorithm with back-track pruning for accurate prediction of heart disease. *ISSN: 0970-938X (Print)*. https://www.biomedres.info/biomedical-research/c45-classification-algorithm-with-backtrack-pruning-for-accurate-prediction-of-heart-disease.html

Karo, I. M. K. (2020). *Implementasi Metode XGBoost dan Feature Importance untuk Klasifikasi pada Kebakaran Hutan dan Lahan*. *1*(1), 10–16.

Lestari, M. (2014). Penerapan Algoritma Klasifikasi Nearest Neighbor (K-NN) untuk Mendeteksi Penyakit Jantung. *Faktor Exacta*, *7*(September 2010), 366–371.

Marleni, L., & Alhabib, A. (2017). Faktor Risiko Penyakit Jantung Koroner di RSI SITI Khadijah Palembang. *Jurnal Kesehatan*, *8*(3), 478. https://doi.org/10.26630/jk.v8i3.663

Muslim, F. (2019). *Penerapan Brute Force dan Decrease and Conquer pada Parameter Tuning XGBoostClassifier*.

Normawati, D., & Winarti, S. (2017). Seleksi Fitur Menggunakan Penambangan Data Berbasis Variable Precision Rough Set (VPRS) untuk Diagnosis Penyakit Jantung Koroner. *Jurnal Ilmiah Teknik Elektro Komputer Dan Informatika*, *3*(2), 100. https://doi.org/10.26555/jiteki.v3i2.8072

Novandya, A. (2017). Penerapan Algoritma Klasifikasi Data Mining C4.5 Pada Dataset Cuaca Wilayah Bekasi. *KNiST*, *XIV*(2), 368–372.

Nuraeni, A. (2016). Faktor yang Memengaruhi Kualitas Hidup Pasien dengan Penyakit Jantung Koroner. *Jurnal Keperawatan Padjadjaran*, *v4*(n2), 107–116. https://doi.org/10.24198/jkp.v4n2.1

Nurhayati, Busman, & Iswara, R. P. (2019). Pengembangan Algoritma Unsupervised Learning Technique Pada Big Data Analysis di Media Sosial sebagai media promosi Online Bagi Masyarakat. *Jurnal Teknik Informatika*, *12*(1), 79–96. https://doi.org/10.15408/jti.v12i1.11342

Omer, M. K., Sheta, O. E., Adrees, M. S., Stiawan, D., Riyadi, M. A., & Budiarto, R. (2018). Deep neural network for heart disease medical prescription expert system. *Indonesian Journal of Electrical Engineering and Informatics*, *6*(2), 217–224. https://doi.org/10.11591/ijeei.v6i2.456

Pareza Alam Jusia. (2018). Analisis komparasi pemodelan algoritma decision tree menggunakan metode particle swarm optimization dan metode adaboost untuk prediksi awal penyakit jantung. *Seminar Nasional Sistem Informasi 2018*, 1048–1056.

Peryanto, A., Yudhana, A., & Umar, R. (2020). Klasifikasi Citra Menggunakan Convolutional Neural Network dan K Fold Cross Validation. *Journal of Applied Informatics and Computing*, *4*(1), 45–51. https://doi.org/10.30871/jaic.v4i1.2017

Pinata, N. N. P., Sukarsa, I. M., & Rusjayanthi, N. K. D. (2020). *Prediksi Kecelakaan Lalu Lintas di Bali dengan XGBoost pada Python*. *8*(3), 188–196.

Prasetyo, E., & Prasetiyo, B. (2020). *PENINGKATAN AKURASI KLASIFIKASI ALGORITMA C4.5 MENGGUNAKAN TEKNIK BAGGING PADA DIAGNOSIS PENYAKIT JANTUNG*. *7*(5), 1035–1040. https://doi.org/10.25126/jtiik.202072379

Purnamasari, D., Henrata, J., Sasmita, Y. P., Ihsani, F., & Wicaksana, I. W. S. (2013). *Get Easy Using WEKA*.

Puspitasari, A. M., Ratnawati, D. E., & Widodo, A. W. (2018). Klasifikasi Penyakit Gigi Dan Mulut Menggunakan Metode Support Vector Machine. *Jurnal Pengembangan Teknologi Informasi Dan Ilmu Komputer*, *2*(2), 802–810.

Putra, P. D., & Rini, D. P. (2019). Prediksi Penyakit Jantung dengan Algoritma Klasifikasi.

*Prosiding Annual Research Seminar 2019*, *5*(1), 978–979.
Retnasari, T., & Rahmawati, E. (2017). Diagnosa Prediksi Penyakit Jantung Dengan Model Algoritma Naïve Bayes Dan Algoritma C4.5. *Konferensi Nasional Ilmu Sosial & Teknologi (KNiST)*, 7–12.
Rohman, A., Suhartono, V., & Supriyanto, C. (2017). Penerapan Agoritma C4.5 Berbasis Adaboost Untuk Prediksi Penyakit Jantung. *Jurnal Teknologi Informasi*, *13*, 13–19.
Saifullah, Zarlis, M., Zakaria, & Sembiring, R. W. (2017). Analisa Terhadap Perbandingan Algoritma Decision Tree Dengan Algoritma Random Tree Untuk Pre-Processing Data. *J-SAKTI (Jurnal Sains Komputer Dan Informatika)*, *1*(2), 180. https://doi.org/10.30645/j-sakti.v1i2.41
Sanjaya, W. (2008). *Perencanaan dan Desain Sistem Pembelajaran*. Kencana Group.
Setiawati, D., Taufik, I., Jumadi, J., & Zulfikar, W. B. (2016). Klasifikasi Terjemahan Ayat Al-Quran Tentang Ilmu Sains Menggunakan Algoritma Decision Tree Berbasis Mobile. *Jurnal Online Informatika*, *1*(1), 24. https://doi.org/10.15575/join.v1i1.7
Syukron, M., Santoso, R., & Widiharih, T. (2020). *Perbandingan Metode Smote Random Forest dan Smote XgBoost Untuk Klasifikasi Tingkat Penyakit Hepatitis C Pada Imbalance Class Data*. *9*, 227–236.
Utomo, D. P., & Mesran, M. (2020). Analisis Komparasi Metode Klasifikasi Data Mining dan Reduksi Atribut Pada Data Set Penyakit Jantung. *Jurnal Media Informatika Budidarma*, *4*(2), 437. https://doi.org/10.30865/mib.v4i2.2080
Wibisono, A. B., & Fahrurozi, A. (2019). Perbandingan Algoritma Klasifikasi Dalam Pengklasifikasian Data Penyakit Jantung Koroner. *Jurnal Ilmiah Teknologi Dan Rekayasa*, *24*(3), 161–170. https://doi.org/10.35760/tr.2019.v24i3.2393
Widiastuti, N. A., Santosa, S., & Supriyanto, C. (2014). Algoritma Klasifikasi Data Mining Naïve Bayes Berbasis Particle Swarm Optimization Untuk Deteksi Penyakit Jantung. *Pseudocode*, *1*, 11–14.
Wiharto, W., Suryani, E., & Cahyawati, V. (2019). The methods of duo output neural network ensemble for prediction of coronary heart disease. *Indonesian Journal of Electrical Engineering and Informatics*, *7*(1), 50–57. https://doi.org/10.11591/ijeei.v7i1.458
Yualinda, S., Wijaya, D. R., & Hernawati, E. (2020). Aplikasi Berbasis Dataset E-Commerce Untuk Prediksi Kemiskinan Menggunakan Algoritma Naive Bayes, XgBoost dan Similarity Based Feature Selection. *Jurnal Borneo Cendekia*, *3*(2), 40–46.
Zahrawardani, D., Herlambang, K. S., & Anggraheny, H. D. (2013). Analisis Faktor Risiko Kejadian Penyakit Jantung Koroner di RSUP Dr Kariadi Semarang. *Jurnal Kedokteran Muhammadiyah*, *1*(3), 13. http://jurnal.unimus.ac.id/index.php/kedokteran/article/view/1341
Zulaekah, S., Rahmawati, A. C., & Rahmawaty, S. (2009). Aktivitas Fisik dn Rasio Kolesterol (HDL) pada Penderita Penyakit Jantung Koroner di Poliklinik Jantung RSUD Dr Moewardi Surakarta. *Jurnal Kesehatan*, *2*(1), 11–18.