Prediksi Penyakit Jantung dengan Menggunakan Algoritma XgBoost dan Randomized Search Optimizer

Proposal

Disusun oleh:
Reo Sahobby
123170067

PROGRAM STUDI INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNIK INDUSTRI
UNIVERSITAS PEMBANGUNAN NASIONAL "VETERAN"
YOGYAKARTA

2021

Daftar Isi

Daftar Gambar

Daftar Tabel

Daftar Persamaan

BAB I
PENDAHULUAN

1.1.Latar Belakang

Jantung merupakan organ dalam manusia yang fungsinya sangatlah penting yaitu untuk mengedarkan darah yang berisi oksigen dan nutrisi ke seluruh tubuh 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.

Proses pendeteksian apakah seseorang tersebut terkena penyakit jantung dapat dilakukan dengan 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 efisien, 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. Penelitian yang sudah dilakukan untuk menciptakan pendeteksian penyakit jantung secara digital seperti penelitian yang dilakukan oleh Retnasari (Retnasari & Rahmawati, 2017), penelitian yang dilakukan oleh Wibisono (Wibisono & Fahrurozi, 2019), dan penelitian yang dilakukan oleh Prasetyo (Prasetyo & Prasetiyo, 2020). Penelitian tersebut dilakukan 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. Salah satu teknik identifikasi penyakit jantung adalah menggunakan metode klasifikasi. Klasifikasi adalah jenis analisis data yang digunakan untuk memprediksi label kelas dari data tersebut (Annisa, 2019).

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 Wibisono dan Achmad Fahrurozi, 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 sebelumnya, tentunya masing-masing algoritma memiliki kelebihan dan kelemahan. Contohnya adalah terjadinya *overfitting*, ciri *overfitting* adalah memiliki hasil *training* yang sangat bagus, namun pada saat dilakukan pengujian terhadap *data testing* diperoleh performa yang buruk (Septadaya et al., 2019). Metode yang memiliki overfitting contohnya adalah C4,5 (Rahayu et al., 2015), yang dalam penelitian tersebut dapat diselesaikan dengan menggunakan *threshold pruning*. Kemudian penelitian dengan algoritma serupa (Afianto et al., 2017), dapat mengatasi *overfitting* menggunakan algoritma Random Forest. Penelitian yang dilakukan untuk memprediksi ketahanan hidup pasien jantung koroner (Kusuma & Srinandi, 2013), menggunakan metode *Partial Least Square* (PLS) untuk mengatasi *overfitting*. Sedangkan penelitian yang dilakukan untuk mendiagnosis penyakit jantung berdasarkan suara (Lubis & Gondawijaya, 2019), menerapkan *cross validation* dalam pembuatan modelnya untuk mengatasi *overfitting*.

Pada penelitian ini algoritma yang dipilih untuk mengatasi permasalahan *overfitting* adalah menggunakan algoritma XgBoost. Algoritma XgBoost adalah algoritma *gradient boosting* yang dibuat dengan *tree-based* yang dapat membuat *boosted tree* secara efisien dan dapat dikerjakan secara paralel (Karo, 2020). Algoritma XgBoost juga memiliki regularization yang dapat berfungsi untuk menghindari overfitting yang terjadi (Zhang et al., 2018).

Berdasarkan latar belakang dan analisis permasalahan yang telah dilakukan. Tujuan dari penelitian ini adalah untuk mengetahui hasil identifikasi penyakit jantung dengan

menggunakan algoritma XgBoost. Selanjutnya, akan dilakukan analisis performa dari model yang dibuat, seperti hasil akurasi. Penelitian ini akan dilakukan dengan menggunakan data rekam jantung berupa data *tabular* yang nantinya berdasarkan data tersebut akan diklasifikasikan ke dalam terkena penyakit jantung, atau tidak terkena penyakit jantung.

1.2. Rumusan Masalah

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

a. Proses identifikasi penyakit jantung yang tidak efisien jika dilakukan dengan cara konvensional.
b. *Overfitting* yang masih terjadi pada algoritma lain dalam mengidentifikasi 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. Algoritma klasifikasi yang digunakan adalah algoritma XgBoost.
c. Hasil identifikasi penyakit jantung adalah terkena penyakit jantung dengan label 1, atau tidak terkena penyakit jantung dengan label 0.

1.4. Tujuan Penelitian

Tujuan yang ingin dicapai dari penelitian ini adalah sebagai berikut:

a. Mengidentifikasi penyakit jantung berdasarkan data *tabular* data rekam jantung.
b. Menerapkan algoritma klasifikasi XgBoost untuk mengidentifikasi penyakit jantung dan mengatasi *overfitting*.

1.5. Manfaat Penelitian

a. Membantu dalam proses deteksi dini penyakit jantung. Sehingga membuat kita semakin sadar akan kesehatan jantung kita.
b. Mengetahui performa model yang dibuat menggunakan algoritma XgBoost dalam menyelesaikan permasalahan identifikasi penyakit jantung

1.6. Tahapan Penelitian

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

a. Studi Literatur

   Tahap pertama yang dilakukan dalam penelitian ini adalah melakukan studi literatur 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. Pembuatan Model *Machine Learning*

   Tahap selanjutnya adalah pembuatan model *machine learning*, yaitu pada tahap ini dilakukan pembuatan model menggunakan algoritma dan teknik yang sudah dipilih.

d. Pengujian dan Evaluasi Model

   Setelah model *machine learning* 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.

e. Implementasi Perangkat Lunak dan Pengujian

   Selanjutnya, setelah model yang dibuat memiliki performa yang bagus, model tersebut diimplementasikan dalam bentuk perangkat lunak yang bisa digunakan oleh pengguna. Dalam pembuatan perangkat lunak ini, menggunakan metodologi *waterfall*. Setelah perangkat lunak selesai dibuat, dilakukan pengujian perangkat lunak untuk memastikan perangkat lunak yang dibuat berjalan normal tanpa ada kendala.

f. 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 LITERATUR

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 identifikasi 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
TINJAUAN LITERATUR

2.1.Tinjauan Studi

Saat ini, penyakit jantung menjadi tantangan tersendiri di indsutri pelayanan kesehatan terutama dalam satu dekade terakhir (Prasetyo & Prasetiyo, 2020). 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 (Prasetyo & Prasetiyo, 2020) dengan judul **Peningkatan Akurasi Klasifikasi Algoritma C4,5 Menggunakan Teknik Bagging Pada Diagnosis Penyakit Jantung**, penelitian tersebut dilakukan menggunakan *dataset heart disease* dengan data sebanyak 303, dengan jumlah kolom yang digunakan 13 kolom seperti *age*, *sex*, *cp*, *chol*, dan lain-lain. 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 *library* 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 (Putra & Rini, 2019) dengan judul **Prediksi Penyakit Jantung Dengan Algoritma Klasifikasi**. 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. *Dataset* yang digunakan dalam penelitian tersebut adalah *Statelog Heart Disease Dataset* yang berasal dari UCI *machine learning* yang dapat diunduh secara online. *Dataset* 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 *dataset*, 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.

Penelitian yang dilakukan oleh (Wibisono & Fahrurozi, 2019) dengan judul **Perbandingan Algoritma Klasifikasi Dalam Pengklasifikasian Data Penyakit Jantung Koroner**. Penelitian tersebut dilakukan untuk mengimplemantasikan beberapa metode klasifikasi seperti *Naive Bayes*, *K-Nearest Neighbor*, *Decision Tree*, *Random Forest*, dan *Support Vector Machine* pada kasus pengenalan penyakit jantung koroner. Hasil dari pengukuran yang didapat adalah akurasi, *recall*, dan presisi. *Dataset* 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 *dataset* tersebut berjumlah 13, 1 kolom sebagai data target dari hasil klasifikasi. Parameter yang ada di dalam *dataset* 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 mendapatkan hasil bahwa akurasi tertinggi didapatkan pada algoritma *Random Forest* 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 koroner.

Penelitian serupa juga dilakukan oleh (Annisa, 2019) dengan judul **Analisis Komparasi Algoritma Data Mining Untuk Prediksi Penderita Penyakit Jantung**. 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*. *Dataset* yang digunakan untuk melakukan penelitian tersebut adalah *dataset* laki-laki penderita penyakit jantung yang terdiri dari 8 atribut, *dataset* 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 menggunakan uji t (*t-test*). Tahapan yang dilakukan di dalam pengujian tersebut adalah pembagian *dataset* 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 *dataset* tersebut.

Pada penelitian yang dilakukan oleh (Lestari, 2014) dengan judul **Penerapan Algoritma Klasifikasi Nearest Neighbor (K-NN) Untuk Mendeteksi Penyakit Jantung**. Penelitian tersebut dilakukan menggunakan algoritma *K-Nearest Neighbor* (K-NN) yang diterapkan untuk mendeteksi penyakit jantung. Dalam penelitian tersebut *dataset* 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 dataset* yang akan digunakan yaitu berjumlah 110 data. Selanjutnya, membagi *dataset* 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 (Rohman et al., 2017) dengan judul **Penerapan Algoritma C4,5 Berbasis Adaboost Untuk Prediksi Penyakit Jantung**. Penelitian tersebut membahas tentang prediksi penyakit jantung yang dilakukan menggunakan algoritma *Decision Tree* atau C4.5 yang dikombinasikan dengan metode *Adaboost*. Tujuan penelitian tersebut adalah melakukan kombinasi algoritma dan metode untuk mengoptimalkan atribut-atribut yang dimiliki oleh *dataset*, dan diharap dengan menerapkan metode *Adaboost* dapat

meningkatkan performa model dalam melakukan klasifikasi. *Dataset* yang digunakan di dalam penelitian tersebut adalah data gabungan dari *dataset* Cleveland yang berjumlah 303 data, *dataset* Statlog yang berjumlah 270 data, dan *dataset* 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 *dataset* 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.

Pada penelitian yang dilakukan oleh (Widiastuti et al., 2014) yang berjudul **Algoritma Klasifikasi Data Mining Naïve Bayes Berbasis Particle Swarm Optimization Untuk Deteksi Penyakit jantung**. 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*.

Penelitian yang dilakukan oleh (Utomo & Mesran, 2020) dengan judul **Analisis Komparasi Metode Klasifikasi Data Mining dan Reduksi Atribut Pada Data Set Penyakit Jantung**. 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*, *dataset* 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 (Normawati & Winarti, 2017) dengan judul **Seleksi Fitur Menggunakan Penambangan Data Berbasis Variable Precision Rough Set (VPRS) Untuk Diagnosis Penyakit Jantung Koroner**. 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 *dataset Cleveland Heart Disease* yang berjumlah 303 data yang bersumber dari UCI *machine learning*. *Dataset* 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 *dataset*, *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 (Gunawan et al., 2020) dengan judul **Sistem Diagnosis Otomatis Identifikasi Penyakit Jantung Koroner Menggunakan Ciri GLCM dan Klasifikasi SVM**. 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.

Pada penelitian yang dilakukan oleh (Omer et al., 2018) dengan judul ***Deep Neural Network for Heart Disease Medical Prescription Expert System***. 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*. *Dataset* yang digunakan di dalam penelitian tersebut adalah data pasien yang terkait dengan IHD yang bersumber dari sebuah rumah sakit di Jakarta. *Dataset* 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 et al., 2019) dengan judul ***The Methods of Duo Output Neural Network Ensemble for Prediction of Coronary Heart Disease***. 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.

Penelitian yang dilakukan oleh (Pareza Alam Jusia, 2018) dengan judul **Analisis Komparasi Pemodelan Algoritma Decision Tree Menggunakan Metode Particle Swarm Optimization Dan metode Adaboost Untuk Prediksi Awal Penyakit Jantung**. 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. *Dataset* 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 dataset*, 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 (Retnasari & Rahmawati, 2017) dengan judul **Diagnosa Prediksi Penyakit Jantung Dengan Model Algoritma Naïve Bayes dan Algoritma C4,5**. 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 *dataset* 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.

Terdapat penelitian tentang penyakit jantung yang dilakukan oleh (Aini et al., 2018) dengan judul **Seleksi Fitur *Information Gain* Untuk Klasifikasi Penyakit Jantung Menggunakan Kombinasi Metode *K-Nearest Neighbor* dan *Naive Bayes***. 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*. *Dataset* 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 (Karo, 2020) dengan judul **Implementasi Metode XgBoost dan *Feature Important* Untuk Klasifikasi Pada Kebakaran Hutan dan Lahan**. 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%.

Pada penelitian yang dilakukan oleh (Yualinda et al., 2020) dengan judul ***Application Based On ECommerce Dataset For Poverty Prediction Using Naive Bayes Algorithm, XgBoost, And Simillarity Based Feature Selection***. 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.

Penelitian yang dilakukan oleh (Syukron et al., 2020) dengan judul **Perbandingan Metode *Smote Random Forest* dan *Smote XgBoost* Untuk Klasifikasi Tingkat Penyakit Hepatitis C Pada *Imbalance Class Data***. 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 (Pinata et al., 2020) dengan judul **Prediksi Kecelakaan Lalu Lintas di Bali dengan XgBoost Pada Python**. 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.

Penelitian yang dilakukan oleh (Ibrahem Ahmed Osman et al., 2020) dengan judul ***Extreme Gradient Boosting (XgBoost) model to predict the groundwater level in Selangor Malaysia***. 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.

Ringkasan penelitian-penelitian sebelumnya yang sudah disebutkan di atas dapat dilihat pada Tabel 2.1 dibawah ini. Dalam penelitian ini akan dilakukan identifikasi penyakit jantung dengan data tabular berupa data rekam jantung yang diklasifikasikan menggunakan algoritma XgBoost. Nantinya, hasil identifikasi tersebut akan menunjukan berdasarkan data yang diklasifikasi akan menunjukan hasil klasifikasi berupa label 1 untuk terkena penyakit jantung, dan label 0 untuk tidak terkena penyakit jantung. Algoritma XgBoost dipilih karena memiliki performa yang baik, memiliki waktu komputasi yang cepat, dan dapat mengatasi permasalahan model yang *overfitting* karena memiliki *regularization*.

Tabel 2. 1 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 Diagnosis Penyakit Jantung	2020	Algoritma C4,5 dan *Bagging*	Algoritma C4,5 murni akurasi 72,98% Algoritma C4,5 digabungkan *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%

Tabel 2. 2 Lanjutan Tabel State of the Art

No.	Penulis	Judul	Tahun	Metode	Hasil
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 Precision Rough Set* (VPRS) Untuk Diagnosis Penyakit Jantung Koroner	2017	Algoritma *Variabel Precision Rough Set* (VPRS) dan MFS	Akurasi kombinasi 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%

Tabel 2. 3 Lanjutan Tabel State of The Art

No.	Penulis	Judul	Tahun	Metode	Hasil
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 Kebakaran Hutan dan Lahan	2020	Algoritma XgBoost	Nilai SE sebesar 91,32% SP sebesar 93,16% 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 *dataset* 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. Contoh dataset yang akan digunakan ditampilkan dalam Tabel 2.4 di bawah ini.

Tabel 2. 4 Tabel Contoh Dataset

age	sex	cp	Trestbp	chol	fbs	restecg	thalac	exang	oldpeak	slope	ca	thal	Target
63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
58	1	1	120	284	0	0	160	0	1.8	1	0	2	0
58	1	2	132	224	0	0	173	0	3.2	2	2	3	0

Parameter yang digunakan dalam pendeteksian penyakit jantung dapat dilihat pada Tabel 2.5 dan Tabel 2.6 di bawah ini.

Tabel 2. 5 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*

Tabel 2. 6 Lanjutan Parameter Data

Atribut	Deskripsi	Keterangan
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*	1:*normal*, 2:cacat tetap, 3: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.

$$similarity = \frac{\sum(Residual_i)^2}{\sum[prev\ probability_i \times (1 - prev\ probability_i)] + \lambda} \quad \text{.....(2.1)}$$

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.

$gain = left\ similarity + right\ similarity - root\ similarity$ .....(2.2)

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*(γ), apabila nilai $(gain - \gamma)$ adalah negatif maka *node* tersebut dapat dipangkas atau dihapus. Rumus untuk menghitung *cover* adalah sebagai berikut.

$cover = \sum[prev\ probability_i \times (1 - prev\ probability_i)]$.....(2.3)

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.

$$O_{value} = \frac{\sum(Residual_i)}{\sum[prev\ probability_i \times (prev\ probability_i)] + \lambda}$$.....(2.4)

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.

$$odds = \frac{p}{1-p} \quad (2.5)$$

$$\log(odds) = \log\left(\frac{p}{1-p}\right) \quad (2.6)$$

$$\log(odds)\, probability = \log\left(\frac{p}{1-p}\right) + \sum[\varepsilon \times O_{value}]_i \quad (2.7)$$

$$probability = \frac{e^{\log(odds) probability}}{1 + e^{\log(odds) probability}} \quad (2.8)$$

Keterangan:

$p = initial\ probability$

$\varepsilon = learning\ rate$, 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.

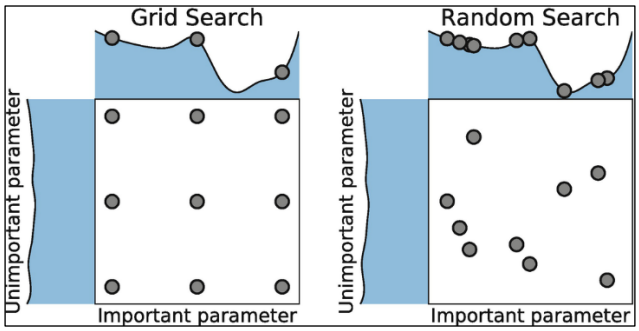
Tabel 2. 7 Parameter Xgboost

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

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

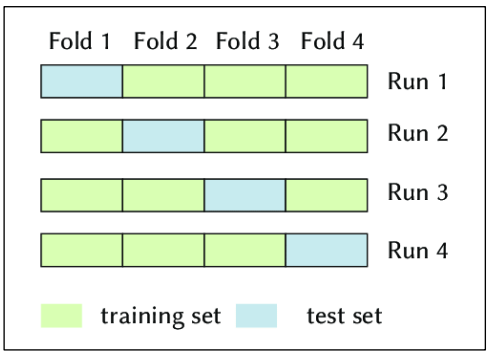
cepat tanpa harus mencoba seluruh pilihan yang diberikan. Untuk lebih jelas, berikut terdapat ilustrasi perbandingan *grid search* dan *randomized search optimizer* dalam menentukan nilai parameter.



Gambar 2. 1 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. 2 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 2. 8 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.

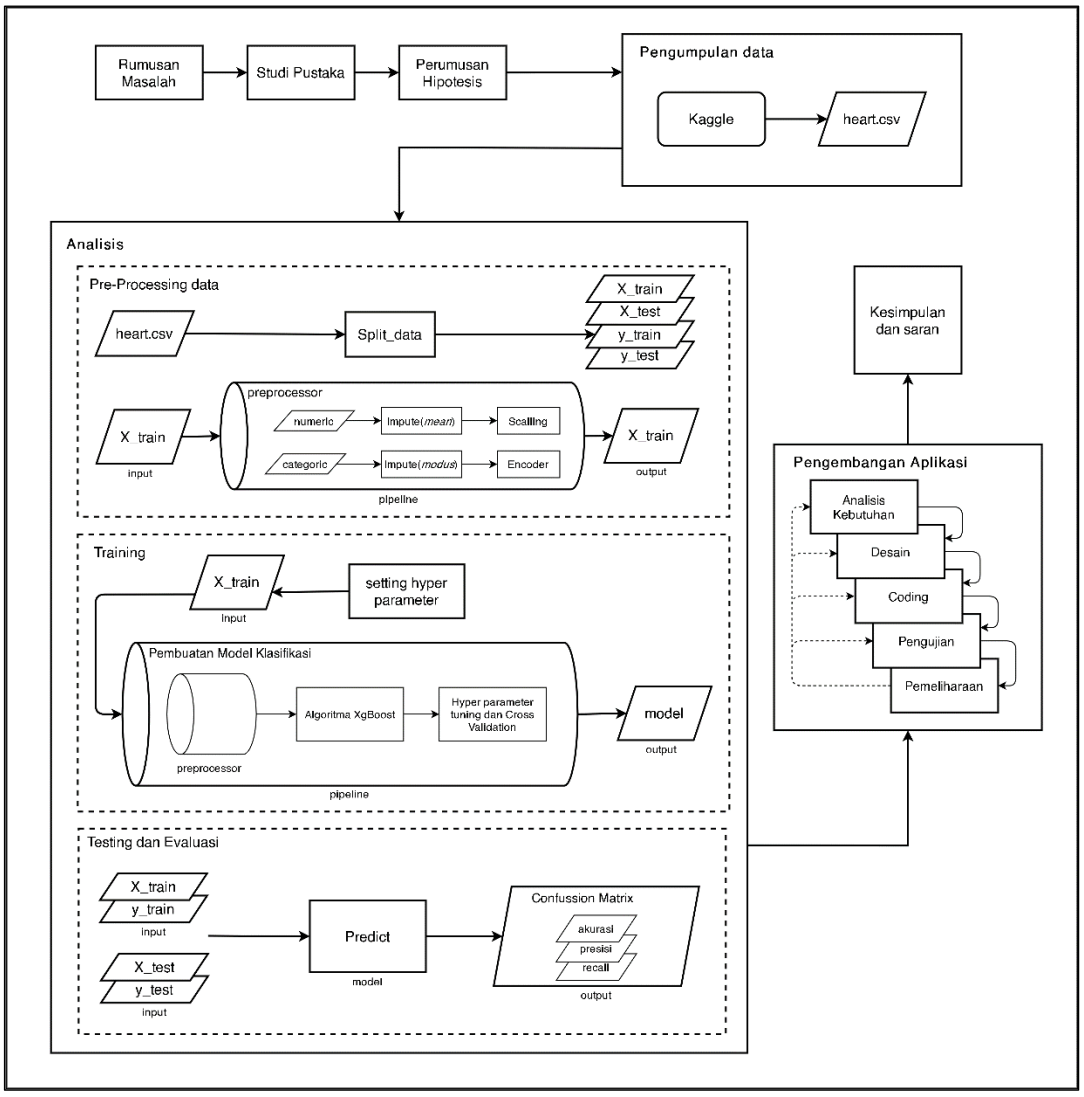
1. *Accuracy* = $\frac{TP+TN}{TP+FP+FN+TN}$ .....(2.9)
   $Accuracy$ adalah rasio prediksi benar (positif dan negatif) dengan keseluruhan data yang ada.
2. *Recall* = $\frac{TP}{TP+FN}$.....(2.10)
   $Recall$ adalah rasio prediksi jumlah benar positif dibandingkan dengan keseluruhan data yang benar positif.
3. *Precision* = $\frac{TP}{TP+FP}$.....(2.11)
   $Precision$ atau presisi adalah rasio prediksi jumlah benar positif dibandingkan dengan keseluruhan hasil yang diprediksi positf.
4. *F1 score* = $2 * \frac{recall*precision}{recall+precision}$.....(2.12)
   $F1\ score$ adalah perbandingan rata-rata nilai presisi dan nilai recall yang dibobotkan.

BAB III
METODOLOGI PENELITIAN

3.1.Metode Penelitian

Metode penelitian yang digunakan dalam penelitian ini adalah penelitian kuantitatif. Sedangkan jenis penelitian yang akan dilakukan 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*, sedangkan pengembangan yang dilakukan dalam penelitian ini adalah pembuatan model *machine learning* yang akan digunakan untuk melakukan klasifikasi pada prediksi penyakit jantung. Data yang akan digunakan adalah data tabular atau data berbentuk tabel, sedangkan *output* klasifikasi yang dihasilkan adalah label Ya untuk data yang diklasifikasikan terkena penyakit jantung, atau Tidak untuk data yang diklasifikasikan tidak terkena penyakit jantung. Model tersebut dibuat menggunakan algoritma XgBoost dan menggunakan *randomized search optimizer* untuk membantu menentukan *hyper 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*.

Penelitian ini diawali dengan perumusan rumusan masalah, kemudian dilanjutkan studi literatur yang dilakukan dengan mencari informasi mengenai topik yang bersangkutan lewat sumber-sumber 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 mengetahui perkembangan topik prediksi penyakit jantung seperti algoritma apa saja yang sudah pernah 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. Setelah itu, dilakukan pengumpulan data dan analisis menggunakan algoritma XgBoost. Dilanjutkan dengan pembuatan sistem agar sistem prediksi dapat digunakan oleh pengguna, dan menemukan kesimpulan dan saran dari penelitian yang dilakukan. Tahapan penelitian dapat dilihat pada Gambar 3.1 di bawah ini.



Gambar 3. 1 Tahapan Penelitian

Tahapan-tahapan penelitian yang akan dilakukan digambarkan pada gambar 3.1 diatas. Perincian setiap tahapan penelitian yang dilakukan akan dijelaskan di bawah ini.

3.1.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 *dataset* terdapat pada Tabel 3.1 di bawah ini.

Tabel 3. 1 Kolom Pada *Dataset*

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*	1:*normal*, 2:cacat tetap, 3:cacat *reversible*

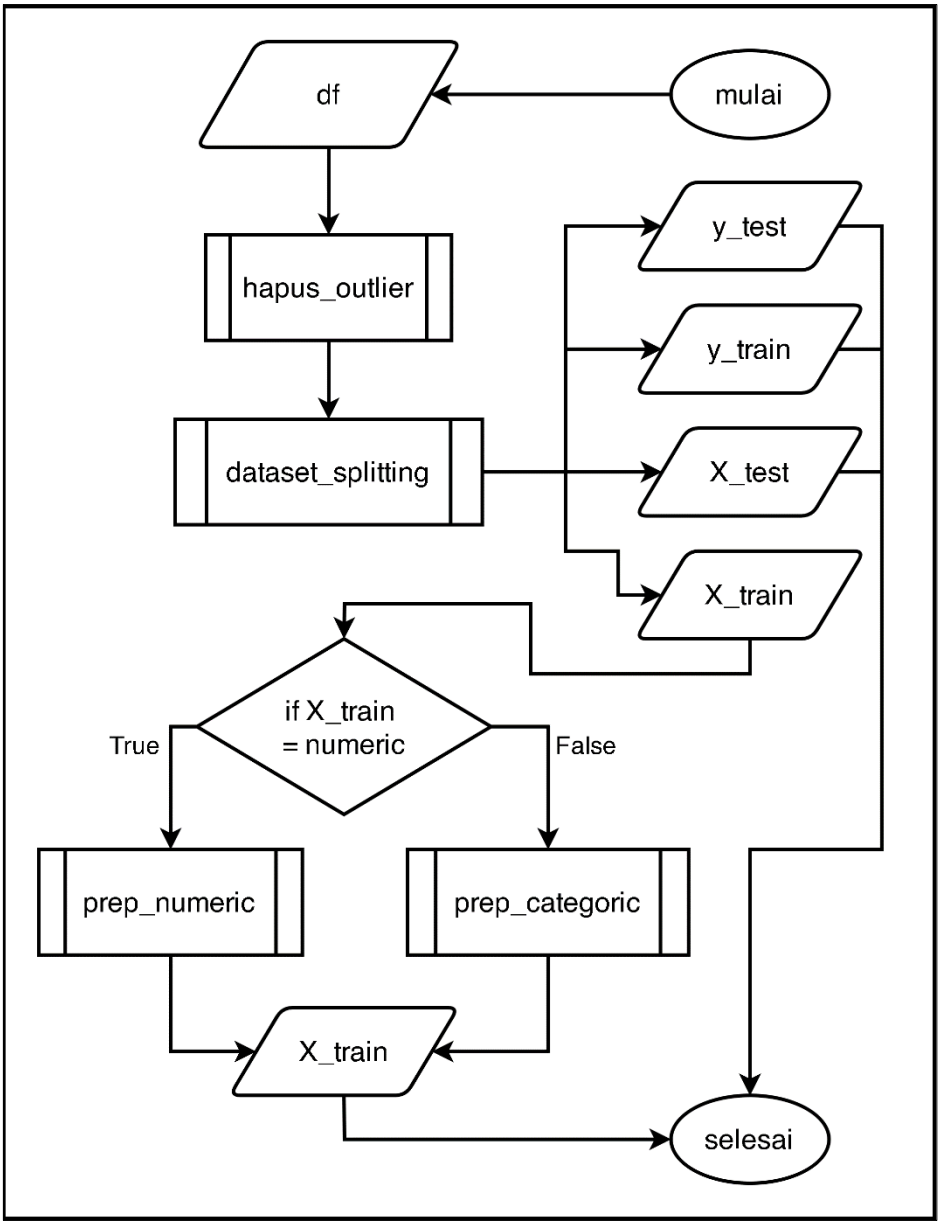
Sedangkan untuk contoh data mentah yang akan digunakan dapat dilihat pada Tabel 3.2 di bawah ini.

Tabel 3. 2 Dataset

Age	Sex	cp	Trestbp	chol	fbs	restecg	thalac	exang	oldpeak	slope	ca	thal	Target
63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
58	1	1	120	284	0	0	160	0	1.8	1	0	2	0
58	1	2	132	224	0	0	173	0	3.2	2	2	3	0

3.2.Preprocessing

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). *Flowhcart* tahapan *preprocessing* terdapat pada gambar 3.2 dibawah ini.



Gambar 3. 2 Flowchart preprocessing

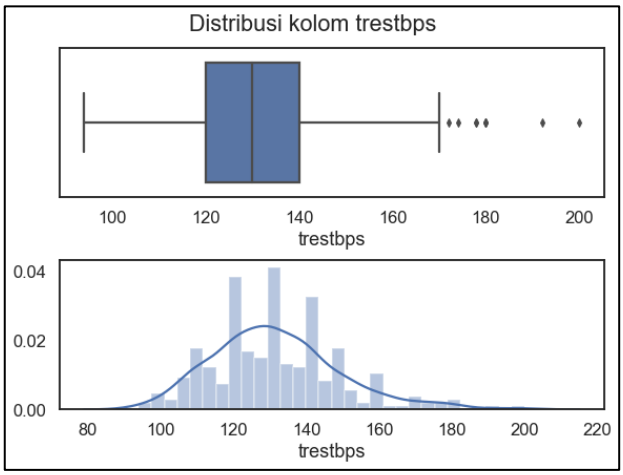
Dalam preprocessing, tahapan yang akan dilakukan adalah sebagai berikut.

1. Menghapus *outlier*.
2. Membagi data menjadi *data training* dan *data testing*.
3. Normalisasi pada data *numeric*.
4. *One hot encoding* pada data *categoric*.

3.2.1. Menghapus Outlier

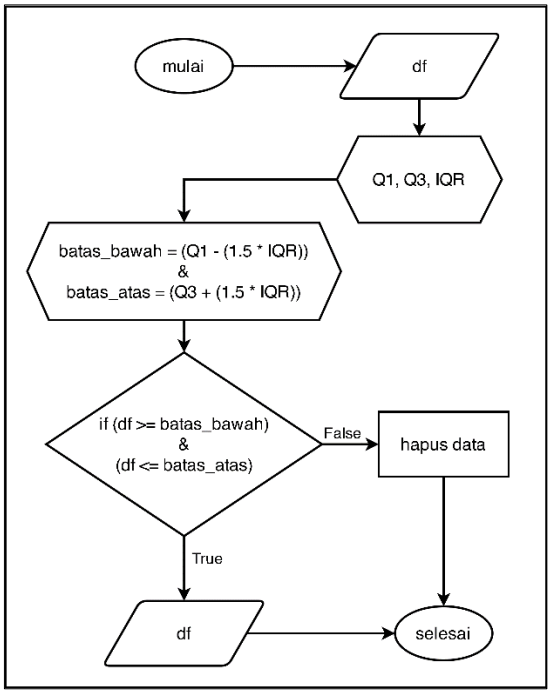
Tahap *preprocessing* yang pertama adalah membersihkan *outlier* data, apabila data yang digunakan memiliki nilai terlalu kecil atau terlalu besar dari dominan rentang datanya, maka data *outlier* tersebut dapat dihilangan supaya data lebih baik dan menghasilkan model yang lebih baik pula. Untuk mencari *outlier* pada data dapat dilihat dengan menggunakan *boxplot*

dan *distplot* seperti yang terlihat pada Gambar 3.3 di bawah ini. Pada gambar di bawah ini terlihat bahwa kolom tersebut memiliki *outlier* pada data di atas, yang terlihat dalam *boxplot*.



Gambar 3. 3 *Boxplot* dan *Distplot* kolom trestbps

Proses membersihkan *outlier* dilakukan dengan menghitung nilai IQR terlebih dahulu, selanjutnya ditentukan batas atas dan batas bawah dengan cara operasi perhitungan $Q1$ atau $Q3$ terhadap nilai IQR yang sudah ditentukan. *Flowchart* dari proses pembersihan oulier dapat dilihat pada Gambar 3.4 di bawah ini.



Gambar 3. 4 Flowchart hapus_*outlier*

Rumus untuk perhitungan menghapus *outlier* dapat dilihat pada persamaan di bawah ini.

$$Q1 = X\frac{1\times(n+1)}{4} \quad \text{.....(3.1)}$$

$Q3 = X\frac{3\times(n+1)}{4}$ .....(3.2)
$IQR = Q3 - Q1$ .....(3.3)
$batas\_bawah = Q1 - (1.5 \times IQR)$.....(3.4)
$batas\_atas = Q3 + (1.5 \times IQR)$.....(3.5)

Data dalam kolom trestbps dapat dilihat pada Tabel 3.3 di bawah ini. Sesuai dengan tahapan dan rumus yang telah dijelaskan di atas, maka penghapusan *oulier* pada kolom trestbps adalah sebagai 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

Tabel 3. 3 Data pada kolom trestbps

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

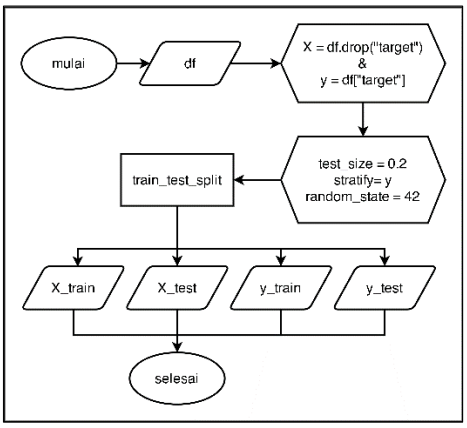
Sesuai dengan perhitungan diatas, kemudian dapat dilakukan *filter* pada kolom trestbps hanya menyisakan data yang memiliki nilai >= 90 dan <= 170. Sehingga nilai min dan max pada kolom tersebut dapat dilihat pada Tabel 3.4 di bawah ini.

Tabel 3. 4 Inforrmasi kolom trestbps

	Trestbps
Min	94
Mean	130
Max	170

3.2.2. Dataset Splitting

Pembagian *dataset* 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*. Flowchart proses *dataset splitting* dapat dilihat pada Gambar 3.5 di bawah ini.

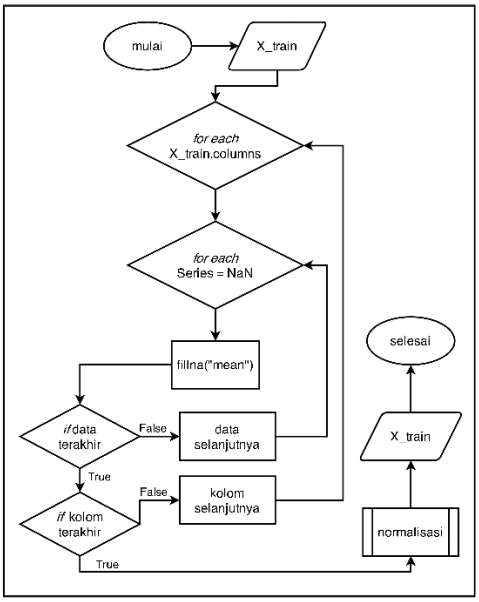


Gambar 3. 5 Flowchart dataset_splitting

Dari Gambar 3.5 di atas dapat dilihat bahwa hasil atau output dari proses tersebut akan menghasilkan 4 *dataframe* baru berupa *X_train*, *X_test*, *y_train*, dan *y_test*. Selanjutnya data yang masih akan dilakukan *preprocessing* hanya data *X_train*.

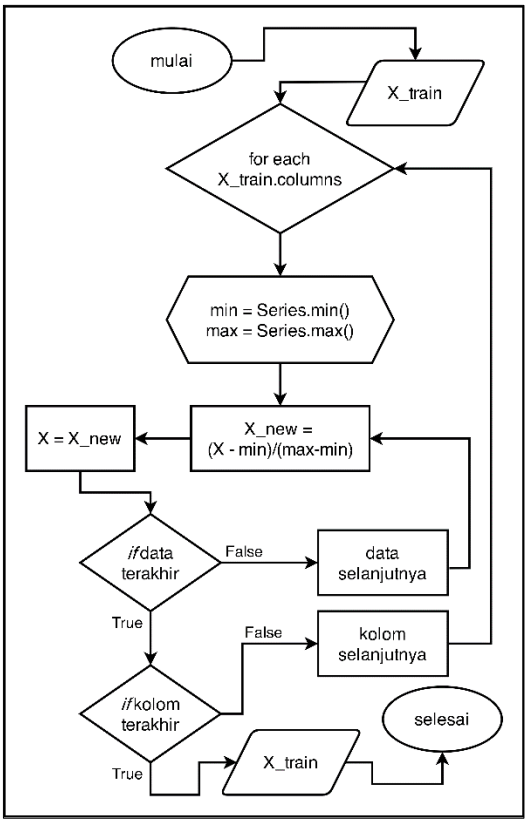
3.2.3. Data Numerik

Selanjutnya data *X_train* akan dibagi menjadi dua, yaitu data yang memiliki nilai numerik, dan data yang memiliki nilai kategorik. Untuk data numerik akan dilakukan proses *impute* menggunakan nilai rata-rata dari kolom tersebut, dan dilanjutnya dengan proses normalisasi berupa *scalling*. Flowchart proses preprocessing pada data numerik dapat dilihat pada gambar 3.6 di bawah ini.



Gambar 3. 6 Flowchart prep_numeric

Setelah data setiap kolom dilakukan impute, maka dilanjutkan dengan normalisasi berupa *scalling*. Proses normalisasi dilakukan supaya proses *training* menjadi lebih cepat, karena dapat memudahkan model dalam memahami data (Hanifa et al., 2017). *Scalling* dalam proses ini dilakukan dengan menggunakan *MinMax*, yaitu merubah data ke dalam range 0 untuk data terkecil dan 1 untuk data terbesar. Flowchart dari proses *scalling* dapat dilihat pada Gambar 3.7 di bawah ini.



Gambar 3. 7 Flowchart normalisasi MinMax

Rumus dan data yang akan digunakan dapat dilihat pada Tabel 3.5 dan 3.6 di bawah ini.

Tabel 3. 5 Contoh data untuk scalling

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

Tabel 3. 6 Lanjutan contoh data untuk scalling

No.	Data lama	Data Baru
289.	164	0.92
290.	140	0.6
291.	110	0.21
292.	144	0.65
293.	130	0.47
294.	130	0.47

$X_{new} = \frac{X - X_{min}}{X_{max} - X_{min}}$ .....(3.6)

Keterangan:

$X_{min} = data\ terkecil$

$X_{max} = data\ terbesar$

Sehingga dari contoh data yang ditampilkan dalam Tabel 3.5 dan Tabel 3.6 dapat diketahui bahwa $X_{min} = 94$ dan nilai $X_{max} = 170$. Contoh perhitungan pada data ke-1 adalah sebagai berikut, untuk dapat lainnya dapat dilihat pada Tabel 3.5 dan Tabel 3.6 di atas.

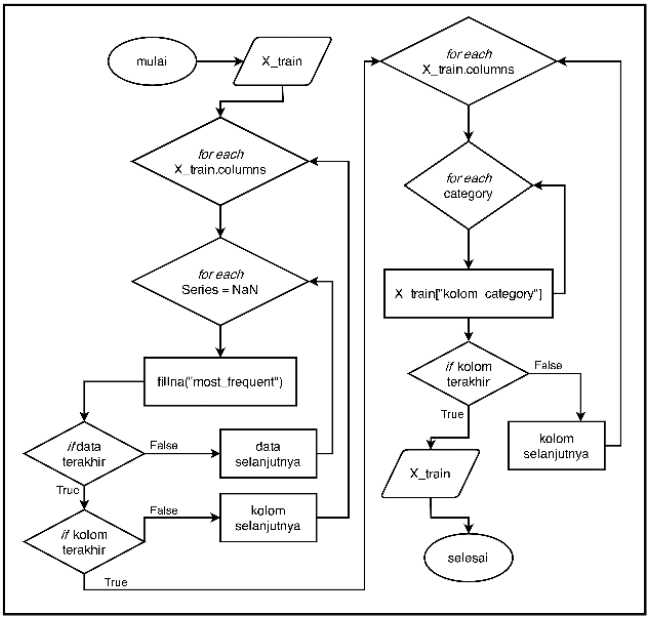
$$X_{new} = \frac{145 - 94}{170 - 94}$$

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

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

3.2.4. Data Kategorik

Kemudian pada data kategorik dilakukan *preprocessing* berupa *impute* data yang kosong dengan *most_frequent* atau modus dari kolom tersebut, dilanjutkan proses *encoding* menggunakan *One Hot Encoding*. Flowchart proses *preprocessing* pada data kategorik dapat dilihat pada Gambar 3.8 di bawah ini.



Gambar 3. 8 Flowchart prep_categoric

Untuk proses *One Hot Encoding* adalah proses mengganti nilai kategorikal dengan nilai *binary* berupa 0 atau 1. Dilakukan dengan cara membuat kolom baru sebanyak kategori yang ada, kemudian kolom baru yang sesuai dengan nilai kategori tersebut akan diberi nilai 1 dan kolom yang tidak sesuai dengan kategori akan diberi nilai 0. Contoh penerapan *One Hot Encoding* dapat dilihat pada Tabel 3.7 dan hasilnya ditampilkan pada Tabel 3.8 di bawah ini.

Tabel 3. 7 Contoh data untuk One Hot Encoding

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

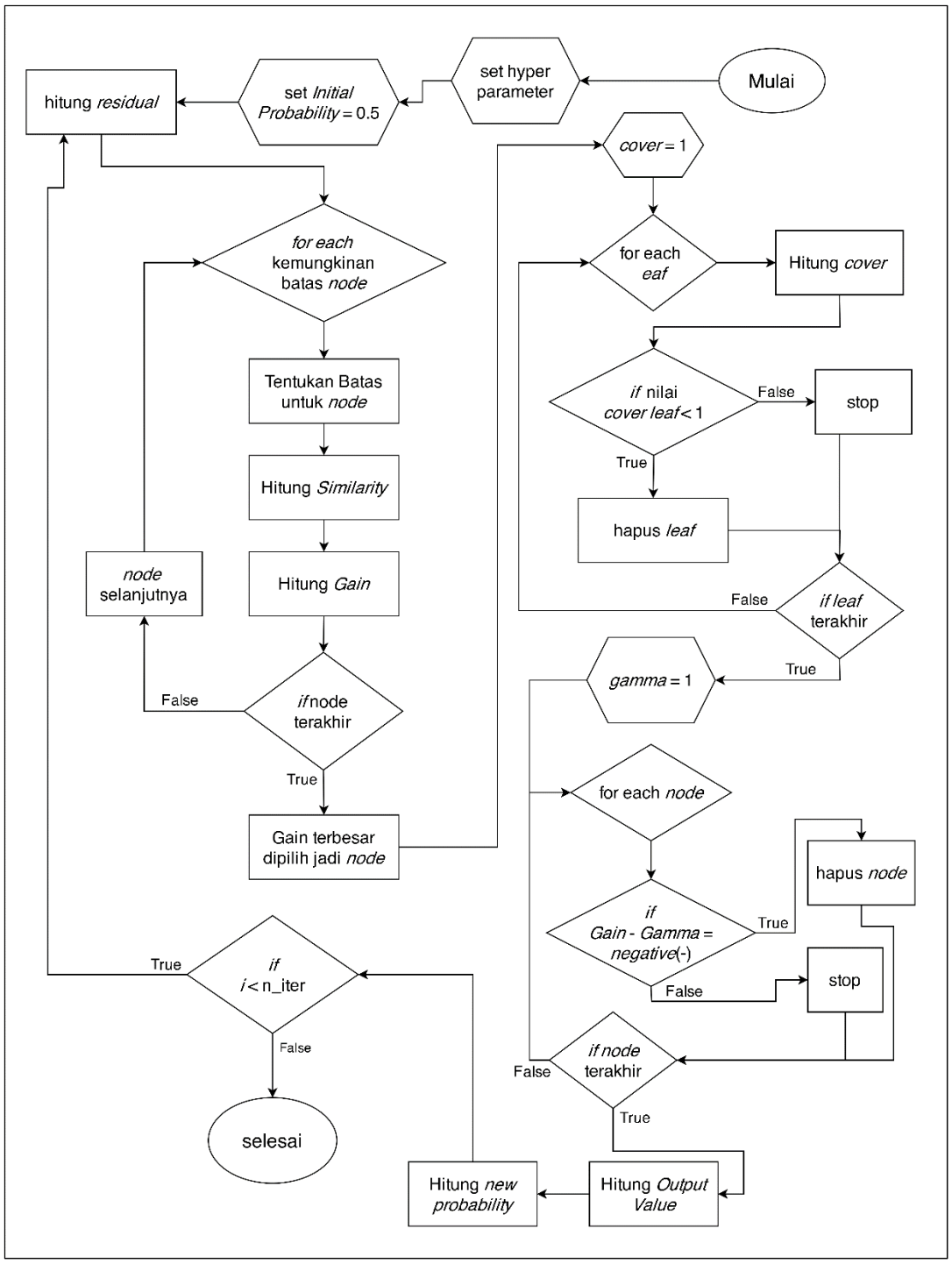
Tabel 3. 8 Hasil proses One Hot Encoding

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

Dapat dilihat pada Tabel 3.8 di atas, bawah nilai kategori *slope* akan menjadi kolom baru sesuai kategori yang ada. Dalam contoh di atas nilai kategori yang ada adalah 0, 1, 2. Kemudian kolom yang baru akan diberi nilai 1 jika kategori sesuai, dan 0 jika tidak sesuai.

3.3.Training

Proses training yang dilakukan oleh algoritma xgboost dapat dilihat pada flowchart Gambar 3.9 di bawah ini.



Gambar 3. 9 Flowchart XgBoost

Sesuai dengan Gambar 3.9 *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 yang ditampikan pada persamaan 3.7 di bawah ini.

$$residual = actual\ value - probability \quad \text{.....(3.7)}$$

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 $gain$ dapat dilihat pada persamaan 3.8 dan 3.9 di bawah ini.

$$similarity = \frac{\sum(residual_i)^2}{\sum[prev\ probability_i \times (1 - prev\ probability_i)] + \lambda} \quad \text{.....(3.8)}$$

$$gain = left\ similarity + right\ similarity - root\ similarity \quad \text{.....(3.9)}$$

Kemudian, setelah satu *tree* terbentuk dapat dilakukan pengecekan untuk memastikan apakah *tree* tersebut perlu dipangkas atau tidak dengan menggunakan *cover* dan *gamma*(γ). 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* dapat dilihat pada persamaan 3.10 berikut.

$$cover = \sum[prev\ probability_i \times (1 - prev\ probability_i)] \quad \text{..... (3.10)}$$

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.

$$O_{value} = \frac{\sum(residual_i)}{\sum[prev\ probability_i \times (1 - prev\ probability_i)] + \lambda} \quad \text{..... (3.11)}$$

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.

$$\log(odds) = \log\left(\frac{p}{1-p}\right) + \sum[\varepsilon \times O_{value}]_i \quad \text{..... (3.12)}$$

$$probability = \frac{e^{\log(odds)}}{1 + e^{\log(odds)}} \quad \text{..... (3.13)}$$

Contoh *dataset* yang akan digunakan untuk proses *training* terdapat pada Tabel 3.9 di bawah ini.

Tabel 3. 9 Contoh Dataset Untuk Training

Cp	Thalach	Target
3	150	1
3	190	1
0	96	0
1	174	0
1	202	1

Pertama, buat *initial probability* dengan nilai 0,5. Kemudian dapat dihitung nilai *residual* masing-masing data dengan rumus $actual\ value - probability$. Kemudian buat tabel untuk memperbarui data nilai $residual$ yang sudah dihitung. Nilai $residual$ dapat dilihat pada tabel 3.10 di bawah ini.

Tabel 3. 10 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 $similarity$ yang nantinya akan dijadikan $Root_{similarity}$ pada saat perhitungan nilai $gain$ untuk masing-masing kemunkinan *node* yang ada. Perhitungan $similarity$ adalah sebagai berikut. Untuk perhitungan dengan data di atas, kita asumsikan nilai $lambda(\lambda)$ adalah 0.

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

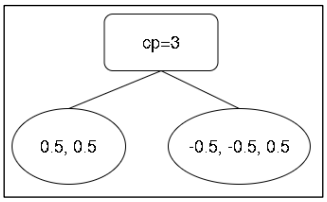
$$Root_{similarity} = \frac{(0.5)^2}{0.25 \times 5}$$

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

$$Root_{similarity} = 0.2$$

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. 10 Tree Untuk Node Cp=3

Pertama, yang dilakukan adalah menghitung $similarity$ untuk *leaf* sebelah kiri yang memiliki nilai 0,5 dan 0,5. Cara menghitung sama seperti rumus $similarity$ yang sudah dijelaskan di atas, perhitungan $Left_{similarity}$ dapat dilihat di bawah ini.

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

Kemudian dilanjutkan dengan menghitung $similarity$ untuk *leaf* sebelah kanan, yang memiliki tiga nilai yaitu -0.5, -0.5 dan 0.5. Perhitungan $Right_{similarity}$ dapat dilihat di bawah ini.

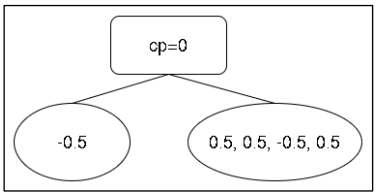
$$Right_{similarity} = \frac{(-0.5 - 0.5 + 0.5)^2}{\left(\left(0.5 \times (1 - 0.5)\right) \times 3\right) + 0}$$
$$Right_{similarity} = \frac{(-0.5)^2}{(0.25 \times 3)}$$
$$Right_{similarity} = \frac{0.25}{0.75}$$
$$Right_{similarity} = 0.33$$

Setelah mengetahui nilai $Left_{similarity}$ dan $Right_{similarity}$, langkah selanjutnya adalah menghitung nilai $gain$. Nilai $gain$ didapatkan dengan menghitung jumlah $similarity$ dikurangi dengan $Root_{similarity}$. Perhitungan nilai $gain$ untuk node cp=3 dapat dilihat di bawah ini.

$$gain = Left_{similarity} + Right_{similarity} - Root_{similarity}$$
$$gain = 2 + 0.33 - 0.2$$
$$gain = 2.13$$

Setelah kita mendapatkan nilai $gain$ dari *probabillity node* ini, simpan nilai $gain$ itu dan kita hitung nilai $gain$ untuk *probabillity node* selanjutnya.

2. Cp = 0



Gambar 3. 11 Tree Untuk Node Cp=0

Selanjutnya adalah perhitungan untuk *probabillity node* berupa cp=0. Pertama kita gambarkan *probabillity node* seperti yang terlihat pada Gambar 3.11 di atas. Kemudian kita hitung nilai $similarity$ untuk masing-masing *leaf*. Perhitungan $Left_{similarity}$ dapat dilihat di bawah ini.

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

Kemudian menghitung nilai $Right_{similarity}$ yang memiliki nilai $residual$ berupa 0.5, 0.5, -0.5, dan 0.5. perhitungan $Right_{similarity}$ dapat dilihat di bawah ini.

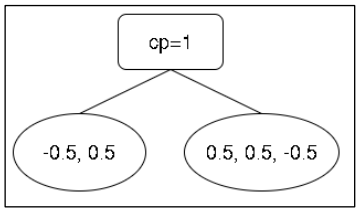
$$Right_{similarity} = \frac{(0.5 + 0.5 - 0.5 + 0.5)^2}{\big(\big(0.5 \times (1 - 0.5)\big) \times 4\big) + 0}$$
$$Right_{similarity} = \frac{(0.5 + 0.5)^2}{(0.25 \times 4) + 0}$$
$$Right_{similarity} = \frac{1}{1}$$
$$Right_{similarity} = 1$$

Setelah menghitung nilai $Left_{similarity}$ dan nilai $Right_{similarity}$, kemudian dilanjutkan menghitung nilai $gain$ dari *probabillity node* ini. Perhitungan nilai $gain$ pada *probabillity node* dilakukan dengan menjumlahkan $similarity$ dan dikurangi dengan nilai $Root_{similarity}$, seperti yang dapat dilihat di bawah ini.

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

Setelah mendapatkan nilai $gain$, simpan nilai $gain$ tersebut untuk nantinya dibandingkan dengan $gain$ *probabillity node* lainnya. Kemudian dilanjutkan menghitung $similarity$ dari *probabillity node* lainnya.

3. Cp = 1



Gambar 3. 12 Tree Untuk Node Cp=1

Probabillity node selanjutnya adalah cp=1. Pertama kita gambarkan *probabillity node* tersebut seperti pada Gambar 3.12 di atas, kemudian dilanjutkan menghitung nilai $similarity$ baik dari *leaf* sebelah kiri dan sebelah kanan. Untuk perhitungan $Left_{similarity}$ dapat dilihat pada operasi perhitungan di bawah ini.

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

$$Left_{similarity} = \frac{0^2}{0.25 \times 2}$$

$$Left_{similarity} = 0$$

Kemudian setelah mendapatkan nilai $Left_{similarity}$, selanjutnya adalah menghitung nilai $similarity$ untuk leaf sebelah kanan yang berisi nilai $residual$ berupa 0.5, 0.5, dan -0.5. Perhitungan $Right_{similarity}$ dapat dilihat di bawah ini.

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

$$Right_{similarity} = \frac{0.5^2}{0.25 \times 3}$$

$$Right_{similarity} = \frac{0.25}{0.75}$$

$$Right_{similarity} = 0.33$$

Setelah mendapatkan nilai $Left_{similarity}$ dan $Right_{similarity}$, selanjutnya menghitung $gain$ dari *probabillity node* tersebut dengan cara menjumlahkan $similarity$ dan dikurangi dengan $Root_{similarity}$ yang sudah dihitung di awal. Perhitungan $gain$ dapat dilihat di bawah ini.

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

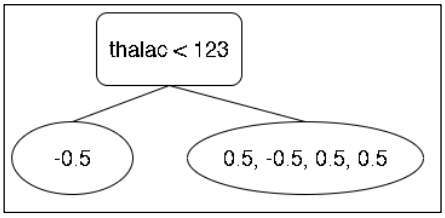
$$gain = 0 + 0.33 - 0.2$$

$$gain = 0.13$$

Nilai $gain$ yang sudah didapat disimpan untuk nantinya dibandingkan dengan $gain$ *probabillity node* lainnya. Selanjutnya adalah menghitung $similarity$ dan $gain$ dari *probabillity node* selanjutnya.

4. Thalac < 123

Untuk menentukan *probabillity node* ini kita melakukan sorting pada nilai numerik kemudian mengambil nilai tengah dari dua nilai numerik yang ada. Diawali dari *node* ini, maka *probabillity node* adalah $96 + (\frac{150-96}{2})$ yaitu 123. Kemudian kita buat *probabillity node* thalac<123. Gambar dari probabillity node ini dapat dilihat pada Gambar 3.13 di bawah ini.



Gambar 3. 13 Tree Untuk Node Thalac<123

Kita menghitung nilai $similarity$ dari leaf sebelah kiri yang hanya berisi satu nilai $residual$. Perhitungan $Left_{similarity}$ dapat dilihat di bawah ini.

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

$$Left_{similarity} = \frac{0.25}{0.25}$$

$$Left_{similarity} = 1$$

Setelah mendapatkan nilai $Left_{similarity}$, maka kita juga menghitung nilai $Right_{similarity}$ yang berisi empat nilai $residual$ seperti yang terlihat pada Gambar 3.13 di atas. Perhitungan $Right_{similarity}$ dapat dilihat seperti di bawah ini.

$$Right_{similarity} = \frac{(0.5 - 0.5 + 0.5 + 0.5)^2}{\Big(\big(0.5 \times (1-0.5)\big) \times 4\Big) + 0}$$

$$Right_{similarity} = \frac{(0.5 + 0.5)^2}{0.25 \times 4}$$

$$Right_{similarity} = \frac{1}{1}$$

$$Right_{similarity} = 1$$

Kemudian setelah mendapatkan $Left_{similarity}$ dan $Right_{similarity}$ pada *probabillity node* ini, selanjutnya adalah mencari nilai $gain$ dengan cara menjumlahkan nilai $similarity$ dikurangi dengan nilai $Root_{similarity}$ seperti yang dapat dilihat pada perhitungan di bawah ini.

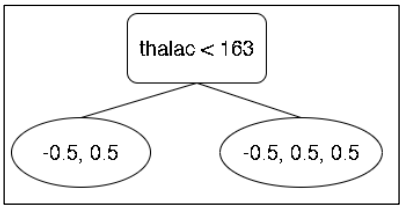
$$gain = Left_{similarity} + Right_{similarity} - Root_{similarity}$$

$$gain = 1 + 1 - 0.2$$

$$gain = 1.8$$

5. Thalac < 163

Untuk menentukan *probabillity node* ini, maka kita mencari nilai tengah dari kedua nilai numerik 150 dan 174. Oleh karena itu dilakukan perhitungan $150 + (\frac{174-150}{2})$ dan mendapatkan nilai 163 sebagai batas *node*. Gambar *probabillity node* dapat dilihat pada Gambar 3.14 di bawah ini.



Gambar 3. 14 Tree Untuk Node Thalac<163

Pertama kita mencari nilai $similarity$ yang terdapat pada *leaf* sebelah kiri yang berisi dua nilai $residual$ yaitu -0.5 dan 0.5. perhitungan $Left_{similarity}$ dapat dilihat di bawah ini.

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

$$Left_{similarity} = \frac{0^2}{0.25 \times 2}$$

$$Left_{similarity} = 0$$

Selanjutnya kita mencari nilai $similarity$ untuk leaf sebelah kanan. Perhitungan $Right_{similarity}$ dapat dilihat seperti di bawah ini.

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

$$Right_{similarity} = \frac{0.5^2}{0.25 \times 3}$$

$$Right_{similarity} = \frac{0.25}{0.75}$$

$$Right_{similarity} = 0.33$$

Setelah kita mendapatkan kedua nilai $similarity$ pada masing-masing *leaf*, maka selanjutnya kita bisa mencari $gain$ untuk *probabillity node* ini. Perhitungan $gain$ dapat dilihat di bawah ini.

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

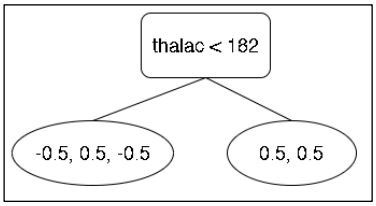
$$gain = 0 + 0.33 - 0.2$$

$$gain = 0.13$$

Setelah mendapatkan nilai $gain$, kita simpan nilai $gain$ pada *probabillity node* ini kemudian dilanjutkan mencari nilai $gain$ pada *probabillity node* lainnya.

6. Thalac < 182

Untuk menentukan batas pada *node* ini, kita mencari nilai tengah dari kedua nilai numerik 174 dan 190 dengan melakukan perhitungan $174 + (\frac{190-174}{2})$ dan didapatkan hasil 182 sebagai batas *probabillity node* ini. Gambar *probabillity node* dapat dilihat pada Gambar 3.15 di bawah ini.



Gambar 3. 15 Tree Untuk Node Thalac<182

Pertama kita mencari nilai $Left_{similarity}$ yang berisi tiga nilai $residual$ seperti yang terlihat pada Gambar 3.15 di atas. Perhitungan $Left_{similarity}$ dapat dilihat di bawah ini.

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

$$Left_{similarity} = \frac{-0.5^2}{0.25 \times 3}$$

$$Left_{similarity} = \frac{0.25}{0.75}$$

$$Left_{similarity} = 0.33$$

Setelah mendapatkan nilai $Left_{similarity}$, maka kita mencari $similarity$ untuk *leaf* selanjutnya yang ada di sebelah kanan. Perhitungan $Right_{similarity}$ dapat dilihat seperti di bawah ini.

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

$$Right_{similarity} = \frac{1^2}{0.25 \times 2}$$

$$Right_{similarity} = \frac{1}{0.5}$$

$$Right_{similarity} = 2$$

Setelah mendapatkan nilai $similarity$ dari kedua *leaf*, selanjutnya menjumlahkan $similarity$ tersebut dan dikurangi $Root_{similarity}$ untuk mendapatkan $gain$. Perhitungan nilai $gain$ pada *probabillity node* ini dapat dilihat di bawah ini.

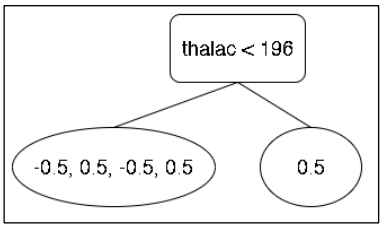
$$gain = Left_{similarity} + Right_{similarity} - Root_{similarity}$$

$$gain = 0.33 + 2 - 0.2$$

$$gain = 2.13$$

7. Thalac < 196

Untuk menentukan batas *probabillity node* ini, kita melakukan perhitungan $190 + (\frac{202-190}{2})$ untuk mencari nilai tengah dari dua nilai numerik yang ada pada kolom tersebut yaitu 190 dan 202, perhitungan tersebut menghasilkan 196 sebagai batas *node*. Gambar *probabillity node* ini dapat dilihat pada Gambar 3.16 di bawah ini.



Gambar 3. 16 Tree Untuk Node Thalac<196

Pertama kita mencari $similarity$ untuk *leaf* yang ada di sebelah kiri. *Leaf* tersebut berisi empat nilai $residual$ yaitu -0.5, 0.5, -0.5 dan 0.5. Perhitungan $Left_{similarity}$ dapat dilihat di bawah ini.

$$Left_{similarity} = \frac{(-0.5 + 0.5 - 0.5 + 0.5)^2}{\Big((0.5 \times (1 - 0.5)) \times 4\Big) + 0}$$

$$Left_{similarity} = \frac{0^2}{1}$$

$$Left_{similarity} = 0$$

Kemudian setelah mendapatkan nilai $Left_{similarity}$, selanjutnya adalah mencari nilai $similarity$ untuk *leaf* belah kanan yang hanya berisi satu nilai $residual$ yaitu 0.5. Perhitungan $Right_{similarity}$ dapat dilihat seperti di bawah ini.

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

$$Right_{similarity} = \frac{0.25}{0.25}$$

$$Right_{similarity} = 1$$

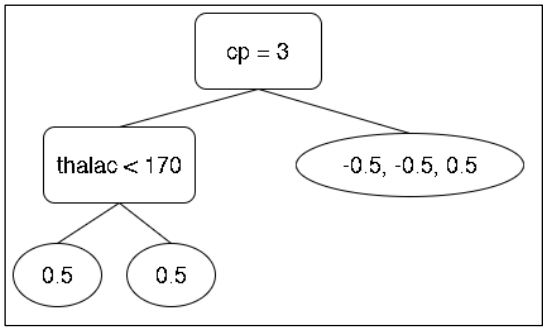
Kemudian setelah mendapatkan nilai $Left_{similarity}$ dan nilai $Right_{similarity}$, maka kita jumlahkan nilai $similarity$ tersebut dan dikurangi dengan nilai $Root_{similarity}$ untuk mendapatkan $gain$ *probabillity node* ini. Perhitungan $gain$ untuk *probabillity node* ini dapat dilihat di bawah ini.

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

$$gain = 0 + 1 - 0.2$$

$$gain = 0.8$$

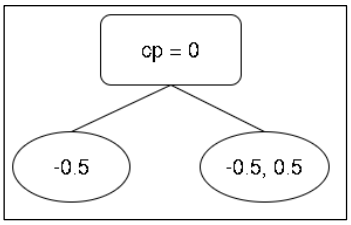
Setelah semua kemungkinan *node* dihitung nilai $gain$, maka node yang memiliki nilai $gain$ yang paling tinggi nantinya yang akan dijadikan *root node*. Sesuai dengan perhitungan di atas, maka yang akan menjadi *root node* adalah cp=3 karena memiliki nilai $gain$ yang paling tinggi yaitu 2.17. Setelah pemilihan *root node* tersebut, maka *tree* sementara yang dapat dibangun dapat dilihat pada Gambar 3.17 di bawah ini.



Gambar 3. 17 Tree Sementara Yang Terbentuk

Kemudian, kita masih melakukan hal yang sama untuk perhitungan *probabillity node* yang tersisa pada *leaf* sebelah kanan yang terlihat pada Gambar 3.17 di atas. *Probabillity node* tersebut berisi tiga nilai $residual$ yang tersisa yaitu -0.5, -0.5 dan 0.5. Perhitungan *probabillity node* yang ada pada leaf tersebut dapat dilihat di bawah ini.

1. Cp=0



Gambar 3. 18 Tree Untuk Node Cp=0

Sesuai dengan ketiga nilai $residual$ yang tersisa, yang menjadi *probabillity node* pertama adalah cp=0. Sama seperti perhitungan untuk menentukan *root node* sebelumnya, kita menentukan nilai $similarity$ untuk *leaf* sebelah kiri terlebih dahulu. Perhitungan $Left_{similarity}$ dapat dilihat di bawah ini.

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

$$Left_{similarity} = \frac{0.25}{0.25}$$

$$Left_{similarity} = 1$$

Kemudian kita mencari $similarity$ dari *leaf* selanjutnya, yaitu $Right_{similarity}$ yang memiliki sisa dua nilai $residual$. Perhitungan $Right_{similarity}$ dapat dilihat di bawah ini.

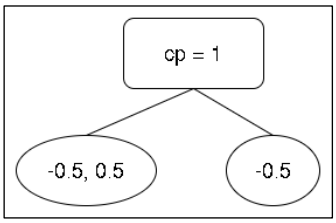
$$Right_{similarity} = \frac{(-0.5 + 0.5)^2}{\left(\left(0.5 \times (1 - 0.5)\right) \times 2\right) + 0}$$
$$Right_{similarity} = \frac{0^2}{0.25 \times 2}$$
$$Right_{similarity} = 0$$

Setelah mendapatkan $Left_{similarity}$ dan $Right_{similarity}$, selanjutnya kita menghitung nilai $gain$ dari *probabillity node* ini. $Root_{similarity}$ yang digunakan dalam perhitungan ini adalah $Right_{similarity}$ dari *node* cp=3 yang sudah menjadi *root node*, dapat dilihat pada Gambar 3.17 *tree* sementara yang dibangun di atas. Perhitungan mencari $gain$ adalah sebagai berikut.

$$gain = Left_{similarity} + Right_{similarity} - Root_{similarity}$$
$$gain = 1 + 0 - 0.33$$
$$gain = 0.67$$

Kemudian $gain$ yang sudah didapat kita simpan dahulu, dilanjutnya mencari $gain$ pada *probabillity node* selanjutnya. Nilai $gain$ dari masing-masing *probabillity node* nantinya akan dibandingkan, *probabillity node* yang memiliki $gain$ tertinggi yang akan menjadi *node*.

2. Cp=1



Gambar 3. 19 Tree Untuk Node Cp=1

Probabillity node selanjutnya adalah cp=1. Pertama kita menghitung $similarity$ dari *leaf* yang berada di kiri yang berisi dua nilai $residual$ yaitu -0,5 dan 0.5. Perhitungan $Left_{similarity}$ dapat dilihat di bawah ini.

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

Kemudian dilanjutkan dengan mencari nilai $similarity$ dari *leaf* sebelah kanan. Perhitungan nilai $Right_{similarity}$ dapat dilihat seperti di bawah ini.

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

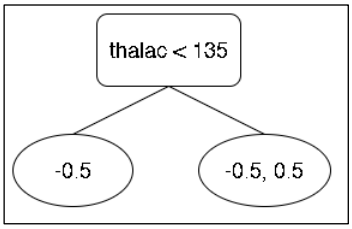
$$Right_{similarity} = \frac{0.25}{0.25}$$
$$Right_{similarity} = 1$$

Setelah mendapatkan kedua nilai $similarity$, langkah selanjutnya adalah mencari nilai $gain$ dari *probabillity node* ini dengan cara menajumlahkan $similarity$ dan dikurangi dengan $Root_{similarity}$ pada *tree* yang telah dibuat. Perhitungan $gain$ pada *probabillity node* ini dapat dilihat di bawah ini.

$$gain = Left_{similarity} + Right_{similarity} - Root_{similarity}$$
$$gain = 0 + 1 - 0.33$$
$$gain = 0.67$$

Setelah mendapatkan $gain$ untuk *probabillity node* ini, maka dilanjutkan dengan mencari nilai $gain$ untuk *probabillity node* yang tersisa. Nantinya jika semua *probabillity node* sudah diketahui nilai $gain$nya, maka $gain$ akan dibandingkan dan dicari $gain$ paling tinggi untuk dijadikan *node*.

3. Thalac<135



Gambar 3. 20 Tree Untuk Node Thalac<135

Untuk menentukan batas pada *probabillity node* ini, nilai numerik pada kolom thalac dari ketiga $residual$ tersebut kita urutkan yaitu 96, 174, dan 202. Kemudian dari masing-masing urutan kita cari nilai tengah diantara dua nilai numerik. Sehingga pada *probabbility node* ini dihitung $96 + (\frac{174-94}{2})$ dan didapatkan nilai 135 sebagai batas *probabillity node*. Gambar *probabillity node* ini dapat dilihat pada Gambar 3.20 di atas. Selanjutnya kita mencari nilai $similarity$ dari masing-masing *leaf*, diawali dari *leaf* sebelah kiri. Perhitungan $Left_{similarity}$ dapat dilihat di bawah ini.

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

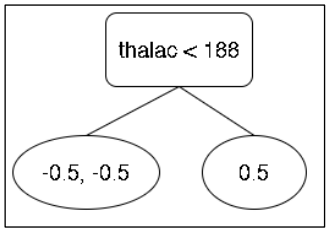
Kemudian dilanjutkan dengan mencari $similarity$ pada *leaf* sebelah kanan yang memiliki dua nilai $residual$. Perhitungan $Right_{similarity}$ dapat dilihat pada operasi matematika di bawah ini.

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

Setelah semua nilai $similarity$ diketahui, kemudian dilanjutkan mencari nilai $gain$ untuk *probabillity node* ini. Caranya adalah menjumlahkan semua nilai $similarity$ yang tadi sudah dicari dan dikurangi dengan $Root_{similarity}$ yang berisi nilai $Right_{similarity}$ pada level *root node*. Perhitungan mencari $gain$ dapat dilihat di bawah ini.

$$gain = Left_{similarity} + Right_{similarity} - Root_{similarity}$$
$$gain = 1 + 0 - 0.33$$
$$gain = 0.67$$

4. Thalac<188



Gambar 3. 21 Tree Untuk Node Thalac<188

Probabillity node terakhir pada level ini adalah *probabillity node* dengan batas 188. Batas tersebut didapat dengan menghitung nilai tengah diantara dua nilai numerik dengan cara $174 + (\frac{202-174}{2})$ dan mendapatkan nilai 188 sebagai batas. Kemudian digambarkan *tree* berdasarkan batas yang ada dengan *leaf* berupa $residual$ yang tersisa, seperti yang terlihat pada Gambar 3.21 di atas. Selanjutnya kita mencari $similarity$ untuk masing-masing *leaf*, diawali dari *leaf* sebelah kiri yang memiliki dua nilai $residual$. Perhitungan $Left_{similarity}$ dapat dilihat di bawah ini.

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

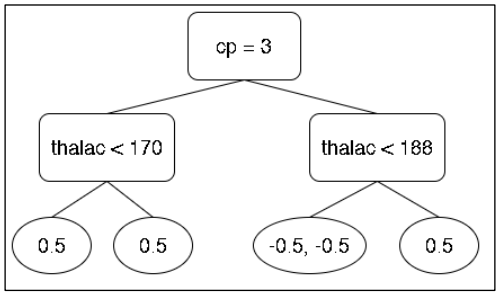
Kemudian setelah kita mendapatkan nilai $Left_{similarity}$, langkah selanjutnya adalah mencari nilai $similarity$ untuk *leaf* sebelah kanan. Perhitungan mencari $Right_{similarity}$ dapat dilihat seperti di bawah ini.

$$Right_{similarity} = \frac{0.5^2}{(0.5 \times (1 - 0.5))}$$
$$Right_{similarity} = \frac{0.25}{0.25}$$
$$Right_{similarity} = 1$$

Setelah mendapatkan semua nilai $similarity$, kita mencari nilai $gain$ untuk *probabillity node* ini. Perhitungan nilai $gain$ dicari dengan menjumlahkan semua nilai $similarity$ dan dikurangi dengan nilai $Root_{node}$ pada level ini. Perhitungan nilai $gain$ dapat dilihat di bawah ini.

$$gain = Left_{similarity} + Right_{similarity} - Root_{similarity}$$
$$gain = 2 + 1 - 0.33$$
$$gain = 2.67$$

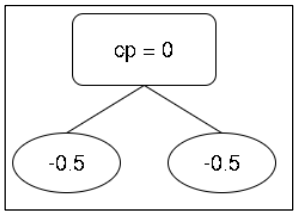
Setelah mengetahui semua nilai $gain$ pada semua *probabillity node* yang ada untuk level ini. Nilai $gain$ dari masing-masing *probabillity node* untuk level ini akan dibandingkan, dan *probabillity node* yang memiliki nilai $gain$ paling tinggi akan menjadi *node* untuk level ini. Sesuai dengan perhitungan nilai $gain$ di atas, maka *node* yang akan dipilih adalah thalac<188 karena memiliki nilai $gain$ paling tinggi diantara *probabillity node* lainnya, yaitu sebesar 2.67. Setelah pemilihan *node* untuk level ini, maka kita dapat menggambarkan *tree* sementara yang bisa dibangun. *Tree* sementara yang sudah dibangun berdasarkan perhitungan di atas dapat dilihat pada Gambar 3.22 di bawah ini.



Gambar 3. 22 Tree Sementara Yang Sudah Dibuat

Berdasarkan Gambar 3.22 di atas, terlihat bahwa node thalac<188 masih memiliki dua sisa nilai $residual$ yaitu -0.5 dan -0.5. Kita masih akan melakukan perhitungan lagi pada *probabillity node* yang ada untuk level atau kedalaman *tree* selanjutnya. Beberapa *probabillity node* yang ada untuk level ini berdasarkan data yang kita gunakan untuk proses *training* adalah cp=0, cp=1, dan thalac<135. Dari *probabillity node* tersebut nantinya kita lakukan perhitungan untuk menemukan masing-masing nilai $gain$, dan *probabillity node* yang memiliki nilai $gain$ paling tinggi akan menajdi *node*, dan dengan dipilihnya *node* tersebut *tree* pertama sudah selesai dibangun dengan tiga level kedalaman. Berikut perhitungan untuk mencari nilai $gain$ pada masing-masing *probabillity node* untuk level ini dapat dilihat di bawah ini.

1. Cp=0



Gambar 3. 23 Tree Untuk Node Cp=0

Probabillity node pertama yang ada untuk level ini adalah cp=0. Pertama kita hitung nilai $similarity$ untuk masing-masing *leaf* yang hanya memiliki sisa satu nilai $residual$. Diawali dengan perhitungan $similarity$ pada *leaf* sebelah kiri atau disebut $Left_{similarity}$. Perhitungan pada $Left_{similarity}$ dapat dilihat di bawah ini.

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

Setelah kita mendapatkan nilai $Left_{similarity}$, tahap selanjutnya adalah melakukan perhitungan yang sama untuk leaf sebelah kanan atau disebut $Right_{similarity}$. Perhitungan $Right_{similarity}$ dapat dilihat di bawah ini.

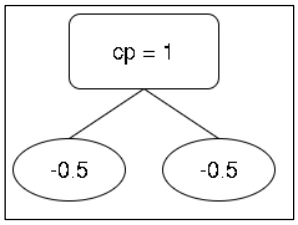
$$Right_{similarity} = \frac{(-0.5)^2}{\big(0.5 \times (1-0.5)\big) + 0}$$
$$Right_{similarity} = \frac{0.25}{0.25}$$
$$Right_{similarity} = 1$$

Setelah kita mendapatkan semua nilai $similarity$ untuk *leaf* yang ada. Selanjutnya adalah menghitung nilai $gain$ pada *probabillity node* ini. Perhitungan $gain$ pada *probabillity node* ini dilakukan dengan menjumlahkan nilai $similarity$ yang sudah kita cari, dan dikurangi dengan $Root_{node}$ yang ada untuk level kedalaman ini. $Root_{node}$ untuk level kedalaman ini adalah $Left_{similarity}$ dari node thalac<188. Perhitungan mencari nilai $gain$ untuk *probabillity node* ini dapat dilihat pada operasi matematika yang ada di bawah ini.

$$gain = Left_{similarity} + Right_{similarity} - Root_{similarity}$$
$$gain = 1 + 1 - 2$$
$$gain = 0$$

Setelah kita mengetahui nilai $gain$ pada *probabillity node* ini, maka kita lanjutkan dengan mencari nilai $gain$ pada *probabillity node* lainnya untuk nantinya dibandingkan dan dicari nilai $gain$ yang paling besar untuk dijadikan *node* untuk level kedalaman ini.

2. Cp=1



Gambar 3. 24 Tree Untuk Node Cp=1

Probabillity node selanjutnya yang akan kita lakukan perhitungan nilai $gain$ adalah *probabillity node* cp=1, sesuai dengan sisa data yang kita gunakan untuk *training*. Pertama kita lakukan perhitungan untuk mencari nilai $similarity$ pada masing-masing *leaf* yang ada. Diawali dari perhitungan *leaf* sebelah kiri atau disebut $Left_{similarity}$ yang dapat dilihat pada operasi matematika di bawah ini.

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

Kemudian setelah kita mendapatkan nilai $Left_{similarity}$, maka selanjutnya kita melakukan perhitungan yang sama untuk mencari nilai $gain$ pada *leaf* sebelah kanan atau disebut $Right_{similarity}$. Perhitungan untuk mencari $Right_{similarity}$ dapat dilihat di bawah ini.

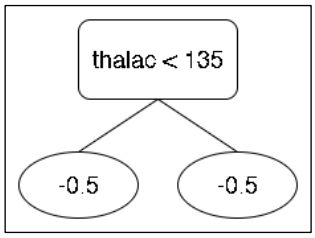
$$Right_{similarity} = \frac{(-0.5)^2}{\big(0.5 \times (1-0.5)\big) + 0}$$
$$Right_{similarity} = \frac{0.25}{0.25}$$
$$Right_{similarity} = 1$$

Setelah kita mengetahui nilai $Right_{similarity}$ dan nilai $Left_{similarity}$, maka kita jumlahkan kedua nilai $similarity$ tersebut dan dikurangi dengan nilai $Root_{nsimilarity}$ yang ada pada level kedalaman diatasnya untuk menemukan nilai $gain$ pada *probabillity node* ini. Perhitungan untuk mencari nilai $gain$ pada *probabillity node* ini dapat dilihat di bawah ini.

$$gain = Left_{similarity} + Right_{similarity} - Root_{similarity}$$
$$gain = 1 + 1 - 2$$
$$gain = 0$$

Selanjutnya kita simpan dahulu nilai $gain$ pada *probabillity node* ini, kita lanjutkan untuk mencari nilai $gain$ pada sisa *probabillity node* yang ada untuk level kedalaman ini.

3. Thalac<135



Gambar 3. 25 Tree Dengan Node Thalac<135

Perhitungan *probabillity node* terakhir adalah untuk thalac<135. Seperti perhitungan pada *probabillity node* sebelumnya, nilai batas untuk *probabillity node* ini didapatkan dengan mencari nilai tengah dari kedua nilai numerik yang ada, kemudian melakukan perhitungan $96 + (\frac{174-96}{2})$. Setelah mendapatkan nilai batas untuk *probabillity node*, maka kita dapat menggambarkan *tree* seperti yang ditampilkan pada Gambar 3.25 di atas. Kemudian kita lakukan perhitungan untuk mencari nilai $similarity$ dari kedua *leaf* yang ada. Diawali dari perhitungan $Left_{similarity}$ yang dapat dilihat di bawah ini.

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

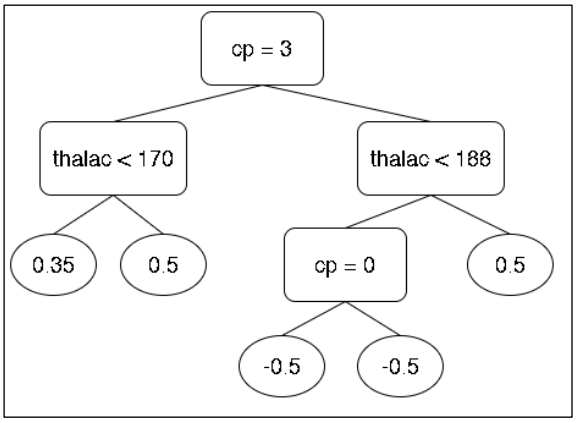
Setelah mengetahui nilai $Left_{similarity}$, maka dilanjutkan dengan mencari nilai $similarity$ pada *leaf* sebelah kanan atau disebut $Right_{similarity}$. Nilai $residual$ yang ada pada leaf sebelah kanan hanya memiliki satu nilai $residual$ yaitu -0.5. Perhitungan mencari nilai $Right_{similarity}$ dapat dilihat di bawah ini.

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

Kemudian, setelah kita mengetahui kedua nilai $similarity$ di atas, kita jumlahkan $Left_{similarity}$ dan $Right_{similarity}$ dan dikurangi dengan nilai $Root_{similarity}$ untuk mendapatkan nilai $gain$ pada *probabillity node* ini. Perhitungan untuk mencari nilai $gain$ pada *probabillity node* untuk level kedalam ini dapat dilihat pada operasi matematika di bawah ini.

$$gain = Left_{similarity} + Right_{similarity} - Root_{similarity}$$
$$gain = 1 + 1 - 2$$
$$gain = 0$$

Setelah mengetahui masing-masing nilai $gain$ pada *probabillity node* yang ada untuk level kedalaman ini, kemudian *probabillity node* dengan nilai $gain$ yang paling tinggi akan dipilih untuk menjadi *node*. Namun karena semua *probabillity node* yang sudah kita lakukan perhitungan menghasilkan nilai $gain$ yang sama, yaitu 0. Maka ketiga *probabillity node* di atas boleh dijadikan *node*, dan *node* yang dipilih untuk level kedalaman ini adalah cp=0. Setelah menetapkan *node* untuk level kedalam ini, maka kita dapat menggambar *tree* yang sudah berhasil kita bangun berdasarkan perhitungan yang kita lakukan. Untuk hasil *tree* pertama yang dibangun dapat dilihat pada Gambar 3.26 di bawah ini.



Gambar 3. 26 Hasil Tree Pertama Yang Dibuat

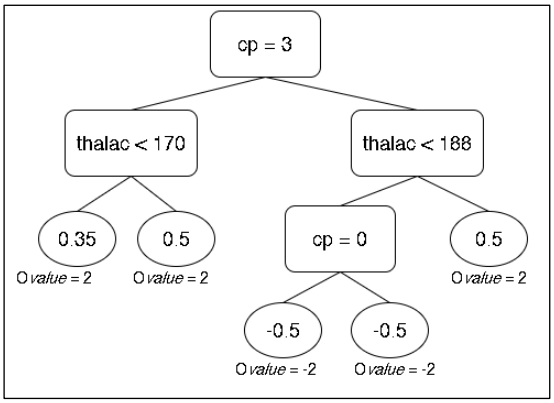
Setelah kita berhasil membangun *tree* pertama. Langkah selanjutnya adalah melakukan pengecekan pada masing-masing *leaf* dan *node* sesuai dengan *flowchart* algoritma Xgboost. Namun, dalam contoh proses *training* ini nilai $cover$ dan $gamma(\gamma)$ kita asumsikan dengan nilai 0, dikarenakan keterbasan contoh data yang digunakan dalam perhitungan manual kali ini. Sehingga tidak ada *leaf* ataupun *node* yang dipangkas. Kemudian dilanjutkan langkah berikutnya yaitu menghitung *output value* (O_{value}) untuk masing-masing *leaf* yang ada pada *tree* yang sudah dibangun. Perhitungan O_{value} pada masing-masing *leaf* dapat dilihat pada operasi matematika di bawah ini, sedangkan hasil perhitungan O_{value} dapat dilihat pada Gambar 3.27 di bawah ini setelah perhitungan. Rumus untuk menghitung O_{value} dapat dilihat pada persamaan 3.14 di bawah ini.

$$O_{value} = \frac{\sum(residual_i)}{\sum[prev\ probability_i \times (1 - prev\ probability_i)] + \lambda} \quad \text{..... (3.14)}$$

Sedangkan perhitungan O_{value} pada masing-masing leaf dapat dilihat di bawah ini.

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. 27 Tree Pertama dan O_{Value}

Setelah mengetahui nilai O_{value} dari masing-masing *leaf*. Maka kita dapat menghitung nilai $probabillity$ baru yang nantinya dapat kita gunakan untuk membuat *tree* kedua dengan menghitung nilai $residual$ yang baru dari nilai $probabillity$ yang didapat. Rumus perhitungan $probabillity$ dapat dilihat pada persamaan 3.15 dan persamaan 3.16 di bawah ini dengan nilai $learning\ rate(\varepsilon)$ adalah 0.3.

$\log(odds) = \log\left(\frac{p}{1-p}\right) + \sum[\varepsilon \times O_{valie}]_i$ ..... (3.15)

$probability = \frac{e^{\log(odds)}}{1+ e^{\log(odds)}}$ ..... (3.16)

Keterangan:

$p = 0.5\ (initial\ probability)$

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

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

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

Sedangkan untuk perhitungan nilai $probabillity$ dari setiap data yang digunakan dapat dilihat pada operasi matematika di bawah ini. Urutan pada perhitungan yang ditunjukan di bawah ini sudah disesuaikan dengan urutan data yang digunakan.

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 semua nilai $probabillity$ untuk masing-masing data, maka kita dapat menghitung nilai $residual$ yang baru. Dan nantinya kita dapat membuat *tree* lagi untuk iterasi berikutnya. Nilai $residual$ yang baru didapatkan dengan menghitung selisih dari target dan $probabillity$ yang baru saja kita hitung. Data yang memuat nilai $probabillity$ dan nilai $residual$ yang baru dapat dilihat pada Tabel 3.11 di bawah ini.

Tabel 3. 11 Data Terbaru Dan Residu

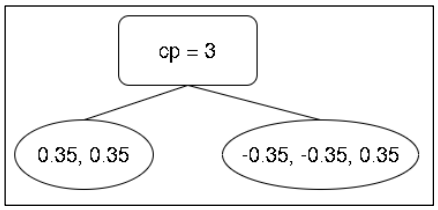
Cp	Thalac	Traget	Probability	Residual
3	150	1	0.65	0.35
3	190	1	0.65	0.35
0	96	0	0.35	-0.35
1	174	0	0.35	-0.35
1	202	1	0.65	0.35

Setelah mengetahui nilai $residual$ pada semua data, kemudian kita dapat menggunakan nilai $residual$ tersebut untuk membangun *tree* pada iterasi kedua. Proses yang dilakukan untuk membangun *tree* kedua sama seperti yang dilakukan pada *tree* pertama. Yang pertama dilakukan adalah mengasumsikan semua nilai $residual$ sebagai satu *leaf* yang akan dihitung nilai $similarity$ dan dijadikan sebagai $Root_{similarity}$. Proses perhitungan $Root_{similarity}$ pada *tree* kedua dapat dilihat di bawah ini. Menggunakan nilai $residual$ yang baru yaitu 0.35 dan -0.35.

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

Kemudian kita lanjutkan dengan melakukan perhitungan nilai $gain$ dari *probabillity node* yang ada. *Node* yang memiliki nilai $gain$ paling tinggi akan dijadikan sebagai *root node* untuk *tree* kedua. Perhitungan nilai $gain$ pada semua *probabillity node* dapat dilihat di bawah ini.

1. Cp=3



Gambar 3. 28 Tree Kedua Untuk Node Cp=3

Setelah kita menentukan *probabillity node* dan menggambar seperti yang terlihat pada Gambar 3.28 di atas. Selanjutnya adalah mulai menghitung nilai $similarity$ pada setiap *leaf* yang ada. Berdasarkan pembagian *probabillity node* tersebut, *leaf* kiri memiliki dua nilai $residual$ yaitu 0.35 dan 0.35. sedangkan *leaf* kanan memiliki tiga nilai $residual$ yaitu -0.35, -0.35, dan 0.35. Perhitungan nilai $similarity$ pada *leaf* kiri atau disebut $Left_{similarity}$ dapat dilihat seperti di bawah ini.

$$Left_{similaroty} = \frac{(0.35 + 0.35)^2}{\Big(\big(0.65 \times (1 - 0.65)\big) \times 2\Big) + 0}$$
$$Left_{similarity} = \frac{0.7^2}{0.2275 \times 2} = \frac{0.49}{0.455}$$
$$Left_{similarity} = 1.0769$$

Setelah kita mendapatkan nilai $Left_{similarity}$, selanjutnya kita lakukan perhitungan nilai $similarity$ yang sama untuk *leaf* sebelah kanan. Proses perhitungan $Right_{similarity}$ dapat dilihat pada operasi matematika di bawah ini.

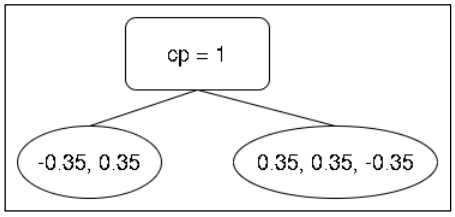
$$Right_{similarity} = \frac{(-0.35 - 0.35 + 0.35)^2}{\big(0.65 \times (1 - 0.65)\big) + \Big(\big(0.35 \times (1 - 0.35)\big) \times 2\Big) + 0}$$
$$Right_{similarity} = \frac{-0.35^2}{0.2275 + 0.455} = \frac{0.1225}{0.6825}$$
$$Right_{similarity} = 0.1794$$

Kemudian, setelah kita mendapatkan semua nilai $similarity$ pada setiap *leaf* yang ada. Kita lakukan perhitungan untuk mencari nilai $gain$ pada *probabillity node* ini dengan cara menjumlahkan nilai $similarity$ tersebut dan dikurangi dengan nilai $Root_{similarity}$ yang sudah kita cari pada perhitungan di atas. Perhitungan nilai $gain$ untuk *probabillity node* ini dapat dilihat di bawah ini.

$$gain = Left_{similarity} + Right_{similarity} - Root_{similarity}$$
$$gain = 1.0769 + 0.1794 - 0.1076$$
$$gain = 1.1487$$

Nilai $gain$ yang sudah kita dapatkan kita simpan terlebih dahulu, kemudian dilanjutkan dengan mencari nilai $gain$ pada *probabillity node* lainnya. Nantinya, setiap nilai $gain$ dari masing-masing *probabillity node* akan dibandingkan, dan *probabillity node* yang memiliki nilai $gain$ paling tinggi akan menjadi *root node*.

2. Cp=1



Gambar 3. 29 Tree Kedua Untuk Node Cp=1

Selanjutnya, *probabillity node* yang akan kita hitung adalan cp=1. Yang akan dilakukan pertama adalah menghitung nilai $similarity$ pada masing-masing *leaf* yang ada. Dari Gambar 3.29 di atas dapat dilihat bahwa *leaf* sebelah kiri memiliki dua nilai $residual$ yaitu -0.35 dan 0.35. sedangkan *leaf* sebelah kanan memiliki tiga nilai $residual$ yaitu 0.35, 0.35, dan -0.35. Kita menghitung nilai $similarity$ untuk *leaf* sebelah kiri terlebih dahulu. Perhitungan $Left_{similarity}$ dapat dilihat di bawah ini.

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

Setelah kita mendapatkan nilai $Left_{similarity}$ dari hasil perhitungan di atas, kita melakukan perhitungan yang sama lagi untuk *leaf* yang ada di sebelah kanan yang memiliki tiga nilai $residual$. Perhitungan $Right_{similarity}$ dapat dilihat pada operasi matematika di bawah ini.

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

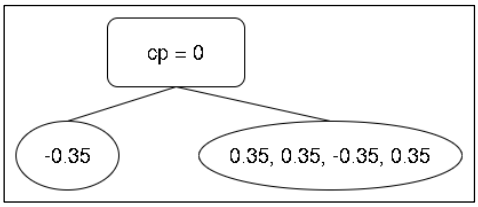
$$Right_{similarity} = \frac{0.35^2}{0.455 + 0.2275} = \frac{0.1225}{0.6825}$$
$$Right_{similarity} = 0.1794$$

Setelah mendapatkan semua nilai $similarity$ pada setiap *leaf* yang ada. Langkah selanjutnya adalah mencari nilai $gain$ untuk *probabillity node* ini. Caranya adalah dengan menjumlahkan nilai $similarity$ dan dikurangi dengan nilai $Root_{similarity}$ yang sudah kita cari pada perhitungan sebelumnya. Proses perhitungan nilai $gain$ pada *probabillity node* ini dapat dilihat di bawah ini.

$$gain = Left_{similarity} + Right_{similarity} - Root_{similarity}$$
$$gain = 0 + 0.1794 - 0.1076$$
$$gain = 0.0718$$

Setelah kita mendapatkan nilai $gain$ pada *probabillity node* ini, selanjutnya kita lakukan perhitungan lagi untuk menentukan nilai $gain$ pada *probabillity node* lainnya.

3. Cp=0



Gambar 3. 30 Tree Kedua Untuk Node Cp=0

Selanjutnya kita melakukan perhitungan untuk menentukan nilai $gain$ pada probabillity node cp=0. Perhitungan untuk mencari nilai $gain$ diawali dengan mencari nilai $similarity$ pada masing-masing leaf yang ada. Perhitungan untuk mencari nilai $Left_{similarity}$ dapat dilihat pada operasi matematika di bawah ini.

$$Left_{similarity} = \frac{(-0.35)^2}{\big(0.35 \times (1 - 0.35)\big) + 0} = \frac{0.1225}{0.2275}$$
$$Left_{similarity} = 0.5384$$

Setelah kita mendapatkan nilai $similarity$ untuk *leaf* sebelah kiri, selanjutnya kita mencari nilai $similarity$ untuk *leaf* sebelah kanan dengan perhitungan yang sama. Perhitungan untuk mencari $Right_{similarity}$ dapat dilihat di bawah ini.

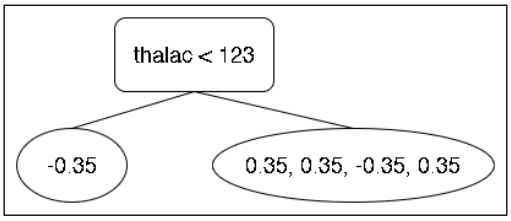
$$Right_{similarity} = \frac{(0.35 + 0.35 - 0.35 + 0.35)^2}{\Big(\big(0.65 \times (1 - 0.65)\big) \times 3\Big) + \big(0.35 \times (1 - 0.35)\big) + 0}$$
$$Right_{similarity} = \frac{(0.35 + 0.35)^2}{0.6825 + 0.2275} = \frac{0.7^2}{0.91} = \frac{0.49}{0.91}$$
$$Right_{similarity} = 0.5384$$

Setelah kita mendapatkan semua nilai $Left_{similarity}$ dan nilai $Right_{similarity}$. Langkah selanjutnya adalah mencari nilai $gain$ untuk *probabillity node* ini. Caranya adalah dengan menjumlahkan kedua nilai $Left_{similarity}$ dan nilai $Right_{similarity}$ dan dikurangi dengan nilai $Root_{similarity}$ untuk *tree* kedua ini. Perhitungan nilai $gain$ untuk *probabillity node* ini dapat dilihat di bawah ini.

$$gain = Left_{similarity} + Right_{similarity} - Root_{similarity}$$
$$gain = 0.5384 + 0.5384 - 0.1076$$
$$gain = 0.9692$$

Setelah mendapatkan nilai $gain$ dari *probabillity node* ini, kemudian kita lanjutkan dengan mencari nilai $gain$ untuk *probabillity node* lainnya hingga tidak ada lagi *probabillity node* yang tersisa.

4. Thalac<123



Gambar 3. 31 Tree Kedua Untuk Node Thalac<123

Selanjutnya kita menghitung nilai $gain$ untuk *probabillity node* yang ada di kolom thalac. Untuk mencari batas pada *probabillity node* tersebut dilakukan dengan mengurutkan nilai numerik yang ada di kolom thalac, kemudian mencari nilai tengah untuk setiap dua nilai numerik yang ada. Pada *probabillity node* ini batas dicari dengan melakukan perhitungan $96 + (\frac{150-96}{2})$ dan mendapatkan hasil 123 sebagai batas. Gambar *tree* pada *probabillity node* ini dapat dilihat pada Gambar 3.31 di atas. Kemudian kita mencari nilai $similarity$ untuk setiap leaf, diawali dari *leaf* sebelah kiri atau $Left_{similarity}$. Perhitungan $Left_{similarity}$ dapat dilihat di bawah ini.

$$Left_{similarity} = \frac{(-0.35)^2}{\big(0.35 \times (1 - 0.35)\big) + 0} = \frac{0.1225}{0.2275}$$
$$Left_{similarity} = 0.5384$$

Setelah mendapatkan nilai $Left_{similarity}$, kita lakukan perhitungan yang sama untuk mencari nilai $similarity$ pada *leaf* sebelah kanan yang memiliki empat nilai $residual$. Perhitungan untuk mencari nilai $Right_{similarity}$ dapat dilihat di bawah ini.

$$Right_{similarity} = \frac{(0.35 + 0.35 - 0.35 + 0.35)^2}{\Big(\big(0.65 \times (1 - 0.65)\big) \times 3\Big) + \big(0.35 \times (1 - 0.35)\big) + 0}$$
$$Right_{similarity} = \frac{(0.35 + 0.35)^2}{0.6825 + 0.2275} = \frac{0.7^2}{0.91} = \frac{0.49}{0.91}$$

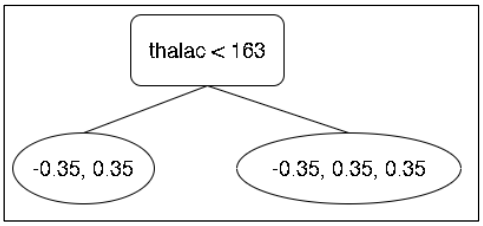
$Right_{similarity} = 0.5384$

Setelah mendapatkan nilai $Left_{similarity}$ dan nilai $Right_{similarity}$, selanjutnya kita menjumlahkan nilai tersebut untuk mendapatkan nilai $gain$ dari *probabillity node* ini. Proses perhitungan nilai $gain$ dari *probabillity node* ini dapat dilihat di bawah ini.

$gain = Left_{similarity} + Right_{similarity} - Root_{similarity}$
$gain = 0.5384 + 0.5384 - 0.1076$
$gain = 0.9692$

Nilai $gain$ dari *probabillity node* ini kita simpan terlebih dahulu dan kita lanjutnkan untuk mencari nilai $gain$ untuk *probabillity node* yang masih tersisa. Nantinya jika semua nilai $gain$ dari *probabillity node* sudah didapatkan, nilai $gain$ tersebut akan dibandingkan dan dicari nilai $gain$ yang tertinggi.

5. Thalac<163



Gambar 3. 32 Tree Kedua Untuk Node Thalac<163

Selanjutnya kita menghitung nilai $gain$ untuk *probabillity node* thalac<163. Untuk menentukan batas tersebut, dilakukan perhitungan untuk mencari nilai tengah dari dua nilai numerik dan mendapat nilai 163 untuk batasnya, perhitungan ini sama seperti menentukan batas pada *probabillity node* sebelumnya atau bahkan *tree* pertama. Kemudian dilanjutkan menghitung nilai $similarity$ dari kedua *leaf* yang ada. Dapat dilihat pada Gambar 3.32 di atas, *leaf* sebelah kiri memiliki dua nilai $residual$ berupa -0.35 dan 0.35 sedangkan *leaf* sebelah kanan memiliki tiga nilai $residual$ berupa -0.35, 0.35, dan 0.35. Pertama yang dilakukan adalah menghitung nilai $similarity$ untuk *leaf* sebelah kiri. Perhitungan $Left_{similarity}$ dapat dilihat di bawah ini.

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

Setelah kita mendapatkan nilai $Left_{similarity}$, selanjutnya kita melakukan perhitungan yang sama untuk mencari nilai $similarity$ pada *leaf* yang ada di sebelah kanan. Perhitungan $Right_{similarity}$ dapat dilihat pada operasi matematika di bawah ini.

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

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

$$Right_{similarity} = 0.1794$$

Kemudian setelah kita mendapatkan kedua nilai $Left_{similarity}$ dan nilai $Right_{similarity}$, maka kita jumlahkan nilai tersebut dan dikurangi dengan nilai $Root_{similarity}$ yang sudah kita hitung pada awal pembuatan *tree* di atas untuk mendapatkan nilai $gain$ dari *probabillity node* ini. Perhitungan untuk mencari nilai $gain$ dapat dilihat di bawah ini.

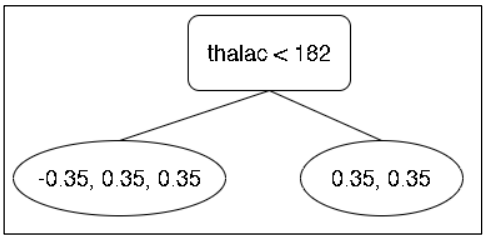
$$gain = Left_{similarity} + Right_{similarity} - Root_{similarity}$$

$$gain = 0 + 0.1794 - 0.1076$$

$$gain = 0.0718$$

Nilai $gain$ yang sudah didapatkan dari perhitungan di atas kita simpah terlebih dahulu untuk nantinya dibandingkan dengan nilai $gain$ dari *probabillity node* lainnya.

6. Thalac<182



Gambar 3. 33 Tree Kedua Untuk Node Thalac<182

Selanjutnya adalah *probabillity node* dengan batas 182. Batas tersebut didapatkan dengan melakukan perhitungan dari dua nilai numerik 174 dan 190. Dari kedua nilai numerik tersebut kita cari nilai tengahnya sehingga perhitungannya menjadi $174 + (\frac{190-174}{2})$ dan menghasilkan 182 sebagai batas. Gambar untuk *probabillity node* ini dapat dilihat pada Gambar 3.33 di atas. Kemudian kita dapat menghitung nilai $gain$ *probabillity node* ini dengan menghitung nilai $similarity$ dari kedua *leaf* terlebih dahulu. Perhitungan nilai $similarity$ untuk *leaf* sebelah kiri atau $Left_{similarity}$ dapat dilihat di bawah ini.

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

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

$$Left_{similarity} = 0.1794$$

Setelah mengetahui nilai $similarity$ untuk *leaf* sebelah kiri, maka kita lakukan perhitungan yang sama untuk mencari nilai $similarity$ pada *leaf* sebelah kanan. Perhitungan untuk mencari $Right_{similarity}$ dapat dilihat pada operasi matematika di bawah ini.

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

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

$$Right_{similarity} = 1.0769$$

Setelah kita mendapatkan semua nilai $similarity$ baik dari $Left_{similarity}$ maupun $Right_{similarity}$, kita jumlahkan nilai $similarity$ tersebut dikurangi dengan nilai $Root_{similarity}$ yang akan meghasilkan nilai $gain$ untuk *probabillity node* ini. Perhitungan untuk mencari nilai $gain$ dari *probabillity node* ini dapat dilihat pada perhitungan di bawah ini.

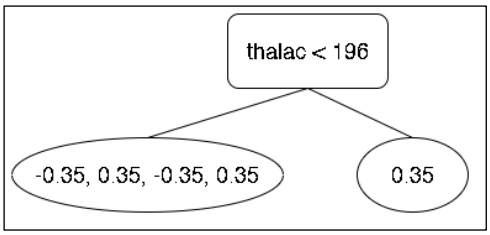
$$gain = Left_{similarity} + Right_{similarity} - Root_{similarity}$$

$$gain = 0.1794 + 1.0769 - 0.1076$$

$$gain = 1.1487$$

Nilai $gain$ milik *probabillity node* yang sudah didapat disimpan terlebuh dahulu, selanjutnya kita mencari nilai $gain$ untuk *probabillity node* lainnya.

7. Thalac<196



Gambar 3. 34 Tree Kedua Untuk Node Thalac<196

Kemudian yang terakhir adalah kita menghitung nilai $gain$ untuk *probabillity node* dengan batas 196. Batas tersebut didapatkan dengan melakukan perhitungan $190 + (\frac{202-190}{2})$ dan mendapatkan nilai 196 sebagai batas. Gambar *probabillity node* yang dibuat dapat dilihat pada Gambar 3.34 di atas. Kemudian kita dapat mencari nilai $similarity$ untuk masing-masing *leaf* yang ada. Perhitungan untuk mencari nilai $Left_{similarity}$ dapat dilihat pada operasi matematika di bawah ini.

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

$$= 0$$

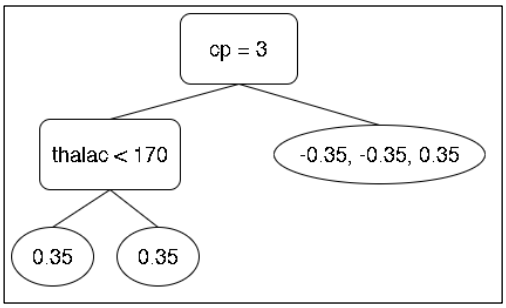
Setelah kita mendapatkan nilai $Left_{similarity}$, selanjutnya kita melakukan perhitungan yang sama untuk mencari $Right_{similarity}$. Perhitungan $Right_{similarity}$ dapat dilihat di bawah ini.

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

Setelah semua nilai $similarity$ diketahui, kita dapat menjumlahkan nilai $similarity$ tersebut dan dikurangi dengan nilai $Root_{similarity}$ untuk mendapatkan nilai $gain$ dari *probabillity node* ini. Perhitungan untuk mencari nilai $gain$ pada *probabillity node* ini dapat dilihat pada operasi matematika di bawah ini.

$$gain = Left_{similarity} + Right_{similarity} - Root_{similarity}$$
$$gain = 0 + 0.5384 - 0.1076$$
$$gain = 0.4308$$

Setelah semua nilai $gain$ kita dapatkan dari perhitungan masing-masing *probabillity node* yang ada. *Node* yang memiliki nilai $gain$ paling tinggi akan dijadikan sebagai *root node*. Sehingga, sesuai dengan perhitungan nilai $gain$ diatas, maka yang akan dijadikan *root node* adalah *node* cp=3. Sehingga setelah dipilih *probabillity node* mana yang akan menjadi *root node*, maka *tree* kedua sementara yang dibangun dapat dilihat pada gambar 3.35 di bawah ini.



Gambar 3. 35 Hasil Tree Kedua Sementara Yang Terbentuk

Kemudian, setelah kita menentukan *root node* dan menggambarkan *tree* yang berhasil dibangun seperti yang terlihat pada Gambar 3.35 di atas. Selanjutnya kita akan melakukan perhitungan yang sama lagi untuk menentukan *node* pada level kedalaman selanjutnya, pada Gambar 3.35 di atas terlihat *leaf* sebelah kanan *root node* masih memiliki sisa tiga nilai $residual$. Dari ketiga nilai $residual$ tersbeut akan dipecah untuk mendapatkan *node* yang tepat dengan melakukan perhitungan nilai $gain$ untuk setiap *probabillity node* yang ada. Perhitungan dan $Root_{similarity}$ untuk ketiga $residual$ tersebut dapat dilihat di bawah ini.

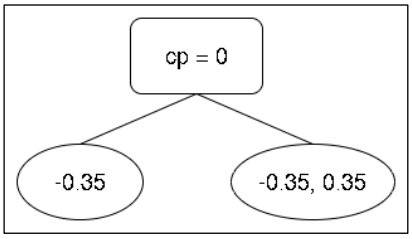
$$Root_{similarity} = \frac{(-0.35 - 0.35 + 0.35)^2}{\Big(\big(0.35 \times (1 - 0.35)\big) \times 2\Big) + \big(0.65 \times (1 - 0.65)\big) + 0}$$

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

$$Root_{similarity} = 0.1794$$

Setelah kita mengetahui nilai $Root_{similarity}$ yang dapat dilihat dari perhitungan di atas. Selanjutnya kita melakukan perhitungan untuk mencari nilai $gain$ dari setiap *probabillity node* yang ada. Masing-masing *probabillity node* dan perhitungan untuk mencari nilai $gain$ dapat dilihat di bawah ini.

1. Cp=0



Gambar 3. 36 Tree Kedua Untuk Node Cp=0

Probabillity node pertama yang ada dalam nilai $residual$ di atas adalah cp=0. Pertama yang kita lakukan adalah mencari nilai $similarity$ untuk setiap *leaf* yang ada. Pertama adalah *leaf* sebelah kiri atau disebut $Left_{similariry}$ perhitungan untuk mencari nilai $Left_{similarity}$ dapat dilihat di bawah ini.

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

$$Left_{similarity} = 0.5384$$

Setelah menentukan nilai $Left_{similarity}$ kita dapat melanjutkan perhitungan yang sama yaitu untuk mencari nilai $similarity$ pada *leaf* sebelah kanan atau disebut $Right_{similarity}$. Perhitungan $Right_{similarity}$ dapat dilihat di bawah ini.

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

$$Right_{similarity} = 0$$

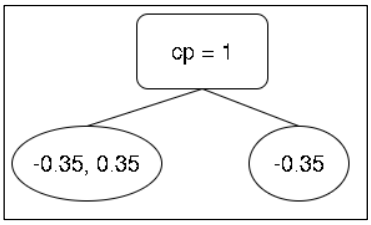
Setelah kita mendapatkan semua nilai $similarity$, kedua nilai tersebut dapat kita jumlahkan untuk mendapatkan nilai $gain$ dari *probabillity node* ini. Perhitungan nilai $gain$ untuk *probabillity node* ini dapat dilihat di bawah ini.

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

$$gain = 0.5384 + 0 - 0.1794$$

$$gain = 0.359$$

2. Cp=1



Gambar 3. 37 Tree Kedua Untuk Node Cp=1

Kemudian *probabillity node* berikutnya adalah *probabillity node* cp=1, seseuai dengan sisa nilai $residual$ yang ada di atas. Pertama adalah mencari nilai $similarity$ untuk setiap *leaf* yang ada, mulai dari *leaf* sebelah kiri atau $Left_{similarity}$ yang memiliki dua nilai $residual$ yaitu -0.35, dan 0.35 kemudian dilanjutkan *leaf* sebelah kanan atau $Right_{similarity}$ yang memiliki satu sisa nilai $residual$ yaitu -0.35. Perhitungan untuk mencari $Left_{similarity}$ dapat dilihat pada operasi matematika di bawah ini.

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

Setelah kita berhasil menghitung nilai $Similarity$ untuk *leaf* sebelah kiri, maka kita lakukan perhitungan yang sama untuk menentukan nilai $similarity$ untuk *leaf* sebelah kanan. Perhitungan untuk mencari nilai $Right_{similarity}$ dapat dilihat di bawah ini.

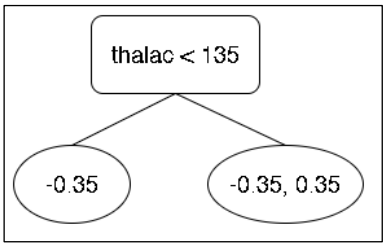
$$Right_{similarity} = \frac{-0.35^2}{\big(0.35 \times (1 - 0.35)\big) + 0} = \frac{0.1225}{0.2275}$$
$$Right_{similarity} = 0.5384$$

Kemudian setelah kita mendapatkan semua nilai $similarity$ baik dari *leaf* sebelah kiri dan *leaf* sebelah kanan, kita dapat menjumlahkan *leaf* tersebut dan mengurangi dengan nilai $Root_{similarity}$ untuk mendapatkan nilai $gain$ dari *probabillity node* ini. Perhitungan untuk mencari nilai $gain$ adalah seperti yang dapat dilihat di bawah ini.

$$gain = Left_{similarity} + Right_{similarity} - Root_{similarity}$$
$$gain = 0 + 0.5384 - 0.1794$$
$$gain = 0.359$$

Setlah semua nilai $similarity$ dan nilai $gain$ untuk probabillity node ini ditemukan, simpan ilai $gain$ untuk probabillity node ini, nilai $gain$ tersebut nantiny akan kita bandingkan dengan nilai $gain$ milik probabillity node lainnya. Selanjutnya kita lanjutkan untuk mencari nilai $gain$ untuk probabillity node lainnya yang masih tersisa.

3. Thalac<135



Gambar 3. 38 Tree Kedua Untuk Node Thalac<135

Probabillity node selanjutnya adalah thalac<135. Cara menentukan batas untuk *probabillity node* ini adalah melakukan perhitungan pada dua nilai numerik yang ada pada kolom tersebut dan mencari nilai tengahnya. Untuk mencari nilai tengahnya dilakukan perhitungan $96 + (\frac{174-96}{2})$ dan mendapatkan nilai 135 sebagai batas *probabillity node* ini. Untuk mencari nilai $gain$ untuk *probabillity node* ini diawali dengan mencari nilai $similarity$ masing-masing *leaf* yang ada. Pertama adalah mencari nilai $Left_{similarity}$ untuk *leaf* sebelah kiri. Perhitungan untuk mencari nilai $Left_{similarity}$ dapat dilihat di bawah ini.

$$Left_{similarity} = \frac{-0.35^2}{\big(0.35 \times (1 - 0.35)\big) + 0} = \frac{0.1225}{0.2275}$$
$$Left_{similarity} = 0.5384$$

Setelah kita mendapatkan nilai $similarity$ untuk *leaf* sebelah kiri, maka selanjutnya kita lakukan perhitungan yang sama untuk *leaf* yang berada di sebelah kanan atau disebut $Right_{similarity}$. Perhitungan untuk mencari nilai $Right_{similarity}$ dapat dilihat di bawah ini.

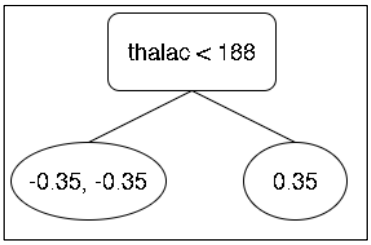
$$Right_{similarity} = \frac{(-0.35 + 0.35)^2}{\big(0.35 \times (1 - 0.35)\big) + \big(0.65 \times (1 - 0.65)\big) + 0}$$
$$Right_{similarity} = 0$$

Kemudian apabila kedua nilai $similarity$ dari kedua *leaf* sudah ditemukan, maka kita dapat menjumlahkan nilai $similarity$ tersebut dan mengurangi dengan nilai $Root_{similarity}$ untuk mendapatkan nilai $gain$ dari *probabillity node* ini. Perhitungan untuk mencari nilai $gain$ dapat dilihat di bawah ini.

$$gain = Left_{similarity} + Right_{similarity} - Root_{similarity}$$
$$gain = 0.5384 + 0 - 0.1794$$
$$gain = 0.359$$

Setelah nilai $gain$ pada *probabillity node* ini ditemukan, nilai $gain$ tersebut disimpan dan kita melanjutkan untuk mencari nilai $gain$ dari sisa *probabillity node* yang ada. Jika semua nilai $gain$ sudah ditemukan, maka kita dapat membandingkan nilai $gain$ tersebut untuk menemukan *probabillity node* yang dipilih untuk menjadi *node*.

4. Thalac<188



Gambar 3. 39 Tree Kedua Untuk Node Thalac<188

Probabillity node terakhir yang ada pada sisa nilai $residual$ di atas adalah probabillity node untuk batas thalac<188. Untuk menemukan batas probabillity node tersebut, dilakukan perhitungan untuk menemukan nilai tengah dari dua nilai numerik yang ada pada data sisa $residual$ yang ada. Untuk menemukan batas probabillity node, dilakukan perhitungan $174 + (\frac{202-174}{2})$ dan mendapatkan nilai 188 sebagai batas dari probabillity node ini. Gambar dari probabillity node ini dapat dilihat pada Gambar 3.39 di atas. Pertama yang kita lakukan adalah mencari nilai $similarity$ untuk leaf sebelah kiri. Perhitungan untuk mencari nilai $Left_{similarity}$ dapat dilihat di bawah ini.

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

$$Left_{similarity} = 1.0769$$

Setelah menemukan nilai $Left_{similarity}$, maka kita lakukan perhitungan yang sama untuk mencari nilai $similarity$ untuk *leaf* yang berada di sebelah kanan yang hanya memiliki satu nilai $residual$ berdasarkan Gambar 3.39 di atas. Perhitungan untuk mencari nilai $Right_{similarity}$ dapat dilihat di bawah ini.

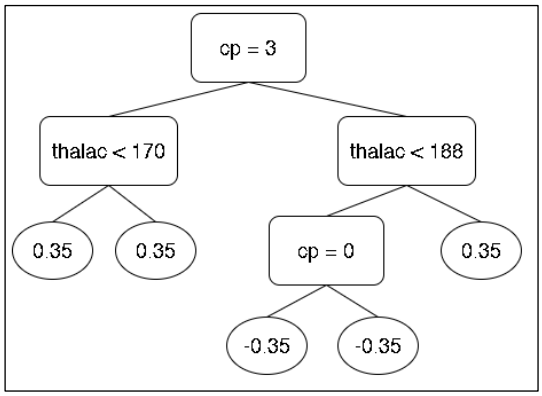
$$Right_{similarity} = \frac{0.35^2}{\big(0.65 \times (1 - 0.65)\big) + 0} = \frac{0.1225}{0.2275}$$

$$Right_{similarity} = 0.5384$$

Kemudian dapat dilanjutkan untuk mencari nilai $gain$ dari *probabillity node* ini apabila sudah mendapatkan semua nilai $similarity$ yang ada. Perhitungan untuk mencari nilai $gain$ dari *probabillity node* ini dapat dilihat di bawah ini.

$$gain = left_{similarity} + Right_{similarity} - Root_{similarity}$$
$$gain = 1.0769 + 0.5384 - 0.1794$$
$$gain = 1.4359$$

Setelah nilai $gain$ dari masing-masing *probabillity node* ditemukan, maka kita membandingkan masing-masing nilai gain, dan *probabillity node* yang memiliki nilai $gain$ paling tinggi akan menjadi *node*. Maka *node* yang memiliki $gain$ tertinggi adalah thalac<188. Sehingga hasil *tree* kedua yang dibangun dapat dilihat pada Gambar 3.40 di bawah ini.

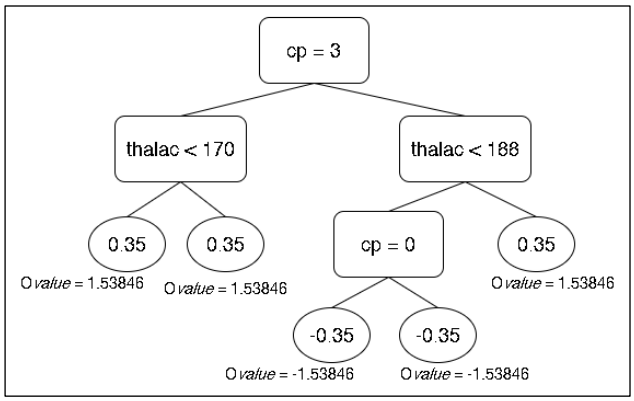


Gambar 3. 40 Hasil Tree Kedua

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* pada *tree* kedua dapat dilihat di bawah ini, sedangkan *tree* kedua yang sudah mendapatkan nilai O_{value} ditampilkan pada Gambar 3.41 di bawah ini setelah perhitungan selesai.

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

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. 41 Tree Kedua Dan O_{Value}

Setelah mendapatkan nilai O_{value} dari masing-masing *leaf* yang ada, dan Gambar tree kedua yang berhasil dibuat dapat dilihat pada Gambar 3.41 di atas. Maka kita dapat menghitung $probabillity$ untuk masing-masing data. Dari hasil $probabillity$ tersebut nantinya akan mendapatkan nilai $residual$ yang baru yang nantinya akan digunakan untuk

membuat *tree* pada iterasi berikutnya yaitu *tree* ketiga, kemudian melakukan perhitungan yang sama sesuai nilai $residual$ yang ada. Namun, pada contoh ini hanya dituliskan proses hingga *tree* kedua. Perhitungan untuk mencari $probabillity$ menggunakan kedua *tree* yang sudah dibangun dapat dilihat di bawah ini, urutan perhitungan sudah disesuaikan dengan urutan data yang digunakan.

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 $probabilllity$ maka dapat digunakan untuk mencari nilai $residual$ yang akan digunakan untuk membuat *tree* pada iterasi berikutnya. Dalam algoritma XgBoost, biasanya *tree* yang dibuat 100 *tree* atau lebih. Hasil $pribabillity$ dan nilai $residual$ dari kedua *tree* yang dibangun terdapat pada Tabel 3.12 di bawah ini.

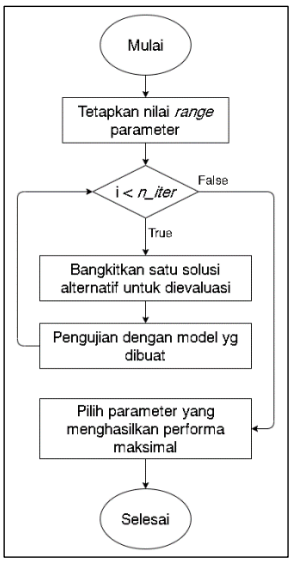
Tabel 3. 12 Data Dan Residual Tree Kedua

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 3.12 di atas, nilai $residual$ yang baru dapat digunakan untuk pembuatan *tree* yang ketiga. Dan dari beberapa iterasi pembuatan *tree* 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 yang 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* ditampilkan pada gambar 3.36 berikut.



Gambar 3. 42 Flowchart Randomized Search

3.4.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(ε) adalah 0,3. Perhitungan dan hasil prediksi dapat dilihat pada tabel 3.12 di bawah ini.

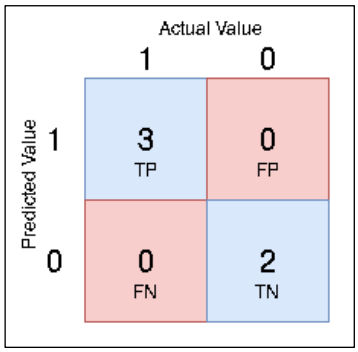
$$\log(odds) = 0 + \sum [0.3 \times O_{value}]_i$$

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

Tabel 3. 13 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 $probabilty \geq 0.5$ adalah label 1, dan $probability < 0.5$ adalah dengan label 0. Sehingga bentuk *confusion matrix* pada gambar 3.37 berikut.



Gambar 3. 43 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 *dataset* yang digunakan dalam contoh proses *training* dan menggunakan nilai *lambda*(λ), *gamma*(γ) dan *cover* adalah 0. Apabila menggunakan seluruh *dataset* maka hasil dapat berbeda. Menggunakan *randomized search* juga akan membantu menemukan parameter terbaik untuk mendapatkan model dengan performa yang baik dan tidak *overfit*.

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. 14 Hardware yang digunakan

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.

Tabel 3. 15 Software yang digunakan

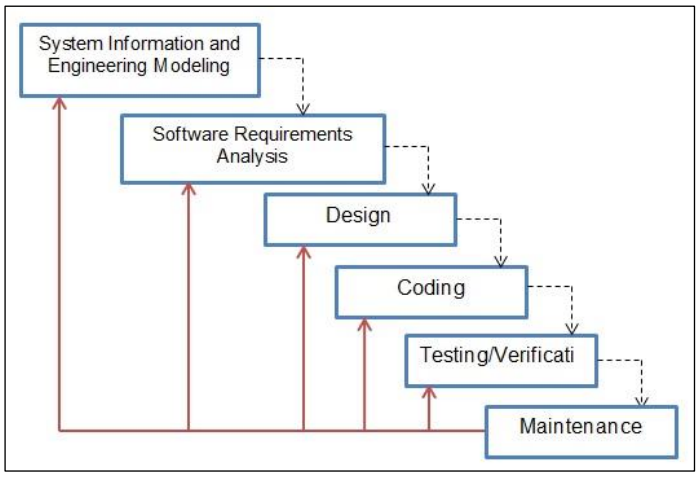
No.	Perangkat Lunak	Keterangan
1.	Operating System	Windows 10 64bit
2.	Bahasa python 3.7	Bahasa Pemrograman
3.	Jupyter Notebook	*Software* Code Editor
4.	Flask API	*Framework* untuk membuat API
5.	Draw.io	*Software* untuk membuat diagram
6.	Adobe XD	*Software* untuk mendesain UI

3.6.Proses Desain

Di dalam subbab proses desain ini berisi tahap-tahapan yang akan menjelaskan proses pembuatan perangkat lunak sederhana klasifikasi penyakit jantung dengan model klasifikasi yang dibuat menggunakan algoritma xgboost yang sudah dijelaskan pada subbab

sebelumnya. Di dalam proses desain akan dibagi menjadi beberapa subbab lagi yang akan membahas tentang perancangan sistem, perancangan proses, dan desain perancangan antarmuka perangkat lunak yang akan dibangun. Tujuan dari proses desain adalah untuk memberikan gambaran tentang perangkat lunak yang akan dibangun, mulai dari proses kerja perangkat lunak tersebut, hingga tampilan antarmuka yang akan dibangun.

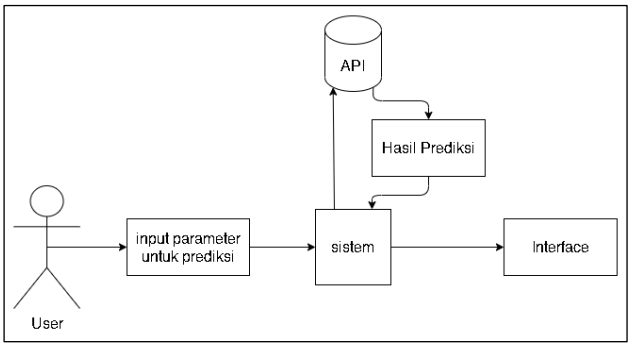
3.6.1. Perancangan Sistem



Gambar 3. 44 Model Pengembangan Waterfall

Pengembangan perangkat lunak pada penelitian ini akan menggunakan model pengembangan *waterfall*. Tahapan model pengembangan *waterfall* dapat dilihat pada gambar 3.38 di atas. Di dalam model pengembangan waterfall dibagi ke dalam beberapa tahapan diantaranya adalah.

1. System Information and Engginering Modelling
   Dalam tahap ini dilakukan pengumpulan kebutuhan sistem. Dilakukan dengan mencatat kebutuhan fungsional dan kebutuhan non fungsional.
2. Software Requirements Analysis
   Setelah mendapatkan kebutuhan sistem, pada tahap ini dilakukan analisis terhadap data-data kebutuhan yang sudah didapatkan pada tahap sebelumnya.
3. Design
   Pada tahap ini dilakukan proses desain meliputi desain dfd sistem hingga desain *interface* sistem.
4. Coding
   Setelah menyelesaikan tahapan desain dan mendapatkan gambaran desain sistem yang akan dibuat, pada tahap ini dilakukan proses pengkodean atau implementasi dari desain menjadi perangkat lunak.
5. Testing
   Pada tahap ini dilakukan pengujian terhadap perangkat lunak yang sudah dibuat.
6. Maintenance
   Tahapan terakhir adalah proses *maintenance* atau pemeliharaan perangkat lunak tersebut. Apabila ditemukan *error* maka dapat dilakukan perbaikan dan dilakukan proses *testing* kembali, dan memungkikan untuk melakukan pengembanga terhadap sistem tersebut.

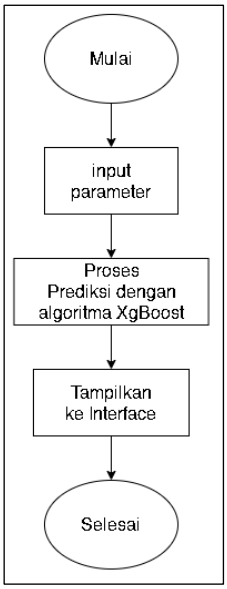


Gambar 3. 45 Arsitektur Perangkat Lunak

Gambaran umum arsitektur sistem yang akan dibuat adalah seperti gambar 3.39 di atas. Dimulai dari *user* yang melakukan input parameter berupa *features* data seperti *age*, *cp*, *thalac*, dan lain-lain. Data tersbut kemudian akan dikirimkan sistem dan diprediksi oleh model menggunakan API. Kemudian model akan menerima data yang diinput oleh *user*, dan model akan melakukan prediksi berdasarkan data tersebut. kemudian hasil prediksi akan dikirimkan ke sistem menggunakan API, dan sistem akan menampilkan hasil prediksi tersebut ke *interface* perangkat lunak.

3.6.2. Perancangan Proses

Dalam perancangan proses ini, akan dijelaskan proses-proses yang terjadi di dalam perangkat lunak yang akan dibuat. *Flowchart* proses sistem terdapat pada gambar 3.40 di bawah ini.

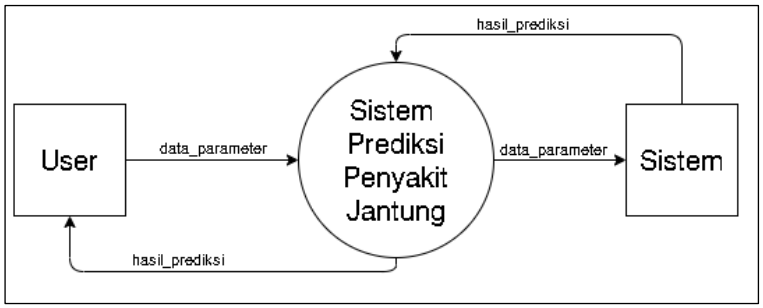


Gambar 3. 46 Flowchart Sistem

Sesuai dengan *flowchart* di atas, proses yang terjadi pada sistem adalah proses memprediksi penyakit jantung dengan algoritma XgBoost. Dimulai dari *user* yang melakukan input parameter, kemudian inputan parameter tersebut akan dibawa menuju model prediksi dengan API, yang nantinya model prediksi akan melakukan prediksi

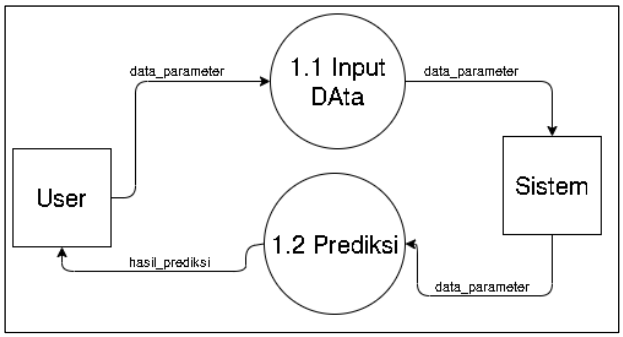
berdasarkan parameter yang diinput oleh *user*. Kemudian setelah model melakukan prediksi, nantinya hasil prediksi yang diberikan oleh model akan dibawa ke sistem melalui API dan akan ditampilkan ke *interface* perangkat lunak.

Berikut pada gambar 4.41 di bawah ini adalah DFD dari sistem yang akan dibuat. Berisi gambaran proses dan data apa saja yang akan berlangsung dalam sistem tersebut.



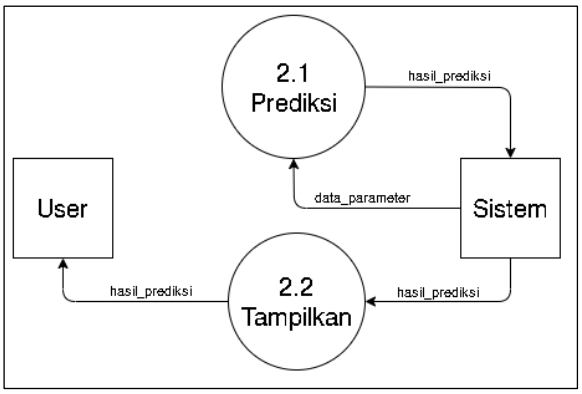
Gambar 3. 47 Dfd Level 0

Pada gambar 3.41 di atas terlihat bahwa sistem yang akan dibuat adalah sistem prediksi yang sudah siap digunakan, *model* yang digunakan untuk memprediksi sudah dibuat dan ditraining sebelum proses pembuatan sistem. Jadi, sistem yang dibuat hanya digunakan untuk memprediksi. Entitas yang ada di dalam dfd tersebut hanya satu yaitu user. User dapat melakukan input berupa data parameter yang akan diproses oleh sistem dan diprediksi oleh *model*. Kemudian hasil prediksi yang diberikan oleh *model* akan ditampilkan kepada user.



Gambar 3. 48 Dfd Level 1

Di dalam dfd level 1 yang ditampilkan pada gambar 3.42 di atas, dfd tersebut dipecah menjadi dua proses utama yaitu proses input parameter oleh *user*, dan yang kedua adalah proses prediksi yang dilakukan oleh *model*. Hasil dari proses prediksi tersebut nantinya akan ditampilkan ke *user*.

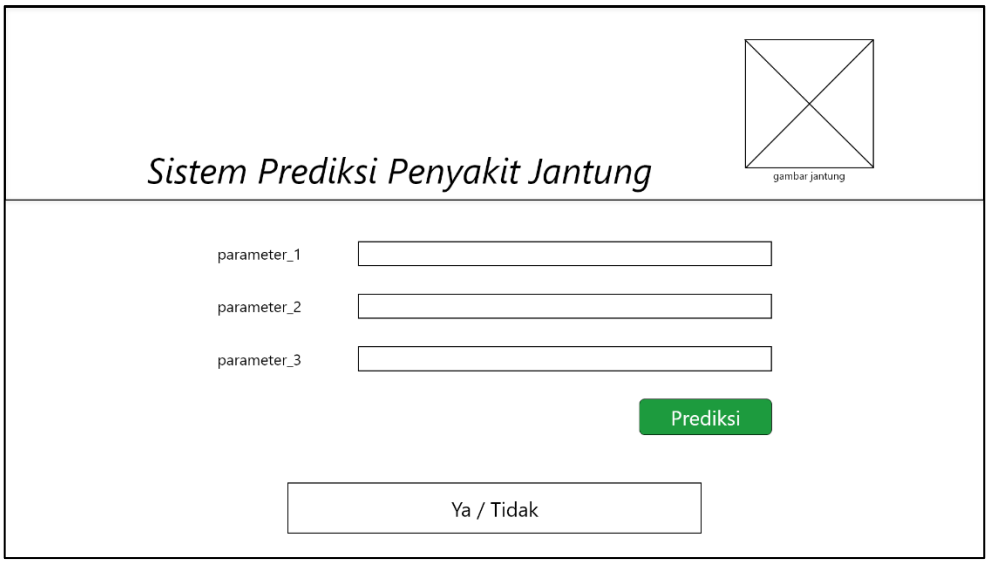


Gambar 3. 49 Dfd Level 2 Proses Identifikasi

Gambar 3.43 di atas menjelaskan dfd level 2 untuk proses identifikasi, yaitu data yang diinputkan oleh *user* akan diterima sistem, sistem nantinya akan mengirimkan data yang diinputkan tersebut ke model untuk diprediksi. Hasil prediksi yang dihasilkan akan dikirimkan melalui API dan akan ditampilkan ke user melalui *interface* sistem.

3.6.3. Perancangan Antarmuka

Tampilan antarmuka perangkat lunak atau disebut *user interface* adalah sebuah media yang digunakan untuk komunikasi antara pengguna dan sistem. Tampilan antarmuka atau *user interface* berisi fungsi-fungsi yang ada di dalam sistem yang dibuat dengan tampilam grafis yang memudahkan pengguna untuk menggunakan sistem tersebut. Dalam sistem prediksi penyakit jantung yang akan dibuat ini, nantinya akan menggunakan satu tampilan yang berisi form inputan untuk memasukkan nilai dari parameter-parameter, terdapat tombol untuk menginput parameter dan terdapat space untuk menampung hasil prediksi sesuai dengan parameter yang diinputkan. Rancangan *user interface* yang akan dibuat adalah seperti gambar 3.44 di bawah ini.



Gambar 3. 50 User Interface Sistem

DAFTAR PUSTAKA

Afianto, M. F., Faraby, S. Al, & Adiwijaya. (2017). *Kategorisasi Teks pada Hadits Sahih Al-Bukhari menggunakan Random Forest*. *4*(3), 4874–4881.

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

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

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.
Kusuma, P. P. B., & Srinandi, I. G. A. M. (2013). Prediksi Waktu Ketahanan Hidup Dengan Metode Partial Least Square. *E-Jurnal Matematika*, *2*(1), 49. https://doi.org/10.24843/mtk.2013.v02.i01.p028
Lestari, M. (2014). Penerapan Algoritma Klasifikasi Nearest Neighbor (K-NN) untuk Mendeteksi Penyakit Jantung. *Faktor Exacta*, *7*(September 2010), 366–371.
Lubis, C., & Gondawijaya, F. (2019). Heart Sound Diagnose System with BFCC, MFCC, and Backpropagation Neural Network. *IOP Conference Series: Materials Science and Engineering*, *508*(1). https://doi.org/10.1088/1757-899X/508/1/012119
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.
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.
Rahayu, E. S., Satria, R., & Supriyanto, C. (2015). Penerapan Metode Average Gain,

Threshold Pruning dan Cost Complexity Pruning Untuk Split Atribut Pada Algoritma C4.5. *Journal of Intelligent Systems*, *1*(2), 91–97.

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.

Septadaya, A., Dewi, C., & Rahayudi, B. (2019). Implementasi Extreme Learning Machine dan Fast Independent Component Analysis untuk Klasifikasi Aritmia Berdasarkan Rekaman Elektrokardiogram. *Jurnal Pengembangan Teknologi Informasi Dan Ilmu Komputer E-ISSN*, *2548*(5), 964X.

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.

Zhang, D., Qian, L., Mao, B., Huang, C., Huang, B., & Si, Y. (2018). A Data-Driven Design for Fault Detection of Wind Turbines Using Random Forests and XGboost. *IEEE Access*, *6*, 21020–21031. https://doi.org/10.1109/ACCESS.2018.2818678

LAMPIRAN

Lampiran 1. Tabel Dataset

age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
57	0	0	120	354	0	1	163	1	0.6	2	0	2	1
57	1	0	140	192	0	1	148	0	0.4	1	0	1	1
56	0	1	140	294	0	0	153	0	1.3	1	0	2	1
44	1	1	120	263	0	1	173	0	0	2	0	3	1
52	1	2	172	199	1	1	162	0	0.5	2	0	3	1
57	1	2	150	168	0	1	174	0	1.6	2	0	2	1
54	1	0	140	239	0	1	160	0	1.2	2	0	2	1
48	0	2	130	275	0	1	139	0	0.2	2	0	2	1
49	1	1	130	266	0	1	171	0	0.6	2	0	2	1
64	1	3	110	211	0	0	144	1	1.8	1	0	2	1
58	0	3	150	283	1	0	162	0	1	2	0	2	1
50	0	2	120	219	0	1	158	0	1.6	1	0	2	1
58	0	2	120	340	0	1	172	0	0	2	0	2	1
66	0	3	150	226	0	1	114	0	2.6	0	0	2	1
43	1	0	150	247	0	1	171	0	1.5	2	0	2	1
69	0	3	140	239	0	1	151	0	1.8	2	2	2	1
59	1	0	135	234	0	1	161	0	0.5	1	0	3	1
44	1	2	130	233	0	1	179	1	0.4	2	0	2	1
42	1	0	140	226	0	1	178	0	0	2	0	2	1
61	1	2	150	243	1	1	137	1	1	1	0	2	1
40	1	3	140	199	0	1	178	1	1.4	2	0	3	1
71	0	1	160	302	0	1	162	0	0.4	2	2	2	1
59	1	2	150	212	1	1	157	0	1.6	2	0	2	1
51	1	2	110	175	0	1	123	0	0.6	2	0	2	1
65	0	2	140	417	1	0	157	0	0.8	2	1	2	1
53	1	2	130	197	1	0	152	0	1.2	0	0	2	1
41	0	1	105	198	0	1	168	0	0	2	1	2	1
65	1	0	120	177	0	1	140	0	0.4	2	0	3	1
44	1	1	130	219	0	0	188	0	0	2	0	2	1
54	1	2	125	273	0	0	152	0	0.5	0	1	2	1
51	1	3	125	213	0	0	125	1	1.4	2	1	2	1
46	0	2	142	177	0	0	160	1	1.4	0	0	2	1
54	0	2	135	304	1	1	170	0	0	2	0	2	1
54	1	2	150	232	0	0	165	0	1.6	2	0	3	1
65	0	2	155	269	0	1	148	0	0.8	2	0	2	1
65	0	2	160	360	0	0	151	0	0.8	2	0	2	1

51	0	2	140	308	0	0	142	0	1.5	2	1	2	1
48	1	1	130	245	0	0	180	0	0.2	1	0	2	1
45	1	0	104	208	0	0	148	1	3	1	0	2	1
53	0	0	130	264	0	0	143	0	0.4	1	0	2	1
39	1	2	140	321	0	0	182	0	0	2	0	2	1
52	1	1	120	325	0	1	172	0	0.2	2	0	2	1
44	1	2	140	235	0	0	180	0	0	2	0	2	1
47	1	2	138	257	0	0	156	0	0	2	0	2	1
53	0	2	128	216	0	0	115	0	0	2	0	0	1
53	0	0	138	234	0	0	160	0	0	2	0	2	1
51	0	2	130	256	0	0	149	0	0.5	2	0	2	1
66	1	0	120	302	0	0	151	0	0.4	1	0	2	1
62	1	2	130	231	0	1	146	0	1.8	1	3	3	1
44	0	2	108	141	0	1	175	0	0.6	1	0	2	1
63	0	2	135	252	0	0	172	0	0	2	0	2	1
52	1	1	134	201	0	1	158	0	0.8	2	1	2	1
48	1	0	122	222	0	0	186	0	0	2	0	2	1
45	1	0	115	260	0	0	185	0	0	2	0	2	1
34	1	3	118	182	0	0	174	0	0	2	0	2	1
57	0	0	128	303	0	0	159	0	0	2	1	2	1
71	0	2	110	265	1	0	130	0	0	2	1	2	1
54	1	1	108	309	0	1	156	0	0	2	0	3	1
52	1	3	118	186	0	0	190	0	0	1	0	1	1
41	1	1	135	203	0	1	132	0	0	1	0	1	1
58	1	2	140	211	1	0	165	0	0	2	0	2	1
35	0	0	138	183	0	1	182	0	1.4	2	0	2	1
51	1	2	100	222	0	1	143	1	1.2	1	0	2	1
45	0	1	130	234	0	0	175	0	0.6	1	0	2	1
44	1	1	120	220	0	1	170	0	0	2	0	2	1
62	0	0	124	209	0	1	163	0	0	2	0	2	1
54	1	2	120	258	0	0	147	0	0.4	1	0	3	1
51	1	2	94	227	0	1	154	1	0	2	1	3	1
29	1	1	130	204	0	0	202	0	0	2	0	2	1
51	1	0	140	261	0	0	186	1	0	2	0	2	1
43	0	2	122	213	0	1	165	0	0.2	1	0	2	1
55	0	1	135	250	0	0	161	0	1.4	1	0	2	1
51	1	2	125	245	1	0	166	0	2.4	1	0	2	1
59	1	1	140	221	0	1	164	1	0	2	0	2	1
52	1	1	128	205	1	1	184	0	0	2	0	2	1
58	1	2	105	240	0	0	154	1	0.6	1	0	3	1
41	1	2	112	250	0	1	179	0	0	2	0	2	1
45	1	1	128	308	0	0	170	0	0	2	0	2	1
60	0	2	102	318	0	1	160	0	0	2	1	2	1
52	1	3	152	298	1	1	178	0	1.2	1	0	3	1

42	0	0	102	265	0	0	122	0	0.6	1	0	2	1
67	0	2	115	564	0	0	160	0	1.6	1	0	3	1
68	1	2	118	277	0	1	151	0	1	2	1	3	1
46	1	1	101	197	1	1	156	0	0	2	0	3	1
54	0	2	110	214	0	1	158	0	1.6	1	0	2	1
58	0	0	100	248	0	0	122	0	1	1	0	2	1
48	1	2	124	255	1	1	175	0	0	2	2	2	1
57	1	0	132	207	0	1	168	1	0	2	0	3	1
52	1	2	138	223	0	1	169	0	0	2	4	2	1
54	0	1	132	288	1	0	159	1	0	2	1	2	1
45	0	1	112	160	0	1	138	0	0	1	0	2	1
53	1	0	142	226	0	0	111	1	0	2	0	3	1
62	0	0	140	394	0	0	157	0	1.2	1	0	2	1
52	1	0	108	233	1	1	147	0	0.1	2	3	3	1
43	1	2	130	315	0	1	162	0	1.9	2	1	2	1
53	1	2	130	246	1	0	173	0	0	2	3	2	1
42	1	3	148	244	0	0	178	0	0.8	2	2	2	1
59	1	3	178	270	0	0	145	0	4.2	0	0	3	1
63	0	1	140	195	0	1	179	0	0	2	2	2	1
42	1	2	120	240	1	1	194	0	0.8	0	0	3	1
50	1	2	129	196	0	1	163	0	0	2	0	2	1
68	0	2	120	211	0	0	115	0	1.5	1	0	2	1
69	1	3	160	234	1	0	131	0	0.1	1	1	2	1
45	0	0	138	236	0	0	152	1	0.2	1	0	2	1
50	0	1	120	244	0	1	162	0	1.1	2	0	2	1
50	0	0	110	254	0	0	159	0	0	2	0	2	1
64	0	0	180	325	0	1	154	1	0	2	0	2	1
57	1	2	150	126	1	1	173	0	0.2	2	1	3	1
64	0	2	140	313	0	1	133	0	0.2	2	0	3	1
43	1	0	110	211	0	1	161	0	0	2	0	3	1
55	1	1	130	262	0	1	155	0	0	2	0	2	1
37	0	2	120	215	0	1	170	0	0	2	0	2	1
41	1	2	130	214	0	0	168	0	2	1	0	2	1
56	1	3	120	193	0	0	162	0	1.9	1	0	3	1
46	0	1	105	204	0	1	172	0	0	2	0	2	1
46	0	0	138	243	0	0	152	1	0	1	0	2	1
64	0	0	130	303	0	1	122	0	2	1	2	2	1
59	1	0	138	271	0	0	182	0	0	2	0	2	1
41	0	2	112	268	0	0	172	1	0	2	0	2	1
54	0	2	108	267	0	0	167	0	0	2	0	2	1
39	0	2	94	199	0	1	179	0	0	2	0	2	1
34	0	1	118	210	0	1	192	0	0.7	2	0	2	1
47	1	0	112	204	0	1	143	0	0.1	2	0	2	1
67	0	2	152	277	0	1	172	0	0	2	1	2	1

52	0	2	136	196	0	0	169	0	0.1	1	0	2	1
74	0	1	120	269	0	0	121	1	0.2	2	1	2	1
54	0	2	160	201	0	1	163	0	0	2	1	2	1
49	0	1	134	271	0	1	162	0	0	1	0	2	1
42	1	1	120	295	0	1	162	0	0	2	0	2	1
41	1	1	110	235	0	1	153	0	0	2	0	2	1
41	0	1	126	306	0	1	163	0	0	2	0	2	1
49	0	0	130	269	0	1	163	0	0	2	0	2	1
60	0	2	120	178	1	1	96	0	0	2	0	2	1
62	1	1	128	208	1	0	140	0	0	2	0	2	1
57	1	0	110	201	0	1	126	1	1.5	1	0	1	1
64	1	0	128	263	0	1	105	1	0.2	1	1	3	1
51	0	2	120	295	0	0	157	0	0.6	2	0	2	1
43	1	0	115	303	0	1	181	0	1.2	1	0	2	1
42	0	2	120	209	0	1	173	0	0	1	0	2	1
67	0	0	106	223	0	1	142	0	0.3	2	2	2	1
76	0	2	140	197	0	2	116	0	1.1	1	0	2	1
70	1	1	156	245	0	0	143	0	0	2	0	2	1
44	0	2	118	242	0	1	149	0	0.3	1	1	2	1
60	0	3	150	240	0	1	171	0	0.9	2	0	2	1
44	1	2	120	226	0	1	169	0	0	2	0	2	1
42	1	2	130	180	0	1	150	0	0	2	0	2	1
66	1	0	160	228	0	0	138	0	2.3	2	0	1	1
71	0	0	112	149	0	1	125	0	1.6	1	0	2	1
64	1	3	170	227	0	0	155	0	0.6	1	0	3	1
66	0	2	146	278	0	0	152	0	0	1	1	2	1
39	0	2	138	220	0	1	152	0	0	1	0	2	1
58	0	0	130	197	0	1	131	0	0.6	1	0	2	1
47	1	2	130	253	0	1	179	0	0	2	0	2	1
35	1	1	122	192	0	1	174	0	0	2	0	2	1
58	1	1	125	220	0	1	144	0	0.4	1	4	3	1
56	1	1	130	221	0	0	163	0	0	2	0	3	1
56	1	1	120	240	0	1	169	0	0	0	0	2	1
55	0	1	132	342	0	1	166	0	1.2	2	0	2	1
41	1	1	120	157	0	1	182	0	0	2	0	2	1
38	1	2	138	175	0	1	173	0	0	2	4	2	1
38	1	2	138	175	0	1	173	0	0	2	4	2	1
67	1	0	160	286	0	0	108	1	1.5	1	3	2	0
67	1	0	120	229	0	0	129	1	2.6	1	2	3	0
62	0	0	140	268	0	0	160	0	3.6	0	2	2	0
63	1	0	130	254	0	0	147	0	1.4	1	1	3	0
53	1	0	140	203	1	0	155	1	3.1	0	0	3	0
56	1	2	130	256	1	0	142	1	0.6	1	1	1	0
48	1	1	110	229	0	1	168	0	1	0	0	3	0

58	1	1	120	284	0	0	160	0	1.8	1	0	2	0
58	1	2	132	224	0	0	173	0	3.2	2	2	3	0
60	1	0	130	206	0	0	132	1	2.4	1	2	3	0
40	1	0	110	167	0	0	114	1	2	1	0	3	0
60	1	0	117	230	1	1	160	1	1.4	2	2	3	0
64	1	2	140	335	0	1	158	0	0	2	0	2	0
43	1	0	120	177	0	0	120	1	2.5	1	0	3	0
57	1	0	150	276	0	0	112	1	0.6	1	1	1	0
55	1	0	132	353	0	1	132	1	1.2	1	1	3	0
65	0	0	150	225	0	0	114	0	1	1	3	3	0
61	0	0	130	330	0	0	169	0	0	2	0	2	0
58	1	2	112	230	0	0	165	0	2.5	1	1	3	0
50	1	0	150	243	0	0	128	0	2.6	1	0	3	0
44	1	0	112	290	0	0	153	0	0	2	1	2	0
60	1	0	130	253	0	1	144	1	1.4	2	1	3	0
54	1	0	124	266	0	0	109	1	2.2	1	1	3	0
50	1	2	140	233	0	1	163	0	0.6	1	1	3	0
41	1	0	110	172	0	0	158	0	0	2	0	3	0
51	0	0	130	305	0	1	142	1	1.2	1	0	3	0
58	1	0	128	216	0	0	131	1	2.2	1	3	3	0
54	1	0	120	188	0	1	113	0	1.4	1	1	3	0
60	1	0	145	282	0	0	142	1	2.8	1	2	3	0
60	1	2	140	185	0	0	155	0	3	1	0	2	0
59	1	0	170	326	0	0	140	1	3.4	0	0	3	0
46	1	2	150	231	0	1	147	0	3.6	1	0	2	0
67	1	0	125	254	1	1	163	0	0.2	1	2	3	0
62	1	0	120	267	0	1	99	1	1.8	1	2	3	0
65	1	0	110	248	0	0	158	0	0.6	2	2	1	0
44	1	0	110	197	0	0	177	0	0	2	1	2	0
60	1	0	125	258	0	0	141	1	2.8	1	1	3	0
58	1	0	150	270	0	0	111	1	0.8	2	0	3	0
68	1	2	180	274	1	0	150	1	1.6	1	0	3	0
62	0	0	160	164	0	0	145	0	6.2	0	3	3	0
52	1	0	128	255	0	1	161	1	0	2	1	3	0
59	1	0	110	239	0	0	142	1	1.2	1	1	3	0
60	0	0	150	258	0	0	157	0	2.6	1	2	3	0
49	1	2	120	188	0	1	139	0	2	1	3	3	0
59	1	0	140	177	0	1	162	1	0	2	1	3	0
57	1	2	128	229	0	0	150	0	0.4	1	1	3	0
61	1	0	120	260	0	1	140	1	3.6	1	1	3	0
39	1	0	118	219	0	1	140	0	1.2	1	0	3	0
61	0	0	145	307	0	0	146	1	1	1	0	3	0
56	1	0	125	249	1	0	144	1	1.2	1	1	2	0
43	0	0	132	341	1	0	136	1	3	1	0	3	0

62	0	2	130	263	0	1	97	0	1.2	1	1	3	0
63	1	0	130	330	1	0	132	1	1.8	2	3	3	0
65	1	0	135	254	0	0	127	0	2.8	1	1	3	0
48	1	0	130	256	1	0	150	1	0	2	2	3	0
63	0	0	150	407	0	0	154	0	4	1	3	3	0
55	1	0	140	217	0	1	111	1	5.6	0	0	3	0
65	1	3	138	282	1	0	174	0	1.4	1	1	2	0
56	0	0	200	288	1	0	133	1	4	0	2	3	0
54	1	0	110	239	0	1	126	1	2.8	1	1	3	0
70	1	0	145	174	0	1	125	1	2.6	0	0	3	0
62	1	1	120	281	0	0	103	0	1.4	1	1	3	0
35	1	0	120	198	0	1	130	1	1.6	1	0	3	0
59	1	3	170	288	0	0	159	0	0.2	1	0	3	0
64	1	2	125	309	0	1	131	1	1.8	1	0	3	0
47	1	2	108	243	0	1	152	0	0	2	0	2	0
57	1	0	165	289	1	0	124	0	1	1	3	3	0
55	1	0	160	289	0	0	145	1	0.8	1	1	3	0
64	1	0	120	246	0	0	96	1	2.2	0	1	2	0
70	1	0	130	322	0	0	109	0	2.4	1	3	2	0
51	1	0	140	299	0	1	173	1	1.6	2	0	3	0
58	1	0	125	300	0	0	171	0	0	2	2	3	0
60	1	0	140	293	0	0	170	0	1.2	1	2	3	0
77	1	0	125	304	0	0	162	1	0	2	3	2	0
35	1	0	126	282	0	0	156	1	0	2	0	3	0
70	1	2	160	269	0	1	112	1	2.9	1	1	3	0
59	0	0	174	249	0	1	143	1	0	1	0	2	0
64	1	0	145	212	0	0	132	0	2	1	2	1	0
57	1	0	152	274	0	1	88	1	1.2	1	1	3	0
56	1	0	132	184	0	0	105	1	2.1	1	1	1	0
48	1	0	124	274	0	0	166	0	0.5	1	0	3	0
56	0	0	134	409	0	0	150	1	1.9	1	2	3	0
66	1	1	160	246	0	1	120	1	0	1	3	1	0
54	1	1	192	283	0	0	195	0	0	2	1	3	0
69	1	2	140	254	0	0	146	0	2	1	3	3	0
51	1	0	140	298	0	1	122	1	4.2	1	3	3	0
43	1	0	132	247	1	0	143	1	0.1	1	4	3	0
62	0	0	138	294	1	1	106	0	1.9	1	3	2	0
67	1	0	100	299	0	0	125	1	0.9	1	2	2	0
59	1	3	160	273	0	0	125	0	0	2	0	2	0
45	1	0	142	309	0	0	147	1	0	1	3	3	0
58	1	0	128	259	0	0	130	1	3	1	2	3	0
50	1	0	144	200	0	0	126	1	0.9	1	0	3	0
62	0	0	150	244	0	1	154	1	1.4	1	0	2	0
38	1	3	120	231	0	1	182	1	3.8	1	0	3	0

66	0	0	178	228	1	1	165	1	1	1	2	3	0
52	1	0	112	230	0	1	160	0	0	2	1	2	0
53	1	0	123	282	0	1	95	1	2	1	2	3	0
63	0	0	108	269	0	1	169	1	1.8	1	2	2	0
54	1	0	110	206	0	0	108	1	0	1	1	2	0
66	1	0	112	212	0	0	132	1	0.1	2	1	2	0
55	0	0	180	327	0	2	117	1	3.4	1	0	2	0
49	1	2	118	149	0	0	126	0	0.8	2	3	2	0
54	1	0	122	286	0	0	116	1	3.2	1	2	2	0
56	1	0	130	283	1	0	103	1	1.6	0	0	3	0
46	1	0	120	249	0	0	144	0	0.8	2	0	3	0
61	1	3	134	234	0	1	145	0	2.6	1	2	2	0
67	1	0	120	237	0	1	71	0	1	1	0	2	0
58	1	0	100	234	0	1	156	0	0.1	2	1	3	0
47	1	0	110	275	0	0	118	1	1	1	1	2	0
52	1	0	125	212	0	1	168	0	1	2	2	3	0
58	1	0	146	218	0	1	105	0	2	1	1	3	0
57	1	1	124	261	0	1	141	0	0.3	2	0	3	0
58	0	1	136	319	1	0	152	0	0	2	2	2	0
61	1	0	138	166	0	0	125	1	3.6	1	1	2	0
42	1	0	136	315	0	1	125	1	1.8	1	0	1	0
52	1	0	128	204	1	1	156	1	1	1	0	0	0
59	1	2	126	218	1	1	134	0	2.2	1	1	1	0
40	1	0	152	223	0	1	181	0	0	2	0	3	0
61	1	0	140	207	0	0	138	1	1.9	2	1	3	0
46	1	0	140	311	0	1	120	1	1.8	1	2	3	0
59	1	3	134	204	0	1	162	0	0.8	2	2	2	0
57	1	1	154	232	0	0	164	0	0	2	1	2	0
57	1	0	110	335	0	1	143	1	3	1	1	3	0
55	0	0	128	205	0	2	130	1	2	1	1	3	0
61	1	0	148	203	0	1	161	0	0	2	1	3	0
58	1	0	114	318	0	2	140	0	4.4	0	3	1	0
58	0	0	170	225	1	0	146	1	2.8	1	2	1	0
67	1	2	152	212	0	0	150	0	0.8	1	0	3	0
44	1	0	120	169	0	1	144	1	2.8	0	0	1	0
63	1	0	140	187	0	0	144	1	4	2	2	3	0
63	0	0	124	197	0	1	136	1	0	1	0	2	0
59	1	0	164	176	1	0	90	0	1	1	2	1	0
57	0	0	140	241	0	1	123	1	0.2	1	0	3	0
45	1	3	110	264	0	1	132	0	1.2	1	0	3	0
68	1	0	144	193	1	1	141	0	3.4	1	2	3	0
57	1	0	130	131	0	1	115	1	1.2	1	1	3	0
57	0	1	130	236	0	0	174	0	0	1	1	2	0