UNIVERSITAS LOGISTIK & BISNIS INTERNASIONAL

KOMPARASI PERFORMA MODEL TERHADAP KLASIFIKASI SINYAL MIT-BIH ARRHYTHMIA DATABASE

M. RIZKY
RONI ANDARSYAH

KOMPARASI PERFORMA MODEL TERHADAP KLASIFIKASI SINYAL MIT-BIH ARRHYTHMIA DATABASE

M. RIZKY
Roni Andarsyah

bukupedia

KOMPARASI PERFORMA MODEL TERHADAP KLASIFIKASI SINYAL MIT-BIH ARRHYTHMIA DATABASE

Penulis :
M. RIZKY
Roni Andarsyah

ISBN : -

Editor :
M. Yusril Helmi Setyawan

Penyunting :
M. Yusril Helmi Setyawan

Desain sampul dan Tata letak :
M. RIZKY

Penerbit :
Penerbit Buku Pedia

Redaksi :
Athena Residence Blok. E No. 1, Desa Ciwaruga,
Kec. Parongpong, Kab. Bandung Barat 40559
Tel. 628-775-2000-300
Email : penerbit@bukupedia.co.id

Distributor :
Informatics Research Center
Jl. Sariasih No. 54
Bandung 40151
Email : irc@poltekpos.ac.id

Cetakan Pertama, 2023

PRAKATA

Machine Learning merupakan teknologi yang berkembang sangat pesat saat ini. Banyak sekali produk-produk digital yang memanfaatkan teknologi Machine Learning mulai dari bidang transportasi, kedokteran, dan lainnya. Seiring banyaknya yang menggunakan Machine Learning, hampir semua produk yang di develop sekarang sudah mengimplementasikan Machine Learning.

Pemanfaatan Machine Learning di bidang kesehatan memang sangat perlu dilakukan terlebih lagi pentingnya penanganan dini kepada pasien yang mengidap penyakit khususnya pada penyakit jantung. Dengan melakukan pendeteksian dini pada suatu penyakit dapat meminimalisir berkembangnya penyakit tersebut.

Buku ini berisi bertujuan sebagai pembelajaran tentang bagaimana melakukan ekstraksi terhadap sinyal EKG dan melakukan klasifikasi pada sinyal tersebut berdasarkan *class*-nya. Buku ini diterbitkan bukan hanya karena penulis yang berperan, tetapi ada banyak pihak yang membantu. Oleh karena itu, penulis ingin mengucapkan terima kasih kepada pihak-pihak yang telah memberikan kontribusi besar dalam penyusunan buku ini.

Link Github: https://github.com/rizkydoang/Komparasi-Klasifikasi-Sinyal-ECG

DAFTAR ISI

DAFTAR GAMBAR

BAB 1
PENDAHULUAN

Dalam buku panduan ini, akan membahas tentang proses bagaimana melakukan ekstraksi sinyal EKG pada sebuah dataset yang akan dikombinasikan dengan melakukan klasifikasi pada sebuah detak jantung atau *beat* dengan menggunakan sebuah model Machine Learning.

Tidak bisa dipungkiri lagi bahwa perkembangan Artificial Intelligence begitu sangat cepat. Seiring berkembangnya teknologi yang sangat cepat, banyak sekali jenis - jenis teknologi yang bermunculan untuk membantu bahkan menggantikan pekerjaan manusia salah satunya di bidang kesehatan.Salah satu cabang Artificial Intelligence yang sekarang banyak sekali diminati adalah Machine Learning dan sampai saat ini masih terus berkembang pesat di kalangan programmer atau khususnya di dunia IT. Machine Learning sendiri terdiri atas 2 bagian yaitu Machine Learning Supervised dan Machine Learning Unsupervised.

Machine Learning Supervised adalah struktur dari suatu data yang hendak dianalisis telah ditentukan dahulu dan Machine Learning mencari data di struktur tersebut, sedangkan Machine Learning Unsupervised struktur dari suatu data dicari oleh Machine Learning itu sendiri [2]. Salah satu algoritma Supervised Learning yang sering digunakan pada proses klasifikasi adalah algoritma Random Forest (RF). RF adalah teknik bagging yang memiliki karakteristik signifikan yang berjalan efisien pada dataset besar. Random forest

dapat menangani ribuan variabel masukan tanpa penghapusan variabel dan memperkirakan fitur penting untuk klasifikasi [1].

Di dalam dunia medis, teknologi - teknologi banyak sekali diterapkan untuk memenuhi kebutuhan medis itu sendiri seperti AI pendeteksi pasien positif Covid atau tidak dengan memanfaatkan hembusan nafas dari pasien tersebut. Dan masih banyak lagi hal - hal yang bisa kita manfaatkan untuk mengembangkan teknologi di bidang kesehatan salah satunya adalah dengan memanfaatkan sinyal EKG atau Elektrokardiogram untuk mengklasifikasi penyakit gagal jantung pada pasien. EKG merupakan sebuah informasi sinyal yang digambarkan dalam bentuk diagram yang menampilkan informasi penting mengenai keadaan jantung manusia. Electrocardiography atau EKG adalah rekaman aktivitas listrik yang dihasilkan melalui siklus detak jantung [2]. Pada kegiatan internship yang penulis lakukan akan berfokus pada Ekstraksi dan klasifikasi sinyal MIT-BIH Arrhythmia menggunakan model Random Forest.

1.1. *MIT-BIH Arrhythmia Database*

MIT-BIH Arrhythmia database adalah rangkaian uji standar yang umumnya tersedia untuk mengevaluasi aritmia deteksi. Sejak 1980, *database* ini telah digunakan untuk dasar penelitian untuk dinamika jantung di sekitar 500 lokasi di seluruh dunia. Basis data ini sebagian besar digunakan untuk tujuan medis dan penelitian dari deteksi dan analisis aritmia jantung yang berbeda. Basis data ini mencoba menyediakan informasi yang tepat untuk mendeteksi aritmia ventrikel [3].

Aritmia adalah perubahan detak jantung yang tidak normal karena detak jantung yang tidak tepat yang menyebabkan kegagalan dalam

pemompaan darah. Aritmia dapat menyebabkan kematian jantung mendadak. Gejala aritmia yang umum adalah denyut prematur, jantung berdebar, pusing, kelelahan, dan pingsan. Aritmia lebih sering terjadi pada orang yang menderita tekanan darah tinggi, diabetes dan arteri koroner penyakit [4]. Sinyal *Electrocardiogram* yang akan digunakan pada kegiatan internship ini diambil dari *MIT-BIH Arrhythmia database*.

1.2. Elektrokardiogram (EKG)

Elektrokardiogram (EKG) adalah tes medis yang mengukur aktivitas listrik jantung. EKG digunakan untuk mendiagnosis dan memantau berbagai kondisi jantung, seperti serangan jantung, aritmia, dan gagal jantung. EKG akan merekam aktivitas listrik kecil yang dihasilkan oleh jantung selama periode waktu tertentu dengan menempatkan elektroda pada tubuh pasien [5]. Rekaman EKG berisi *noise* dan amplitudo yang bervariasi dari setiap orang sehingga sulit dalam proses mendiagnosis [6].

Elektrokardiogram (EKG) memberikan informasi penting tentang berbagai kondisi manusia [7]. Untuk melakukan EKG, petugas kesehatan akan menempelkan tambalan kecil dan lengket yang disebut elektroda ke dada, lengan, dan kaki pasien. Elektroda terhubung ke mesin EKG, yang merekam sinyal listrik yang dihasilkan oleh jantung saat bergerak ke seluruh tubuh. Mesin tersebut menghasilkan jejak aktivitas listrik jantung, yang disebut strip EKG, yang kemudian diinterpretasikan oleh petugas kesehatan.

Machine Learning dapat diterapkan pada analisis data EKG untuk meningkatkan akurasi dan efisiensi diagnosis dan pengobatan. Misalnya, algoritma pembelajaran mesin dapat dilatih untuk mengenali pola dalam data

EKG yang menunjukkan kondisi jantung tertentu. Algoritma ini kemudian dapat digunakan untuk menganalisis data EKG secara otomatis dan memberikan rekomendasi diagnosis atau pengobatan. Algoritma pembelajaran mesin dapat dilatih untuk mengklasifikasikan data EKG ke dalam kategori yang berbeda, seperti normal atau abnormal, atau untuk mengidentifikasi kondisi jantung tertentu.

Biasanya, klasifikasi sinyal EKG memiliki empat fase: preprocessing, segmentasi, ekstraksi fitur, dan klasifikasi. Fase preprocessing terutama ditujukan untuk mendeteksi dan melemahkan frekuensi sinyal EKG yang terkait dengan artefak, yang juga biasanya melakukan normalisasi dan peningkatan sinyal. Setelah preprocessing, segmentasi akan membagi sinyal menjadi segmen yang lebih kecil, yang dapat mengekspresikan aktivitas listrik jantung dengan lebih baik [6].

1.3. Machine Learning

Pembelajaran Mesin atau Machine Learning merupakan kemajuan teknologi yang penting karena dapat membantu dalam mengambil keputusan dengan mekanisme prediksi dan klasifikasi berdasarkan data yang ada [8]. Berfokus pada performance yang tinggi, teknik pembelajaran mesin atau machine learning diterapkan pada bisnis dengan data yang berkembang pesat. Karena pendekatan desain cocok untuk komunikasi komputasi paralel dan terdistribusi yang berevolusi atau data bisnis yang dinamis dan berkembang kedalam model Machine Learning [9].

Teknologi berbasis komputer modern banyak yang telah menggunakan Pembelajaran Mesin atau Machine Learning. Machine Learning merupakan

cabang dari Kecerdasan Buatan atau Artificial Intelligence yang luas dan sudah berkembang pesat saat ini yang memungkinkan komputer untuk belajar dan berkembang secara otomatis tanpa harus diprogram secara eksplisit. Teknologi ini berasal dari mempelajari pengenalan pola dan teori pembelajaran komputasi. Secara umum, metode pembelajaran yang umum digunakan oleh Machine Learning dapat diklasifikasikan menjadi Supervised, Unsupervised, dan Reinforcement Learning [10].

1.3.1. Supervised Learning

Supervised Learning merupakan metode Machine Learning untuk menyimpulkan fungsi dari data *train* ada. Algoritma Supervised Learning biasanya berisi kumpulan sampel input (*feature*) dan label yang berkaitan dengan kumpulan data tersebut. Tujuan dari pengklasifikasian adalah untuk menemukan batas yang sesuai yang dapat memprediksi label yang benar pada data *test*. Secara singkat, dalam Supervised Learning memiliki setiap contoh data yang berpasangan yang terdiri dari objek masukan (*input*) dan objek keluaran (*output*) yang diinginkan. Algoritma Supervised Learning menganalisis data *train* dan menghasilkan fungsi (*model*) [11]. Beberapa contoh metode algoritma pada Supervised Learning:

1.3.1.1. Random Forest

Random Forest adalah metode Machine Learning yang diperkenalkan pada tahun 2001 oleh Leo Breiman. Metode ini menggunakan serangkaian besar dari Decision Tree dengan korelasi

timbal balik yang rendah dan fitur yang dipilih secara acak menggunakan metode bagging (Bootstrap AGGregatING) [12].

Random Forest merupakan salah satu metode pengklasifikasian terbaik dan banyak digunakan untuk regresi dan juga klasifikasi. Random Forest memiliki algoritma yang sederhana sehingga menjadi salah satu pilihan yang menarik untuk mengklasifikasi teks. Selain itu, Random Forest juga memiliki kemampuan untuk mengolah data berdimensi tinggi dan memiliki performa yang tinggi walaupun menggunakan data yang banyak sehingga menjadi salah satu keuntungan menggunakan model ini dibandingkan dengan model Machine Learning lainnya [13].

Pemilihan model ini didasarkan karena pada faktanya bahwa Random Forest secara luas dianggap sebagai salah satu metode Machine Learning yang paling sukses dan banyak digunakan hingga saat ini [14].

1.3.1.2. Naive Bayes

Naive Bayes adalah salah satu pengklasifikasi terkemuka yang telah banyak dikutip oleh banyak peneliti dan digunakan di banyak aspek karena kesederhanaannya dan kinerja dari klasifikasi yang nyata [15]. Diantara bermacam-macam teknik atau metode klasifikasi saat ini, pengklasifikasi Naive Bayes (*NB*) berperan penting karena kesederhanaan, traktabilitas dan efisiensinya [16].

Naive Bayes juga merupakan pengklasifikasi yang sangat kompeten dalam banyak aplikasi di dunia nyata. Meskipun Naive

Bayes telah menunjukan akurasi klasifikasi yang luar biasa, namun output yang dihasilkan jarang benar dalam kenyataan [17]. Terlepas dari hal itu, pada kenyataanya Naive Bayes bekerja dengan baik diimplementasikan di dunia ini seperti memprediksi waktu, pemfilteran spam, prakiraan cuaca, dan diagnosis medis [18].

Pengklasifikasian Naive Bayes didasarkan pada kombinasi Teorema Bayes dan asumsi independensi atribut. Pengklasifikasi Naive Bayes didasarkan pada asumsi yang disederhanakan bahwa nilai atribut bersifat independen secara kondisional, berdasarkan asumsi nilai target yang diberikan. Pendekatan Bayes untuk klasifikasi kasus baru terdiri dari penetapan nilai target yang paling mungkin, dengan asumsi bahwa ada [19].

1.3.1.3. K-Nearest Neighbor

K-Nearest Neighbor (KNN) merupakan salah satu algoritma klasifikasi yang paling stabil dalam kelompok algoritma klasifikasi supervised. Dikarenakan kesederhanaan dan implementasi algoritma yang mudah.

K-Nearest Neighbor (KNN) merupakan salah satu metode klasifikasi nonparametric. KNN menjadi terkenal karena algoritmanya yang luas dan yang paling mudah. KNN dapat menyimpan semua masalah atau studi kasus yang ada dan mengklasifikasikan berdasarkan kesamaannya. Secara umum, KNN menggunakan jarak Euclidean untuk menemukan data yang paling mirip dengan kelompoknya [20].

Pada metode ini, nilai yang hilang dari variabel tertentu diganti dengan nilai rata-rata atau nilai mean dari KNN terdekat dari pengamatan variabel yang sama. Fungsi jarak yang berbeda dapat digunakan untuk memilih tetangga yang memungkinkan metode untuk menyertakan variabel numerik dan kategori. Keuntungan utama KNN adalah tidak memerlukan spesifikasi model prediktif apapun [21].

1.3.1.4. Support Vector Machine (SVM)

Support Vector Machine membuktikan bahwa salah satu algoritma yang memiliki performance yang powerful selama beberapa dekade terakhir dan mengandalkan prinsip SRM. Support Vector Learning (SVM) umumnya digunakan untuk masalah klasifikasi dan regresi.

Support Vector Machine (SVM) bekerja dengan membangun hyperplane yang memisahkan sampel berdasarkan pendekatan margin yang maksimum. Berbeda dengan Artificial Neural Network (ANN) yang memiliki kelemahan local minimal. Support Vector Machine memberikan solusi dengan menyelesaikan masalah optimasi dengan konveks [22].

Metode Support Vector Machine juga disebut model klasifikasi biner. Dalam ruang dua dimensi, garis lurus menjadikan garis segmentasi yang paling cocok di tengah 2 kelas data, dan untuk kumpulan data berdimensi tinggi, ini untuk menetapkan bidang keputusan yang optimal sebagai tolak ukur klasifikasi. Prinsip dasar

Support Vector Machine (SVM) mensyaratkan bahwa ketika masalah klasifikasi diselesaikan, jarak dari titik sampel terdekat ke permukaan keputusan adalah yang terbesar, yaitu jarak minimum memaksimalkan dua kelas titik sampel untuk memisahkan tepi [23].

1.3.2. Unsupervised Learning

Unsupervised Learning hanya dapat digunakan untuk tugas pengelompokan (clustering). Banyak pendekatan menggunakan Unsupervised Learning untuk mendukung tugas klasifikasi. Misalnya, algoritma pengelompokan (clustering) dapat meningkatkan kinerja tugas klasifikasi dengan mengelompokan objek data ke dalam kelompok yang lebih homogen.

Unsupervised Learning banyak digunakan untuk preprocessing data seperti ekstraksi fitur, pemilihan fitur, dan resampling. Namun, ada banyak juga kasus penggunaan pembelajaran tanpa pengawasan sebagai algoritma pilihan untuk klasifikasi dengan kinerja yang sebanding dan mungkin lebih baik dibandingkan Supervised Learning [24].

Pada algoritma Unsupervised Learning yang mampu memisahkan data tanpa sebuah pengetahuan yang dalam tentang berbagai jenis peristiwa meningkatkan efisiensi analisis secara luar biasa, dan memungkinkan analisis hilir untuk berkonsentrasi pada upaya penyesuaian hanya pada peristiwa yang menarik. Selain itu, algoritma pengelompokan memungkinkan lebih banyak eksplorasi data, berpotensi memungkinkan jenis reaksi baru dan tak terduga [25].

1.3.2.1. K-Means

Di era big data, sejumlah besar sumber daya data dikumpulkan dari kehidupan orang sehari-hari, ditransfer ke dalam internet, dan disimpan pada pusat data [26]. Pengelompokan data (Clustering), sebagai bagian penting dari data mining, dan sudah dianggap sebagai tugas penting dalam Unsupervised Learning.

Untuk kumpulan data tertentu, clustering akan membaginya menjadi beberapa kelompok atau cluster yang sedemikian rupa sehingga objek data dalam kelompok atau cluster yang sama berupa satu sama lain [27]. K-Means adalah pengelompokan masalah yang dipelajari dengan baik yang menghasilkan aplikasi di banyak bidang dan merupakan bagian dari Unsupervised Learning. K-Means merupakan salah satu masalah paling mendasar dalam ilmu komputer [28].

BAB 2
PENGENALAN TOOLS

2.1. Web Browser

Web browser adalah software yang digunakan untuk mengakses dan menampilkan konten baik berupa teks informasi, gambar, video dan konten lainnya dari internet. Beberapa contoh web browser populer saat ini termasuk Google Chrome, Mozilla Firefox, Microsoft Edge, Safari, dan Opera. Setiap web browser memiliki kelebihan dan kekurangan masing-masing, dan pengguna dapat memilih browser yang sesuai dengan kebutuhan dan preferensi yang diinginkan.

2.1. Google Colaboratory

Google Colaboratory atau disingkat Colab, adalah platform berbasis cloud yang memanfaatkan Notebook Jupyter untuk memfasilitasi penelitian dan pendidikan machine learning. Colab dilengkapi dengan runtime yang secara khusus dikonfigurasi untuk deep learning task. Keuntungan menggunakan Colab adalah menawarkan akses gratis ke sumber daya komputasi yang andal, seperti GPU dan TPU, yang berguna untuk melatih model machine learning yang besar. Selain itu, Colab memungkinkan pengguna berkolaborasi dengan mudah dalam proyek dengan berbagi notebook dan memberikan akses kepada orang lain [29]. Berikut adalah langkah-langkah umum menggunakan google colaboratory :

1. Buka web browser anda dan masuk ke halaman utama Google Colaboratory di https://colab.research.google.com/. Anda akan memerlukan akun Google untuk menggunakan Colab.
2. Setelah Anda masuk ke akun Google Anda, Anda akan diarahkan ke halaman utama Colab. Di sini, Anda dapat membuat notebook baru atau membuka notebook yang sudah ada.
3. Untuk membuat notebook baru, klik pada "File" di sudut kiri atas layar kemudian pilih "New Notebook". Anda dapat memilih bahasa pemrograman dan lingkungan runtime untuk notebook Anda.
4. Anda dapat mulai menulis kode di sel-sel notebook Anda. Colab mendukung berbagai bahasa pemrograman, termasuk Python, R, dan Julia, antara lain.
5. Colab juga menyediakan GPU gratis untuk pengguna agar dapat melatih model machine learning dengan lebih cepat. Untuk mengaktifkan GPU, buka menu "Runtime" dan pilih "Change runtime type". Anda dapat memilih "GPU" sebagai akselerator perangkat keras.
6. Ketika Anda selesai menggunakan notebook, Anda dapat menyimpannya ke Google Drive Anda atau mengunduhnya sebagai notebook IPython atau file PDF.

2.3. Jupyter Notebook

Jupyter notebook adalah dokumen elektronik yang dirancang untuk mendukung pengolahan data interaktif, analisis, dan visualisasi dalam format

yang mudah dibagikan. Jupyter notebook memanfaatkan cloud computing untuk memulai pemrograman dengan persyaratan yang sedikit [30]. Jupyter notebook mendukung berbagai bahasa pemrograman, antara lain Python, R, dan Julia. Salah satu fitur utama Notebook Jupyter adalah "cells", yang merupakan blok kode individual yang dapat dijalankan secara terpisah atau bersama-sama dalam urutan tertentu. Hal ini memungkinkan pengguna memecah tugas kompleks menjadi lebih mudah dikelola dan mengujinya secara iteratif. Secara keseluruhan, Jupyter Notebook adalah alat yang ampuh untuk ilmu data dan komputasi ilmiah, dan banyak digunakan dalam dunia akademis, penelitian, dan industri.

Berikut ini adalah langkah-langkah umum untuk menginstal Jupyter Notebook:

1. Pastikan bahwa Python sudah terinstal di komputer Anda. Anda dapat memeriksa ini dengan membuka terminal atau command prompt dan mengetikkan perintah python --version.
2. Buka terminal atau command prompt dan ketikkan perintah pip install jupyter untuk menginstal Jupyter Notebook menggunakan pip.
3. Tunggu proses instalasi selesai. Ini dapat memakan waktu beberapa menit tergantung pada kecepatan internet dan spesifikasi komputer Anda.
4. Setelah instalasi selesai, ketikkan perintah jupyter notebook di terminal atau command prompt untuk membuka Jupyter Notebook di browser Anda.

5. Anda sekarang dapat mulai menggunakan Jupyter Notebook untuk menulis kode Python dan menjalankan proses data science dan machine learning.

2.4. Visual Studio Code

Visual Studio Code merupakan sebuah aplikasi editor kode sumber terbuka yang dibuat oleh Microsoft untuk Windows, Linux, dan MacOS. Aplikasi ini memudahkan penulisan kode untuk beberapa jenis pemrograman, seperti C++, C#, Java, Python, PHP, dan GO. Visual Studio Code juga dapat mengenali jenis bahasa pemrograman yang digunakan dan memberikan warna yang berbeda untuk setiap fungsi dalam kode tersebut. Selain itu, aplikasi ini telah terintegrasi dengan Github. Fitur lainnya yang dimiliki oleh Visual Studio Code adalah kemampuan untuk menambahkan ekstensi sehingga pengembang dapat menambahkan fitur yang tidak tersedia di aplikasi tersebut [31].

Berikut adalah langkah-langkah umum untuk menginstal Visual Studio Code:

1. Kunjungi situs web resmi Visual Studio Code di https://code.visualstudio.com/ dan unduh instalasi yang sesuai dengan sistem operasi yang Anda gunakan (Windows, macOS, atau Linux).
2. Buka file instalasi yang sudah Anda unduh dan ikuti petunjuk untuk menginstal Visual Studio Code di komputer Anda. Di Windows, Anda dapat memilih opsi "Add to PATH" saat instalasi untuk memudahkan akses ke Visual Studio Code melalui terminal atau command prompt.

3. Setelah instalasi selesai, jalankan Visual Studio Code dengan mengklik ikon aplikasi atau membuka program dari menu Start di Windows.
4. Saat pertama kali membuka Visual Studio Code, Anda akan melihat layar sambutan yang menawarkan pilihan konfigurasi. Anda dapat memilih konfigurasi default atau menyesuaikan pilihan Anda sendiri.
5. Anda sekarang dapat mulai menggunakan Visual Studio Code untuk menulis kode dengan berbagai bahasa pemrograman. Visual Studio Code memiliki dukungan yang baik untuk bahasa pemrograman populer seperti Python, Java, JavaScript, dan banyak lagi.

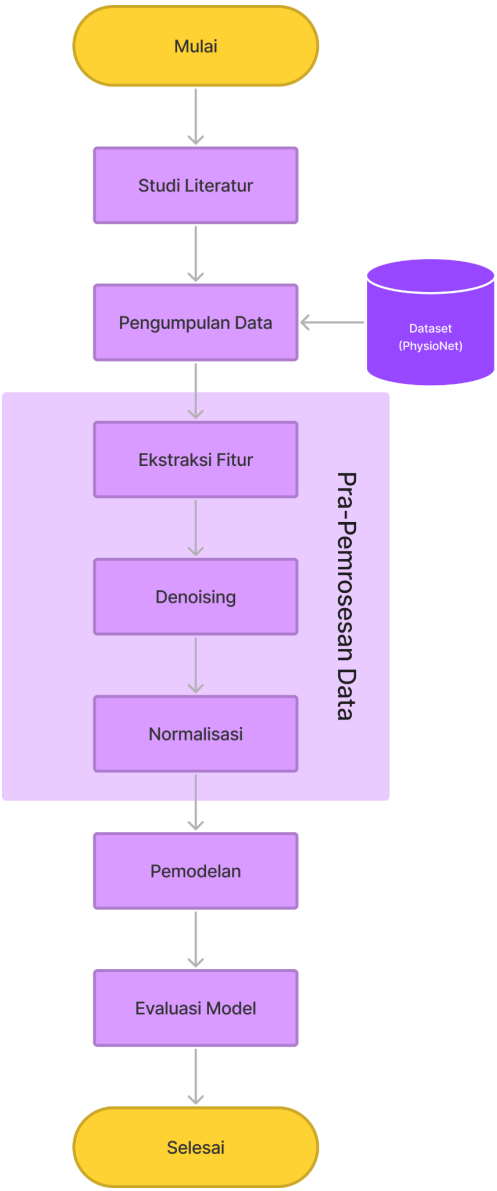
BAB 3
ALUR PEMBUATAN MODEL

3.1. Batasan Alur Pembuatan

Pada proses pembuatan model ini akan mencangkup beberapa batasan yang akan dibahas yaitu tentang ekstraksi dan komparasi hasil klasifikasi MIT-BIH Arrhythmia Database dengan menggunakan beberapa model Machine Learning. Sumber data yang akan digunakan berasal dari situs PhysioNet yang merupakan Database Complex Physiologic Signals. Data yang akan diambil pada penelitian ini adalah MIT-BIH Arrhythmia Database dengan 48 record dan masing-masing durasi yang tersedia. Dari data-data tersebut akan digabungkan lalu dilakukan ekstraksi dan klasifikasi sinyal dengan Random Forest.

3.2. Alur Pembuatan

Pada proses pembuatan ini model ini akan menjadi hal penting dalam melakukan ekstraksi hingga klasifikasi sinyal karena akan mempengaruhi hasil klasifikasi atau output dari model tersebut. Oleh karena itu, alur pembuatan yang tepat untuk mendapatkan hasil yang terbaik. Alur pembuatan akan ditunjukan pada gambar 3.1.



Gambar 3.1 Flow Diagram Pembuatan Model

3.3. Indikator Capaian

Berdasarkan gambar diatas, terdapat beberapa indikator capaian yaitu sebagai berikut:

Tabel 3.1 Indikator Capaian

No	Tahapan	Indikator Capaian
1	Studi Literatur	Uraian Teori - teori
2	Pengumpulan Data	Dataset sinyal MIT-BIH Arrhythmia
3	Pra-Pemrosesan Data	Ekstraksi Fitur, Denoising, Normalisasi
4	Pemodelan	Model Random Forest untuk Klasifikasi Sinyal
5	Evaluasi Model	Melakukan perbandingan dengan beberapa model

3.3.1. Studi Literatur

Pada studi literatur ini berisikan tentang uraian teori-teori bahan penelitian orang lain yang didapatkan pada jurnal - jurnal nasional maupun internasional. Pada bagian ini akan dijadikan acuan dari kegiatan penelitian ini untuk mengimplementasikan dan juga mengembangkan sesuai dengan teori-teori yang dijelaskan.

3.3.2. Pengumpulan Data

Pada tahap ini, data yang akan digunakan adalah data yang diambil langsung dari website PhysioNet yang merupakan *Research Resource for Complex Physiological Signals* untuk melakukan penelitian dan pendidikan biomedis dan menawarkan akses gratis pada database

yang disediakan. PhysioNet didirikan pada tahun 1999 dibawah naungan *National Institutes of Health* (*NIH*).

Data yang dikumpulkan merupakan data hasil rekaman detak jantung dari beberapa orang. Data yang digunakan berbentuk CSV dengan nilai hasil dari indikator grafik yang digambarkan. Pada data ini juga diberikan anotasi di setiap detak jantungnya untuk dijadikan label dari dataset tersebut.

3.3.3. Pra-Pemrosesan Data

Pra-Pemrosesan data merupakan proses pembersihan data dengan melakukan ekstraksi fitur dengan menggabungkan data menjadi satu data frame dengan masing-masing notasinya yang nantinya data tersebut akan dilakukan *denoising* agar data yang digunakan menjadi optimal dan tidak terdapat data-data yang *outlier* atau jauh dari nilai aslinya. Setelah itu, data yang sudah dilakukan denoising akan di normalisasi agar skala yang dipakai memiliki nilai yang sama.

3.3.4. Pemodelan

Pada tahap ini akan dilakukan pemodelan dengan menggunakan data yang sebelumnya. Data yang tersedia perlu dilakukan split untuk keperluan data *train* dan data *test* untuk nanti dilakukan proses pemodelan menggunakan data *train* tersebut.

3.3.5. Evaluasi Model

Tahap evaluasi model merupakan tahapan untuk mengukur kinerja dari model yang dihasilkan apakah sudah memiliki akurasi yang

baik atau tidak sehingga pada proses ini kita bisa melakukan perbandingan kinerja antara model yang tersedia dan model mana yang memiliki akurasi yang baik sesuai dengan hasil pra-pemrosesan data tersebut.

BAB 4
PROSES PEMBUATAN MODEL

4.1. Data

Data yang akan digunakan diambil langsung dari website PhysioNet yang merupakan *Research Resource for Complex Physiological Signals* untuk melakukan penelitian dan pendidikan biomedis. Data yang dikumpulkan merupakan data hasil rekaman detak jantung dari beberapa orang. Sejak tahun 1975, laboratorium Beth Israel Deaconess Medical Center dan MIT menyediakan wadah untuk menganalisis aritmia dan beberapa penelitian terkait. Dan MIT-BIH merupakan bagian utama dari penelitian tersebut yang diselesaikan serta didistribusikan pada tahun 1980. Sehingga dataset ini merupakan perangkat uji standar pertama yang tersedia secara umum untuk evaluasi detektor aritmia.

Bentuk data yang digunakan merupakan data CSV dengan nilai yang membentuk grafik dari sebuah detak jantung atau *beat*. Data tersebut memiliki *annotation* nya masing-masing disetiap *beat* yang bisa dijadikan label untuk proses klasifikasi. Berikut record data yang akan digunakan:

Tabel 4.1 Record Dataset

Record	Channel	Gender	Age	Medications
100	MLII, V5	Male	69	Aldomet, Inderal
101	MLII, V1	Female	75	Diapres

Record	Channel	Gender	Age	Medications
102	V5, V2	Female	84	Digoxin
103	MLII, V2	Male	-	Diapres, Xyloprim
104	V5, V2	Female	66	Digoxin, Pronestyl
105	MLII, V1	Female	73	Digoxin, Nitropaste, Pronestyl
106	MLII, V1	Female	24	Inderal
107	MLII, V1	Male	63	Digoxin
108	MLII, V1	Female	87	Digoxin, Quinaglute
109	MLII, V1	Male	64	Quinidine
111	MLII, V1	Female	47	Digoxin, Lasix
112	MLII, V1	Male	54	Digoxin, Pronestyl
113	MLII, V1	Female	24	-
114	V5, MLII	Female	72	Digoxin
115	MLII, V1	Female	39	-
116	MLII, V1	Male	68	-
117	MLII, V2	Male	69	-
118	MLII, V1	Male	69	Digoxin, Norpace
119	MLII, V1	Female	51	Pronestyl
121	MLII, V1	Female	83	Digoxin, Isordil, Nitropaste
122	MLII, V1	Male	51	Digoxin, Lasix, Pronestyl
123	MLII, V5	Female	63	Digoxin, Inderal
124	MLII, V4	Male	77	Digoxin, Isordil, Quinidine
200	MLII, V1	Male	64	Digoxin, Quinidine
201	MLII, V1	Male	68	Digoxin, Hydrochlorothiazide, Inderal, KCl
202	MLII, V1	Male	68	Digoxin, Hydrochlorothiazide, Inderal, KCl

Record	Channel	Gender	Age	Medications
203	MLII, V1	Male	43	Coumadin, Digoxin, Heparin, Hygroton, Lasix
205	MLII, V1	Male	59	Digoxin, Quinaglute
207	MLII, V1	Female	89	Digoxin, Quinaglute
208	MLII, V1	Female	23	-
209	MLII, V1	Male	62	Aldomet, Hydrodiuril, Inderal
210	MLII, V1	Male	89	-
212	MLII, V1	Female	32	-
213	MLII, V1	Male	61	Digoxin
214	MLII, V1	Male	53	Digoxin, Dilantin
215	MLII, V1	Male	81	-
217	MLII, V1	Male	65	Digoxin, Lasix, Quinidine
219	MLII, V1	Male	-	Digoxin
220	MLII, V1	Female	87	Digoxin
221	MLII, V1	Male	83	Hydrochlorthiazide, Lasix
222	MLII, V1	Female	84	Digoxin, Quinidine
223	MLII, V1	Male	73	-
228	MLII, V1	Female	80	Digoxin, Norpace
230	MLII, V1	Male	32	Dilantin
231	MLII, V1	Female	72	-
232	MLII, V1	Female	76	Aldomet, Inderal
233	MLII, V1	Male	57	Dilantin
234	MLII, V1	Female	56	-

Dari tabel 4.1 menunjukan record dataset yang dihimpun dari website resmi PhysioNet akan digunakan dengan masing-masing kolom seperti *Record, Channel, Gender, Age,* dan *Medications*. Kolom *Channel* merupakan kolom yang nilainya akan digunakan sebagai proses klasifikasi detak jantung yang nantinya akan digabungkan pada annotations nya masing-masing tiap *beat*-nya. Berikut contoh dataset pada sebuah record:

Tabel 4.2 Contoh Record Dataset MLII dan V1

'sample #'	'MLII'	'V1'
0	955	992
1	955	992
2	955	992
3	955	992
4	955	992
5	955	992
6	955	992
7	955	992
8	958	994
9	960	995
10	960	990
…	…	…
649999	1024	1024

Pada tabel 4.2 memiliki 3 kolom yang masing-masing nilai yang berbeda-beda tiap barisnya. Pada kolom MLII memiliki nilai yang akan dijadikan

nilai plotting pada sebuah detak jantung dengan range plotting 360 baris sehingga 360 baris pada kolom MLII merupakan 1 kali plotting detak jantung atau *beat*.

Di Setiap kolom tidak memiliki nilai yang kosong atau NULL, sehingga bisa langsung digunakan tanpa harus mengisi lagi nilai-nilai yang kosong. Setelah data-data disiapkan, proses selanjutnya adalah dengan melakukan Pra-Pemrosesan data.

4.2. Pra-Pemrosesan Data

Pada tahap ini, kita akan mempersiapkan data dengan mengolahnya agar data tersebut dapat digunakan untuk proses akhir atau pemodelan. Beberapa langkah Pra-Pemrosesan data yang akan dilakukan seperti Split Extension File, Record Signal Dataset, Denoising, Normalisasi, Merge Annotation, Rebalancing Dataset, Split Dataset.

4.2.1. Split Extension File

Dataset yang digunakan merupakan kumpulan beberapa record dan annotation, untuk itu perlu dilakukan split untuk untuk memisahkan file record dan annotation masing-masing record.

```
def split_file(filenames):
 records = list()
 annotations = list()

 for f in filenames:
    filename, file_extension = os.path.splitext(f)
```

```
    if(file_extension == '.csv'):
        records.append(path + filename + file_extension)
    else:
        annotations.append(path + filename + file_extension)

return records, annotations
```

Pada fungsi diatas terdapat kondisional yang memisahkan jenis extension file “.csv” dan memasukkan ke dalam sebuah array. Terdapat 2 extension yang di pisah yaitu “.csv” untuk dataset attribute dan “.txt” untuk label.

4.2.2. Record Signal Dataset

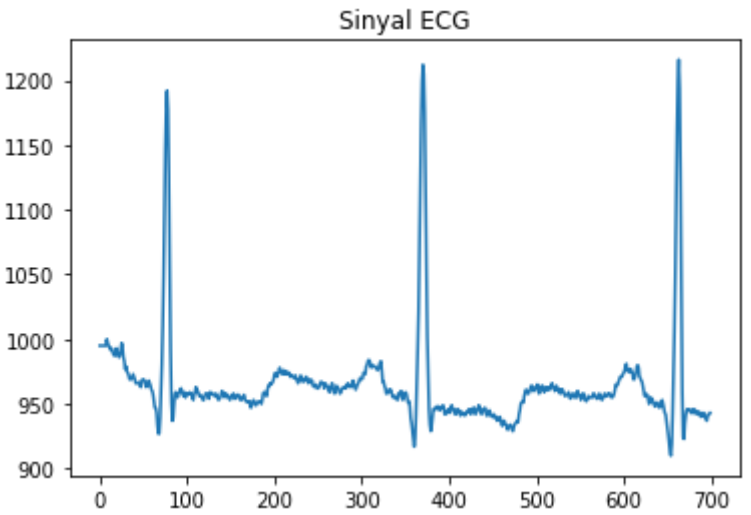
Tahap ini dilakukan untuk proses menyiapkan bentuk nilai dari record dataset untuk dilakukan plotting apakah hasil plotting sudah sesuai dengan bentuk sinyal EKG atau tidak.

```
def get_record(record):
 signals = []

 with open(record, 'rt') as csvfile:
    spamreader = csv.reader(csvfile, delimiter = ',', quotechar = '|')
    row_index = -1
    for row in spamreader:
        if(row_index >= 0):
            signals.insert(row_index, int(row[1]))
        row_index += 1

 return signals
```

Fungsi diatas akan membaca file CSV pada parameter dengan mengambil nilai MLII pada CSV untuk dijadikan attribute pada proses plot sinyal maupun proses klasifikasi. Nilai tersebut akan disimpan pada sebuah array yang nantinya akan diplot menggunakan array tersebut. Untuk melakukan plot sinyal menggunakan library matplotlib dari python.

```
plt.title("Sinyal ECG")
plt.plot(signals[0:700])
plt.show()
```

Dari potongan kode diatas akan menghasilkan plot dari contoh sinyal EKG dari dataset yang disiapkan sebelumnya. Berikut contoh gambar hasil plot sinyal EKG:



Gambar 4.1 Contoh Sinyal EKG

Pada gambar diatas terlihat bahwa hasil plotting membentuk sinyal EKG atau Detak Jantung. Terdapat 3 *beat* yang dihasilkan dari hasil plotting dengan mengambil nilai mulai dari baris ke 1 sampai baris ke 700. Tetapi pada sinyal EKG tersebut masih memiliki *noise* yang merupakan sinyal gangguan. Untuk itu, perlu dilakukan Denoising pada dataset untuk menghilangkan *noise* tersebut.

4.2.3. Denoising

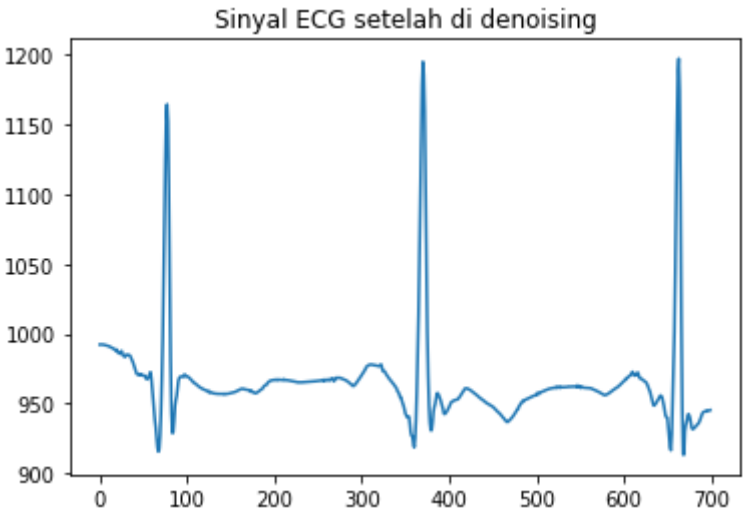
Denoising merupakan tahapan yang sangat penting dilakukan untuk menghilangkan *noise* pada sinyal. Jika *noise* tersebut tidak dihilangkan akan mengganggu dan menurunkan performa dari hasil modeling nanti karena banyak bentuk sinyal yang hanya mengganggu dari hasil sinyal yang sebenarnya.

```
def denoise(df):
  w = pywt.Wavelet('sym4')
  maxlev = pywt.dwt_max_level(len(df), w.dec_len)
  threshold = 0.04

  coeffs = pywt.wavedec(df, 'sym4', level=maxlev)
  for i in range(1, len(coeffs)):
    coeffs[i] = pywt.threshold(coeffs[i], threshold*max(coeffs[i]))

  datarec = pywt.waverec(coeffs, 'sym4')

  return datarec
```

Untuk melakukan Denoising menggunakan library “pywt” dengan mengkalkulasikan Wavelet untuk mengelompokan nilai menjadi sebuah Wavelet tunggal. Untuk nilai *Threshold* yang digunakan adalah “0.04”. Setelah sinyal dilakukan Denoising, maka akan membentuk sinyal yang sempurna dan tidak terdapat banyak *noise.*

```
signals = denoise(signals)
plt.title("Sinyal ECG setelah di denoising")
plt.plot(signals[0:700])
plt.show()
```

Potongan kode diatas akan melakukan Denoising menggunakan fungsi yang telah dibuat sebelumnya yaitu fungsi denoise(). Setelah dilakukan Denoising maka akan melakukan plot sinyal sebanyak 700 baris nilai pada dataset.



Gambar 4.2 Contoh Sinyal EKG setelah Denoising

Terlihat pada gambar diatas bahwa sinyal yang di plotting sudah bersih dari *noise* dan memiliki bentuk sinyal yang lebih sempurna daripada sinyal yang sebelumnya belum dilakukan Denoising. Karena sinyal EKG memiliki nilai yang berbeda-beda tiap recordnya, maka perlu dilakukan *Normalisasi* pada dataset tersebut untuk menyamakan skalanya.

4.2.4. Normalisasi

Normalisasi merupakan tahapan yang perlu dilakukan untuk mengubah nilai dengan tipe data Numerik pada himpunan data agar skala dari himpunan data tersebut sama. Akan Tetapi, tidak semua dataset bisa dilakukan normalisasi tergantung dari algoritma atau model yang akan dipakai untuk proses klasifikasi.

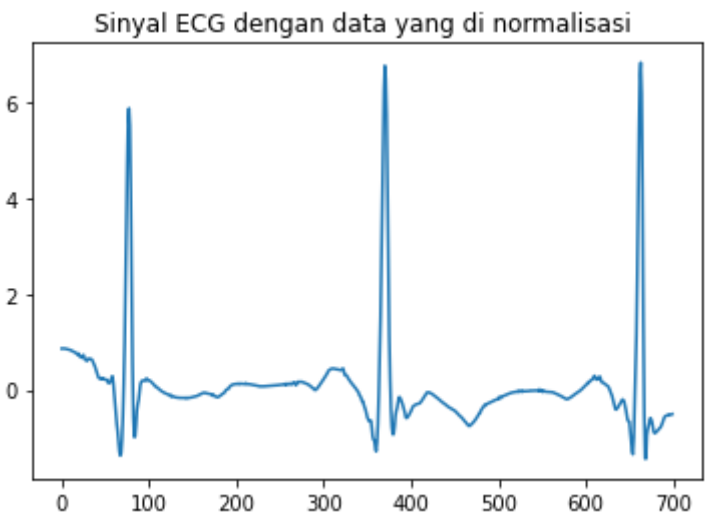
```
signals = stats.zscore(signals)
```

Fungsi diatas akan melakukan normalisasi nilai dari data atau sinyal yang telah dilakukan Denoising sebelumnya. Setelah dilakukan normalisasi maka data sinyal tersebut bisa langsung digunakan untuk klasifikasi. Untuk proses normalisasi ini akan mengubah nilai dari MLII pada data sebelumnya menjadi nilai yang sudah dinormalisasikan tentunya.

```
plt.title("Sinyal ECG dengan data yang di normalisasi ")
plt.plot(signals[0:700])
```

```
plt.show()
```

Dari fungsi diatas menggunakan data atau nilai sinyal yang sudah dinormalisasikan dari data yang sudah di denoising sebelumnya. Karena normalisasi hanya mengubah nilai skalanya menjadi kecil, maka harusnya hasil plotingan yang sudah di denoising dengan yang dinormalisasikan tidak akan berubah.



Gambar 4.3 Contoh Sinyal EKG setelah Normalisasi

Terlihat pada nilai Y dari hasil plotingan diatas berubah yang awalnya bernilai 900an menjadi nilai dengan range -1 sampai 7 tetapi tetap membentuk sinyal atau detak jantung yang sama seperti sebelumnya. Sebelum ke proses selanjutnya harus dilakukan penggabungan annotation pada setiap detak jantung atau *beat*.

4.2.5. Merge Annotation

Pada tahapan ini, menggabungkan annotation atau label pada data yang akan kita lakukan klasifikasi sangat penting, karena annotation inilah yang akan menentukan bahwa model yang dibuat berjalan dengan baik atau tidak nantinya. Pada proses ini juga akan dilakukan plotting dari masing-masing *class* atau label pada dataset tersebut.

```
example_beat_N_printed = False
example_beat_L_printed = False
example_beat_R_printed = False
example_beat_A_printed = False
example_beat_V_printed = False
```

Diatas merupakan variabel yang akan digunakan untuk pengkondisian untuk melakukan plotting pada masing-masing *class* atau label. Karena untuk mendapatkan nilai yang sesuai dengan *class* yang diinginkan, perlu pengkondisian pada suatu *looping*.

```
for r in range(0, len(records)):
  signals = []

  with open(records[r], 'rt') as csvfile:
     spamreader = csv.reader(csvfile, delimiter = ',', quotechar = '|')
     row_index = -1
     for row in spamreader:
        if(row_index >= 0):
           signals.insert(row_index, int(row[1]))
        row_index += 1
```

Pada kodingan diatas akan melakukan *looping* sesuai dengan banyaknya file records yang telah di split sebelumnya lalu akan membuka isi file CSV tersebut dan membaca nilainya. Nilai tersebut akan dimasukan kedalam array untuk nantinya digabungkan dengan record yang lainnya.

```
with open(annotations[r], 'r') as fileID:
  data = fileID.readlines()
  beat = list()

  for d in range(1, len(data)):
    splitted = data[d].split(' ')
    splitted = filter(None, splitted)
    next(splitted)

    pos = int(next(splitted))

    arrhythmia_type = next(splitted)
```

Setelah membuka dan membaca nilai dari CSV, maka dilanjutkan dengan membuka file annotation untuk mengambil sampel dari masing-masing beat dan juga mengambil *class* atau label dari data CSV sebelumnya. Proses selanjutnya ada dengan mengecek apakah *class* atau label tersebut sudah sesuai dengan yang diperlukan untuk proses klasifikasi.

```
    if(arrhythmia_type in classes):
      arrhythmia_index = classes.index(arrhythmia_type)
      count_classes[arrhythmia_index] += 1
```

```
if(window_size <= pos and pos < (len(signals) - window_size)):
  beat = signals[pos - window_size : pos + window_size]

  X.append(beat)
  y.append(arrhythmia_index)
```

Kodingan diatas menunjukan bahwa apabila *class* atau label yang diambil dari file annotation ada pada variabel *classes*, maka *beat* tersebut bisa di pakai untuk dataset untuk proses pengklasifikasian nanti. Setelah proses pengkondisian, maka beat tersebut dimasukan kedalam array untuk dijadikan dataframe nantinya. Pada variabel X merupakan kolom beat sedangkan untuk variabel y untuk *class* atau label dari beat tersebut.

Selanjutnya adalah dengan melakukan plot pada masing-masing *class* berdasarkan nilai pada variabel *classes* yang terdiri dari 5 *class* seperti berikut:

```
classes = ['N', 'L', 'R', 'A', 'V']
```

Keterangan:

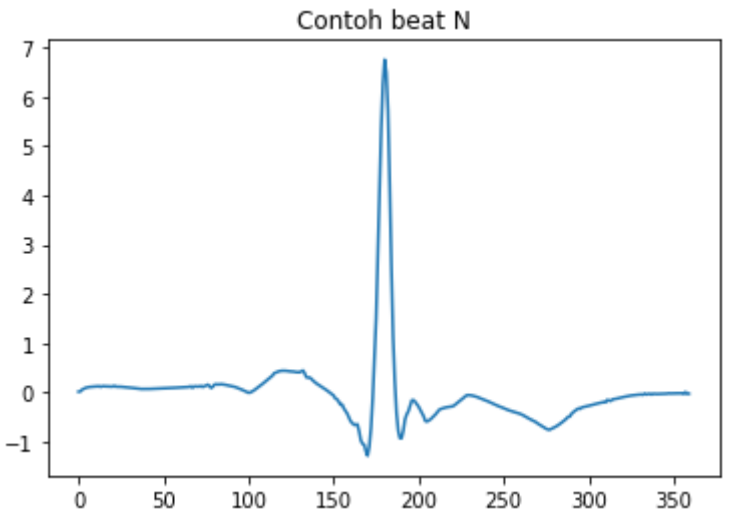
- N: Normal beat (displayed as "·" by the PhysioBank ATM, LightWAVE, pschart, and psfd)
- L: Left bundle branch block beat
- R: Right bundle branch block beat
- A: Atrial premature beat

- V: Premature ventricular contraction

Dari keterangan diatas merupakan arti dari masing-masing label yang akan dipakai untuk proses klasifikasi. N merupakan beat normal manusia dan diikuti dengan A dan V untuk beat yang prematur.

```
if arrhythmia_type == 'N' and not example_beat_N_printed:
  print("Contoh Beat")
  example_beat_N_printed = True
  plt.title("Contoh beat " + arrhythmia_type)
  plt.plot(beat)
  plt.show()
```

Pada kodingan diatas apabila *class* bernilai ‘N’ dan *class* tersebut belum dilakukan plot, maka kodingan tersebut akan melakukan plot detak jantung atau *beat* dengan label ‘N’.

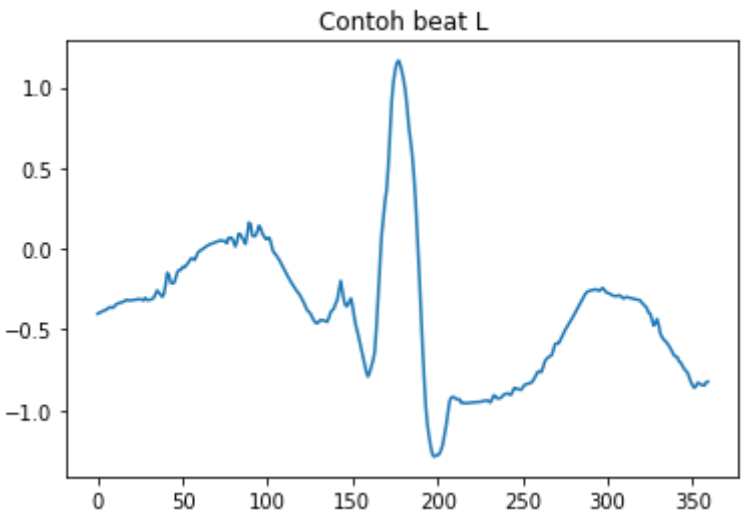


Gambar 4.4 Contoh Sinyal EKG dengan *class* N

Hasil plot diatas merupakan merupakan detak jantung Normalnya manusia yang digambarkan dengan satu gelombang tengah.

```
if arrhythmia_type == 'L' and not example_beat_L_printed:
  print("Contoh Beat")
  example_beat_L_printed = True
  plt.title("Contoh beat " + arrhythmia_type)
  plt.plot(beat)
  plt.show()
```

Kodingan diatas akan menampilkan plotting *beat* dengan *class* L apabila variabel *arrhythmia_type* bernilai 'L'.

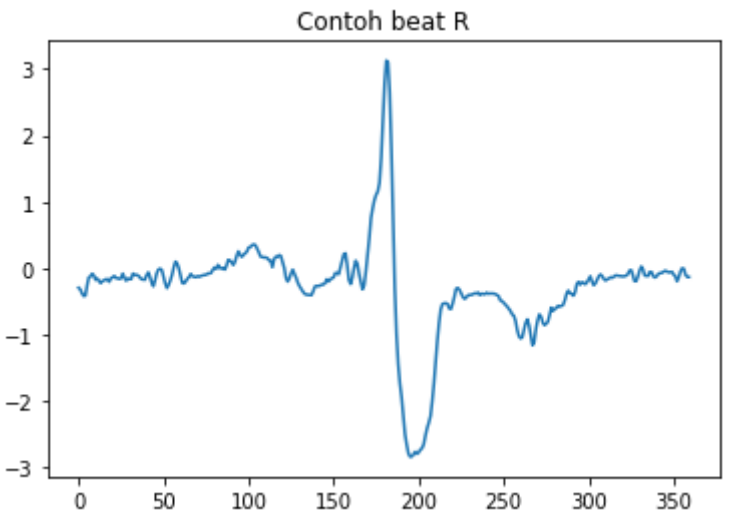


Gambar 4.5 Contoh Sinyal EKG dengan *class* L

Terlihat bahwa bentuk plot dari detak jantung dengan *class* L sangat berbeda dengan bentuk *beat* dengan *class* N.

```
if arrhythmia_type == 'R' and not example_beat_R_printed:
    print("Contoh Beat")
    example_beat_R_printed = True
    plt.title("Contoh beat " + arrhythmia_type)
    plt.plot(beat)
    plt.show()
```

Kodingan diatas akan menampilkan plotting *beat* dengan *class* R apabila variabel *arrhythmia_type* bernilai 'R'.



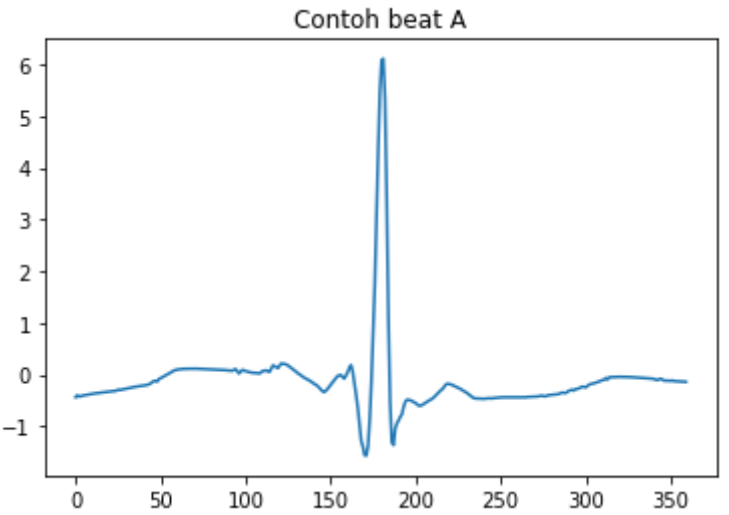
Gambar 4.6 Contoh Sinyal EKG dengan *class* R

Terlihat bahwa bentuk plot dari detak jantung dengan *class* L sangat tidar beraturan dan masih memiliki banyak *noise*.

```
if arrhythmia_type == 'A' and not example_beat_A_printed:
    print("Contoh Beat")
```

```
        example_beat_A_printed = True
        plt.title("Contoh beat " + arrhythmia_type)
        plt.plot(beat)
        plt.show()
```

Kodingan diatas akan menampilkan plotting *beat* dengan *class* A apabila variabel *arrhythmia_type* bernilai ‘A’.



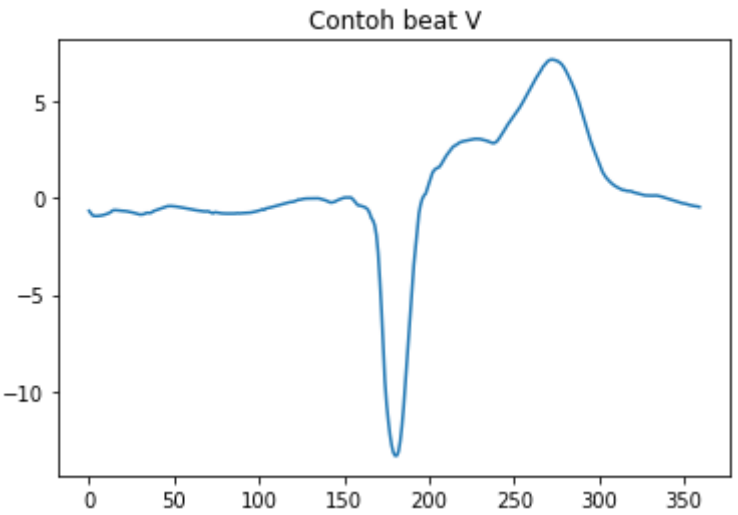
Gambar 4.7 Contoh Sinyal EKG dengan *class* A

Terdapat sedikit perbedaan bentuk plot dari detak jantung dari *class* N dan *class* A. Perbedaan terletak pada sisi kiri gelombang dengan adanya lekukan kebawah.

```
    if arrhythmia_type == 'V' and not example_beat_V_printed:
        print("Contoh Beat")
        example_beat_V_printed = True
```

```
plt.title("Contoh beat " + arrhythmia_type)
plt.plot(beat)
plt.show()
```

Kodingan diatas akan menampilkan plotting *beat* dengan *class* V apabila variabel *arrhythmia_type* bernilai 'V'.



Gambar 4.8 Contoh Sinyal ECG dengan *class* V

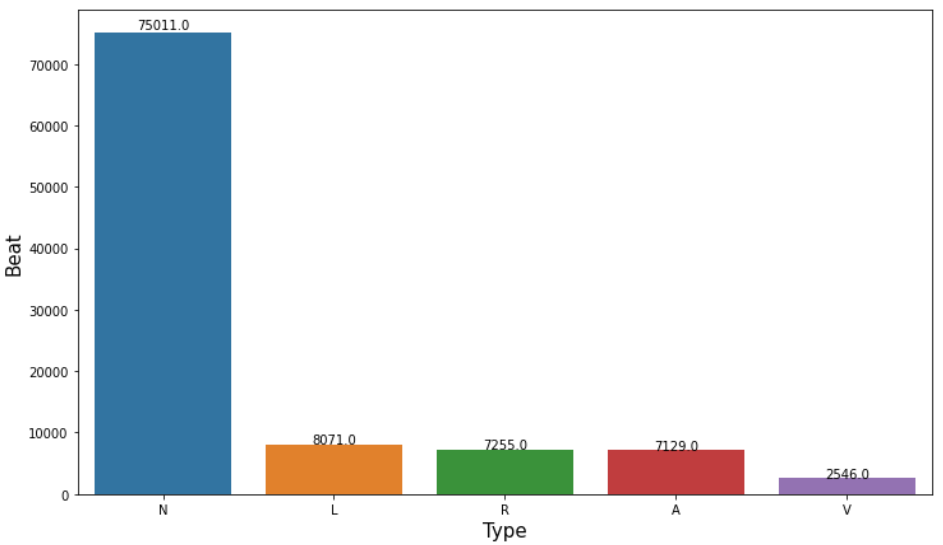
Perbedaan yang sangat menonjol adalah berbentuk gelombang terbalik dari bentuk beat dengan *class* N.

Kemudian, langkah selanjutnya adalah melakukan plotting diagram semua beat pada dataset dengan mengelompokan berdasarkan jumlah dari *class* yang akan digunakan.

```
per_class = X_train_df[X_train_df.shape[1] - 1].value_counts()
```

```
plt.figure(figsize=(12,7))
ax = sns.barplot(x=['N', 'L', 'R', 'A', 'V'], y=per_class)
plt.xlabel("Type", fontsize=15)
plt.ylabel("Beat", fontsize=15)
for p in ax.patches:
  width = p.get_width()
  height = p.get_height()
  x = p.get_x()
  y = p.get_y()
  ax.annotate(f"{height}", (x + width/2, y+ height*1.01), ha="center")
plt.show()
```

Kemudian, langkah selanjutnya adalah melakukan plotting diagram semua beat pada sesuai dengan *class* nya masing-masing.



Gambar 4.9 Diagram jumlah *beat* masing-masing *class*

Pada diagram diatas terlihat bahwa *beat* dengan *class* N mencapai 75.011 *beat* sedangkan *class* lainnya memiliki jumlah *beat* kurang dari 10.000 *beat*. Oleh karena itu, data tersebut perlu dilakukan penyetaraan data sesuai dengan *class* nya atau Rebalancing Dataset.

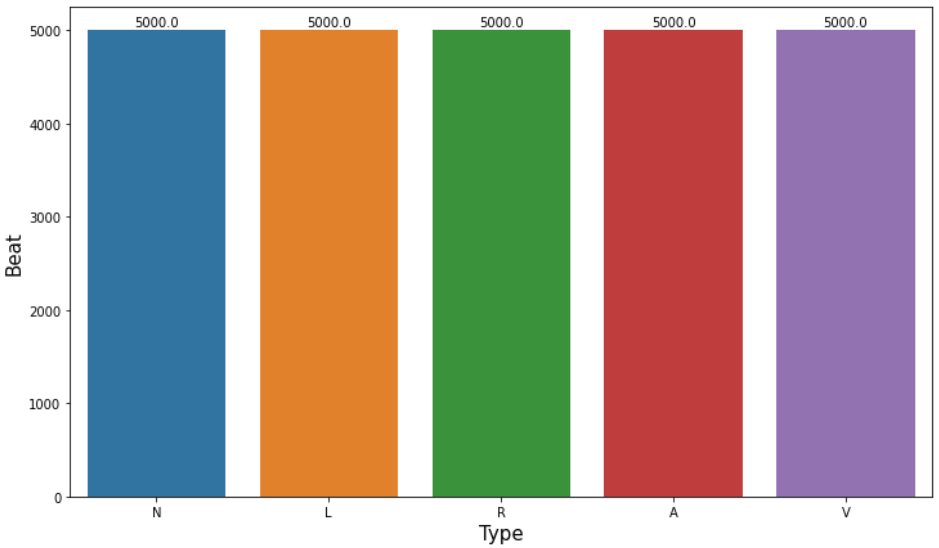
4.2.6. Rebalancing Dataset

Tahap Rebalancing Dataset merupakan proses penyeimbangan data sesuai dengan *class* masing-masing. Rasio data yang tidak seimbang sesuai dengan *class* perlu dilakukan rebalancing data agar data yang dilakukan *train* nanti menjadi proporsional.

```
def rebalancing_dataframe(X_train_df):
 df_0 = (X_train_df[X_train_df[X_train_df.shape[1]-1] == 0])
 .sample(n = 5000, random_state = 42)

 df_1 = X_train_df[X_train_df[X_train_df.shape[1]-1] == 1]
 df_2 = X_train_df[X_train_df[X_train_df.shape[1]-1] == 2]
 df_3 = X_train_df[X_train_df[X_train_df.shape[1]-1] == 3]
 df_4 = X_train_df[X_train_df[X_train_df.shape[1]-1] == 4]

 df_1_upsample = resample(df_1, replace = True, n_samples = 5000,
 random_state = 122)
 df_2_upsample = resample(df_2, replace = True, n_samples = 5000,
 random_state = 123)
 df_3_upsample = resample(df_3, replace = True, n_samples = 5000,
 random_state = 124)
 df_4_upsample = resample(df_4, replace = True, n_samples = 5000,
 random_state = 125)

 X_train_df = pd.concat([df_0, df_1_upsample, df_2_upsample,
 df_3_upsample, df_4_upsample])
```

```
return X_train_df
```

Dari kodingan di atas, tahapan rebalancing ini mengambil 5.000 sampel atau *beat* masing-masing *class* nya. Sehingga model mempelajari data yang seimbang dengan masing-masing *class* tersebut. Setelah dilakukan rebalancing, maka bentuk diagram dari dataset tersebut akan seimbang. Untuk mengetahui apakah data sudah seimbang atau tidak, perlu dilakukan plotting untuk mengetahui bentuk diagramnya.



Gambar 4.10 Diagram dataset setelah di Rebalancing

Terlihat dari diagram tersebut sudah memiliki masing-masing *beat* sebanyak 5.000 *beat* dengan jumlah *class* yang sama. Sehingga dataset tersebut sudah bisa dilakukan split untuk dilakukan *train* dan *test* pada tahap pemodelan.

4.2.7. Split Dataset

Pada tahap ini akan membagi 2 dataset untuk dilakukan *train* dan *test* untuk dilakukan pemodelan.

```
X = X_train_df.loc[:, :359]
Y = X_train_df[360]
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.3,
random_state=1)

print("X_train : ", np.shape(x_train))
print("X_test  : ", np.shape(x_test))
```

Output:
X_train : (17500, 360)
X_test : (7500, 360)

Setelah dilakukan split dataset, maka akan ada data *train* dan data *test.* Data *train* berfungsi untuk model melakukan pembelajaran terhadap data yang telah disiapkan. Setelah model melakukan pembelajaran terhadap data tersebut, maka model tersebut akan melakukan proses pengecekan terhadap data *test* yang sudah mempunyai hasil atau outputnya untuk divalidasi apakah hasil klasifikasi atau output dari model tersebut tepat atau tidak.

4.3. Pemodelan

Tahap modeling merupakan tahap yang paling penting dari sebuah Machine Learning dikarenakan tahap modeling akan menghasilkan sebuah

model yang memiliki output yang dapat mengklasifikasi sebuah masalah atau objek serta memiliki akurasi dalam pembuatannya.

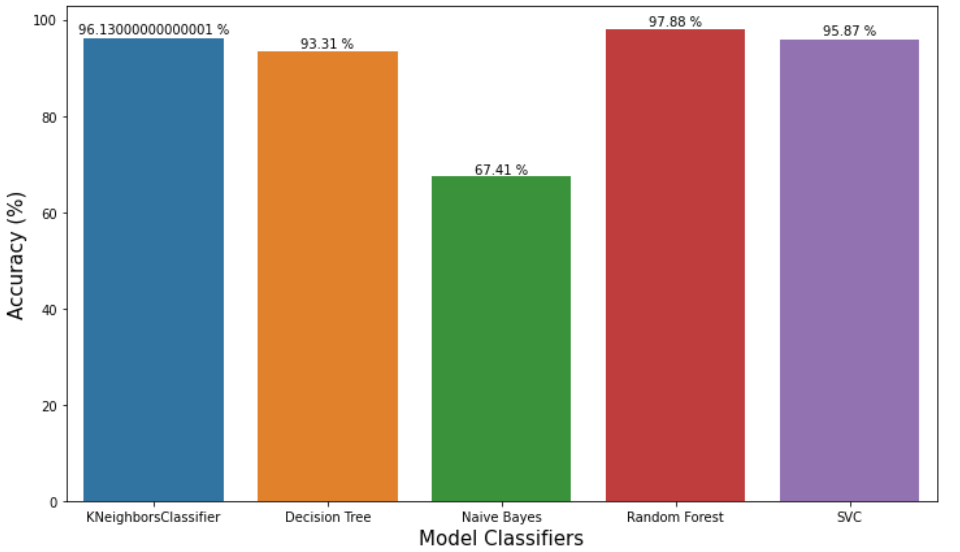
```
rf_model = RandomForestClassifier()
rf_model.fit(x_train, y_train)
rf_prediction = rf_model.predict(x_test)
rf_accuracy = (round(accuracy_score(rf_prediction, y_test), 4) * 100)
accuracy_list.append(rf_accuracy)
print("Accuracy (%) : ", rf_accuracy)
```

Output:
Accuracy (%) : 97.88%

Dari hasil modeling tersebut mendapatkan akurasi yang tinggi yaitu sebesar 97.88%. Akurasi tersebut sudah termasuk akurasi yang tinggi dikarenakan tidak termasuk dalam model yang *overfitting* maupun *underfitting*.

4.4. Evaluasi Model

Untuk memastikan model yang dibuat adalah model yang memiliki akurasi yang paling tinggi, maka perlu dilakukan evaluasi model. Pada evaluasi ini akan membandingkan hasil klasifikasi dari beberapa model seperti K-Neighbors Classifier, Decision Tree, Naive Bayes, dan Support Vector Machine.



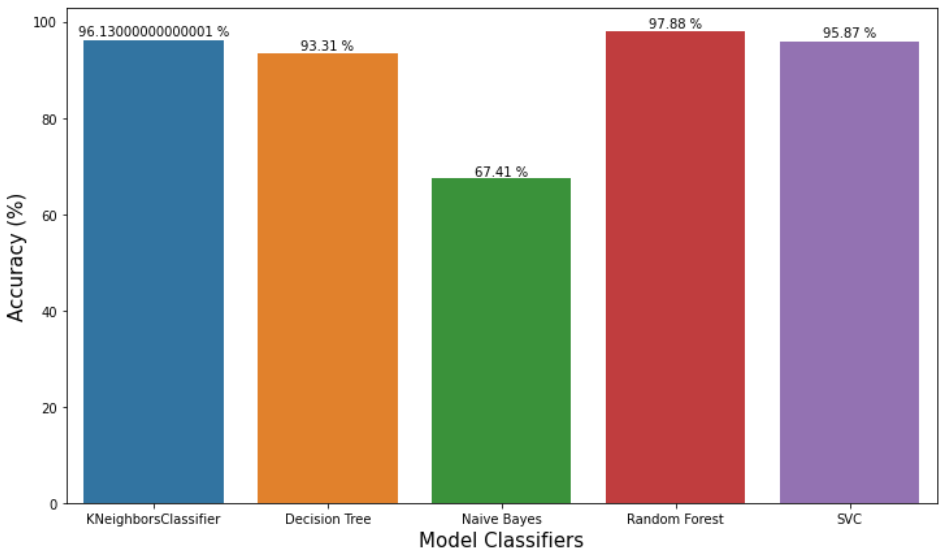
Gambar 4.11 Hasil Akurasi Model

Setelah dilakukan plot terlihat pada gambar diatas bahwa Random Forest memiliki akurasi yang paling tinggi dengan akurasi 97.88% diantara model yang ada dan diikuti dengan model Naive Bayes yang memiliki akurasi yang paling rendah dengan akurasi 67.41%.

BAB 5
HASIL KOMPARASI

5.1. Hasil Komparasi

Dari model yang telah dilakukan komparasi, hasil dari setiap model memiliki karakteristik dan algoritmanya masing-masing sehingga hal tersebut dapat mempengaruhi hasil akurasi dari masing-masing model tersebut. Berdasarkan hasil pemodelan yang dilakukan, didapatkan model dengan akurasi yang paling tinggi yaitu model Random Forest dengan nilai akurasi 97.88% dan di posisi paling rendah yaitu model Naive Bayes dengan nilai akurasi 67.41%.



Gambar 4.12 Perbandingan Akurasi Model

Pada proses komparasi di atas digambarkan melalui diagram dengan masing-masing model dan hasil akurasinya. Proses komparasi dilakukan dengan menggunakan data yang sama dengan pra-pemrosesan data yang sama juga sehingga dapat dilihat model mana yang cocok dengan hasil proses data tersebut.

Parameter yang digunakan pada masing-masing model juga menggunakan parameter *default* pada model yang dipakai karena pada umumnya, parameter yang di set *default* merupakan parameter yang umum dan banyak digunakan, tetapi tidak menutup kemungkinan menggunakan parameter lain akan mendapatkan akurasi yang lebih tinggi.

Berikut penjelasan dari karakteristik dari masing-masing model yang dilakukan komparasi:

5.1.1. KNeighborsClassifier

KNN identik dengan menghitung jarak masing-masing kelas yang akan dilakukan klasifikasi. Nilai jarak yang dihitung merupakan hasil kemiripan dari data lama (K) yang terdekat. Nilai K pada algoritma ini dijadikan parameter pada pemodelan untuk mencari nilai K terbaik. Sehingga algoritma ini akan langsung mengklasifikasikan berdasarkan data (K) yang terdekat dengan hasil nilai perhitungan tersebut.

```
knn_model = KNeighborsClassifier()
knn_model.fit(x_train, y_train)
knn_prediction = knn_model.predict(x_test)
knn_accuracy = (round(accuracy_score(knn_prediction, y_test), 4) * 100)
accuracy_list.append(knn_accuracy)
```

```
print("Accuracy (%) : ", knn_accuracy)
```

Output:
Accuracy (%) : 96.13%

Pada model KNN tersebut menggunakan parameter (K) dengan nilai default yaitu 5. Karena menggunakan nilai default, tidak perlu untuk menginisialisasikannya pada *function* model tersebut. Dari hasil akurasi model KNN didapatkan nilai akurasi sebesar 96.13%.

5.1.2. Decision Tree

Karakteristik dari Decision Tree adalah dengan memprediksi suatu masalah dengan membentuk suatu pohon keputusan dengan memecah kedalam himpunan yang lebih kecil dan secara bertahap akan terus dilakukan pengembangan dalam pengambilan keputusannya. Node keputusan dan node daun merupakan hasil akhir dari algoritma ini. Parameter *max_deph* sangat mempengaruhi hasil klasifikasi karena model tersebut akan mengambil keputusan berdasarkan nilai dari *max_deph* tersebut.

```
dt_model = DecisionTreeClassifier()
dt_model.fit(x_train, y_train)
dt_prediction = dt_model.predict(x_test)
dt_accuracy = (round(accuracy_score(dt_prediction, y_test), 4) * 100)
accuracy_list.append(dt_accuracy)
print("Accuracy (%) : ", dt_accuracy)
```

Output:

```
Accuracy (%) : 93.31%
```

Untuk model DecisionTree setelah dilakukan pemodelan menggunakan dataset yang telah diproses mendapatkan akurasi sebesar 93.31%. Lebih rendah dibandingkan dengan model KNN sebelumnya.

5.1.3. Naive Bayes

Pada model ini akan menghitung nilai probabilitas dari suatu data atau objek yang akan dilakukan klasifikasi dan sangat cocok untuk diterapkan pada data yang bernilai biner.

```
nb_model = GaussianNB()
nb_model.fit(x_train, y_train)
nb_prediction = nb_model.predict(x_test)
nb_accuracy = (round(accuracy_score(nb_prediction, y_test), 4) * 100)
accuracy_list.append(nb_accuracy)
print("Accuracy (%) : ", nb_accuracy)
```

```
Output:
Accuracy (%) : 67.41%
```

Model Naive Bayes akan bergantung pada data dan otomatis akan menyesuaikan parameter dengan data yang akan dilakukan modeling. Jika parameter *prior* ditentukan maka tidak akan menyesuaikan dengan data yang ada. Dari hasil modeling diatas mendapatkan nilai akurasi yang cukup rendah yaitu sebesar 67.41%.

5.1.4. Random Forest

Random Forest merupakan salah satu metode dari Decision Tree sehingga tidak heran kalau alur dari algoritma ini memiliki karakteristik yang hampir sama. Tetapi yang membedakannya adalah Random Forest itu sendiri merupakan kombinasi dari beberapa Tree atau pohon yang dijadikan sebuah model.

```
rf_model = RandomForestClassifier()
rf_model.fit(x_train, y_train)
rf_prediction = rf_model.predict(x_test)
rf_accuracy = (round(accuracy_score(rf_prediction, y_test), 4) * 100)
accuracy_list.append(rf_accuracy)
print("Accuracy (%) : ", rf_accuracy)
```

```
Output:
Accuracy (%) : 97.88%
```

Random Forest menjadi model yang mendapatkan nilai akurasi tertinggi pada proses klasifikasi ini. Model ini memiliki parameter yang hampir mirip dengan Decision Tree yang memiliki *max_depth* atau nilai cabang keputusan. Random Forest memiliki nilai akurasi sebesar 97.88%.

5.1.5. SVC

Model Support Vector Classification memiliki karakteristik dengan mencari hyperplane terbaik dengan cara mencari jarak diantara kedua kelas tersebut. Fungsi dari hyperplane itu sendiri adalah dengan membagi 2 kelas.

```
svc_model = SVC()
svc_model.fit(x_train, y_train)
svc_prediction = svc_model.predict(x_test)
svc_accuracy = (round(accuracy_score(svc_prediction, y_test), 4) * 100)
accuracy_list.append(svc_accuracy)
print("Accuracy (%) : ", svc_accuracy)
```

Output:
Accuracy (%) : 95.87%

Dari model SVC memiliki akurasi cukup tinggi yaitu sebesar 95.87% dengan menggunakan parameter *default* dari *function* model tersebut.

5.2. Kesimpulan dan manfaat

Dari beberapa model yang telah dilakukan komparasi, dapat diambil kesimpulan bahwa masing-masing model memiliki karakteristik dan parameter yang berbeda-beda sehingga memiliki akurasi yang berbeda di setiap modelnya.

Model yang sudah dilatih bisa digunakan untuk melakukan klasifikasi masing-masing beat dan menentukan beat tersebut merupakan beat yang prematur atau normal dan tentu akan membawa manfaat besar bagi pasien yang menderita penyakit jantung untuk dideteksi sejak dini melalui klasifikasi detak jantung atau beat dari pasien tersebut. Dan diharapkan pada pengembangan selanjutnya hasil dari klasifikasi masing-masing beat dapat diprediksi apakah pasien menderita penyakit jantung atau tidak.

DAFTAR PUSTAKA

[1] Wasimuddin, M., Elleithy, K., Abuzneid, A. S., Faezipour, M., & Abuzaghleh, O. (2020). Stages-based ECG signal analysis from traditional signal processing to machine learning approaches: A survey. *IEEE Access*, *8*, 177782-177803.

[2] Pojon, M. (2017). *Using machine learning to predict student performance* (Master's thesis).

[3] Kuila, S., Dhanda, N., & Joardar, S. (2020). Feature Extraction and Classification of MIT-BIH Arrhythmia Database. In Proceedings of the 2nd International Conference on Communication, Devices and Computing (pp. 417-427). Springer, Singapore.

[4] Apandi, Z. F. M., Ikeura, R., & Hayakawa, S. (2018, August). Arrhythmia detection using MIT-BIH dataset: A review. In *2018 International Conference on Computational Approach in Smart Systems Design and Applications (ICASSDA)* (pp. 1-5). IEEE.

[5] Li, T., & Zhou, M. (2016). ECG classification using wavelet packet entropy and random forests. *Entropy*, *18*(8), 285.

[6] Kumar, R. G., & Kumaraswamy, Y. S. (2012). Investigating cardiac arrhythmia in ECG using random forest classification. *International Journal of Computer Applications*, *37*(4), 31-34.

[7] Wasimuddin, M., Elleithy, K., Abuzneid, A. S., Faezipour, M., & Abuzaghleh, O. (2020). Stages-based ECG signal analysis from

traditional signal processing to machine learning approaches: A survey. *IEEE Access*, *8*, 177782-177803.

[8] Kamila, N. K., Frnda, J., Pani, S. K., Das, R., Islam, S. M., Bharti, P. K., & Muduli, K. (2022). Machine learning model design for high performance cloud computing & load balancing resiliency: An innovative approach. *Journal of King Saud University-Computer and Information Sciences*.

[9] Renggli, C., Ashkboos, S., Aghagolzadeh, M., Alistarh, D., & Hoefler, T. (2019, November). Sparcml: High-performance sparse communication for machine learning. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis* (pp. 1-15).

[10] Bhatt, H., Shah, V., Shah, K., Shah, R., & Shah, M. (2022). State-of-the-art machine learning techniques for melanoma skin cancer detection and classification: a comprehensive review. *Intelligent Medicine*.

[11] Khanzadeh, M., Chowdhury, S., Marufuzzaman, M., Tschopp, M. A., & Bian, L. (2018). Porosity prediction: Supervised-learning of thermal history for direct laser deposition. *Journal of manufacturing systems*, *47*, 69-82.

[12] Khanzadeh, M., Chowdhury, S., Marufuzzaman, M., Tschopp, M. A., & Bian, L. (2018). Porosity prediction: Supervised-learning of thermal history for direct laser deposition. *Journal of manufacturing systems*, *47*, 69-82.

[13] Jalal, N., Mehmood, A., Choi, G. S., & Ashraf, I. (2022). A novel improved random forest for text classification using feature ranking and optimal number of trees. *Journal of King Saud University-Computer and Information Sciences*.

[14] Makariou, D., Barrieu, P., & Chen, Y. (2021). A random forest based approach for predicting spreads in the primary catastrophe bond market. *Insurance: Mathematics and Economics*, *101*, 140-162.

[15] Alizadeh, S. H., Hediehloo, A., & Harzevili, N. S. (2021). Multi independent latent component extension of naive bayes classifier. *Knowledge-Based Systems*, *213*, 106646.

[16] Blanquero, R., Carrizosa, E., Ramírez-Cobo, P., & Sillero-Denamiel, M. R. (2021). Variable selection for Naïve Bayes classification. *Computers & Operations Research*, *135*, 105456.

[17] Chen, S., Webb, G. I., Liu, L., & Ma, X. (2020). A novel selective naïve Bayes algorithm. *Knowledge-Based Systems*, *192*, 105361.

[18] Shaban, W. M., Rabie, A. H., Saleh, A. I., & Abo-Elsoud, M. A. (2021). Accurate detection of COVID-19 patients based on distance biased Naïve Bayes (DBNB) classification strategy. *Pattern Recognition*, *119*, 108110.

[19] Andrejiova, M., & Grincova, A. (2018). Classification of impact damage on a rubber-textile conveyor belt using Naïve-Bayes methodology. *Wear*, *414*, 59-67.

[20] Zamri, N., Pairan, M. A., Azman, W. N. A. W., Abas, S. S., Abdullah, L., Naim, S., ... & Gao, M. (2022). River quality classification using different distances in k-nearest neighbors algorithm. *Procedia Computer Science*, *204*, 180-186.

[21] Cubillos, M., Wøhlk, S., & Wulff, J. N. (2022). A bi-objective k-nearest-neighbors-based imputation method for multilevel data. *Expert Systems with Applications*, 117298.

[22] Kumar, B., & Gupta, D. (2021). Universum based Lagrangian twin bounded support vector machine to classify EEG signals. *Computer Methods and Programs in Biomedicine*, *208*, 106244.

[23] Zhang, H., Shi, Y., Yang, X., & Zhou, R. (2021). A firefly algorithm modified support vector machine for the credit risk assessment of supply chain finance. *Research in International Business and Finance*, *58*, 101482.

[24] Jafari-Marandi, R. (2021). Supervised or unsupervised learning? Investigating the role of pattern recognition assumptions in the success of binary predictive prescriptions. *Neurocomputing*, *434*, 165-193.

[25] Solli, R., Bazin, D., Hjorth-Jensen, M., Kuchera, M. P., & Strauss, R. R. (2021). Unsupervised learning for identifying events in active target experiments. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, *1010*, 165461.

[26] Zhao, D., Hu, X., Xiong, S., Tian, J., Xiang, J., Zhou, J., & Li, H.

(2021). K-means clustering and kNN classification based on negative databases. *Applied Soft Computing*, *110*, 107732.

[27] Wang, X., Wang, Z., Sheng, M., Li, Q., & Sheng, W. (2021). An adaptive and opposite K-means operation based memetic algorithm for data clustering. *Neurocomputing*, *437*, 131-142.

[28] Zhang, Z., Feng, Q., Huang, J., Guo, Y., Xu, J., & Wang, J. (2021). A local search algorithm for k-means with outliers. *Neurocomputing*, *450*, 230-241.

[29] Carneiro, T., Da Nóbrega, R. V. M., Nepomuceno, T., Bian, G. B., De Albuquerque, V. H. C., & Reboucas Filho, P. P. (2018). Performance analysis of google colaboratory as a tool for accelerating deep learning applications. *IEEE Access*, *6*, 61677-61685.

[30] Vallejo, W., Díaz-Uribe, C., & Fajardo, C. (2022). Google colab and virtual simulations: practical e-learning tools to support the teaching of thermodynamics and to introduce coding to students. *ACS omega*, *7*(8), 7421-7429.

[31] Ramdhan, N. A., & Nufriana, D. A. (2019). Rancang Bangun Dan Implementasi Sistem Informasi Skripsi Online Berbasis WEB. *Jurnal Ilmiah Intech: Information Technology Journal of UMUS*, *1*(02), 1-12.

GLOSARIUM

A

Arrhythmia: Perubahan detak jantung yang tidak normal karena detak jantung yang tidak tepat yang menyebabkan kegagalan dalam pemompaan darah.

B

Big Data: Jumlah data yang sangat besar hingga melebihi pemrosesan dari kapasitas sistem database konvensional.

C

Class: Blueprint atau mockup untuk menghasilkan sebuah object.

D

Denoising: Menghilangkan noise pada suatu citra serta mempertahankan informasi penting didalamnya.

E

Electrocardiography: Rekaman aktivitas listrik yang dihasilkan melalui siklus detak jantung.

F

Feature: 8

G

Google Colab: Software untuk menulis kode program secara online yang sudah terintegrasi dengan drive.

H

***Hyperplane*:** Batas keputusan yang membedakan dua kelas klasifikasi.

***Hydrochlorothiazide*:** Obat penurun tekanan darah bagi penderita hipertensi.

I

Inderal: Obat pencegah dan meredakan nyeri dada bagi penderita jantung koroner.

K

Klasifikasi: Sistem pengelompokan atau golongan menurut kaidah dan ketentuan.

L

Lasix: Obat untuk mengeluarkan cairan berlebih pada penderita gagal jantung atau gagal ginjal.

M

Metode: *Flow* atau jalan yang akan ditempuh untuk mendapatkan tujuan tertentu.

Medical: Hal yang berkaitan dengan kedokteran dan kesehatan.

N

Non Parametrik: Pengolahan data yang tidak didasari dengan asumsi parametrik.

Neural Network: Cabang dari AI yang mengadopsi syaraf-syaraf otak manusia.

O

***Overfitting*:** Model machine learning yang memiliki prediksi dengan akurasi paling tepat pada data latihan tetapi tidak untuk data baru atau selain data latihan.

P

Plot: Alur rangkaian kejadian yang digambarkan.

Probabilitas: Nilai peluang akan terjadinya sesuatu.

Prematur: Memiliki bentuk yang tidak normal dari biasanya.

R

Rasio: Angka yang menunjukan hubungan secara matematis antara jumlah satu dengan yang lainnya.

Rebalancing: Menyeimbangkan jumlah data untuk dilakukan pelatihan oleh model.

S

Segmentasi: Membagi sinyal menjadi segmen yang lebih kecil, yang dapat mengekspresikan aktivitas listrik jantung dengan lebih baik.

T

Threshold: Metode segmentasi citra image.

Variable: Tempat untuk menampung data dalam memori dengan nilai yang dapat berubah - ubah.

W

***Wavelet*:** Fungsi matematis yang mengelompokan energi citra digital.

INDEX

TENTANG PENULIS

M. RIZKY, lahir di Kabupaten Dompu pada tanggal 17 April 2000. Pendidikan tingkat dasar hingga menengah atas ditempuh di Kabupaten Dompu. Dan melanjutkan pendidikan sarjananya dengan mengambil Diploma 4/D4 di Program Studi Teknik Informatika Politeknik Pos Indonesia (sekarang Universitas Logistik dan Bisnis Internasional). Dari kecil hingga SMA saya termasuk orang yang kurang memahami bagaimana dunia IT bekerja, hingga pada saat saya masuk di dunia perkuliahan saya mulai mempelajari berbagai macam teknologi IT dan mulai mendalaminya sampai sekarang. Terlebih lagi dunia IT tidak akan ada matinya dan bahkan akan terus berkembang. Saya sangat suka berinovasi untuk menciptakan produk - produk yang bermanfaat dan berguna bagi masyarakat luas, ditambah lagi pemanfaat teknologi sekarang ini yang meningkat untuk menangani suatu masalah. Saya juga selalu mengikuti perkembangan teknologi seiring dengan perkembangan zaman sekarang karena di setiap tahunnya akan ada teknologi - teknologi yang menarik dan wajib kita ikuti dan pelajari. Pada dunia bisnis, saya sangat tertarik dan ingin sekali membuat sebuah startup dibidang teknologi yang dimana visi dan misinya tentu mampu memberikan manfaat serta menyelesaikan masalah terhadap masyarakat sekitar.

Link Github: https://github.com/bukped/Komparasi-Klasifikasi-Sinyal-ECG

DI ERA MODERN INI, BANYAK SEKALI KEGIATAN – KEGIATAN OPERASIONAL MAUPUN KEGIATAN LAINNYA YANG MELIBATKAN ATAU MENGGUNAKAN ARTIFICIAL INTELLIGENCE (AI). TIDAK BISA DIPUNGKIRI LAGI BAHWA TEKNOLOGI YANG BERKEMBANG PESAT SEPERTI SEKARANG INI TENTU DIBUAT UNTUK MEMPERMUDAH PEKERJAAN MANUSIA BAHKAN MENGGANTIKAN PERAN MANUSIA. SALAH SATU BAGIAN DARI ARTIFICIAL INTELLIGENCE (AI) ADALAH MACHINE LEARNING. MACHINE LEARNING SEBAGAI METODE DALAM SISTEM KECERDASAN BUATAN YANG MAMPU MENGKLASIFIKASIKAN DATA YANG DIMASUKKAN UNTUK KEPERLUAN DAN KEBUTUHAN MASING - MASING. BANYAK APLIKASI YANG MENERAPKAN MACHINE LEARNING UNTUK KEPERLUAN KLASIFIKASI DATA, MEMPREDIKSI HUBUNGAN ANTAR DATA, MEMBACA POLA DATA, DAN BANYAK IMPLEMENTASI LAINNYA. DALAM LAPORAN INI AKAN DIBAHAS PENGGUNAAN KELUARAN STRUKTUR DARI MACHINE LEARNING DIGUNAKAN UNTUK MENDETEKSI SINYAL ECG APAKAH SUDAH SESUAI DENGAN DATA YANG SUDAH DI TETAPKAN PADA ANOTASI. BERDASARKAN HASIL PENELITIAN MENUNJUKKAN PEMODELAN STRUKTUR DARI MACHINE LEARNING DAPAT DIGUNAKAN UNTUK MENDETEKSI KESESUAIAN DARI SINYAL BAIK PEMODELAN SECARA TERPISAH ATAUPUN PEMODELAN SECARA GABUNGAN.